Binary classification posed as a quadratically constrained quadratic programming and solved using particle swarm optimization

Deepak Kumar · A G Ramakrishnan

Abstract Particle swarm optimization (PSO) is used in several combinatorial optimization problems. In this work, particle swarms are used to solve quadratic programming problems with quadratic constraints. The central idea is to use PSO to move in the direction towards optimal solution rather than searching the entire feasible region. Binary classification is posed as a quadratically constrained quadratic problem and solved using the proposed method. Each class in the binary classification problem is modeled as a multidimensional ellipsoid to form a quadratic constraint in the problem. Particle swarms help in determining the optimal hyperplane or classification boundary for a data set. Our results on the Iris, Pima, Wine, Thyroid, Balance, Bupa, Haberman, and TAE datasets show that the proposed method works better than a neural network and the performance is close to that of a support vector machine.

Keywords Quadratic programming · Particle swarm optimization · Hyperplane · Quadratic constraints · Binary classification

1 Introduction

A class of algorithms originated for minimizing or maximizing a function $f(\mathbf{x})$, while satisfying some constraints $g(\mathbf{x})$. In the early history of optimization, the function $f(\mathbf{x})$ and the constraints $g(\mathbf{x})$ were linear and the problem was known as linear programming (LP). One of the earliest algorithms for solving the linear programming problem was given by Dantzig, popularly known as Simplex method (Dantzig, 1963). As the number of dimensions and constraints increased, solving the LP problem using simplex method became hard. The inability of simplex method was that it could not solve LP problem in polynomial time. Khachiyan

Deepak Kumar, A G Ramakrishnan
Medical Intelligence and Language Engineering (MILE) Laboratory
Department of Electrical Engineering,
Indian Institute of Science, Bangalore - 560012, India
Tel.: +91-80-22932935
Fax: +91-80-23600444
E-mail: dipkmr@gmail.com,ramkiag@ee.iisc.ernet.in

proposed ellipsoid algorithm as an alternative to simplex method and proved that it could reach the solution iteratively in polynomial time (Khachiyan, 1979). The practical infeasible condition of ellipsoid algorithm led to the evolution of several interior or barrier point methods. One of the well known interior point methods is Karmarkar's method proposed by Narendra Karmarkar (Karmarkar, 1984).

Binary classification is one of the active research areas in machine learning (Bishop, 2006; Duda et al, 2001). There are several ways to train a binary classifier. The features and the class labels of the training data set can be stored and retrieved during classification using the approach of nearest neighbor (Derrac et al, 2014). A hyperplane is learnt for classification by training a neural network, which may not always be optimal (Bishop, 1995). Vapnik and others formulated the problem of classification as optimization. This method is known as support vector machines (SVMs) (Cortes and Vapnik, 1995). Sequential minimal optimization (SMO) is a technique which solves the optimization problem in SVMs (Platt, 1998). Decision trees, bagging, and boosting techniques are also used in binary classification (Duda et al, 2001; Fernández et al, 2013; Galar et al, 2013; López et al, 2013).

1.1 Motivation

Nearest neighbor method does not involve any modeling to reduce the storage of the training data set (Derrac et al, 2014). On the other hand, neural network and SVM model the data with an objective function to estimate a hyperplane, which is used in classification. In neural network approach, the objective function is a least squares, which is quadratic in nature and is minimized for the given data set. The hyperplane obtained by training a neural network may or may not be optimal, since it depends on the number of layers and weights used to train the network. SVM uses quadratic programming formulation with linear constraints for minimizing the objective function (Gonzalez-Abril et al, 2013). Even though the objective function used in a neural network or SVM is a quadratic programming problem, the constraints are linear.

If there is a way to model linear constraints as quadratic constraints, then the objective function becomes quadratically constrained quadratic programming (QCQP). In this paper, binary classification is posed as a QCQP problem and a novel solution is proposed using particle swarm optimization (PSO). One of the advantages of this approach is that it solves the QCQP problem without the need for gradient estimation.

The paper is organized as follows: QCQP and PSO are described in the background section and the solution for quadratically constrained quadratic programming using particle swarms is described in Sec. 3. The proposed method is compared with Khachiyan's and Karmarkar's algorithms for linear programming and with neural networks and SVM for quadratic programming in Sec. 4 on experiments and results. Section 5 concludes the paper with suggestions for future work.

2 Background

The formulation of QCQP and PSO are described in the following subsections.

2.1 Quadratically Constrained Quadratic Programming

An optimization problem has the form (Boyd and Vandenberghe, 2004):

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq b_i,\ i = 1, 2, \ldots, m \end{array} \tag{1}$$

where, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is the variable of the problem. The function $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is the objective function. The function $g_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \ldots, m$ are the constraint functions and the constants $b_1, b_2, \ldots, b_m$ are the limits for the constraints.

When the objective function is quadratic and the constraints are quadratic, then the optimization problem becomes a quadratically constrained quadratic program. A general quadratically constrained quadratic programming problem is expressed as follows (Bomze, 1998; Bomze and Schachinger, 2010; Boyd and Vandenberghe, 2004):

$$\begin{array}{lrcl} \text{minimize} & f(\mathbf{x}) & = & \mathbf{x}^T P_0 \mathbf{x} + 2q_0^T \mathbf{x} + r_0 \\ \text{subject to} & \mathbf{x}^T P_i \mathbf{x} + 2q_i^T \mathbf{x} + r_i & \leq & 0, \quad i = 1, 2, \ldots, m \end{array} \tag{2}$$

where, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^n$. $P_0 \in \mathbb{R}^{n*n}$, $q_0 \in \mathbb{R}^n$, and $r_0 \in \mathbb{R}$ define the quadratic function $f(\mathbf{x})$. $P_i \in \mathbb{R}^{n*n}$, $q_i \in \mathbb{R}^n$, and $r_i, i = 1, 2, \ldots, m$ are the constants of the quadratic constraints.

