Contents

Chapter 1

Introduction and Matlab Tutorial

1.1 Overview of Scientific Computing

Scientific computing. Scientific computation is a discipline of solving science and engineering problems with mathematical modeling and computer simulations. Statistical methods are widely used in image and signal processing, patten recognition, machine learning, genomics, data mining, drug design, computational physics, and so on. To solve a practical problem with computer simulations, it is often composed of three steps: mathematical modelling, numerical methods for the model, and computer implementation.

These give arise to a lot of math problems, and we will focus on the numerical analysis for statistical models. To design a good numerical algorithm, people usually have the following considerations:

1. It only includes addition, subtraction, multiplication, division and logic operations;

2. Theoretical analysis(Accuracy and Stability);

3. Fast implementation. Time and CPU memory;

4. Numerical demonstrations. From simple to complex.

The numerical analysis approach is widely and increasingly used in science and engineering problems. In some disciplines, it becomes the dominate

tools for scientific research. Successful mathematical tools include:

Numerical PDEs (in Fluid dynamics, potential theory, elastic mechanics),

Numerical linear algebra / eigenvalues (in Image processing, nanosciences),

Nonlinear optimization (in Everything such as economics and biology),

Discrete mathematics, combinatorics (in biology and computer sciences) and

Fast Fourier algorithm (in particular in Materials sciences).

Each of them can be an isolated course.

Top 10 algorithms.

In 2000, Jack Dongarra and Francis Sullivan published a list of "The Top Ten Algorithms of the Century" in the journal of *Computing in Science and Engineering*. Their list included:

1. 1946, Metropolis algorithm (Monte Carlo method)

2. 1947, Simplex method for linear programming

3. 1950, Krylov subspace iteration methods

4. 1951, Matrix decomposition approach (Householder)

5. 1957, Fortran compiler

6. 1959-1961, QR algorithm for matrix eigenvalues

7. 1962, Quicksort algorithm: Pick one element as a pivot, separate the rest into piles of big and small elements (as compared with the pivot), and then repeat this procedure on each pile.

8. 1965, Fast Fourier transform (Cooley and Tukey)

9. 1977, Integer relation detection algorithm: Given a bunch of real numbers, say $x_1, x_2, \cdots, x_n$, are there integers $a_1, a_2, \cdots, a_n$ (not all 0) for which $a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$?

10. 1987, Fast multipole algorithm.

Most are deterministic algorithms, and their applications in stochastic com-

putation will be the issues of this course.

The topics of our class will cover:

Approximate theory, numerical integration, numerical algebra, Monte Carlo, numerical optimization, and EM optimization.

1.2 Maximum Likelihood and Bayesian Methods

The maximum likelihood and Bayesian methods are two classes of statistical approaches: The former is with a *maximization problem* for an implicit definition of estimators as solutions, and the latter use an *integration problem* to explicitly represent the estimators.

Maximum likelihood estimation.

The MLE is a popular technique for parameter estimates. Suppose we have independent identically distributed samples $x_1, \cdots, x_n$ from probability density function, $f(x|\theta)$, with $\theta = (\theta_1, \cdots, \theta_k)$ being parameters to be inferred. The likelihood function is defined by the joint density,

$$L(\theta|\mathbf{x}) = f(x_1, \cdots, x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta). \tag{1.1}$$

When the samples are not i.i.d., the likelihood function can be still represented by the joint density. L is a PDF for fixed θ. While, if we consider $x_1, \cdots, x_n$ fixed, L is called the likelihood function of θ. This can be understood the likelihood of different θ with the given samples.

The value of θ (say $\hat{\theta}$), at which $L(\theta|\mathbf{x})$ attains the maximum value, is known as a *maximum likelihood estimator*. People typically calculate the MLE by the use of the logarithmic likelihood function as the log function is a convex function which implies the equivalence of the two minimization problem. The log-likelihood function is of the form,

$$l(\theta|\mathbf{x}) = \log L(\theta|\mathbf{x}), \tag{1.2}$$

which simplifies the numerical computations.

The maximization of the likelihood function is equivalent to finding the root of a system,

$$\nabla_\theta l(\theta) = 0. \tag{1.3}$$

This can sometimes be implemented with the method of least squares for linear regression problem. The nonlinearity may lead to a challenge in numerical algorithms and computational implementation.

