A Newton-PCG Like Algorithm Based on the Progress Behavior of Newton's Method

Ping Zhong

China Agricultural University, Beijing

Abstract

It is known that the Newton-PCG like algorithms are generally very successful, which is shown by a large amount of numerical experiments, but unfortunately short of theoretical justification on its efficiency. Recently, a theoretical analysis of the efficiency is developed in [5], which shows that the efficiency of a Newton-PCG like algorithm is theoretical superior to that of Newton's method under the assumption: Newton's method is precisely quadratically convergent. In this paper, the assumption in [5] is weakened to a more general case, and an encouraging theoretical analysis on the efficiency of a modified Newton-PCG like algorithm is obtained. It further shows the superiority of the Newton-PCG like algorithm from the theoretical point of view.

Key words: Newton's method, Preconditioned conjugate gradient method

1 Introduction

We are interested in the local algorithm for the unconstrained optimization problem

$$\min f(x),\ x \in R^n, \tag{1.1}$$

where $f(x)$ satisfies

Assumption 1. The problem (1.1) has a solution x^* with a symmetric positive definite Hessian $\nabla^2 f(x^*)$.

Assumption 2. In a neighborhood of x^*, the Hessian $\nabla^2 f$ is Lipschitz continuous.

Newton like methods are usually considered as the best methods provided that the Hessian matrix $\nabla^2 f$ is available. Based on the analysis on the inexact Newton method in [3] and the properties of conjugate gradient method (CG method) (see e.g. [8] and

[7]), Steihaug and Toint proposed Newton-CG like algorithms (see [9] and [10]), where the Newton equation was solved by CG method approximately. Later, these algorithms were well developed to a kind of Newton-PCG like algorithms, where the CG method was replaced by PCG method (preconditioned CG method). It has been shown by a large amount of experiments that, generally speaking, Newton-PCG like algorithms are very successful (see e.g. [1], [6] and [4]), but unfortunately short of theoretical justification on its efficiency.

Recently, a Newton-PCG like algorithm is proposed in [5]. It has been shown that the algorithm in [5] has the same convergence order as Newton's method's, but *theoretically*, its average number of arithmetic operations per step is much less than the corresponding number of Newton's method for middle and large scale problems. However, all of the above conclusions are obtained under the assumption that Newton's method is precisely quadratically convergent, i.e. there exist positive scalars δ_1, $M_2 \geq M_1 > 0$, such that when $||x - x^*|| < \delta_1$, the Newton iterate $x_+ = x - \nabla^2 f(x)^{-1}\nabla f(x)$ satisfies

$$M_1||x - x^*||^2 \leq ||x_+ - x^*|| \leq M_2||x - x^*||^2 \tag{1.2}$$

Obviously, the assumption (1.2) is very strong. Moreover, its validity is difficult to be verified even it actually holds.

In fact, for the general case (under Assumptions 1 and 2), Newton's method has at least Q-quadratic order of convergence, i.e., there exist positive scalars δ_2, $M_3 > 0$, such that when $||x - x^*|| < \delta_2$, the Newton iterate $x_+ = x - \nabla^2 f(x)^{-1}\nabla f(x)$ satisfies

$$||x_+ - x^*|| \leq M_3||x - x^*||^2. \tag{1.3}$$

This paper deals with the above general case. Our algorithm consists of circles. The basic structure of a circle is that every exact Newton step is followed by several PCG steps. The number of the PCG steps depends on the progress of the previous exact Newton step. Therefore, the algorithm depends on the behavior of Newton's method.

Our algorithm has the following properties: for a fixed dimension n, there exists a critical value such that when the progress speed of Newton's method is smaller than it, our algorithm is superior to Newton's method. In addition, when (1.2) is valid, our algorithm and the one in [5] have the same efficiency estimates from the theoretical point of view.

2 A one-dimensional optimization problem

Our algorithm is concerned with the following one-dimensional optimization problem with respect to K

$$P(n,\alpha): \qquad \min_K \ u(K;n,\alpha) = \frac{1+\sum_{m=1}^{K} \varphi(\alpha,m)Q(n)}{1+K}, \tag{2.1}$$

$$\text{s.t. } K \text{ is a nonnegative integer,} \tag{2.2}$$

where n is the dimension of (1.1), α is a parameter, $\varphi(\tau,q)$ is an integer function defined by

$$(\tau)^q(\tau-1) < \varphi(\tau,q) \le (\tau)^q(\tau-1)+1 \tag{2.3}$$

and $Q(n)$ is defined by

$$Q(n) = (2n^2+6n+2)/(n^3/6+3n^2/2-2n/3). \tag{2.4}$$

(Note that when $K=0$, we define the 'sum' $\sum_{m=1}^{K}\cdots = 0$). In fact, in the i-th circle, the number p_i of the PCG steps should be selected to be the solution to (2.1)-(2.2) with $\alpha = \alpha_i$. However, if the problem (2.1)-(2.2) with different α_i is solved respectively, the corresponding computation cost will break down the theoretical efficiency of the algorithm. This is the main difficult when the algorithm in [5] is extended. Fortunately, the solution has an analytic expression as shown in Theorem 2.1 below. By this expression, the solutions to various values of α_i can be obtained easily without solving the problem numerically.

