Introduction to Gradient-Based Optimisation

Part 2: Multivariate methods

Dr. J.-D. Müller
School of Engineering and Materials Science,
Queen Mary, University of London
j.mueller@qmul.ac.uk

UK Fluids Network SIG on Numerical Optimisation with Fluids
Cambridge, 8-10 August 2018

Notes

Outline

Some simple multi-variate examples
Multivariate optimality conditions
The steepest descent method
Wolfe conditions for inexact line searches
Newton's Method
Conjugate Gradient methods
Truncated Newton's method
Quasi-Newton methods

Notes

Outline

Some simple multi-variate examples
Multivariate optimality conditions
The steepest descent method
Wolfe conditions for inexact line searches
Newton's Method
Conjugate Gradient methods
Truncated Newton's method
Quasi-Newton methods

Notes

Example I: Tank design

Properties of a tank:

$$\text{Volume of a tank:} \quad V = x_1 x_2 x_3 \tag{1}$$
$$\text{Surface:} \quad S = 2x_1x_2 + 2x_1x_3 + x_2x_3 \tag{2}$$

Express this constraint by eliminating one of the variables, $x_3 = V^* x_1^{-1} x_2^{-1}$
Unconstrained optimisation:

$$\text{Min} \qquad S = 2x_1x_2 + 2V^* x_2^{-1} + V^* x_1^{-1}. \tag{3}$$

Notes

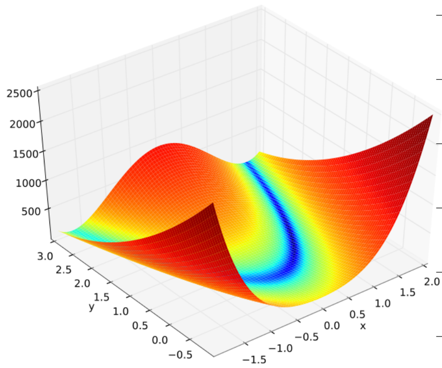
Example II: Rosenbrock function

Bivariate:
$f(x, y) = (1-x)^2 + 100(y - x^2)^2$
Global min. at $[x, y] = [1, 1]$ with $f = 0$.

N-variate:

$$f(\mathbf{x}) = \sum_{i=1}^{N/2} [100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2].$$

$N = 3$: single minimum at $[1, 1, 1]$,
$4 \leq N \leq 7$ two min., $N > 7$ no analytic solution

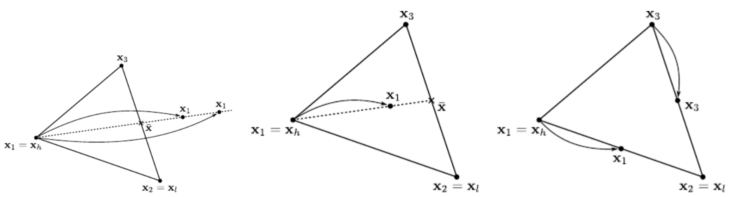


(Source: (Image) Wikipedia)

Notes

Gradient-free: the Nelder-Mead simplex method

- An equivalent of the bisection method, does not require explicit computation of the gradient.
- Reconstruct simple (linear) behaviour by evaluating the function at the vertices of a simplex, e.g. triangle in bi-variate cases:
- Adapt the locations of the vertices to bracket the minimum

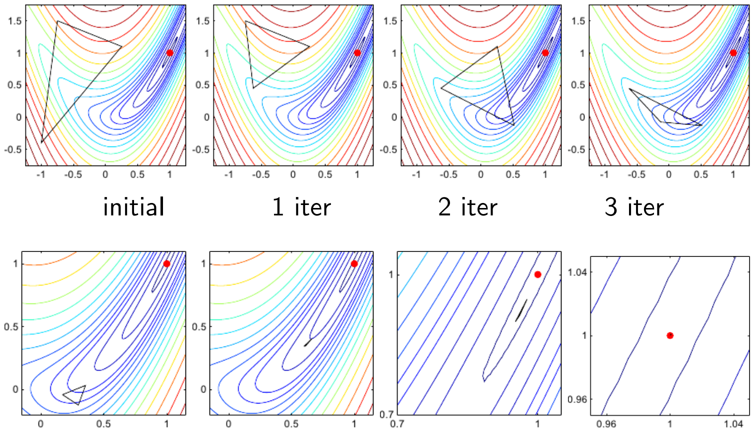


(Source: http://www.brnt.eu/)

