Principles of unconstrained optimization

C1-C2-C3 of " Calculus: Real analysis and optimization"

Professor Manuel Samuelides

manuel.samuelides@isae.fr
ISAE, University of Toulouse
Toulouse, France

2nd November 2010

Outline

Outline

Notations

- $\Omega \subset \mathbb{R}^d$ denotes an open set,
- $x \in \Omega$ will denote a current point of Ω,
- $u \in \mathbb{R}^d$ will denote a current direction of $\mathbb{R}^d$.
- $J : \Omega \to \mathbb{R}$ will be the **objective function** or the **cost function** to minimize on Ω,

Properties of objective functions

- Objective functions will be continuous functions.
- Objective functions will be compact level set functions i.e.

$$\exists\alpha, J^{-1}([\cancel{\alpha, -\infty}[)\text{is compact}$$

Theorem

*Let J a real continuous compact level set function on Ω. There exists $x^\star \in \Omega$ which is a **minimizer** of J i.e.*

$$\forall x \in \Omega, J(x^\star) \leq J(x)$$

- Notation: $x^\star \in \arg\min_{x\in\Omega} J(x) \Leftrightarrow x^\star$ is solution of $\min_{x\in\Omega} J(x)$
- *The proof will be detailed as an exercise*

Gradient and Hessian

Definition

- *If J is differentiable, $\nabla_x J$, the* ***gradient*** *of J at x is the vector of the partial derivatives of J at x.*
- *If J is twice differentiable, $\nabla_x^2 J$, the* ***Hessian*** *of J at x is the symmetric matrix of second partial derivatives of J at x.*

Theorem

Suppose J is twice continuously differentiable, $J \in \mathcal{C}^2(\Omega)$, then

- $J(x+h) = J(x) + (\nabla_x J \mid h) + o(\| h \|)$
- $J(x+h) = J(x) + (\nabla_x J \mid h) + \frac{1}{2}(h \mid \nabla_x^2 J\, h) + o(\| h \|^2)$
- $\nabla_{x+h} J = \nabla_x J + \nabla_x^2 J\, h + o(\| h \|)$

Straightforward consequences on optimization

Corollary

- *If $J \in \mathcal{C}^1(\Omega)$, any minimizer $x^\star$ of J is a stationary point i.e.* $\nabla_{x^\star} J = 0$
- *If $J \in \mathcal{C}^2(\Omega)$ and if $x^\star$ is a minimizer of J, then $\nabla^2_{x^\star} J$ is a non negative symmetric matrix (eigenvalues are positive or null)*

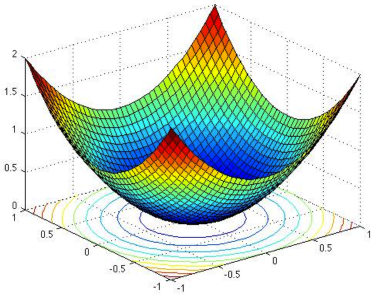
The proofs are easy consequences of previous Taylor approximations.

Example 1: Quadratic function: $J(x, y) = x^2 + y^2$

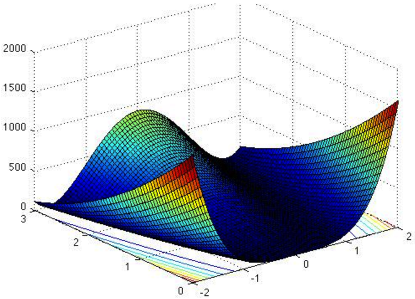
$$J(x,y) = x^2 + y^2$$

$$\nabla J(x,y) = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

$$\nabla^2 J(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$



Example 2: The Rosenbrock banana function $J(x, y) = 100(y - x^2)^2 + (1 - x)^2$



$$J(x,y) = 100\left[(y - x^2)\right]^2 + (1 - x)^2$$

$$\nabla J(x,y) = \begin{pmatrix} -400x(y - x^2) - 2(1 - x) \\ 200(y - x^2) \end{pmatrix} \qquad \nabla^2 J(x,y) = \begin{pmatrix} 1200x^2 + 2 & -400x \\ -400x & 200 \end{pmatrix}$$

Outline

Convex functions

Definition

- *J is said **convex** iff* $\forall\alpha \in]0,1[, \forall(x,y) \in \Omega \times \Omega$, $J(\alpha x + (1-\alpha)y) \leq \alpha J(x) + (1-\alpha)J(y)$
- *J is said **strictly convex** iff the upper inequality is strict*

- A convex function is continuous proof is difficult.
- The hessian of a $\mathcal{C}^2$ convex function is non-negative at any point of the domain (exercise).
- A stationary point of a $\mathcal{C}^\infty$ convex function is a minimizer (exercise).

