Simple Solvers for Large Quadratic Programming Tasks

Vojtěch Franc and Václav Hlaváč

Center for Machine Perception, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University
{xfrancv, hlavac}@cmp.felk.cvut.cz,
http://cmp.felk.cvut.cz

Abstract. This paper describes solvers for specific quadratic programming (QP) tasks. The QP tasks in question appear in numerous problems, e.g., classifier learning and probability density estimation. The QP task becomes challenging when large number of variables is to be optimized. This the case common in practice. We propose QP solvers which are simple to implement and still able to cope with problems having hundred thousands variables.

1 Introduction

A lot of problems in machine learning and pattern recognition lead to quadratic programming (QP) tasks. Although optimization of a general QP task have been studied at length, it is still a challenging problem when the number of variables is large. The large QP tasks are quite common in practice. In such cases it is necessary to derive specialized solvers which exploit all particular properties of the task at hand. In this paper, we propose solvers for special instances of QP tasks which can be applied to many problems. The proposed solvers are simple to implement and they allow to cope with QP tasks having hundred thousands of variables.

The quadratic programming (QP) task which we tackle in this paper is very related to two geometrical tasks. These task are the minimal norm problem (MNP) and the nearest point problem (NPP). The MNP aims to find the minimal norm vector from a convex hull defined by a finite set of vectors. The NPP searches for two nearest vectors from two different convex hulls defined again by finite sets of vectors. Algorithms to solve the MNP and the NPP first appeared in computational geometry and control engineering but later they were applied in pattern recognition. A simple algorithm for the MNP was published by Gilbert [4]. Kozinec [9] proposed a very similar method which he used for separation of two convex hulls given by finite vector sets. A different algorithm for the MNP was described by Mitchell, Demyanov and Malozemov [10]. Various modifications of the Kozinec's algorithm applied for analysis of linear discriminant functions were described in the book by Schlesinger and Hlaváč [11]. Keerthi et al. [7] proposed a new method to solve the NPP which was applied for training

W. Kropatsch, R. Sablatnig, and A. Hanbury (Eds.): DAGM 2005, LNCS 3663, pp. 75–84, 2005.

of binary Support Vector Machines (SVM) classifiers. Another method for the NPP was proposed by Kowalczyk [8] who also used it for training of the SVM's. A modification of the Kozinec's algorithm for training of the SVM's was further proposed by Franc and Hlaváč [2].

We concentrate on a slightly more general QP tasks the special cases of which are the MNP and the NPP. We denote this task as the generalized MNP and the generalized NPP problem. The generalization consists in adding a linear term to the QP criterion and assuming that the Hessian of the criterion is an arbitrary symmetric positive definite matrix. The original problems are recovered after the linear term is removed and the Hessian equals to a product of two matrices. The exact definitions of the generalized MNP and NPP are given in Section 2.

We claim that the above mentioned methods designed to solve the original MNP and NPP can be simply modified to solve the general problems. Moreover, we will introduce a general framework for sequential algorithms suitable to solve the generalized problems. The special instances of the general framework are the MNP and NPP methods. The introduced framework allows for their comparison and better understanding. We also derived a new method which proved to outperform the original ones. Due to a lack of space, we describe only the new method in Section 3. Detailed description and experimental evaluation can be found in [1].

While the original solvers were used mainly for training of binary SVM's their generalization proposed in this paper is applicable in many other problems. For example, training of multi-class SVM classifiers [3], Reduced Set Density Estimation [5], Support Vector Data Description [12], several modifications of SVM's for classification and regression introduced in [6].

The paper is organized as follows. The generalized MNP and NPP are defined in Section 2. The general framework and two particular algorithms are described in Section 3. Conclusions are drawn in Section 4.

