1^{st} International Conference "From Scientific Computing to Computational Engineering"
1^{st} IC-SCCE
Athens, 8-10 September 2004
©IC-SCCE

BOXCQP: AN ALGORITHM FOR BOUND CONSTRAINED CONVEX QUADRATIC PROBLEMS

C. Voglis, and I.E. Lagaris

Department of Computer Science
University of Ioannina
P.O. BOX 1186 - GR 45110 Ioannina, Greece
email: voglis@cs.uoi.gr, lagaris@cs.uoi.gr

Keywords: Quadratic programming, active set, exterior point, Lagrange multipliers

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 exterior point, active set techniques. An iteration of our algorithm modifies both the minimization parameters in the primal space and the Lagrange multipliers in the dual space. Comparative results of numerical experiments are also reported.

1 INTRODUCTION

Consider the Quadratic Programming problem:

$$\min_{x} \frac{1}{2}x^T Bx + x^T d, \text{ subject to: } a_i \leq x_i \leq b_i, \forall i \in I \tag{1}$$

where $x, d \in R^N$ and B a symmetric, positive definite $N \times N$ matrix and $I = \{1, 2, \cdots, N\}$.

The problem of minimizing a convex quadratic function subject to bound constraints appears frequently in applications. For instance, many Computational Physics and Engineering problems, are reduced to quadratic programming problems. Portfolio management can also be formulated as quadratic programming problem[18]. In the field of Artificial Intelligence, and especially in Support Vector Machines an efficient quadratic solver is crucial for the training process[6, 7]. Finally many non linear optimization techniques are based on solving quadratic subproblems[3].

So far two major strategies exist in the literature, both of which require feasible steps to be carried out. 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 on that face or a point on the boundary of that face is reached. Since the function values are strictly decreasing and the iterates are feasible, finite convergence can be proved. The basic disadvantages of the classical approach, especially in the large-scale case, is that constraints are added or removed one at a time, requiring so a number of iterations proportional to the problem size. To overcome this, gradient projection methods[4, 5] were proposed. In that framework the active set algorithm is allowed to add or remove many constraints per iteration.

The other solution strategy consists in treating the inequality constraints using interior point algorithms [14, 13, 15]. In brief, an interior point algorithm consists of a series of parameterized barrier functions which are minimized using Newton's method. The major computational cost is due to the solution of the Newton system, which provides a feasible search direction.

Our proposal to solve the problem of Eq. (1) is an exterior point 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. The problem:

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

is equivalent to the dual:

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

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 Eq.(2) as:

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

2 BOUND CONSTRAINED QP

Consider the problem in Eq. (1). We construct the associated Lagrangian:

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

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

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

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:

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

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

2. Set:

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

3. Solve:

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

   for the N unknowns:

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

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 above, 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 $\lambda_i^{(k+1)} = \mu_i^{(k+1)} = 0$, 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:

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

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

3 EXPERIMENTAL RESULTS

In order to explore the practical behaviour of the proposed algorithm, we have contacted four different types of minimization tasks: random problems, the circus tent problem,the biharmonic equation problem, and the support vector machine training problem.

The testing platform used was Matlab version 6.5, where our algorithm was implemented. The other quadratic programming algorithms were:

MOSEK-QP: Mosek[8] is commercial product that among other algorithms provide a quadratic programming code and has an easy to use Matlab interface.

QUADPROG: Quadprog is Matlab' s quadratic programming solver.

MINQ: Minq[9] is an implementation of a quadratic programming algorithm subject to simple bounds.

QPOPT: Qpopt[10] is a commercial software included in TOMLAB[19] optimization environment.

QLD: Qld[12] is a free software included in TOMLAB optimization environment.

SQOPT: Sqopt[11] is a commercial software included in TOMLAB optimization environment.

From the algorithms tested, only QUADPROG and MINQ were written in Matlab, while all the others were precompiled dll's with a Matlab interface. For each method the total CPU time is measured. Due to the interpreter nature of Matlab environment, a time comparison is not totally fair for our implementation, although it performed better in the majority of the experiments.

