Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Efficient Multi-Stage Conjugate Gradient for Trust Region Step

Pinghua Gong and **Changshui Zhang**
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Automation, Tsinghua University, Beijing 100084, China
{gph08@mails, zcs@mail}.tsinghua.edu.cn

Abstract

The trust region step problem, by solving a sphere constrained quadratic programming, plays a critical role in the trust region Newton method. In this paper, we propose an efficient Multi-Stage Conjugate Gradient (MSCG) algorithm to compute the trust region step in a multi-stage manner. Specifically, when the iterative solution is in the interior of the sphere, we perform the conjugate gradient procedure. Otherwise, we perform a gradient descent procedure which points to the inner of the sphere and can make the next iterative solution be a interior point. Subsequently, we proceed with the conjugate gradient procedure again. We repeat the above procedures until convergence. We also present a theoretical analysis which shows that the MSCG algorithm converges. Moreover, the proposed MSCG algorithm can generate a solution in any prescribed precision controlled by a tolerance parameter which is the only parameter we need. Experimental results on large-scale text data sets demonstrate our proposed MSCG algorithm has a faster convergence speed compared with the state-of-the-art algorithms.

1 Introduction

The trust region Newton method (TRON) has received increasing attention and it has been successfully applied to many optimization problems in artificial intelligence and machine learning communities (Lin, Weng, and Keerthi 2008; Kim, Sra, and Dhillon 2010; Yuan et al. 2010). The trust region Newton method minimizes an objective function $l(\mathbf{w})$ by generating the solution at the $(k+1)$-th iteration via $\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{d}^k$; $\mathbf{d}^k$ is a trust region step computed by the following trust region step problem:

$$\min_{\mathbf{d}\in\mathbb{R}^n} \{f(\mathbf{d}) = \frac{1}{2}\mathbf{d}^T H^k \mathbf{d} + (\mathbf{g}^k)^T \mathbf{d}\} \; s.t. \; \|\mathbf{d}\| \leq \lambda_k, \quad (1)$$

where $\mathbf{g}^k$ and H^k are respectively the gradient and the Hessian matrix of the objective function; λ_k is the trust region radius controlling the size of the sphere constraint ($\|\mathbf{d}\| \leq \lambda_k$). We accept $\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{d}^k$ only if $\mathbf{d}^k$ makes the ratio $(l(\mathbf{w}^{k+1}) - l(\mathbf{w}^k))/f(\mathbf{d}^k)$ large enough. Otherwise, we update λ_k and compute $\mathbf{d}^k$ until the above ratio is large enough (Lin and Moré 1999; Lin, Weng, and Keerthi 2008; Yuan et al. 2010). In the trust region Newton method, a key issue is how to efficiently compute the trust region step in Eq. (1), which is the focus of this paper.

Existing algorithms for solving Eq. (1) can be broadly classified into two categories. The first category reformulates Eq. (1) as other optimization problems such as the root finding (Mor and Sorensen 1983), the parameterized eigenvalue finding (Rojas, Santos, and Sorensen 2000) and the semi-definite programming (Rendl and Wolkowicz 1997; Fortin and Wolkowicz 2004). The second category directly solves Eq. (1), which includes the conjugate gradient (Steihaug 1983), the Gauss quadrature technique (Golub and Von Matt 1991) and the subspace minimization (Hager 2001; Erway, Gill, and Griffin 2009; Erway and Gill 2009). An interesting one among these algorithms is Steihaug's algorithm (Steihaug 1983), which utilizes the conjugate gradient method to obtain an *approximate* solution quickly. However, the conjugate gradient procedure in Steihaug's algorithm terminates once its iterative solution first reaches the boundary of the sphere. Thus, Steihaug's algorithm may terminate even if it *doesn't converge* and the precision of the solution can't be specified by users (Gould et al. 1999). But in practice, different applications may have different requirements for the precision of the trust region step. Thus, it's preferred that the precision of the solution can be controlled by a parameter (Erway, Gill, and Griffin 2009).

Recently, a class of first-order algorithms which can efficiently solve Eq. (1) attract considerable attention. In artificial intelligence and machine learning communities, they are often used to solve large-scale optimization problems with simple constraints (e.g., sphere constraint) (Shalev-Shwartz, Singer, and Srebro 2007; Figueiredo, Nowak, and Wright 2007; Bach et al. 2011). The most typical algorithms include Projected Gradient (PG) algorithm (Lin 2007; Duchi et al. 2008; Daubechies, Fornasier, and Loris 2008; Wright, Nowak, and Figueiredo 2009) and Accelerated Projected Gradient (APG) algorithm (Nesterov 2004; Liu, Ji, and Ye 2009b; 2009a; Beck and Teboulle 2009; Gong, Gai, and Zhang 2011; Yuan, Liu, and Ye 2011; Liu, Sun, and Ye 2011). Due to the simplicity of projection a vector onto the sphere constraint, both PG and APG are efficient algorithms for solving the trust region step problem in Eq. (1), especially in large-scale scenarios. However, both algorithms are general optimization techniques and they don't consider that the objective function in Eq. (1)

is quadratic.

In this paper, we propose an efficient Multi-Stage Conjugate Gradient (MSCG) algorithm to solve the trust region problem in a multi-stage manner. The main contributions of this paper include:

(1) We propose an efficient Multi-Stage Conjugate Gradient (MSCG) algorithm by extending the conjugate gradient algorithm from the *unconstrained* optimization to the *sphere constrained* quadratic optimization in a multi-stage manner. Specifically, when the iterative solution is in the interior of the sphere, we perform the conjugate gradient procedure. Otherwise, we perform a gradient descent procedure which points to the inner of the sphere and can make the next iterative solution be a interior point. Subsequently, we proceed with the conjugate gradient procedure again. We repeat the above procedures until convergence.

