Pure and Applied Functional Analysis
Volume 8, Number 3, 2023, 829–845

Yokohama Publishers
ISSN 2189-3764 ONLINE JOURNAL

TO RESCALE OR TO PROJECT? SOLVING QUADRATIC PROGRAMMING PROBLEMS WITH LAGRANGE MULTIPLIERS METHODS

IGOR GRIVA

Abstract. We conduct numerical experiments to estimate the complexity of two Lagrange multipliers methods for solving quadratic programming (QP) problems with equality constraints and simple bounds. One is based on a fast gradient method treating the bounds with a projection. Another one is based on Newton's method treating the bounds with a nonlinear rescaling principle. Linear equality constraints are handled with an augmented Lagrangian method in both approaches. For the numerical experiments we selected a QP problem for training dual soft margin support vector machines with n training examples.

We found from the numerical experiments that the time of solving QP problems for training the dual soft-margin SVM grows approximately as $\mathcal{O}(n^3)$ for both methods, i.e. as a cube of the number of training examples. The results suggest a hypothesis that the time of solving QP problems with inequality constraints may scale up similarly to that of solving QP with just equality constraints, or solving a linear system of equations with n variables.

1. Introduction

A convex quadratic programming problem (QP) without inequalities with n variables and m linear equality constraints ($m < n$) can be addressed by solving the Lagrange system of $n + m$ linear equations with $\mathcal{O}(n^3)$ arithmetic operations.

There are several conceptual approaches that deal with inequality constraints. One approach is based on using classical barrier functions that keep the iterates in the relative interior of a feasible set. This approach leads to classical barrier function methods [8] and, in turn, to interior-point methods [23]. Second approach keeps iterates in the feasible set by projecting iterates onto the feasible set if they become infeasible [18]. In the third approach the Lagrange multipliers drive iterates into feasible and optimal sets [16].

In theory, interior-point methods can solve a QP problem in $\mathcal{O}(\sqrt{n}L)$ iterations, where L is the number of bits in the input. Keeping in mind that each iteration of interior point method requires solving a linear system and can be done in $\mathcal{O}(n^3)$, the final complexity becomes $\mathcal{O}(n^{3.5}L)$, polynomial in n (see e.g. [3,10,11,13,25]). With the help of numerical linear algebra the complexity can be brought down to $\mathcal{O}(n^3L)$ in some cases [3]. It has been also observed that empirical numerical performance

2020 *Mathematics Subject Classification.* 65K05, 90C06, 90C20, 90C90.

Key words and phrases. Quadratic programming, Lagrange multipliers methods, fast projected gradient, nonlinear rescaling, augmented Lagrangian, support vector machines.

of the interior-point method tends to be better than the worst case of $\mathcal{O}(\sqrt{n}L)$ iterations.

For the second and the third approaches we are not aware of similar bounds, but the asymptotic bounds for rate of convergence are provided (see, e.g. [20]). Therefore, the goal of this manuscript is to estimate empirically the time required to solve a QP problem with the second and the third approaches as a function of the number of variables n. As a representative of the second approach a fast projected gradient method (see [7, 19]) is selected. As a representative the third approach a nonlinear rescaling (NR) method (see [16, 20]) with Newton steps is selected. Equality constraints are treated with augmented Lagrangian method [9,21] for both algorithms. Thus the manuscript is dealing with augmented Lagrangian approach with different ways of treating the simple bounds: first, using fast projected gradient for primal minimization; second, using Newton's method for primal minimization in the framework of NR method. Therefore any difference that we observe in behavior of the two algorithms can be attributed mainly to the different treatment of the simple bounds and to whether the first- or the second-order methods are used to produce iterates.

We would like to mention that any problem with linear inequality constraints may be converted to a problem with linear equality constraints and simple bounds by introduction the slack variables. Thus we are dealing with the two methods that can be used to solve a general QP problem with linear inequality and equality constraints. In case the inequality constraints are simple bounds (or just nonnegativity), then the projection on such feasible set is computationally inexpensive and takes only $\mathcal{O}(n)$ operations as detailed further.

In the last decade the family of fast gradient method, originally proposed by Nesterov [12], gained a popularity due to its theoretical and computational superiority over simple gradient methods (steepest descent methods). Fast gradient methods belong to a family of first-order methods that do not use Hessians or solve linear systems. Therefore, they are attractive for solving large problems. Adding a projection step onto a set defined by simple bounds does not compromise their computational efficiency. Fast projected gradient methods have similar convergence bounds to those of their nonprojective variants [2, 20].

At the same time it is important to understand better how the computational time of fast projected gradient methods scales up with an increase of the number of variables for QP. Therefore one of the goals of the manuscript is to address this question by conducting computational experiments.

A good alternative to first-order methods is Newton's method, arguably the most famous second-order method. Besides being by itself one of the most widely used algorithms for solving smooth unconstrained minimization problems, Newton's method is responsible for the success of some constrained optimization algorithms such as, for example, interior- and exterior-point methods [4,5]. Every Newton step is capable of producing significant progress of iterates towards the solution resulting in quadratic rate of convergence in the neighborhood of the solution for a large class of problems (see [20] for more details). Of course, the main challenge for the algorithms based on Newton iterations is the necessity of solving a linear system of

equations to find Newton directions each iteration, where most of the computational time is spent.

Therefore conceptually the question is whether it is rewarding to spend computational time in calculating Newton directions since they produce significant progress of the iterates each step, or it is better to stick to first-order methods with their computationally inexpensive one iteration and settle for a much larger number of simple iterations? Addressing this question by running computational experiments is the second goal of the manuscript.

Finally, since Newton's method is efficient only for smooth problems, combining Newton's method with some kind of projection would result in the loss of the quadratic convergence of the Newton iterates while still requiring solving linear system of equations at every iteration. Therefore with Newton's method we employ a different mechanism of attaining feasibility and optimality: nonlinear rescaling principle (NRP) introduced by Polyak [16]. The NRP solves a constrained optimization problem as a sequence of smooth and well-conditioned unconstrained optimization subproblems. Therefore the NRP is well suited to be used with Newton's method. By running computational experiments we aim to get some insight on how the solution time for Newton nonlinear rescaling method scales up comparing to that of a fast projected gradient method.

The manuscript is organized as follows. The next section describes a fast projected gradient augmented Lagrangian algorithm (FPGAL). Section 3 describes Newton nonlinear rescaling algorithm (NNRAL). Section 4 provides numerical results. Section 5 provides a summary and discussion of the results, conclusions and possible future directions of investigating the topic further.