For details of the algorithm, see B-B 5.2

Notes

Example: Nelder-Mead on Rosenbrock's function



initial 1 iter 2 iter 3 iter

8 iter 20 iter 70 iter 85 iter

(Source: http://www.brnt.eu/)

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

Conjugate Gradient methods

Truncated Newton's method

Quasi-Newton methods

Notes

Multivariate Optimality conditions I

Taylor expansion in two variables:

$$
\begin{aligned}
&F(x+\delta x, y+\delta y)\\
&\quad= F + F_x\delta x + F_y\delta y + \\
&\qquad \frac{1}{2}(F_{xx}\delta x^2 + F_{xy}\delta x\delta y + F_{yx}\delta y\delta x + F_{yy}\delta y^2) + \\
&\qquad O(\delta x^3, \delta y^3)\\
&\quad= F + [\delta x, \delta y]\begin{bmatrix} F_x \\ F_y \end{bmatrix} + \frac{1}{2}[\delta x, \delta y]\begin{bmatrix} F_{xx} & F_{xy} \\ F_{yx} & F_{yy} \end{bmatrix}\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} + O(\delta x^3, \delta y^3)\\
&\quad= F + s^T\nabla F + \frac{1}{2}s^T\nabla^2 F\, s + O(\delta x^3, \delta y^3)
\end{aligned}
$$

with the step-width $s = [\delta x, \delta y]^T$, the gradient ∇F and the Hessian $\nabla^2 F$.

Notes

Multivariate optimality conditions II

In mono-variate calculus: a local minimum exists for $F(x)$ if

$$\frac{dF}{dx} = F'(x) = 0 \quad \text{and} \quad \frac{d^2F}{dx^2} = F''(x) > 0 \tag{4}$$

If (4) is satisfied for $x = x^*$ and

$$F(x) \geq F(x^*) \quad \text{for all} \quad x, \tag{5}$$

then x^* is a global minimum.

How to extend this to the multi-variate case?

Notes

Multivariate optimality conditions III

$$F(x+s) = F + s^T \nabla F + \frac{1}{2} s^T (\nabla^2 F)\, s + O(\delta x^3, \delta y^3)$$

In multivariate calculus:

1. If $s^T \nabla F < 0$, we have descent.
2. In a stationary point $\nabla F = 0$.
3. In a minimum F increases for any $x \neq x^*$, $F(x) > F(x^*)$ in the vicinity of x^*, i.e. $|x - x^*| < \varepsilon$.
4. That is: $s^T (\nabla^2 F)\, s > 0$ for $|s| < \varepsilon$.
5. A matrix H for which $s^T H\, s > 0$ is called *positive-definite*.

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

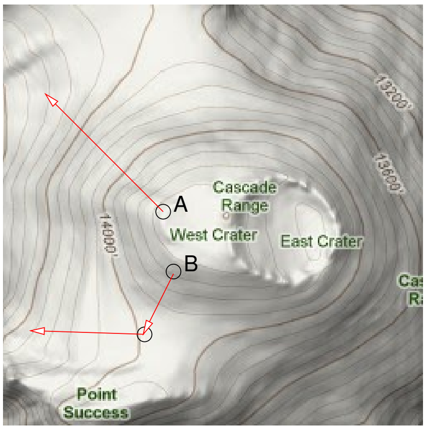
Conjugate Gradient methods

Truncated Newton's method

Quasi-Newton methods

Notes

Steepest Descent



Steepest Descent: evaluate the gradient and follow it.

From A we can descend a long time.

From B we need to limit how far we descend, then pick a new direction at the saddlepoint.