Bayesian methods. The Bayesian approach often results in an integration problem. The starting point of the Bayesian approach is that, before the sampling we have a prior knowledge of the parameters θ, specified in a *prior distribution* of density $\pi(\theta)$. Then with the new samples, the joint probability density $f(x_1, \cdots, x_n|\theta)$ for new samples is combined with the prior distribution, resulting in a probability density, $\pi(\theta|\mathbf{x})$, called the *posterior distribution*. According to the *Bayes formula*, this is derived from the joint distribution $f(\mathbf{x}|\theta)\pi(\theta)$,

$$\pi(\theta|\mathbf{x}) = \frac{f(\mathbf{x}|\theta)\pi(\theta)}{\int f(\mathbf{x}|\theta)\pi(\theta)d\theta}, \tag{1.4}$$

where the denominator in the RHS,

$$m(x) = \int f(\mathbf{x}|\theta)\pi(\theta)d\theta, \tag{1.5}$$

is the *marginal density* of of $\mathbf{x}$.

After the Bayes formula, any statistical inference will be based on the posterior distribution. A usual way to obtain the estimators is taking it mean value. The high-dimensional integral leads to a difficulty in computation. We will have several lectures for numerical integration.

1.3 Computer Languages

In learning statistical computation, the S-Plus and R language is a good choice. In this course, we will use the Matlab for the project assignments. While in large scale computation of practical simulations, the C and For-

tran are often the first choice because of their running speed. The key is each student must be familiar with one of these languages.

1.4 Error Analysis

Machine error in float-point arithmetic. Computer represents numbers by using binary numbers. The IEEE standard for a double precision floating point arithmetic (for more details, see web sources such as Wikipedia):

S	EEEEEEEEEEE	FFFFFFFFFF...FFFFFFF
0	$1-11$	$12-63$
$sign$	$Exponent$	$Mantissa$

1. If E=2047 and F is nonzero, then V=NaN ("Not a number")
2. If E=2047 and F is zero and S is 1, then V=-Infinity
3. If E=2047 and F is zero and S is 0, then V=Infinity
4. If $0 < E < 2047$ then $V = (-1)^S \times 2^{(E-1023)} \times (1.F)$, where "1.F" is intended to represent the binary number created by prefixing F with an implicit leading 1 and a binary point.
5. If E=0 and F is nonzero, then $V = (-1)^S \times 2^{(-1022)} \times (0.F)$. These are "unnormalized" values.
6. If E=0 and F is zero and S is 1, then V=-0.
7. If E=0 and F is zero and S is 0, then V=0.

Approximate decimal exponent range: 10^{-308} to 10^{308} (from 2^{-1022} to about 2^{1024}). Accuracy: 16 digits. The largest number can be represented exactly is $2^{53} \approx 9.0 \times 10^{15}$.

Example 1. We calculate the value of $x = 1+1/2^n$, for $n = 0, 1, 2 \cdots$, in MATLAB, and check if $x > 1$ or not. It is found is that when $n \geq 53$,

we have $x = 1$.

$$
\begin{array}{l}
function\ n = onepesp \\
y = 1; \\
n = 0; \\
while\ 1 + y > 1 \\
y = y/2; \\
n = n + 1; \\
end
\end{array}
$$

Example 2.(A Matlab exercise.) Compute the exponential e^x by Taylor expansion:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots,$$

and take $x = -21$. The exact value $e^{-21} \approx 7.582560427911907 \times 10^{-10}$. So simply summing the right hand side gives a solution -3.1649×10^{-9}. Why?
(**Hint**: at $n = 19$ to 21, $x^n/n! > 10^8$.)

Absolute error and relative error.

As numerical algorithms results in an approximate solution, the error analysis plays the most important role in scientific computation. Let x be the exact solution of a problem, and $\tilde{x}$ be its approximate solution. Then the **absolute error** is defined by $|x - \tilde{x}|$, and the **relative error** is defined by $|x - \tilde{x}|/|x|$ provided $x \neq 0$.

The error due to the numerical method is called the *truncation error* or the *methodic error*. For example, truncating the Taylor series of function $f(x)$:

$$P_n(x) = f(0) + \frac{f'(0)}{1!}x + \cdots + \frac{f^{(n)}(0)}{n!}x^n,$$

we have truncation error,

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}x^{n+1}, \quad 0 \leq \xi \leq x.$$

This error is the main concern of the course of numerical analysis.

Wellposedness and condition number.

In some practical problems, small perturbation in initial data may lead to a large error, such a problem is called an *ill-posed problem*. The wellposedness is often measured by the *condition number.* The *condition number* of a function with respect to an argument measures the asymptotically worst case of how much the function can change in proportion to small changes in the argument. A problem with low condition number is called a well-posed problem, or it is called a ill-posed problem.