2. Fast projected gradient - augmented Lagrangian algorithm

Consider the problem

$$\begin{aligned} &\underset{x\in B}{\text{minimize}}\ f(x) \\ &\text{subject to}\ g(x)=0, \end{aligned} \tag{2.1}$$

where

$$f(x)=\frac{1}{2}x^TQx+q^Tx,$$

$$g(x)=Ax-b,$$

where Q is an $n\times n$ positive definite matrix, A is an $m\times n$ matrix, $q\in\mathbb{R}^n$, $b\in\mathbb{R}^m$, $m<n$, and the set

$$B=\{x\in\mathbb{R}^n : l_i\le x_i\le u_i,\ i=1,\dots n\}.$$

Some lower and upper bounds could be $-\infty$ or $+\infty$. Therefore some variables may be unbounded.

We assume that the feasible set $Ax-b=0$, $x\in B$ contains nonempty relative interior points. Since Q is positive definite, the resulting problem is strongly convex.

The Lagrangian for (2.1) is given by

$$L(x,\lambda)=f(x)-\lambda^Tg(x),$$

and the augmented Lagrangian is as follows:

$$\mathcal{L}_k(x,\lambda) = f(x) - \lambda^T g(x) + \frac{k}{2} g(x)^T g(x),$$

where $\lambda \in \mathbb{R}^m$ is a vector of Lagrange multipliers that corresponds to the equality constraints and $k > 0$ is the scaling parameter.

The augmented Lagrangian method consists of a sequence of inexact minimizations of $\mathcal{L}_k(x,\lambda)$ in x on the B set

$$\hat{x} \approx \hat{x}(\lambda) = \underset{x \in B}{\operatorname{argmin}}\ \mathcal{L}_k(x,\lambda). \tag{2.2}$$

followed by updating the Lagrange multipliers:

$$\hat{\lambda} = \lambda - kg(\hat{x}).$$

For the stopping criteria of the problem (2.2), we use the following function that measures the violation of the first order optimality conditions for problem (2.2):

$$\mu(x,\lambda) = \max_{1 \le i \le m} \mu_i(x,\lambda), \tag{2.3}$$

where

$$\mu_i(x,\lambda) = \begin{cases} |(\nabla_x \mathcal{L}(x,\lambda))_i|, & \text{if } l_i < x_i < u_i, \\ \max\{0, -(\nabla_x \mathcal{L}(x,\lambda))_i\}, & \text{if } x_i = l_i, \\ \max\{0, (\nabla_x \mathcal{L}(x,\lambda))_i\}, & \text{if } x_i = u_i. \end{cases} \tag{2.4}$$

Note that the function $accur(x,\lambda) =: \max\{\mu(x,\lambda), \|g(x)\|\}$ measures the violation of the optimality conditions for the problem (2.1), and $accur(x,\lambda) = 0$ is equivalent to satisfying the first order optimality conditions.

Figure 1 describes the augmented Lagrangian method for solving problem (2.1).

1. Set $x \in B$, $\lambda = \lambda_0 = 0$, $rec = accur(x,\lambda)$.
   Select $k > 0$, $\epsilon > 0$, $0 < \theta < 1$, $\delta \ge 1$.
2. Find $\hat{x} \approx \underset{x \in B}{\operatorname{argmin}}\ \mathcal{L}_k(x,\lambda)$ with FPGM such that $\mu(\hat{x},\lambda) \le \epsilon/k$
3. Set $rec := accur(\hat{x},\lambda)$
4. Find $\hat{\lambda} = \lambda - kg(\hat{x})$.
5. Set $x := \hat{x}$, $\lambda := \hat{\lambda}$, $\epsilon := \theta\epsilon$, $k := \delta k$.
6. If $rec > RequiredAccuracy$ then Goto 2.
7. Stop.

FIGURE 1. Augmented Lagrangian Method

The minimization of the set B in Step 2 of the augmented Lagrangian algorithm is performed with the fast projected gradient method using FISTA [2] formulas for iterate update (see [6,7] for more details). The algorithm is shown in Figure 2.

Since $\mathcal{L}_k$ is a quadratic form with respect to x, $L = \|\nabla^2_{xx}\mathcal{L}_k(x,\lambda)\| = \|Q + kA^TA\|$, where the matrix spectral norm and is the largest singular value of a matrix, i.e. the constant that depends only on Q, A and the parameter k.

Note that the matrix-vector products Qx and $A^T\lambda$ are the most computationally expensive parts for the $\nabla_x \mathcal{L}_k(x,\lambda)$ calculation, which takes $\mathcal{O}(n^2)$ operations.

1. Input (x, λ).
2. Set $\bar{x} = x$, $t = 1$. Select $L > 0$.
3. Set $\hat{x} = P_B(x - \frac{1}{L}\nabla_x \mathcal{L}_k(x, \lambda))$
4. Set $\bar{t} = 0.5(1 + \sqrt{1 + 4t^2})$
5. Set $x = \hat{x} + (\hat{x} - \bar{x})(t - 1)/\bar{t}$
6. Set $\bar{x} = \hat{x}$, $t = \bar{t}$
7. If $\mu(\hat{x}, \lambda) > RequiredAccuracy$, Goto 3.
8. Output $\hat{x}$.

FIGURE 2. Fast Projected Gradient Method

The projection step P_B is shown in Figure 3 and requires only $\mathcal{O}(n)$ arithmetic operations.

1. Loop over all $i = 1, \ldots, n$.
2. If $x_i < l_i$ then Set $x_i = l_i$
3. If $x_i > u_i$ then Set $x_i = u_i$
4. Return x.

FIGURE 3. Operator P_B : Projection of $x \in \mathbb{R}^n$ onto the set B

Theoretical convergence analysis of FPGAL algorithm in Figure 1-3 can be found in [7].

3. Newton nonlinear rescaling - augmented Lagrangian algorithm

Another method we selected for computational experiments is based on a combination of nonlinear rescaling principle for treating inequality constraints with augmented Lagrangian method for handling equations. Unconstrained minimizations were performed by Newton's method.

Nonlinear rescaling principle was developed by Polyak [16]. It is the generalization of the modified barrier function method [14]. The main idea is that the functions that define inequality constraints are rescaled with with a transformation that has certain characteristics described below. As a result, the rescaled problem is equivalent to the original problem yet possesses additional useful properties that can be exploited. Therefore such rescaling mechanism is an alternative to the projection when treating simple bounds.

The rescaling of the functions that define constraints is accomplished with a transformation $\psi : \mathbb{R} \to \mathbb{R}$ with the following properties:

(1) $\psi'(t) > 0$, $\psi''(t) < 0$ for all $t \in \mathbb{R}$.