Theorem 2.1 *Consider the problem $P(n,\alpha)$ with a fixed positive integer n. If $\alpha \ge 2$, then there exists the smallest global solution $K^*(n,\alpha)$. Furthermore, setting $q = K^*(n,2)$, there are $q+1$ scalars*

$$b_0 = b_0(n) \ge b_1 = b_1(n) \ge \cdots \ge b_q = b_q(n) = 2, \tag{2.5}$$

such that $K^*(n,\alpha)$, as a function of α, can be expressed as

$$K^*(n,\alpha) = \begin{cases} 0, & \text{when } \alpha \in [b_0(n), \infty); \\ 1, & \text{when } \alpha \in [b_1(n), b_0(n)); \\ \cdots & \\ q, & \text{when } \alpha \in [b_q(n), b_{q-1}(n)), \end{cases} \tag{2.6}$$

where the interval $[b_j, b_{j-1})$ is empty if $b_j = b_{j-1}, j = 1, \cdots, q$.

Theorem 2.2 *The optimal value $u^*(n,\alpha)$ to the problem $P(n,\alpha)$ is nondecreasing with respect to α.*

3 Algorithm

Algorithm

Step 0. Initial data: choose the initial point $x^0 \in R^n$. Set $k = 0$. Go to Step 3.

Step 1. Termination test: if $|| \nabla f(x^k)|| = 0$, stop.

Step 2. Switch test: if $\bar{\alpha}_k < b_0$ defined in Theorem 2.1 and $m \leq p_k$, go to Step 4; otherwise go to Step 3.

Step 3. Exact Newton step: find the exact solution s^k to the Newton equation and estimate the progress of Newton's method as follows:

3.0 Find the solution s^k to the Newton equation

$$\nabla^2 f(x^k)s = - \nabla f(x^k) \tag{3.1}$$

by Cholesky factorization $\nabla^2 f(x^k) = L_k D_k L_k^T$.

3.1 Set $v = x^k$ and $B = \nabla^2 f(x^k)$. Set $x^{k+1} = x^k + s^k$ and $m = 1$.

3.2 Compute

$$\alpha_{k+1} = \ln || \nabla f(x^{k+1})||/ \ln ||v - x^{k+1}||, \tag{3.2}$$

and set

$$\bar{\alpha}_{k+1} = \max\{\alpha_{k+1}, 2\}. \tag{3.3}$$

3.3 Set

$$p_{k+1} = K^*(n, \bar{\alpha}_{k+1}), \tag{3.4}$$

where $K^*(n, \bar{\alpha}_{k+1})$ is defined by (2.6) in Theorem 2.1 with α there being replaced by $\bar{\alpha}_{k+1}$. Go to Step 5.

Step 4. PCG step: find s^k by Algorithm PCG(B^{-1}, $\nabla^2 f(x^k)$, $-\nabla f(x^k)$, $\bar{l}_m$, $\bar{l}_m/(\bar{\alpha}_k)^m$), where $\bar{l}_m = \varphi(\bar{\alpha}_k, m)$ defined by (2.3) with α there being replaced by $\bar{\alpha}_k$. Set $\bar{\alpha}_{k+1} = \bar{\alpha}_k$, $p_{k+1} = p_k$. Set $x^{k+1} = x^k + s^k$ and $m = m + 1$.

Step 5. Set $k = k + 1$, and go to Step 1.

Note: Algorithm PCG(C, A, b, l, e) is the standard preconditioned conjugate gradient method (see e.g. [7]), which is used to solve the linear system $As = b$, where C is the preconditioner, l is the maximum number of subiterations, and e is a scalar used in the termination criterion.

For convenience, the segment of $\{x^k\}$ generated by i-th circle of the algorithm is expressed as follows:

$$\begin{aligned} \{x^k\}_{k=i(p_i+1)}^{(i+1)(p_i+1)} &= \{x^{i(p_i+1)},\ x^{i(p_i+1)+1},\ \cdots,\ x^{i(p_i+1)+p_i},\ x^{(i+1)(p_i+1)}\} \\ &= \{x^{j_i},\ x^{j_i+1},\ \cdots,\ x^{j_{i+1}-1},\ x^{j_{i+1}}\}, \end{aligned} \tag{3.5}$$

where $j_{i+1} = j_i + p_i + 1$.

4 The convergence speed of our algorithm

We assume therefore that the following assumption holds, where the scalars α_l and α_h in the following Assumption 3 characterize the convergence speed of Newton's method. It reduces to (1.2) when $\alpha_l = \alpha_h = 2$.

Assumption 3. There exist $\delta > 0$, $M_l \geq M_h > 0$, α_l and α_h such that when $\|x_c - x^*\| < \delta$,

$$M_h \|x_c - x^*\|^{\alpha_h} \leq \|x_+ - x^*\| \leq M_l \|x_c - x^*\|^{\alpha_l},$$

where x_+ is the Newton mapping: $x_+ = x_c - \nabla^2 f(x_c)^{-1} \nabla f(x_c)$.