Mathematically, for a given $f(x)$, x has perturbation $\Delta x = x - \tilde{x}$ and the relative error of x is $\Delta x/x$. The relative error of the function is, $\frac{f(x)-f(\tilde{x})}{f(x)}$. Then the ratio of the relative error gives,

$$\left| \frac{f(x) - f(\tilde{x})}{f(x)} / \frac{\Delta x}{x} \right| \approx \left| \frac{x f'(x)}{f(x)} \right| = C_p,$$

where C_p is called the *conditional number*.

Example. For example, linear system:

$$\begin{cases} x + \alpha y = 1, \\ \alpha x + y = 0. \end{cases}$$

When $\alpha \neq 1$, the solution is given by $x = 1/(1-\alpha^2)$ and $y = -\alpha/(1-\alpha^2)$. The condition number is

$$C_p = \left| \frac{\alpha x'(\alpha)}{x(\alpha)} \right| = \left| \frac{2\alpha^2}{1 - \alpha^2} \right|.$$

Chapter 2

Iteration and Convergence for Nonlinear Equations

Optimization problem is much related to the solution of nonlinear equations. We have known in maximum likelihood estimation is equivalent to solve the nonlinear system $\nabla l(\theta) = \mathbf{0}$. In this chapter, we will learn numerical methods to solution of single-variable nonlinear equations by the iteration method, i.e., solve $f(\theta) = l'(\theta) = 0$.

For a nonlinear function, $f : R^n \to R^n$, solving its **roots**

$$\text{Find } \mathbf{x}^* \in R^n, \quad \text{such that } f(\mathbf{x}^*) = 0.$$

are often important.

For example, the solution of the kepler equation $x - a \sin x = b$ can be not obtained directly. Usually, they are solved by the iteration method.

Iteration

The basic procedure of the iteration method is, for a given initial value, $\mathbf{x}_0$, generating a vector sequence, $\mathbf{x}_k$, for $k = 1, 2, \cdots$ under some rule, until converging to a number which is the solution of the problem. We hope the convergence is fast enough. Here is a definition of the convergence rate.

Definition. Let the limit of $\{\mathbf{x}_k\}$ is $\mathbf{x}^*$, and let $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$. If there exists

a positive r and constant $C > 0$, such that,

$$\lim_{k\to\infty} \frac{||\mathbf{e}_{k+1}||}{||\mathbf{e}_k||^r} < C,$$

where $||\cdot||$ is some norm of vector, then we call the sequence converges to $\mathbf{x}^*$, with **convergence rate** of order r.

Often, if $r = 1$, it is the first order convergence, or linear rate of convergence. If, $r > 1$, we call the superlinear rate of convergence. If $r = 2$, we call the second order convergence.

The methods of finding roots can be also classified into two groups: local solution, and global solution.

2.1 The Bisection Method

The bisection method is the most intuitive method, and the idea is simple and effective. If we know there is a root in the region $[a, b]$, then by the mean value theorem, $f(a)f(b) < 0$. We can divide the region into two parts at $x_0 = (a + b)/2$. Then we have either $f(a)f(x_0) < 0$ or $f(x_0)f(b) < 0$. If $f(a)f(x_0) < 0$ then the next value is $x_1 = (a + x_0)/2$, otherwise, $x_1 = (x_0 + b)/2$. Repeat this procedure until x_k in a very small interval or $|f(x_k)|$ less then the given error tolerance δ.

It can be proved that the error estimate of the bisection method is,

$$|x_n - x^*| \leq \frac{(b-a)}{2^n},$$

and thus

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|} = \frac{1}{2},$$

i.e., the rate of convergence of the bisection is 1.

The disadvantage of the bisection method is that it can only solve one root, and it is limited when $f(a)f(b) > 0$. The bisection method is also difficult for solving high dimensional problem.

2.2 Fixed-point Iteration

The fixed point iteration is a method of computing fixed points of iterated functions.

More specifically, rewrite $f(x) = 0$ into its equivalent form $x = \phi(x)$ (equivalent classes). If x^* satisfies, $x^* = \phi(x^*)$, we call x^* is **a fixed point** of function $\phi(x)$.

Then, given an initial point x_0, the fixed point iteration is

$$x_{k+1} = \phi(x_k),\ k = 0, 1, 2, \ldots, \qquad (*)$$

which gives rise to the sequence $x_0, x_1, x_2, \ldots$, which is hoped to converge to a point x^*. We call ϕ be the **iteration function.** If ϕ is continuous and the sequence is convergent, then one can prove that the obtained x is a fixed point of ϕ, i.e., $\phi(x) = x$.

One possible choice for the iteration function is by taking $\phi(x) = \alpha f(x) + x$, where α is called *scale factor* used to adjust the convergence.