2 Generalized Minimal Norm and Nearest Point Problems

Let a quadratic objective function

$$Q(\boldsymbol{\alpha}) = \frac{1}{2}\langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\alpha}\rangle + \langle \boldsymbol{c}, \boldsymbol{\alpha}\rangle \,, \tag{1}$$

be determined by a vector $\boldsymbol{c} \in \mathbb{R}^m$ and a symmetric positive definite matrix $\mathbf{H} \in \mathbb{R}^{m\times m}$. The symbol $\langle \cdot, \cdot \rangle$ stands for the dot product Let $\mathcal{A} \subseteq \mathbb{R}^m$ be a convex set of feasible solutions $\boldsymbol{\alpha} \in \mathcal{A}$. The goal is to solve the following task

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathcal{A}} \frac{1}{2}\langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\alpha}\rangle + \langle \boldsymbol{c}, \boldsymbol{\alpha}\rangle \,. \tag{2}$$

Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be non-empty disjoint sets of indices such that $\mathcal{I}_1 \cup \mathcal{I}_2 = \mathcal{I} = \{1, 2, \ldots, m\}$. Let vectors $\boldsymbol{e}, \boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{R}^m$ be defined as follows

$$[\boldsymbol{e}]_i = 1 \,, i \in \mathcal{I} \,, \quad [\boldsymbol{e}_1]_i = \begin{cases} 1 \text{ for } i \in \mathcal{I}_1 \\ 0 \text{ for } i \in \mathcal{I}_2 \end{cases} \,, \quad [\boldsymbol{e}_2]_i = \begin{cases} 0 \text{ for } i \in \mathcal{I}_1 \\ 1 \text{ for } i \in \mathcal{I}_2 \end{cases} \,,$$

where $[\cdot]_i$ denotes i-the coordinate of a vector. The optimization problems (2) with two distinct feasible sets $\mathcal{A}$ are assumed. In the first case, the *generalized Minimal Norm Problem* is the optimization problem (2) with the feasible set $\mathcal{A}$ determined by

$$\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \langle \boldsymbol{\alpha}, \boldsymbol{e} \rangle = 1, \boldsymbol{\alpha} \geq \mathbf{0}\} \,. \tag{3}$$

In the second case, the *generalized Nearest Point Problem* is the optimization problem (2) with the feasible set $\mathcal{A}$ determined by

$$\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \langle \boldsymbol{\alpha}, \boldsymbol{e}_1 \rangle = 1, \langle \boldsymbol{\alpha}, \boldsymbol{e}_2 \rangle = 1 \,, \boldsymbol{\alpha} \geq \mathbf{0}\} \,, \tag{4}$$

where vectors $\boldsymbol{e}, \boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{R}^m$ are defined as follows

$$[\boldsymbol{e}]_i = 1 \,, i \in \mathcal{I} \,, \quad [\boldsymbol{e}_1]_i = \begin{cases} 1 \text{ for } i \in \mathcal{I}_1 \\ 0 \text{ for } i \in \mathcal{I}_2 \end{cases} \,, \quad [\boldsymbol{e}_2]_i = \begin{cases} 0 \text{ for } i \in \mathcal{I}_1 \\ 1 \text{ for } i \in \mathcal{I}_2 \end{cases} \,.$$

The generalized MNP and generalized NPP are convex optimization problems as both the objective functions (1) and the feasible sets (3) and (4), respectively, are convex.

3 Algorithms

3.1 Framework of Sequential Algorithms

The optimization problem (2) with the feasible set $\mathcal{A}$ can be transformed to a sequence of auxiliary optimization problems with the same objective function (1) but with much simpler auxiliary feasible sets $\mathcal{A}^{(0)}$, $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(t)}$. Solving the problem (2) with respect to the auxiliary feasible sets yields a sequence of solutions $\boldsymbol{\alpha}^{(0)}, \boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(t)}$. The sequence of feasible sets $\mathcal{A}^{(0)}$, $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(t)}$ is assumed to be constructed such that (i) the sequence $Q(\boldsymbol{\alpha}^{(0)}) > Q(\boldsymbol{\alpha}^{(1)}) > \ldots > Q(\boldsymbol{\alpha}^{(t)})$ converges to the optimal solution $Q(\boldsymbol{\alpha}^*)$ and (ii) the auxiliary problems can be solved efficiently. The sequential algorithm solving the QP task which implements the idea mentioned above is summarized by Algorithm 1.