If P_i is positive semi-definite matrix for $i = 0, 1, 2, \ldots, m$, then the problem becomes convex QCQP. When P_0 is zero, then the problem becomes linear programming with quadratic constraints.

2.2 Particle Swarm Optimization

Particle swarm optimization was proposed for optimizing the weights of a neural network (Kennedy and Eberhart, 1995). PSO has been applied to numerous applications for optimizing non-linear functions (Kennedy et al, 2001; Spadoni and Stefanini, 2012; Zhan et al, 2009). PSO evolved by simulating bird flocking and fish schooling. The advantages of PSO are that it is simple in conception and easy to implement. Particles are deployed in search space and each particle is evaluated against an optimization function. The best particle is chosen as a directing agent for the rest. The velocity of each particle is controlled by both the particle's personal best and the global best. During the movement of the particles, a few of them may reach the global best. Several iterations are required to reach the global best.

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\}$ be the particles deployed in the search space of the optimization function, where k is the number of particles and $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k\}$ be the velocities of the respective particles. $\mathbf{x}_i,\ \mathbf{v}_i \in \mathbb{R}^n$ for all the k particles. A simple PSO update is as follows.

- Velocity update equation

$$\mathbf{v}_i^j = w\mathbf{v}_i^{j-1} + c_1 r_1\ (\mathbf{x}_{bi} - \mathbf{x}_i^{j-1}) + c_2 r_2 (\mathbf{x}_{bg} - \mathbf{x}_i^{j-1}) \tag{3}$$

where w is the weight for the previous velocity; c_1, c_2 are constants and r_1, r_2 are random values varied in each iteration. $\mathbf{x}_{bi}$ is the personal best value for particle i and $\mathbf{x}_{bg}$ is the global best value among all the particles. $\mathbf{v}_i^j$ is the updated velocity of the i^{th} particle in the j^{th} iteration and $\mathbf{v}_i^{j-1}$ is its velocity in the $(j-1)^{th}$ iteration. $\mathbf{x}_i^{j-1}$ is the position of the i^{th} particle after the $(j-1)^{th}$ iteration.

- For position update, the updated velocity is added to the existing position of the particle. The position update equation is

$$\mathbf{x}_i^j = \mathbf{x}_i^{j-1} + \mathbf{v}_i^j \tag{4}$$

Table 1 presents a pseudo code of Particle Swarm Optimization. Initially, the particle with minimum $f(\mathbf{x})$ is considered as the global best. After every iteration, the global best value is updated if necessary. Also, the personal best value is updated.

Table 1 Pseudo code of PSO

```
Inputs: $k$, $t_{max} = T$ and minimize $f(x)$; initialize parameters $\mathbf{x_i},\ \mathbf{v_i}$ and set $w,\ c_1,\ c_2$
Outputs: Global best value $\mathbf{x}_{bg}$
  $t = 0$,
  $\mathbf{x}_{bg} = \mathbf{x}_0$
  for $i = 1$ to $k$
    $\mathbf{x}_{bi} = \mathbf{x}_i$
    if $f(\mathbf{x}_{bg}) > f(\mathbf{x}_{bi})$
      $\mathbf{x}_{bg} = \mathbf{x}_{bi}$
    end if
  end for
  while $t < t_{max}$
    $t \leftarrow t + 1$
    Velocity update step:
      Randomly choose values for $r_1,\ r_2$ in the range '0' to '1'.
      Then update the velocity of each particle.
      $\mathbf{v}_i^j = w\mathbf{v}_i^{j-1} + c_1 r_1\ (\mathbf{x}_{bi} - \mathbf{x}_i^{j-1}) + c_2 r_2 (\mathbf{x}_{bg} - \mathbf{x}_i^{j-1})$
    Position update step:
      Add the updated velocity to the existing position.
      for $i = 1$ to $k$
        $\mathbf{x}_i^j = \mathbf{x}_i^{j-1} + \mathbf{v}_i^j$
        if $f(\mathbf{x}_{bi}) > f(\mathbf{x}_i)$
          $\mathbf{x}_{bi} = \mathbf{x}_i$
        end if
        if $f(\mathbf{x}_{bg}) > f(\mathbf{x}_{bi})$
          $\mathbf{x}_{bg} = \mathbf{x}_{bi}$
        end if
      end for
  end while
```

3 Proposed method

Our interest lies in determining the shortest path between the two non-intersecting ellipsoids to perform binary classification. The shortest path between the two ellipsoids in $\mathbb{R}^n$ can be formulated as a convex QCQP problem.

3.1 Formulation

Arriving at the two end points of the shortest path, one on each ellipsoid, can be posed as minimization in a quadratic form and formulated as follows.

$$\begin{array}{lll} \text{minimize } f_1(\mathbf{x}) \; = \; (\mathbf{x}-\mathbf{y})^T P_0(\mathbf{x}-\mathbf{y}) & & \\ \text{subject to} \qquad \mathbf{x}^T P_1 \mathbf{x} \leq 1 & \mathbf{x} \in X & \\ \qquad\qquad\qquad \mathbf{y}^T P_2 \mathbf{y} \leq 1 & \mathbf{y} \in Y & \end{array} \tag{5}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and X,Y are non-intersecting regions in $\mathbb{R}^n$, $X \cap Y = 0$. P_1 and P_2 are the matrices that characterize the ellipsoids modeling the two classes.