(2) The MSCG algorithm can generate a solution in any prescribed precision controlled by a tolerance parameter which is the only parameter we need.

(3) We present a detailed theoretical analysis which shows that our proposed MSCG algorithm decreases the objective function value in each iteration and further guarantees the convergence of the algorithm. Moreover, empirical studies on large-scale text data sets demonstrate the MSCG algorithm has a faster convergence speed compared with the state-of-the-art algorithms.

The rest of this paper is organized as follows: In Section 2, we introduce some notations and preliminaries on the conjugate gradient method. In Section 3, we present the proposed MSCG algorithm. Experimental results are present in Section 4 and we conclude the paper in Section 5.

2 Preliminaries

Notations

We introduce some notations used throughout the paper. Scalars are denoted by lower case letters (e.g., $x \in \mathbb{R}$) and vectors by lower case bold face letters (e.g., $\mathbf{x} \in \mathbb{R}^n$). Matrix is denoted by capital letters (e.g., A) and the Euclidean norm of the vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^n x_i^2}$. $\nabla f(\mathbf{x})$ denotes the first derivative (gradient) of $f(\mathbf{x})$ at $\mathbf{x}$.

Conjugate Gradient

Conjugate gradient (CG) is an efficient method to solve the following *unconstrained* quadratic programming problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \{h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{b}^T\mathbf{x}\},$$

where $Q \in \mathbb{R}^{n\times n}$ is positive definite and $\mathbf{b} \in \mathbb{R}^n$. CG is an iterative method which generates the solution $\mathbf{x}^{i+1}$ at the $(i+1)$-th iteration given by

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \frac{\|\nabla h(\mathbf{x}^i)\|^2}{(\mathbf{p}^i)^T Q\mathbf{p}^i}\mathbf{p}^i,$$

where $\nabla h(\mathbf{x}^i)$ is the gradient of $h(\mathbf{x})$ at $\mathbf{x}^i$ and $\mathbf{p}^i$ is the conjugate gradient direction given by

$$\mathbf{p}^i = -\nabla h(\mathbf{x}^i) + \frac{\|\nabla h(\mathbf{x}^i)\|^2}{\|\nabla h(\mathbf{x}^{i-1})\|^2}\mathbf{p}^{i-1} \text{ with } \mathbf{p}^0 = -\nabla h(\mathbf{x}^0).$$

In theory, CG terminates with an optimal solution after at most n steps (Bertsekas 1999). However, in practice, CG often terminates after *much less* than n steps, especially when the data is high dimensional (n is large). Thus, CG is very efficient to solve the *unconstrained* quadratic optimization. However, extending CG to efficiently solve the *constrained* quadratic optimization is still a hard problem.

3 Proposed Multi-Stage Conjugate Gradient

In this section, we propose a Multi-Stage Conjugate Gradient (MSCG) algorithm to efficiently solve the trust region step problem. The MSCG algorithm extends the conjugate gradient method as in the traditional *unconstrained* quadratic optimization to solve the *sphere constrained* quadratic optimization in a *multi-stage* manner. We unclutter Eq. (1) by omitting the superscript as follows:

$$\min_{\mathbf{d}\in\mathbb{R}^n} \left\{ f(\mathbf{d}) = \frac{1}{2}\mathbf{d}^T H\mathbf{d} + \mathbf{g}^T\mathbf{d} \right\} \quad s.t.\ \|\mathbf{d}\| \le \lambda. \qquad (2)$$

In the subsequent discussion, we consider the case that H is positive definite, which is very common in real applications such as logistic regression, support vector machines (Lin, Weng, and Keerthi 2008) and the wavelet-based image deblurring problems (Beck and Teboulle 2009). Our proposed MSCG algorithm involves conjugate gradient (C procedure) and gradient descent procedures (G procedure). The MSCG algorithm calculates the iterative solution switching between them. Specifically, when the i-th iterative solution $\mathbf{d}^i$ is an interior point of the sphere, i.e., $\|\mathbf{d}^i\| < \lambda$, we perform C procedure. Otherwise, we turn to G procedure. The detailed MSCG algorithm is presented in Algorithm 1.

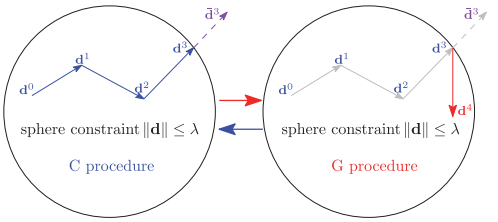

Figure 1: Illustration for the MSCG algorithm. The initial solution $\mathbf{d}^0$ is in the interior of the sphere and $\mathbf{d}^1, \mathbf{d}^2, \mathbf{d}^3$ are generated by C procedure. $\mathbf{d}^3$ is on the boundary of the sphere and we turn to G procedure which generates $\mathbf{d}^4$. Note that if the algorithm starts from $\mathbf{0}$ and terminates when the iterative solution ($\mathbf{d}^3$) first reaches the boundary of the sphere, the MSCG algorithm degrades into Steihaug's algorithm. Please refer to the text for more detailed explanations.

To better understand Algorithm 1, we illustrate a simple case of the MSCG algorithm in Figure 1. The initial solution $\mathbf{d}^0$ is in the interior of the sphere. Thus, we perform C procedure which generates the iterative solutions $\mathbf{d}^1$ and $\mathbf{d}^2$ lying in the interior of the sphere. When proceeding with C procedure, we obtain $\bar{\mathbf{d}}^3$ which exceeds the sphere constraint. At this time, we truncate $\bar{\mathbf{d}}^3$ such that the iterative solution $\mathbf{d}^3$ lies on the boundary of the sphere (line 21-24). Then, we turn to G procedure where the descent direction points to the inner of the sphere and G procedure generates the iterative solution $\mathbf{d}^4$ lying in the interior of the sphere. Thus,

we reset the conjugate gradient direction (line 14-15) and turn to C procedure. We repeat the above process between C procedure and G procedure until the algorithm converges.

