BOXCQP: An algorithm for convex quadratic programming subject to bound constraints

C. Voglis and I.E. Lagaris*

Dept. of Computer Science, University of Ioannina

P.O. Box 1186, Ioannina 45110 - GREECE

Abstract

A quadratic programming problem with positive definite Hessian and bound constraints is solved, using a Lagrange multiplier approach. The proposed method falls in the category of active set techniques. The algorithm, at each iteration, modifies both the minimization parameters in the primal space and the Lagrange multipliers in the dual space. Comparative results of numerical experiments are reported.

PACS: 02.60.-x; 02.60.Pn; 07.05.Kf; 07.05.-t

*Corresponding author e-mail: lagaris@cs.uoi.gr

Keywords: Convex quadratic programming, optimization, active set, Lagrange multipliers.

1 Introduction

The problem of minimizing a convex quadratic function subject to bound constraints appears quite frequently in applications. For instance, many problems in computational physics and engineering, are reduced to quadratic programming problems. Portfolio management can also be formulated as quadratic programming problem [14]. In the field of Artificial Intelligence, and especially in Support Vector Machines (SVM) an efficient quadratic solver is crucial for the training process [7, 8]. Also methods for calculating the radiation intensity in oncology treatment are formulated as quadratic optimization problems [18]. Finally, many non linear optimization techniques are based on solving quadratic model subproblems [1, 2, 15].

The Quadratic Programming problem with simple bounds is stated as:

$$\min_x \ \frac{1}{2}x^T Bx + x^T d, \text{ subject to: } a_i \leq x_i \leq b_i, \forall i \in I = \{1, 2, \cdots, N\} \quad (1)$$

where $x, d \in R^N$ and B a symmetric, positive definite $N \times N$ matrix.

For the problem in Eq. (1) two major strategies exist in the literature,

both of which require feasible steps to be taken. The first one is the Active Set strategy [1, 2], which generates iterates on a face of the feasible box until either a minimizer of the objective function is found or a point on the boundary of that face is reached. The basic disadvantage of this approach, especially in the large-scale case, is that constraints are added or removed one at a time, thus requiring a number of iterations proportional to the problem size. To overcome this, gradient projection methods [5, 6] were proposed. In that framework the active set algorithm is allowed to add or remove many constraints per iteration.

The second strategy consists in treating the inequality constraints using interior point algorithms. In brief, an interior point algorithm consists of a series of parametrized barrier functions which are minimized using Newton's method. The major computational cost is due to the solution of a linear system, which provides a feasible search direction.

Our proposed approach to solve the problem of Eq. (1) is an active set algorithm, which does not guarantee strictly descent iterations. In this paper we investigate a series of convex quadratic test problems. We recognize that bound constraints are a very special case of linear inequalities, which may in general have the form $Ax \geq b$, A being an mxn matrix and b is a vector

$\in R^m$. Our investigation is also motivated by the fact that in the convex case, every problem subject to inequality constraints can be transformed to a bound constrained one, using duality, i.e. the problem:

$$\min_{x \in R^n} \ \frac{1}{2} x^T B x + x^T d \tag{2}$$
$$\text{subject to: } Ax \geq b$$

is equivalent to the dual:

$$\max_{y \in R^m} \ -\frac{1}{2} y^T \tilde{B} y + y^T \tilde{d} \tag{3}$$
$$\text{subject to: } y \geq 0$$

where $\tilde{B} = AB^{-1}A^T$ a positive definite matrix and $\tilde{d} = AB^{-1}d + b$. The dual problem in Eq. (3) is also a quadratic problem subject to bounds. Let y^* be the solution of the dual problem. We can then obtain the solution to the initial problem of Eq. (2) as:

$$x^* = B^{-1}(A^T y^* - d) \tag{4}$$

The paper is organized as follows. The proposed algorithm is described in detail in Section 2. In Section 3 we briefly present four quadratic programming codes that were used against our method on five different problem

types, that are described in Section 4. Finally, in Section 5 a new trust region like method is proposed which takes full advantage of our quadratic programming algorithm.

