An incremental piecewise linear classifier based on polyhedral conic separation

Gurkan Ozturk · Adil M. Bagirov · Refail Kasimbeyli

Received: 18 October 2013 / Accepted: 18 May 2014 / Published online: 28 June 2014
© The Author(s) 2014

Abstract In this paper, a piecewise linear classifier based on polyhedral conic separation is developed. This classifier builds nonlinear boundaries between classes using polyhedral conic functions. Since the number of polyhedral conic functions separating classes is not known a priori, an incremental approach is proposed to build separating functions. These functions are found by minimizing an error function which is nonsmooth and nonconvex. A special procedure is proposed to generate starting points to minimize the error function and this procedure is based on the incremental approach. The discrete gradient method, which is a derivative-free method for nonsmooth optimization, is applied to minimize the error function starting from those points. The proposed classifier is applied to solve classification problems on 12 publicly available data sets and compared with some mainstream and piecewise linear classifiers.

Keywords Classification · Polyhedral conic separation · Nonsmooth nonconvex optimization · Discrete gradient method

Editors: Vadim Strijov, Richard Weber, Gerhard-Wilhelm Weber, and Süreyya Ozogur Akyüz.

G. Ozturk (✉) · R. Kasimbeyli
Department of Industrial Engineering, Faculty of Engineering, Anadolu University,
Eskisehir 26555, Turkey
e-mail: gurkan.o@anadolu.edu.tr; gurkanozturk@gmail.com

R. Kasimbeyli
e-mail: rkasimbeyli@anadolu.edu.tr

A. M. Bagirov
School of Science, Information Technology and Engineering, Federation University Australia,
Ballarat 3353, VIC, Australia
e-mail: a.bagirov@federation.edu.au

1 Introduction

Supervised data classification is an important task in data mining. Given input-output examples the aim of supervised data classification is to find a function that maps input data to output data. Such functions, called also classifiers, are computed using classification algorithms. There exist many classifiers based on statistical, decision tree, neural networks and optimization approaches. Among these classifiers piecewise linear classifiers have shown to be effective. They require very low memory and a little testing time. Therefore, these classifiers have very specific applications where other classifiers can not be applied. These applications include small reconnaissance robots, autonomous mobile robots, intelligent cameras, imbedded and real-time systems, portable devices, industrial vision systems, automated visual surveillance systems, monitoring systems (Kostin 2006). In all these systems classifiers must learn without any human intervention.

Piecewise linear classifiers have been a subject of study for more than three decades. To find a piecewise linear boundary between pattern classes is a difficult global optimization task. The objective function in this optimization problem is nonconvex and nonsmooth. The training of piecewise linear classifiers may take a long time. Existing piecewise linear classifiers can be divided into two groups. The first group contains classifiers in which each segment of the piecewise linear boundary is trained locally (*multiple optimization approach*). The second group contains classifiers in which the problem of finding a piecewise linear boundary is formulated as a single optimization problem (*single optimization approach*). Piecewise linear classifiers based on multiple optimization approaches were developed in Gasimov and Ozturk (2006), Kostin (2006), Park and Sklansky (1989), Schulmeister and Wysotzki (1994), Sklansky and Michelotti (1980), Tenmoto et al. (1998). Such classifiers based on single optimization approach were developed in Astorino and Gaudioso (2002), Bagirov (2005) (see also Astorino et al. 2008; Carrizosa and Romero Morales 2013 for review of such classifiers).

Recently incremental approach has become increasingly popular to design classifiers. Note that there are two types of incremental algorithms in supervised data classification. In the first type of algorithms data points incrementally are added and a classifier is updated accordingly. In contrast to this type, in the second type of algorithms data sets are fixed and decision boundaries between classes are built incrementally. Such incremental classifiers were introduced in Bagirov et al. (2011a, b, 2013).

In this paper, we design a new incremental piecewise linear classifier using polyhedral conic separability. Although polyhedral conic functions (PCF) are more complex than linear functions they can provide better approximation to nonlinear boundaries between classes with less number of conic functions. Furthermore linear functions are special case of polyhedral conic functions. Therefore, one can expect that the use of polyhedral conic functions will lead to better approximation of nonlinear boundaries between classes.

In order to find piecewise linear boundaries between classes using PCFs, we introduce a classification error function which is nonsmooth and nonconvex. We also introduce an auxiliary function and design a special procedure to generate starting points by minimizing it. The proposed algorithm is based on the single optimization approach. We test the incremental algorithm using several publicly available large data sets and compare it with some mainstream classifiers.

This paper is organized as follows. In Sect. 2 some preliminaries about PCFs are provided. An auxiliary function is introduced and an algorithm for finding starting points is described in Sect. 3. Section 4 presents an incremental PCF algorithm and its implementation. Results

of numerical experiments are reported in Sect. 5. Finally, Sect. 6 contains some concluding remarks.

2 Preliminaries: separation via polyhedral conic functions

In this section we briefly describe the notions of polyhedral conic functions and polyhedral conic separation. More detailed description can be found in [Gasimov and Ozturk \(2006\)](#). The notion of conic separation is also discussed in [Astorino et al. \(2012\)](#), [Kasimbeyli \(2009\)](#), [Kasimbeyli \(2010\)](#), [Kasimbeyli and Mammadov \(2009\)](#).

Let A and B be given disjoint sets in R^n containing m and p points, respectively:

$$\begin{aligned} A &= \{a^1, \dots, a^m\}, a^i \in R^n, i = 1, \dots, m, \\ B &= \{b^1, \dots, b^p\}, b^j \in R^n, j = 1, \dots, p. \end{aligned}$$

Polyhedral conic functions (PCFs) have recently been proposed in [Gasimov and Ozturk \(2006\)](#) to construct a separation function for the sets A and B .

Definition 1 ([Gasimov and Ozturk 2006](#)) A function $g : R^n \rightarrow R$ is called *polyhedral conic* if its graph is a cone and all its level sets

$$S(\alpha) = \{x \in R^n : g(x) \leq \alpha\}, \quad (1)$$

for $\alpha \in R$, are polyhedral sets.