(2) $\psi(0) = 0$, $\psi'(0) = 1$.

(3) $\psi'(t) \leq a/(t + 1)$ for all $t \geq 0$ and some $a > 0$.

(4) $-\psi''(t) \leq b/(t + 1)^2$ for all $t \geq 0$ and some $b > 0$.

The examples of the transformations ψ that satisfy properties (1) - (4) can be found in [17, 20].

To conduct computational experiments we used a smooth combination of the modified barrier function and a parabola:

$$\psi(t) = \begin{cases} \ln(t+1), & t > -0.5 \\ -2t^2 + \ln(.5) + .5, & t \le -0.5. \end{cases} \tag{3.1}$$

Consider the functions $c_i(x) = x_i - l_i$, $i = 1, \ldots, n$, $c_i(x) = u_{i-n} - x_{i-n}$, $i = n+1, \ldots, 2n$. The the bounded set B can be described as follows:

$$B = \{x \in \mathbb{R}^n : l_i \le x_i \le u_i,\ i = 1, \ldots n\}$$

$$= \{x \in \mathbb{R}^n : c_i(x) \ge 0,\ i = 1, \ldots 2n\}.$$

The QP (2.1) can be restated as

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}}\ f(x) \\ &\text{subject to}\ g(x) = 0, \\ &\qquad\qquad c(x) \ge 0, \end{aligned} \tag{3.2}$$

where $c(x) = (c_1(x), \ldots c_n(x), c_{n+1}(x), ..., c_{2n}(x))^T$ is a vector function.

The nonlinear transformation ψ rescales functions c_i, $i = 1, ..., 2n$ without changing set B :

$$B = \{x \in \mathbb{R}^n : \psi(kc_i(x))/k \ge 0,\ i = 1, \ldots 2n\},$$

where $k > 0$ is a fixed scaling parameter.

Consider the Lagrangian for the rescaled problem

$$\text{Ł}(x, \lambda, \mu) = f(x) - \sum_{i=1}^{2n} \mu_i \psi(kc_i(x))/k - \lambda^T g(x)$$

and the augmented Lagrangian

$$\mathcal{L}_k(x, \lambda, \mu) = f(x) - \sum_{i=1}^{2n} \mu_i \psi(kc_i(x))/k - \lambda^T g(x) + 0.5kg(x)^T g(x).$$

Here $\mu = (\mu_1, \ldots, 2n) \in \mathbb{R}^{2n}_+$ is a vector of the Lagrange multipliers that correspond to the inequality constraints, while $\lambda \in \mathbb{R}^m$, as previously, the vector of the Lagrange multipliers that correspond to the equations.

The nonlinear rescaling - augmented Lagrangian (NRAL) method consists of a sequence of inexact unconstrained minimizations of $\mathcal{L}_k(x, \lambda)$ in x on the $\mathbb{R}^n$

$$\hat{x} \approx \hat{x}(\lambda, \mu) = \underset{x \in \mathbb{R}^n}{\text{argmin}}\ \mathcal{L}_k(x, \lambda, \mu). \tag{3.3}$$

followed by updating the Lagrange multipliers:

$$\hat{\mu}_i = \mu_i \psi'(kc_i(\hat{x})),\ i = 1, \ldots, 2n.$$

$$\hat{\lambda} = \lambda - kg(\hat{x}).$$

The algorithm performs an unconstrained minimization of a smooth function $\mathcal{L}_k$. Therefore $\|\nabla \mathcal{L}_x(\hat{x}, \lambda, \mu)\| \le \varepsilon$ can be used for the stopping criteria of the unconstrained subproblem (3.3). A function $\nu(x, \lambda, \mu)$ that measures the violation of the

KKT conditions

$$\nu(x,\lambda,\mu)=\max\left\{\|\nabla_x L(x,\lambda,\mu)\|, c(x)^T\mu, \max_{1\le i\le 2n}\{-c_i(x)\}, \|g(x)\|\right\}$$

can be used for the stopping condition of the NRAL method.

Figure 4 describes the nonlinear rescaling augmented Lagrangian method for solving problem (3.2).

1. Set $x\in\mathbb{R}^n$, $\lambda=0\in\mathbb{R}^m$, $\mu=(1,\dots,1)^T\in\mathbb{R}^{2n}_{++}$, $r=\nu(x,\lambda,\mu)$. Select $k>0$, $\epsilon>0$, $0<\theta<1$, $\delta\ge 1$.
2. Find $\hat{x}\approx \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\ \mathcal{L}_k(x,\lambda,\mu)$ with NM: $\|\nabla_x\mathcal{L}_k(\hat{x},\lambda,\mu)\|\le\epsilon/k$
3. Find $\hat{\mu}_i=\mu_i\psi'(kc_i(\hat{x}))$, $i=1,\dots,2n$.
4. Find $\hat{\lambda}=\lambda-kg(\hat{x})$.
5. Set $x:=\hat{x}$, $\lambda:=\hat{\lambda}$, $\mu=\hat{\mu}$, $r:=\nu(x,\lambda,\mu)$, $\epsilon:=\theta\epsilon$, $k:=\delta k$.
6. If $r>$ *RequiredAccuracy* then Goto 2.
7. Stop.

FIGURE 4. Nonlinear Rescaling - Augmented Lagrangian Method

Since Q is positive definite, the problem (3.2) satisfies the standard second-order optimality conditions. Therefore for the sequence of the primal-dual triples $z^s=(x^s,y^s,z^s)$ generated by the NRAL method and k large enough, the Q-linear rate of convergence takes place, i.e. the following bound holds:

$$\|z^{s+1}-z^*\|\le\frac{\sigma}{k}\|z^s-z^*\|, \tag{3.4}$$

where $z^{s+1}=(x^{s+1},\lambda^{s+1},\mu^{s+1})$ and $z^s=(x^s,\lambda^s,\mu^s)$ are two consecutive primal-dual iterates of the NRAL method shown in Figure 4, and $\sigma>0$ is a constant related to problem data (see [20] for more detail). From the convergence of $\{z^s\}$, in

1. Input (x,λ,μ).
2. Set $t=1$, $\hat{g}=\nabla_x\mathcal{L}_k(x,\lambda,\mu)$
3. Solve the system $\nabla^2_{xx}\mathcal{L}_k(x,\lambda,\mu)\Delta x=-\hat{g}$ for Δx.
4. While $\mathcal{L}_k(x+t\Delta x,\lambda,\mu)>\mathcal{L}_k(x,\lambda,\mu)+0.1t\hat{g}^T\Delta x$
   $t:=t/2$
5. Set $x=x+t\Delta x$
6. If $\|\nabla_x\mathcal{L}_k(x,\lambda,\mu)\|>$ *RequiredAccuracy*, Goto 2.
7. Output x.

