Study Notes of Numerical Optimization

Pei Zhong

Update on December 1, 2023

Contents

1 Preliminary Knowledge **4**
- 1.1 Inner Products and Norms . . . 4
  - 1.1.1 Inner Products . . . 4
  - 1.1.2 Vector Norms . . . 5
  - 1.1.3 Matrix Norms . . . 5
- 1.2 Quadratic Form . . . 6
- 1.3 Differentiation . . . 7
- 1.4 Approximation Method . . . 9
  - 1.4.1 First-order Approximation . . . 9
  - 1.4.2 Second-order Approximation . . . 9
  - 1.4.3 Approximation with Integral . . . 9
- 1.5 Reference . . . 9

I Convexity & Unconstrained Optimization **10**

2 Unconstrained Optimization and Optimality Conditions **11**
- 2.1 Reference . . . 11

3 Convex Optimization **12**
- 3.1 General Convex Optimization Problems . . . 12
- 3.2 How to Check Convexity of Functions? . . . 14

4 Descent Algorithms **17**
- 4.1 Motivation . . . 17
- 4.2 Descent Algorithms Overview . . . 17
  - 4.2.1 Iterative Form . . . 17
  - 4.2.2 Stopping Criteria . . . 20
- 4.3 Rates of Convergence . . . 21
- 4.4 Reference . . . 21

5 Gradient Descent **22**
5.1 Convergence 22
5.2 Exact Line Search 22
5.3 Barzilai-Borwein (BB) method 22
5.4 Reference 23

6 Line Search **24**
6.1 Motivation 24
6.2 The Basic Backtracking Algorithm 24
6.3 Convergence for Backtracking Line Search 26
6.4 The Wolfe Conditions 26
6.5 A Bisection Method for the Weak Wolfe Conditions 27
6.6 Convergence for Line Search Based on the Weak Wolfe Conditions 27
6.7 Interpolation Line Search 27
6.8 Reference 27

7 Newton's Method **28**
7.1 Motivation 28
7.2 Pure Newton method 28
7.3 Quadratic Convergence of Newton's Method 29
7.4 Modified Newton Method 30
7.5 Reference 31

8 Conjugate Direction Methods **32**
8.1 Reference 32

9 Quasi-Newton Methods **33**
9.1 Motivation 33
9.2 Newton's Method for Solving Equations 33
9.3 Reference 34

10 Trust Region **35**
10.1 Reference 35

11 Nonlinear Least Squares Problem **36**
11.1 Reference 36

II Duality **37**

Update progress

- writing ch3... 2023.10.20

Preface

all the codes in the notes are completed by python. You can get the codes in the following website:

- -

Chapter 1

Preliminary Knowledge

The goal of this chapter is to refresh your memory on some topics in linear algebra and multivariable calculus that will be relevant to the following content. You can use this as a reference throughout the semester.

1.1 Inner Products and Norms

1.1.1 Inner Products

Without hypotheses, we use $x \in \mathbb{R}^n$ to denote vectors of $\mathbb{R}^n$ with $x = (x_1, x_2, \cdots, x_n)$ and $x \geqslant y$ to denote point-wise order $x_i \geqslant y_i, 1 \leqslant i \leqslant n$. Then, $\mathbb{R}^n_+$ and $\mathbb{R}^n_{++}$ are defined by $\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x \geqslant 0\}$ and $\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x > 0\}$. Besides, e_i is the i-th standard basis $e_i = (0, \cdot, 0, \overset{i}{1}, 0, \cdots, 0)^T$.

Definition 1.1: Inner product and vector norm

(i) For given vectors $x, y \in \mathbb{R}^n$, their **inner product** and the induce **Euclidean norm**(or, l^2**-norm**) are defined by

$$\langle x, y \rangle := x^T y = \sum_{i=1}^{n} x_i y_i$$

$$\|x\|_2 := (\langle x, x \rangle)^{\frac{1}{2}} = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}.$$

(ii) For a given vector $x \in \mathbb{R}^n$, the l^1**-norm** and l^∞**-norm** are defined by

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \|x\|_\infty = \max_{1 \leqslant i \leqslant n} |x_i|.$$

Proposition 1.1: Fundamental properties of norm

For l^p-norm $\|\cdot\|_p, p = 1, 2, \infty$ of the vectors $x, y \in \mathbb{R}^n$, one have
(i) Positive defininteness: $\|x\|_p \geqslant 0$ and $\|x\|_p = 0 \Leftrightarrow x = 0$.
(ii) Positive Homogeneity: $\|kx\|_p = k\|x\|_p$ for $k \geqslant 0$.
(iii) Subadditivity: $\|x + y\|_p \leqslant \|x\|_p + \|y\|_p$.

Remark. Essentially, any functional $p : \mathbb{R}^n \to \mathbb{R}_+$ satisfying the above three properties is a norm of $\mathbb{R}^n$.

The following inequality plays a important role in theoretical analysis.

Theorem 1.1: Cauchy-Schwarz Inequality

$$\langle x, y\rangle = x^T y \leqslant \|x\|_2\|y\|_2, x, y \in \mathbb{R}^n,$$

the equality holds if and only if $x = ky, k \in \mathbb{R}$ i.e. x, y are parallel.

1.1.2 Vector Norms

1.1.3 Matrix Norms

Analogous to vector norm, we could define norm for matrices, here are some frequently used matrix norms:

Definition 1.2: matrix norms

For a given matrix $A \in \mathbb{R}^{n\times n}$, there are some norms as following:

(i) **Frobenius norm**: $\|A\|_F := \left(\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} a_{ij}^2\right)^{\frac{1}{2}}$.

(ii) 2-**norm**, or **spectral norm**: $\|A\|_2 := \max\limits_{\|x\|_2=1} \|Ax\|_2$.

Definition 1.3: condition number

For a given matrix $A \in \mathbb{R}^{n\times n}$, the condition of A is defined as

$$\kappa(A) = ||A||_2 \cdot ||A^{-1}||_2,$$

If A is singular then $\kappa(A) = \infty$. In numerical analysis the condition number of a matrix A is is a way of describing how well or badly the system $Ax = b$ could be approximated. If $\kappa(A)$ is small the problem is well-conditioned and if $\kappa(A)$ is large the problem is rather ill-conditioned.