Example 1 Consider the following function

$$g(x_1, x_2) = 0.11(x_1 - 2) + 0.11(x_2 - 2) + 0.33(|x_1 - 2| + |x_2 - 2|) - 1.$$

This function is polyhedral conic and its graph and level set are illustrated in Figs. 1 and 2, respectively.

Given $w, c \in R^n$, $\xi, \gamma \in R$ a polyhedral conic function $g_{(w, \xi, \gamma, c)} : R^n \rightarrow R$ is defined as follows:

$$g_{(w, \xi, \gamma, c)}(x) = \langle w, x - c \rangle + \xi \|x - c\|_1 - \gamma, \quad (2)$$

where $\|x\|_1 = |x_1| + \dots + |x_n|$ is an l_1 -norm of the vector $x \in R^n$ and $\langle \cdot, \cdot \rangle$ is an inner product in R^n .

Fig. 1 The graph of the polyhedral conic function g

Fig. 2 The level set of the polyhedral conic function g

Fig. 3 Separation using a polyhedral conic function

Lemma 1 (Gasimov and Ozturk 2006) A graph of the function $g_{(w,\xi,\gamma,c)}$ defined in (2) is a polyhedral cone with a vertex at $(c, -\gamma) \in R^n \times R$.

The sets A and B are polyhedral conic separable if there exist a finite number of PCFs $g_l = g_{(w^l,\xi^l,\gamma^l,c^l)}$, $l = 1, \dots, L$ such that

$$\min_{l=1,\dots,L} g_l(a) \leq 0 \quad \forall a \in A$$

and

$$\min_{l=1,\dots,L} g_l(b) > 0 \quad \forall b \in B.$$

The way a PCF can separate two sets A and B in R^2 is shown in Fig. 3. In this figure the set A is shown by red squares and the set B by blue balls. It is obvious that these two sets are linearly inseparable, however the constructed polyhedral conic function can completely separate them.

An algorithm generating a polyhedral conic separating function, therefore called a PCF algorithm, is developed in Gasimov and Ozturk (2006).

An error function for polyhedral conic separation of A and B can be formulated as follows (Bagirov et al. 2013):

$$\Phi(w^1, c^1, \xi^1, \gamma^1, \dots, w^L, c^L, \xi^L, \gamma^L) = \frac{1}{m} \sum_{a \in A} \max \left\{ 0, \min_{l=1, \dots, L} g_l(a) \right\} + \frac{1}{p} \sum_{b \in B} \max \left\{ 0, -\min_{l=1, \dots, L} g_l(b) \right\}. \quad (3)$$

Then the problem of finding polyhedral conic functions separating sets A and B is reduced to the following mathematical programming problem:

$$\text{minimize } \Phi(w^1, c^1, \xi^1, \gamma^1, \dots, w^L, c^L, \xi^L, \gamma^L) \quad (4)$$

subject to

$$w^i, c^i \in R^n, \xi^i, \gamma^i \in R, i = 1, \dots, L. \quad (5)$$

The objective function in Problem given by (4)–(5) is nonsmooth and nonconvex for any $L \geq 1$. It may have many local minimizers and the number of local minimizers increases as the number of polyhedral conic functions L increases. However, global minimizers of Problem given by (4)–(5) are of interest since they provide the least number of PCFs separating the sets A and B with maximum accuracy. The number of variables in Problem given by (4)–(5) is $2(n+1)L$ and becomes large as the number L of PCFs increases. Such problems are out of reach for many existing global optimization techniques and the finding of global minimizers of Problem given by (4)–(5) can become very time consuming.

Therefore, it is the aim of this paper to design an algorithm which is able to find either global or near global minimizers of Problem given by (4)–(5). This algorithm involves a special procedure for generating “promising” starting points which is crucial when one applies a local search method to minimize the function Φ . This procedure is given in the next section.

3 Computation of starting points

In this section we propose an algorithm for finding starting points for solving Problem given by (4)–(5). First we introduce the auxiliary function.

Assume that the solution $g_1, \dots, g_k$ to Problem given by (4)–(5) for $L = k \geq 1$ is known. This means that we have k quadruples $(w^l, c^l, \xi^l, \gamma^l), l = 1, \dots, k$. Then the function separating sets A and B can be expressed as follows:

$$G_k(x) = \min_{l=1, \dots, k} g_l(x). \quad (6)$$

The value of global minimum for the problem given by (4)–(5) is:

$$\Phi_{k, \min} = \frac{1}{|A|} \sum_{a \in A} \max\{0, G_k(a)\} + \frac{1}{|B|} \sum_{b \in B} \max\{0, -G_k(b)\},$$

where $|C|$ is the cardinality of a set C . Using the function G_k we can divide the class A into two subsets: the subset of correctly classified points A^{cc} and the subset of misclassified points A^{mc} . These sets can be described as

$$A^{mc} = \{a \in A : G_k(a) > 0\} \quad (7)$$

and

$$A^{cc} = A \setminus A^{mc}.$$

Let $\bar{g}_{(w,c,\xi,\gamma)}$ be a polyhedral conic function depending on a set of parameters (w, c, ξ, γ) , $w, c \in R^n$, $\xi, \gamma \in R$. Define the function

$$\begin{aligned} \varphi_k(w, c, \xi, \gamma) = & \frac{1}{|A|} \sum_{a \in A} \max \{0, \min \{G_k(a), \bar{g}_{(w,c,\xi,\gamma)}(a)\}\} \\ & + \frac{1}{|B|} \sum_{b \in B} \max \{0, -\min \{G_k(b), \bar{g}_{(w,c,\xi,\gamma)}(b)\}\} \end{aligned} \quad (8)$$

which is called the k -th auxiliary function. This function is nonsmooth and nonconvex. The following problem is called the k -th auxiliary problem:

$$\text{minimize } \varphi_k(w, c, \xi, \gamma) \quad \text{subject to} \quad w, c \in R^n, \xi, \gamma \in R. \quad (9)$$