In the subsections below a brief description of the problem is made and the computational results are reported.

3.1 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 in a way so that the solution is outside the box. In Table 1, we report

Dimension	BOXCQP	MOSEK	QUADPROG	MINQ	QPOPT	QLD	SQOPT
100	0.036333	0.365000	0.286333	0.067667	0.020667	0.015667	0.015333
200	0.026333	0.411667	1.390667	0.390333	0.047000	0.046667	0.114333
300	0.052000	0.448000	2.255333	1.296667	0.213667	0.208333	0.406333
400	0.099000	0.651000	3.963333	2.729333	0.515667	0.547000	0.518667
500	0.161667	0.781000	6.109333	5.463333	1.031667	1.099000	1.354000
600	0.281667	1.197667	8.786333	9.359333	1.791667	2.041667	2.114667
700	0.364667	1.745000	12.078000	15.609333	2.781000	2.838667	2.630000
800	0.531333	2.390333	15.625333	22.437000	4.166667	9.354333	3.942667
900	0.651000	2.870000	20.500000	31.739667	5.854333	6.083333	4.510333
1000	0.843333	3.750000	25.041667	44.682667	8.036667	10.515667	6.068000
1100	1.130000	4.542000	30.640667	58.838667	10.562333	11.140333	6.849000
1200	1.390667	5.630000	37.646000	76.765333	13.135667	24.172000	9.062333
1300	1.719000	7.411000	42.849000	99.542000	16.182000	18.197667	9.391000
1400	2.318333	9.640333	63.895667	128.744333	19.833000	33.245000	12.083333
1500	2.781667	11.713667	77.765667	176.025667	23.739333	27.812667	13.656000

Table 1: CPU times (in secs): Random Tests

the average CPU times in seconds. The average is taken over ten runs for problems of the same dimension. The generated matrices B, were constructed so as to have a condition number around 10^3.

3.2 Circus Tent: An energy minimization application

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.

As we can see in Figure 1, the problem has only lower bounds imposed by the five poles and the ground.

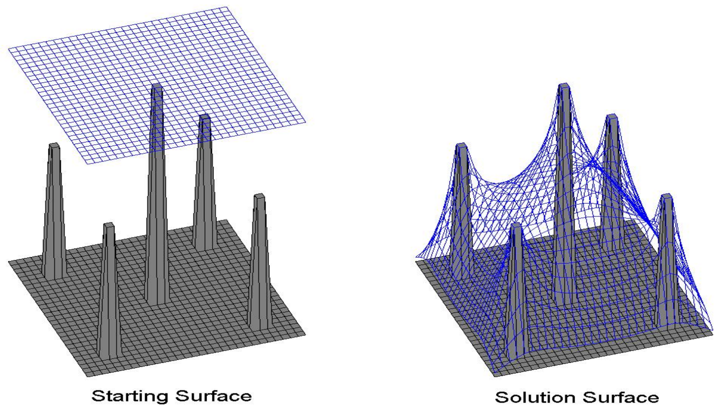


Figure 1: Circus tent problem

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

$$\min_x f(x) = \frac{1}{2} x^T H x + x^T c \quad (11)$$
$$\text{subject to: } l \le x$$

where $f(x)$ is the discrete approximant to the energy function and H is a 5-point finite difference Laplacian over a square grid. The results for different grid sizes are shown in Table 2.

3.2 Biharmonic Equation: An application from mathematical physics

We consider the problem of describing small vertical deformations of an horizontal, elastic membrane clamped on

Dimension	BOXCQP	MOSEK	QUADPROG	MINQ	QPOPT	QLD	SQOPT
25	0.021000	0.661000	0.104333	0.025667	0.005333	0.000000	0.010667
100	0.010667	0.364667	0.135333	0.099000	0.010333	0.010000	0.015667
225	0.010333	0.401000	0.271000	0.328000	0.005333	0.041667	0.021000
400	0.026000	0.489333	0.380333	0.979000	0.052000	0.203000	-
625	0.036000	0.693000	0.666667	2.589000	0.125000	0.791667	-
900	0.047000	0.474000	1.119667	6.281667	0.359000	2.307333	-
1225	0.052000	0.578000	1.614667	12.494667	0.693000	5.573000	-

Table 2: CPU times (in secs): Circus Tent

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[16].