Remark. If A is a symmetric matrix, another expression for the condition number is $\kappa(A) = |\frac{\lambda_{max}}{\lambda_{min}}|$, where λ_{max} and λ_{min} denote the largest and smallest eigenvalues of A.

Theorem 1.2: Spectral norm characterization

For a given matrix $A \in \mathbb{R}^{n\times n}$,

$$\|A\|_2 = \max_{\|x\|=1} \|Ax\|_2 = \left(\lambda_{\max}(A^TA)\right)^{\frac{1}{2}},$$

in which $\lambda_{\max}(A)$ is the maximal eigenvalues of A. Furthermore, by notation $\lambda(A)$ denoted the set of all eigenvalues of A, we have $\|A\|_2 = \max(|\lambda(A)|)$ when A is symmetric.

Proof. By Cauchy-Schwarz inequality one have

$$\|Ax\|_2^2 = \langle Ax, Ax\rangle = x^T(A^TAx) \leqslant \|x\|_2\left\|A^TAx\right\|_2 = \left\|A^TAx\right\|_2, \forall\, x \in \mathbb{R}^n, \|x\| = 1,$$

and the equality holds if and only if x, A^TAx are parallel, that is $A^TAx = \lambda x$ for some $\lambda \in \mathbb{R}$ i.e. x is an eigenvector of A^TA associated with eigenvalue λ. Hence

$$\max_{\|x\|_2=1} \|Ax\|_2 \leqslant \max_{\|x\|_2=1} \left(\|x\|_2\left\|A^TAx\right\|_2\right)^{\frac{1}{2}} \leqslant \max_{\|x\|_2=1} \left(\|x\|_2(\lambda\|x\|_2)\right)^{\frac{1}{2}} \leqslant (\lambda_{\max}(A^TA))^{\frac{1}{2}},$$

and there is eigenvector x^* associated with maximal eigenvalue $\lambda_{\max}$ of A^TA such that $\|Ax^*\|_2 = (\lambda_{\max}(A^TA))^{\frac{1}{2}}$, therefore $\max\limits_{\|x\|_2=1} \|Ax\|_2 = (\lambda_{\max}(A^TA))^{\frac{1}{2}}$.

When A is symmetric, $\lambda_{\max}(A^TA) = \lambda_{\max}(A^2) = (\max(|\lambda(A)|))^2$, then the conclusion is obvious. □

The next property of 2-norm is common in theoretical analysis. Its proof is similar to theorem 1.2 hence it is remained to an exercise.

Corollary 1.1: Compatibility of 2-norm

For a given $A \in \mathbb{R}^{n\times n}, x \in \mathbb{R}^n$,
(i) $\|Ax\|_2 \leqslant \|A\|_2\|x\|_2$.
(ii) When A is symmetric, the quadratic form has the attained upper bound and lower bound

$$\lambda_{\min}(A)\|x\|_2 \leqslant x^TAx \leqslant \lambda_{\max}(A)\|x\|_2. \tag{1.1}$$

(iii) As a corollary of (ii), A is positive (semi-)definited if and only if $\lambda_{\min}(A) > 0 (\geqslant 0)$.

1.2 Quadratic Form

The theoretical analysis and algorithm design always involve matrix, especially symmetric positive definited matrix. We briefly introduction the concept of quadratic form.

Definition 1.4: Quadratic form

Give a multivariate quadratic function $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(x_1, x_2, \cdots, x_n) = a_{11}x_1^2 + \cdots + a_{nn}x_n^2 + 2a_{12}x_1x_2 + \cdots + 2a_{n-1,n}x_{n-1}x_n = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j,$$

one rewrite f as

$$f(x) = \frac{1}{2}x^TAx, x = (x_1, \cdots, x_n)^T, A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix},$$

such form is called **quadratic form**, the symmetric matrix A is called coefficient matrix for f. Besides, a symmetric matrix A is said to be **positive (semi-)definited** if quadratic form $\frac{1}{2}x^TAx > 0 (\geqslant 0)$ for every nonzero vector $x \in \mathbb{R}^n, x \neq 0$.

Remark. Quadratic form also could be rewritten to inner product form $\frac{1}{2}x^TAx = \frac{1}{2}\langle x, Ax\rangle$. Note that since A is symmetric then $\langle x, Ax\rangle = \langle Ax, x\rangle$. Actually, by definition $\langle x, Ay\rangle = \langle A^Tx, y\rangle$ holds for every $x \in \mathbb{R}^n, y \in \mathbb{R}^m, A \in \mathbb{R}^{n\times m}$.

1.3 Differentiation

Definition 1.5: Gradient and Hessian

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a multivariate function, the partially derivative of component x_i, is defined by

$$\frac{\partial f}{\partial x_i} := \lim_{t\downarrow 0} \frac{f(x + te_i) - f(x)}{t} = f_i'(x_i),$$

that is derivative of function $f(x)$ as a function of x_i alone. By notation $f_i'(x)$ denoted derivative function $\frac{\partial f}{\partial x_i}$, the definition of second-order partially derivative is defined natrually,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} := \frac{\partial f_i'}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\partial f}{\partial x_i}\right).$$

(i) The **gradient** of f, is vector consists of partially derivatives $\nabla f(x) := (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n})^T$.
(ii) The **Hessian** of f, is matrix consists of second-order partially derivatives

$$\nabla^2 f(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Remark. Almost every case we considered, $\nabla^2 f(x)$ is symmetric i.e. $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$.

Definition 1.6: Jacobian

Consider vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, $f = (f_1(x), \cdots, f_m(x))^T$, the **Jacobian** of f is matrix consists of partially derivatives with dimension $m \times n$,

$$J(f) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

Remark. One could see that $\nabla^2 f(x) = J(\nabla f(x))$.

Definition 1.7: vector

A vector is a matrix with only one column. Thus, all vectors are inherently column vectors.

Remark. We follows the convention of the gradient being a column vector, while the derivative is a row vector.