Algorithm 1: A Sequential Optimization Algorithm

1. Initialization. Select $\boldsymbol{\alpha}^{(0)} \in \mathcal{A}$.
2. Repeat until stopping condition is satisfied:
   (a) Select a feasible set $\mathcal{A}^{(t+1)}$ such that

$$Q(\boldsymbol{\alpha}^{(t)}) > \min_{\boldsymbol{\alpha} \in \mathcal{A}^{(t+1)}} Q(\boldsymbol{\alpha}) \,. \tag{5}$$

   (b) Solve the auxiliary task

$$\boldsymbol{\alpha}^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathcal{A}^{(t+1)}} Q(\boldsymbol{\alpha}) \,. \tag{6}$$

The auxiliary feasible sets $\mathcal{A}^{(t+1)}$ is assumed to be a line segment

$$\mathcal{A}_L^{(t+1)} = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} = (1-\tau)\boldsymbol{\alpha}^{(t)} + \tau\boldsymbol{\beta}^{(t)}, 0 \leq \tau \leq 1\}\,, \tag{7}$$

between the current solution $\boldsymbol{\alpha}^{(t)}$ and a vector $\boldsymbol{\beta}^{(t)} \in \mathcal{A}$. The optimization task (6) has an analytical solution in the case in which the feasible set is a line segment. Moreover, there exist simple rules to construct the vectors $\boldsymbol{\beta}^{(t)}$ which guarantee that condition (5) is satisfied.

3.2 Stopping Conditions

There is a need to stop the algorithm when it gets sufficiently close to the optimum. We assume the following two reasonable stopping conditions:

1. ε-optimal solution. The algorithm stops if
$$Q(\boldsymbol{\alpha}) - Q(\boldsymbol{\alpha}^*) \leq \varepsilon\,. \tag{8}$$
2. Scale invariant ε-optimal solution. The algorithm stops if
$$Q(\boldsymbol{\alpha}) - Q(\boldsymbol{\alpha}^*) \leq \varepsilon|Q(\boldsymbol{\alpha})|\,. \tag{9}$$

The prescribed $\varepsilon > 0$ controls the precision of the found solution. The stopping conditions (8) and (9) can be evaluated despite the unknown optimal value $Q(\boldsymbol{\alpha}^*)$ because a lower bound $Q_{LB}(\boldsymbol{\alpha})$ can be used instead. Let the inequality $Q(\boldsymbol{\alpha}^*) \geq Q_{LB}(\boldsymbol{\alpha})$ hold. Then the satisfaction of the condition $Q(\boldsymbol{\alpha}) - Q_{LB}(\boldsymbol{\alpha}) \leq \varepsilon$ implies that condition (8) holds as well. Similarly, if the condition $Q(\boldsymbol{\alpha}) - Q_{LB}(\boldsymbol{\alpha}) \leq \varepsilon|Q(\boldsymbol{\alpha})|$ holds then (9) is also satisfied.

The computation of the lower bound $Q_{LB}(\boldsymbol{\alpha})$ depends on the feasible set used. We give the lower bounds without derivation which can be found in [1]. In the case of the generalized MNP with the feasible set $\mathcal{A}$ defined by (3), the following lower bound $Q_{LB}(\boldsymbol{\alpha})$ can be used

$$Q_{LB}(\boldsymbol{\alpha}) = \min_{i \in \mathcal{I}} [\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{c}]_i - \frac{1}{2}\langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\alpha}\rangle\,. \tag{10}$$

In the case of the generalized NPP with the feasible set $\mathcal{A}$ defined by (4), the lower bound Q_{LB} reads

$$Q_{LB}(\boldsymbol{\alpha}) = \min_{i \in \mathcal{I}_1} [\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{c}]_i + \min_{i \in \mathcal{I}_2} [\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{c}]_i - \frac{1}{2}\langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\alpha}\rangle\,. \tag{11}$$

3.3 Solution for a Line Segment

Let the feasible set $\mathcal{A}_L^{(t+1)}$ be a line segment (7) between the current solution $\boldsymbol{\alpha}^{(t)}$ and a vector $\boldsymbol{\beta}^{(t)} \in \mathcal{A}$. The quadratic objective function (1) defined over the line segment $\mathcal{A}_L^{(t+1)}$ reads

$$\begin{aligned} Q_L^{(t+1)}(\tau) &= Q\left(\boldsymbol{\alpha}^{(t)}(1-\tau) + \tau\boldsymbol{\beta}^{(t)}\right) \\ &= \frac{1}{2}(1-\tau)^2\langle \boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)}\rangle + \tau(1-\tau)\langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)}\rangle \\ &\quad + \frac{1}{2}\tau^2\langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\beta}^{(t)}\rangle + (1-\tau)\langle \boldsymbol{c}, \boldsymbol{\alpha}^{(t)}\rangle + \tau\langle \boldsymbol{c}, \boldsymbol{\beta}^{(t)}\rangle\,. \end{aligned} \tag{12}$$