The formulation of the problem is given by Eq.(12).

$$\min_u \frac{1}{2}u^T Qu + u^T f \qquad (12)$$
$$\text{subject to: } u \le \psi$$

Dimension	BOXCQP	MOSEK	QUADPROG	MINQ	QPOPT	QLD	SQOPT
25	0.021000	0.250000	0.119333	0.010667	0.010333	0.005000	0.000000
100	0.062667	0.177000	0.208333	0.072667	0.005333	0.015667	0.005000
225	0.031000	0.229667	0.411667	0.171667	0.010333	0.088667	0.005333
400	0.052333	0.244667	0.984333	0.755333	0.057000	0.567333	0.031333
625	0.125333	0.411000	2.088667	2.682000	0.229333	1.932333	0.031000
900	0.213667	0.453000	4.786667	7.885000	0.370000	5.291667	0.047000
1225	0.375000	0.593667	8.036333	19.437333	0.463667	12.286000	0.047000

Table 3: CPU times (in secs): Biharmonic

3.4 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 ($D = \{(x^1, y^1), (x^2, y^2), \dots (x^l, y^l)\},\ x \in R^n,\ y \in -1, 1)$. Furthermore it is desirable to produce a classifier that will work well on unseen examples, i.e. it generalizes well. Consider the example in Fig 2. There are many possible linear classifiers that can separate the data, but there is only one that maximizes the distance between it and 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

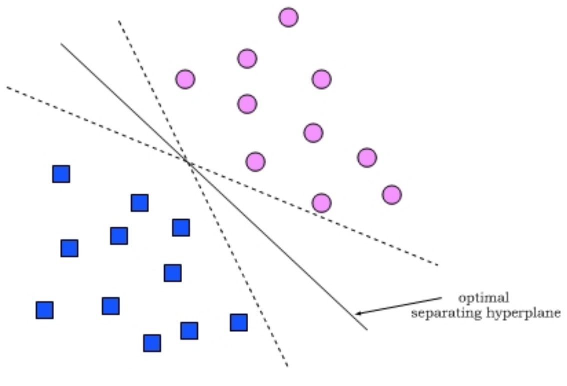


Figure 2: Maximum distance classifier

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

$$\min_{a} \frac{1}{2} a^T Q a - a^T e \quad \text{subject to: } 0 \le a_i \le C \tag{13}$$

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

$$w^{*T} x = \sum_{i=1}^{l} a_i^* y^i K(x^i, x) \tag{14}$$

Hence the separating surface is given by:

$$f(x) = \text{sgn}(w^{*T} x) \tag{15}$$

Many employed kernels $K(x, y)$ contain an implicit bias term and thus omitting the constant term b does not affect significantly the generalization of the resulting classifier. It is important for our implementation to make this assumption because it simplifies the optimization problem. An explicit bias would add the equality constraint $\sum_{i=1}^{l} a_i y^i = 0$ to the problem in Eq. (13).

In our experiments we used the CLOUDS [17] data set, which is a two-dimensional data set with two classes. We have constructed the problem in Eq. (13) using an RBF Kernel function $K(x, y) = \exp(-\frac{||x-y||^2}{2\sigma^2})$, 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.(13)
- Apply each solver, obtain the corresponding separating surface and test-set error.

In these experiments our algorithm did not perform as well. This was due to the large condition number of matrix Q ($cond(Q) \ge 10^{12}$), that affects the linear solver. To circumvent this, we added in the main diagonal of Q a small positive term. This leads to a slightly different problem, which however is solved efficiently. All the other algorithms solved the problem without modifying Q, and reached identical solutions, slightly different from ours.

The results in Table 4 show that despite the modification of Q, the separating surface[2], is almost identical to the surface constructed by the other techniques. This is depicted in Fig 3.