Algorithm 1: MSCG: Multi-Stage Conjugate Gradient algorithm for trust region step problem

Input : $H \in \mathbb{R}^{n\times n},\ \mathbf{g}, \mathbf{d}^0 \in \mathbb{R}^n,\ \lambda > 0,\ 0 < \epsilon < 1$

1 **if** $\|\mathbf{d}^0\| < \lambda$ **then**
2 $\quad flag = 1;$
3 **else**
4 $\quad \mathbf{d}^0 = \frac{\lambda}{\|\mathbf{d}^0\|}\mathbf{d}^0;\ flag = 0;$
5 **end**
6 $reset = flag;$
7 **for** $i = 0, 1, \cdots$ **do**
8 $\quad \mathbf{q}^i = \nabla f(\mathbf{d}^i) + \frac{\|\nabla f(\mathbf{d}^i)\|}{\lambda}\mathbf{d}^i;$
9 $\quad$ **if** $\|\mathbf{q}^i\| \le \epsilon\|\mathbf{g}\|$ **then**
10 $\quad\quad \mathbf{d}^\star = \mathbf{d}^i,\ iter = i;$ break;
11 $\quad$ **end**
12 $\quad$ **if** *flag* **then**
13 $\quad\quad$ ***(C procedure:Line 14-24)***
14 $\quad\quad$ **if** *reset* **then**
15 $\quad\quad\quad \mathbf{p}^i = -\nabla f(\mathbf{d}^i);\ reset = 0;$
16 $\quad\quad$ **else**
17 $\quad\quad\quad \mathbf{p}^i = -\nabla f(\mathbf{d}^i) + \|\nabla f(\mathbf{d}^i)\|^2/\|\nabla f(\mathbf{d}^{i-1})\|^2\mathbf{p}^{i-1};$
18 $\quad\quad$ **end**
19 $\quad\quad \alpha_i = \|\nabla f(\mathbf{d}^i)\|^2/((\mathbf{p}^i)^T H\mathbf{p}^i);$
20 $\quad\quad \mathbf{d}^{i+1} = \mathbf{d}^i + \alpha_i\mathbf{p}^i;$
21 $\quad\quad$ **if** $\|\mathbf{d}^{i+1}\| \ge \lambda$ **then**
22 $\quad\quad\quad$ *Find a* $\tau > 0$ *such that* $\|\mathbf{d}^i + \tau\mathbf{p}^i\| = \lambda;$
23 $\quad\quad\quad \mathbf{d}^{i+1} = \mathbf{d}^i + \tau\mathbf{p}^i;\ flag = 0;$
24 $\quad\quad$ **end**
25 $\quad$ **else**
26 $\quad\quad$ ***(G procedure:Line 27-32)***
27 $\quad\quad \beta_i = (\mathbf{q}^i)^T\nabla f(\mathbf{d}^i)/((\mathbf{q}^i)^T H\mathbf{q}^i);$
28 $\quad\quad \mathbf{d}^{i+1} = \mathbf{d}^i - \beta_i\mathbf{q}^i;\ flag = 1; reset = 1;$
29 $\quad\quad$ **if** $\|\mathbf{d}^{i+1}\| \ge \lambda$ **then**
30 $\quad\quad\quad$ *Find a* $\tau > 0$ *such that* $\|\mathbf{d}^i - \tau\mathbf{q}^i\| = \lambda;$
31 $\quad\quad\quad \mathbf{d}^{i+1} = \mathbf{d}^i - \tau\mathbf{q}^i;\ flag = 0;$
32 $\quad\quad$ **end**
33 $\quad$ **end**
34 **end**

Output: $\mathbf{d}^\star,\ iter$

Theoretical Analysis

In this subsection, we present some theoretical results for the MSCG algorithm (Algorithm 1). First of all, inspired by (Kučera 2007), we provide an important lemma which motivates us to solve the sphere constrained quadratic optimization problem in a multi-stage manner.

Lemma 1 *Let* $\mathbf{q}^i = \nabla f(\mathbf{d}^i) + \frac{\|\nabla f(\mathbf{d}^i)\|}{\lambda}\mathbf{d}^i$ *and* $\|\mathbf{d}^i\| = \lambda$. *If* $\mathbf{d}^i$ *is not the optimal solution of Eq. (2), then we have* $(\mathbf{q}^i)^T\mathbf{d}^i > 0$ *and* $(\mathbf{q}^i)^T\nabla f(\mathbf{d}^i) > 0$.

Proof The KKT conditions of Eq. (2) are

$$\nabla f(\mathbf{d}^\star) + \mu^\star\mathbf{d}^\star = \mathbf{0},\ \mu^\star(\|\mathbf{d}^\star\| - \lambda) = 0 \text{ and } \mu^\star \ge 0, \quad (3)$$

where $\mathbf{d}^\star$ and $\mu^\star$ are the optimal primal and dual variables of Eq. (2), respectively. If $\mu^\star > 0$, we have

$$\|\mathbf{d}^\star\| = \lambda \Rightarrow \mu^\star = \frac{\|\nabla f(\mathbf{d}^\star)\|}{\|\mathbf{d}^\star\|} = \frac{\|\nabla f(\mathbf{d}^\star)\|}{\lambda}.$$

Otherwise, $\mu^\star = 0$ and $\nabla f(\mathbf{d}^\star) = \mathbf{0}$ hold. Thus, we can rewrite the above KKT conditions in Eq. (3) into the following compact form:

$$\mathbf{q}^\star = \nabla f(\mathbf{d}^\star) + \frac{\|\nabla f(\mathbf{d}^\star)\|}{\lambda}\mathbf{d}^\star = \mathbf{0}. \quad (4)$$