The objective function is now parameterized by a single variable $0 \leq \tau \leq 1$. It is obvious that $Q_L^{(t+1)}(0) = Q(\boldsymbol{\alpha}^{(t)})$ and $Q_L^{(t+1)}(1) = Q(\boldsymbol{\beta}^{(t)})$. The improvement gained from the optimization over the line segment $\mathcal{A}_L^{(t+1)}$ is

$$\Delta^{(t+1)} = Q(\boldsymbol{\alpha}^{(t)}) - Q(\boldsymbol{\alpha}^{(t+1)}) = Q(\boldsymbol{\alpha}^{(t)}) - \min_{0 \leq \tau \leq 1} Q_L^{(t+1)}(\tau) \,. \tag{13}$$

A new vector $\boldsymbol{\alpha}^{(t+1)}$ is determined as

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)}(1-\tau^*) + \tau^* \boldsymbol{\beta}^{(t)} \quad \text{where} \quad \tau^* = \operatorname*{argmin}_{0 \leq \tau \leq 1} Q_L^{(t+1)}(\tau) \,. \tag{14}$$

The vector $\boldsymbol{\beta}^{(t)} \in \mathcal{A}$ must be selected such that the improvement (13) is positive. This occurs when the derivative of $Q_L^{(t+1)}(\tau)$ evaluated in zero is negative, i.e.,

$$\left. \frac{\partial Q_L^{(t+1)}(\tau)}{\partial \tau} \right|_{\tau=0} = \langle (\boldsymbol{\beta}^{(t)} - \boldsymbol{\alpha}^{(t)}), (\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}) \rangle < 0 \,. \tag{15}$$

The solution of (14) can be found analytically by setting the derivative of $Q_L^{(t+1)}(\tau)$ to zero and solving for τ. This yields

$$\tau^* = \min \left(1, \frac{\langle (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}), (\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}) \rangle}{\langle \boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle - 2\langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle + \langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\beta}^{(t)} \rangle} \right) \,, \tag{16}$$

where the minimum $\min(1, \cdot)$ guarantees that the solution does not leave the feasible set $\mathcal{A}_L^{(t+1)}$. If $\tau^* = 1$ then $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\beta}^{(t)}$ which implies that the value of improvement (13) equals to

$$\Delta^{(t+1)} = Q(\boldsymbol{\alpha}^{(t)}) - Q(\boldsymbol{\beta}^{(t)}) \,. \tag{17}$$

If $\tau^* < 0$ then the value of improvement can be derived by substituting (12) and (16) to (13). After some manipulation we get

$$\Delta^{(t+1)} = \frac{\langle (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}), (\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}) \rangle^2}{2(\langle \boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle - 2\langle \boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\beta}^{(t)} \rangle + \langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\beta}^{(t)} \rangle)} \,. \tag{18}$$

3.4 Algorithm for Generalized Minimal Norm Problem

The algorithm described in this section fulfills the general framework outlined by Algorithm 1. The particular algorithm is determined by a rule for selection of the line segment $\mathcal{A}_L^{(t+1)}$. The line segment is constructed between the current solution $\boldsymbol{\alpha}^{(t)}$ and a vector $\boldsymbol{\beta}^{(t)}$. In this case, we assume that all entries of the vector $\boldsymbol{\beta}^{(t)}$ equal to the current solution $\boldsymbol{\alpha}^{(t)}$ except for two entries u and v, i.e.,

$$[\boldsymbol{\beta}^{(t)}]_i = \begin{cases} [\boldsymbol{\alpha}^{(t)}]_u + [\boldsymbol{\alpha}^{(t)}]_v & \text{for } i = u \,, \\ 0 & \text{for } i = v \,, \\ [\boldsymbol{\alpha}^{(t)}]_i & \text{for } i \neq u \wedge i \neq v, \end{cases} \qquad \forall i \in \mathcal{I} \,. \tag{19}$$