Consider the following set

$$C_k = \{(w, c, \xi, \gamma) : \bar{g}_{(w,c,\xi,\gamma)}(a) \geq G_k(a), \quad \forall a \in A\}. \quad (10)$$

It is obvious that on this set the function φ_k is constant. Moreover,

$$\varphi_k(w, c, \xi, \gamma) = \Phi_{k,min} = \max \{\varphi_k(\bar{w}, \bar{c}, \bar{\xi}, \bar{\gamma}) : \bar{w}, \bar{c} \in R^n, \bar{\xi}, \bar{\gamma} \in R\}, \forall (w, c, \xi, \gamma) \in C_k$$

and the function φ_k does not depend on variables w, c, ξ, γ in C_k . Therefore, its subdifferential at any point from this set contains the origin meaning that all these points are stationary points of the function φ_k . Any local method may terminate if it starts from these points and therefore such points cannot be considered as good starting points for solving the Problem (9). Consequently we will choose starting points from the complement $\bar{C}_k$ of the set C_k defined as:

$$\bar{C}_k = R^n \setminus C_k = \{(w, c, \xi, \gamma) : \exists a \in A \text{ such that } \bar{g}_{(w,c,\xi,\gamma)}(a) < G_k(a)\}. \quad (11)$$

It is clear that the value of the function φ_k at any point from $\bar{C}_k$ is strictly less than its maximum value $\Phi_{k,min}$ and can be used as a starting point to minimize this function. However, not all these points guarantee decrease of the number of misclassified points and sufficient decrease of the value of φ_k comparing with $\Phi_{k,min}$. In order to achieve such decrease we propose to use misclassified points from A as a center c in the function $\bar{g}$. In this case the new function will correctly classify at least one misclassified point which is the center. An algorithm generating starting points proceeds as follows.

Algorithm 1 Computation of starting points for solving Problem (9).

Input The data set with two classes A and B , k PCFs with the parameters $(w^l, c^l, \xi^l, \gamma^l)$, $l = 1, \dots, k$, a number $u \in [0, 1]$ and a sufficiently small number $\varepsilon > 0$.

Output The set P of starting points for solving Problem (9).

Step 1 (Computing a new PCF) Take each misclassified point $a \in A^{mc}$ as a center and consider k new PCFs with the parameters $(w^l, a, \xi^l, \gamma^l)$, $l = 1, \dots, k$. Find the value $\bar{\gamma}^l$ of γ^l satisfying the following inequalities

$$\bar{g}_{(w^l, a, \xi^l, \bar{\gamma}^l)}(b) > 0, \quad \forall b \in B.$$

To find such a $\bar{\gamma}^l$ compute

$$\eta^l = \min_{b \in B} \bar{g}_{(w^l, a, \xi^l, \gamma^l)}(b)$$

Fig. 4 Computing η in Step 1 of Algorithm 1: Case $\eta < 0$

and define $\bar{\gamma}^l$ as follows:

$$\bar{\gamma}^l = \gamma^l + \eta^l + \varepsilon.$$

Step 2 (Computation of maximum decrease for each misclassified point). For each $a \in A^{mc}$ calculate the decrease in the minimum value of the error function Φ_k when adding the new PCF $\bar{g}_{(w^l, a, \xi^l, \bar{\gamma}^l)}$:

$$D_l(a) = \sum_{x \in A^{mc}} [\max\{0, G_k(x) - \max\{\bar{g}_{(w^l, a, \xi^l, \bar{\gamma}^l)}(x), 0\}\}].$$

Compute

$$D_{max}(a) = \max_{l=1, \dots, k} D_l(a)$$

and

$$(\bar{w}, a, \bar{\xi}, \bar{\gamma}) = \operatorname{argmax}_{l=1, \dots, k} D_l(a).$$

Step 3 (Finding candidates for starting points) Compute the maximum decrease among all misclassified points from the set A^{mc} :

$$\bar{D}_{max} = \max_{a \in A^{mc}} D_{max}(a).$$

Define the set of best candidate centers and parameters as follows:

$$\begin{aligned} \bar{A}^{mc} &= \{a \in A^{mc} : D_{max}(a) \geq \bar{D}_{max}\}, \\ P &= \{(\bar{w}, \bar{a}, \bar{\xi}, \bar{\gamma}) : \bar{a} \in \bar{A}^{mc}\}. \end{aligned}$$

Points from the set P are starting points for solving Problem (9).

In Step 1, we take each misclassified point $a \in A^{mc}$ as a center and compute k new PCFs using the solution to Problem given by (4)–(5) for $L = k$. We update the parameter γ to ensure that the level sets $S_l(0)$, $l = 1, \dots, k$ of the new PCFs, computed by (1), do not contain any point from the set B . If $\eta < 0$ then this set contains points from B . This case is illustrated in Fig. 4. Here the set A is described by blue squares and the set B by red balls. In

Fig. 5 Computing η in Step 1 of Algorithm 1: Case $\eta > 0$

Fig. 6 Updating γ in Step 1 of Algorithm 1

this case we decrease γ by adding $\eta < 0$ and make the sets $S_l(0)$ smaller, so that all points from B lie outside of $S_l(0)$, $l = 1, \dots, k$. It is obvious that in this case $|\eta| < \gamma$ due to the fact that $a \in S_l(0)$, $l = 1, \dots, k$ (see, for details Gasimov and Ozturk 2006).

If $\eta > 0$ the sets $S_l(0)$, $l = 1, \dots, k$ do not contain any point from B . This case is illustrated in Fig. 5. In this case we increase γ by adding $\eta > 0$ and make the sets $S_l(0)$ larger, so that all points from B lie outside of $S_l(0)$, $l = 1, \dots, k$. This may lead to the decrease of the number of misclassified points from the set A^{mc} . Figure 6 illustrates update of the parameter γ .