Definition 1.8: Jacobian Matrix

Let $y = f(x)$ where y is an m-element vector, and x is an n-element vector. The symbol

$$\frac{\partial y}{\partial x} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{1.2}$$

will denote the $m \times n$ matrix of first-order partial derivatives of the function from x to y. Such a matrix is called the Jacobian martix of the function f. ($J_{ij} = \frac{\partial y_i}{\partial x_j}$)

Remark. If x is actually a scalar in (1.2), then the resulting Jacobian matrix is a $m \times 1$ matrix; that is, a single column(a vector). On the other hand, if y is actually a scalar in (1.2) , then the resulting Jacobian matrix is a $1 \times n$ matrix; that is, a single row (the transpose of a vector).

Remark. We follows the convention of $\frac{\partial y^T}{\partial x} = (\frac{\partial y}{\partial x})^T$

1.4 Approximation Method

1.4.1 First-order Approximation

1.4.2 Second-order Approximation

1.4.3 Approximation with Integral

1.5 Reference

- matrix norm: lecture notes from sjsu
- quadratic form: lecture notes from mit
- understand matrix norm using function analysis : lecture notes from

Part I

Convexity & Unconstrained Optimization

Chapter 2

Unconstrained Optimization and Optimality Conditions

2.1 Reference

- lecture notes from princeton university

Chapter 3

Convex Optimization

"The great watershed in Optimization is not between linearity and nonlinearity, but convexity and nonconvexity. "

3.1 General Convex Optimization Problems

Definition 3.1: (Convex Set)

a set $\Omega \subset \mathbb{R}^n$ is convex if

$$\forall x, y \in \Omega, t \in [0,1] : x + t(y-x) \in \Omega. \tag{3.1}$$

Remark. $x + t(y-x) \in \Omega \iff ty + (1-t)x \in \Omega$. As x, y are arbitrary points, we can say Ω is convex if $\forall x, y \in \Omega, t \in [0,1] : tx + (1-t)y \in \Omega$.

Remark. This definition is equivalent to saying that all connecting lines lie inside set.

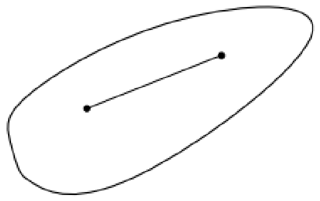
Figure 3.1: an example of a convex set

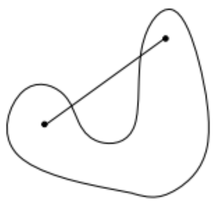
Figure 3.2: an example of a non convex set

Definition 3.2: (Convex Function)

a function $f : \Omega \to \mathbb{R}$ is convex, if Ω is convex and if

$$\forall x, y \in \Omega, t \in [0,1] : f(x + t(y-x)) \leq f(x) + t(f(y) - f(x)). \tag{3.2}$$

Remark. $f(x+t(y-x)) \leq f(x)+t(f(y)-f(x)) \iff f(ty+(1-t)x) \leq tf(y)+(1-t)f(x)$. As x, y are arbitrary points, we can say f is convex function if $\forall x, y \in \Omega, t \in [0,1] : f(tx+(1-t)y) \leq$

$tf(x) + (1-t)f(y)$.

Remark. This definition is equivalent to saying that all secants are above graph.

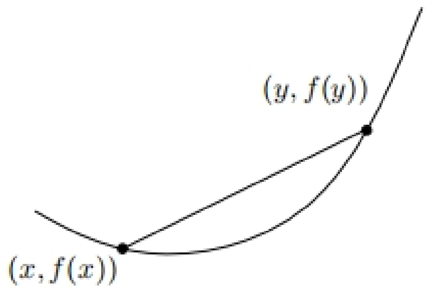

Figure 3.3: For a convex function, the line segment between any two points on the graph lies above the graph

Definition 3.3: (Convex Optimizaiton Problem)

an optimization problem with

- a convex feasible set Ω and
- a convex objective function $f : \Omega \to \mathbb{R}$

is called a ”convex Optimization problem”

Theorem 3.1: (Local Implies Global Optimality for Convex Problems)

for a convex Optimization problem, every local minimum is also a global one.

Proof. Consider a local minimum x^* of the convex optimization problem

$$\min_{x \in \mathbb{R}^n} f(x)$$
$$s.t. x \in \Omega$$

We will show that for each y $\in \Omega$ it holds $f(y) \geq f(x^*)$.

Suppose that x^* is not the global minmium, that is $\exists \, \widetilde{x} \in \Omega \; s.t. \; f(\widetilde{x}) < f(x^*)$.

Consider the line segement $x(t) = tx^* + (1-t)\widetilde{x}, t \in [0,1]$, noting that $x(t) \in \Omega$ by the convexity of Ω. By the convexity of f,

$$f(x(t)) \leq tf(x^*) + (1-t)f(\widetilde{x}) < tf(x^*) + (1-t)f(x^*) = f(x^*), \forall t \in [0,1]$$

As x^* is a local minmium, we know that $\exists N (N$ is a neighbourhood of $x^*)$, $\forall x \in N$, $f(x) \geq f(x^*)$. We can pick t sufficiently to 1 such that $x(t) \in N$. Then $f(x(t)) \geq f(x^*)$. This is a contradiction as $f(x(t)) < f(x^*)$ by the above inequality.

Hence, x^* is the global minimum. □

3.2 How to Check Convexity of Functions?

Theorem 3.2: (Convexity for C^1 Functions)

Assume that $f : \Omega \to \mathbb{R}$ is continuously differentiable and Ω is convex. Then it holds that f is convex if and only if

$$\forall x, y \in \Omega : f(y) \geq f(x) + (\nabla f(x))^T (y - x) \tag{3.3}$$

$i.e.$ tangents lie below the graph.

Remark. The gradient (or gradient vector field) of a scalar function $f(x_1, x_2, x_3, \dots, x_n)$ is denoted ∇f and is a row vector.