FIGURE 5. Newton's method for minimizing $\mathcal{L}_k$

particular, follows its boundedness.

Figure 5 describes Newton's method (NM) used in Step 2 of the nonlinear rescaling-augmented Lagrangian algorithm. Convergence properties of Newton's method mainly depend on the Lipschitz constant $M(\lambda^s,\mu^s)>0$ for the Hessian $\nabla^2_{xx}\mathcal{L}_k(x,\lambda^s,\mu^s)$,

and the strong convexity constant $m(\lambda^s,\mu^s)$ of $\mathcal{L}_k(x(\lambda^s,\mu^s),\lambda^s,\mu^s)$, where $x(\lambda^s,\mu^s) = \operatorname{argmin}_{x\in\mathbb{R}^n}\mathcal{L}_k(x,\lambda^s,\mu^s)$. The existence of the Lipschitz constant $M(\lambda^s,\mu^s)$ follows from boundedness of the closed set $S(x^s,\lambda^s,\mu^s) = \{x : \mathcal{L}_k(x,\lambda^s,\mu^s) \le \mathcal{L}_k(x^s,\lambda^s,\mu^s)\}$ for any $\lambda^s \in \mathbb{R}^m$, $\mu^s \in \mathbb{R}^{2n}_{++}$, and $x^s \in \mathbb{R}^n$. Further, from the convergence and hence boundedness of $\{z^s\} = \{(x^s,\lambda^s,\mu^s)\}$ of the sequence generated by the nonlinear rescaling - augmented Lagrangian method in Figure 5 and the continuity in x of $\mathcal{L}_k(x,\lambda,\mu)$ follows the existence of $M < \infty$ such as $\sup_s M(\lambda^s,\mu^s) \le M$.

The strong convexity constant $m(\lambda^s,\mu^s)$ of $\mathcal{L}_k(x,\lambda^s,\mu^s)$ is separated from zero by the minimum eigenvalue m of matrix $Q : m(\lambda,\mu) \ge m > 0$. Therefore in the neighborhood of $x(\lambda^s,\mu^s)$ Newton's method has a quadratic rate of convergence and the following bound holds:

$$\|x_{s+1} - x(\lambda^s,\mu^s)\| \le \frac{M}{2(m - M\|x_s - x(\lambda^s,\mu^s)\|)}\|x_s - x(\lambda^s,\mu^s)\|^2, \tag{3.5}$$

where x_s, x_{s+1} are two consecutive iterates of Newton's method shown in Figure 5 (see [20] for the proof of (3.5)).

4. Numerical experiments

For numerical experiments we selected the dual problem for training soft margin support vector machines (SVM) with Gaussian kernels (see [24] for details).

Let $\{(z_1,y_1),\dots,(z_n,y_n)\}$ be a set of n argument-value training examples of an unknown binary function that the SVM needs to approximate. Here $z_i \in \mathbb{R}^p$, $i = 1,\dots,m$ are examples of function arguments and $y_i \in \{-1,1\}$ represent the corresponding function values, or labels indicating the class to which z_i belongs. The dual form of the soft margin SVM problem finds support vectors by solving the following quadratic QP problem for $x^* = (x_1^*,\dots,x_n^*)^T$ [24].

$$\begin{aligned} &\text{minimize } f(x) = \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n y_i y_j x_i x_j K(z_i,z_j) - \sum_{i=1}^n x_i \\ &\text{subject to } g(\alpha) = \sum_{i=1}^n y_i x_i = 0, \\ &\qquad 0 \le x_i \le C, i = 1,\dots,n, \end{aligned} \tag{4.1}$$

where $K(\cdot,\cdot)$ is a Kernel function.

To show that problem (4.1) can be formulated as problem (2.1), let us introduce the following bounded set:

$$B = \{x \in \mathbb{R}^n : 0 \le x_i \le C,\ i = 1,\dots n\}.$$

k	Training time (sec)	# of Iterations	# of minimizations on B
2^{-5}	> 100	reached iter limit	reached iter limit
2^{-4}	> 100	reached iter limit	reached iter limit
2^{-3}	> 100	reached iter limit	reached iter limit
2^{-2}	32.39	7192	22
2^{-1}	24.96	5924	12
2^{0}	24.92	5863	8
2^{1}	25.56	6056	5
2^{2}	38.03	9085	4
2^{3}	56.78	13222	3
2^{4}	123.38	28643	3
2^{5}	201.70	47535	2

TABLE 1. Training SVM with FPGAL method with different k ($n = 1000$)

Then the optimization problem (4.1) can be rewritten as follows:

$$\begin{aligned} &\underset{x\in B}{\text{minimize}}\ f(x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j x_i x_j K\left(z_i, z_j\right) - \sum_{i=1}^{n} x_i \\ &\text{subject to } g(x) = \sum_{i=1}^{n} y_i x_i = 0, \end{aligned} \tag{4.2}$$

or equivalently as

$$\begin{aligned} &\underset{x\in B}{\text{minimize}}\ f(x) \\ &\text{subject to } g(x) = 0, \end{aligned} \tag{4.3}$$

where

$$f(x) = \frac{1}{2}x^T Q x - e^T x,$$
$$g(x) = y^T x,$$

where Q is an $n \times n$ matrix with the elements $q_{ij} = y_i y_j K(z_i, z_j)$, $i, j = 1, \ldots, n$, $e = (1, \ldots, 1)^T \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ is a vector with the entries made of 1 and -1. Using Gaussian kernels with non-repeating data points ensures that Q is positive definite. In other words, problem (4.3) is a particular representative of a general problem (2.1).

We selected SVM problem (4.1) for testing experiments mainly because it is easy to change the size of the problem while keeping all the other characteristics of the problem similar. All we have to do is to select a different number of data points n from the same large data set. A particular choice of a data set is not important as long as it has a sufficient number of data points, so we can observe how the solution

time scales up. The data we use are HSLS09 study of high school student indicators (https://nces.ed.gov/surveys/hsls09/), where the predicted binary variable $y_i \in \{-1, 1\}$ represent the label indicating where a high student attended a college or not. The SVM was trained to predict the college attendance using a vector of high school student descriptors $z_i \in \mathbb{R}^{501}$. The Gaussian kernel function was selected $K(z_i, z_j) = \exp(-\gamma\|z_i - z_j\|^2)$. The parameters $\gamma = 0.004$ and $C = 10$ were selected to achieve high SVM predictive accuracy. The accuracy in the stopping criteria of optimization algorithms was selected as $RequiredAccuracy = 0.001$. We implemented both algorithms in Matlab "from scratch" except the subroutine for solving linear systems of equations, for which we used a built in Matlab solver. We used a Windows laptop with Intel Xeon CPU E3-1535M v6 to run computational experiments.

