Classical Methods for Multidimensional Unconstrained Optimization

Dr. José Ernesto Rayas-Sánchez

1

Outline

- Unconstrained optimization problems
- Generic line search method
- Conjugate gradient methods
- Global and local convergence
- Rate of convergence
- Newton's methods
- Damped Newton method
- Quasi-Newton Methods

Unconstrained Optimization Problems

Formulation:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} u(\boldsymbol{x})$$

- $\boldsymbol{x} \in \Re^n$
- u: $\Re^n \rightarrow \Re$

Generic Line Search Algorithm

begin
  $i = 0$, $\boldsymbol{x}_i = \boldsymbol{x}_0$
  repeat until *StoppingCriteria*
    $\boldsymbol{d}_i$ = SearchDirection(u, $\boldsymbol{x}_i$)
    α_i = LineSearch(u, $\boldsymbol{x}_i$, $\boldsymbol{d}_i$)
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{d}_i$
    $i = i + 1$
end

Conjugate Gradient Methods

- They are much better than the steepest descent methods
- They are slower than Newton-type methods, but do not require to perform matrix operations (only vector operations)
- They generally outperform Newton-type methods when n is large
- Newton methods require $O(n^3)$ operations per iteration, quasi-Newton methods need $O(n^2)$, and conjugate gradient methods need only $O(n)$
- They find a minimizer of a quadratic in n-dimensional space in n iterations

Conjugate Gradient Algorithm

begin
  $i = 0$, $\boldsymbol{x}_i = \boldsymbol{x}_0$, $\beta_i = 0$
  repeat until *StoppingCriteria*
    $\boldsymbol{d}_i = -\nabla u(\boldsymbol{x}_i) + \beta_i \boldsymbol{d}_{i-1}$
    $\alpha_i = \text{LineSearch}(u, \boldsymbol{x}_i, \boldsymbol{d}_i)$
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{d}_i$
    $i = i + 1$
    $\beta_i = \text{UpdateBeta}(u, \boldsymbol{x}_i, \boldsymbol{x}_{i-1})$
end

β_i is selected such that $\boldsymbol{d}_i$ and $\boldsymbol{d}_{i-1}$ are conjugate directions

Conjugate Directions

- Given a symmetric matrix $\boldsymbol{Q} \in \Re^{n\times n}$, a sequence of non-zero directions $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_m$, are Q-conjugate if

$$\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}_j = 0 \text{ for } i \neq j$$

- If $\boldsymbol{Q} \in \Re^{n\times n}$ is a positive definite matrix, the sequence of non-zero Q-conjugate directions $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$, with $k \leq n$, are linearly independent

Updating β

- Fletcher-Reeves (F-R) formula

$$\beta_{i+1} = \frac{\nabla u(\boldsymbol{x}_i)^T \nabla u(\boldsymbol{x}_i)}{\nabla u(\boldsymbol{x}_{i-1})^T \nabla u(\boldsymbol{x}_{i-1})}$$

- Polak-Ribiére (P-R) formula

$$\beta_{i+1} = \frac{[\nabla u(\boldsymbol{x}_i) - \nabla u(\boldsymbol{x}_{i-1})]^T \nabla u(\boldsymbol{x}_i)}{\nabla u(\boldsymbol{x}_{i-1})^T \nabla u(\boldsymbol{x}_{i-1})}$$

- For quadratic objective functions both formulas are equivalent
- Both formulas yield similar performance

Conjugate Gradient Methods (cont)

- They use a conjugate direction: a linear combination of the steepest descent direction and the previous direction
- Some implementations reset the search direction to the steepest descent direction every *n* iterations
- When using exact line searches they have a linear convergence rate
- They have a global convergence property when soft line searches are employed
- When resetting is applied, they are also globally convergent

Global and Local Convergence

- Globally convergent algorithm

$$\lim_{i\to\infty} \boldsymbol{x}_i = \boldsymbol{x}^* \text{ for any } \boldsymbol{x}_0 \in \Re^n$$

- Locally convergent algorithm

$$\lim_{i\to\infty} \boldsymbol{x}_i = \boldsymbol{x}^* \text{ for any } \boldsymbol{x}_0 \in \Omega \subseteq \Re^n$$

