A POLYHEDRAL CONIC FUNCTIONS BASED CLASSIFICATION METHOD FOR NOISY DATA

MÜGE ACAR¹ AND REFAIL KASIMBEYLI¹*

¹Department of Industrial Engineering, Faculty of Engineering
Eskisehir Technical University Eskisehir
26555, Turkey

(Communicated by Gerhard-Wilhelm Weber)

Abstract. This paper presents a robust binary classification method, which is an extended version of the Modified Polyhedral Conic Functions (M-PCF) algorithm, earlier developed by Gasimov and Ozturk. The new version presented in this paper, has new features in comparison to the original algorithm. The mathematical model used in the new version, is relaxed by allowing some inaccuracies in an optimal way. By this way, it is aimed to reduce the overfitting and improve the generalization property. In the original version, the sublevel set of a separating function generated at every iteration, does not contain any element of the other set. This is changed in the new version, where the sublevel sets of separating functions generated by the new algorithm, are allowed to contain some elements from other set. On the other hand, the new algorithm uses a tolerance parameter which prevents generating "less productive separating functions". In the original version, the algorithm continues till all points of the “first” set are separated from the second one, where a separating function is generated if there still exist unseparated elements regardless the number of such elements. In the new version, the tolerance parameter is used to terminate iterations if there are only a few unseparated elements. By this way, it is aimed to improve the generalization property of the algorithm, and therefore the new version is called Parameterized Polyhedral Conic Functions (P-PCF) method. The performance and efficiency of the proposed algorithm is demonstrated on well-known datasets from the literature and on noisy data.

1. Introduction. Binary classification problems have been widely studied by researchers, and a number of algorithms have been developed over the years (see e.g., [1, 3, 6]). Gasimov and Ozturk introduced a special class of polyhedral conic functions (PCF), and developed a so-called PCF algorithm, based on these functions [11]. This algorithm uses the idea of separation of nonconvex sets by conic surfaces (see e.g. [15, 17]), which has various applications in nonconvex mathematical programming (see e.g. [4, 12, 14, 16, 18, 19, 26]).

PCF algorithm guarantees 100% accuracy of training set by its greedy approach, and provides good test accuracy for many test instances. Ozturk and Ciftci combined a clustering technique and the PCF algorithm to design a new classifier for solving multi-class classification problems [24]. Cimen and Ozturk presented O-PCF

---

2020 Mathematics Subject Classification. Primary: 62H30, 65K05 90C25, 90C90; Secondary: 68Q32.

Key words and phrases. Binary classification, polyhedral conic functions, noisy data, optimization.

* Corresponding author; The author published under the name Rafail N. Gasimov until 2007.
algorithm for one-class classification problems [8]. Ozturk, Bagirov, and Kasimbeyli developed an incremental piecewise linear classifier by using polyhedral conic functions [23]. Bagirov, et.al. developed a new algorithm using polyhedral conic sets to identify a data center and compute a piecewise linear boundaries [5].

One of the disadvantages of the PCF algorithm is that, its use may lead to a large gap between the training and the test accuracies for many datasets. It is shown in this paper that, in noisy environments if the level of noise increases, test accuracy obtained by the PCF algorithm, decreases rapidly.

In this paper, we introduce the P-PCF algorithm, a version of the Modified PCF algorithm, which has new features in comparison to the original PCF algorithm. The mathematical model used in the new version, is modified by artificially allowing some inaccuracies, by this way it is aiming to reduce the overfitting. In the original version, sublevel set of a separating function generated at every iteration for a “first” set, does not contain any element of the other set. Moreover, the algorithm continues till all points of a “first” set are separated from the second one, where a separating function is generated if there still exist unseparated elements regardless the number of such elements. Due to this property, the original PCF algorithm provides the 100% accuracy on the training set. This is changed in the new method, where the sublevel sets of separating functions generated, are allowed to contain some elements from other set. The other novelty is that, the tolerance parameter is used to terminate iterations, if a most “productive” separating function generates only a few elements at some iteration. By this way, generating the “less productive separating functions” is prevented, which improved the efficiency and the generalization property of the method. This is demonstrated on standard and noisy datasets from the literature.

Overfitting and noise are two different but related problems recognized in classification problems. Overfitting prevents a classification algorithm from making accurate and reliable predictions; noise is common in most real-world datasets and can deteriorate the performance of the algorithm in terms of classification accuracy, training time and generalization ability. The new method uses a tolerance parameter, which allows to control the level of noise in the training set. By this way it can be determined whether the separation is worthwhile at each iteration. This adaptive capability allows the algorithm to be applied to a dataset without knowing whether the data are noisy. This ability also provides the algorithm with robust behavior in noisy domains, thereby reducing overfitting. All these features are demonstrated on illustrative examples. It is also shown that, the number of iterations becomes less than the one performed by the Modified PCF algorithm.

Organization of the paper is as follows: Section 2 briefly describes preliminaries and motivation. In Section 3, the P-PCF algorithm is described in detail with illustrative examples. Section 4 presents computational results and comparisons. Section 5 draws some conclusions from the work.

2. Preliminaries and motivation. Parameterized Polyhedral Conic Functions (P-PCF) algorithm presented in this paper, is an extended version of the Modified Polyhedral Conic Functions (M-PCF) algorithm earlier developed by Gasimov and Ozturk [11]. In this section we recall some basic concepts of this algorithm that will be used throughout the paper.
**Definition 2.1.** ([11, Definition 2.2]) A function $g : \mathbb{R}^n \to \mathbb{R}$ is called polyhedral conic, if its graph is a cone and all its sublevel sets $S_\alpha = \{ x \in \mathbb{R}^n : g(x) \leq \alpha \}$ for $\alpha \in \mathbb{R}$, are polyhedral sets.

