Large-Scale Unconstrained Optimization

Saurav Samantaray

Department of Mathematics

Indian Institute of Technology Madras

March 11, 2024

Introduction

- Many applications give rise to unconstrained optimization problems with thousands or millions of variables.
- Problems of this size can be solved efficiently only if the storage and computational costs of the optimization algorithm can be kept at a tolerable level.
- A diverse collection of large-scale optimization methods has been developed to achieve this goal, each being particularly effective for certain problem types.
- Some of these methods are straightforward adaptations of the methods described until now.
- Other approaches are modifications of these basic methods that allow approximate steps to be calculated at lower cost in computation and storage.

Introduction

- The non-linear conjugate gradient methods of can be applied to large problems without modification, owing to its minimal storage demands and its reliance on only first-order derivative information.
- The Newton method in both line search and trust-region algorithms require matrix factorizations of the Hessian matrices.
- High quality software implementations are available, which are based on factorizations that can be carried out using elimination techniques.
- Newton methods are plagued with issues related to computational cost and memory requirements of these factorization methods.
- If the Hessian matrix can be formed explicitly, with the above problems sorted they constitute an effective approach for solving such problems.

Introduction

- Often, however, the cost of factoring the Hessian is prohibitive, and it is preferable to compute approximations to the Newton step using iterative linear algebra techniques.
- Inexact Newton methods that use these techniques, in both line search and trust-region frameworks have attractive global convergence properties and may be super-linearly convergent for suitable choices of parameters.
- There are variants of the quasi-Newton approach, which use Hessian approximations that can be stored compactly by using just a few vectors of length n.
- These methods are fairly robust, inexpensive, and easy to implement, but they do not converge rapidly.

Inexact Newton Methods

- The basic Newton step p_k^N is obtained by solving the symmetric $n \times n$ linear system

$$\nabla^2 f_k p_k^N = -\nabla f_k. \tag{1}$$

- p_k^N can be obtained by solving the above equation (1) approximately, via inexpensive iterative solvers.
- For example the conjugate gradient (CG) method can be employed to get p_k^N.
- Both line search and trust region approaches can be derived based on this approximation, which falls in the general family of inexact Newton methods.
- In addition, we can implement these methods in a Hessian-free manner, so that the Hessian $\nabla^2 f_k$ need not be calculated or stored explicitly at all.

LOCAL CONVERGENCE OF INEXACT NEWTON METHODS

- Consider the residual for the equation (1) as:

$$r_k = \nabla^2 f_k p_k + \nabla f_k \tag{2}$$

  where p_k is the inexact Newton step.

- The CG iterations are terminated when

$$||r_k|| \leq \eta_k ||\nabla f_k||, \tag{3}$$

  where the sequence $\{\eta_k\}$ (with $0 < \eta_k < 1$ for all k) is called the forcing sequence.

LOCAL CONVERGENCE OF INEXACT NEWTON METHODS

Theorem

Suppose that $\nabla^2 f(x)$ exists and is continuous in a neighbourhood of a minimizer x^*, with $\nabla^2 f(x^*)$ is positive definite. Consider the iteration $x_{k+1} = x_k + p_k$ where p_k satisfies (3), and assume that $\eta_k \leq \eta$ for some constant $\eta \in [0, 1)$. Then, if the starting point x_0 is sufficiently near x^*, the sequence $\{x_k\}$ converges to x^* and satisfies

$$||\nabla^2 f(x^*)(x_{k+1} - x^*)|| \leq \hat{\eta}||\nabla^2 f(x*)(x_k - x^*)||, \tag{4}$$

for some constant $\hat{\eta}$ with $\eta < \hat{\eta} < 1$.