If proper u and v are used then the optimization over the line segment between $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\beta}^{(t)}$ leads to the improvement $\Delta^{(t+1)}(u,v) = Q(\boldsymbol{\alpha}^{(t)}) - Q(\boldsymbol{\alpha}^{(t+1)})$. The exact value of the improvement can be derived by substituting (19) to (18) which for $\tau < 1$ gives

$$\Delta^{(t+1)}(u,v) = \frac{([\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_v - [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_u)^2}{2([\mathbf{H}]_{u,u} - 2[\mathbf{H}]_{u,v} + [\mathbf{H}]_{v,v})}\,, \tag{20}$$

and substituting (19) to (17) which for $\tau = 1$ gives

$$\Delta^{(t+1)}(u,v) = [\boldsymbol{\alpha}^{(t)}]_v([\mathbf{H}\boldsymbol{\alpha}^{(t)}+\boldsymbol{c}]_v - [\mathbf{H}\boldsymbol{\alpha}^{(t)}+\boldsymbol{c}]_u) - \frac{1}{2}[\boldsymbol{\alpha}^{(t)}]_v^2([\mathbf{H}]_{u,u} - 2[\mathbf{H}]_{u,v} + [\mathbf{H}]_{v,v}). \tag{21}$$

From (15) it follows that u and v must be selected such that $[\boldsymbol{\alpha}^{(t)}]_v > 0$ and the inequality

$$\kappa(u,v) = [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_v - [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_u > 0\,,$$

holds to guarantee a non-zero improvement. The computation of τ for $\boldsymbol{\beta}^{(t)}$ given by (19) simplifies to

$$\tau = \frac{[\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_v - [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_u}{[\boldsymbol{\alpha}^{(t)}]_v([\mathbf{H}]_{u,u} - 2[\mathbf{H}]_{u,v} + [\mathbf{H}]_{v,v})}\,. \tag{22}$$

It remains to select the indices u and v such that the improvement will be maximized. The algorithm proposed by Mitchell, Demyanov and Malozemov (MDM) selects the indices u and v so that $\kappa(u,v)$ is maximized. This is reasonable since the value of $\kappa(u,v)$ approximates the value of improvement $\Delta^{(t+1)}(u,v)$. A novel method proposed here is based on searching for the entries u and v which maximize the exact value of the improvement $\Delta^{(t+1)}(u,v)$ instead of its approximation $\kappa(u,v)$.

The computation of the improvement for given u and v requires evaluation of the (22) and, based on the value of τ, (20) or (21) is used. To select the optimal pair (u,v) one has to try $\frac{d(d+1)}{2}$ combinations where d is the number of non-zero entries of the current solution $\boldsymbol{\alpha}^{(t)}$. Notice that zero entries of $\boldsymbol{\alpha}^{(t)}$ can be disregarded as the corresponding vector $\boldsymbol{\beta}^{(t)}$ would be the same as $\boldsymbol{\alpha}^{(t)}$. The search for the optimal (u,v) would require to access d columns of the matrix $\mathbf{H}$ which would be too expensive . To overcome this difficulty, the following strategy of selecting (u,v) is proposed. The index u is selected such that

$$u \in \operatorname*{argmin}_{i \in \mathcal{I}} [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i\,.$$

The found u is fixed and the index v is computed as

$$v \in \operatorname*{argmax}_{i \in \mathcal{I}_V} \Delta^{(t+1)}(u,i)\,,$$

where the $\mathcal{I}_V = \{i \in \mathcal{I} \colon [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i > [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_u \wedge [\boldsymbol{\alpha}^{(t)}]_i > 0\}$ is a set of admissible indices for v for which the improvement can be greater than zero.

A similar strategy would be to fix v and search for the optimal u or to apply both these searches together. All these three combinations were experimentally tested and the proposed strategy required on average the minimal access to the matrix $\mathbf{H}$. Algorithm 2 summarizes the proposed method:

Algorithm 2: Algorithm for generalized MNP

1. Initialization. Set $\boldsymbol{\alpha}^{(0)} \in \mathcal{A}$.
2. Repeat until stopping condition is satisfied:
   (a) Construct vector $\boldsymbol{\beta}^{(t)} \in \mathcal{A}$ such that

$$[\boldsymbol{\beta}^{(t)}]_i = \begin{cases} [\boldsymbol{\alpha}^{(t)}]_u + [\boldsymbol{\alpha}^{(t)}]_v & \text{for } i = u\,, \\ 0 & \text{for } i = v\,, \\ [\boldsymbol{\alpha}^{(t)}]_i & \text{for } i \neq u \wedge i \neq v, \end{cases} \qquad \forall i \in \mathcal{I}\,,$$

   where

$$u \in \operatorname*{argmin}_{i \in \mathcal{I}} [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i\,, \quad \text{and} \quad v \in \operatorname*{argmax}_{i \in \mathcal{I}_V} \Delta^{(t+1)}(u, i)\,.$$

   The improvement $\Delta^{(t+1)}(u, i)$ is computed using (22), (20) and (21).
   (b) Update

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)}(1 - \tau) + \tau \boldsymbol{\beta}^{(t)}\,,$$

   where

$$\tau = \min\left(1, \frac{\langle (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}), (\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}) \rangle}{\langle \boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle - 2\langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle + \langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\beta}^{(t)} \rangle}\right)\,.$$

The number of iterations depends on used stopping condition. Let us assume that ε-optimality condition (8) is applied. We can prove that the number of iterations t can be bounded using the following quantities: the prescribed precision ε, the initial value $Q(\boldsymbol{\alpha}^{(0)})$, the optimal value $Q(\boldsymbol{\alpha}^*)$ and a diameter $D \in \mathbb{R}^+$ of the matrix $\mathbf{H}$ defined as

$$D^2 = \max_{\substack{\boldsymbol{\alpha} \in \mathcal{A} \\ \boldsymbol{\beta} \in \mathcal{A}}} \left(\langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\beta} \rangle - 2\langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\beta} \rangle + \langle \boldsymbol{\beta}, \mathbf{H}\boldsymbol{\beta} \rangle \right)\,.$$

Theorem 1. *Algorithm 2 started from an arbitrary vector $\boldsymbol{\alpha}^{(0)} \in \mathcal{A}$ returns the vector $\boldsymbol{\alpha}$ satisfying for any $\varepsilon > 0$ the ε-optimality condition $Q(\boldsymbol{\alpha}) - Q(\boldsymbol{\alpha}^*) \leq \varepsilon$ after at most $t_{max} < \infty$ iterations, where*

$$t_{max} = \frac{2D^2(m-1)}{\varepsilon^2}(Q(\boldsymbol{\alpha}^{(0)}) - Q(\boldsymbol{\alpha}^*))\,.$$

For proof we refer to [1].

3.5 Algorithm for Generalized Nearest Point Problem

The algorithm to solve the generalized NPP is of the same nature as that solving the generalized MNP problem. Let $\mathcal{A}_1^{(t+1)} = \{\boldsymbol{\alpha} \in \mathcal{A}\colon [\boldsymbol{\alpha}]_i = [\boldsymbol{\alpha}^{(t)}]_i, i \in \mathcal{I}_2\}$ denote a set of vectors from the feasible set $\mathcal{A}$ which have the entries $\mathcal{I}_2$ fixed to corresponding entries of the current solution $\boldsymbol{\alpha}^{(t)}$. Similarly, let $\mathcal{A}_2^{(t+1)} = \{\boldsymbol{\alpha} \in \mathcal{A}\colon [\boldsymbol{\alpha}]_i = [\boldsymbol{\alpha}^{(t)}]_i, i \in \mathcal{I}_1\}$ denote vectors with the entries $\mathcal{I}_1$ fixed. The algorithm for the generalized NPP constructs the vector $\boldsymbol{\beta}^{(t)}$ such that it belongs either to $\mathcal{A}_1^{(t+1)}$ or to $\mathcal{A}_2^{(t+1)}$ which is only the difference compared to Algorithm 2.