Let the function $g_{(w,\xi,\gamma,a)} : \mathbb{R}^n \to \mathbb{R}$ be defined as follows:

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

where $w, a \in \mathbb{R}^n$, $\xi, \gamma \in \mathbb{R}$, $w'x = w_1x_1 + \cdots + w_nx_n$ is scalar product of $w$ and $x$, where $w'$ denotes the transpose of $w$, and $\|x\|_1 = |x_1| + \cdots + |x_n|$ is the $l_1$ norm of the vector $x \in \mathbb{R}^n$.

The following lemma proved in [11] shows that function $g$ defined in (1), is a polyhedral conic function.

**Lemma 2.2.** ([11, Lemma 2.2]) The graph of function $g_{(w,\xi,\gamma,a)}$ defined in (1), is a polyhedral cone with vertex at $(a, -\gamma) \in \mathbb{R}^n \times \mathbb{R}$.

It follows from Lemma 2.2 that, every function of form (1), is a polyhedral conic function.

### 2.1. Modified PCF (M-PCF) algorithm.

Let $A$ and $B$ be two given sets in $\mathbb{R}^n$:

$$A = \{ a^i \in \mathbb{R}^n : i \in I \}, \quad I = \{ 1, \ldots, m \},$$

$$B = \{ b^j \in \mathbb{R}^n : j \in J \}, \quad J = \{ 1, \ldots, p \}$$

The algorithm presented below, at each iteration generates a function of the form (1) by calculating the parameters $w$, $\xi$, and $\gamma$ as solutions of a certain linear programming (LP) problem. The sublevel set of such a function, as it follows from Lemma 2.2, divides the whole space into two parts such that, all points of $B$ remain “outside”, and as many points of $A$ as possible, remain “inside” this sublevel set. By excluding these latter points from $A$, the algorithm passes to the next iteration and generates a new separating function for this updated set. The process continues until an empty set is obtained. The resulting separating function is defined as a pointwise minimum of all functions generated. It was proved that the algorithm terminates in a finite number of steps.

**Initialization Step:** Let $l = 1$, $I_l = I$, $e_m \in \mathbb{R}^m$ be the vector consisting of ones, $A_l = A$ and go to Step 1.

**Step 1:** Solve the problem $(P^l_k)$ given below, for every $k \in I_l$, and every $a^l_k \in A_l$.

$$\begin{align*}
(P^l_k) & \quad \min \left( \frac{w'e_m}{m} \right) \\
& \text{s.t.} \quad w'(a^i - a^l_k) + \xi \|a^i - a^l_k\|_1 - \gamma + 1 \leq y_i, \quad \forall i \in I_l \tag{3} \\
& \quad -w'(b^j - a^l_k) - \xi \|b^j - a^l_k\|_1 + \gamma + 1 \leq 0, \quad \forall j \in J \tag{4} \\
& \quad y = (y_1, \ldots, y_m) \in \mathbb{R}^m_+, w \in \mathbb{R}^n, \xi, \gamma \geq 1. \tag{5}
\end{align*}$$

**Step 2:** Let $(w^l_k, \xi^l_k, \gamma^l_k, y^l_k)$ be an optimal solution of problem $(P^l_k)$, corresponding to the element $a^l_k \in A_l$, and let $N^l_k$ be the number of elements of $A_l$ separated
from \( B \) by the sublevel set of the corresponding function \( g^l_k(x) = g(w^l_k, \xi^l_k, \gamma^l_k)(x) \).
Let \( g^l_{k_0} \) be the separating function of this iteration which corresponds to the index
\( k_0 \) with \( N^l_{k_0} = \max \{ N^l_k : k \in I_l \} \). Choose the separating function with maximum
number of separated elements of this iteration, and denote
\[
g_l(x) = g(w^l_{k_0}, \xi^l_{k_0}, \gamma^l_{k_0})(x) = w^l_{k_0}(x - a^l_{k_0}) + \xi^l_{k_0}\|x - a^l_{k_0}\|_1 - \gamma^l_{k_0}
\]  
(6)

**Step 3:** Let \( I_{l+1} = \{ i \in I_l : g_l(a^i) + 1 > 0 \}, \ A_{l+1} = \{ a^i \in A_l : i \in I_{l+1} \}, \ l = l + 1. \)
If \( A_l \neq \emptyset \) GO TO Step 1. Otherwise, GO TO Step 4.

**Step 4:** Define the function \( g(x) \), which separates the sets \( A \) and \( B \) as follows:
\[
g(x) = \min_l g_l(x).
\]  
(7)

The algorithm solves linear subproblems \((P^l_k)\) for all points \( a^l_k \) of \( A_l \) and computes
the parameters \((w^l_k, \xi^l_k, \gamma^l_k)\). Then, the separating function \( g_l \) is determined as a
function, which separates maximum number of elements of \( A \) from \( B \). By Lemma
2.2, graph of the function \( g_l \) which consists of points \((x, v) \in \mathbb{R}^n \times \mathbb{R} \) with \( v = g_l(x) \),
is a cone with vertex at \((a^l, -\gamma^l)\). A constraint \( \gamma \geq 1 \) stated in constraint set (5),
ensures that the vertex of this cone is placed “under” the hyperplane \( v = 0 \), that is
in the halfspace, \( \mathbb{R}^n \times (0, -\infty) \). The constraint set (3) ensures that the point \( a^l \)
and the points of \( A^l \) that are “close” to \( a^l \), are inside the polyhedral set corresponding
to the sublevel set \( \{ x : g_l(x) \leq -1 \} \). The closeness of these points of \( A^l \) to \( a^l \),
is guaranteed by the objective function (2). Thus, the sublevel set \( \{ x : g_l(x) \leq -1 \} \)
will include as many elements of \( A^l \) as the value of this objective function is close
to zero. On the other hand, the constraint set (4) ensures that all elements of \( B \)
will remain outside the sublevel set \( \{ x : g_l(x) \leq -1 \} \) at each iteration.

