Automatic Preconditioning by Limited Memory Quasi-Newton Updating

Jose Manuel de Frutos

13th November 2022

Index

Index

Conjugate Gradient Method I

- Suppose we have some quadratic function:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c$$

  for $x \in \mathbb{R}^n$ with $A \in \mathbb{R}^{n \times n}$ and $b, c \in \mathbb{R}^n$.
- The gradient of f is: $\nabla f(x) = Ax - b$. Minimizing the previous convex problem is equivalent to solve $Ax = b$.
- $-\nabla f$ vector pointing the direction of steepest descent. Initial guess x_0, compute $-\nabla f(x_0)$, and move in that direction by a step size α.
- Find the best α, this α brings us to the minimum of f constrainted to move in the direction $d_0 = -\nabla f(x_0)$.
- Computing α is equivalent to minimizing the function $g(\alpha) = f(x_0 + \alpha d_0)$. Minimum of this function occurs when $g'(\alpha) = 0$. So we get that α is:

$$\alpha = -\frac{d_i^T (Ax_{x_i} + b)}{d_i^T A d_i}$$

Conjugate Gradient Method II

- Second point in our iterative algorithm: $x_1 = x_0 - \alpha \nabla f(x_0)$.
- Find a new direction d_1 to move in that is conjugate (w.r. A) to d_0. So $d_1 = -\nabla f(x_1) + \beta_0 d_0$. We can derive β_0 from conjugacy, $d_1^T A d_0 = 0$.

$$\beta_0 = \frac{\nabla f(x_1)^T A d_0}{d_0^T A d_0}$$

Iterating this will keep giving us conjugate directions.

Conjugate Gradient Method

Algorithm 1 Conjugate Gradient Method

Require: Let $i = 0$ and $x_i = x_0$ be our initial guess;

1: Compute $d_i = d_0 = \nabla f(x_0)$;

2: **while** Stoping test not satisfied **do**

3: Compute:

$$\alpha_i = -\frac{d_i^T(Ax_{x_i} + b)}{d_i^T A d_i};$$

4: Update $x_{i+1} = x_i + \alpha_i d_i$;

5: Update direction $d_{i+1} = -\nabla f(x_{i+1}) + \beta_i d_i$ where:

$$b_i = \frac{\nabla f(x_{i+1})^T A d_i}{d_i^T A d_i};$$

6: **end while**

Preconditioned conjugate gradient method

- Convergence analysis shows that convergence speed is fast when the condition number of A is close to 1.
- Acceleration of convergence rate by replacing the system $Ax = b$ by the preconditioned system:

$$M^{-1}Ax = M^{-1}b.$$

- The symmetric positive definite matrix M must be chosen s.t. the system $Mz = r$ is solved with less computational work than the original one. $M^{-1}A$ has a more 'favourable' conditional number than A.

Hessian-Free Newton Method

The general idea behind the algorithm is as follows:

Algorithm 2 Hessian-Free Newton Method

Let $i = 0$ and $x_i = x_0$ be our initial guess;
while Stoping test not satisfied **do**
    At x_n compute $\nabla f(x_n)$ and $H(f)(x_n)$ and consider taylor expansion of f:

$$f(x + \Delta x) \approx f(x) + \nabla f(x)^T \Delta x + \Delta x^T H(f) \Delta x.$$

    Compute x_{n+1} with C.G. for quadratic functions on the Taylor expansion.
end while

Index

The quasi-Newton Preconditioner I

- We wish to accelerate the C.G. iteration used in Hessian-free Newton methods for nonlinear optimization. We want to solve the following problem:

$$A_k x = b_k, \quad k = 1, ..., t,$$

  where A_k is the Hessian of the objective function at the current iterate. A_k vary slowly, and b_k are arbitrary.
- Also interested in solving finite element problems, so a sequence of linear systems:

$$Ax = b_i, \quad i = 1, ..., t.$$

- Both cases the coefficient matrices are symmetric and positive define.

The quasi-Newton Preconditioner II

- We deal with the large scale unconstrained optimization problem:

$$\min f(x)$$

where f is a C^2 of n variables.

- Among the iterative methods for large scale unconstrained optimization, when the Hessian matrix is possibly dense, limited memory quasi-Newton methods are often the methods of choice.
- They generate a sequence x_k, according to the following scheme ([NW99]):

$$x_{k+1} = x_k + \alpha p_k, \quad k = 0, ...$$

with $p_k = -H_k \nabla f(x)$, where H_k is an approximation of the inverse of the Hessian matrix and α_k is a steplength.

- Instead of computing H_k at each iteration k, these methods update H_k in a simple manner, in order to obtain the new approximation H_{k+1} to be used in the next iteration.

The quasi-Newton Preconditioner III

- Moreover, instead of storing full dense $n \times n$ approximations, they only save a few vectors of length n, which allow to represent the approximations implicitly.
- L-BFGS method is usually considered very efficient. Well suited for large scale problems because the amount of storage is limited and controlled by the user.
- This method is based on the construction of the approximation of the inverse of the Hessian matrix, by exploiting curvature information gained only from the most recent iterations.

Automatic Preconditioning by limited Memory Quasi-Newton Updating

Algorithm 3 Automatic Preconditioning limited Memory Quasi-Newton Updating

Solve $\{x_1\}$, $\{r_1\} \leftarrow A_1 x = b_1$ with unpreconditioned CG-method;
for $i = 2, ..., k$ **do**
    Compute/store: $s_i = x_{i+1} - x_i, \quad y_i = r_{i+1} - r_i, \quad i = l_1, ..., l_m$;
    Define BFGS matrix H_k using $\{s_i\}, \{y_i\}$; (quasi-Newton preconditioner)
    $\{x_i\}$, $\{r_i\} \leftarrow$ Use H_k preconditioned CG-method to solve $A_i x = b_i$;
end for

Observation:

- The parameter m determines the amount of memory in the preconditioner.
- Two strategies to select the m vectors: 1. Select the last m vector generated during C.G. iteration. 2. Take a uniform sample of them.
- We don't need to compute the Hessian, we need to compute $H(m)v$, for any vector v. Can be done by finite diferences (approximation).

Index

Results

- Test the preconditioned-method in non linear optimization problems and in linear systems arising in finite element models.
- Non linear optimization problems, there is a substantial reduction in the number of C.G. iterations, when $m = 8$. For beyond $m = 10$ most results are indentical to $m = 8$.
- For tight tolerance, the benefit can be modest. But for relaxed tolerance the saving number of C.G iterations are important.
- With Finite element Matrices, there is also a reduction in the number of C.G. iterations. No reduction of CPU time because marices are very sparce.
- **Comparing sampling strategies**: In general a uniform sampling strategy perform better than saving the last m pairs.

References I

Giovanni Fasano and Massimo Roma.
Quasi–newton updates for preconditioned nonlinear conjugate gradient methods.

José Luis Morales and Jorge Nocedal.
Automatic preconditioning by limited memory quasi-newton updating.
SIAM Journal on Optimization, 10(4):1079–1096, 2000.

Jorge Nocedal and Stephen J Wright.
Numerical optimization.
Springer, 1999.