In Step 2, for each $a \in A^{mc}$ we choose the PCF among k PCFs which provides the largest decrease of the error function. In Step 3, the set of best candidates to be starting points are chosen. If $u = 0$ then $\tilde{A}^{mc} = A^{mc}$ that is we select all PCFs computed in Step 3. If $u = 1$ then we select only PCFs providing largest decrease of the error function.

4 Incremental algorithm

In this section we design an incremental algorithm for solving Problem given by (4)–(5).

4.1 The description of the algorithm

Algorithm 2 Incremental algorithm for solving Problem given by (4)–(5).

Input The data set with two classes A and B .

Output The set of PCFs separating the sets A and B .

Step 1 (Initialization) Select the numbers $u \in [0, 1]$, $v \in [1, \infty)$, the maximum number of PCFs $K_{max} > 0$ and tolerances $\varepsilon_1, \varepsilon_2 > 0$.

Step 2 (Computation of the first PCF) Compute the centroid of the set A and a point $a^1 \in A$ closest to this centroid. Select the point $c^1 = a^1$ as the center of the first PCF and solve Problem given by (4)–(5) for $L = 1$ to find the first PCF $g_1 = g_{(w^1, c^1, \xi^1, \gamma^1)}$. Define G_1 by (6) and compute the set $A^{1,mc} = A^{mc}$ using (7). Set $k := 1$.

Step 3 (Stopping criteria) If one of the following conditions

1. $k > K_{max}$;
2. $A^{k,mc} = \emptyset$;
3. $\Phi(w^1, c^1, \xi^1, \gamma^1, \dots, w^k, c^k, \xi^k, \gamma^k) \leq \varepsilon_1$;
4. $k \geq 2$ and

$$\frac{\Phi(w^1, c^1, \xi^1, \gamma^1, \dots, w^{k-1}, c^{k-1}, \xi^{k-1}, \gamma^{k-1}) - \Phi(w^1, c^1, \xi^1, \gamma^1, \dots, w^k, c^k, \xi^k, \gamma^k)}{\Phi(w^1, c^1, \xi^1, \gamma^1)} \leq \varepsilon_2$$

is satisfied then the algorithm terminates. Otherwise set $k := k + 1$.

Step 4 (Finding the starting points for the auxiliary problem (9)).

Apply Algorithm 1 using the set $A^{mc} = A^{k-1,mc}$. This algorithm generates a set of starting points P .

Step 5 (Solving auxiliary problem (9)).

Take each $(w, a, \xi, \gamma) \in P$ as a starting point and solve the auxiliary problem (9) and find a local minimizer $(\tilde{w}, \tilde{a}, \tilde{\xi}, \tilde{\gamma})$. Denote by $\tilde{P}$ the set of all such local minimizers.

Step 6 (Finding a set of starting points for the k -th PCF in Problem given by (4)–(5)).

Compute

$$\bar{\varphi}_k = \min \left\{ \varphi_k(w, c, \xi, \gamma) : (w, c, \xi, \gamma) \in \tilde{P} \right\}$$

and define the set of starting points as follows:

$$\hat{P} = \left\{ (w, c, \xi, \gamma) \in \tilde{P} : \varphi_k(w, c, \xi, \gamma) \leq v\bar{\varphi}_k \right\}.$$

Step 7 (Solving Problem given by (4)–(5)).

For each $(w, a, \xi, \gamma) \in \hat{P}$ set $(w^k, a^k, \xi^k, \gamma^k) := (w, a, \xi, \gamma)$, take $(w^l, a^l, \xi^l, \gamma^l)$, $l = 1, \dots, k$ as a starting point and solve Problem given by (4)–(5). As a result we get $|\hat{P}|$ number of solutions to this problem. Select a solution with least value of the error function as a solution to Problem given by (4)–(5). Go to Step 3.

4.2 Discussion on Algorithm 2 and its implementation

Algorithm 2 computes a piecewise linear boundary between the sets A and B using polyhedral conic functions. It is an incremental algorithm. This algorithm starts with the computation of one PCF (Step 2) and adds one PCF at each iteration until the separation is obtained with

respect to some tolerance. The algorithm involves solving of two optimization problems at each iteration: the auxiliary problem (9) (Step 5) and the problem given by (4)–(5) (Step 7). Both problems are nonsmooth and nonconvex. Therefore the choice of starting points is crucial if one applies a local method to solve them.

Starting points for solving the auxiliary problem (9) are computed applying Algorithm 1 (Step 4). This algorithm uses misclassified points from the set A as centers for polyhedral conic functions. Moreover, it uses a threshold to determine the most “promising” starting points. Starting points for solving the problem given by (4)–(5) are found by solving the auxiliary problem (9) (Steps 5 and 6). In Step 5 several local minimizers of the auxiliary problem is computed and in Step 6 the best auxiliary function value is computed among all obtained local minimizers. Then using a threshold local minimizers which provide sufficient decrease of the auxiliary function are chosen to be starting points for the k -th PCF. These starting points are used in Step 7 to solve the problem given by (4)–(5).

Algorithm 2 has four stopping criteria given in Step 3. The first stopping criterion is used to restrict the number of PCFs separating two sets. This allows one to avoid possible overfitting. The second stopping criterion in Step 3 means that with the given number of PCFs a perfect separation of the set A from the set B is achieved. The third stopping criterion means that a satisfactory separation is achieved with respect to a given tolerance. Finally, the fourth stopping criterion implies that adding more PCFs will not improve the separation quality and may lead to overfitting.

In order to implement Algorithm 2, one has to select the maximum number K_{max} of PCFs for separating sets A and B . The value of this number depends on the size of a data set. For small data sets this number should be small (between 2 and 4) whereas for large data sets it can be large (between 5 and 10). Such a choice can help to prevent possible overfitting. In Step 3 we compute the centroid of the set A and a point from this set closest to the centroid. This point is selected as a center for the first PCF. In some data sets large neighborhood of the centroid may not contain any point from this data set. In such situations if one chooses the centroid as a center of a PCF, such PCF may not separate even one point from the set A . Therefore, it is preferable to choose a data point as a center. If the center is fixed then the problem given by (4)–(5) is a linear programming problem for one PCF and any linear programming solver can be applied to solve it.