Example. Computing the square root of $a > 0$, which is equivalent to find the root of $x^2 - a = 0$. If we take $\phi(x) = \frac{1}{2}\left(\frac{a}{x} + x\right)$, we have the fixed point problem, $x = \phi(x)$. For example, let us consider $a = 2$ case. We have sequences for two different initial values:

$1.0000, 1.5000, 1.4167, 1.4142, 1.4142$

$10.0000, 5.1000, 2.7461, 1.7372, 1.4442, 1.4145, 1.4142, 1.4142$

Contraction mapping theorem (principle). If the function $\phi(x) \in C[a, b]$, in the iteration method (*) satisfies,

1. $\phi(x) \in [a, b]$, for any $x \in [a, b]$;
2. $\exists\ 0 < L < 1$, such that $|\phi(x) - \phi(y)| \leq L|x - y|$, $\forall\ x, y \in [a, b]$.

Then, there exists a unique fixed point x^* in $[a, b]$. And for any initial $x_0 \in [a, b]$, the sequence converges to x^*.

Proof. Existence. Let $F(x) = x - \phi(x)$. Because of $a \leq \phi(x) \leq b$, it is obvious that $F(a) \geq 0$ and $F(b) \leq 0$. So there is at least one fixed point in $[a, b]$ for ϕ.

Uniqueness. Suppose we have two fixed points x_1^* and x_2^*, then

$$|x_1^* - x_2^*| = |\phi(x_1^*) - \phi(x_2^*)| \leq L|x_1^* - x_2^*| \leq |x_1^* - x_2^*|.$$

The equal sign holds when and only when $x_1^* = x_2^*$.

Convergence. For any $x_0 \in [a, b]$, we have,

$$|x_{n+1} - x_n| = |\phi(x_n) - \phi(x_{n-1})| \leq L|x_n - x_{n-1}| \cdots \leq L^n|x_1 - x_0|.$$

Therefore,

$$|x_{n+k} - x_n| \leq \sum_{m=1}^{k} |x_{n+m} - x_{n+m-1}| \leq L^n \frac{1 - L^k}{1 - L}|x_1 - x_0|.$$

Clearly, $\{x_n\}$ is a Cauchy sequence, and therefore it is convergent. Let $x_n \to x^*$ ($n \to \infty$), then we have $x^* = \phi(x^*)$.

Let $k \to \infty$, we have estimate,

$$|x_n - x^*| \leq \frac{L^n}{1 - L}|x_1 - x_0|.$$

We can see that the convergence rate of the method is 1.

Local convergence theorem. Let x^* be the fixed point of $\phi(x)$ which is a continuously differentiable function on neighborhood of the fixed point, and $\phi'(x^*) < 1$, then there must exist a $\delta > 0$, only if the initial data satisfies $|x - x^*| < \delta$, the sequence converges to x^*.

Proof. The proof is simple. By the continuity of $\phi(x)$, there exists a neighborhood $\Delta : |x - x^*| \leq \delta$, such that for any $x \in \Delta$, we have $|\phi'(x)| \leq L < 1$, then

$$|\phi(x) - x^*| = |\phi(x) - \phi(x^*)| \leq L|x - x^*| \leq |x - x^*| \leq \delta.$$

Thus, for any $x \in \Delta$, $\phi(x) \in \Delta$. Therefore, $\phi(x)$ is the contraction mapping on Δ. So the iteration is convergent for any x_0.

Example. Consider $x^2 - 10x + 21 = 0$, which has a root $x = 3$, in interval $[2.5, 3.5]$. Let us take the fixed point function,

$$\phi_1(x) = (x^2 + 21)/10, \quad \text{and} \ \ \phi_2(x) = \sqrt{10x - 21}.$$

Then, as $|\phi_1'(x)| = x/5 \leq 0.7 < 1$ and $|\phi_2'(x)| = |\frac{5}{\sqrt{10x-21}}| \geq \frac{5}{\sqrt{14}} > 1$, the first iteration sequence is convergent, and the second one is divergent.

2.3 The Newton and Secant Method

The Newton method is sometimes called the *Newton-Raphson* method.

The Newton method is also an iterative method, which is based on the linear approximation of the function around the root. If we guess there is **a root near** x_k with $f'(x_k) \neq 0$, we have Taylor expansion,

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k).$$

Then finding the root $f(x) = 0$ is approximately represented by solving the linear equation,

$$f(x_k) + f'(x_k)(x - x_k) = 0,$$

in which the solution is the form of the fixed point problem,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

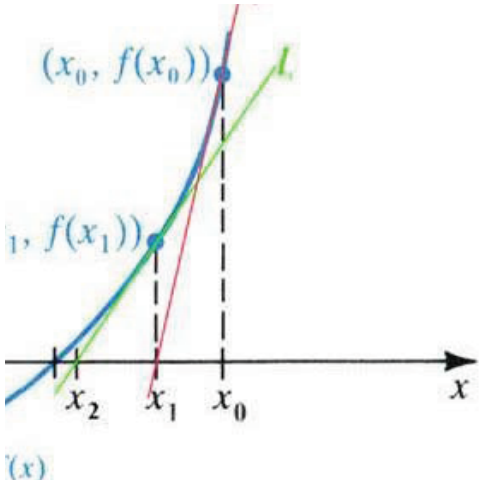
By this procedure, we obtain a series $x_0, x_1, \cdots$, until converging to the root x^*. This is the **Newton's iteration method.**

Picture illustration of how the Newton method works.