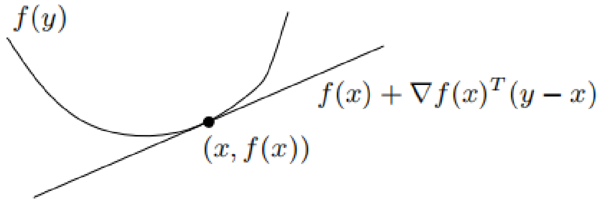

Figure 3.4

Proof. "$\Rightarrow$": If f is convex, by definition, $\forall x \in (0, 1]$,

$$f(x + t(y - x)) \leq f(x) + t(f(y) - f(x))$$
$$\Longrightarrow f(y) - f(x) \geq \frac{f(x + t(y - x)) - f(x)}{t(y - x)} \cdot (y - x)$$

As $t \to 0$, we get

$$f(y) - f(x) \geq (\nabla f(x))^T \cdot (y - x).$$

"$\Leftarrow$": Take any $x, y \in \Omega$, $t \in [0, 1]$ and let

$$z = tx + (1 - t)y.$$

As Ω is convex, we have $z \in \Omega$. Then, by (3.3)

$$f(x) \geq f(z) + (\nabla f(z))^T (x - z)$$
$$f(y) \geq f(z) + (\nabla f(z))^T (y - z).$$

Then,

$$\begin{aligned} tf(x) + (1-t)f(y) &\geq tf(z) + t(\nabla f(z))^T(x-z) + (1-t)(f(z) + (1-t)(\nabla f(z))^T(y-z)) \\ &= f(z) + (\nabla f(z))^T(tx + (1-t)y - z) \\ &= f(z) + (\nabla f(z))^T(z-z) \\ &= f(tx + (1-t)y). \end{aligned}$$

By definition3.2, f is a convex function. □

Definition 3.4: (Generalized inequality for Symmetric Matrices)

B is a symmetric matrix in $\mathbb{R}^{n\times n}$. Define "$B \succcurlyeq 0$" if and only if B is positive semi-definite. i.e.

$$\begin{aligned} B \succcurlyeq 0 &\iff \forall z \in \mathbb{R}^n : z^T B z \geq 0 \\ &\iff \min\{eig(B)\} \geq 0 \end{aligned}$$

Remark. $B \succcurlyeq 0 \iff$ all eigenvalues of B are non-negative real value. B is a symmetric matrix $\Rightarrow$ all eigenvalues of B is real value.

Theorem 3.3: (Convexity for C^2 Functions)

Assume that $f : \Omega \to \mathbb{R}$ is twice continuously differentiable and Ω is convex and open. Then it holds that f is convex if and only if for all $x \in \Omega$ the Hessian is positive semi-definite, i.e.

$$\forall x \in \Omega : \nabla^2 f(x) \succcurlyeq 0. \tag{3.4}$$

Proof. Using Taylor expansion. □

Theorem 3.4

Consider an unconstrained optimization problem

$$\begin{aligned} &\min f(x) \\ &s.t.\ x \in \mathbb{R}^n, \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable. Then,

$$\nabla f(x^*) = 0 \iff f(x) \geq f(x^*), \forall x \in dom(f).$$

Proof. □

Theorem 3.5

Consider an optimization problem

$$\begin{aligned} \min f(x) \\ s.t.\ x \in \Omega, \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable and Ω is convex. Then for any $x^* \in \Omega$,

$$\nabla f(x^*)(y - x^*) \geq 0, \forall y \in \Omega \Longleftrightarrow f(x) \geq f(x^*), \forall x \in \Omega.$$

Proof. □

Chapter 4

Descent Algorithms

4.1 Motivation

Let's recall our unconstrained optimization problem:

$$\min_x f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$.
Where we stand so far:

- Learned about some structural properties of local optimal solutions (first and second order conditions for optimality).
- Learned that for convex problems, local optima are automatically global.

But how to find a local optimum? How to even find a stationary point (i.e., a point where the gradient vanishes)? Recall that this would suffice for global optimality if is convex. We now begin to see some algorithms for this purpose. These will be iterative algorithms: start at a point, jump to a new point that hopefully has a lower objective value and continue.

4.2 Descent Algorithms Overview

4.2.1 Iterative Form

General form of the iterations:

$$x_{k+1} = x_k + \alpha_k d_k$$

where,

- $k \in \mathbb{Z}_+$: iteration number
- $x_k \in \mathbb{R}^n$: current point
- $d_k \in R^n$: direction to move along at iteration k
- $x_{k+1} \in \mathbb{R}^n$: next point

- $\alpha_k \in \mathbb{R}_+$: step size at iteration k.

Goal is to make the sequence $\{f(x_k)\}$ decrease as much as possible. But how to choose d_k? How to choose α_k? In first order method, the direction to move along at step is chosen "based on information from" $\nabla f(x)$". Why is $\nabla f(x)$ a natural vector to look at? Lemmas 4.1 and 4.2 below provide two reasons.

Lemma 4.1

$f \in C^1$. Consider yourself sitting at a point $x \in \mathbb{R}^n$ and looking (locally) at the value of the function f in all directions around you. The direction with the maximum rate of decrease is along $-\nabla f(x)$.

Remark. When we speak of direction, the magnitude of the vector does not matter; e.g., $\nabla f(X), 5\nabla f(x), \frac{\nabla f(x)}{20}$ all are in the same direction.

Proof. Consider a point x, a direction d, and the univariate function

$$g(\alpha) = f(x + \alpha\frac{d}{||d||}).$$

As the rate of change of f at x in direction d(derivative in direction d) is

$$\begin{aligned}\nabla_d f(x) &= \lim_{\alpha\to 0}\frac{f(x+\alpha\frac{d}{||d||}) - f(x)}{\alpha} = \lim_{\alpha\to 0}\frac{g(\alpha)-g(0)}{\alpha} = g'(0)\\ &= (\nabla f(x))^T\frac{d}{||d||} = \frac{1}{||d||} < \nabla f(x), d >,\end{aligned}$$

by the Cauchy-Schwarz inequality, we have:

$$-\frac{d}{||d||}\cdot||\nabla f(x)||\cdot||d|| \leq \frac{1}{||d||} < \nabla f(x), d > \leq \frac{d}{||d||}\cdot||\nabla f(x)||\cdot||d||,$$

which after simplifying gives

$$-||\nabla f(x)|| \leq \frac{1}{||d||} < \nabla f(x), d > \leq ||\nabla f(x)||.$$

So $\nabla_d f(x)$ cannot be larger than $||f(x)||$, or smaller than $-||f(x)||$. However, if we take $d = \nabla f(x)$, the right inequality is achieved:

$$\frac{1}{||\nabla f(x)||} < \nabla f(x), \nabla f(x) > = \frac{1}{||\nabla f(x)||}\cdot||\nabla f(x)||^2 = ||\nabla f(x)||.$$

Similarly, if we take $d = -\nabla f(x)$, then the left inequality is achieved. □

Definition 4.1

For a given point $x \in R^n$, a direction $d \in \mathbb{R}^n$ is called a desent direction, if there exists $\bar{\alpha} > 0$ such that

$$f(x + \alpha d) < f(x), \forall \alpha \in (0, \bar{\alpha})$$

Lemma 4.2

Consider a point $x \in \mathbb{R}^n$. Any direction d satisfying

$$\langle \nabla f(x), d \rangle < 0$$

is a descent direction.(In particular, $-\nabla f(x)$ is a desent direction.)