In case the Euclidean distance metric is used for minimization of the path length, then P_0 is the identity matrix. The modified equation is,

$$\begin{array}{lll} \text{minimize } f_2(\mathbf{x}) \; = \; \| \mathbf{x}-\mathbf{y} \|^2 & & \\ \text{subject to} \qquad \mathbf{x}^T P_1 \mathbf{x} \leq 1 & \mathbf{x} \in X & \\ \qquad\qquad\qquad \mathbf{y}^T P_2 \mathbf{y} \leq 1 & \mathbf{y} \in Y & \end{array} \tag{6}$$

Thus, in this paper, we address only this limited problem for binary classification, rather than the more general problem given by Eq(2).

3.2 Solution using modified PSO

Suppose one end point ($\mathbf{y}$) in the shortest path is known and fixed as $\mathbf{a}$. Then, Eq (6) can be reformulated as,

$$\begin{array}{ll} \text{minimize } f_3(\mathbf{x}) \; = \; \| \mathbf{a}-\mathbf{x} \|^2 & \\ \text{subject to} \qquad \mathbf{x}^T P_1 \mathbf{x} \leq 1 & \mathbf{x} \in X \end{array} \tag{7}$$

Figure 1 shows the ellipsoid with the covariance matrix P_1 with points $\mathbf{x}$ (inside), $\mathbf{x}_B$ (on the boundary) and $\mathbf{a}$ (outside). $\mathbf{x}_B$ is the boundary point nearest to the point $\mathbf{a}$ outside the ellipsoid. We need to determine the unknown $\mathbf{x}_B$ by minimizing $f_3(\mathbf{x})$.

The novelty of the proposed approach is the application of particle swarms for solving the QCQP problem. Particle swarms are deployed within the ellipsoid to determine $\mathbf{x}_B$. The function $f_3(\mathbf{x}) \; = \; \| \mathbf{a} - \mathbf{x} \|^2$ needs to be evaluated for each particle in the search space. Instead of using $f_3(\mathbf{x}) \; = \; \| \mathbf{a} - \mathbf{x} \|^2$, we use $f_4(\mathbf{x}) \; = \; \| \mathbf{a} - \mathbf{x} \|$ as the modified objective function. The particle with the minimum $f_4(\mathbf{x})$ is considered as the closest to the point $\mathbf{a}$. Only one particle of the swarm is shown in Figure 1 for ease of representation.

The value of the function $f_4(\mathbf{x})$ for the i^{th} particle at the j^{th} iteration is

$$f_4(\mathbf{x}_i^j) \; = \; \| \mathbf{a} \; - \; \mathbf{x}_i^j \| \tag{8}$$

Let the present position of the i^{th} particle be split into the previous position and the velocity vector of the particle using Eq (4).

$$f_4(\mathbf{x}_i^j) \; = \; \| \mathbf{a} \; - \; \mathbf{x}_i^{j-1} \; - \; \mathbf{v}_i^j \| \tag{9}$$

Here, $\mathbf{a}$ is fixed and the particle position $\mathbf{x}_i^{j-1}$ is known and dependent on the velocity vector $\mathbf{v}_i^{j-1}$. So, the possible option for minimizing the function is to change the velocity vector of the particle towards the direction of $(\mathbf{a} - \mathbf{x}_i^{j-1})$. The arrow (for representation purpose) shown in Figure 1 is in the direction of minimizing the function $f_4(\mathbf{x})$. The function $f_3(\mathbf{x})$ is also minimized when minimizing the function $f_4(\mathbf{x})$.

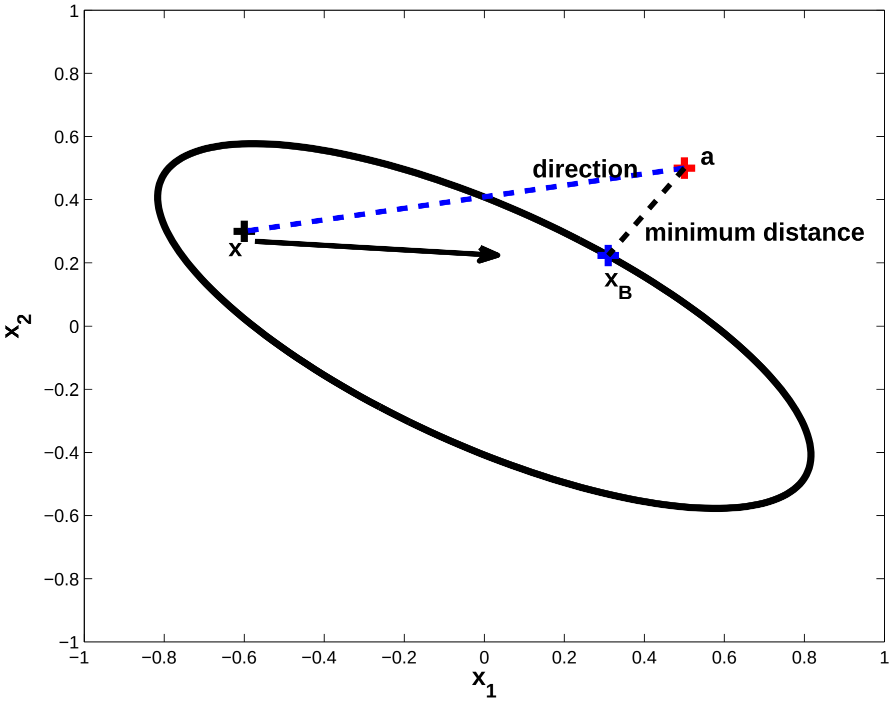

Fig. 1 An ellipsoid with a particle at a point $\mathbf{x}$ inside and a point $\mathbf{x}_B$ on its boundary nearest to the point $\mathbf{a}$ outside it (boundary point on the other ellipsoid). The dotted lines connecting the point $\mathbf{a}$ to the points $\mathbf{x}$ and $\mathbf{x}_B$ are shown. The desired direction of movement from $\mathbf{x}$ is also shown by an arrow, which is required to reach point $\mathbf{x}_B$.