If $\mathbf{d}^i$ is not the optimal solution, then $\mathbf{q}^i \neq \mathbf{0}$. Thus $\nabla f(\mathbf{d}^i)$ and $\mathbf{d}^i$ are not collinear. Moreover, we have $\|\mathbf{d}^i\| = \lambda$, thus Lemma 1 follows from the facts:

$$(\mathbf{q}^i)^T\mathbf{d}^i = (\mathbf{d}^i)^T\nabla f(\mathbf{d}^i) + \lambda\|\nabla f(\mathbf{d}^i)\|$$
$$> \lambda\|\nabla f(\mathbf{d}^i)\| - \|\mathbf{d}^i\| \cdot \|\nabla f(\mathbf{d}^i)\| = 0,$$
$$(\mathbf{q}^i)^T\nabla f(\mathbf{d}^i) = \|\nabla f(\mathbf{d}^i)\|\left(\|\nabla f(\mathbf{d}^i)\| + \frac{(\mathbf{d}^i)^T\nabla f(\mathbf{d}^i)}{\lambda}\right) > 0.$$

□

Lemma 1 indicates that when the iterative solution $\mathbf{d}^i$ is on the boundary of the sphere, $-\mathbf{q}^i$ is a descent direction and simultaneously points to the inner of the sphere. Thus, conducting the gradient descent along $-\mathbf{q}^i$ can decrease the objective function value and make the next iterative solution $\mathbf{d}^{i+1}$ be an interior point of the sphere. Once $\mathbf{d}^{i+1}$ is in the interior of the sphere constraint, we can perform the conjugate gradient procedure (C procedure).

Next, we formally present a result which indicates the descent property of the MSCG algorithm (Algorithm 1).

Lemma 2 *If* $\mathbf{d}^i$ *is not the optimal solution of Eq. (2), then* $\mathbf{d}^{i+1}$ *generated by Algorithm 1 decreases the objective function value, i.e.,* $f(\mathbf{d}^{i+1}) < f(\mathbf{d}^i)$.

Proof Algorithm 1 includes C procedure and G procedure, we prove Lemma 2 in two cases:

(1) $\mathbf{d}^{i+1}$ is generated by C procedure. If $\mathbf{d}^{i+1} = \mathbf{d}^i + \alpha_i\mathbf{p}^i$ satisfies $\|\mathbf{d}^{i+1}\| < \lambda$, we have

$$\nabla f(\mathbf{d}^{i+1}) = f(\mathbf{d}^i) + \alpha_i H\mathbf{p}^i \text{ and } (\mathbf{p}^i)^T\nabla f(\mathbf{d}^{i+1}) = 0,$$

according to the properties of the conjugate gradient method (Bertsekas 1999). Thus, we obtain:

$$f(\mathbf{d}^{i+1}) - f(\mathbf{d}^i) = \alpha_i(\mathbf{p}^i)^T\nabla f(\mathbf{d}^i) + \frac{1}{2}\alpha_i^2(\mathbf{p}^i)^T H\mathbf{p}^i$$
$$= \alpha_i(\mathbf{p}^i)^T(\nabla f(\mathbf{d}^i) + \alpha_i H\mathbf{p}^i) - \frac{1}{2}\alpha_i(\mathbf{p}^i)^T H\mathbf{p}^i$$
$$= \alpha_i(\mathbf{p}^i)^T\nabla f(\mathbf{d}^{i+1}) - \frac{1}{2}\alpha_i(\mathbf{p}^i)^T H\mathbf{p}^i$$
$$= -\frac{1}{2}\alpha_i(\mathbf{p}^i)^T H\mathbf{p}^i < 0.$$

Otherwise, $\mathbf{d}^{i+1} = \mathbf{d}^i + \tau\mathbf{p}^i (0 < \tau \le \alpha_i)$ and we have

$$f(\mathbf{d}^{i+1}) - f(\mathbf{d}^i) = \tau(\mathbf{p}^i)^T\nabla f(\mathbf{d}^i) + \frac{1}{2}\tau^2(\mathbf{p}^i)^T H\mathbf{p}^i$$
$$= \tau(\mathbf{p}^i)^T(\nabla f(\mathbf{d}^i) + \alpha_i H\mathbf{p}^i) + \left(\frac{1}{2}\tau^2 - \tau\alpha_i\right)(\mathbf{p}^i)^T H\mathbf{p}^i$$
$$= \left(\frac{1}{2}\tau^2 - \tau\alpha_i\right)(\mathbf{p}^i)^T H\mathbf{p}^i < 0.$$

Thus, $f(\mathbf{d}^{i+1}) < f(\mathbf{d}^i)$ holds.

(2) $\mathbf{d}^{i+1}$ is generated by G procedure. Denote

$$\begin{aligned} l(\theta) &= f(\mathbf{d}^{i+1}) - f(\mathbf{d}^i - \theta\mathbf{q}^i) \\ &= \frac{1}{2}\theta^2(\mathbf{q}^i)^T H\mathbf{q}^i - \theta(\mathbf{q}^i)^T\nabla f(\mathbf{d}^i). \end{aligned}$$

$l(\theta)$ is a strictly decreasing function of θ in the interval $[0, \beta_i]$, where $\beta_i = (\mathbf{q}^i)^T\nabla f(\mathbf{d}^i)/((\mathbf{q}^i)^T H\mathbf{q}^i) > 0$. Moreover, we have $l(0)$=0 and hence $l(\tau) < 0$, $l(\beta_i) < 0$ for all $0 < \tau \leq \beta_i$. Therefore, either $\mathbf{d}^{i+1} = \mathbf{d}^i - \beta_i\mathbf{q}^i$ or $\mathbf{d}^{i+1} = \mathbf{d}^i - \tau\mathbf{q}^i (0 < \tau \leq \beta_i)$ is adopted, we both have $f(\mathbf{d}^{i+1}) < f(\mathbf{d}^i)$. □