Example. Suppose we have the maximum likelihood function, $g(x) = \log x/(1 + x)$, for which the exact maximum is unknown. Construct the Newton scheme for solving the maximum.

Solution. The maximization is equivalent to finding the root $f(x) = g'(x) = 0$, where

$$f(x) = \frac{1 + 1/x - \log x}{(1 + x)^2}. \tag{2.1}$$



and its derivative,

$$f'(x) = \frac{-1 - 4x - 3x^2 + 2x^2 \log x}{x^2(1+x)^3}. \tag{2.2}$$

Then

$$\frac{f(x)}{f'(x)} = \frac{x^2(x+1)(1+1/x - \log x)}{-1 - 4x - 3x^2 + 2x^2 \log x}. \tag{2.3}$$

Fixed-point iteration.
Let $\phi(x) = x - f(x)/f'(x)$, then the Newton method is a fixed-point iteration.

Second order convergence.
Let $\phi(x) = x - \frac{f(x)}{f'(x)}$ be the fixed point iteration function. So $\phi'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$. If x^* is the unique root of $f(x)$, then $f(x^*) = 0$ and $f'(x^*) \neq 0$, then we have $\phi'(x^*) = 0$. By the local convergence theorem, the sequence is convergent.

Furthermore, because $\phi'(x^*) = 0$, we have

$$x_{k+1} - x^* = \phi(x_k) - \phi(x^*) = \phi'(x^*)(x_k - x^*) + \frac{1}{2}\phi''(\xi_k)(x_k - x^*)^2.$$

So $x_{k+1} - x^* = \phi(x_k) - \phi(x^*) = \frac{1}{2}\phi''(\xi_k)(x_k - x^*)^2$, and thus the convergence rate is 2.

Secant method.
The secant method is to use the difference $f'(x_k) \approx \frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$ to substitute the differentiation in the Newton's method. We then have the scheme:

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k),\ k = 1, 2, \cdots$$

In the iteration, we need to use two initial values x_0 and x_1. The convergence rate of the secant method is 1.618.

The case of multiple roots

Recall the Newton's method for the root of $f(x) = 0$,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

which is second order of convergence when $f'(x^*) \neq 0$. As an example, consider the equation $xe^x - 1 = 0$, i.e., $f(x) = xe^x - 1$. The Newton's iteration formula is,

$$x_{k+1} = x_k - \frac{x_k - e^{-x_k}}{1 + x_k}.$$

No Problem!

Multiple root. The problem will turn to be bad if $f'(x^*) = 0$, for example when $f(x) = e^x - x - 1$ which has a root $x^* = 0$. In fact, $f(x)$ can be expressed in the form,

$$f(x) = (x - 0)^2 \frac{e^x - x - 1}{x^2}$$

where the L'Hôpital's rule implies the limit of $(e^x - x - 1)/x^2$ is a finite number 1/2. Numerical convergence of this case will be slow.

Solution of multiple roots. For handling this problem of of multiple roots, one can define,

$$\mu(x) = \frac{f(x)}{f'(x)}.$$

If p is a zero of f of multiplicity m, and $f(x)$ can be naturally written as $f(x) = (x-p)^m q(x)$, then

$$\mu(x) = (x-p)\frac{q(x)}{mq(x) + (x-p)q'(x)}$$

also has a zero at p. However, $q(p) \neq 0$ implies $\mu'(p) \neq 0$ because $\frac{q(p)}{mq(p)+(p-p)q'(p)} = \frac{1}{m} \neq 0$.

There we could use a new Newton's method to $\mu(x)$,

$$x_{k+1} = x_k - \frac{\mu(x_k)}{\mu'(x_k)},$$

or

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{[f'(x_k)]^2 - f(x_k)f''(x_k)},$$

which yields a second-order convergence.