Algorithm 2 has four stopping criteria given in Step 3. First one is about the restriction on the number of PCFs. In the implementation of the algorithm we choose $K_{max} = 5$. The second criterion is applied when there is no misclassified point in the set A which means that the set A is perfectly separated from the set B with the given number of PCFs. The third stopping criterion is satisfied when the sets A and B can be separated with the given number of PCFs within the tolerance ε_1 . We choose $\varepsilon_1 = 0.001$ to allow the algorithm to reach the separation with high accuracy. The fourth criterion is applied when the adding new PCF does not lead to any significant improvement in separation of these two sets. We choose $\varepsilon_2 = 0.01$ to get significant improvement in separation of the sets at each iteration of the algorithm.

In Step 4 Algorithm 1 is applied to find starting points for solving the auxiliary problem (9). This algorithm requires the parameter $u \in [0, 1]$. Small values of u will lead to finding large number of starting points for solving the auxiliary problem. The use of a large number of starting points will make the solution of the auxiliary problem time consuming. On the other side this allows to find either global or near global solution of the auxiliary problem. Values of u close to 1 will allow us to reduce the number of starting points and to choose only those with the decrease of the error function close to the largest decrease. In all data sets we choose $u = 1$. In Step 5 we find the set of local minimizers of the auxiliary problem (9) starting from the points found in Step 4.

In Step 6 the least value of the auxiliary function is computed among all local minimizers found in Step 5. Then using this value the set of starting points for the next PCF to be added to the separating function is computed. In order to compute this set the value of the parameter v should be given. The number of starting points depends on this value. If $v = 1$ then one chooses the local minimizers with the least value of the auxiliary function. If the value of v is significantly large then all local minimizers of the auxiliary problem found in Step 5 are used as the starting points for the next PCF. In our numerical experiments we choose $v = 1$ that is we use only best local minimizers as starting points.

The solution to the problem given by (4)–(5) is found in Step 7. In order to compose starting points for solving this problem local minimizers of the auxiliary problem chosen in Step 6 is added to the $k - 1$ PCFs found at the previous iteration of the incremental algorithm. Thus, the number of starting points is the number of local minimizers of the auxiliary problem. The problem given by (4)–(5) is solved starting from each of these points and the solution with the least value of the error function is accepted as a solution to Problem given by (4)–(5). Such an approach allows to find either global or near global solution of this problem.

The objective functions in problems given by (4)–(5) and (9) are nonsmooth and non-convex. The computation of subgradients of such functions is not an easy task. We propose to apply the discrete gradient method to solve these problems. Details of this method can be found in Bagirov (2003); Bagirov et al. (2008). The discrete gradient method does not require the exact calculation of subgradients, uses only values of a function to approximate subgradients and applicable for solving nonsmooth nonconvex optimization problems. The objective function in both problems are piecewise partially separable. Therefore we apply the version of the discrete gradient method for such problems introduced in Bagirov and Ugon (2006).

In order to solve multi-class data classification problems we use the one-vs-all strategy. This means that for given data set A with $q \geq 2$ classes $A_1, \dots, A_q$ we take any class $A_j, j \in \{1, \dots, q\}$ as the set A and define the set B as a union of all remaining classes. Algorithm 2 generates necessary number of PCFs for each class A_j and this number can be different for different classes. The final separating function for each class is defined as a pointwise minimum of its PCFs. The classification rule is defined as follows. For every new data point (observation) the values of final separating functions for all classes are calculated and the point is classified to the class whose separating function has a minimum value. We implement the proposed algorithm in Fortran 77 and compile it using the *gfortran compiler*.

5 Numerical results

We tested the proposed algorithm—Incremental Polyhedral Conic Separation (IPCS) algorithm on medium sized and large scale real world data sets readily available from the UCI machine learning repository (Bache and Lichman 2013). The selected data sets contain either continuous or integer attributes and have no missing values. Table 1 presents a brief description of the data sets. It contains the number of data points in training and test sets, the number of attributes and classes in each data set.

In our experiments we used some classifiers from WEKA (Waikato Environment for Knowledge Analysis, Version 3.7.10) for a comparison. WEKA is a popular machine learning suite for data mining tasks written in Java and developed at the University of Waikato, New Zealand (see, for details, Hall et al. 2009). We choose representatives of different types of classifiers from WEKA: Naive Bayes (with kernel) (NB kernel), Logistic, Multi-Layer Perceptron (MLP), support vector machines classifiers Linear LibSVM (LibSVM (LIN)), Lib-

Table 1 Brief description of data sets

Data sets	(Train,test)	No. of input attributes	No. of classes
Shuttle control (SH)	(43500,14500)	9	7
Letter recognition (LET)	(15000,5000)	16	26
Landsat satellite image (LSI)	(4435,2000)	36	6
Pen-based recognition of			
Handwritten digits (PD)	(7494,3498)	16	10
Page blocks (PB)	(4000,1473)	10	5
Optical recognition of			
Handwritten digits (OD)	(3823, 1797)	64	10
Spambase (SB)	(3682,919)	57	2
Abalone (AB)	(3133,1044)	8	3
DNA	(2000, 1186)	180	3
Isolet (ISO)	(6238, 1559)	617	26
Phoneme_ CR (PHON)	(4322, 1082)	5	2
Texture_ CR (TEXT)	(4400, 1100)	40	11

SVM with polynomial kernel (LibSVM(POL)), LibSVM with RBF kernel (LibSVM(RBF)), SMO with normalized polynomial kernel (SMO (NPOL)) and SMO (PUK), a decision tree classifier J48 (which is an implementation of the C4.5 algorithm) and a rule based classifier PART. We apply all classifiers from WEKA with the default parameter values, except LibSVM(POL) and LibSVM(RBF) classifiers. In these two classifiers a special procedure is applied to estimate optimal values of parameters. We also include two piecewise linear classifiers: the incremental max-min separability algorithm (CIMMS) from [Bagirov et al. \(2011b\)](#) and the Hybrid Polyhedral Conic and Max–min Separability (HPCAMS) classifier from [Bagirov et al. \(2013\)](#) in our experiments.