The descent property in Lemma 2 is a critical result for the convergence of Algorithm 1. Finally, we present the convergence guarantee as follows:

Theorem 1 *The sequence $\{\mathbf{d}^i\}$ generated by Algorithm 1 converges to the unique optimal solution $\mathbf{d}^\star$.*

Proof We first show that the sequence $\{\mathbf{d}^i\}$ generated by Algorithm 1 has limit points. Since $\|\mathbf{d}^i\| \leq \lambda$ (see Algorithm 1) and $f(\mathbf{d}^i) < f(\mathbf{d}^0)$ (see Lemma 2), so the level set $\{\mathbf{d} \mid f(\mathbf{d}) \leq f(\mathbf{d}^0)\}$ is nonempty and bounded. Thus, $\{\mathbf{d}^i\}$ generated by Algorithm 1 has limit points.

Next, we show that every limit point of $\{\mathbf{d}^i\}$ is the unique optimal solution $\mathbf{d}^\star$. $f(\mathbf{d})$ is continuously differentiable and $\{\mathbf{d}^i\}$ is a sequence satisfying $f(\mathbf{d}^{i+1}) < f(\mathbf{d}^i)$ for all i. In addition, $\{\mathbf{d}^i\}$ is generated by a gradient method $\mathbf{d}^{i+1} = \mathbf{d}^i + \theta_i\mathbf{s}^i$, where $\theta_i = \alpha_i$ (or τ), $\mathbf{s}^i = \mathbf{p}^i$ for C procedure and $\theta_i = \beta_i$ (or τ), $\mathbf{s}^i = -\mathbf{q}^i$ for G procedure. According to Proposition 1.2.5 in (Bertsekas 1999), every limit point of $\{\mathbf{d}^i\}$ generated by Algorithm 1 is a stationary point of $f(\mathbf{d})$. Note that we consider the case that H is positive definite and hence $f(\mathbf{d})$ is strictly convex. Thus, $f(\mathbf{d})$ has a unique optimal solution $\mathbf{d}^\star$ and hence every limit point of $\{\mathbf{d}^i\}$ is the optimal solution $\mathbf{d}^\star$. □

Implementation Issues and Discussions

We address some implementation issues and discussions in the following aspects:

(1) In the implementation of Algorithm 1, we calculate the matrix-vector product instead of storing the Hessian matrix H explicitly. This saves memory for storing a large H and makes the MSCG algorithm can tackle large scale trust region step problems.

(2) The most costly computation in each iteration is the matrix-vector product and we implement Algorithm 1 carefully such that only one matrix-vector product (for computing the step size) is necessary in each iteration. Specifically, when computing the step size α_i (or β_i) in the i-th iteration, we record the matrix-vector product $H\mathbf{p}^i$ (or $H\mathbf{q}^i$) as $\mathbf{Hv}$. Then we compute the gradient of the next iteration via $\nabla f(\mathbf{d}^{i+1}) = \nabla f(\mathbf{d}^i) + \theta_i\mathbf{Hv}$, where θ_i equals α_i or τ if C procedure is adopted, $-\beta_i$ or τ otherwise.

(3) For the termination criterion of Algorithm 1, we consider the KKT condition in Eq. (4). Strictly speaking, Algorithm 1 converges if and only if Eq. (4) is satisfied. In practical implementation, we can use any small parameter ϵ to specify the precision of the solution (see line 9 of Algorithm 1). We notice that the convergence result for TRON in (Lin, Weng, and Keerthi 2008) (Theorem 2) is established by requiring the precision of the solution of the sub-problem in Eq. (2) to satisfy some conditions. Our proposed MSCG algorithm guarantees convergence and can find a solution in any prescribed precision controlled by users. Thus, the MSCG algorithm can always meet the precision requirement of Theorem 2 in (Lin, Weng, and Keerthi 2008). Moreover, Steihaug's algorithm can be treated as a special case of the MSCG algorithm with a specific termination criterion. If we set the initial point $\mathbf{d}^0$ as $\mathbf{0}$ and terminate Algorithm 1 when the iterative solution first reaches the boundary of the sphere (see Figure 1), the MSCG algorithm degrades into Steihaug's algorithm (Steihaug 1983). The advantage of the MSCG algorithm over Steihaug's algorithm is its ability of controlling the precision of the solution and it is more flexible in different applications.

(4) The step sizes α_i and β_i are both computed according to the minimization rule (Bertsekas 1999). In the truncated step of line 22-23, τ is the nonnegative root of the following quadratic equation:

$$\|\mathbf{p}^i\|^2\tau^2 + 2(\mathbf{p}^i)^T\mathbf{d}^i\tau + \|\mathbf{d}^i\|^2 - \lambda^2 = 0.$$

But in the truncated step of line 30-31, we have $\|\mathbf{d}^i\|^2 = \lambda^2$. Thus, τ is computed by:

$$\tau = 2(\mathbf{q}^i)^T\mathbf{d}^i/\|\mathbf{q}^i\|^2.$$

4 Experiments

In this section, we present empirical studies on the MSCG algorithm compared with the state-of-the-art algorithms.