LINE SEARCH NEWTON–CG METHOD

- In the line search Newton–CG method, also known as the truncated Newton method, the search direction is computed by applying the CG method to the Newton equations:

$$\nabla^2 f_k p_k^N = -\nabla f_k;$$

- and attempt to satisfy a termination test of the form

$$||r_k|| \leq \eta_k ||\nabla f_k||,$$

- The CG method is designed to solve positive definite systems.
- However, the Hessian $\nabla^2 f_k$ may have negative eigenvalues when x_k is not close to a solution.
- The CG iteration is terminated as soon as a direction of negative curvature is generated.

LINE SEARCH NEWTON–CG METHOD

- This adaptation of the CG method produces a search direction p_k that is a descent direction.
- Moreover, the adaptation guarantees that the fast convergence rate of the pure Newton method is preserved, provided that the step length $\alpha_k = 1$ is used whenever it satisfies the acceptance criteria.
- For purposes of this algorithm rewrite the linear system (1) in the form
$$B_k p = -\nabla f_k \tag{5}$$
where B_k represents $\nabla^2 f_k$.
- For the inner CG iteration, denote the search direction by d_j and the sequence of iterates that it generates by z_j.

LINE SEARCH NEWTON–CG METHOD

- When B_k is positive definite, the inner iteration sequence $\{z_j\}$ will converge to the Newton step p_k^N that solves (5).
- At each major iteration, a tolerance ε_k that specifies the required accuracy of the computed solution, is prescribed
- For concreteness the forcing sequence is chosen to be $\eta_k = \min(0.5, \sqrt{||\nabla f_k||})$ to obtain a super-linear convergence rate (one may choose differently as well).

Line Search Newton–CG

Algorithm

```
Given initial point $x_0$;
for $k = 0, 1, 2, \ldots$
    Define tolerance $\varepsilon = \min(0.5, \sqrt{||\nabla f_k||})||\nabla f_k||$;
    Set $z_0 = 0$, $r_0 = \nabla f_k$, $d_0 = -r_0 = -\nabla f_k$;
    for $j = 0, 1, 2, \ldots$
        if $d_j^T B_k d_j \leq 0$
            if $j = 0$
                return $p_k = -\nabla f_k$;
            else
                return $p_k = z_j$;
        set $\alpha_j = r_j^T r_j / d_j^T B_k d_j$;
        Set $z_{j+1} = z_j + \alpha_j d_j$;
        Set $r_{j+1} = r_j + \alpha_j B_k d_j$;
        if $||r_{j+1}|| < \varepsilon_k$
            return $p_k = z_{j+1}$;
        Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$;
        Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$;
    end (for)
    Set $x_{k+1} = x_k + \alpha_k p_k$, where $\alpha_k$ satisfies the Wolfe, Goldstein, or
        Armijo backtracking conditions (using $\alpha_k = 1$ if possible);
end
```

LINE SEARCH NEWTON–CG METHOD

- The main differences between the inner loop of the above algorithm and the original CG are that the specific starting point $z_0 = 0$ is used;
- and the use of a positive tolerance ε_k allows the CG iterations to terminate at an inexact solution;
- and the negative curvature test $d_j^T B_k d_j \leq 0$ ensures that p_k is a descent direction for f at x_k.
- If negative curvature is detected on the first inner iteration $j = 0$, the returned direction $p_k = -\nabla f_k$ is both a descent direction and a direction of non-positive curvature for f at x_k.

LINE SEARCH NEWTON–CG METHOD

- When the user cannot easily supply code to calculate second derivatives, or where the Hessian requires too much storage automatic differentiation and finite differencing techniques can be used to calculate these Hessian–vector products.
- Methods of this type are known as Hessian-free Newton methods.
- In the finite-differencing technique, we use the approximation

$$\nabla^2 f_k d \approx \frac{\nabla f(x_k + hd) - \nabla f(x_k)}{h}, \tag{6}$$

  for some small differencing interval h.
- It is easy to prove that the accuracy of this approximation is $\mathscr{O}(h)$;
- The price we pay for bypassing the computation of the Hessian is one new gradient evaluation per CG iteration.