PSO is a population based stochastic optimization technique, which takes several iterations to reach the optimal value. The velocity vector update of the PSO algorithm given by Eq (3) is modified by including the direction term from the function $f_4(\mathbf{x})$. The addition of evaluation function restricts the particle from moving away from the actual course towards the global best position. This addition provides an advantage in computation. However, it also constrains the particle to move in a particular direction. To counter this effect, we add a craziness term in the velocity update equation. The modified velocity update equation is:

$$\mathbf{v}_i^j = w\mathbf{v}_i^{j-1} + c_1 r_1(\mathbf{x}_{bi} - \mathbf{x}_i^{j-1}) + c_2 r_2(\mathbf{x}_{bg} - \mathbf{x}_i^{j-1}) + c_3 r_3(\mathbf{a} - \mathbf{x}_i^{j-1}) + c_4 r_4 \quad (10)$$

where c_3 and c_4 are constants, and r_3 is a random value that is varied during each iteration. The value of c_3 is chosen such that the term $(\mathbf{a} - \mathbf{x}_i^{j-1})$ does not take the particle outside the ellipsoid. r_4 is a random point on the surface of a sphere of dimension n with randomly varying radius. $c_4 r_4$ forms the craziness term.

On the assumption that one end point is known in the shortest path, particle swarms are placed in the search space of the other region and the other end point of the shortest path is determined. In order to evaluate the objective function $f_2(\mathbf{x})$ in Eq (6) for binary classification, we need to determine one end point from region X and the other from region Y. Two sets of particle swarms, one for each region, are placed within the search spaces of the respective regions. The objective function $f_4(\mathbf{x})$ is evaluated based on the particles present in both the regions. In every iteration, the best position of a particle in one region is used as the known end point '**a**' in the shortest path in the velocity update equation (10) of the particles of the other region. The objective function $f_4(\mathbf{x})$ reaches its optimal value after several iterations. Thus, the objective function $f_2(\mathbf{x})$ also reaches its optimal value.

In the process of optimization, some particles may often go out of the search space. To limit the particles within the search space, we inspect after every tentative position update as to whether the particle is lying within the search space by carrying out a check on the QCQP constraints. If the intended new position of a particle is going to violate the constraint, then its position is not updated. In other words, the particles likely to go out of the search space are redeployed back to their previous positions.

The proposed solution may be used in control system problems such as optimization of sensor networks or collaborative data mining, which are based on multiple agents or gradient projection (Zanella et al, 2012). General consensus problem may be solved using the proposed method, where multiple agents need to reach a common goal (Matei and Baras, 2012; Nedi and Ozdaglar, 2009; Nedi et al, 2010). Consensus or distributed optimization is discussed in (Boyd et al, 2010) as 'consensus and sharing' using alternating direction method of multipliers (ADMM). ADMM uses one agent for each constraint. Such solvers are used for solving SVM in distributed format (Forero et al, 2010).

3.3 The pseudo code of the proposed method

Table 2 presents a pseudo code for the algorithm. The parameters w, c_1, c_2, c_3, and c_4 are set to fixed values and the randomly varying parameters r_1, r_2, r_3, and r_4 are updated in each iteration. The position, velocity, personal best, and global best of each particle are stored. The maximum number of iterations for the algorithm is specified as T in the experiments and the size of swarm used in the algorithm is '10'. The proposed algorithm is implemented in MATLAB.

4 Experiments and Results

In this section, the linear and quadratic programming problems with quadratic constraints are solved using the proposed method.

Table 2 Proposed algorithm for QCQP using PSO

```
Inputs: $k = 10$, $t_{max} = T$ and $f_4(\mathbf{x})$; set $w = 0.05$, $c_1 = 0.05$, $c_2 = 0.05$, $c_3 = 0.05$, $c_4 = 0.20$ and initialize parameters $\mathbf{x_i}$, $\mathbf{v_i}$
Outputs: Global best value
  $t = 0$,
  while $t < t_{max}$
    $t \leftarrow t + 1$
    Function evaluation step:
      Calculate the function $f_4(\mathbf{x})$ for $\mathbf{x}_i$
    Velocity update step:
      Randomly choose values for $r_1$, $r_2$, $r_3$, $r_4$ in the range '0' to '1'.
      Then update the velocity of each particle as in Eq (10).
    Position update step:
      Add the updated velocity to the existing position.
      Check for the constraint $\mathbf{x}^T P_1 \mathbf{x} \leq 1$ on all the particles.
      for $m = 1$ to $k$
        if $\mathbf{x}_m^T P_1 \mathbf{x}_m > 1$ then
          $\mathbf{x}_m = \mathbf{x}_{mprev}$
        end if
      end for
  end while
```

4.1 Linear programming with quadratic constraints

A LP problem with quadratic constraints is chosen in order to compare the convergence performance of our method with those of Khachiyan's and Karmarkar's methods. The LP problem is given below:

$$\begin{array}{cc} \max & f_5(x_1, x_2) = x_1 + x_2 \\ \text{subject to} & x_1^2 + x_2^2 \leq 1 \qquad x_1, x_2 \in X \end{array} \tag{11}$$

where X is the region, satisfying the constraint.

This LP problem is reformulated as a QCQP problem:

$$\begin{array}{cc} \max & f_5(\mathbf{x}) = \mathbf{a}^T \mathbf{x} \\ \text{subject to} & \mathbf{x}^T A \mathbf{x} \leq 1 \quad \mathbf{x} \in X \end{array} \tag{12}$$

where vector $\mathbf{a} = [1\ 1]^T$, $\mathbf{x} = [x_1\ x_2]^T$ and A is the identity matrix of size 2 (positive definite).