Experimental Setup

We consider the trust region step problem in the following logistic regression:

$$\min_{\mathbf{w}}\{l(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \tau\sum_{i=1}^{m}\log\left(1 + \exp(-y_i\mathbf{w}^T\mathbf{x}_i)\right)\},$$

where $\mathbf{w}$ is the weight vector; $\mathbf{x}_i$ and $y_i \in \{+1, -1\}$ are the i-th training data and corresponding class label, respectively; τ is a parameter to balance the Euclidean norm regularization and the logistic loss. To solve the trust region step problem in Eq. (2), we need to compute the gradient and Hessian matrix of $l(\mathbf{w})$, which can be obtained as follows (Lin, Weng, and Keerthi 2008):

$$\mathbf{g} = \mathbf{w} + \tau\sum_{i=1}^{m}\left(\sigma(y_i\mathbf{w}^T\mathbf{x}_i) - 1\right)y_i\mathbf{x}_i, \tag{5}$$

$$H = I + \tau X^T DX, \tag{6}$$

where $\sigma(\mathbf{s})$ is the sigmoid function

$$\sigma(\mathbf{s}) = (1 + \exp(-\mathbf{s}))^{-1} \text{ and } X = \left[\mathbf{x}_1^T; \cdots; \mathbf{x}_m^T\right]$$

is the data matrix with each row as a sample. I is the identity matrix and D is a diagonal matrix with the i-th diagonal entry as

$$D_{ii} = \sigma(y_i\mathbf{w}^T\mathbf{x}_i)\left(1 - \sigma(y_i\mathbf{w}^T\mathbf{x}_i)\right).$$

Table 1: Text data sets statistics: m is the number of samples ($\sharp$ positive and $\sharp$ negative are the number of positive and negative samples, respectively) and n is the dimensionality of the data. $\sharp$ nonzero denotes the number of the nonzero entries of the data.

No.	1	2	3	4	5	6	7	8	9	10	11	12
datasets	classic	hitech	k1b	la12	la1	la2	news20	ng3sim	ohscal	real-sim	reviews	sports
m	7094	2301	2340	2301	3204	3075	19996	2998	11162	72309	4069	8580
n	41681	10080	21839	31472	31472	31472	1355191	15810	11465	20958	18482	14866
$\sharp$ positive	2431	1030	2024	2413	1250	1163	9999	1000	4497	22238	2132	4967
$\sharp$ negative	4663	1271	316	3866	1954	1912	9997	1998	6665	50071	1937	3613
$\sharp$ nonzero	223839	331373	302992	939407	484024	455383	9097916	335879	674365	3709083	758635	1091723

Table 2: The averaged time (seconds) and the averaged number of the matrix-vector product ($\sharp$ average prodnum) over 10 independent runs for $\lambda = 10$ (left) and $\lambda = 1000$ (right) on the text data sets $1 - 4$.

data sets	averaged time (seconds)			$\sharp$ averaged prodnum			averaged time (seconds)			$\sharp$ averaged prodnum		
	MSCG	APG	PG	MSCG	APG	PG	MSCG	APG	PG	MSCG	APG	PG
classic	**0.0391**	0.0942	0.0671	**8.0**	29.0	20.0	**0.3636**	18.3538	24.9720	**76.0**	5516.0	6896.0
hitech	**0.0523**	0.0865	0.0624	**20.0**	39.0	28.0	**2.0385**	17.6176	23.4262	**742.0**	7737.0	9983.0
k1b	**0.0245**	0.0938	0.0683	**7.0**	33.0	24.0	**2.6893**	18.7076	58.3748	**705.0**	6459.0	18994.0
la12	**0.1134**	0.3464	0.1977	**14.0**	47.0	27.0	**7.4677**	57.5597	144.6517	**862.0**	7669.0	18693.0

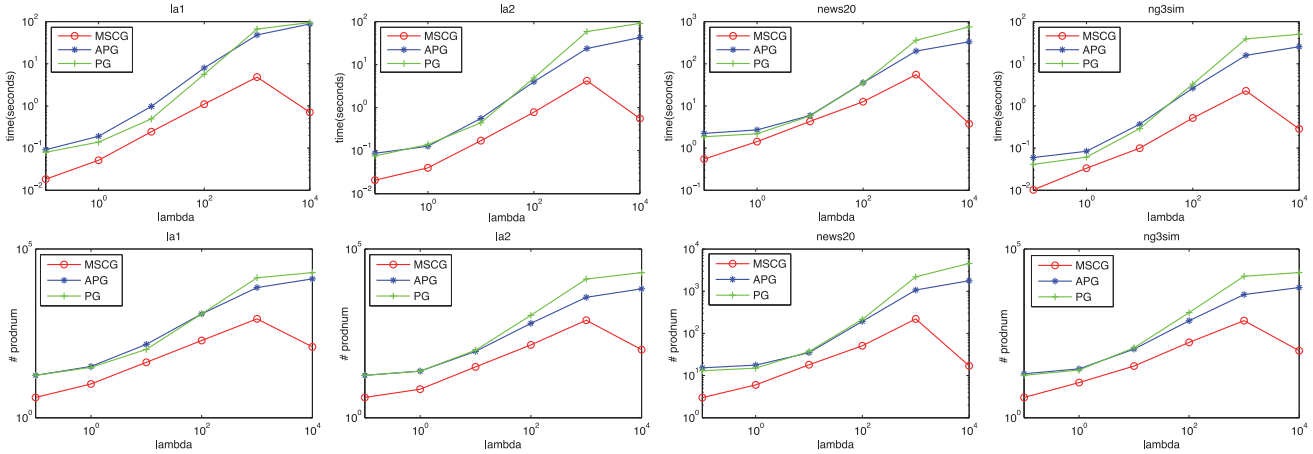

Figure 2: Averaged time (top) and averaged $\sharp$ prodnum (bottom) vs. the trust region radius (λ) plots on the text data sets $5 - 8$.