Notes

Steepest descent algorithm

set $k = 1$, $x_k = x_{start}$
do
 compute $F(x_k)$, $\nabla F(x_k)$
 set $p_k = -\nabla F(x_k)$
 find s to minimise $\varphi(s) = F(x_k + s p_k)$! line search
 set $x_{k+1} = x_k + s p_k$
 set $k = k + 1$
while $||\nabla F(x_k)|| \geq \varepsilon$

Finding the *best* s along p_k is called a *line search*

Notes

Exact and inexact line searches

- If we minimise $F(x_k + s\, p_k)$ exactly at each step we perform an *exact* line search.
- At this minimum the search direction p_k becomes orthogonal to the gradient ∇F.
- This is typically very expensive and not very effective, as we are only looking along the gradient line $s\, p_k$.
- Typically *inexact* line searches are used: a *reasonable* reduction in $F(x_k + s\, p_k)$ is sufficient.
- What is *reasonable*?
- We need to formulate *descent conditions.*
- We need to compute an estimate for s.

Notes

Convergence of the steepest descent method

Under the condition that the Hessian (matrix of second derivatives) of F is positive-definite,

$$||x_{k+1} - x^*|| < K||x_k - x^*||$$

i.e. the steepest descent method converges linearly.

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

Conjugate Gradient methods

Truncated Newton's method

Quasi-Newton methods

Notes

First Wolfe condition

First Wolfe condition:

$$p^T g_k \leq -\eta_0 \, ||p|| \, ||g_k||$$

where $g_k = \nabla F(x_k)$. Typically $\eta_0 = 0.01$.

- Recall that the cosine of the angle ϕ between vectors p, g is given as $\cos\phi = \frac{p^T g}{||p|| \cdot ||g||}$.
- This is a stronger condition than $p^T g < 0$.
- This condition requires the angle between $-g$ and p to be smaller than $\text{acos}(\eta_0)$.

Notes

Second Wolfe condition

Second Wolfe condition:

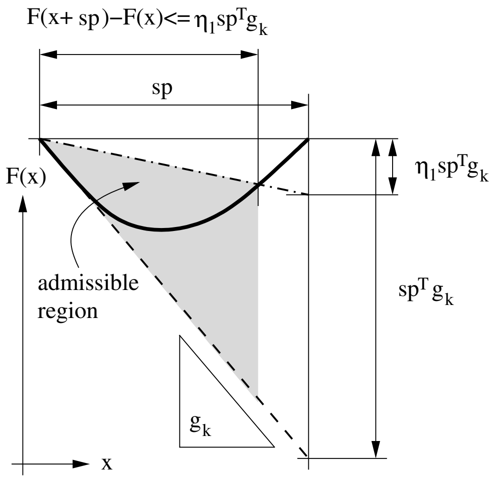
$$F(x_k + s\, p_k) - F(x_k) \le \eta_1 s p^T g_k$$

with $0.0 \le \eta_1 \le 0.5$, typically $\eta_1 = 0.1$.

- requires that the actual decrease $F(x_k + s\, p_k) - F(x_k)$ is at least a fraction η_1 of the predicted linear decrease $sp^T g_k$,
- we can always achieve this by reducing the step s:for an infinitesimally small step $s \to 0$ the linear approximation becomes exact and $F(x_k + s\, p_k) - F(x_k) = sp^T g_k$.

Notes

Second Wolfe condition



- Actual decrease $F(x+sp) - F(x)$ is at least a fraction η_1 of the predicted linear decrease $sp^T g_k$.
- Condition is satisfied for steps that are too small.

Notes

Third Wolfe condition

We want to progress toward the minimum, hence reduce the gradient along the search direction:

$$p_k^T \nabla F(x_k + s\, p_k) \ge (1-\eta_2) p_k^T \nabla F(x_k) = .(1-\eta_2) p_k^T g_k. \qquad (6)$$

(Recall that for descent $p_k^T \nabla F(x_k) = p_k^T g_k < 0$). We compare changes in gradient, so this is also called the 'curvature' condition.