2 Solving the quadratic problem

For the problem in Eq. (1), we construct the associated Lagrangian:

$$L(x, \lambda, \mu) = \frac{1}{2} x^T B x + x^T d - \lambda^T (x - a) - \mu^T (b - x) \tag{5}$$

The KKT necessary conditions at the minimum x^*, λ^*, μ^* require that:

$$\begin{aligned} Bx^* + d - \lambda^* + \mu^* &= 0 \\ \lambda_i^* \geq 0, \quad \mu_i^* &\geq 0, \ \forall i \in I \\ \lambda_i^* (x_i^* - a_i) &= 0, \ \forall i \in I \\ \mu_i^* (b_i - x_i^*) &= 0, \ \forall i \in I \\ x_i^* &\in [a_i, b_i], \ \forall i \in I \end{aligned} \tag{6}$$

A solution to the above system (6), can be obtained through an active set strategy described in detail in Algorithm 1:

Algorithm 1 BOXCQP

Initially set: $k = 0$, $\lambda^{(0)} = \mu^{(0)} = 0$ and $x^{(0)} = -B^{-1}d$.

If $x^{(0)}$ is feasible, **Stop**, the solution is: $x^* = x^{(0)}$.

At iteration k, the quantities $x^{(k)}, \lambda^{(k)}, \mu^{(k)}$ are available.

1. Define the sets:

$$\begin{aligned} L^{(k)} &= \{i : x_i^{(k)} < a_i, \text{ or } x_i^{(k)} = a_i \text{ and } \lambda_i^{(k)} \geq 0\} \\ U^{(k)} &= \{i : x_i^{(k)} > b_i, \text{ or } x_i^{(k)} = b_i \text{ and } \mu_i^{(k)} \geq 0\} \\ S^{(k)} &= \{i : a_i < x_i^{(k)} < b_i, \text{ or } x_i^{(k)} = a_i \text{ and } \lambda_i^{(k)} < 0, \\ &\text{or } x_i^{(k)} = b_i \text{ and } \mu_i^{(k)} < 0\} \end{aligned}$$

   Note that $L^{(k)} \cup U^{(k)} \cup S^{(k)} = I$

2. Set:

$$\begin{aligned} x_i^{(k+1)} &= a_i, \ \mu_i^{(k+1)} = 0, \quad \forall i \in L^{(k)} \\ x_i^{(k+1)} &= b_i, \ \lambda_i^{(k+1)} = 0, \quad \forall i \in U^{(k)} \\ \lambda_i^{(k+1)} &= 0, \ \mu_i^{(k+1)} = 0, \quad \forall i \in S^{(k)} \end{aligned}$$

3. Solve:

$$Bx^{(k+1)} + d = \lambda^{(k+1)} - \mu^{(k+1)}$$

for the N unknowns:

$$x_i^{(k+1)},\ \forall i\ \in S^{(k)}$$

$$\mu_i^{(k+1)},\ \forall i\ \in U^{(k)}$$

$$\lambda_i^{(k+1)},\ \forall i\ \in L^{(k)}$$

4. Check if the new point is a solution and decide to either stop or iterate.

If $(x_i^{(k+1)} \in [a_i, b_i]\ \forall i \in S^{(k)}$ **and** $\mu_i^{(k+1)} \geq 0,\ \forall i \in U^{(k)}$

and $\lambda_i^{(k+1)} \geq 0,\ \forall i \in L^{(k)})$ **Then**

Stop, the solution is: $x^* = x^{(k+1)}$.

Else

set $k \leftarrow k+1$ and iterate from **Step 1.**

Endif

The solution of the linear system in Step 3 of Algorithm 2, needs further consideration. Let us rewrite the system in a componentwise fashion.

$$\sum_{j \in I} B_{ij} x_j^{(k+1)} + d_i = \lambda_i^{(k+1)} - \mu_i^{(k+1)},\ \forall i \in I \tag{7}$$

Since $\forall i \in S^{(k)}$ we have that $\lambda_i^{(k+1)} = \mu_i^{(k+1)} = 0$, hence we can calculate $x_i^{(k+1)},\ \forall i \in S^{(k)}$ by splitting the sum in Eq. (7) and taking into account