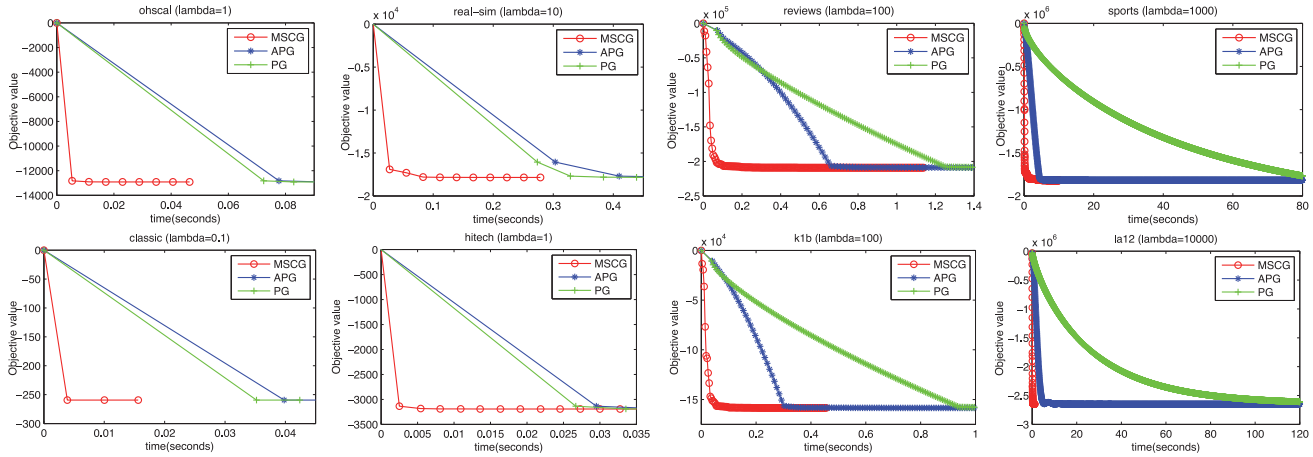

Figure 3: Objective (function) value vs. time plots with different trust region radiuses on the text data sets $9 - 12$ and $1 - 4$.

We should note that the positive definite Hessian matrix H is very large when the data is in high dimension. Moreover, H is dense even though X is sparse. Thus it's impossible to store the Hessian matrix explicitly. To address this issue, we calculate the following matrix-vector product

$$H\mathbf{p} = \mathbf{p} + \tau X^T(D(X\mathbf{p})) \tag{7}$$

instead, since the MSCG algorithm haven't to store H explicitly as long as the matrix-vector product is available. If the matrix X is sparse, we can efficiently calculate the above matrix-vector product without storing H.

Competing Algorithms and Data Sets

We compare the proposed MSCG algorithm with two competing algorithms: Projected Gradient (PG) and Accelerated Projected Gradient (APG). The two algorithms are very popular in the communities of artificial intelligence and machine learning and they have been successfully applied to efficiently solve many optimization problems with simple constraints (e.g., sphere constraint). Especially in large-scale scenarios, the PG and APG algorithms are very efficient. We should mention that, we don't include Steihaug's algorithm (Steihaug 1983) for comparison, since it is a special case of the MSCG algorithm with a specific termination criterion.

We consider twelve text data sets from different sources. These data sets are high dimensional and sparse. They are summarized in Table 1. Two of them (news20, real-sim) are downloaded from http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/, and they have been preprocessed as two-class data sets (Lin, Weng, and Keerthi 2008). The other ten are available at http://www.shi-zhong.com/software/docdata.zip and they are multi-class data sets. We transform the multi-class data sets into two-class (Lin, Weng, and Keerthi 2008) by labeling the first half of all classes as positive class, and the remaining classes as the negative class.

Experimental Evaluation and Analysis

The three algorithms are implemented in Matlab and we execute the experiments on an Intel(R) Core(TM)2 Quad CPU (Q6600 @2.4GHz) with 8GB memory. We set $\tau = 1$ and generate each entry of $\mathbf{w}$ from the normal distribution with mean 0 and standard deviation 1. Thus we can obtain the gradient and the matrix-vector product via Eq. (5) and Eq. (7). To fully evaluate the computational efficiency of the MSCG algorithm, we set the trust region radius λ in Eq. (2) as 10^i $(i = -1, 0, 1, 2, 3, 4)$. We terminate the three algorithms (MSCG, APG and PG) when the precision controlling tolerance ϵ is less than 10^{-4} and record the CPU time and the number of the matrix-vector product ($\sharp$ prodnum).

We report the CPU time and the number of the matrix-vector product which are averaged over 10 independent runs as in Table 2. From these results, we have the following observations: (a) The MSCG algorithm is the most efficient among the three algorithms, especially when the trust region radius is large ($\lambda = 1000$). (b) The CPU time is proportional to the number of the matrix-vector product ($\sharp$ prodnum) for all the three algorithms, since the matrix-vector product is the most costly operation in each iteration. This is consistent with the analysis in Section 3. (c) PG is more efficient than APG when the trust region radius is small, but APG is faster than PG when the trust region radius is large.

We further study the performance of the computational efficiency on the MSCG algorithm when the trust region radius varies. Figure 3 shows the averaged CPU time and the averaged number of the matrix-vector product ($\sharp$ prodnum) vs. the trust region radius (λ) plots over 10 independent runs. From these curves, we observe: (i) The MSCG algorithm is always the most efficient among the three algorithms. (ii) For the PG and APG algorithms, the CPU time and the number of matrix-vector product increase when the trust region radius increases. But for the MSCG algorithm, when the trust region radius is large enough ($\lambda = 10000$), both the CPU time and the number of the matrix-vector product decrease. When the trust region radius is large enough ($\lambda = 10000$), the optimal solution of the trust region step problem in Eq. (2) is in the interior of the sphere constraint. Thus, Eq. (2) is equivalent to an unconstrained quadratic optimization problem. In this situation, the PG and APG algorithms are computationally more expensive, while the MSCG algorithm only involves C procedure and it is computationally inexpensive. (iii) We observe the same phenomenons as in the above observations (b) and (c).

To check the convergence detail of the MSCG algorithm, we show the objective (function) value vs. time plots as in Figure 3. From these curves, we have the following observations: (1) The MSCG algorithm always converges the fastest among the three algorithms. (2) The MSCG algorithm rapidly decreases the objective value and make it close to the minimum objective value in very short time. This is very useful when we don't require high-precision solutions.