Remark. The condition $< \nabla f(x), d >< 0$ in Lemma 4.2 geometrically means that the vectors and make an angle of more than 90 degrees (on the plane that contains them).
Why?(Hint: $< \nabla f(x), d >= ||\nabla f(x)|| \cdot ||d|| \cdot cos\theta$)

Proof. By Taylor's theorem, we have

$$f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + o(\alpha).$$

Since $\lim_{\alpha \to 0} \frac{|o(\alpha)|}{\alpha} = 0$, there exists $\bar{\alpha} > 0$ such that

$$\frac{|o(\alpha)|}{\alpha} < |\nabla f(x)^T d|, \forall \alpha \in (0, \bar{\alpha}).$$

This, together with our assumption that $\nabla f(x)^T d < 0$, implies that $\forall \alpha \in (0, \bar{\alpha})$ we must have:

$$\begin{aligned} f(x + \alpha d) &< f(x) + \alpha \nabla f(x)^T d + \alpha |\nabla f(x)^T d| \\ &< f(x) + \alpha \nabla f(x)^T d - \alpha \nabla f(x)^T d \\ &< f(x) \end{aligned}$$

Hence, by Definition 4.1 , d is a descent direction. $\square$

Lemma 4.3

Consider any positive definite matrix B. For any point x, with $\nabla f(x) \neq 0$, the direction $-B\nabla f(x)$ is a descent direction.

Proof. By the assumption that B is positive define, we have

$$\langle \nabla f(x), -B\nabla f(x) \rangle = -\nabla f(x)^T B \nabla f(x) < 0.$$

$\square$

Lemma 4.3 suggests a general paradigm for our descent algorithms:

$$x_{k+1} = x_k - \alpha_k B_k \nabla f(x_k), (\text{with } B_k \succ 0, \forall k).$$

Common choices of descent direction:

- Steepest Descent: $B_k = I, \forall k$.

  Simplest descent direction but not always the fastest.

- Newton Direction: $B_k = (\nabla^2 f(x_k))^{-1}$, assuming $\nabla^2 f(x) \succ 0, \forall k$.

  More expensive, but can have much faster convergence.

- Modified Newton Direction: $B_k = (\nabla^2 f(x_0))^{-1}, \forall k$

  Compute Newton direction only at the beginning, or once every M steps.

Common choices of the step size α_k:

- Constant step size: $\alpha_k = s, \forall k (s > 0)$
  - * Simplest rule to implement, but may not converge if s too large; may be too slow if s too small.
- Diminishing step size: $\alpha_k \to 0$, $\sum\limits_{k=1}^{\infty} \alpha_k = \infty$. (e.g., $\alpha_k = \frac{1}{k}$)
- Minimization rule (exact line search): $\alpha_k = argmin_{\alpha \geq 0} f(x_k + \alpha_k d_k)$
  - * A minimization problem itself, but an easier one (one dimensional).
  - * If f convex, the one dimensional minimization problem also convex
- Successive step size reduction: well-known examples are Armijo rule and Wolf rule. We will cover Armijo in the following chapter.
  - * Try to ensure enough decrease in line search without spending time to solve α_k to optimality.

4.2.2 Stopping Criteria

Once we have a rule for choosing the search direction and the step size, we are good to go for running the algorithm. Typically the initial point x_0 is picked randomly, or if we have a guess for the location of local minima, we pick x_0 close to them. But when to stop the algorithm? Some common choices ($\epsilon > 0$ is a small prescribed threshold):

- $||\nabla f(x)|| < \epsilon$

  If we have $\nabla f(x_k) = 0$, our iterates stop moving. We have found a point satisfying the first order necessary condition for optimality. This is what we are aiming for.

- $|f(x_{k+1}) - f(x_k)| < \epsilon$

  Improvements in function value are saturating.

- $||x_{k+1} - x_k|| < \epsilon$

  Movement between iterates has become small.

4.3 Rates of Convergence

Now we make precise a useful notion of speed or rate of convergence, which is convenient for our convergence analysis.

Definition 4.2

Let $\{x_k\}$ converge to x^*. We say the convergence is of order $p(\geq 1)$ and with factor $\gamma(> 0)$, if $\exists k_0$ such that $\forall k \geq k_0$,

$$||x_{k+1} - x^*|| \leq \gamma ||x_k - x^*||^p.$$

Make sure the following comments make sense:

4.4 Reference

- lecture notes from princeton university

Chapter 5

Gradient Descent

5.1 Convergence

Proposition 5.1: S

ppose f is convex and in C^1. The following statements are equivalent.
(1) Lipschitz continuity of $\nabla f(x)$: there exists an $L > 0$ such that

$$||\nabla f(x) - \nabla f(y)|| \leqslant L||x - y||$$

Theorem 5.1

Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and additionally

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y||, \forall x, y$$

i.e. ∇f is Lipschitz continuous with constant $L > 0$. Then Gradient Descent with $0 \leq \alpha_k \leq \frac{1}{L}$ satisfies

$$f(x_k) - f(x^*) \leq \frac{||x_0 - x^*||^2}{2\alpha_k k}.$$

i.e.

Proof. 1 □

Remark. proof reference: notes from cmu
todo: add theorem when f is strongly convex

5.2 Exact Line Search

5.3 Barzilai-Borwein (BB) method

In the next chapter, we will introduce more about line search to choose appropriate step size.