The following theorem, proved in [11] shows that the algorithm terminates in a
finite number of iterations and that the resulting function \( g \) defined by equation (7)
separates two arbitrary finite element sets \( A \) and \( B \) in \( \mathbb{R}^n \) with \( A \cap B = \emptyset \).

**Theorem 2.3.** ([11, Theorem 2.3]) The Modified PCF algorithm terminates in a
finite number of iterations, and the function \( g : \mathbb{R}^n \to \mathbb{R}^n \) defined by equation (7),
strictly separates the sets \( A \) and \( B \) in the sense that
\[
g(a) < 0, \forall a \in A,
\]  
(8)
\[
g(b) > 0, \forall b \in B.
\]  
(9)

Number of experiments and computations made since the publication of this
algorithm, demonstrate that the two properties of this algorithm, that is the constraint
set (4), and the condition that the algorithm continues till the last element of \( A \) is
separated from \( B \), makes the algorithm greedy, and deteriorates its generalization
property.

In this paper, we change the right-hand sides in constraint set (4), by allowing the
sublevel sets of functions \( g^l \) generated at every iteration, to include elements from
\( B \) in an optimal way. The optimality is provided by using new set of nonnegative
decision variables on right hand sides of these constraints (instead of zeros in (4)),
and minimizing their sum in the objective function together with the sum of right
hand sides of constraint sets (3).

The second novelty is the modified stopping criterion of the algorithm. The M-
PCF algorithm is terminated if all elements of \( A \) are separated from \( B \), and for
Table 1. Data illustrating different noise types

| No. | Attribute 1 | Attribute 2 | Class   | Status     |
|-----|-------------|-------------|---------|------------|
| 1   | 0.26        | small       | A       |            |
| 2   | 0.25        | small       | A       |            |
| 3   | 0.29        | small       | B       | class noise|
| 4   | 1.62        | large       | B       |            |
| 5   | 1.05        | large       | B       |            |
| 6   | 0.30        | large       | B       | attribute noise|

Every unseparated element of $A$, the algorithm generates new separation function. The new algorithm uses a tolerance parameter $t$ for checking the modified stopping criterion. At Step 2, where the algorithm generates separation functions and chooses the one with maximum number of separated elements, the new algorithm will check whether this maximum number is less than the tolerance $t$ or not. If this maximum number of separated elements, is less than $t$, the algorithm will stop. The reason is that, the new algorithm will not generate less productive separating functions.

Both widely used datasets from the literature and datasets with noise are utilized to demonstrate the performance of the proposed method. The next subsection gives some explanations on noisy datasets.

2.2. Classification with noisy data. The goal of a classification algorithm is to generate a separation technique using training instances in order to maximize test accuracy. This maximum accuracy is affected by two factors [28]:

- quality of the training data, and
- learning ability of the algorithm.

The problem of classification in noisy environments has attracted the attention of researchers investigating ways to search in noisy conditions and to deal with noise. Noise in datasets can usually be characterized by the type of information source:

- Attribute noise. This type of noise is utilized for corruption in the attribute values of the datasets [28].
- Class noise. This noise type refers to corruption caused by incorrectly classified (labeled) instances that are misclassifications (mislabelings) [20, 27].

The following case simply explains the two types of the noise in datasets.

Assume that classes $A$ and $B$ consist of two types of elements each of which has two attributes, first class has numerical values between $0.25 - 0.30$ for attribute 1 and “small” for the second attribute. For the second class, attribute 1 changes between $0.90 - 1.50$ and has a value “large” for attribute 2. Data examples of these two types of noises are illustrated in Table 1.

In this paper, we demonstrate the performance of our algorithm, on the noisy data with class noise.

Methods described in the literature for handling noisy environments follow two main approaches [28]. One approach is to adapt the main algorithm to handle the noise properly (see for example, [9, 25]). Quinlan [25] enhanced a technique for pruning decision trees for binary classification problems. The other approach is to preprocess the data with a different algorithm by changing or filtering the noisy points. Brodley and Friedl [7] used a technique that preprocesses the data to identify the noisy points.
In this study, we use a new approach for adapting the algorithm to handle the noise. It is shown that, the use of the new decision variables, and the new stopping criteria in the proposed algorithm, help to obtain higher test accuracies for standard and noisy datasets, and to increase the generalization property of the algorithm by reducing gaps between training and test accuracies. The effect of these novelties on the robustness of the algorithm is explained in detail below with illustrative examples.

3. The Parameterized Polyhedral Conic Functions (P-PCF) Algorithm. This section presents the P-PCF algorithm. Assume that the sets \( A, B, I, J \)
and the vectors \( e_m \) and \( e_p \) are defined as in Section 2.1 above.

### 3.1. The P-PCF algorithm. Initialization Step: Let \( l = 1, p = 1, I_l = I, A_l = A, J_l = J, B_l = B \), choose a tolerance parameter \( t \) with \( 0 < t < 1 \), and go to Step 1.