2.4 Newton methods for multidimensional problems

Using the Taylor expansion, we get (both f and x are vectors),

$$f(x^*) = f(x^{(k)}) + \nabla f(x^{(k)})(x^* - x^{(k)}) + O(||x^* - x^{(k)}||^2),$$

where $\nabla f(x^{(k)})$ is the Jacobi matrix,

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1}, \frac{\partial f_1(x)}{\partial x_2}, \cdots, \frac{\partial f_1(x)}{\partial x_n}, \\ \vdots \\ \frac{\partial f_n(x)}{\partial x_1}, \frac{\partial f_n(x)}{\partial x_2}, \cdots, \frac{\partial f_n(x)}{\partial x_n} \end{pmatrix}.$$

Omitting the higher order term yields,

$$f(x^{(k)}) + \nabla f(x^{(k)})(x^* - x^{(k)}) \approx 0,$$

thus we have the iteration scheme,

$$x^{(k+1)} = x^{(k)} - \nabla f(x^{(k)})^{-1} f(x^{(k)}).$$

This is the Newton's iteration.

Note: The inverse of the Jacobi matrix is difficult. A simple improvement will be:
Step 1: Solve the system $\nabla f(x^{(k)})y^{(k)} = -f(x^{(k)})$, and
Step 2: Let $x^{(k+1)} = x^{(k)} + y^{(k)}$.

Quasi-Newton algorithm

Using the scheme,

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)}(M^{(k)})^{-1}f(x^{(k)}).$$

For example $M^{(k)} = -I$ is called ascending algorithm.

Discrete Newton method

Let $M^{(k)} = (M_{ij})_{n\times n}$ and

$$M_{ij} = \frac{f_i(x^{(k)} - h_{ij}^{(k)}e_j) - f_i(x^{(k)})}{h_{ij}^{(k)}}$$

where $h_{ij}^{(k)} = h$ is often used (one order convergence). In general, if $h_{ij}^{(k)} = x_j^{(k)} - x_j^{(k-1)}$ for all i, then the convergence rate is similar to the secant method.

2.5 Project 1. Maximum likelihood estimate by finding root

We have the i.i.d. samples from Cauchy($\theta, 1$) distribution (The pdf is $p(x) = 1/\pi[1 + (x - \theta)^2]$.):

$$1.77,\ -0.23,\ 2.76,\ 3.80,\ 3.47,\ 56.75,\ -1.34,\ 4.24,$$

$$-2.44,\ 3.29,\ 3.71,\ -2.40,\ 4.53,\ -0.07,\ -1.05,\ -13.87,$$

$$-2.53,\ -1.75,\ 0.27,\ 43.21$$

1. Plot the log likelihood function $\ell(\theta)$.

2. Let $f(\theta) = \ell'(\theta)$. Take initial data as $\theta_0 = -1$, write a code with fixed point iteration scheme: $\theta_{k+1} = \theta_k + \alpha f(\theta_k)$, for scale factor $\alpha = 1,\ 0.64,\ 0.25$. Test the convergence and stability.

3. Write the program with Newton's iteration. Find θ of the maximum likelihood estimate by the algorithm by using different initial values. Discuss your result.

4. Analyze your results by comparing two methods.

5. Solving the same problem by the secant method. Choose different initial values for comparison of convergence rate.

6. Using your methods (fixed-point iteration, Newton method, and secant method) to study a set of samples from normal distribution $N(\theta, \sigma^2)$ (supposing θ, or σ^2 is known). Analyze your results.

Chapter 3

Random Number Generators and Integration as an Expectation

3.1 Random number generators of uniform deviates

Uniform deviates are random numbers that lie in a specific range (typically between 0 and 1), where any one number is as likely as any other.

In the early stage of computer invention, von Neumann proposed the mid-square method to generate pseudo random number. For example, $3333^2 = 11108889$, then 1088 is obtained and divided by 10^4 to yield a random number in region $[0, 1]$. This algorithm was used in the early computation of nuclear reaction.

System-supplied RNG in Matlab is RAND(N, M) to generate a $N \times M$ matrix with random entries between 0 and 1.

Linear congruential generator. System-supplied RGCs are almost always linear congruential generators (LCG). A LCG represents one of the oldest and best-known pseudorandom number generator algorithms, which is of the following form,

$$X_{n+1} = aX_n + b \mod m,$$

where m is the modulus, a, b are called the multiplier and the increment, and X_n is the sequence of pseudorandom values.

An important criterion to measure the random number generator is the so called **cycle period.**

The period of a general LCG is at most m, and for some choices of a much less than that. Provided that c is nonzero, the LCG will have a full period for all seed values if and only if:

1. b and m are relatively prime,
2. $a - 1$ is divisible by all prime factors of m,
3. $a - 1$ is a multiple of 4 if m is a multiple of 4.