Numerical experiments were carried on a PC Intel(R) Core(TM) i5-3470S CPU 2.90 GHz with 8GB of RAM running Windows 7. In the tables a dash line shows that an algorithm requires more memory than available.

Results for test set accuracies on different data sets using different classifiers are given in Tables 2 and 3. The proposed classifier achieves best test accuracy in two data sets: Abalone and Spambase data sets. In two data sets: Shuttle Control and Texture_ CR this classifier's test accuracy is close to the best accuracy achieved by all used classifiers. In all other data sets its accuracy is reasonably high in comparison with other classifiers. This means that the proposed classifier produces stable results across different data sets.

We say that two classifiers produce the similar test set accuracy if the difference between their accuracies is within the range of 0.5 %. Table 4 presents pairwise comparison of the IPCS classifier with others using testing accuracy. These results clearly demonstrate that the proposed classifier produces better or similar test set accuracy in most data sets in pairwise comparison with other classifiers, except the LibSVM(RBF) classifier. Furthermore, the IPCS classifier generates better test set accuracy than others, except the LibSVM(RBF) and SMO(PUK) classifiers, in at least half of data sets.

Tables 5, 6, 7 present the total training (t_{train}) and testing (t_{test}) time used by the classifiers. Note that these results were obtained using different platforms (WEKA and Fortran 77), however they give some estimation on training and testing time required by the classifiers. One can see that with a few exceptions all three piecewise linear classifiers require (sometimes

Table 2 Test set accuracy for different classifiers

Data set Classifier	AB	DNA	LSI	LET	OD	PD
NB kernel	57.85	93.34	82.10	74.12	90.32	84.13
Logistic	64.27	88.36	83.75	77.40	92.21	92.85
MLP	63.51	93.68	88.50	83.20	96.55	89.85
LibSVM (LIN)	60.73	93.09	85.05	82.40	96.55	95.00
LibSVM(POL)	55.93	50.84	83.65	94.98	97.89	97.31
LibSVM(RBF)	57.75	94.52	89.60	97.44	98.36	98.37
SMO (NPOL)	60.25	95.36	79.60	82.18	96.66	96.86
SMO (PUK)	64.18	57.93	91.45	95.92	96.61	97.88
J48	60.15	92.50	85.35	87.70	85.75	92.05
PART	57.95	91.06	85.25	87.32	89.54	93.65
CIMMS	65.80	93.42	88.15	91.90	94.27	96.63
HPCAMS	66.09	94.18	87.15	91.04	93.10	96.57
IPCS	66.19	94.18	88.20	90.50	94.05	96.77

Table 3 Test set accuracy for different classifiers (cont.)

Data set Classifier	PHON	SH	TEXT	ISO	PB	SB
NB(kernel)	76.53	98.32	81.00	84.22	88.39	76.17
Logistic	74.58	96.84	99.64	–	91.79	92.06
MLP	81.52	99.75	99.91	96.34	92.80	92.06
LibSVM (LIN)	77.54	20.78	99.18	96.02	87.03	90.97
LibSVM(POL)	77.26	99.94	88.36	91.85	84.32	38.52
LibSVM(RBF)	84.19	98.26	99.45	95.70	90.92	92.27
SMO (NPOL)	78.74	96.81	97.27	–	89.48	92.60
SMO (PUK)	83.27	99.50	99.55	–	88.53	93.04
J48	85.67	99.95	93.91	83.45	93.55	92.93
PART	82.72	99.98	93.82	82.81	92.46	91.40
CIMMS	81.05	99.84	99.82	95.19	87.10	93.80
HPCAMS	80.13	99.86	99.36	93.52	89.55	93.47
IPCS	81.24	99.78	99.55	93.84	88.53	93.91

significantly) more training time than other classifiers. However testing time required by the piecewise linear classifiers is significantly less than that of support vector machines classifiers and also the Naive Bayes classifier. Logistic and MLP classifiers, in general, require more testing time than the piecewise linear classifiers. Testing time used by the rule-based classifier PART and the decision tree based classifier J48 is similar to that of used by the piecewise linear classifiers.

Table 8 contains the total and average (per class) number of PCFs used by the IPCS algorithm to separate classes. One can see that this classifier uses in average a few polyhedral conic functions to separate each class from the rest of a data set. Therefore, in testing phase the memory usage of this classifier is very low.

Table 4 Pairwise comparison of the IPCS classifier with others using test set accuracy

Classifier	# and proportion of data sets for which the proposed classifier is similar or better		# and proportion of data sets for which the proposed classifier is strictly better	
	Number	Proportion (%)	Number	Proportion (%)
NB(kernel)	12	100.00	12	100.00
Logistic	11	91.67	10	83.33
MLP	10	75.00	7	50.00
LibSVM(LIN)	10	83.33	10	83.33
LibSVM(POL)	9	75.00	8	66.67
LibSVM(RBF)	5	41.67	4	33.33
SMO (NPOL)	9	75.00	8	66.67
SMO (PUK)	8	66.67	5	41.67
J48	10	83.33	9	75.00
PART	10	83.33	9	75.00
CIMMS	10	83.33	7	58.33
HPCAMS	10	83.33	8	66.67