We don't want to evaluate $\nabla F(x_k + s\, p_k)$, but can approximate this using the secant along the search direction p

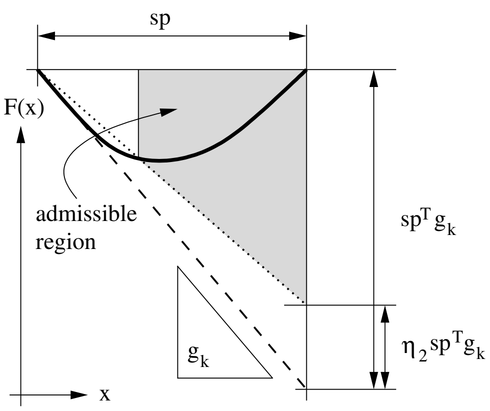
$$\nabla F(x_k + sp_k) \approx \frac{F(x_k + sp_k) - F(x_k)}{s||p||}$$

The curvature condition (6) can then be approximated as

$$\frac{F(x_k + sp_k) - F(x_k)}{s||p||} \ge (1-\eta_2) \frac{p^T g_k}{||p||}$$
$$F(x_k + sp_k) - F(x_k) \ge (1-\eta_2) sp^T g_k$$

Notes

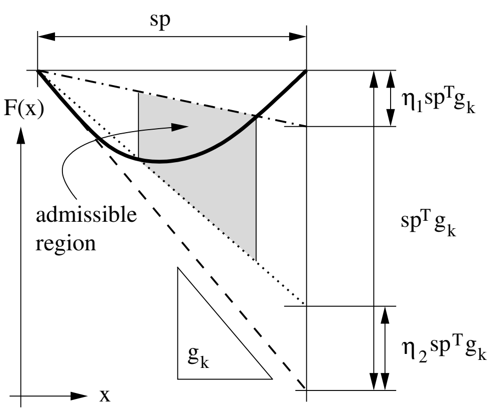
Third Wolfe condition



- Actual slope reduction is at least a fraction $1-\eta_2$, approximating the slope at $x+sp$ using the secant.
- Prevents steps that are too small.

Notes

Armijo conditions



- Combining second and third Wolfe conditions:
- Step is neither too large,
- nor too small.

Notes

Interpretation of Wolfe's conditions

Consider the following expression for the ratio $D(s)$:

$$D(s) = \frac{F(x_k + s\, p_k) - F(x_k)}{sp^T g_k}$$

$s = 0$: Using L'Hôpital's rule, $D(0) = 1$,

$s = \bar{s}$: where $F(x_k + \bar{s}\, p_k) = F(x_k)$, then $D(\bar{s}) = 0$,

$s = s^*$: where s^* minimises $F(x_k + s\, p_k)$, then for a quadratic function $D(s^*) = 0.5$.

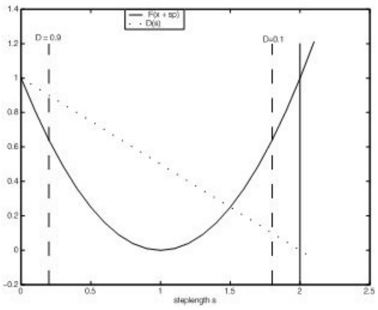
The second Wolfe cond. bounds s away from $\bar{s}$ by enforcing $D \geq \eta_1$,

The third Wolfe cond. bounds s away from 0 by enforcing $D \leq 1 - \eta_2$.

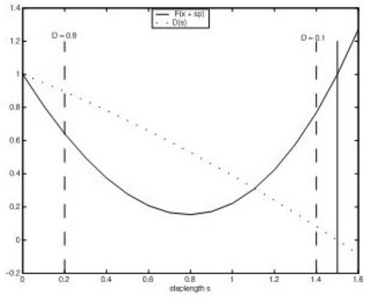
(Source: See B-B, Sec. 8.1)