The solution for this LP problem is x_1, $x_2 = 0.7071$ with $f_5(\mathbf{x}) = 1.4142$. This LP problem is solved using Khachiyan's ellipsoid algorithm, Karmarkar's algorithm and our method. Figure 2 shows the value of the function $f_5(\mathbf{x})$ for each iteration for a typical independent run of the experiment with 50 iterations. The length vector in Karmarkar's method is scaled by a variable δ (Karmarkar, 1984). Two values are used for δ namely 0.05 and 0.50, for the evaluation of the function. As the value of δ increases, the algorithm reaches the optimal value of $f_5(\mathbf{x})$ in less number of iterations. Table 3 shows the error value of function $f_5(\mathbf{x})$ for all the algorithms after the 25^{th} iteration as shown in Figure 2. The error values for the ellipsoid and our methods are less than $10\text{x}10^{-3}$.

In our algorithm, the values of the variables w, c_1, c_2, and c_3 are set to 0.05. These values are small so that the vector addition to position keeps the particles within the region X. Only the value of c_4 is varied since it scales the neighboring

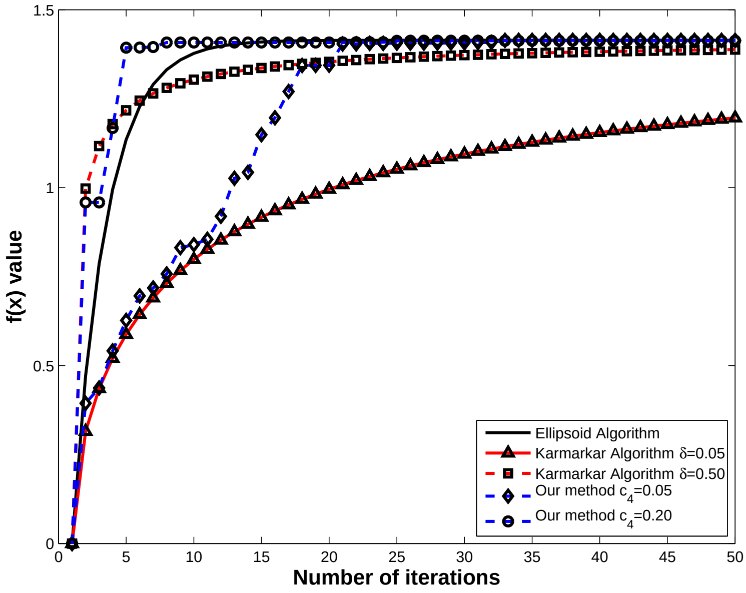

Fig. 2 A plot of evaluated value of $f_5(\mathbf{x})$ against the number of iterations for different algorithms for the LP problem given by Eq (12). At around 25 iterations, all the algorithms except Karmarkar's reach a value close to the solution. We observe that Karmarkar algorithm takes more iterations to reach the solution.

Table 3 Comparison of the error value of function $f_5(\mathbf{x})$ of our algorithm with those of two other methods for the LP problem given by Eq (12) after the 25^{th} iteration.

Algorithm →	Ellipsoid	Karmarkar $\delta = 0.05$	Karmarkar $\delta = 0.50$	Our method $c_4 = 0.05$	Our method $c_4 = 0.20$
Error value →	0.0001	0.3623	0.0494	0.0082	0.0016

search space of a particle. The results for two different values of c_4 are shown in Figure 2. We carried out 50 independent runs for this LP problem with two different values of c_4. The average, minimum, maximum and standard deviation of error value are tabulated in Table 4. The error values for $c_4 = 0.20$ are less than those for $c_4 = 0.05$, indicating that as the neighboring search space is scaled to a higher value, the error value of the function fit becomes better in fewer number of iterations.

Table 4 The minimum, mean, maximum and standard deviation of error values for the function $f_5(\mathbf{x})$ over 50 independent runs of our algorithm for the LP problem defined by Eq (12).

c_4 ↓	Minimum	Mean	Maximum	Standard deviation
0.05	0.0020	0.0197	0.1346	0.0272
0.20	0.0006	0.0182	0.0899	0.0208

4.2 Binary classification of simulated data

Several variants of PSO have been applied for classification problems (Kennedy and Eberhart, 1997). Generally PSO is deployed to learn the rules for classification. A suitable classifier is chosen for classification; for example, a neural network (Kennedy and Eberhart, 1995). The parameters like network weights are tuned using particle swarms. There are discrete versions of swarms, which can take a finite set of values (Cervantes et al, 2005). Here, swarms learn the rule for classifying the test samples. In our method, binary classification is posed as QCQP and swarms are used to solve this QCQP problem. The input feature space is considered as the particle swarm space.