Minimization of strictly convex functions

Theorem

If a compact level set convex function J is strictly convex, it is minimal at just one point

- $J_1 = x^2 + y^2$ is compact level set and strictly convex,
- The banana function is compact level set and is not convex
- $J_3(x) = \sin(\frac{1}{|x|+1})$ is not a compact level set function

Elliptic functions

Definition

*A $\mathcal{C}^2$ function J is called **elliptic** when its hessian $\nabla_x^2 J$ at every point $x \in \Omega$ is definite positive i.e.*

$$\exists\alpha > 0, \forall h, (h \mid \nabla_x^2 J\ h) \geq \alpha \parallel x \parallel^2$$

Theorem

An elliptic function is strictly convex. It admits at most one stationary point $x^\star$. If it exists, it is the unique minimizer of J.

Proof is easy

Exercise

Discuss compact level set and ellipticity properties of the following functions.

- $J(x, y) = x^2 + y^2$
- $J(x, y) = e^x(4x^2 + 2y^2 + 4xy + 2y + 1)$
- $J(x, y) = 100(y - x^2)^2 + (1 - x)^2$

Determine their minimizers if any. Are these functions convex in the neighborhood of these points ?

Iterative algorithms and finite algorithms

Consider a quadratic objective function

$$J(x) = \frac{1}{2}(x \mid A\,x) + (b \mid x)$$

Suppose that A is definite-positive, then J is strictly convex and the only minimizer $x^\star$ is given by the exact formula

$$x^\star = A^{-1}b$$

- This formula is exact and is supposed to be computed through a finite number of elementary operations.
- However if the size of A is large, iterative solutions of the optimization problem may be less time-consuming to compute the solution with a "reasonable" precision.
- Moreover ill-conditioning of matrix A may induce large computational error of straightforward matrix inversion implementation

Outline

Presentation of a descent algorithm

Algorithm

The basic iteration of an optimization algorithm is

$$x_{n+1} = x_n + h_{n+1} u_{n+1}$$

*where at $(n+1)$th iteration, u_{n+1} is the **descent direction** and h_{n+1} is the **step***

- Sometimes, the descent direction is given without the step (i.e. $h_n = 1$, e.g. Newton method)
- When the descent direction is given, the step is determined through solving a 1d optimization subproblem: the **line-search** problem.

Gradient descent

$$J(x + hu) = J(x) + h(\nabla_x J \mid u) + o(h)$$

- Taylor approximation formula shows near x_n, $u_n = -\nabla_{x_n} J$ is the steepest descent direction. So, steepest descent algorithm choses it at each iteration and determines the step through a line search algorithm.
- The steepest descent algorithm is shown to converge locally with a geometric speed (linear convergence).

Wasting time in steepest descent

Proposition

Let $x_1, x_2 = x_1 + h^* u_1$ be successive steps of steepest descent algorithm, then the successive steepest descent directions are orthogonal, i.e. $(\nabla_{x_2} J \mid \nabla_{x_1} J) = 0$

Proof h^* is a stationary value of $h \to f(h) = J(x_1 + hu)$

$$0 = f'(h^*) = (\nabla_{x_1 + h^* u} J \mid u) = -(\nabla_{x_2} J \mid \nabla_{x_1} J)$$

Notice

During steepest descent algorithm, successive descent directions are orthogonal, steps are smaller and smaller, much computing time is wasted. ***Practically, steepest descent algorithm is never used.***

Outline

Principles of line search algorithm

Problem

The problem consists in searching the minimizer of a $\mathcal{C}^1$ function q defined on the real interval $[h^-, h^+]$

In optimization problem $q(h) = J(x_n + hu_n)$
The algorithm is iterative, each iteration consists into two parts:

1. producing a new value $h \in]h^-, h^+[$
2. testing to select one issue among three
   - (a) $h^- = h$
   - (b) $h^+ = h$
   - (c) select h and stop