Step 2 of the algorithm, i.e.:

$$\sum_{j \in S^{(k)}} B_{ij} x_j^{(k+1)} = - \sum_{j \in L^{(k)}} B_{ij} a_j - \sum_{j \in U^{(k)}} B_{ij} b_j - d_i, \ \forall i \in S^{(k)} \tag{8}$$

The submatrix B_{ij}, with $i, j \in S^{(k)}$ is positive definite as can be readily verified, given that the full matrix B is. The calculation of $\lambda_i^{(k+1)}, \ \forall i \in L^{(k)}$ and of $\mu_i^{(k+1)}, \ \forall i \in U^{(k)}$ is straightforward and is given by:

$$\lambda_i^{(k+1)} = \sum_{j \in I} B_{ij} x_j^{(k+1)} + d_i, \ \forall i \in L^{(k)} \tag{9}$$

$$\mu_i^{(k+1)} = - \sum_{j \in I} B_{ij} x_j^{(k+1)} - d_i, \ \forall i \in U^{(k)} \tag{10}$$

The main computational task of the algorithm above, is the solution of the linear system in Step 3. The size of the system may vary according to the size of the active set in each iteration. In our implementation we solve the linear system using either the conjugate gradient method (Variant 1) or a direct solver via LDL^T decomposition (Variant 2). For large scale problems the conjugate gradient method may be preferable.

3 Other quadratic codes

3.1 QPBOX

QPBOX [11] is a Fortran77 package for box constrained quadratic programs developed in IMM in Technical University of Denmark. The bound constrained quadratic program is solved via a dual problem, which is the minimization of an unbounded, piecewise quadratic function. The dual problem involves a lower bound of λ_1, i.e the smallest eigenvalue of a symmetric, positive matrix, and is solved by Newton iteration with line search.

3.2 QLD

This code [10] is due to K.Schittkowski of the University of Bayreuth, Germany and is a modification of routines due to MJD Powell at the University of Cambridge. It is essentially an active set, interior point method and supports general linear constraints too.

3.3 DUALQP

This method was originally proposed by Goldfarb and Idnani [12] for positive definite quadratic programming problems. It takes advantage of the fact that

the unconstrained minimum of the objective function can be used as a starting point (notice that we choose the same starting point). DUALQP can be applied to problems with general linear constraints also. The implementation is due to P. Spellucci of the Technical University of Darmstadt [17].

3.4 QUACAN

This algorithm combines conjugate gradients with gradient projection techniques, as the algorithm of Moré and Toraldo [4]. A new strategy for the decision of leaving the current face is introduced, that makes possible to obtain finite convergence even for a singular Hessian and in the presence of dual degeneracy. QUACAN [16] is specialized for convex problems subject to simple bounds.

4 Experimental results

To verify the effectiveness of the proposed approach we used five different problem types, and measured cpu times to make a comparison possible. We have implemented BOXCQP in Fortan 77 and used an Intel Pentium–4 processor with Linux operating system.

In the subsections below we describe in brief the different test problems used for the experiments, and report our results.

4.1 Random problems

The first set of experiments includes randomly generated problems. We create random positive–definite B matrices, random bounds a and b and we choose d so that the unconstrained minimum falls outside the box. (See Table 1).

4.2 Circus Tent problem

The circus tent problem is taken from Matlab's optimization demo as an example of large-scale quadratic programming with simple bounds. The problem is to build a *circus tent* to cover a square lot. The tent is elastic and is to be supported by five poles. The question is to find the shape of the tent at equilibrium, that corresponds to the minimum of the energy function. As we can see in Figure 1, the problem has only lower bounds imposed by the five poles and the ground.

The surface formed by the elastic tent, is determined by solving the bound constrained optimization problem:

$$\min_x f(x) = \frac{1}{2}x^T Hx + x^T c \tag{11}$$
$$\text{subject to: } l \leq x$$

where $f(x)$ corresponds to the energy function and H is a 5-point finite difference Laplacian over a square grid.

4.3 Biharmonic Equation problem

We consider the problem of describing small vertical deformations of an horizontal, elastic membrane clamped on a rectangular boundary, under the influence of a vertical force. The membrane is constrained to remain below an obstacle. For an in depth discussion of this problem see [21]. The formulation of the problem is given by Eq.(12).

$$\min_u \frac{1}{2}u^T Qu + u^T f \tag{12}$$
$$\text{subject to: } u \leq \psi$$

We see an example in Fig 2 of a membrane under the influence of a vertical force.