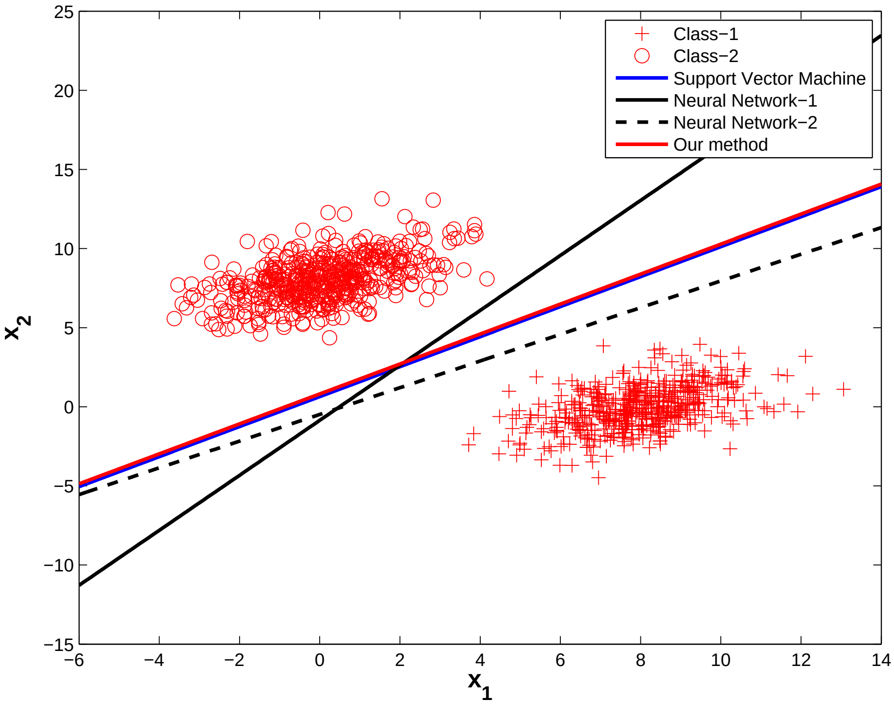

Fig. 3 Hyperplanes obtained by SVM, neural networks and our method are shown for a synthetic dataset for two classes. The hyperplane estimated by our method is closely aligned with that obtained by the SVM with a linear kernel. Other hyperplanes are arrived at by two neural networks (perceptron) and are not optimal.

Figure 3 shows a simulated data set for two classes synthesized using two Gaussian distributions with means $\mu_1 = [8; 0]; \mu_2 = [0; 8]$ and the same covariance matrix $\Sigma_1, \Sigma_2 = [2\ 1; 1\ 2]$. A generic decision boundary for a binary classification problem is a hypersurface. The classes in the data are linearly separable thus reducing a hypersurface to a hyperplane. The equation of a hyperplane for this kind of data is given by,

$$z = w_1 x_1 + w_2 x_2 + w_0 \tag{13}$$

where w_1,w_2 are the weights for the individual features and w_0 is the bias of the hyperplane. This hyperplane equation is used for classification:

$$\begin{array}{l} \text{If } z \geq 0, (x_1, x_2) \in \Omega_1 \\ \text{If } z < 0, (x_1, x_2) \in \Omega_2 \end{array} \tag{14}$$

where Ω_1 and Ω_2 are the regions for the classes 1 and 2, respectively.

The MATLAB programming platform is used to implement and test our method and also a SVM and neural network for comparison. We trained the SVM with a linear kernel and a neural network on this synthetic data. The hyperplanes learnt by the SVM and the neural network are shown in Figure 3. A single layer perceptron with 2 weights, 1 bias and log sigmoid transfer function is used with 100 epochs for training the neural network. SMO algorithm is used for training the SVM with linear kernel (Duda et al, 2001).

We now explain the determination of the optimal hyperplane for binary classification by our method. The values of sample mean and covariance for the two classes are calculated from the simulated samples. Mahalanobis distance (Duda et al, 2001) is determined from the mean value of a class to the data points of the other class and the closest data point of the other class label is found. This closest point is used to fix the reference boundary of the search space (ellipsoid) of the other class. This process is repeated from the other class and the boundary of the first class is also determined. With the boundaries, ellipsoidal regions are formed with estimated means of classes as the centers. This is posed as a QCQP problem formulated in Eq (6) with the estimated covariance matrices normalized to boundary points as P_1 and P_2. Our algorithm is implemented by placing particle swarms near the mean value of the classes and evaluating the optimization function for 300 iterations. The shortest path and the closest point on each boundary are estimated.

$$\begin{array}{ll} \text{minimize} & (\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \\ \text{subject to } \mathbf{x}^T & E[(x-\mu_1)(x-\mu_1)^T] \;\; \mathbf{x} \;\leq\; 1, \quad \mathbf{x} \in X \\ \qquad\quad\;\; \mathbf{y}^T & E[(y-\mu_2)(y-\mu_2)^T] \;\; \mathbf{y} \;\leq\; 1, \quad \mathbf{y} \in Y \end{array} \tag{15}$$

Eq (15) is optimized using the proposed algorithm. The intersection between the two ellipsoidal regions is assumed to be zero to eliminate a situation where particles reach to different solutions. Once the closest points on the boundaries are determined, the perpendicular bisector of the line joining the closest points is the hyperplane. The hyperplane obtained for binary classification is shown in Figure 3. We can observe that the optimal hyperplane calculated by SMO algorithm and our method are closely placed, while other hyperplanes are not optimal. The estimated weight and bias values for each method are tabulated in Table 5. The estimated weight values can be normalized as

$$\begin{array}{l} w_1^1 = w_1 \;/\; \sqrt{w_1^2 + w_2^2} \\ w_2^1 = w_2 \;/\; \sqrt{w_1^2 + w_2^2} \end{array} \tag{16}$$

The normalized weights of SVM and our method are equal and they are $w_1^1 = 0.6875$ and $w_2^1 = -0.7260$.

Table 5 Weights and bias values calculated by SVM with a linear kernel, neural networks (perceptron) and our method for the binary classification problem with synthetic data.

Algorithm	w_1	w_2	w_0
Neural Network-1	-13.2972	7.6540	6.5689
Neural Network-2	-6.2830	7.4473	3.5954
SVM	0.2718	-0.2868	0.1838
Our method	1.6697	-1.7638	1.4376

4.3 Performance on real datasets

Our algorithm is also tested on some of the datasets available from UCI ML repository (UCIMLR, 2014). The datasets used in our experiments are Iris, Wine, Pima, Thyroid, Balance, Bupa, Haberman, and Tae. These eight datasets have been chosen based on the consideration of minimal number of datasets with the maximum coverage of the different types of attributes (namely, binary, categorical value as integer, and real values). Further, the number of classes in each case is 2 or 3. So, the maximum number of hyperplanes to be obtained for any of these datasets is limited to three. The main characteristics of these datasets are tabulated in Table 6. The cross-validation performance on these datasets is compared with those of a SVM with linear and RBF kernel, a neural network, and GSVM. GSVM (Gonzalez-Abril et al, 2014) estimates a classification hyperplane and is developed on the fundamentals of optimizing the SVM problem. GSVM modifies the bias (w_0) obtained from SVM by moving the hyperplane such that the geometric mean of recall and precision is improved.