To select a good value for the scaling parameter for FPGAL method we ran a few experiments for

$$k = 2^l,\ l = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5.$$

The results are shown in Table 4. The cases with $k = 2^{-5}, 2^{-4}, 2^{-3}$ took too much time and hit an iteration limit. This happened because when the value of k is too small, each minimization on B does not produce a good enough progress of dual iterates λ towards the dual solution λ^*. The value of $k = 1$ provides an optimal setting. This is consistent with results obtained in the previous study [6].

We also checked the range of k from 1 to 200 to investigate how the training SVM by FPGAL scales up with further increase of k. Figure 6 shows that solving time scales up as $\mathcal{O}(\sqrt{k})$. The result can be explained by the estimate for the number of iterations that is needed to solve the problem. As the scaling parameter grows, most of the computational effort goes to finding the first minimization (2.2) on B. According to [19] for the first minimization, for the iterates x_s, $s = 0, 1, \ldots$, the following bound takes place:

$$\mathcal{L}_k(x_s, \lambda_0) - \mathcal{L}_k(x(\lambda_0), \lambda_0) \leq \frac{2L\|x_0 - x(\lambda_0)\|^2}{(s+1)^2}, \tag{4.4}$$

where $x(\lambda_0)$ is the solution to the first minimization problem (2.2), L is the Lipschitz constant for $\nabla_x \mathcal{L}_k(x, \lambda_0)$ such that

$$\|\nabla_x \mathcal{L}_k(x_1, \lambda_0) - \nabla_x \mathcal{L}_k(x_2, \lambda_0)\| \leq L\|x_1 - x_2\|$$

for any $x_1, x_2 \in \mathbb{R}^n$.

Since $\mathcal{L}_k(x, \lambda)$ is a quadratic form in x, its Hessian

$$\nabla^2_{xx} \mathcal{L}_k(x, a, \lambda) = Q + kyy^T,$$

and the spectral norm

$$L = \|Q + kyy^T\| \tag{4.5}$$

grows linearly with increase of k. Therefore, according to (4.4), the iteration count s must grow as $\mathcal{O}(\sqrt{L}) = \mathcal{O}(\sqrt{k})$ to counter the linear growth of the numerator of the right-hand-side of (4.4) to achieve the same level of accuracy of the left-hand-side of (4.4). Similar scalings of L take place on all subsequent minimization on

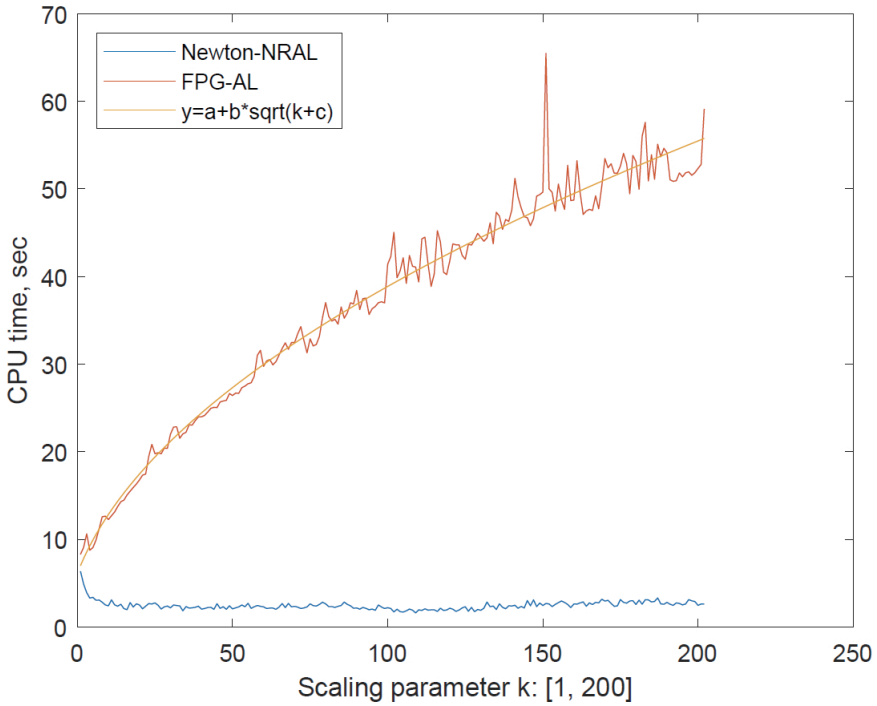

FIGURE 6. CPU times for training SVM with FPGAL and NNRAL as a function of the scaling parameter k, number of training points $n = 1000$

B. Therefore the growth of solving time $\mathcal{O}(\sqrt{k})$ that one can see in Figure 6 is expected.

The NNRAL method allows more freedom in selecting the scaling parameter k as Figure 6 suggests. Any value of k from 50 to 200 results in about the same solution time. So we selected $k = 100$ for the NNRAL experiments.

We trained SVM with both FPGAL and NNRAL methods to the accuracy of 10^{-3} for n ranging from 1000 to 15000 training examples and recorded the solution times. Then we created logarithmic plots as shown in Figure 7.

The logarithmic plots are useful because if they show a linear growth of solution times, that means the actual growth is polynomial $\mathcal{O}(n^d)$ with a slope approximating the degree d of the polynomial. The height of the line $\log H$ (y-intercept) is the logarithm of a coefficient before the highest degree d :

$$s.t.(solution\,time) \approx Hn^d. \tag{4.6}$$

Indeed, by taking the logarithm of both side of (4.6) we have a linear law in $\log n$:

$$\log(s.t.) \approx \log H + d \log n. \tag{4.7}$$

First observation is that for both FPGAL and NNRAL methods the logarithmic plots follow well linear pattern suggesting that the scaling of both algorithms is closed to polynomial. The slope of each line approximates the highest degree of the polynomial. The larger n, the more accurately the slope approximates the degree