4.4 Intensity Modulated Radiation Therapy

This problem arises in the field of radiotherapy and concerns the determination of the spatial distribution of the radiation, in a way that the patient's vital organs are minimally irradiated. Knowing the beam settings and the intensity profile, one can calculate the radiation dose. Inversely, when a desired dose is required, the proper intensity profile for given beam settings can be retrieved by solving a quadratic problem. The beam settings are successively modified in an effort to satisfy a set of clinical constraints, and hence the quadratic subproblem (shown in Eq. (13)), must be solved a large number of times [18].

$$\min_f s(f) = \frac{1}{2} f^T A f + f^T b \qquad (13)$$

$$\text{subject to } f \geq 0$$

The results reported in Table 4, correspond to real world data, kindly provided by S. Breedveld [19]. In this example, seven beams are combined resulting to a quadratic problem with 2342 parameters.

4.5 Support Vector Classification

In this classification problem, the goal is to separate two classes using a hyperplane $f(x) = w^T x + b$, which is determined from available examples ($D = \{(x^1, y^1), (x^2, y^2), \ldots (x^l, y^l)\}$, $x \in R^n$, $y \in -1, 1$). Furthermore it is desirable to produce a classifier that will work well on unseen examples, i.e. it will generalize well. Consider the example in Fig. 3. There are many possible linear classifiers that can separate the data, but there is only one that maximizes the distance to the nearest data point of each class. This classifier is termed the optimal separating hyperplane and intuitively, one would expect that generalizes optimally.

The formulation of the maximum distance linear classifier (if we omit the constant term b of the hyperplane equation[1]) is a convex quadratic problem with simple bounds on the variables. The resulting problem has the form:

$$\min_a \frac{1}{2} a^T Q a - a^T e \tag{14}$$

$$\text{subject to: } 0 \leq a_i \leq C$$

where $e \in R^l$ and with $e_i = 1$, $Q_{ij} = y^i y^j K(x^i, x^j)$ and $K(x, y)$ is the kernel function performing the non-linear mapping into the feature space.

[1] Also known as explicit bias.

The parameters $a \in R^l$ are Lagrange multipliers of an original quadratic problem, that define the separating hyperplane using the relation:

$$w^{*T}x = \sum_{i=1}^{l} a_i^* y^i K(x^i, x) \tag{15}$$

Hence the separating surface is given by:

$$f(x) = \text{sgn}(w^{*T}x) \tag{16}$$

In our experiments we used the CLOUDS [13] data set, which is a two-dimensional data set with two classes. We have constructed the problem in Eq. (15) using an RBF Kernel function $K(x, y) = exp(\frac{-||x-y||^2}{2^2})$, and setting $C = 100$. The experiments conducted follow the procedure:

- Form the training set by extracting l examples from the dataset and let the rest examples (5000-l) form the test set.
- Construct the matrix Q for the problem in Eq. (15)
- Apply each solver, obtain the corresponding separating surface and test-set error.

In these experiments the large condition number of matrix Q leads to ill conditioned problems. To circumvent this, we added in the main diagonal of Q a small positive term. The resulting classification surfaces for $l = 200$,

500, 1000 and 2000 training examples from CLOUDS dataset are shown in Fig. 4.

5 Bound Constrained Nonlinear Optimization

We present here a trust region method for non-linear optimization with bound constraints, where the trust region is a hyperbox, in contrast with the usual hypersphere or hyperellipsoid shapes. The rectangular trust region is natural for problems with bound constraints, because even when it overlaps with the feasible region, its geometry is preserved. Trust region methods fall in the category of sequential quadratic programming. These algorithms are iterative and the objective function $f(x)$ (assumed to be twice continuously differentiable), is approximated in a proper neighborhood of the current iterate (the trust region), by a quadratic model. Namely, at the k^{th} iteration the model is given by:

$$f(x^k + s) \approx m^{(k)}(s) = f(x^{(k)}) + s^T g^{(k)} + \frac{1}{2} s^T B^{(k)} s \tag{17}$$

where $g^{(k)} = \nabla f(x^{(k)})$ and $B^{(k)}$ in the case of Newton's method is a positive definite modification of the Hessian, while in the case of quasi-Newton methods is a positive definite matrix produced by the relevant update.