Producing a new value

- The simpler way is to choose the middle point according to the **dichotomy method**: $h = \frac{h^- + h^+}{2}$
- It is more precise to choose for h the minimizer of a polynomial approximation $\hat{q}$ of q. For instance if an analytic expression of q' is available, we can choose $\hat{q}$ as the cubic polynomial such that

$$\begin{aligned} \hat{q}(h^+) &= q(h^+) \\ \hat{q}(h^-) &= q(h^-) \\ \hat{q}'(h^+) &= q'(h^+) \\ \hat{q}'(h^-) &= q'(h^-) \end{aligned}$$

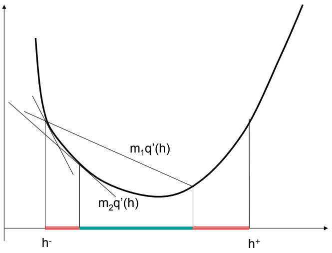
Testing the new value

The simpler test

$$\begin{aligned} q'(h) < -\epsilon \quad &\Rightarrow \quad h^- = h \\ q'(h) > \epsilon \quad &\Rightarrow \quad h^+ = h \\ -\epsilon \leq q'(h) \leq \epsilon \quad &\Rightarrow \quad \text{select } h \text{ and stop} \end{aligned}$$

This simple test is too precise, no need to waste time in achieving the line search algorithm: Wolfe's stopping rule coupled with efficient descent direction search is more efficient.

Wolfe's rule



<u>Wolfe's rule:</u> $0 < m_1 < m_2 < 1$

$$
\begin{aligned}
q(h) \leq q(h^-) + m_1(h - h^-)q'(h^-) \;\&\; q'(h) < m_2 q'(h^-) &\quad \Rightarrow \quad h^- = h \\
q(h) > q(h^-) + m_1(h - h^-)q'(h^-) &\quad \Rightarrow \quad h^+ = h \\
q(h) \leq q(h^-) + m_1(h - h^-)q'(h^-) \;\&\; q'(h) \geq m_2 q'(h^-) &\quad \Rightarrow \quad \text{select } h
\end{aligned}
$$

Outline

Newton's algorithm

Newton's algorithm consists in using the 2nd order approximation at the current point and so from

- $J(x+h) = J(x) + (\nabla_x J \mid h) + \frac{1}{2}(h \mid \nabla_x^2 J\, h) + o(\| h \|^2)$
- $\nabla_{x+h} J = \nabla_x J + \nabla_x^2 J\, h + o(\| h \|)$

infer the following current iteration

$$x_{n+1} = x_n - [\nabla_{x_n}^2 J]^{-1} \nabla_{x_n} J$$

It is possible to prove for smooth strictly convex objective function the quadratic convergence of the algorithm in a neighbourhood of the solution.

Problems of Newton's algorithm

Nice local convergence property BUT

- One has to compute the hessian and to invert it (may be very costly)
- Stability of the algorithm is not ensured, local convergence is hard to forecast.
- These problems are much more difficult to face if the convexity is not ensured everywhere: Newton algorithm may converge to stationary points which are not minimizers.

Outline

Quasi-Newton's approximation

Algorithm

1. *Compute the descent direction:* $u_n = -W_{n-1}\nabla_{x_{n-1}}J$.
2. *Find* $x_n = x_{n-1} + h_n u_n$ *using a line search.*

We have to compute a matrix W_n which approaches $(\nabla^2_{x_n}J)^{-1}$.
It has to check **quasi -Newton equation** with $g_n = \nabla_{x_n}J$

$$W_n(g_n - g_n - 1) = x_n - x_{n-1}$$

The first solution is produced by Broyden:

$$W_n = W_{n-1} + \frac{(\delta x_n - W_{n-1}\delta g_n)\otimes\delta g_n}{(\delta g_n|\delta g_n)} \text{ with } \begin{cases} \delta x_n = x_n - x_{n-1} \\ \delta g_n = g_n - g_{n-1} \end{cases}$$

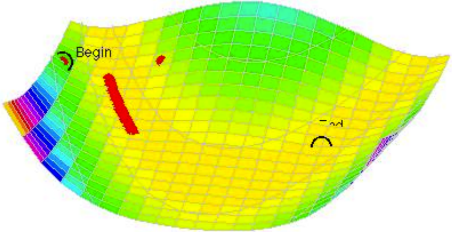
BFGS method

Broyden's solution checks quasi-Newton equation but is not symmetric and may lead to spurious solutions. The following BFGS solution of Quasi-Newton equation is symmetric positive