Rate of Convergence

Let $\boldsymbol{e}_i = \boldsymbol{x}_i - \boldsymbol{x}^*$ and $\|\boldsymbol{e}_{i+1}\|_2 \leq \|\boldsymbol{e}_i\|_2$ for $i > N$

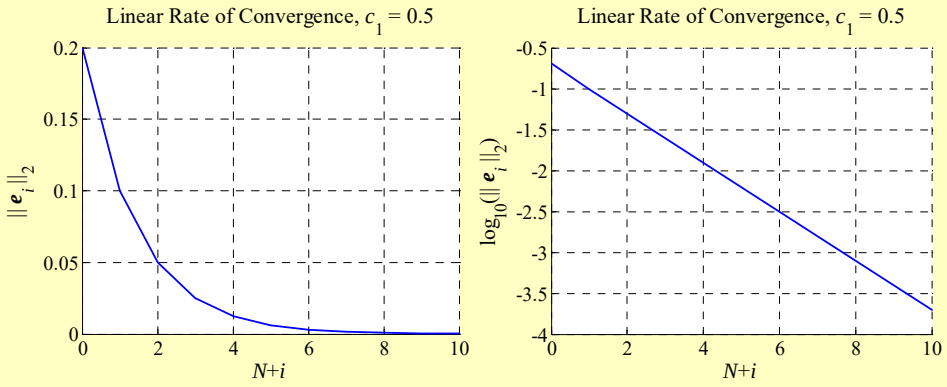
- Linear rate of convergence

$$\|\boldsymbol{e}_{i+1}\|_2 \leq c_1 \|\boldsymbol{e}_i\|_2 \text{ with } 0 < c_1 < 1$$

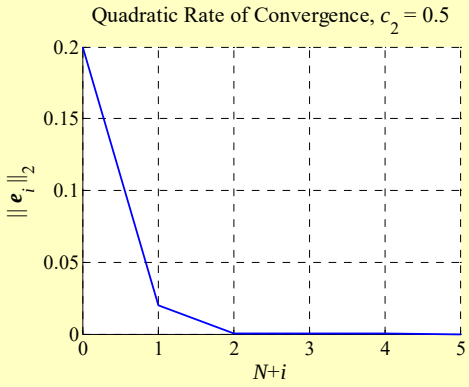
- Quadratic rate of convergence

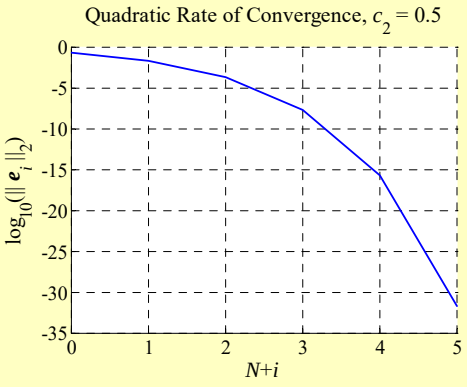
$$\|\boldsymbol{e}_{i+1}\|_2 \leq c_2 \|\boldsymbol{e}_i\|_2^2 \text{ with } 0 < c_2 < 1$$

Linear Rate of Convergence – Example



Quadratic Rate of Convergence – Example





Newton's Methods

- Newton's method find a minimizer of a quadratic in *n*-dimensional space in one iteration
- If the starting point is sufficiently close to a local minimizer and the Hessian remains positive definite, Newton's method converges quadratically towards the minimizer
- It is not globally convergent; it may converge to a maximum or a saddle point; requires second order derivatives

Newton's Algorithm

begin
  $i = 0$, $\boldsymbol{x}_i = \boldsymbol{x}_0$
  repeat until *StoppingCriteria*
    solve $\boldsymbol{H}(u(\boldsymbol{x}_i))\boldsymbol{d}_i = -\nabla u(\boldsymbol{x}_i)$ for $\boldsymbol{d}_i$
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{d}_i$
    $i = i + 1$
end

Newton's Algorithm with Line Search

begin
  $i = 0$, $\boldsymbol{x}_i = \boldsymbol{x}_0$
  repeat until *StoppingCriteria*
    solve $\boldsymbol{H}(u(\boldsymbol{x}_i))\boldsymbol{d}_i = -\nabla u(\boldsymbol{x}_i)$ for $\boldsymbol{d}_i$
    $\alpha_i = \text{LineSearch}(u, \boldsymbol{x}_i, \boldsymbol{d}_i)$
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{d}_i$
    $i = i + 1$
end