Table 6 The characteristics of datasets chosen for experimentation from UCI ML repository (UCIMLR, 2014) with varied nature of attributes.

Dataset	No. of samples	No. of Attributes	No. of classes	Nature of attributes
Iris	150	4	3	real
Wine	178	13	3	real & integer
Pima	768	8	2	real & integer
Thyroid	215	5	3	real & binary
Balance	625	4	3	integer
Bupa	345	6	2	real & integer
TAE	151	5	3	integer
Haberman	306	3	2	integer

4.3.1 Iris Dataset

The Iris dataset consists of four different measurements on 150 samples of iris flower. There are 50 samples of three different species of iris forming the dataset. The features, whose values are available from the dataset, are length and width of leaves and petals of different iris plants. Out of the three species, two are not linearly separable from each other, whereas the third is linearly separable from the rest of the species. The classification task is to determine the species of the iris plant, given the 4-dimensional feature vector.

4.3.2 Wine Dataset

The Wine dataset contains the different physical and chemical properties of three different types of wines derived from three different strains of plants. Some of the physical properties such as hue and colour intensity have integer values, whereas chemical properties such as ash or phenol content have real values. The feature vector has 13 dimensions and there are a total of 178 samples. The classification task is to determine the type of wine, given the values of the content of the thirteen physical and chemical components.

4.3.3 Pima Indians Diabetes Dataset

The Pima dataset contains eight different parameters measured from 768 adult females of Pima Indian heritage. Once again, some of them are integer valued, such as age and number of pregnancies. Certain other parameters, such as serum insulin, are real valued. It is a two-class classification problem of identifying normal and diabetic subjects, given the 8-dimensional feature vector as input.

4.3.4 Thyroid Disease Dataset

This dataset contains ten distinct databases of different dimensions. The particular database chosen for our study contains 5 different parameters measured from 215 individuals. Some of the variables have binary values, while others have real values. The classification task is to assign an individual to one of 3 classes, given the 5-dimensional feature vector as input.

4.4 Balance Scale Weight & Distance Dataset

The dataset consists of 625 samples from a modeled psychological experiment. There are four attributes and they are the left weight, the left distance, the right weight, and the right distance. The classification task is to assign the sample into one of 3 classes namely, balance tip to the right, balance tip to the left, and be balanced.

4.5 BUPA liver disorders Dataset

Bupa dataset consists of 345 samples related to liver disorders. There are six attributes in the dataset, where the first 5 variables are all blood test values thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. The last attribute is related to the number of drinks consumed. The classification task is to decide whether a sample belongs to a liver disorder or not.

4.6 Teaching Assistant Evaluation Dataset

This dataset contains the evaluations of 151 teaching assistants (TA) for their teaching performance over three regular semesters and two summer semesters at the Statistics Department of the University of Wisconsin-Madison. The attributes are categorical. The classification task is to assign a test assistant into one of the following three performance categories: low, medium, and high.

4.7 Haberman's Survival Dataset

This dataset consists of 306 patients surviving from the breast cancer disease. The attributes are numerical and they are age, year of operation and the number of positive axillary nodes detected. The classification task is to assign whether the patient survived more than 5 years or not.

4.7.1 Data projection

We perform a preprocessing step on these real datasets. This step is necessary since,

- Some of the attributes of the dataset have larger variance than others. This may result in skewed ellipsoid formation at the mean value of the dataset.
- The number of samples available in the datasets is also less.

To overcome these two problems, we perform the two steps given below.

- Eigen value decomposition is performed on dataset covariance matrix. The dataset is projected on to these Eigen vectors. Scaling is performed in such a way that each component in the new projected dataset has unit variance.
- This step is performed independently on each of the subsets used in the cross-validation stage. We project the subset of samples into two dimensions. First is the direction of the vector joining the sample means of the classes. The equation for this vector is,

$$p = (\mu_1 - \mu_2) \tag{17}$$

where p is the projected vector, μ_1 and μ_2 are the sample means of classes-1 and 2, respectively. The second direction is that of the eigen vector corresponding to the largest eigen value of the covariance matrix of the subset.

The projected samples are used in the estimation of new sample means and covariance matrices. The estimated hyperplane is used to classify the test samples.

4.7.2 Cross-validation

These datasets do not have separate training and test samples; hence we perform cross-validation. In cross-validation, a small subset is used for testing while the remaining are used as training samples. We use ten fold cross-validation: split the datasets into ten subsets and use one of them for testing and the others for training at a time and then rotate the subsets. Equation (15) is used in the estimation of the hyperplane, which in turn is used to classify the test samples. Ten trials are performed and the average cross-validation errors are reported in Table 7.

The performance of our method is close to that of the variants of SVM and is superior to that of the neural network. We notice that in Iris and Wine datasets, the

Table 7 Cross-validation (CV) error (in %) using our method for different datasets from UCI ML repository (UCIMLR, 2014) compared with those reported in (Gonzalez-Abril et al, 2014) and our own implementation of SVM and neural network.