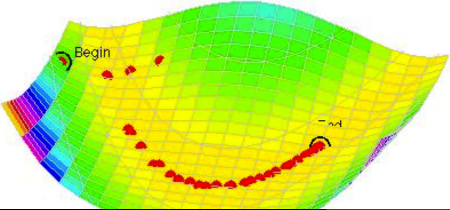
$$W_n = W_{n-1} - \frac{(\delta x_n \otimes \delta g_n) W_{n-1} + W_{n-1} (\delta x_n \otimes \delta g_n)^t}{(\delta g_n \mid \delta x_n)} + \left[1 + \frac{(\delta g_n \mid W_{n-1} \delta g_n)}{(\delta g_n \mid \delta x_n)} \right] \frac{\delta x_n \otimes \delta x_n}{(\delta g_n \mid \delta x_n)}$$

- The Newton approximation is improved by a search line along BFGS descent direction. Wolfe's line search with $m_1 < 0.5$ is generally chosen.
- Theoretical results ensure superlinear vcnovergence of BFGS coupled with Wolfe's search line algorithm (see [1])

Example with Banana function



Minimization through Steepest descent (56 iterations)



Minimization through BFGS algorithm (34 iterations)

Outline

Solving quadratic programming into a finite number of iterations

Problem

Let A be a definite positive symmetric matrix, set

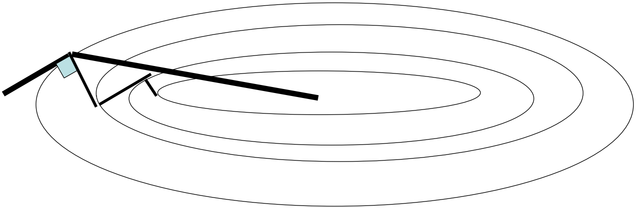
$$J(x) = \frac{1}{2}(x \mid Ax) + (b \mid x), \;\; \nabla_x J = Ax + b$$

$$x^\star = \arg\min_{x \in \mathbb{R}^d} J(x) \Leftrightarrow x^\star = A^{-1}b$$

The conjugate gradient method is a powerful algorithm which

- avoids the implementation of an infinite convergent descent algorithm(is completed through d iterations)
- avoids the inversion of the matrix A to compute $x^\star = A^{-1}b$

Comparison between conjugate gradient and steepest descent



Comparison between speepest descent and conjugate gradient for an ill-conditioned quadratic form.
The problem lies in the anisotropy of the elliptic level sets which is characteristic of a great difference between eigen-values of the quadratic form

Sequential optimization

Theorem

Consider the affine space $V_n = x_n + \mathbb{R}u_1 + ... + \mathbb{R}u_n$ we have

$$x_n = \arg\min_{x \in V_n} J(x) \Leftrightarrow \forall p = 1...n, \ (Ax_n + b \mid u_p) = 0$$

Now, suppose we want to minimize J on $V_{n+1} = V_n + \mathbb{R}v$.

Corollary

With the previous notations, the best descent direction checks:
If $x_{n+1} = x_n + h_{n+1}u_{n+1} = \arg\min_{x \in V_{n+1}} J(x)$ then
$\forall p = 1...n, \ (Au_{n+1} \mid u_p) = 0$

Comes from $\nabla_{x_{n+1}} J = Ax_{n+1} + b = (Ax_n + b) + h_{n+1}Au_{n+1}$

Conjugate directions

Definition

*Two directions u and v are said **conjugate** w.r.t. the symmetric definite positive matrix A when $(Au \mid v) = 0$*

The good new is that keeping the conjugation with the last direction in the plane $x_n + \mathbb{R}g_{n+1} + \mathbb{R}u_n$

$$(u_{n+1} \mid Au_n) = 0$$

ensures conjugation with all the previous directions.
This good property is complex to show and is no more ensured if the objective function is not quadratic.

Outline

Algorithm description

Initialization

$$u_0 = 0, x_0, g_0 = \nabla_{x_0} J = Ax_0 + b$$

Current step

- Conjugate descent direction $u_{n+1} = -g_n + \alpha_{n+1} u_n$

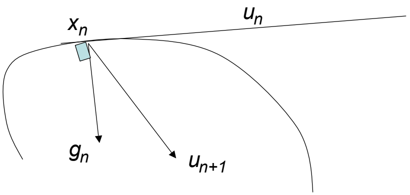
$$(u_n + 1 \mid Au_n) = 0 \Rightarrow \alpha_{n+1} = \frac{(g_n \mid Au_n)}{(u_n \mid Au_n)}$$