**Step 1:** For all \( k \in I_l \), and all elements \( a_i^k \in A_l \), solve the problem \((PP^l_k)\) given below:

\[
(PP^l_k) \quad \min \left( \frac{w^c_i}{m} + \frac{\gamma^c_i}{p} \right) \\
\text{s.t.} \quad w'(a^i - a_0^k) + \xi_i |a^i - a_0^k|_1 - \gamma + 1 \leq y_i, \quad \forall i \in I_l, \\
- w'(b^j - a_0^k) - \xi_j |b^j - a_0^k|_1 + \gamma + 1 \leq z_j, \quad \forall j \in J_l, \\
y_i \geq 0, i = 1, \ldots, m, z_j \geq 0, j = 1, \ldots, p, w \in \mathbb{R}^n, \xi \in \mathbb{R}, \gamma \geq 1. \quad (13)
\]

**Step 2:** Let \((w^l_k, \xi^l_k, \gamma^l_k, y^l_k, z^l_k)\) be an optimal solution of problem \((PP^l_k)\), corresponding to the element \( a_i^k \in A_l \), and let \( N_{k_0}^l \) be the number of elements of \( A_l \) separated from \( B \) by the sublevel set of the corresponding function \( g^l_k(x) = g(w^l_k, \xi^l_k, \gamma^l_k, y^l_k, z^l_k)(x) \). Let \( g^l_{k_0} \) be the separating function of this iteration which corresponds to the index \( k_0 \) with \( N_{k_0}^l = \max \{ N^l_k : k \in I_l \} \).

If

\[
N_{k_0}^l > t \times \text{card}(A), \quad (14)
\]

then define the separating function with the maximum number of separated elements of this iteration, in the following form:

\[
g_i(x) \equiv g(w^l_k, \xi^l_k, \gamma^l_k, y^l_k, z^l_k)(x) = w^l_k (x - a^l_{k_0}) + \xi^l_{k_0} \| x - a^l_{k_0} \|_1 - \gamma^l_{k_0}, \quad (15)
\]

and GO TO Step 3, otherwise GO TO Step 5.

**Step 3:** Let \( I_{l+1} = \{ i \in I_l : g_i(a^l_i) + 1 > 0 \} \), \( J_{l+1} = \{ j \in J_l : g_j(b^j) + 1 > 0 \} \), \( A_{l+1} = \{ a^l_i \in A_l : i \in I_{l+1} \} \), \( l = l + 1 \). If \( A_l \neq \emptyset \) GO TO Step 1. Otherwise, GO TO Step 5.

**Step 5:** Define the function \( g(x) \), which separates the sets \( A \) and \( B \) in the following form:

\[
g(x) = \min g_i(x) \quad (16)
\]

and Stop.
In the new algorithm, equation (12) is the changed version of equation (4), where the new decision variables \( z_j \) on the right-hand side, serve to relax new separating functions by allowing the sublevel sets of these functions to include some elements from set \( B \) in an optimal way. The optimality is provided by minimizing the sum of these new variables, in the objective function.

The second novelty is the modified stopping criteria, used at Step 2. The M-PCF algorithm is terminated if the last element of \( A \), is separated from \( B \) and for every unseparated element of \( A \), the algorithm generates new separating function. The new algorithm, due to the tolerance parameter \( t \), stops if the number of elements separated by the new separating function generated at Step 2, are recognized as nonproductive. As a result, the new algorithm reduces overfitting by slightly compromising training accuracy and increasing test accuracy. Computational results show that the tolerance parameter controls the level of noise in the data. Every separation function generated by the mathematical model, is examined according to the number of points separated. If the number of points separated by a most productive PCF, is less than the tolerance level \( t \times card(A) \), that PCF is not included in the classifier and algorithm terminates. As a result, some points belonging to set \( A \), are artificially allowed to be nonseparated, and P-PCF algorithm assumes that they are noisy points. We demonstrate that, this approach improves test accuracy and that, overfitting may be decreased by this noise-robust mechanism. This also means that by examining PCFs according to the tolerance parameter, the algorithm filters noisy points.

The P-PCF algorithm is inspired by the studies of Gasimov and Ozturk [11] and Ozturk and Ciftci [24] who used a similar mathematical model for clustering without the tolerance parameter approach.

3.2. Examples. This section presents two illustrative examples for the new method.

3.2.1. Example 1. The data related to this example, are given in Table 2. The whole data in \( A \cup B \), is divided into two subsets: the training set and the test set. For comparison, both the results obtained by applying the M-PCF and P-PCF methods, are presented below, in Figures 1 - 4. The training sets are used to generate separating functions by applying both the methods separately. Then these functions are applied to the trainig and the test subsets of \( A \) and \( B \) to test their abilities to recognize elements of these sets. By simply dividing the numbers of correct predictions to the total number of input samples, the training and the test accuracies are computed.

As it can be seen from Figure 1, M-PCF algorithm performs separation on the training set with 100% accuracy. The algorithm generates 9 separating functions, 4 of them separate only 1 point each, and none of the points from set \( B \) are included in the sets generated by these 9 functions. The result obtained by M-PCF algorithm for the test set is depicted in Figure 2. As it can be seen from this figure, some points of \( B \) are wrongly recognized as points of \( A \) and some functions do not separate any point from \( A \); these are the demonstration of an “unnecessary effort”. These functions may decrease test accuracies.