The next theorem shows the convergence speed of our new algorithm.

Theorem 4.1 *Under Assumptions 1-3, there exists $\delta > 0$ such that when $\|x_N^{j_i} - x^*\| \le \delta$, the segment (3.5) satisfies*

$$\|x^{j_{i+1}} - x^*\| \le M\|x^{j_i} - x^*\|^{\alpha_l^{(1+p_i)}}, \tag{4.1}$$

where M is a constant which depends only on $\nabla^2 f(x^*)$ and Lipschitz constant L.

5 Efficiency Comparison

Now we shall reply the question that compared with Newton's method, whether and how much saving our algorithm can be expected in theory. According to Theorem 4.1, it is reasonable to consider that their progress speed are almost the same. So we only need to investigate the computation cost of both our algorithm and Newton's method. To obtain a new iterate, the computation cost consists of the following two parts:

(1) evaluating one Hessian $\nabla^2 f$ and one gradient ∇f which yield a Newton equation.

(2) solving the Newton equation(using different ways).

However, for a significant number of widely differing application problems, the linear-algebra cost of the part (2) tends to dominate (see e.g.[2]). So in order to compare their efficiency, we are mainly interested in comparing their arithmetic operations in solving Newton equations, or for simplicity, comparing only their numbers of the multiplicative operations involved.

For Newton's method with Cholesky factorization, the number of the multiplication operations to solve a Newton equation is

$$Q_N = \frac{1}{6}n^3 + \frac{3}{2}n^2 - \frac{2}{3}n.$$

However, the corresponding number of our algorithm is different in different iterations. So, we consider its average value in $j_{i+1} - j_i$ iterations in the segment (3.5):

$$w_{j_i} = W_{j_i}/(j_{i+1} - j_i),$$

where W_{j_i} is the total number of the multiplicative operations involved in solving the Newton equations to obtain $x^{j_{i+1}}$ from x^{j_i} in our algorithm.

Definition 5.1 *Cost Ratio*[5] *: suppose that the sequence $\{x^k\}$ is generated by our algorithm, and $\{x^{j_i}, i = 0, 1, \cdots\}$ is its subsequence. The ratio of computation cost in solving Newton equations by our algorithm against the Newton's method is defined by*

$$\eta = \sup_{\{x^k\}}\{\limsup_{k\to\infty} \frac{w_{j_i}}{Q_N} \;\Big|\; \text{for any } \{x^k\} \text{ which is generated by our algorithm and converges to } x^*\}. \tag{5.1}$$

Theorem 5.1 *If $\alpha_h < b_0$, the ratio η defined by* (5.1) *satisfies*

$$\eta \le u^*(n, \alpha_h) < 1, \tag{5.2}$$

where α_h and b_0 are given in Assumption 3 and Theorem 2.1 respectively, $u^*(n, \alpha_h)$ is the optimal value to the one-dimensional optimization problem $P(n, \alpha_h)$ defined by (2.1) - (2.4) with α there being replaced by α_h.

It is shown by Theorem 5.1 that the upper bound $u^*(n, \alpha_h)$ of cost ratio η depends on the value of α_h, which, to some extent, reflects the progress speed of Newton's method. In fact, there exists a critical value b_0 such that when the progress speed of Newton's method is smaller than it, our algorithm is superior to Newton's method. According to Theorem 2.2, the smaller the value of α_h, the smaller the value of $u^*(n, \alpha_h)$. The following table lists some typical value of $u^*(n, \alpha_h)$ and, therefore generally speaking, shows our algorithm is much more efficient than Newton's method.

	n					
α_h	100	200	300	400	500	1000
2	.67	.53	.46	.43	.41	.33
2.5	.73	.61	.52	.47	.44	.36
3	.84	.70	.64	.59	.54	.44

Table 5.1: The values of $u^*(n, \alpha_h)$

References

[1] A.R.Conn, N.I.M.Gould, and P.L. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer Ser. Comput. Math. 17, Springer-Verlag, Heidelberg, Berlin, New York, 1992.

[2] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization*, Technical Report, 92–075, Rutherford Appleton Laboratory, Chilton, England, 1992.

[3] R. Dembo, S. Eisenstat, and T. Steihaug, *Inexact Newton method*, SIAM Journal on Numerical Analysis, 19 (1982), 400–408.

[4] L. C. W. Dixon and R. C. Price, *Numerical experience with the truncated Newton method for unconstrained optimization*, Journal of Optimization Theory and Applications, 56(1988), 245–255.

[5] N. Y. Deng and Z. Z. Wang, *Theoretical efficiency of an inexact Newton method*, in Journal of Optimization Theory and Applications, 105(2000), 97-112.

[6] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17(1996), 33–46.

[7] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[8] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer, 1999.

[9] T. Steihaug, *The conjugate gradient method and trust region in large scale optimization*, SIAM Journal on Numerical Analysis, 20(1983), 626–637.

[10] P.L.Toint, *Towards an Efficient Sparsity Exploiting Newton Method for Minimization*, Sparse Matrices and Their Uses, Edited by I.S. Duff, Academic Press, London, England, 1981, 57–88.