The trust region may be defined by:

$$\mathbf{T}^{(k)} = \{x \in \Re^n \mid ||x - x^{(k)}|| \leq \Delta^{(k)}\} \tag{18}$$

It is obvious that different choices for the norm lead to different trust region shapes. The Euclidean norm $||\cdot||_2$, corresponds to a hypershpere, while the $||\cdot||_\infty$ norm defines a hyperbox.

Given the model and the trust region, we seek a step $||s^{(k)}|| \leq \Delta^{(k)}$, that minimizes $m^{(k)}(s)$. We compare the actual reduction $\delta f^{(k)} = f(x^{(k)}) - f(x^{(k)} + s^{(k)})$, to the model reduction $\delta m^{(k)} = m^{(k)}(0) - m^{(k)}(s^{(k)})$. If they agree to a certain extend, the step is accepted and the trust region is either expanded or remains the same. Otherwise the step is rejected and the trust region is contracted. The basic trust region algorithm is sketched in Algorithm 2.

Algorithm 2 Basic trust region

---

1. Pick the initial point and trust region parameter $x^{(0)}$ and $\Delta^{(0)}$, and set $k = 0$.

2. Construct a quadratic model:

$$f(x^k + s) \approx m^{(k)}(s) = f(x^{(k)}) + s^T g^{(k)} + \frac{1}{2} s^T B^{(k)} s$$

3. Minimize $m^{(k)}(s)$ and hence determine $||s^{(k)}|| \leq \Delta^{(k)}$

4. Compute the ratio of actual to expected reduction: $r^{(k)} = \frac{\delta f^{(k)}}{\delta m^{(k)}}$, and update the trust region, following the strategy of Dennis & Schnabel [20] (Appendix A, page 338).

5. Increment $k \leftarrow k + 1$ and repeat from 1.

Consider the bound constrained problem:

$$\min_x f(x), \quad \text{subject to: } l_i \leq x_i \leq u_i \tag{19}$$

(The unconstrained case is obtained by letting $u_i = -l_i \to \infty$.)

Let $x^{(k)}$ be the k-th iterate of the trust region algorithm.

Hence step 3 of Algorithm 2 becomes:

$$\min_s \; m^{(k)}(s) = s^T g^{(k)} + \frac{1}{2} s^T B^{(k)} s \tag{20}$$

$$\text{subject to: } \; \max(l_i - x_i^{(k)}, -\Delta^{(k)}) \leq s_i \leq \min(u_i - x_i^{(k)}, \Delta^{(k)})$$

In the unconstrained case, our experiments (that used a BFGS update), showed similar performance to spherical trust region implementations. For the bound constrained case our method is obviously superior, since it maintains the simplicity of the rectangular trust region, where our efficient quadratic solver is applicable [15]. Note that the trust region formed by the intersection of a sphere with the rectangular box defined by the parameter bounds,

is not easy to treat. Concluding, further numerical tests showed that our method performs similarly to active set methods with line–search.

6 Conclusions

We have presented an active set algorithm for solving bound constrained convex quadratic problems, using a Lagrange multiplier approach that modifies, at each iteration, both the primal and the dual variables. Extensive experimental testing showed significant improvement over several known quadratic programming codes. In addition a trust region method for nonlinear objectives has emerged, that takes advantage of the proposed algorithm in order to efficiently solve unconstrained and bound constrained problems.

References

[1] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley &Sons, Inc., New York (1987).

[2] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, New York (1981).

[3] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *Testing a class of methods for solving minimization problems with simple bounds on the variables*, Mathematics of Computation, **50**, pp. 399-430, (1988).

[4] J.J. More', G. Toraldo, On the solution of quadratic programming problems with bound constraints - SIAM J. on Opt. 1, (1991), pp.93-113.

[5] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, Springer Series in Computational Mathematics**17**, Springer–Verlag (1992).