The results of separation in training and test sets, applying the P-PCF algorithm, are depicted in Figures 3 and 4, respectively. As can be seen from these figures, the P-PCF algorithm does not follow a data-focused separation approach. This means that the optimization model lets some points from \( B \) to belong to the “separated set” together with points of \( A \) and some points of \( A \) may remain outside of this
This example demonstrates that, the algorithm obtains better prediction on the test set, where only one separating function is generated for the training set which separates all points of $A$ with $100\%$ accuracy in test set, see Figure 4.

### Table 2. Data for Example 1

| Training set $A$ (x,y) | Test set $A$ (x,y) | Training set $B$ (x,y) | Test set $B$ (x,y) |
|------------------------|--------------------|------------------------|--------------------|
| (2,4)                  | (3,5)              | (4,19)                 | (5,20)             |
| (2,6)                  | (3,7)              | (6,19)                 | (7,20)             |
| (2,8)                  | (3,9)              | (8,19)                 | (11,20)            |
| (2,10)                 | (3,11)             | (10,19)                | (13,20)            |
| (2,12)                 | (3,13)             | (12,19)                | (15,20)            |
| (2,14)                 | (5,5)              | (14,19)                | (17,20)            |
| (4,4)                  | (5,7)              | (16,19)                | (17,18)            |
| (4,8)                  | (5,9)              | (18,19)                | (17,16)            |
| (4,10)                 | (5,11)             | (16,17)                | (17,14)            |
| (4,14)                 | (5,13)             | (18,17)                | (17,12)            |
| (6,4)                  | (7,5)              | (16,15)                | (17,10)            |
| (6,6)                  | (7,7)              | (18,15)                | -                  |
| (6,8)                  | (7,9)              | (16,13)                | -                  |
| (6,10)                 | (7,11)             | (18,13)                | -                  |
| (6,12)                 | (7,13)             | (16,11)                | -                  |
| (6,14)                 | (9,5)              | (16,9)                 | -                  |
| (8,4)                  | (9,7)              | (18,9)                 | -                  |
| (8,6)                  | (9,9)              | (17,8)                 | -                  |
| (8,8)                  | (9,11)             | (17,6)                 | -                  |
| (8,10)                 | (9,13)             | (17,4)                 | -                  |
| (8,12)                 | (11,5)             | (14,20.5)              | -                  |
| (8,14)                 | (16,7)             | (4,6)                  | -                  |
| (10,4)                 | (18,7)             | (10,6)                 | -                  |
| (10,8)                 | (16,5)             | (4,12)                 | -                  |
| (10,10)                | (16,3)             | (10,12)                | -                  |
| (10,14)                | (11,7)             | (6,21)                 | -                  |
| (12,4)                 | (11,9)             | (10,21)                | -                  |
| (12,6)                 | (11,11)            | (12,21)                | -                  |
| (12,8)                 | (11,13)            | (16,21)                | -                  |
| (12,10)                | -                  | (18,21)                | -                  |
| (12,12)                | -                  | -                      | -                  |
| (12,14)                | -                  | -                      | -                  |
| (8,21)                 | -                  | -                      | -                  |
| (14,21)                | -                  | -                      | -                  |
| (18,11)                | -                  | -                      | -                  |
| (18,5)                 | -                  | -                      | -                  |

3.2.2. Example 2. ([11, Example 2.5])

This example is taken from [11], and aims to illustrate the robustness of P-PCF method, by applying it to a noisy data. The original data consists of 48 points in $R^2$, 24 points in each sets $A$ and $B$. Because the number of data points is not too many, 4−fold cross validation is performed by using the both methods M-PCF and the P-PCF. Totally 12 points of 48 are used for the test sets. The data related to the “first-fold” for the training and the test sets, are given in Table 3. By using Weka filters described in [21], a noisy data is obtained with random noise level of $60\%$. The corresponding noisy data related to the “first-fold” for the training and the test sets, are given in Table 4. For the sake of better explanations, we give
here a set of separating functions obtained by using the both methods just for the “first-folds” of the original and the noisy datasets.
By applying the M-PCF algorithm to the original data, three PCFs denoted by $g_1, g_2, g_3$, and given below, are generated:
Figure 3. The illustration of P-PCF algorithm on the training set for Example 1.

Figure 4. The illustration of P-PCF algorithm on the test set for Example 1.

\[ g_1(x) = 0.028(x_1 - 0.5) + 0.34(x_2 - 2) + 0.33(|x_1 - 0.5| - |x_2 - 2|) - 1, \]

\[ g_2(x) = -0.11(x_1 - 12) - 0.11(x_2 - 2) + 0.33(|x_1 - 12| + |x_2 + 2|) - 1, \]

\[ g_3(x) = 3.38(x_1 - 16) - 23.33(x_2 + 0.5) + 41.33(|x_1 - 16| + |x_2 + 2|) - 52.67. \]

It is remarkable that for the same original data, the P-PCF algorithm generated just one PCF which is denoted by \( g_4 \) and given below:
In a similar form, the results obtained for the noisy data show that, the M-PCF method performs the classification by generating 9 functions, while the P-PCF method generates just 4 functions. All these functions are given below, where \( g_5 \) to \( g_{13} \) denote the functions generated by the M-PCF method, and \( g_{14} \) to \( g_{17} \) denote the functions generated by the P-PCF method.