- Best step $h_{n+1} = \arg\min_h J(x_n + hu_{n+1})$
  Gradient computation: $g_{n+1} = \nabla_{x_{n+1}} J = g_n + h_{n+1} Au_{n+1}$

$$h_{n+1} = \frac{(-g_n|u_{n+1})}{(u_{n+1}|Au_{n+1})} = \frac{(-g_n|-g_n+\alpha_{n+1}u_n)}{(Au_{n+1}|-g_n+\alpha_{n+1}u_n)} = -\frac{\|g_n\|^2}{(Au_{n+1}|g_n)}$$

- Update $x_{n+1} = x_n + h_{n+1} u_{n+1}$ and $g_{n+1} = g_n + h_{n+1} Au_{n+1}$

Illustration of conjugate gradient



u_{n+1} and u_n are A conjugate directions

Algorithm properties

The good properties of the algorithm are proved by induction in Stieffel theorem

Theorem

1. $\forall p \leq n, (u_{n+1} \mid Au_p) = 0$
2. $\forall p \leq n+1, (g_{n+1} \mid u_p) = 0$
3. $\forall p \leq n, (g_{n+1} \mid g_p) = 0$
4. $\forall p \leq n, (g_{n+1} \mid Au_p) = 0$

Proof

(1) If $p = n,\ (u_{n+1} \mid Au_n) = 0$ is true by construction
If $p < n,\ (g_{n+1} \mid g_p) = -(g_{n+1} \mid u_{p+1}) + \alpha_{p+1}(g_{n+1} \mid u_p)$
follows by induction hypothesis from (2) and (1).

End of Stieffel theorem proof

(2) If $p = n+1$, $(g_{n+1} \mid u_{n+1}) = 0$ is true by construction
If $p \leq n$, $(g_{n+1} \mid u_p) = -(g_n \mid u_p) + h_{n+1}(Au_{n+1} \mid u_p)$
follows by induction hypothesis (2) and by (1).

(3) If $p = n$, $(g_{n+1} \mid g_n) = \| g_n \|^2 + h_{n+1}(Au_{n+1} \mid g_n)$ is true for computed h_{n+1}
If $p \leq n$, $(g_{n+1} \mid u_p) = -(g_{n+1} \mid u_{p+1}) + \alpha_{p+1}(g_{n+1} \mid u_p)$
follows by (2).

(4) $(g_{n+1} \mid Au_p) = \frac{1}{h_p}(g_{n+1} \mid g_p - g_{p-1})$ follows by (3).

Rigorous results

Stieffel theorem states good properties of conjugate gradient in quadratic optimization.

- It shows that conjugate gradient algorithm reaches the minimizer of a strictly convex quadratic objective function within at most d iterations if d is the dimension of the state space.
- It cannot be extended to non quadratic objective since the hessian A is changing at each iteration. In that case, the algorithm is NOT concluded within finite time steps.
- Nevertheless, it is successfully applied as discussed in next slide.

Outline

Extension to non-quadratic functions

The problem is to approximate the computation of conjugate descent direction since A is no more constant

$$u_n = -g_{n-1} + \frac{(g_{n-1} \mid Au_{n-1})}{(u_{n-1} \mid Au_{n-1})} u_{n-1}$$

Fletcher-Reeves algorithm (is proved to converge)

$$u_n = -g_{n-1} + \frac{(g_{n-1} \mid g_{n-1})}{(g_{n-2} \mid g_{n-2})} u_{n-1}$$

Polak-Ribière algorithm (is better in practice)

$$u_n = -g_{n-1} + \frac{(g_{n-1} \mid g_{n-1} - g_{n-2})}{(u_{n-1} \mid g_{n-1} - g_{n-2})} u_{n-1}$$

Summary

- We reviewed unconstrained differential optimization algorithms for smooth functions which are based upon 2nd order approximation.
- These methods are BFGS and conjugate gradient algorithms.
- These algorithms are available in common platforms such as Matlab optimization toolbox (fminunc)

Practical advices

- If convexity is not ensured, local minima or stationary points may be obtained and one has to check different initialization
- The computation of gradient is given by analytical formula or by finite difference computations
- High dimension environment may be difficult for implementing gradient computation
- The structure of sparse matrix may be exploited in special cases
- When the function is not smooth, completely different methods are available (stochastic search, genetic algorithms and so...)

For Further Reading I

J.F.. Bonnans, J.C.. Gilbert, C. Lemaréchal, C.A. Sagastizabal.
Numerical optimization, theoretical and practical aspects ,
Part I, Springer (2009).