Using the same reasoning as discussed in Section 3.4 we first find the indices

$$u_1 \in \operatorname*{argmin}_{i \in \mathcal{I}_1} [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i \,, \qquad u_2 \in \operatorname*{argmin}_{i \in \mathcal{I}_2} [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i \,. \tag{23}$$

Second, the optimal v_1 and v_2 are sought for so that

$$v_1 \in \operatorname*{argmin}_{i \in \mathcal{I}_{V1}} \Delta^{(t+1)}(u_1, i) \,, \qquad v_2 \in \operatorname*{argmin}_{i \in \mathcal{I}_{V2}} \Delta^{(t+1)}(u_2, i) \,, \tag{24}$$

where $\mathcal{I}_{V1} = \{i \in \mathcal{I}_1\colon [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i > [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_{u_1} \wedge [\boldsymbol{\alpha}^{(t)}]_i > 0\}$ and $\mathcal{I}_{V2} = \{i \in \mathcal{I}_2\colon [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_i > [\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c}]_{u_2} \wedge [\boldsymbol{\alpha}^{(t)}]_i > 0\}$ are sets of admissible indices. Finally, the pair of indices (u_1, v_1) or (u_2, v_2) which yields the bigger improvement is used to construct the vector $\boldsymbol{\beta}^{(t)}$. The formula for the improvement $\Delta^{(t+1)}(u, v)$ can be derived using the same way as in the the generalized MNP case. So that the improvement is computed by the formula (20) if $\tau < 1$ and (21) if $\tau = 1$. Algorithm 3 summarizes the proposed method:

Algorithm 3: Algorithm for generalized NPP

1. Initialization. Set $\boldsymbol{\alpha}^{(0)} \in \mathcal{A}$.
2. Repeat until stopping condition is satisfied:
   (a) Construct vector $\boldsymbol{\beta}^{(t)} \in \mathcal{A}$

$$[\boldsymbol{\beta}^{(t)}]_i = \begin{cases} [\boldsymbol{\alpha}^{(t)}]_u + [\boldsymbol{\alpha}^{(t)}]_v & \text{for } i = u \,, \\ 0 & \text{for } i = v \,, \\ [\boldsymbol{\alpha}^{(t)}]_i & \text{for } i \neq u \wedge i \neq v, \end{cases}$$

   where the indices (u, v) are computed as follows. First, the pairs (u_1, v_1) and (u_2, v_2) are computed by (23) and (24). Second, the pair which yields the bigger improvement $\Delta^{(t+1)}(u, v)$ is taken for (u, v).

   (b) Update

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)}(1 - \tau) + \tau\boldsymbol{\beta}^{(t)} \,,$$

   where

$$\tau = \min\left(1, \frac{\langle(\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}), (\mathbf{H}\boldsymbol{\alpha}^{(t)} + \boldsymbol{c})\rangle}{\langle\boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)}\rangle - 2\langle\boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)}\rangle + \langle\boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\beta}^{(t)}\rangle}\right) \,.$$

Theorem 2. *Algorithm 3 started from an arbitrary vector* $\boldsymbol{\alpha}^{(0)} \in \mathcal{A}$ *returns the vector* $\boldsymbol{\alpha}$ *satisfying for any* $\varepsilon > 0$ *the* ε*-optimality condition* $Q(\boldsymbol{\alpha}) - Q(\boldsymbol{\alpha}^*) \leq \varepsilon$ *after at most* $t_{max} < \infty$ *iterations, where*

$$t_{max} = \frac{8D^2(m-2)}{\varepsilon^2}(Q(\boldsymbol{\alpha}^{(0)}) - Q(\boldsymbol{\alpha}^*)) \,.$$

The upper bound on the maximal number of iterations is defined by the same quantities as in Theorem 1 except for the diameter D. We refer to [1] for the definition of D and the proof of Theorem 2.

3.6 Efficient Implementation

The main requirement on the developed QP solvers is the ability to deal with large problems, i.e., problems where the matrix $\mathbf{H}$ in the definition of the QP task is large and cannot be stored in the memory. The aim is to minimize an access of the QP solver to the entries of the matrix $\mathbf{H}$. It is seen that the algorithms described in Sections 3.4 and 3.4 require only two columns of the matrix $\mathbf{H}$ in each iteration. Moreover, in many cases only a small subset of the columns is requested by the algorithms, i.e., those which corresponds to the non-zero entries $\mathcal{I}_\emptyset = \{i \in \mathcal{I} \colon [\alpha]_i > 0\}$ of the vector $\boldsymbol{\alpha}^{(t)}$. This allows to use a cache for the most often requested columns without the need to store the whole matrix $\mathbf{H}$. For instance, the *First In First Out* (FIFO) turned out to be suitable.