[6] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific (1996)

[7] S. Gunn, *Support vector machines for classification and regression*, ISIS technical report, Image Speech & Intelligent Systems Group, University of Southampton (1997).

[8] E. Osuna, R. Freund, and F. Girosi, *Support vector machines: Training and applications*, A.I. Memo (in press) 1602, MIT A. I. Lab. (1997).

[9] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright, *Inertia-controlling methods for general quadratic programming* , SIAM Rev. 33 (1991), pp. 1–36

[10] K. Schittkowski, *QLD: A FORTRAN Code for Quadratic Programming, Userős Guide*, Mathematisches Institut, Universität Bayreuth, Germany (1986).

[11] K. Madsen, H. B. Nielsen, and M.C. Pinar, *Bound Constrained Quadratic Programming via Piecewise Quadratic Functions*, Mathematical Programming **85** (1) (1999)

[12] D. Goldfarb and A. Idnani, *A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs*, Mathematical Programming, **27** (1983) 1–33.

[13] C. L. Blake, and C. J. Merz, *UCI Repository of machine learning databases*, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Irvine, University of California, Department of Information and Computer Science (1998).

[14] A. F. Perold, *Large-Scale Portfolio Optimization*, Management Science **30**, 1143-1160 (1984).

[15] C. Voglis and I. E. Lagaris, *A Rectangular Trust-Region Approach for Unconstrained and Box-Constrained Optimization Problems*, Interna-

tional Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004), Athens, (2004)

[16] A. Friedlander and M. Martinez, On the maximization of a concave quadratic function with box constraints, SIAM J. Optimization 4, pp. 177-192 (1994).

[17] `http://www.mathematik.tu-darmstadt.de:8080/ags/ag8/Mitglieder/spellucci_de.html`

[18] S. Breedveld, P. R. M. Storchi, M. Keijzer and B. J. M. Heijmen *Fast, multiple optimizations of quadratic dose objective functions in IMRT*

[19] S. Breedveld, private communication.

[20] J. E. Dennis, R. B. Schnabel *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM (1996).

[21] K. Kunisch, and F. Rendl, *An Infeasible Active Set Method For Quadratic Problems With Simple Bounds*, SIAM J. Optimization 14(1), pp. 35-52 (2003)

Prob. Size	BOXCQP	QPBOX	QLD	DUALQP	QUACAN
100	0.41	1.22	0.61	0.91	1.33
200	0.48	1.53	0.83	1.06	1.63
300	0.67	1.74	0.91	1.12	2.05
500	0.8	1.93	1.32	1.67	2.22
1000	1.1	2.76	2.12	2.33	3.11
2000	1.8	3.17	2.37	3.13	4.12

Table 1: CPU times (secs) for the Random test problems.

Prob. Size	BOXCQP	QPBOX	QLD	DUALQP	QUACAN
400	0.02	0.48	0.20	0.38	0.05
625	0.03	0.69	0.79	0.66	0.12
900	0.04	0.77	2.30	1.11	0.35
1225	0.05	0.93	5.57	1.61	0.69
1600	0.07	1.21	6.32	1.87	0.74
2025	0.10	1.56	8.11	2.16	0.89
2500	0.12	1.78	9.12	2.51	1.02
3600	0.17	2.01	11.89	3.24	1.23

Table 2: CPU times (secs) for the Circus Tent problem.

Prob. Size	BOXCQP	QPBOX	QLD	DUALQP	QUACAN
400	0.05	0.07	0.56	0.38	0.18
625	0.42	0.65	1.13	1.71	1.12
900	0.81	1.07	1.29	2.23	1.65
1225	1.37	2.47	3.01	3.11	2.10
1600	2.18	2.98	3.87	4.21	2.76
2025	2.67	3.34	4.42	4.88	3.64
2500	3.52	4.07	5.10	5.43	4.41
3600	4.67	5.33	6.02	6.23	5.53

Table 3: CPU times (secs) for the Biharmonic Equation problem.

Prob. Size	BOXCQP	QPBOX	QLD	DUALQP	QUACAN
2342	1.91	2.12	1.78	1.56	1.67

Table 4: CPU times (secs) for the IMRT problem.