While LCGs are capable of producing decent pseudorandom numbers, this is extremely sensitive to the choice of the coefficients b, m, and a. A natural choice is $m = 2^k,\ \ a = 4c + 1,\ \ \text{and}\ \ b$ is odd.

However, for scientific purpose, it is warned by Numerical Recipes 3rd that:

1. Never use a generator principally based on a LCG.
2. Never use a generator with a period less than $\sim 2^{64} \approx 2 \times 10^9$ or with disclosed period.
3. Never use the build-in generators.

The LCG is fast, but it has the disadvantage that it is not free of sequential correlation. If k random numbers are used to plot points in k-dimensional space with each coordinate between 0 and 1. Then the points are not tend to fill up the k-dimensional space, but rather will lie on $(k-1)$-dimensional "planes". There will be at most about $m^{1/k}$ such planes, and will be many few than that if the constants are not very carefully chosen.

Minimal standard generator (MSG). There is good evidence, both theoretical and empirical, that the simple multiplicative congruential algorithm

$$X_{n+1} = aX_n \mod m$$

can be as good as any the more general LCGs with $b \neq 0$, if the multiplier a and modulus m are chosen carefully. In 1969, Lewis, Goodman and Miller proposed a LCG, which is known as a "Minimal Standard" genera-

tor (MSG),

$$a = 7^5 = 16807, \quad m = 2^{31} - 1 = 2147483647$$

which passed all theoretical tests.

The cycle period of the MSG is $2^{31} - 2 \approx 2.1 \times 10^9$.

But the MSG may be also questionable. For example, since successive numbers differ by a multiple of only 1.6×10^4 out of a modulus of more than 2×10^9, very small random numbers will tend to be followed by smaller than average values. This may lead to wrong results in studying rare events.

Implementation

The MSG is usually implemented by the Schrage's algorithm which is based on an approximate factorization of m,

$$m = aq + r, \quad i.e., \;\; q = [m/a], \; r = (m \bmod a)$$

where $q = 127773$ and $r = 2836$, then as $z = q[z/q] + (z \mod q)$,

$$az \mod m = \{a(q[z/q] + (z \mod q)) + r[z/q] - r[z/q]\} \mod m$$

that is,

$$az \mod m = \begin{cases} a(z \mod q) - r[z/q] & \text{if it is} \geq 0, \\ a(z \mod q) - r[z/q] + m & \text{otherwise.} \end{cases}$$

Recommended RNG. L'Ecuyer recommended the use of two generators $m_1 = 2147483563$ and $a_1 = 40014$, and $m_2 = 2147483399$ and $a_2 = 40692$, and gave a good way of combining the two different sequences of different periods by a Bays-Durham shuffle algorithm. The basic idea is simply to add the two sequences, and modulo the modulus of either of them. The obtained generator has a cycle period $\approx 2.3 \times 10^{18}$.

3.2 Numerical integration by estimating expectation

Numerical integration for definite integrals, usually multidimensional ones, can be done by estimating the expectation using random numbers. The

usual algorithms evaluate the integrand at a regular grid. Monte Carlo methods, however, randomly choose the points at which the integrand is evaluated.

Let us consider the integration,

$$I(f) = \int_0^1 f(x)dx.$$

Monte Carlo integration. The MC method considers the integral as the expectation of some random variable. Based on the law of large numbers, we have

$$I(f) \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i) \doteq I_N(f),$$

where X_i are iid random variables.

Clearly, $I_N(f)$ as a random variable has the properties:

$$EI_N(f) = E\left[\frac{1}{N}\sum_{i=1}^{N} f(X_i)\right] = \frac{1}{N}\sum_{i=1}^{N}\int_0^1 f(x)dx = I(f).$$

$$Var[I_N(f)] = Var\left[\frac{1}{N}\sum_{i=1}^{N} f(X_i)\right] = \frac{1}{N^2}\sum_{i=1}^{N} Var[f(X_i)] = \frac{Var[f(X)]}{N}$$

Let $e_N = |I_N(f) - I(f)|$, then $E[e_N^2] = Var[I_N(f)]$. By Schwartz inequality ($(x, y) \leq ||x||||y||$),

$$E|e_N| \leq \sqrt{E|e_N|^2} = \sqrt{\frac{Var[f[X]]}{N}}.$$

Say, the half order convergence rate.

General distribution. For an integration of the following form:

$$\int f(x)p(x)dx,$$

where $p(x)$ is a probability density function and satisfies $\int p(x)dx = 1$ and $p(x \geq 0)$, then we have the approximation

$$I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(X_i)$$

where X_i is i.i.d. random variable with distribution density $p(x)$. The error of the Monte Carlo integration is bounded by

$$E[e_N^2] = Var[I_N(f)] = \frac{Var[f[X]]}{N},$$

i.e., it is determined by two values, the variance of $f(X)$ and the number of samples N. Later on, we will discuss methods of reducing the variance in order to improve the accuracy of the integration.