The classification accuracies obtained by using both the algorithms on the original and noisy data, are shown in Table 5, which demonstrates the higher efficiency and the better generalization property for the P-PCF method.

\[
g_5(x) = 0.66(x_1 - 16) + 0.16(x_2 + 0.5) + 0.16(|x_1 - 16| + |x_2 + 0.5|) - 2.25.
\]

\[
g_6(x) = -0.2(x_1 + 0.5) + 1.23(x_2 + 2) + 1.06(|x_1 + 0.5| + |x_2 + 2|) - 1,
\]
\[
g_7(x) = 3(x_1 - 16) + 0.34(x_2 - 0.5) + 3(|x_1 - 16| + |x_2 - 0.5|) - 1,
\]
\[
g_8(x) = -0.85(x_1 - 16) - 0.32(x_2 - 2) + 0.48(|x_1 - 16| + |x_2 - 2|) - 1,
\]
\[
g_9(x) = 0.2(x_1 - 12) + 0.15(x_2 + 0.5) + 0.64(|x_1 - 12| + |x_2 - 0.5|) - 1,
\]
\[
g_{10}(x) = -0.35(x_1 - 8) + 0.24(x_2 + 6) + 0.64(|x_1 - 8| + |x_2 + 6|) - 1,
\]
\[
g_{11}(x) = 1(x_1 - 13) - 1.07(x_2 + 6) + 1.28(|x_1 - 13| + |x_2 + 6|) - 1,
\]
\[
g_{12}(x) = -0.4(x_1 - 20) + 0.072(x_2 + 6) + 0(|x_1 - 20| + |x_2 + 6|) - 1,
\]
\[
g_{13}(x) = -15(x_1 + 6) + 3(x_2 - 1) + 17(|x_1 + 6| + |x_2 - 1|) - 27.
\]

\[
g_{14}(x) = -0.25(x_1 - 13) + 0.5(x_2 + 6) - 0.25(|x_1 - 13| + |x_2 + 6|) - 1,
\]
\[
g_{15}(x) = -0.13(x_1 + 2) + 0.5(x_2 - 0.5) + 0.32(|x_1 + 2| + |x_2 - 0.5|) - 1.64,
\]
\[
g_{16}(x) = 3.93(x_1 - 12) - 2.37(x_2 + 0.5) + 3.87(|x_1 - 12| + |x_2 + 0.5|) - 1,
\]
\[
g_{17}(x) = -7.8(x_1 + 6) - 4.2(x_2 - 1) + 9.8(|x_1 + 6| + |x_2 - 1|) - 27.
\]

In the next section, the P-PCF method is applied to real-world datasets where the success (in the sense of test accuracy and the overfitting) of the new method becomes more apparent.
4. **Computational results.** This section presents computational results obtained by applying the P-PCF method to real-world datasets, and comparison of these results to those obtained using other methods from the literature. For this purpose, a group of 9 datasets selected from the “UCI repository of machine learning datasets” [2] are used. The explanations and notations on datasets are given in Table 6.

Computational experiments are carried out on original (with 0% noise) and noisy data. A 10-fold cross validation procedure is applied to each dataset to estimate the classification accuracy. The data were randomly divided into 10 (equal) non-overlapping subsets. 9 of those partitions are used as training subsets. By using Weka’s filters, random noise levels of 5%, 10%, 20% and 30% were added to the class variable, only in the training dataset. For this purpose, a procedure described in [21], is used. Separating functions were systematically generated in (noisy) training subsets by applying the classification method, and then tested in the testing subset (remaining group – with 0% noise). This procedure is repeated until each of the 10 subsets has been used as a test set once. Then we take the average of the accuracy measurements over the 10 evaluations and report it as the estimated accuracy. In all computations including the original and the noisy datasets, the value $t = 0.03$ for the parameter $t$ is used in P-PCF method.

M-PCF and P-PCF algorithms are implemented by using the CPLEX solver in GAMS software and the C-Sharp programming language. All the computations are performed on a computer with processor 2.5 GHz Intel Core i7 CPU and 16 GB RAM.

4.1. **Results obtained for the original datasets.** The accuracy results obtained for both the original training and the test sets for each dataset are given in Table 7. These results demonstrate that the P-PCF algorithm reduces overfitting in comparison with the M-PCF algorithm.

4.2. **Results obtained for the noisy data.** In this section, the test accuracy results obtained for the noisy datasets by applying the methods M-PCF, P-PCF, SVM [10], $k$-NN [22] and C4.5 [25], are presented. Default settings of SVM, 1NN, 3NN and C 4.5 algorithms are implemented by using Weka software [13]. All the methods are tested on data with noise levels 0%, 5%, 10%, 20% and 30% and the results are given in Tables 8, 9, 10, 11 and 12, respectively. Bold numbers refer to

| Dataset                  | Short Name | N  | m  | p  | n  |
|--------------------------|------------|----|----|----|----|
| Wisconsin Breast Cancer  | Wis        | 683| 444| 239| 10 |
| German-Credit            | Ger        | 1000| 700| 300| 21 |
| Haberman                 | Hab        | 306| 225| 81 | 4  |
| Hearth-statlog           | Hea        | 270| 137| 160| 14 |
| Ionosphere               | Ion        | 351| 126| 225| 35 |
| Liver-disorders          | Liv        | 345| 145| 200| 7  |
| Sonar                    | Son        | 208| 111| 107| 61 |
| Australian credit        | Aus        | 690| 383| 307| 14 |
| Monk                     | Monk       | 432| 228| 204| 6  |
Table 7. Training and test accuracies obtained by applying M-PCF and P-PCF methods for the original data.