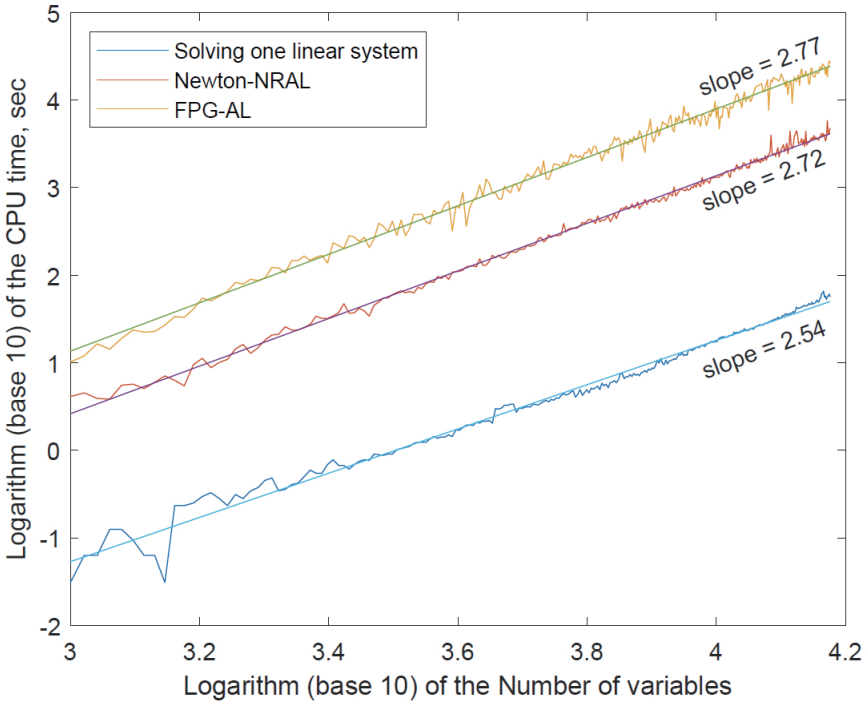

FIGURE 7. CPU times for training SVM with FPGAL and Newton NRAL as a function of the number of training points $n = 1000, 1050, 1100, \ldots, 15000$. The times for solving a linear system of thee size n also provided for a reference.

d in the formula $s.t. = \mathcal{O}(n^d)$. Note that the slopes of the graphs for both FPGAL and NNRAL are less than 3 (2.8 and 2.7 respectively), which is the consequence of using many problems with relatively small n for the slope approximation. To test this hypothesis, we looked into how solving a single linear symmetric system with the SVM matrix Q scales up. The corresponding graph is shown in blue with the slope approximation of 2.54, which is less than the theoretical value of 3, since linear systems scale up as $\mathcal{O}(n^3)$.

The explanation for this discrepancy is in the presence of lower order terms in the accurate count of number of arithmetic operations. For example, to solve a linear system, the formula for the flop count is a polynomial of degree 3:

$$p(n) = c_3 n^3 + c_2 n^2 + c_1 n + c_0.$$

When n is small, the terms such as $c_2 n^2$ simply cannot be ignored and result in the underestimation of the highest degree of 3. In order to be able to ignore the terms $c_2 n^2 + c_1 n + c_0$, we need to consider very large n. Then the number of arithmetic operations will be more accurately approximated with the highest degree term $c_3 n^3$.

Therefore to improve the accuracy of the degree estimate we consider only very large available n. For example, in the range of 10000 and 15000, the approximation of the iteration count $p(n) \approx c_3 n^3$ should become more accurate.

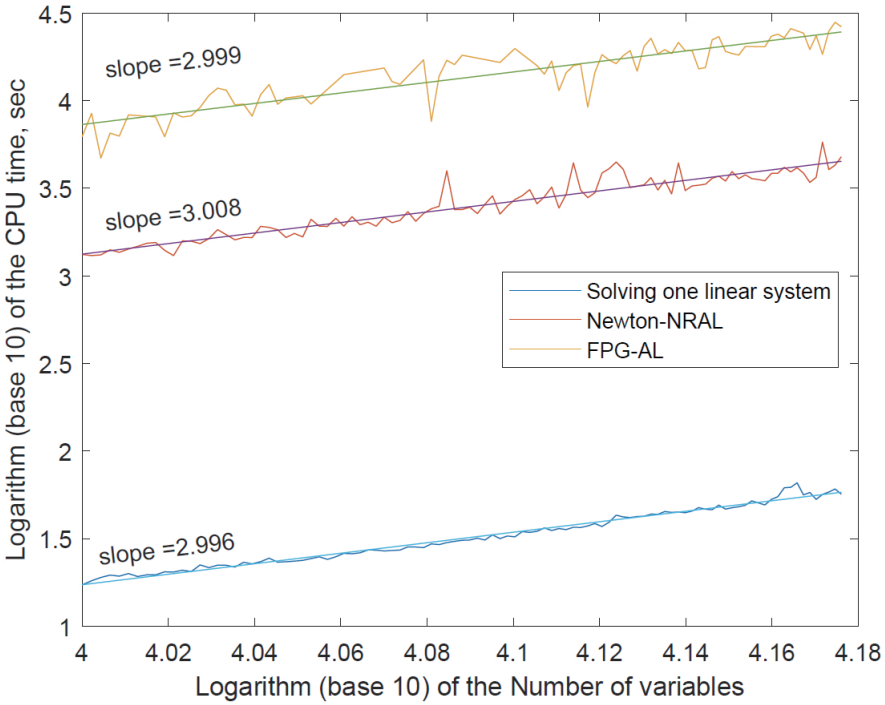

FIGURE 8. CPU times for training SVM with FPGAL and Newton NRAL as a function of the number of training points $n = 10000, 10050, 10100, \ldots, 15000$. The times for solving a linear system of thee size n also provided for a reference.

Indeed, Figure 8 demonstrates that the slopes for the line that shows the scaling of solving a single linear system of equations is about 3. That gives us some confidence to assume that in the case of large n the slopes for scaling FPGAL and NNRAL are also accurate: 2.999 for FPGAL, and 3.008 for NNRAL. Therefore the numerical experiments demonstrate that the the solution time for both FPGAL and NNRAL methods scales up as $\mathcal{O}(n^3)$.

5. Discussion and concluding remarks

There are several conclusions we can draw from the described numerical experiments.