Notes

Plot of Wolfe conditions



Quadratic function



Non-quadratic function

Notes

Armijo line search

An efficient implementation of Wolfe's conditions:

choose $C > 1$, $c < 1$ and $0 < \eta_1, \eta_2 < 0.5$
set $s = 1$, $s_{min} = 0$! *set first step, track a minimal step*
compute $F(x_k), g_k$
set $p_k = -g_k$
compute $F(x_k + s\, p_k)$, $D(s)$

while (D(s) $> 1 - \eta_2$)
set $s = Cs$, $s_{min} = s$! *step too small, enlarge, update min. step*
compute $F(x_k + s\, p_k)$, $D(s)$
end while
while (D(s) $< \eta_1$ and $s > s_{min}$)
set $s = cs$! *step too large, still larger than s_{min}, reduce*
compute $F(x_k + s\, p_k)$, $D(s)$
end while

Notes

Armijo line search, modified

Estimate the position of the minimum along the line p_k by fitting a quadratic, but limiting the step-size s:

choose $C > 1$, $c < 1$ and $0 < \eta_1, \eta_2 < 0.5$
set $s = 1$, $s_{min} = 0$
compute $F(x_k), g_k$
set $p_k = -g_k$
compute $F(x_k + s\, p_k)$, $D(s)$

while (D(s) $> 1 - \eta_2$)
set set $s = \min(Cs, \frac{0.5s}{1-D(s)})$, $s_{min} = s$! step is too small, enlarge
compute $F(x_k + s\, p_k)$, $D(s)$
end while
while (D(s) $< \eta_1$ and $s > s_{min}$)
set $s = \max(cs, \frac{0.5s}{1-D(s)})$! step too large, still $> s_{min}$, reduce
compute $F(x_k + s\, p_k)$, $D(s)$
end while

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

Conjugate Gradient methods

Truncated Newton's method

Quasi-Newton methods

Notes

Quadratic models

The steepest-descent method only uses first derivatives to determine the search direction, what if we used a quadratic to point us to the minimum $x^* = x + p$?

$$F(x+p) = F(x) + p^T g + \frac{1}{2} p^T G p + O(||p^3||)$$

Gradient and Hessian of Q are

$$\nabla F(x+p) = Gp+g+O(||p^2||), \quad \nabla^2 F(x+p) = \nabla^2 F(x) = G+O(||p^1||)$$

In the minimum $\nabla F(x) = 0$ and G is positive-definite

$$\begin{aligned} p &= -G^{-1} g \\ Gp &= -g \end{aligned}$$

Notes

Newton's method

Netwon's method with a *safeguarded* line-search:

set x_1, $k = 1$! *starting point*
do
  compute $g_k = \nabla F(x_k)$
  if $||\nabla F(x_k)|| > \varepsilon$
    compute $G_k = \nabla^2 F(x_k)$
    if G_k is positive-definite **then**
      solve $G_k p_k = -g_k$! *Netwon*
    else
      $p_k = -g_k$! *Steepest-Descent*
    endif
    find s to minimise $F(x_k + sp_k)$! *line search*
    set $x_{k+1} = x_k + sp_k$, $k = k+1$
  endif
while ($||g_k)|| > \varepsilon$)

Notes

Drawbacks of Newton's method

- The second derivatives in the Hessian, or more efficiently Hessian-vector products) need to be computed, which is complex and expensive
- Multi-variate optimisation problems often are multi-modal with many local extrema. Checking for positive-definiteness requires computation of the full Hessian, which is very expensive in memory and operations.
- It needs *safeguarding*, e.g. with a line-search to avoid divergence in non-convex regions.

Notes

Trust-region methods

So far the approach was a) choose a search direction, then find a function-reducing step-length along it.
Alternatively, fix a step-length (e.g. based on the validity of a quadratic model), then find a minimising direction.

$$\min_p F(x+p) + p^T g_k + \frac{1}{2} p^T G p + O(||p^3||), \quad \text{s.t.} \quad ||p||_2 \leq \Delta.$$

This is equivalent to adding a (sufficiently large) diagonal term to the Hessian, which makes the Hessian diagonally dominant and hence positive-definite. The search direction is

$$(\lambda I + G_k) p_k = -g_k$$

Increase the trust region radius if we find the quadratic model prediction very accurate at x_{k+1}, decrease if very inaccurate.