Dimension	BOXCQP		MOSEK		QUADPROG		MINQ	
	Time	Error	Time	Error	Time	Error	Time	Error
200	0.235	13.5%	1.453	13.41%	2.234	13.41%	3.719	13.41%
500	2.156	11.2%	2.532	11.84%	48.781	11.84%	23.828	11.84 %
1000	2.234	10.6%	7.437	10.52%	503.000	10.52%	144.844	10.52%
2000	28.156	10.36%	81.546	10.36%	>1000	10.36%	$>$ 1000	10.36%

Dimension	QPOPT		QLD		SQOPT	
	Time	Error	Time	Error	Time	Error
200	0.311	13.41%	0.0433	13.41%	0.0233	13.41%
500	1.333	11.84%	4.437	11.84%	3.723	11.84%
1000	5.532	10.52 %	10.781	10.52%	11.234	10.52%
2000	38.177	10.36%	30.655	10.36%	55.334	10.36%

Table 4: CPU times (in secs): SVM training

4 CONCLUSIONS

We have presented an infeasible active set method that solves a convex quadratic programming problem with simple bounds. The algorithm uses the unconstrained minimum (Newton point), and projects it on the box defined

[1]Also known as explicit bias
[2]And hence the error in the test set

by the bound constraints. The main computational task of our approach is the solution of a linear system, whose dimension is equal to the number of free parameters at each iteration.

Extensive experimental testing revealed a connection between the iterations of our method and the condition number of the Hessian. In problems where this number is very large ($\approx 10^{12}$), a modification of the Hessian is necessary to obtain a solution efficiently.

Box-shaped trust region methods for non-linear optimization, generate intermediate convex quadratic subproblems that in most cases are treated approximately. Embedding our method in such a framework, will enhance its performance by solving these problems exactly. We are currently investigating the performance of such schemes.

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] 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).

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

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

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

[8] E. D. Andersen and K. D. Andersen *The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm*. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, p.p 197-232, (2000) Kluwer Academic Publishers.

[9] Arnold Neumaier,*MINQ - General Definite and Bound Constrained Indefinite Quadratic Programming*, WWW-Document, (1998), `http://www.mat.univie.ac.at/~neum/software/minq/`.

[10] 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

[11] P. E. Gill, W. Murray and M. A. Saunders, *SQOPT 5.3 User's Guide*, Report NA 97-4, Dept of Mathematics, University of California, San Diego (revised 1998).

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

[13] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM (1997)

[14] R. Vanderbei, *LOQO: An Interior Point Code for Quadratic Programming*, Technical Report SOR 94–15, Program in Statistics & Operations Research, Princeton University (1995)

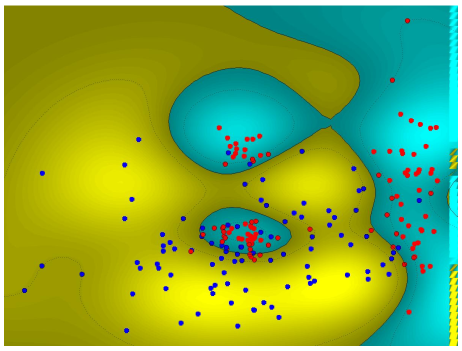
[15] C. Meszaros, *The BPMPD interior point solver for convex quadratic problems* Academy of Sciences, Budapest (1998)

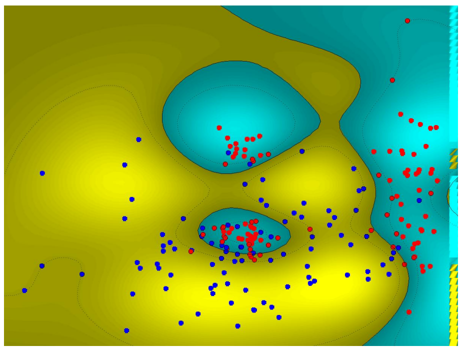
[16] K. Kunisch, and F. Rendl, *An Infeasible Active Set Method For Quadratic Problems With Simple Bounds*, SIAM J. Optim. **14**(1), pp. 35-52 (2003)

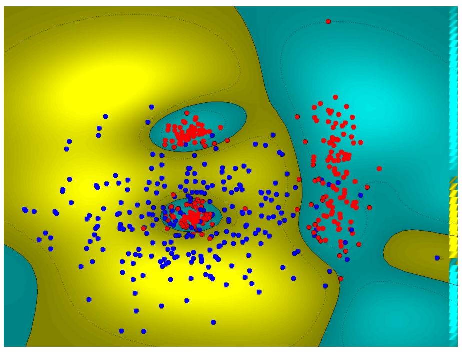
[17] C. L. Blake, and C. J. Merz, *UCI Repository of machine learning databases*, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Irvine, University of California, Department of Information and Computer Science (1998).