Prob. Size	BOXCQP	QPBOX	QLD	DUALQP	QUACAN
200	0.34	0.51	0.04	0.08	0.12
500	2.23	2.55	4.22	3.21	7.86
1000	3.45	4.12	10.81	9.76	10.12
2000	15.66	18.63	20.51	29.01	25.11
3000	24.23	29.91	33.62	52.11	31.49
4000	36.77	43.63	50.04	69.94	45.22

Table 5: CPU times (secs) for the SVM training problem.

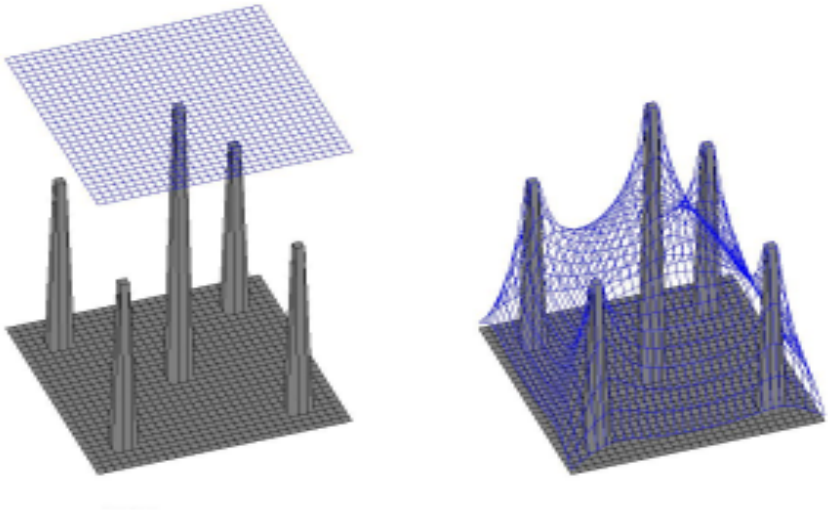


Figure 1: Circus tent problem

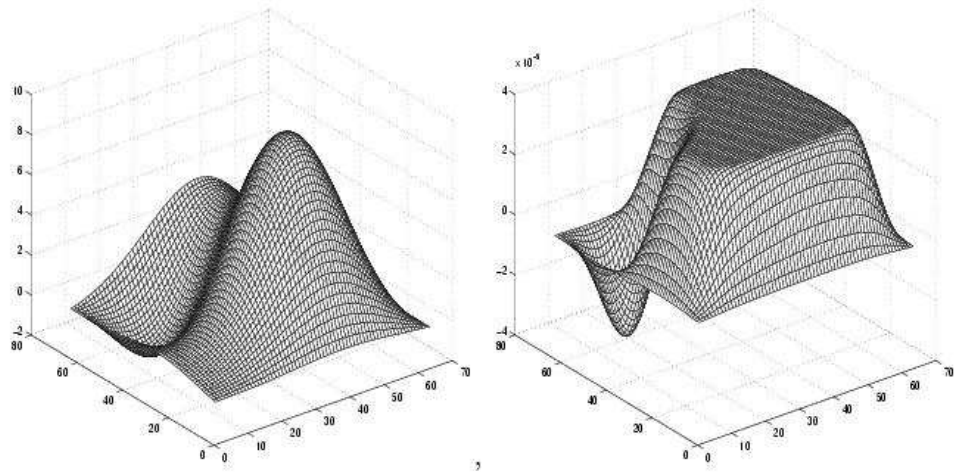