| Dataset | M-PCF Algorithm | P-PCF Algorithm |
|---------|-----------------|-----------------|
|         | Training        | Test            | Training | Test |
| Wis     | 100             | 98.50           | 98.59    | 96.13 |
| Ger     | 100             | 72.41           | 82.56    | 73.80 |
| Hab     | 100             | 74.27           | 86.97    | 74.25 |
| Hea     | 100             | 84.41           | 93.67    | 84.76 |
| Ion     | 100             | 88.42           | 94.87    | 88.96 |
| Liv     | 100             | 68.87           | 78.43    | 69.40 |
| Son     | 100             | 70.24           | 80.47    | 71.09 |
| Aus     | 100             | 85.42           | 87.2     | 86.23 |
| Monk    | 100             | 99.82           | 100      | 99.02 |

Table 8. Test accuracies obtained for datasets with %0 noise.

| Datasets | M-PCF | P-PCF | SVM | 1-NN | 3-NN | C 4.5 |
|----------|-------|-------|-----|------|------|-------|
| Wis      | 98.50 | 96.13 | 95.91 | 91.21 | 95.61 | 92.39 |
| Ger      | 72.41 | 73.80 | 70.35 | 68.50 | 67.70 | 74.5  |
| Hab      | 74.27 | 74.25 | 73.82 | 68.48 | 68.28 | 69.42 |
| Hea      | 84.41 | 84.76 | 78.88 | 69.99 | 68.47 | 70.73 |
| Ion      | 88.42 | 88.96 | 90.48 | 90.22 | 89.98 | 89.87 |
| Liv      | 68.87 | 69.40 | 61.12 | 59.17 | 58.87 | 58.96 |
| Son      | 70.24 | 71.09 | 78.21 | 89.75 | 82.52 | 71.18 |
| Aus      | 85.42 | 86.23 | 85.51 | 80.73 | 85.8  | 84.35 |
| Monk2    | 99.82 | 92.02 | 80.56 | 75.69 | 97.92 | 99.5  |

Table 9. Test accuracies obtained for datasets with %5 noise.

| Datasets | M-PCF | P-PCF | SVM | 1NN | 3NN | C 4.5 |
|----------|-------|-------|-----|-----|-----|-------|
| Wis      | 86.84 | 96.14 | 96.34 | 89.16 | 94.29 | 92.80 |
| Ger      | 70.8  | 72.40 | 73.37 | 68.44 | 65.98 | 63.01 |
| Hab      | 63.9  | 74.44 | 72.17 | 67.29 | 66.25 | 68.47 |
| Hea      | 68.89 | 78.44 | 77.84 | 62.58 | 67.03 | 69.99 |
| Ion      | 67.85 | 85.84 | 89.18 | 88.02 | 89.10 | 88.28 |
| Liv      | 61.46 | 67.76 | 55.29 | 59.52 | 53.85 | 59.45 |
| Son      | 67.14 | 70.52 | 74.37 | 86.53 | 83.31 | 68.64 |
| Aus      | 78.21 | 85.37 | 81.74 | 72.75 | 80.15 | 81.16 |
| Monk     | 88.42 | 98.21 | 77.55 | 73.84 | 90.05 | 95.14 |

the best predictive accuracy of each row. As can be seen from the results given in these tables, the performance achieved by the P-PCF algorithm, compared to the other algorithms, increases with the increasing level of the noise and this method achieves the best test accuracy in seven out of nine used datasets with %20 and %30 noises.

5. Conclusion. This paper presents a classification approach, called the P-PCF algorithm, that reduces overfitting and provides good test accuracies for noisy datasets. This is a new version of the Modified Polyhedral Conic Functions (M-PCF) algorithm, earlier developed by Gasimov and Ozturk. The performance and efficiency of the P-PCF algorithm has been demonstrated on well-known datasets from the literature and on datasets with noise. Computational results show that
Table 10. Test accuracies obtained for datasets with %10 noise.

| Datasets | M-PCF  | P-PCF  | SVM    | 1NN    | 3NN    | C 4.5  |
|----------|--------|--------|--------|--------|--------|--------|
| Wis      | 84.27  | 95.85  | 96.05  | 86.52  | 91.95  | 93.26  |
| Ger      | 68.4   | 73.8   | 70.64  | 65.28  | 64.97  | 60.24  |
| Hab      | 60.54  | 75.65  | 70.28  | 64.23  | 65.42  | 67.93  |
| Hea      | 68.52  | 79.26  | 76.03  | 58.14  | 62.95  | 65.18  |
| Ion      | 65.45  | 83.17  | 81.57  | 86.25  | 88.47  | 85.23  |
| Liv      | 60.29  | 67.49  | 55.50  | 54.07  | 59.20  | 51.36  |
| Son      | 62.64  | 68.56  | 73.97  | 81.18  | 81.11  | 52.36  |
| Aus      | 72.01  | 83.48  | 77.54  | 69.42  | 74.93  | 74.89  |
| Monk     | 78.56  | 96.98  | 74.77  | 69.91  | 81.48  | 89.35  |

Table 11. Test accuracies obtained for datasets with %20 noise.

| Datasets | M-PCF  | P-PCF  | SVM    | 1NN    | 3NN    | C 4.5  |
|----------|--------|--------|--------|--------|--------|--------|
| Wis      | 79.56  | 95.43  | 95.32  | 81.24  | 84.91  | 90.19  |
| Ger      | 65.8   | 71.56  | 65.14  | 60.48  | 62.11  | 61.40  |
| Hab      | 54.28  | 74.54  | 67.25  | 63.28  | 62.47  | 65.37  |
| Hea      | 67.04  | 74.82  | 69.46  | 56.91  | 58.14  | 60.60  |
| Ion      | 62.03  | 80.34  | 81.43  | 82.27  | 86.07  | 79.45  |
| Liv      | 59.72  | 66.70  | 54.28  | 55.01  | 58.76  | 51.69  |
| Son      | 59.64  | 65.98  | 70.84  | 73.91  | 78.21  | 63.9   |
| Aus      | 64.34  | 82.03  | 70.73  | 63.04  | 65.36  | 68.41  |
| Monk     | 71.78  | 92.34  | 66.9   | 62.73  | 70.83  | 77.55  |