5.4 Reference

- Convergence analysis: lecture notes from cuhk
- Exact Line Search: lecture notes from princeton university
- BB Method: lecture notes from UCLA

Chapter 6

Line Search

6.1 Motivation

In last chapter, we have introduce the iterative form for an unconstrained optimization problem

$$\min_x f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$. Suppose that x_k is our current best estimate of a solution to the problem. A standard method for improving the estimate x_k is to choose a direction of search $d \in \mathbb{R}^n$ and then compute a step length $\alpha \in \mathbb{R}^n$ so that $x_k + \alpha d$ approximately optimizes f. The new estimate for the solution to the problem is then $x_{k+1} = x_k + \alpha d$. The procedure for choosing α is called a line search method. If α is taken to be the global solution to the following problem

$$min_{\alpha \in \mathbb{R}} f(x_k + \alpha d),$$

then α is called the Curry step length. However, except in certain very special cases, the Curry step length is far too costly to compute. For this reason we focus on a few easily computed step lengths. We begin the simplest and the most commonly used line search method called backtracking.

6.2 The Basic Backtracking Algorithm

In the backtracking line search we assume that $f : \mathbb{R}^n \to \mathbb{R}$ is is differentiable and that we are given a direction d of strict desent at the current point x_k, that is $f'(x_k; d) =< \nabla f(x_k), d >< 0$.

Algorithm 6.2.1: Descent Algorithm with Backtracking Line Search at iteration k

```
input  : $\gamma \in (0, 1)$ , $c_1 \in (0, 1)$, desent direction $d$, inital step size $\hat{\alpha}$ and current solution $x_k$
output: the next solution $x_{k+1}$

1 $\alpha = \hat{\alpha}$
2 while $f(x_k + \alpha d) > f(x_k) + c_1 \alpha f'(x_k; d)$ do
3     $a \leftarrow \gamma\alpha$;
4 $\alpha^* = \alpha$
5 return $x_{k+1} = x_k + \alpha^* d$
```

The backtracking line search method forms the basic structure upon which most line search methods are built. Due to the importance of this method, we take a moment to emphasize its key features.

(i) The search direction d must satisfy

$$f'(x_k; d) < 0.$$

Any direction satisfying this strict inequality is called a direction of strict descent for f at x_k.

(ii) In algorithm, we require that the step length α^* be chosen so that

$$f(x_k + \alpha^* d) \leq f(x_k) + c_1 \alpha^* f'(x_k; d)$$

This inequality is called the Armijo-Goldstein inequality. It is named after the two researchers to first use it in the design of line search routines (Allen Goldstein is a Professor Emeritus here at the University of Washington). Since $f'(x_k; d) < 0$, this inequality guarantees that

$$f(x_k + \alpha^* d) < f(x_k).$$

And by definition 4.1, if d is a direction of strict descent, it is always possible to choose α^* so that $f(x_k + \alpha^* d) < f(x_k)$. But the Armijo-Goldstein inequality is a somewhat stronger statement. We claim that there exists α satisfying the inequality as $f'(x_k; d) < 0$,

$$\lim_{\alpha \to 0^+} \frac{f(x_k + \alpha d) - f(x_k)}{\alpha} = f'(x_k; d) < c_1 f'(x_k; d) < 0.$$

Hence, there is a $\bar{\alpha} > 0$ such that

$$\frac{f(x_k + \alpha d) - f(x_k)}{\alpha} \leq c_1 f'(x_k; d), \forall t \in (0, \bar{t}),$$

that is

$$f(x_k + \alpha d) \leq f(x_k) + c_1 \alpha f'(x_k; d), \forall \alpha \in (0, \bar{\alpha}).$$

(iii) The Armijo-Goldstein inequality is known as a condition of sufficient decrease. It is essential that we do not choose α^* too small. Because $\alpha^* = \gamma^j \hat{\alpha}$, where $j = \min\{j = 0, 1, 2, ... | f(x_k + \gamma^j \hat{\alpha} d) \leq f(x_k) + c_1 \gamma^j \hat{\alpha} f'(x_k; d)\}$. In general, we always wish to choose α^* as large as possible since it is often the factor that some effort was put into the selection of the search direction d. Indeed, as we will see, for Newton's method we must take $\alpha^* = 1$ in order to achieve rapid local convergence.

(iv) There is a balance that must be struck between taking α^* as large as possible and not having to evaluating the function at many points. Such a balance is obtained with an appropriate selection of the parameters γ and c_1. Typically one takes $\gamma \in [0.5, 0.8]$ while $c_1 \in [0.001, 0.1]$ with adjustments depending on the cost of function evaluation and degree of nonlinearity.

6.3 Convergence for Backtracking Line Search

6.4 The Wolfe Conditions

In exact line search method, we want to gain the stepsize $\aleph_k$ satisfying

$$f(x_k + \alpha_k d_k) = \min_{\alpha \in \mathbb{R}} f(x_k + \alpha d_k).$$

Let $g(\alpha) = f(x_k + \alpha d_k)$, the first-order optimality conditions tell us that we should find α satisfying $0 = g'(\alpha) = \nabla f(x_k + \alpha d_k)^T d_k$. The Wolfe conditions try to combine the Armijo-Goldstein sufficient decrease condition with a condition that tries to push $0 = \nabla f(x_k + \alpha d_k)^T d_k$ either toward zero, or at least to a point where the search direction c_k is less of a direction of descent. To describe these line search conditions, we take parameters $0 < c_1 < c_2 < 1$.
Weak Wolfe Conditions

$$f(x_k + \alpha_k d_k) \le f(x_k) + c_1 \alpha_k f'(x_k; d_k) \tag{6.1a}$$
$$c_2 f'(x_k; d_k) \le f'(x_k + \alpha_k d_k; d_k). \tag{6.1b}$$

Strong Wolfe Conditions

$$f(x_k + \alpha_k d_k) \le f(x_k) + c_1 \alpha_k f'(x_k; d_k) \tag{6.2a}$$
$$|f'(x_k + \alpha_k d_k; d_k)| \le c_2 |f'(x_k; d_k)|. \tag{6.2b}$$

The weak Wolfe condition (6.1b) tries to make d_k less of a direction of descent (and possibly a direction of ascent) at the new point, while the strong Wolfe condition (6.2b) tries to push the directional derivative in the direction d_k closer to zero at the new point. Imposing one or the other of the Wolfe conditions on a line search procedure has become standard practice for optimization software based on line search methods.