This modified Newton's algorithm has a descent property (if $\boldsymbol{H}(u(\boldsymbol{x}_i))$ is positive definite):

$$u(\boldsymbol{x}_{i+1}) < u(\boldsymbol{x}_i)$$

Damped Newton Method

- It is a hybrid between Newton's method and the steepest descent method
- It takes advantage of the global convergence properties of the steepest descent method whenever Newton's method have problems
- It is globally convergent and ill-conditioning may be avoided

A Generic Damped Newton Algorithm

begin
  $i = 0$, $\boldsymbol{x}_i = \boldsymbol{x}_0$, initialize μ
  repeat until *StoppingCriteria*
    solve $[\boldsymbol{H}(u(\boldsymbol{x}_i)) + \mu\boldsymbol{I}]\boldsymbol{d}_i = -\nabla u(\boldsymbol{x}_i)$ for $\boldsymbol{d}_i$
    adjust μ
    if $\boldsymbol{x}_i + \boldsymbol{d}_i$ is acceptable **then**
      $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{d}_i$
      $i = i + 1$
    end
end

Quasi-Newton Methods

- They approximate the Hessian (or its inverse) with some matrix
- Quasi-Newton methods are some of the most powerful methods for solving unconstrained optimization problems

A Generic Quasi-Newton Algorithm

begin
$i = 0$, $\boldsymbol{x}_i = \boldsymbol{x}_0$, $\boldsymbol{B}_i = \boldsymbol{I}$
repeat until *StoppingCriteria*
solve $\boldsymbol{B}_i\boldsymbol{d}_i = -\nabla u(\boldsymbol{x}_i)$ for $\boldsymbol{d}_i$
$\alpha_i = \text{LineSearch}(u, \boldsymbol{x}_i, \boldsymbol{d}_i)$
$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i\boldsymbol{d}_i$
$\boldsymbol{B}_{i+1} = \text{UpdateB}(u, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \boldsymbol{B}_i)$
$i = i + 1$
end

Updating $\boldsymbol{B}$ in Quasi-Newton Methods

- SR1 formula (Symmetric-Rank-One)

$$\boldsymbol{B}_{i+1} = \boldsymbol{B}_i + \frac{(\boldsymbol{y}_i - \boldsymbol{B}_i\boldsymbol{s}_i)(\boldsymbol{y}_i - \boldsymbol{B}_i\boldsymbol{s}_i)^T}{(\boldsymbol{y}_i - \boldsymbol{B}_i\boldsymbol{s}_i)^T \boldsymbol{s}_i}$$

- BFGS formula (Broyden, Fletcher, Goldfarb and Shanno)

$$\boldsymbol{B}_{i+1} = \boldsymbol{B}_i - \frac{\boldsymbol{B}_i\boldsymbol{s}_i\boldsymbol{s}_i^T\boldsymbol{B}_i}{\boldsymbol{s}_i^T\boldsymbol{B}_i\boldsymbol{s}_i} + \frac{\boldsymbol{y}_i\boldsymbol{y}_i^T}{\boldsymbol{y}_i^T\boldsymbol{s}_i}$$

where

$$\boldsymbol{s}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \qquad \boldsymbol{y}_i = \nabla u(\boldsymbol{x}_{i+1}) - \nabla u(\boldsymbol{x}_i)$$

Updating $\boldsymbol{B}$ in Quasi-Newton Method (cont)

- DFP formula (Davidon, Fletcher and Powell)

$$\boldsymbol{D}_{i+1} = \boldsymbol{D}_i + \frac{\boldsymbol{s}_i\boldsymbol{s}_i^T}{\boldsymbol{y}_i^T\boldsymbol{s}_i} - \frac{\boldsymbol{D}_i\boldsymbol{y}_i\boldsymbol{y}_i^T\boldsymbol{D}_i}{\boldsymbol{y}_i^T\boldsymbol{D}_i\boldsymbol{y}_i}$$

where

$$\boldsymbol{D}_i = \boldsymbol{B}_i^{-1} \qquad \boldsymbol{s}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \qquad \boldsymbol{y}_i = \nabla u(\boldsymbol{x}_{i+1}) - \nabla u(\boldsymbol{x}_i)$$