Notes

Properties of trust region methods

- Better convergence properties than safeguarding with a line-search as we always use a quadratic model
- Rigorously ensures positive-definiteness of the modified Hessian.
- λ is proportional to the inverse of the trust region radius Δ.
- The relationship between trust-region radius Δ and diagonal increment λ is highly non-linear and cannot be determined accurately at low computational cost.
- Hence need to estimate λ.
- Still need to compute expensive second derivatives

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

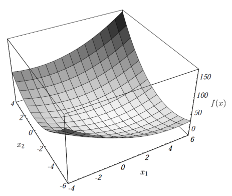
Conjugate Gradient methods

Truncated Newton's method

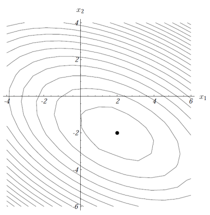
Quasi-Newton methods

Notes

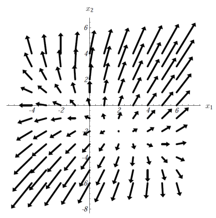
A brief review of Steepest-Descent



Quadratic function



Contours

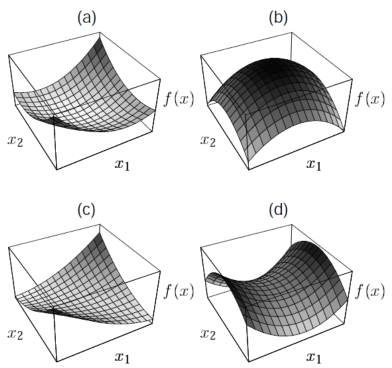


Gradient vectors

(Source: (Images) J. Shewchuck, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain")

Notes

Positive definite matrices



a: positive definite: $x^T Ax > 0$

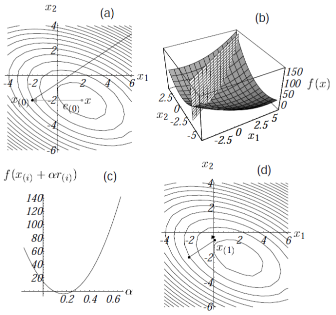
b: negative definite: $x^T Ax < 0$

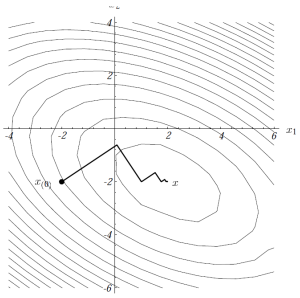
c: positive semi-definite: $x^T Ax \geq 0$

d: indefinite (saddlepoint): $x^T Ax \gtreqless 0$

Notes

Steepest Descent



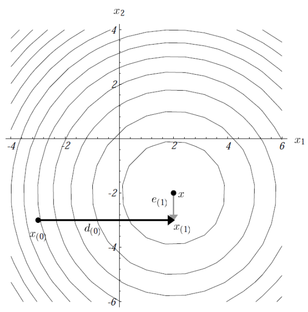


Later steps often repeat an earlier search direction.

Notes

Orthogonal Directions

What if we picked our search directions to solve for each direction only once:

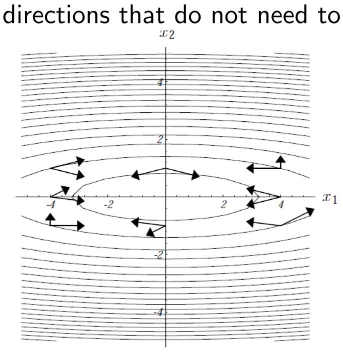


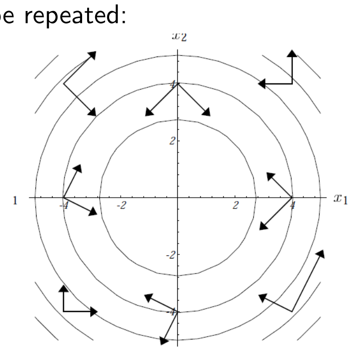
In the general case, we would need to know the solution to be able to do that.