Figure 2: The force is presented on the left, and the deformation on the right

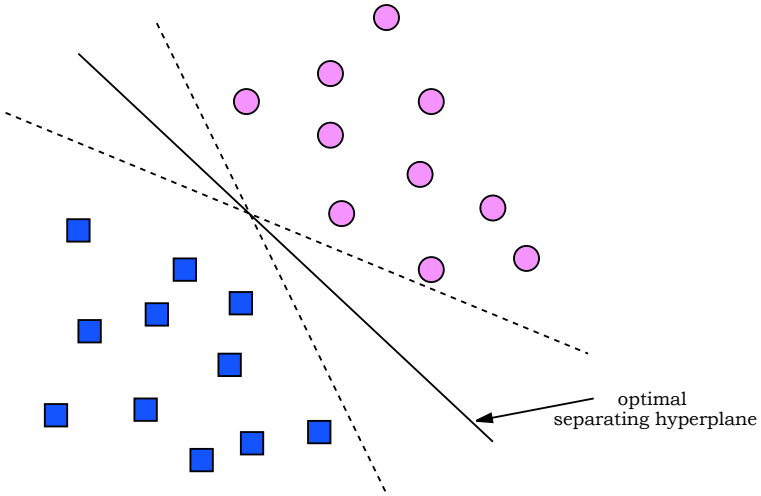
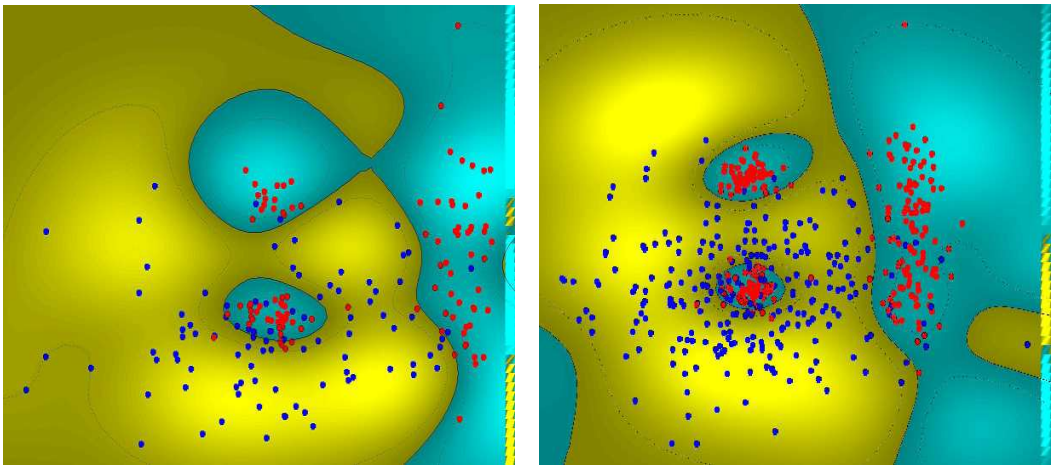
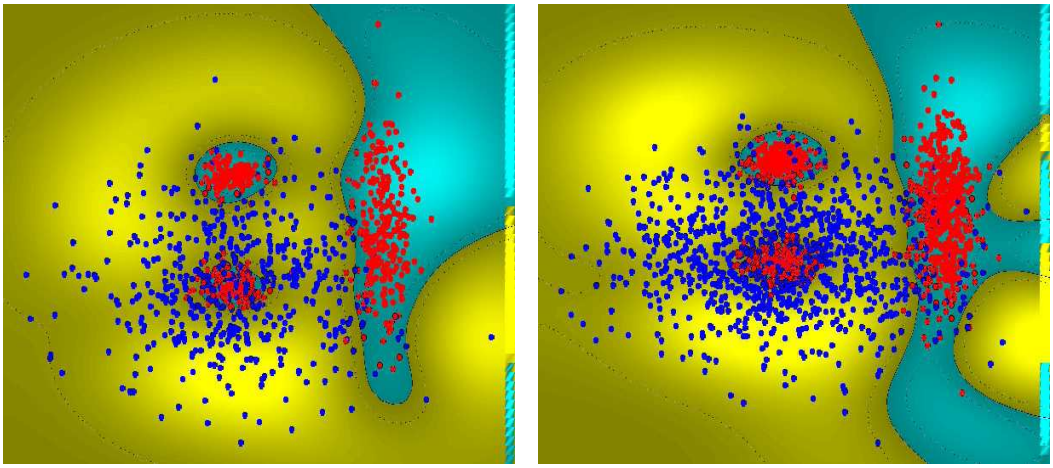


Figure 3: Maximum distance classifier



(a) 200 Training Examples

(b) 500 Training Examples



(c) 1000 Training Examples

(d) 2000 Training Examples

Figure 4: SVM classification surfaces