Table 12. Test accuracies obtained for datasets with %30 noise.

| Datasets | M-PCF  | P-PCF  | SVM    | 1NN    | 3NN    | C 4.5  |
|----------|--------|--------|--------|--------|--------|--------|
| Wis      | 69.95  | 92.85  | 92.74  | 75.41  | 77.31  | 87.56  |
| Ger      | 62.4   | 69.21  | 63.78  | 57.62  | 56.70  | 53.17  |
| Hab      | 51.83  | 68.66  | 62.87  | 61.82  | 60.44  | 63.53  |
| Hea      | 60.37  | 68.15  | 65.28  | 51.47  | 53.09  | 46.60  |
| Ion      | 61.48  | 80.71  | 76.98  | 78.51  | 84.27  | 77.81  |
| Liv      | 56.58  | 63.37  | 54.12  | 48.66  | 55.40  | 43.45  |
| Son      | 52.64  | 60.01  | 68.64  | 65.23  | 72.80  | 65.75  |
| Aus      | 52.78  | 73.77  | 62.46  | 55.65  | 55.8   | 58.84  |
| Monk     | 51.47  | 90.61  | 60.19  | 53.7   | 60.42  | 66.44  |

the new version achieves the best predictive accuracy for many datasets, which increases with the increase in level of noise. As a future research, we aim to study the multi-class and large scale datasets and the ways of reducing the computational time when the proposed method is applied to large scale datasets.

6. Acknowledgement. The authors would like to thank the reviewers for all of their careful, constructive and insightful comments and suggestions, which have been very helpful in improving the manuscript.

REFERENCES

[1] A. Astorino and M. Gaudioso, Polyhedral separability through successive LP, Journal of Optimization Theory and Applications, 112 (2002), 265–293.
[2] K. Bache and M. Lichman, UCI Machine Learning Repository. University of California, School of Information and Computer Science, (2013). http://archive.ics.uci.edu/ml
[3] A. M. Bagirov, Max–min separability, Optimization Methods and Software, 20 (2005), 277–296.
[4] A. M. Bagirov, G. Ozturk and R. Kasimbeyli, A sharp augmented Lagrangian-based method in constrained non-convex optimization, *Optimization Methods and Software*, 34 (2019), 462–488.

[5] A. M. Bagirov, J. Ugon, D. Webb, G. Ozturk, and R. Kasimbeyli, A novel piecewise linear classifier based on polyhedral conic and max–min separabilities, *TOP*, 21 (2013), 3–24.

[6] K. P. Bennett and O. L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, *Optimization Methods and Software*, 1 (1992), 23–34.

[7] C. E. Brodley and M. A. Friedl, Identifying mislabeled training data, *Journal of Artificial Intelligence Research*, 11 (1999), 131–167.

[8] E. Cimen and G. Ozturk, O-PCF algorithm for one-class classification, *Optimization Methods and Software*, (2019), 1–15.

[9] W. W. Cohen, Fast effective rule induction, *Proceedings of the Twelfth International Conference on Machine Learning, ML95*, San Francisco, CA, 1995, 115–123.

[10] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning*, 20 (1995), 273–297.

[11] R. N. Gasimov and G. Ozturk, Separation via polyhedral conic functions, *Optimization Methods and Software*, 21 (2006), 527–540.

[12] R. N. Gasimov and O. Ustun, Solving the quadratic assignment problem using F-MSG algorithm, *Journal of Industrial and Management Optimization*, 3 (2007), 173–191.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, The WEKA data mining software: An update, *SIGKDD Explorations*, 11 (2003), 10–18.

[14] N. Kasimbeyli and R. Kasimbeyli, A representation theorem for Bishop-Phelps cones, *Pacific Journal of Optimization*, 13 (2017), 55–74.

[15] R. Kasimbeyli, A nonlinear cone separation theorem and scalarization in nonconvex vector optimization, *SIAM Journal on Optimization*, 20 (2010), 1591–1619.

[16] R. Kasimbeyli, Radial epiderivatives and set-valued optimization, *Optimization*, 58 (2009), 521–534.

[17] R. Kasimbeyli and M. Karimi, Separation theorems for nonconvex sets and application in optimization, *Operations Research Letters*, 47 (2019), 569–573.

[18] R. Kasimbeyli and M. Mammadov, Optimality conditions in nonconvex optimization via weak subdifferentials, *Nonlinear Analysis: Theory, Methods and Applications*, 74 (2011), 2534–2547.

[19] R. Kasimbeyli, O. Ustun and A. Rubinov, The modified subgradient algorithm based on feasible values, *Optimization*, 58 (2009), 535–560.

[20] D. T. Larose and C. D. Larose, Discovering knowledge in data: An introduction to data mining, John Wiley & Sons, Hoboken, NJ, 2005.

[21] J. A. Sáez, M. Galar, J. Luengo and F. Herrera, Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness, *Information Sciences*, 247 (2013), 1–20.

[22] X. Zhu and X. Wu, Class noise vs. attribute noise: A quantitative study, *Artificial Intelligence Review*, 22 (2004), 177–210.

Received November 2019; 1st revision March 2020; Final revision July 2020.

E-mail address: msouoz@eskisehir.edu.tr
E-mail address: rkasimbeyli@eskişehir.edu.tr