The efficient implementation of the algorithms lies in maintaining a cache of key variables. In the case of the generalized MNP problem, these variables are

$$\delta_\alpha^{(t)} = \langle \boldsymbol{\alpha}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle \,, \quad \boldsymbol{h}_\alpha^{(t)} = \mathbf{H}\boldsymbol{\alpha}^{(t)} \,, \quad \delta_\beta^{(t)} = \langle \boldsymbol{\beta}^{(t)}, \mathbf{H}\boldsymbol{\alpha}^{(t)} \rangle \,, \tag{25}$$

where $\delta_\alpha^{(t)} \in \mathbb{R}$ is a scalar, $\boldsymbol{h}_\alpha^{(t)} \in \mathbb{R}^m$ is a vector and $\delta_\beta^{(t)} \in \mathbb{R}$ is a scalar. Having the variables (25), the number of computations scales with $O(m)$ in each iteration of any QP solver, i.e., it is linear with respect to the number of variables m. It is easy to show that the variables (25) can be updated in each iteration instead of computing them from a scratch. The updates of all variables also require $O(m)$ operations. The exact updating formulas can be simply derived by substituting rule (19) to (25).

The same idea is also applied for the generalized NPP. In this case, there are more key variables to be cached. However, the number of computations required for their updating is also $O(m)$. The overall number of computations required in each iteration is thus $O(m)$ for both described algorithms. We refer to [1] for a detailed description.

4 Conclusions

We introduced a general framework for sequential algorithms suitable to solve the generalized minimal norm problem (MNP) and generalized nearest point problem (NPP). The framework allows for comparison of existing methods designed to solve original MNP and NPP. The methods can be simply modified to

solve the generalized formulations. We also derived a new algorithm with performance superior to other methods. The new algorithm is described in this paper while comparison to other methods was skipped due to a lack of space. We refer to [1] where all the methods are described in details and compared to each other in numerous synthetical and real-life problems.

Acknowledgments

The authors were supported by the projects IST-004176 COSPAL, CONEX GZ 45.535. The first author was also supported by MSM 6840770013. The second author was also supported by INTAS 04-77-7347 (PRINCESS).

References

1. V. Franc. Optimization algorithms for kernel methods. PhD Thesis, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, 2005. To be submitted in July 2005.
2. V. Franc and V. Hlaváč. A Simple Learning Algorithm for Maximal Margin Classifier. In *Kernel and Subspace Methods for Computer Vision, workshop adjoint to the Int. Conference on Neural Netwoks*, pages 1–11. TU Vienna, August 2001.
3. V. Franc and V. Hlaváč. Multi-class Support Vector Machine. In R. Kasturi, D. Laurendeau, and Suen C., editors, *16th International Conference on Pattern Recognition*, volume 2, pages 236–239, Los Alamitos, CA 90720-1314, August 2002. IEEE Computer Society.
4. E.G. Gilbert. Minimizing the quadratic form on a convex set. *SIAM journal on Control and Optimization*, 4:61–79, 1966.
5. M. Girolami and C. He. Probability density estimation from optimally condensed data sample. *IEEE PAMI*, 25(10), 2003.
6. L. Gonzales, C. Angulo, and A. Velasco, F. Catala. Unified dual for bi-class SVM approaches. *Patter Recognition*, 2005. Article in press.
7. S.S. Keerthi, S.K. Shevade, C. Bhattacharya, and K.R.K. Murthy. A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. *IEEE Transactions on Neural Networks*, 11(1):124–136, January 2000.
8. A. Kowalczyk. Maximal margin perceptron. In P.J. Bartlett, B. Schloköpf, D. Schuurmans, and A.J. Smola, editors, *Advances in Large-Margin Classifiers*, pages 75–113. The MIT Press, 2000.
9. B.N. Kozinec. Recurrent algorithm separating convex hulls of two sets. In V.N. Vapnik, editor, *Learning algorithms in pattern recognition*, pages 43–50. Sovetskoje radio, Moskva, 1973.
10. B.F. Mitchell, V.F. Demyanov, and V.N. Malozemov. Finding the point of a polyhedron closest to the origin. *SIAM journal on Control and Optimization*, 12:19–26, 1974.
11. M.I. Schlesinger and V. Hlaváč. *Ten lectures on statistical and structural pattern recognition*. Kluwer Academic Publishers, 2002.
12. D.M.J. Tax and R.P.W Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proceedings ESANN*, pages 251–256, Brussels, 1999.