We now give a result showing that there exists stepsizes satisfying the weak Wolfe conditions. A similar result (with a similar proof) holds for the strong Wolfe conditions.

Lemma 6.1

Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and suppose that $x, d \in \mathbb{R}^n$ are such that the set $\{f(x + \alpha d) | \alpha \ge 0\}$ is bounded below and $f'(x; d) < 0$, then for each $0 < c_1 < c_2 < 1$ the set

$$S = \{\alpha | \alpha > 0, f'(x + \alpha d; d) \ge c_2 f'(x; d) \ and \ f(x + \alpha d) \le f(x) + c_1 \alpha f'(x; d)\}$$

has non-empty interior.

Proof. Let $\phi(\alpha) = f(x + \alpha d) - (f(x) + c_1 \alpha f'(x; d))$. Then

$$\phi(0) = 0$$
$$\phi'(0) = (1 - c_1) f'(x; d) < 0.$$

So there exists $\bar{\alpha} > 0$ such that $\phi(\alpha) < 0$ for $\alpha \in (0, \bar{\alpha})$. Moreover, since $f'(x; d) < 0$ and

$\{f(x+\alpha d)|\alpha > 0\}$ is bounded below, we have $\phi(\alpha) \to +\infty$ as $\alpha \to +\infty$. Hence, by the continuity of f, there exies $\hat{\alpha} > 0$ such that $\phi(\hat{\alpha}) = 0$. Let $\alpha_0 = \inf \alpha|\alpha \geq 0, \phi(\alpha) = 0$.

Since $\phi(0) = \phi(\alpha_0)$ and $\phi(\alpha)$ is continuously differentiable, by Rolle's theorem, there must exists $\widetilde{\alpha} \in (0, \alpha_0)$ with $\phi'(\widetilde{\alpha}) = 0$. That is,

$$\begin{aligned} &\nabla f(x+\widetilde{\alpha}d)^T d - c_1 \nabla f(x)^T d = 0 \\ \Rightarrow &\nabla f(x+\widetilde{\alpha}d)^T d = c_1 \nabla f(x)^T d > c_2 \nabla f(x)^T d. \end{aligned}$$

From the definition of α_0 and the fact that $\widetilde{\alpha} \in (0, \alpha_0)$, we also have

$$f(x+\widetilde{\alpha}d) - (f(x) + c_1\widetilde{\alpha} f'(x;d)) < 0.$$

Hence, $\widetilde{\alpha} \in \inf(S)$ and $\inf(S) \neq \emptyset$. □

6.5 A Bisection Method for the Weak Wolfe Conditions

6.6 Convergence for Line Search Based on the Weak Wolfe Conditions

6.7 Interpolation Line Search

6.8 Reference

- Armijo & Wolfe conditions: lecture note from washington university
- interpolation line search:
  - matlab optimizaiton toolbox tutorial
  - lecture notes from michigan university

Chapter 7

Newton's Method

7.1 Motivation

In this chapter we study the choice of search directions used in our basic updating scheme

$$x_{k+1} = x_k + \alpha_k d_k.$$

for solving

$$\mathcal{P} : \min_{x \in \mathbb{R}^n} f(x).$$

In the previous lecture, we saw the general framework of descent algorithms, with several choices for the step size and the descent direction. Today, we see a wonderful descent method with superlinear (in fact quadratic) rate of convergence: the Newton algorithm.

The Newton's method is nothing but a descent method with a specific choice of a descent direction; one that iteratively adjusts itself to the local geometry of the function to be minimized.

In practice, Newton's method can converge with much fewer iterations than gradient methods. For example, for quadratic functions, while we saw that gradient methods can zigzag for a long time (depending on the underlying condition number), Newton's method will always get the optimal solution in a single step. The cost that we pay for fast convergence is the need to

(1) access second order information (i.e., derivatives of first and second order), and

(2) solve a linear system of equations at every step of the algorithm.

7.2 Pure Newton method

Recall the general form of our descent methods:

$$x_{k+1} = x_k + \alpha_k d_k.$$

Let us have $\alpha_k = 1, \forall k$ for now. So we take full steps at each iteration. (This is sometimes called the pure Newton method.) Newton's method is simply the following choice for the descent direction:

$$d_k = -(\nabla^2 f(x))^{-1} \nabla f(x_k).$$

And iteration only well-defined when $\nabla^2 f(x)$ is invertible.

But Where does d_k come from? One motivation is minimizing a quadratic approximation of the function sequentially.

Around the current iterate x_k, $f(x)$ can be approximated by:

$$f(x) \approx q(x) := f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k),$$

which is the quadratic Taylor expansion of $f(x)$ at $x = x_k$. Notice that $q(x)$ is a quadratic function, which is minimized by solving $\nabla q(x) = 0$. Since the gradient of $q(x)$ is:

$$\nabla q(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k),$$

we therefore are motivated to solve:

$$\nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) = 0,$$

which yields

$$x - x_k = -(\nabla^2 f(x))^{-1} \nabla f(x_k).$$

This leads to the pure Newton method for solving $\mathcal{P}$.

add: solve least square using pure newton method

7.3 Quadratic Convergence of Newton's Method

Proposition 7.1

If $\nabla^2 f(x) \succ 0$ and $d = -(\nabla^2 f(x))^{-1} f(x) \neq 0$, then d is a desent direction.