First, we observed that the time it takes to train SVM with the NNRAL method scales up approximately as $\mathcal{O}(n^3)$. The observation can be explained by the fact that it takes approximately the same number of Newton steps to solve large and small problems. A small growth in a number of Newton steps is still observed, from an average of about 90 Newton steps for the smallest half $n = 1000, ..., 8000$, to an average of about 116 Newton steps for the largest half $n = 8000, ..., 15000$. However, this growth is negligible. Newton's method is mainly immune to the growth of the Lipschitz constant for $\nabla_x \mathcal{L}_k(x, \lambda, \mu)$. Convergence properties of Newton's method mainly depend on the Lipschitz constant M for the Hessian $\nabla^2_{xx} \mathcal{L}_k(x, \lambda, \mu)$, and the

strong convexity constant m of $\mathcal{L}_k(x,\lambda,\mu)$. For SVM problem (4.1), it can be shown that M depends only on the value of the scaling parameter k, Lagrange multipliers μ, and the value of $\psi'''(kc_i(x))$. The Lagrange multipliers μ are bounded since NRAL converges. Moreover, we do not observe the growth of the Lagrange multiplier with the increase of n. Also because of convergence, we have $kc_i(x) \geq -0.5$ in the neighborhood of the solution. Therefore, according to (3.1), $\psi'''(kc_i(x)) \leq 16$, i.e. $\psi'''(kc_i(x))$ is also bounded for any n. Therefore the Lipschitz constant M is not expected to grow with n.

The strong convexity constant m may worsen a little (become smaller) when we increase the number of data points. As the number of data points grows, then we may have more data points, which are close in distance. That may result in the decrease of the smallest eigenvalue of Q.

Also, as the number of data points n grows,we may have more points close to the separating hyperplane, which may result having a larger constant σ in (3.4). Therefore the number of unconstrained minimizations with Newton's method may increase a little.

However, we emphasize that the resulting increase of the number of Newton steps is insignificant and not systematic as shown in Figure 9. So it can be ignored.

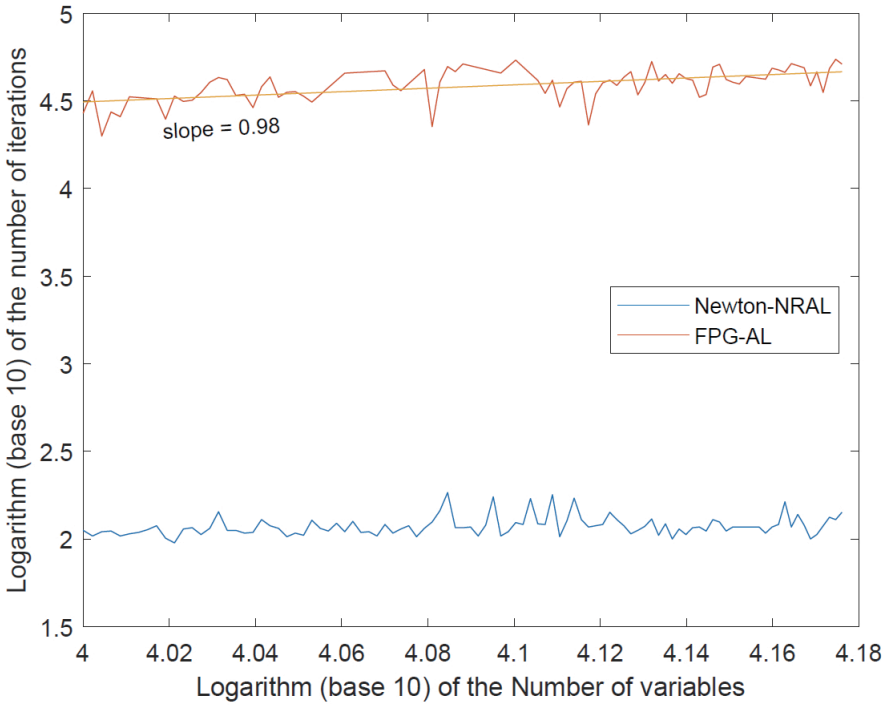

FIGURE 9. Number of iterations for training SVM with FPGAL and Newton NRAL as a function of the number of training points $n = 10000, 10050, 10100, \ldots, 15000$.

Unlike Newton's method, the gradient based first-order methods experience an increase of the number of iterations when n grows. Figure 9 suggests that the

number of iterations scales up as $\mathcal{O}(n)$. This can be explained using formula (4.4). For the SVM problem the Lipschitz constant L grows linearly with n as suggest formula (4.5) together with the fact that the Gaussian kernel is used. Assuming that $x_0 = 0$, the term $\|x_0 - x(\lambda_0)\|^2$, also grows linearly with n. So the numerator in the right-hand-side of (4.4) grows as $\mathcal{O}(n^2)$. Therefore to compensate for this growth and to achieve the same level of accuracy in the left-hand-side of (4.4) the number of iterations must grow as $\mathcal{O}(n)$. Similar considerations are applied to all minimizations on the set B. It has been observed that the number of minimizations on the set B does not grow with the increase of n. Therefore the total solution time grows linearly also as $\mathcal{O}(n)$ consistently with the observation in Figure (9).

It takes $\mathcal{O}(n^2)$ of arithmetic operations to perform each iteration of the FPGAL, since the most computationally expensive part is the calculations of the gradient $\nabla f(x) = Qx + q$. Therefore the the training of SVM by FPGAL method should scale up as $\mathcal{O}(n^3)$, which is confirm by our experiment.

At the same time the coefficient H before the highest degree $p(n) \approx Hn^3$ is larger for the FPGAL than that for NNRAL methods, according to Figure 8 suggesting that NNRAL method is generally faster on SVM problems (at least for our implementation of the algorithms). By fitting the time data to the model (4.7), we found that $\log_{10} H = -8.91$ for the NNRAL method, and $\log_{10} H = -8.13$ for the FPGAL algorithm suggesting, suggesting that the NNRAL method is about $10^{-8.13+8.91} = 10^{0.78} \approx 6$ times faster for problems of the same size.

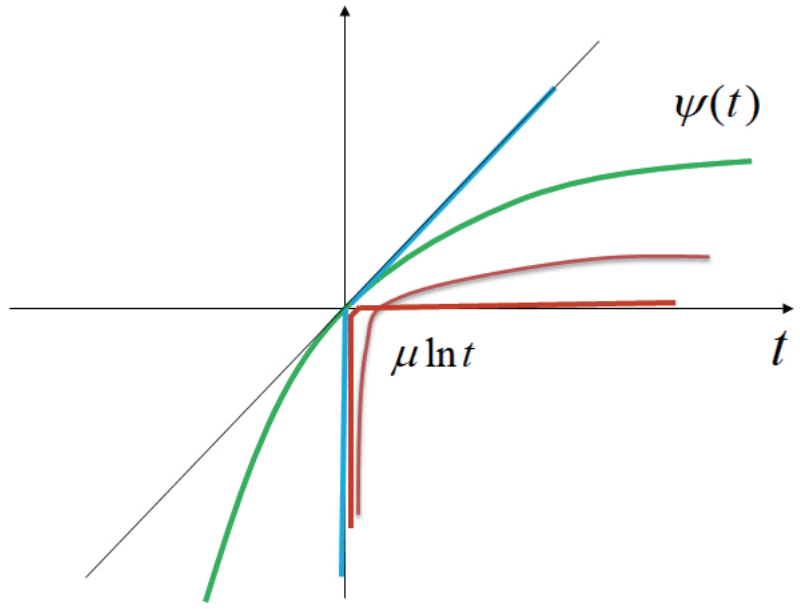

FIGURE 10. Transformation of constraints: no transformation (black), classical barrier (red), projection (blue), nonlinear rescaling (green).