Notes

Orthogonal Directions for quadratic functions

As a special case, if the function is exactly quadratic, we can pick directions that do not need to be repeated:





The directions are orthogonal in a a space scaled by the matrix A, or they are "A-orthogonal.

Notes

The Conjugate Gradient method

Basic idea: Compute the Hessian, but memorise past search directions and make them *conjugate* to each other.
Quadratic model:

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

Stationary point for: $A\,x + b = 0.$

Definition: two vectors u, v are *conjugate* w.r.t a matrix A if

$$u^T Av = 0$$

Notes

Conjugate Gradient algorithm

choose x_0 ! starting point
compute $g_0 = Ax_o + b$
set $p_0 = -g_0$
do k=0,..
  find s to satisfy $p_k^T g_{k+1} = p_k^T (A(x_k + sp_k) + b) = 0$
  set $x_{k+1} = x_k + sp_k$
  exit if $||g_{k+1}|| \leq \varepsilon$
  set $\beta = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$! Fletcher-Reeves
  set $p_{k+1} = -g_{k+1} + \beta p_k$
enddo

Notes

Explanation of C.G.

Computation of the steplength:

$$p_k^T g_{k+1} = p_k^T (A(x_k + sp_k) + b) = 0$$

$$s = -\frac{p_k^T g_k}{p_k^T A p_k}$$

After 2 iterations: $p_1^T g_2 = p_0^T g_2 = 0.$
After k iterations: $p_j^T g_k = 0 \quad \text{for} \quad j = 0, 1, ..., k-1.$

- due to the conjugacy requirement $p_k A p_j = 0$, the search directions form a basis for a k-dimensional space.
- The k-th gradient is orthogonal to all previous p_j.
- The gradient is restricted to a $n-k$-dimensional subspace.
- The C.G. method converges for an n-variate quadratic function in at most n iterations.

Notes

C.G. for non-quadratic functions

- Extension to non-quadratic functions: use C.G. to minimise a local quadratic model. Once this model is (approximately) minimised, restart C.G. with a new model.
- Line-search step can be formulated without computing the Hessian, but exact line search is needed.
- Alternative formulae for β are possible, e.g. Polak-Ribière. They are identical for a quadratic, but differ for a non-quadratic.
- Search directions are no longer truly conjugate, as the Hessian A is no longer constant but changes with x.
- The ultimate convergence rate (near the minimum) is *n-step quadratic*: $||x_k - x^*|| \leq C||x_k - n - x^*||^2$, i.e. slower than Newton and Quasi-Newton.

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

Conjugate Gradient methods

Truncated Newton's method

Quasi-Newton methods

Notes

Truncated Newton: the principle

- The key step in Newton's method is to compute the search direction from the quadratic model solving $Gp = -g$.
- This is expensive in storage and operations
- How about solving $Gp = -g$ only approximately (having ensured that G is pos.-def., and then perform a line-search along p?
- Reduce the computational cost of solving $Gp = -g$, hence iterations become cheaper.
- But lose quadratic convergence, i.e. more iterations.
- Can take advantage of of inexpensive matrix-vector products from algorithmic differentiation (AD), as we can write the iterative solve evaluating only $G_k x$.

Notes

Outline

Some simple multi-variate examples

Multivariate optimality conditions

The steepest descent method

Wolfe conditions for inexact line searches

Newton's Method

Conjugate Gradient methods

Truncated Newton's method

Quasi-Newton methods

Notes

Quasi-Newton methods

We have seen in mono-variate secant methods how to reconstruct a quadratic or cubic from function and gradient values at the bracket endpoints.

Can we perform a similar multi-variate reconstruction from function and gradient values at (nearby) sample points?

Idea: build up a positive-definite approximation of the Hessian H (or better, of its inverse H^{-1}) using sampled gradient values.