Proof. It is sufficient to show that $\nabla f(x)^T d = -\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x) < 0$. This will clearly be the case if $(\nabla^2 f(x))^{-1} \succ 0$. Since $\nabla^2 f(x) \succ 0$, if $y \neq 0$,

$$0 < (\nabla^2 f(x)^{-1} y)^T \nabla^2 f(x)(\nabla^2 f(x)^{-1} y) = y^T \nabla^2 f(x)^{-1} y,$$

thereby showing that $\nabla^2 f(x)^{-1} \succ 0$. □

Proposition 7.2

Suppose that $f(x)$ is twice differentiable. Then

$$\nabla f(z) - \nabla f(x) = \int_0^1 \nabla^2 f(x + t(z - x))(z - x)dt.$$

Proof. Let $\phi(t) = \nabla f(x + t(z - x))$. Then $\phi(0) = \nabla f(x)$ and $\phi(1) = \nabla f(z)$, and $\phi'(t) = \nabla^2 f(x +$

$t(z-x))(z-x)$. From Newton-Leibniz theorem, we have

$$\begin{aligned}\nabla f(z)-\nabla f(x) &= \phi(1)-\phi(0)\\ &= \int_0^1 \phi'(t)dt\\ &= \int_0^1 \nabla^2 f(x+t(z-x))(z-x)dt.\end{aligned}$$

□

Theorem 7.1

Suppose $f(x)$ is twice continuously differentiable and x^* is a point for which $\nabla f(x^*) = 0, \nabla^2 f(x^*) \succ 0$. Suppose that $\nabla^2 f(x)$ is Lipschitz continuous at a neighbourhood of x^* (denoted by $\mathcal{N}_\delta(x^*)$), i.e.

$$||\nabla^2 f(x)-\nabla^2 f(y)|| \le L||x-y||, \forall x, y \in \mathcal{N}_\delta(x^*).$$

Then:
(1) iterative sequence $\{x_k\}$ converges to x^* if inital point is close to x^*(local convergence).
(2) the convergence rate of (i) is quadratic.
(3) sequence $\{||\nabla f(x_k)||\}$ converges to 0 and the convergence rate is quadratic.

Proof. Since $\nabla f(x^*)=0$, we have
$\nabla^2 f(x_k)^{-1}$ is pd since x_k close to x^*

$$\begin{aligned}x_{k+1}-x^* &= x_k-\nabla^2 f(x_k)^{-1}\nabla f(x_k)-x^*\\ &= (x_k-x^*)-\nabla^2 f(x_k)^{-1}\nabla f(x_k) \quad = \nabla^2 f(x_k)^{-1}(\nabla^2 f(x)(x_k-x^*)-(\nabla f(x_k)-\nabla f(x^*))).\end{aligned}$$

□

In practice, to guarante the local convergence, we usually firstly use gradient descent to

7.4 Modified Newton Method

There are some drawbacks of pure Newton method:
(1) When $\nabla^2 f(x)$ is not positve-definite, Newton's direction may not be a descent direction.
(2) When the inital point is far away from the optimal point, by choosing $\alpha_k = 1$, we may not be able to obtain a convergent iterative sequence. This is because the Taylor approximation is limited in region.

Hence, something should be done to modified pure Newton's Method. In this section, we firstly introduce the modified Newton method with line search. The basic idea is:
(1) modified $\nabla^2 f(x)$ so that we can get a positve definite matrix.
(2) use line search method to choose step size.

Algorithm 7.4.1: modified Newton method with line search at iteration k

```
input  : current solution $x_k$
output: the next solution $x_{k+1}$
1 determined matrix $E_k$ such that $B_k \overset{def}{=} \nabla^2 f(x_k) + E_k$ is postive-definite and the condition
   number is small.
2 solve the modified newton equation $B_k d_k = -\nabla f(x_k)$ to obtain desent direction $d_k$.
3 use any line search condition to choose stepsize $\alpha_k$.
4 return $x_{k+1} = x_k + \alpha_k d_k$
```

when the dimension is too big, the calculation of modified newton method is also expansive

7.5 Reference

- lecture notes from princeton university
- lecture notes from MIT

Chapter 8

Conjugate Direction Methods

8.1 Reference

- lecture notes from washington university
- lecture notes from mit

Chapter 9

Quasi-Newton Methods

9.1 Motivation

In last chapter, we have known that All of the search directions considered can be classified as Newton-like since they are all of the form

$$d_k = -H_k \nabla f(x_k),$$

where H_k is a symmetric $n \times n$ matrix. If $H_k = \mu_k I$ for all k, the resulting search direction as a scaled steepest descent direction with scale factors mu_k. More generally, we choose H_k to approximate $\nabla^2 f(x_k)^{-1}$ in order to approximate Newton's method for optimization. The Newton method is important since it possesses rapid local convergence properties, and can be shown to be scale independent.

9.2 Newton's Method for Solving Equations

Newton's method is an iterative scheme designed to solve nonlinear equations of the form

$$g(x) = 0, \tag{9.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}^n$ is assumed to be continuously differentiable. Many problems of importance can be posed in this way. In the context of the optimization problem $\mathcal{P}$, we wish to locate critical points, that is, points at which $\nabla f(x) = 0$. We begin our discussion of Newton's method in the usual context of equation solvng.

Assume that the function g in (9.1) is continuously differentiable and that we have an approximate solution $x_0 \in \mathbb{R}^n$. We now wish to improve on this approximation. If x^* is a solution to (9.1), then

$$0 = g(x^*) = g(x_0) + g'(x_0)(x^* - x_0) + o(||x^* - x_0||).$$

Thus, it is reasonable to suppose that the solution to the linearized system

$$0 = g(x_0) + g'(x_0)(x - x_0) \tag{9.2}$$

is even closer. This is Newton's method for finding the roots of the equation $g(x) = 0$. It has one obvious pitfall. The solution $x = x_0 - [g'(x_0)]^{-1}g(x_0)$ exists if and only if $g'(x_0)$ exists.

For the sake of the present argument, we assume that $g'(x_0)^{-1}$ exists. Under this assumption, equation(9.2) defines the iteration scheme,

$$x_{k+1} = x_k - [g'(x_k)]^{-1}g(x_k), \tag{9.3}$$

called the Newton iteration. The associated direction

$$d_k = -[g'(x_k)]^{-1}g(x_k). \tag{9.4}$$

is called the Newton direction. We analyze the convergence behavior of this scheme under the additional assumption that an approximation to $[g'(x_k)]^{-1}$ is available. We denote this approximation by H_k. The resulting iteration scheme is

$$x_{k+1} = x_k - H_k g(x_k). \tag{9.5}$$

Methods of this type are called **Newton-Like methods**.

9.3 Reference

- lecture notes from washington university
- lecture notes from ucsd
- lecture notes from cornell
- lecture notes from ccu

Chapter 10

Trust Region

10.1 Reference

- lecture notes from brigham
- lecture notes from ox
- lecture notes from ox

Chapter 11

Nonlinear Least Squares Problem

11.1 Reference

- thesis from duke university
- lecutre nots from dtu
- lecture nots from vnav

Part II

Duality