Table 5 Comparison of classifiers using training and testing time

Data set Classifier	AB		DNA		LSI		LET	
	t_{train}	t_{test}	t_{train}	t_{test}	t_{train}	t_{test}	t_{train}	t_{test}
NB kernel	0.02	0.60	0.06	0.65	0.03	1.85	0.05	3.55
Logistic	0.15	0.01	1.79	0.04	2.78	0.02	93.06	0.24
MLP	4.10	0.00	443.99	0.32	60.10	0.05	237.18	0.13
LibSVM (LIN)	0.44	0.12	1.02	0.39	0.81	0.49	7.17	8.96
LibSVM(POL)	0.6	0.14	2.97	1.14	2.17	0.46	6.57	6.98
LibSVM(RBF)	0.69	0.17	4.17	1.58	0.87	0.54	5.67	8.95
SMO (NPOL)	10.90	1.51	9.37	10.63	16.68	12.91	106.03	309.65
SMO (PUK)	10.19	0.80	15.68	18.84	6.00	5.85	60.46	174.22
J48	0.11	0.01	0.31	0.04	0.33	0.02	1.21	0.06
PART	0.33	0.00	0.84	0.04	1.96	0.03	11.71	0.12
CIMMS	27.22	0.00	32.06	0.02	523.28	0.03	9941.34	0.15
HPCAMS	37.03	0.00	42.27	0.03	451.67	0.03	7389.73	0.16
IPCS	18.94	0.00	37.88	0.02	898.78	0.02	8870.91	0.06

Based on the results presented in Tables 5, 6, 7 and 8 we can conclude that although the proposed classifier uses significantly more training time than most of classifiers however it requires a little testing time and its memory usage is very low. Since in many applications training of classifiers can be done off-line, the proposed classifier can be considered as an accurate and efficient real time classifier. This makes it applicable in many real world applications listed in introduction section of this paper.

Table 6 Comparison of classifiers using training and testing time (cont)

Data set Classifier	OD		PD		PHON		SH	
	t_{train}	t_{test}	t_{train}	t_{test}	t_{train}	t_{test}	t_{train}	t_{test}
NB kernel	0.05	1.92	0.03	3.29	0.02	1.04	0.09	1.56
Logistic	12.88	0.04	11.84	0.04	0.08	0.02	738.94	0.15
MLP	151.04	0.10	45.22	0.04	2.80	0.01	118.80	0.12
LibSVM (LIN)	0.65	0.4	0.81	0.33	0.68	0.12	21.41	6.83
LibSVM(POL)	0.71	0.47	0.62	0.31	0.63	0.1	11.97	3.4
LibSVM(RBF)	2.52	0.95	0.66	0.41	0.71	0.13	12.71	3.79
SMO (NPOL)	4.97	10.54	7.83	9.25	16.88	0.70	907.41	67.22
SMO (PUK)	17.61	57.72	10.59	11.59	13.82	0.36	275.17	18.74
J48	0.34	0.03	0.30	0.04	0.08	0.01	0.94	0.01
PART	0.99	0.03	0.63	0.03	0.13	0.00	0.82	0.09
CIMMS	81.88	0.04	203.02	0.03	34.75	0.00	782.47	0.03
HPCAMS	106.31	0.05	158.91	0.03	39.13	0.00	731.70	0.03
IPCS	1652.49	0.05	563.97	0.03	48.42	0.00	2254.62	0.06

Table 7 Comparison of classifiers using training and testing time (cont)

Data set Classifier	TEXT		ISO		PB		SB	
	t_{train}	t_{test}	t_{train}	t_{test}	t_{train}	t_{test}	t_{train}	t_{test}
NB kernel	0.05	9.44	2.01	628.42	0.00	0.27	0.03	0.24
Logistic	5.59	0.04	–	–	9.80	0.01	0.36	0.02
MLP	42.81	0.06	17655.78	5.45	5.22	0.01	40.89	0.03
LibSVM (LIN)	0.68	0.16	24.51	15.35	0.54	0.07	1.26	0.14
LibSVM(POL)	0.64	0.15	23.19	12.92	0.58	0.07	2.14	0.37
LibSVM(RBF)	0.65	0.2	173.05	36.06	0.49	0.08	1.05	0.15
SMO (NPOL)	6.38	3.30	–	–	2.64	0.18	6.63	0.41
SMO (PUK)	2.23	1.09	–	–	1.55	0.12	5.78	0.33
J48	0.64	0.02	27.50	0.21	0.06	0.01	0.21	0.02
PART	2.54	0.03	239.50	0.24	0.08	0.01	0.55	0.01
CIMMS	47.28	0.02	3927.30	1.72	27.63	0.02	295.23	0.00
HPCAMS	55.33	0.02	2994.64	1.86	89.55	0.02	240.20	0.02
IPCS	32.26	0.00	697.23	0.64	40.89	0.00	107.45	0.02

6 Conclusions

In this paper, the piecewise linear classifier is designed to solve supervised data classification problems. This classifier uses several polyhedral conic functions to separate classes and is based on the incremental approach. More specifically, it starts with one polyhedral conic function and adds a new polyhedral conic function at each iteration until the satisfactory separation of classes is achieved. Polyhedral conic functions are computed by minimizing the error function which is nonsmooth and nonconvex. The calculation of subgradients of

Table 8 The total and average number of PCFs used by the IPCS classifier

Data set	total # of PCFs	average # of PCFs
AB	10	3.3
DNA	4	1.3
LSI	21	3.5
LET	122	4.7
OD	24	2.4
PD	25	2.5
PHON	10	5.0
SH	19	2.7
TEXT	11	1.0
ISO	26	1.0
PB	10	2.0
SB	4	2.0

the error function is not an easy task. We apply the discrete gradient method to minimize it. This method does not require the exact calculation of subgradients.

In order to find good starting points for minimization of the error function we formulate the auxiliary problem using polyhedral conic functions computed in previous iterations of the incremental algorithm. Such an approach allows to find either global or near global minimizers of the error function and to compute as few polyhedral conic functions as necessary to separate classes. The proposed classifier is tested using 12 test data sets. Results of numerical experiments demonstrate its high efficiency. Comparison with some mainstream classifiers show that the proposed classifier is able to produce better or similar test set accuracy in data sets used in numerical experiments. The training time of this classifier is more than that of many other classifiers however its testing is almost instantaneous for the whole test set. The proposed classifier needs few polyhedral conic functions to carry out classification task and therefore requires very low memory. All these make this classifier highly efficient for real-time classification.