Notes

Quasi-Newton method: algorithm

set x_1 ! starting point
set $H_1^{-1} = I$! positive-definite approximation to inverse Hessian
compute $g_k = \nabla F(x_k)$
do $k = 1, ..$
    set $p_k = -H_k^{-1} g_k$
    find s to minimise $F(x_k + sp_k)$! line search
    set $x_{k+1} = x_k + sp_k$
    compute $g_{k+1} = \nabla F(x_{k+1})$
    exit if $||\nabla F(x_k)|| \geq \varepsilon$
    set $\gamma_k = g_{k+1} - g_k$
    set $\delta_k = x_{k+1} - x_k$
    find H_{k+1}^{-1} such that $H_{k+1}^{-1}\gamma_k = -\delta_k$! Quasi-Newton
end do

Notes

Quasi-Newton condition

Where does the Quasi-Newton condition $H_{k+1}^{-1}\gamma_k = -\delta_k$ stem from?
Consider a quadratic function $F(x)$ with gradient g

$$F(x) = \frac{1}{2}x^T Hx + b^T x + c$$
$$g(x) = Hx + b$$

then

$$\begin{aligned}\gamma_k &= g_{k+1} - g_k \\ &= (Hx_{k+1} + b) - (Hx_k + b) \\ &= H(x_{k+1} - x_k) = H\delta_k \\ H^{-1}\gamma_k &= \delta_k\end{aligned}$$

If the function F is quadratic, its Hessian G and the approximated inverse Hessian H^{-1} have the same change in gradient g for the same change in x.

Notes

Computation of the inverse Hessian

Use a *low-rank* update to minimise computational effort and not affect existing gradient information

$$H_{k+1}^{-1} = H_k^{-1} + auu^T \quad \text{or} \quad H_{k+1}^{-1} = H_k^{-1} + buu^T + cvv^T$$

The Davidson-Fletcher-Powell (DFP) update is

$$H_{k+1}^{-1} = H_k^{-1} - \frac{H_k^{-1}\gamma_k\gamma_k^T H_k^{-1}}{\gamma_k^T H_k^{-1}\gamma_k} + \frac{\delta_k\delta_k^T}{\delta_k^T\gamma_k}$$

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) update is

$$H_{k+1}^{-1} = H_k^{-1} - \frac{H_k^{-1}\gamma_k\delta_k^T + \delta_k\gamma_k^T H_k^{-1}}{\delta_k^T\gamma_k} + \left[1 + \frac{\gamma_k^T H_k^{-1}\gamma_k}{\delta_k^T\gamma_k}\right]\frac{\delta_k\delta_k^T}{\delta_k^T\gamma_k}$$

Notes

Computation of the inverse Hessian

- Both DFP and BFGS satisfy the Quasi-Newton condition and ensure positive-definiteness of H_{k+1}^{-1},
- For a perfect line search both updates will produce identical iterates. If F is convex and N-variate, both methods will converge in at most N iterations with $H_N^{-1} = \nabla^2 F^{-1}$.
- BFGS is preferred, as the DFP update is more likely to produce a singular matrix,
- Most popular is the L-BFGS variant that builds up an approximation H to the inverse Hessian using only a the N most recent gradient/variable vectors.

Notes

Quasi-Newton vs. Newton

- Steepest descent has linear convergence, $r = 1$,
- In a convex region with $s \rightarrow 1$ (full Newton step), Quasi-Newton methods can exhibit super-linear convergence,

$$||x_{k+1} - x^*|| = C||x_k - x^*||^r \quad \text{with} \quad 1 < r < 2$$

- Newton's method has quadratic convergence, $r = 2$.
- The operations count is $O(N^2)$ in Q-N, and $O(N^3)$ in N
- Newton's method will have faster convergence
- Quasi-Newton will have lower operation count and simpler implementation.

Notes

Organisation of the lectures

1. Univariate optimisation
   - Bisection
   - Steepest Descent
   - Newton's method
2. Multivariate optimisation
   - Steepest descent and line-search methods:
   - Wolfe and Armijo conditions,
   - Newton's method, Trust-region methods,
   - Conjugate Gradient, Truncated Newton's, Quasi-Newton methods,
3. Constrained Optimisation:
   - Projected gradient methods,
   - Penalty methods,
   - Exterior and interior point methods, SQP
4. Adjoint methods
   - Reversing time
   - Automatic Differentiation
   - Adjoint CFD codes

Notes

Notes