5 Conclusions

In this paper, we propose an efficient Multi-Stage Conjugate Gradient (MSCG) algorithm to solve the trust region step problem, which is a key ingredient in the trust region Newton method. The proposed MSCG algorithm extends the conjugate gradient method to solve the *sphere constrained* quadratic optimization problem, which involves C procedure and G procedure. When the iterative solution is in the interior of the sphere, we perform C procedure. Otherwise, we perform G procedure where the descent direction points to the inner of the sphere. Thus, G procedure may make the next iterative solution be an interior point and hence we can proceed with the conjugate gradient again. Through repeating the above procedures until convergence, we utilize the conjugate gradient in a multi-stage stage manner.

The MSCG algorithm can generate a solution in any prescribed precision controlled by a tolerance parameter which is the only parameter we need. Moreover, we provide a detailed theoretical analysis which shows that the MSCG algorithm can decrease the objective function value in each iteration and further guarantees the convergence of the algorithm. Empirical studies on large-scale text data sets demonstrate our proposed MSCG algorithm has a faster convergence speed compared with the state-of-the-art algorithms. More interestingly, our MSCG algorithm can rapidly decrease the objective value and make it close to the minimum objective

value in very short time. Thus, we can obtain a good approximated solution quickly. This is very useful when we don't require high-precision solutions.

In our future work, we hope to extend the MSCG algorithm to efficiently solve more problems such as ℓ_1-norm constrained least squares and other structured sparsity optimization problems. There are three key aspects: (1) In the C procedure, if some feasible solution exceeds the constraint, how can we efficiently truncate it such that it lies on the boundary of the constraint? (2) When some feasible solution lies on the boundary of the constraint, how can we find a descent direction which points to the inner of the constraint? (3) Can we employ the preconditioned conjugate gradient instead of the conjugate gradient method to further accelerate the convergence speed?

Acknowledgments

This work is supported by NSFC (Grant No. 91120301, 61075004 and 60835002).

References

Bach, F.; Jenatton, R.; Mairal, J.; and Obozinski, G. 2011. Optimization with sparsity-inducing penalties. *Arxiv preprint arXiv:1108.0775*.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

Bertsekas, D. P. 1999. *Nonlinear Programming*. Athena Scientific.

Daubechies, I.; Fornasier, M.; and Loris, I. 2008. Accelerated projected gradient method for linear inverse problems with sparsity constraints. *Journal of Fourier Analysis and Applications* 14(5):764–792.

Duchi, J.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *ICML*, 272–279.

Erway, J., and Gill, P. 2009. A subspace minimization method for the trust-region step. *SIAM Journal on Optimization* 20:1439–1461.

Erway, J.; Gill, P.; and Griffin, J. 2009. Iterative methods for finding a trust-region step. *SIAM Journal on Optimization* 20(2):1110–1131.

Figueiredo, M.; Nowak, R.; and Wright, S. 2007. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing* 1(4):586–597.

Fortin, C., and Wolkowicz, H. 2004. The trust region subproblem and semidefinite programming. *Optimization Methods and Software* 19(1):41–67.

Golub, G., and Von Matt, U. 1991. Quadratically constrained least squares and quadratic problems. *Numerische Mathematik* 59(1):561–580.

Gong, P.; Gai, K.; and Zhang, C. 2011. Efficient euclidean projections via piecewise root finding and its application in gradient projection. *Neurocomputing* 2754–2766.

Gould, N.; Lucidi, S.; Roma, M.; and Toint, P. 1999. Solving the trust-region subproblem using the lanczos method. *SIAM Journal on Optimization* 9(2):504–525.

Hager, W. 2001. Minimizing a quadratic over a sphere. *SIAM Journal of Optimization* 12:188–208.

Kim, D.; Sra, S.; and Dhillon, I. 2010. A scalable trust-region algorithm with application to mixed-norm regression. In *ICML*.

Kučera, R. 2007. Minimizing quadratic functions with separable quadratic constraints. *Optimization Methods and Software* 22(3):453–467.

Lin, C., and Moré, J. 1999. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization* 9:1100–1127.

Lin, C.; Weng, R.; and Keerthi, S. 2008. Trust region newton method for logistic regression. *Journal of Machine Learning Research* 9:627–650.

Lin, C. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Computation* 19(10):2756–2779.

Liu, J.; Ji, S.; and Ye, J. 2009a. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *UAI*, 339–348.

Liu, J.; Ji, S.; and Ye, J. 2009b. Slep: Sparse learning with efficient projections. *Arizona State University*.

Liu, J.; Sun, L.; and Ye, J. 2011. Projection onto a nonnegative max-heap. *NIPS*.

Mor, J., and Sorensen, D. 1983. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computations* 4:553–572.

Nesterov, Y. 2004. *Introductory lectures on convex optimization: A basic course*. Springer Netherlands.

Rendl, F., and Wolkowicz, H. 1997. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming* 77(1):273–299.

Rojas, M.; Santos, S.; and Sorensen, D. 2000. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM Journal on Optimization* 11(3):611–646.

Shalev-Shwartz, S.; Singer, Y.; and Srebro, N. 2007. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML*, 814–821.

Steihaug, T. 1983. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis* 626–637.

Wright, S.; Nowak, R.; and Figueiredo, M. 2009. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing* 57(7):2479–2493.

Yuan, G.; Chang, K.; Hsieh, C.; and Lin, C. 2010. A comparison of optimization methods and software for large-scale ℓ_1-regularized linear classification. *Journal of Machine Learning Research* 11:3183–3234.

Yuan, L.; Liu, J.; and Ye, J. 2011. Efficient methods for overlapping group lasso. *NIPS*.