Finally, to put another spin on the question whether one should use projection or rescaling, it is worth mentioning that a projection also can be viewed as a rescaling

transformation as shown in Figure 10. The projection transformation is shown blue, while the NR smooth transformation is shown green. Therefore the projection can be viewed as particular transformation of the functions c_i.

To complete the picture, the classical barrier function (shown red in Figure 10) also can be viewed as a rescaling transformation. In particular, it can be seen that only when the barrier parameter μ is closed to zero, the classical barrier transformation can be used to find a solution, since we need the transformation to go through the origin to deal with the active constraints. Finally "no rescaling transformation" case, or $\psi(t) = t$ is shown black. As one can see, the NR transformation that possesses properties (1) - (4) described earlier is the only nonlinear and smooth at the solution where $c_i(x^*) = 0$ for the active constraints.

Thus, the question whether one should project or rescale is rhetorical, because all three approaches mentioned in the introduction is one or another form of rescaling. A better question posed is how the functions that define constraints need to be rescaled? While various rescaling methods can be used, only nonlinear rescaling with properties (1) - (4) allows using second-order well conditioned approximations at the solution.

Even though our version NNRAL is faster than FPGAL method, still it takes a long time to train large SVM even with NNRAL method. Therefore methods based on selection of smaller working sets of data points is a promising direction of further improving SVM training efficiency. Some preliminary results [1] suggest that using efficient working set selection techniques with FPGAL allow to attain significant training improvement over the FPGAL that uses all the data points simultaneously. Based on the results of this manuscript, we believe that using NRAL with decomposition techniques can results in efficient new algorithms for solving large scale QP in general and SVM training in particular.

In the future we plan to gain more insight on a theoretical confirmation of the hypothesis $\mathcal{O}(n^3)$ for both methods studied here. We also would like to conduct more numerical experiments for other classes of nonlinear problems and other nonlinear optimization methods. Finally, we would like to develop and study decomposition optimization methods for general QP inspired by ideas developed by solving the QP problem for the SVM training.

References

[1] M. Aregbesola and I. Griva, *Augmented Lagrangian - fast projected gradient algorithm with working set selection for training support vector machines*, Journal of Applied Numerical Optimization **3** (2021), 3–20.

[2] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci. **2** (2009), 183–202.

[3] D. Goldfarb and S. Lium *An $O(n^3L)$ primal interior point algorithm for convex quadratic programming*, Mathematical Programming **49** (1991), 325–340.

[4] I. Griva, D. Shanno, R. Vanderbei and H. Benson, *Global convergence of a prima-dual interior-point method for nonlinear programming*, Algorithmic Operations Research **3** (2008), 12–29.

[5] I. Griva and R. Polyak, *1.5-Q-superlinear convergence of an exterior-point method for constrained optimization*, *J. Global Optimization* **40** (2008), 679–695.

[6] V. Bloom, I. Griva and F. Quijada, *Fast projected gradient method for support vector machines*, Optimization and Engineering **17** (2016), 651–662.

[7] I. Griva, *Convergence analysis of augmented Lagrangian - fast projected gradient method for convex quadratic problems*, Pure and Applied Functional Analysis **3** (2018), 417–428.
[8] A. V. Fiacco and G. P. Mccormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York, 1968.
[9] M. R. Hestenes, *Multiplier and gradient methods*, J. Optimization Theory and Applications **4** (1969), 303–320.
[10] R. Monteiro and I. Adler, *Interior path-following primal-dual algorithm. Part 2. convex quadratic programming*, Mathematical Programming **44** (1989), 43–46.
[11] S. Mehrotra and J. Sun, *An algorithm for convex quadratic programming that requires $O(n^{3.5}L$ arithmetic operations*, Mathematics of Operations Research **15** (1990), 342–363.
[12] Y. Nesterov, *Introductory Lectures on Convex Optimization*, Kluwer, Dordrecht, 2004.
[13] Y. Nesterov and A. Nemirovsky, *Acceleration and parallelization of the path-following interior-point method for a linearly constrained convex quadratic problem*, SIAM Journal on Optimization **1** (1991), 548–564.
[14] R. A. Polyak, *Modified barrier functions (theory and methods)*, Mathematical Programming **54** (1992), 177–222.
[15] R. A. Polyak and M. Teboulle, *Nonlinear rescaling and proximal-like methods in convex programming*, Mathematical Programming **76** (1997), 265–284.
[16] R. A. Polyak, *Nonlinear rescaling vs. smoothing technique in constrained optimization*, Mathematical Programming **92** (2002), 197–235.
[17] R. A. Polyak, *Nonlinear rescaling multipliers method as interior quadratic prox*, Computational Optimization and Applications **35** (2006), 347–373.
[18] R. A. Polyak, J. Costa and S. Neyshabouri, *Dual fast projected gradient method for quadratic programming*, Optimization Letters **7** (2013), 631–645.
[19] R. A. Polyak, *Projected gradient method for non-negative least square*, Contemporary Mathematics **636** (2015), 167–179.
[20] R. A. Polyak, *Introduction to Continuous Optimization*, Springer, Switzerland, 2021.
[21] M. J. D. Powell, *A method for Nonlinear Optimization in Minimization Problems*, Optimization,R. Fletcher, ed., Academic Press, New York, 1969, pp. 283–298.
[22] R. T. Rockafellar, *Augmented lagrangians and applications of the proximal point algorithm in convex programming*, Mathematics of Operations Research **1** (1976), 97–116.
[23] R. J. Vanderbei and D. F. Shanno, *An interior-point algorithm for nonconvex nonlinear programming*, Computational Optimization and Applications **13** (1999), 231–252.
[24] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed., Springer, 2000.
[25] Y. Ye and E. Tse, *An extension of Karmarkar's projective algorithm for convex quadratic programming*, Nonlinear Programming **44** (1989), 157–179.

Manuscript received June 3 2022
revised September 10 2022

Igor Griva
Department of Mathematical Sciences, George Mason University, Fairfax, VA 22030, USA
E-mail address: igriva@gmu.edu