Acknowledgments The authors would like to thank three anonymous referees for their criticism and comments which allowed to improve the quality of the paper. The research by Dr. A.M. Bagirov was supported under Australian Research Council's Discovery Projects funding scheme (Project No. DP140103213).

References

- Astorino, A., Fuduli, A., & Gorgone, E. (2008). Non-smoothness in classification problems. *Optimization Methods and Software*, 23(5), 675–688. doi:[10.1080/10556780802264071](https://doi.org/10.1080/10556780802264071).
- Astorino, A., & Gaudioso, M. (2002). Polyhedral separability through successive lp. *Journal of Optimization Theory and Applications*, 112(2), 265–293. doi:[10.1023/A:1013649822153](https://doi.org/10.1023/A:1013649822153).
- Astorino, A., Gaudioso, M., & Seeger, A. (2012). Conic separation of finite sets. I The homogeneous case. *Journal of Convex Analysis*, 53, 301–322.
- Bache, K., & Lichman, M. (2013). UCI machine learning repository <http://archive.ics.uci.edu/ml>.
- Bagirov, A. (2003). Continuous subdifferential approximations and their applications. *Journal of Mathematical Sciences*, 115(5), 2567–2609. doi:[10.1023/A:1023227716953](https://doi.org/10.1023/A:1023227716953).
- Bagirov, A., Karasözen, B., & Sezer, M. (2008). Discrete gradient method: Derivative-free method for non-smooth optimization. *Journal of Optimization Theory and Applications*, 137(2), 317–334. doi:[10.1007/s10957-007-9335-5](https://doi.org/10.1007/s10957-007-9335-5).

- Bagirov, A., & Ugon, J. (2006). Piecewise partially separable functions and a derivative-free algorithm for large scale nonsmooth optimization. *Journal of Global Optimization*, 35(2), 163–195. doi:[10.1007/s10898-005-3834-4](https://doi.org/10.1007/s10898-005-3834-4).
- Bagirov, A., Ugon, J., & Webb, D. (2011a). An efficient algorithm for the incremental construction of a piecewise linear classifier. *Information Systems*, 36(4), 782–790. doi:[10.1016/j.is.2010.12.002](https://doi.org/10.1016/j.is.2010.12.002)
- Bagirov, A., Ugon, J., Webb, D., & Karasözen, B. (2011b). Classification through incremental maxmin separability. *Pattern Analysis and Applications*, 14(2), 165–174. doi:[10.1007/s10044-010-0191-9](https://doi.org/10.1007/s10044-010-0191-9).
- Bagirov, A. M. (2005). Max–min separability. *Optimization Methods and Software*, 20(2–3), 277–296. doi:[10.1080/10556780512331318263](https://doi.org/10.1080/10556780512331318263).
- Bagirov, A. M., Ugon, J., Webb, D., Ozturk, G., & Kasimbeyli, R. (2013). A novel piecewise linear classifier based on polyhedral conic and max-min separabilities. *TOP*, 21(1), 3–24. doi:[10.1007/s11750-011-0241-5](https://doi.org/10.1007/s11750-011-0241-5).
- Carrizosa, E., & Romero Morales, D. (2013). Supervised classification and mathematical optimization. *Computers and Operations Research*, 40, 150–165.
- Gasimov, R. N., & Ozturk, G. (2006). Separation via polyhedral conic functions. *Optimization Methods and Software*, 21(4), 527–540.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 10–18. doi:[10.1145/1656274.1656278](https://doi.org/10.1145/1656274.1656278).
- Kasimbeyli, R. (2009). Radial epiderivatives and set-valued optimization. *Optimization*, 58(5), 521–534. doi:[10.1080/02331930902928310](https://doi.org/10.1080/02331930902928310).
- Kasimbeyli, R. (2010). A nonlinear cone separation theorem and scalarization in nonconvex vector optimization. *SIAM Journal on Optimization*, 20(3), 1591–1619. doi:[10.1137/070694089](https://doi.org/10.1137/070694089).
- Kasimbeyli, R., & Mammadov, M. (2009). On weak subdifferentials, directional derivatives, and radial epiderivatives for nonconvex functions. *SIAM Journal on Optimization*, 20(2), 841–855. doi:[10.1137/080738106](https://doi.org/10.1137/080738106).
- Kostin, A. (2006). A simple and fast multi-class piecewise linear pattern classifier. *Pattern Recognition*, 39(11), 1949–1962 doi:[10.1016/j.patcog.2006.04.022](https://doi.org/10.1016/j.patcog.2006.04.022).
- Park, Y., & Sklansky, J. (1989). Automated design of multiple-class piecewise linear classifiers. *Journal of Classification*, 6(1), 195–222. doi:[10.1007/BF01908599](https://doi.org/10.1007/BF01908599).
- Schulmeister, B., & Wysotzki, F. (1994). The piecewise linear classifier dipol92. In: F. Bergadano, L. Raedt (eds.) *Machine Learning: ECML-94, Lecture Notes in Computer Science*, vol. 784, pp. 411–414. Springer, Berlin Heidelberg. doi:[10.1007/3-540-57868-4_86](https://doi.org/10.1007/3-540-57868-4_86).
- Sklansky, J., & Michelotti, L. (1980). Locally trained piecewise linear classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(2), 101–111. doi:[10.1109/TPAMI.1980.4766988](https://doi.org/10.1109/TPAMI.1980.4766988).
- Tenmoto, H., Kudo, M., & Shimbo, M. (1998). Piecewise linear classifiers with an appropriate number of hyperplanes. *Pattern Recognition*, 31(11), 1627–1634. doi:[10.1016/S0031-3203\(98\)00016-8](https://doi.org/10.1016/S0031-3203(98)00016-8).