Dataset	CV error by Our method	CV error in SVM with linear kernel	CV error in Neural Network	CV error in SVM with RBF kernel	CV error in GSVM
Iris	**2.20**	3.20	3.60	4.21	3.92
Wine	1.06	1.22	1.56	1.60	**0.93**
Pima	25.15	**23.06**	64.78	24.64	25.85
Thyroid	4.32	3.04	9.20	2.05	**1.73**
Balance	29.03	9.03	32.74	**7.44**	12.52
Bupa	36.76	32.94	42.05	29.56	**28.93**
TAE	42.66	49.33	66.00	**24.20**	29.73
Haberman	27.66	**26.00**	74.00	32.03	32.47

cross-validation error obtained by our technique is better than those achieved by SVM with linear and RBF kernels and also the neural network. In Pima, Balance, Bupa, TAE, and Haberman datasets, the cross-validation error is higher than the best.

5 Conclusion and Future work

We have developed a classification method and optimization algorithm for solving QCQP problems. The novelty in this method is the application of particle swarms, a swarm intelligence technique, for optimization. The results indicate that our approach is a possible method in solving general QCQP problems without gradient estimation. We have shown the results of our algorithm under quadratic constraints by evaluating different optimization functions. The issue with PSO based methods is their computational complexity and the need for parameter tuning. The number of function evaluations linearly increases with the number of particles employed and the number of iterations carried out.

The results show that the new approach proposed gives better solutions for some problems, but it is not always the case. Deeper analysis is needed to understand why the method performs poorly for certain problems and how this can be overcome.

In future, we intend to learn multiple hyperplanes by placing multiple kernels in each class and evaluating the performance against multiple-kernel learning algorithms. The hyperplanes estimated for different kernels may reduce the cross-validation error for the Pima, Balance, Bupa, TAE and Haberman datasets.

References

Bishop C M 1995 Neural Networks for Pattern Recognition. Oxford university press

Bishop C M 2006 Pattern Recognition and Machine Learning. Springer, Berlin

Bomze I M 1998 On Standard Quadratic Optimization Problems. Journal of Global Optimization 13:369–387

Bomze I M, Schachinger W 2010 Multi-Standard Quadratic Optimization: interior point methods and cone programming reformulation. Computation Optimization Applications 45:237–256

Boyd S, Vandenberghe L 2004 Convex Optimization. Cambridge University Press

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J 2010 Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning 3(1):1–122

Cervantes A, Galvan I M, Isasi P 2005 A comparison between the Pittsburgh and Michigan approaches for the binary PSO algorithm. In: Congress on Evolutionary Computation, pp 290–297

Cortes C, Vapnik V 1995 Support-Vector Networks. Machine Learning 20(3):273–297

Dantzig G B 1963 Linear Programming. University Press, Princeton, NJ

Derrac J, García S, Herrera F 2014 Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. Information Sciences 260:98–119

Duda R O, Hart P E, Stork D G 2001 Pattern Classification. Wiley

Fernández A, López V, Galar M, Jesus M J, Herrera F 2013 Analysing the classification of imbalanced data-sets with multiple classes:Binarization techniques and ad-hoc approaches. Knowledge-Based Systems pp 97–110

Forero P A, Cano A, Giannakis G B 2010 Consensus-based distributed support vector machines. Journal of Machine Learning Research 11:1663–1707

Galar M, Fernández A, Barrenechea E, Herrera F 2013 EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recognition

Gonzalez-Abril L, Velasco F, Angulo C, Ortega J A 2013 A study on output normalization in multiclass SVMs. Pattern Recognition Letters pp 344–348

Gonzalez-Abril L, Nuñez H, Angulo C, Velasco F 2014 GSVM : An SVM for handling imbalanced accuracy between classes in bi-classification problems. Applied Soft Computing 17:23–31

Karmarkar N 1984 A new polynomial-time algorithm for linear programming. Combinatorica 4:373–395

Kennedy J, Eberhart R C 1995 Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. IV, pp 1942–1948

Kennedy J, Eberhart R C 1997 A discrete binary version of the particle swarm algorithm. In: Proceedings of the World Multiconference on Systems, Cybernetics and Informatics, pp 4104–4109

Kennedy J, Eberhart R C, Shi Y 2001 Swarm intelligence. Morgan Kaufmann Publishers, San Francisco

Khachiyan L G 1979 A polynomial algorithm in linear programming. Doklady Akademia Nauk SSSR 244:S pp 1093–1096

López V, Fernández A, García S, Palade V, Herrera F 2013 An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics. Information Sciences

Matei I, Baras J S 2012 A performance comparison between two consensus-based distributed optimization algorithms. In: 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems, pp 168–173

Nedi A, Ozdaglar A 2009 Distributed subgradient methods for multi-agent optimization. IEEE Transactions on Automatic Control 54(1):48–61

Nedi A, Ozdaglar A, Parrilo P A 2010 Constrained consensus and optimization in multi-agent networks. IEEE Transactions on Automatic Control 55(4):922–938

Platt J 1998 Fast training of support vector machines using sequential minimal optimization. In: Schoelkopf B, Burges C, Smola A (eds) Advances in Kernel Methods - Support Vector Learning

Spadoni M, Stefanini L 2012 A differential evolution algorithm to deal with box, linear and quadratic-convex constraints for boundary optimization. Journal of Global Optimization 52(1):171–192

UC Irvine Machine Learning Repository 2014 http://archive.ics.uci.edu/ml/

Zanella F, Varagnolo D, Cenedese A, Pillonetto G, Schenato L 2012 Multidimensional Newton-Raphson consensus for distributed convex optimization. In: The 2012 American Control Conference (ACC), pp 1079–1084

Zhan Z-H, Zhang J, Li Y, Chung H S-H 2009 Adaptive particle swarm optimization. IEEE Transactions on Systems, Man, and cybernetics - Part B: Cybernetics 39(6):1362–1381