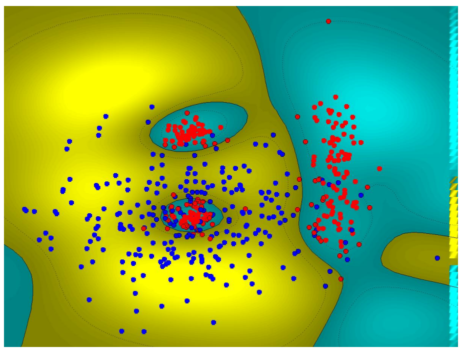
[18] A. F. Perold, *Large-Scale Portfolio Optimization*, Management Science **30**, 1143-1160 (1984).

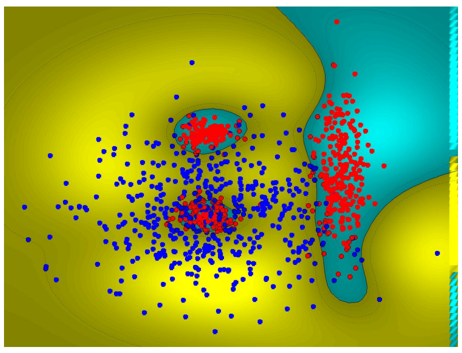
[19] K. Holmström , M. M. Edvall, and A. O. Göran, *TOMLAB - for Large-Scale Robust Optimization* Proceedings, Nordic MATLAB Conference (2003).

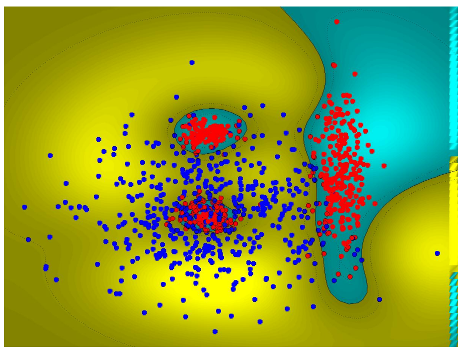

(a) 200 training examples - BOXCQP

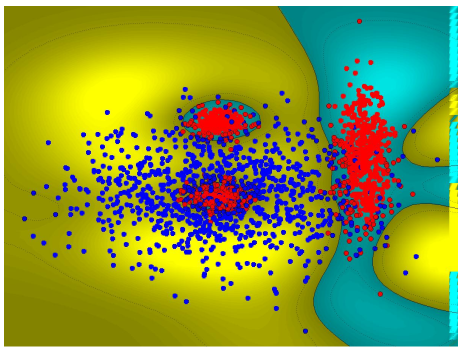

(b) 200 training examples - Others


(c) 500 training examples - BOXCQP


(d) 500 training examples - Others


(e) 1000 training examples - BOXCQP


(f) 1000 training examples - Others


(g) 2000 training examples - BOXCQP

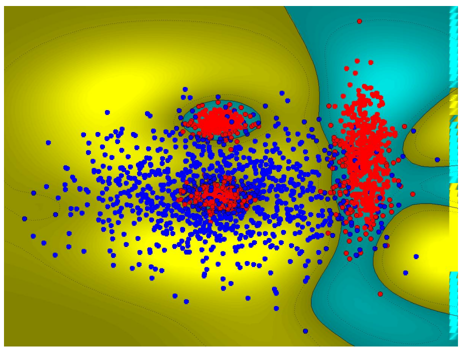

(h) 2000 training examples - Others

Figure 3: SVM classification