3.3 Random number of general distributions

3.3.1 Uniform spherical distribution

How to generate samples uniformly distributed in a sphere?

Suppose the radius is R. A simple idea is to generate a sample point in a box of side length $2R$ with uniform distribution, then accept the sample if it is within the sphere. This is the so-called *accept-reject algorithm.* Obviously, around $(8 - 4\pi/3)/8 \approx 47.6\%$ samples will be rejected.

The *Inverse transform method* is a better way for generating these random samples, which is based on the following Proposition.

Proposition. Let Y be with cumulative distribution function $F(y)$, i.e.,

$$P\{Y \leq y\} = F(y).$$

If $X \sim U[0, 1]$, then $Y = F^{-1}(X)$ is the required distribution.

Proof. As $X \sim U[0, 1]$ and $Y = F^{-1}(X)$, we have,

$$P\{Y \leq y\} = P\{F^{-1}(X) \leq y\} = P\{X \leq F(y)\} = F(y).$$

The proof is then completed.

Solution of spherical problem. For any smooth function f and the spherical domain Ω, we have,

$$\int\int\int f(\mathbf{r})u(\mathbf{r})dxdydz = \int_0^{2\pi}\int_0^{\pi}\int_0^{R} f(\mathbf{r})p(r,\theta,\phi)drd\theta d\phi$$

where u and p are two distribution functions in Cartesian and spherical coordinates,

$$u = \begin{cases} 3/4\pi R^3, & (x,y,z) \in \Omega, \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad p = 3r^2\sin\theta/4\pi R^3.$$

We rewrite p as the form,

$$p = \frac{r^2}{R^3/3}\cdot\frac{\sin\theta}{2}\cdot\frac{1}{2\pi},$$

then each term corresponds to a 1D probability density function defined on domains, $r \in [0,R]$, $\theta \in [0,\pi]$ and $\phi \in [0,2\pi]$, respectively. Their cumulative distribution functions are r^3/R^3, $(1-\cos\theta)/2$, and $\phi/2\pi$. Suppose u, v, w are three random numbers from $U[0,1]$, then we get a sample in spherical coordinates,

$$(r,\theta,\phi) = (\sqrt[3]{u}R, \arccos(1-2v), 2\pi w).$$

Example. The *exponential distribution* with the PDF $p(y) = \begin{cases} 0, & y \le 0 \\ \lambda e^{-\lambda y}, & y \ge 0 \end{cases}$, its distribution function, $F(y) = \int_0^y p(z)dz = 1 - e^{-\lambda y}$, and hence,

$$F^{-1}(x) = -\frac{1}{\lambda}\ln(1-x), \quad x \in (-,1)$$

From the transform method, the random variable of exponential distribution is given by

$$Y_i = -\frac{1}{\lambda}\ln(1-X_i), \quad i = 1,2,\cdots$$

where $X_i \sim U[0,1]$.

Example. The *normal distribution* with PDF $p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ which has distribution function,

$$F(x) = \int_{-\infty}^{x} p(y)dy = \frac{1}{2} + \frac{1}{2} erf\left(\frac{x}{\sqrt{2}}\right),$$

where

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

is the error function. So

$$F^{-1}(x) = \sqrt{2} erf^{-1}(2x - 1).$$

But erf^{-1} is hard to compute.

Box-Muller method. The most used method to generate the standard normal random variable is the Box-Muller method which is based on the 2D normal distribution (Y_1, Y_2). Let $(y_1, y_2) = (r\cos\theta, r\sin\theta)$, then

$$\frac{1}{2\pi} e^{-\frac{y_1^2+y_2^2}{2}} dy_1 dy_2 = \frac{1}{2\pi} e^{-\frac{r^2}{2}} r dr d\theta = \left(\frac{1}{2\pi} d\theta\right)\left(e^{-r^2/2} r dr\right)$$

We have $\theta \sim U[0, 2\pi]$ and the radial direction distribution function

$$F(r) = \int_0^r e^{-s^2/2} s ds = 1 - e^{-r^2/2},$$

and its inverse can be easily obtained. Below is the procedure of the algorithm:

1. Generate two random numbers $X_1, X_2 \sim U[0, 1]$,
2. Define $\begin{cases} Y_1 = \sqrt{-2\ln X_1}\cos(2\pi X_2) \\ Y_2 = \sqrt{-2\ln X_1}\sin(2\pi X_2) \end{cases}$
3. Take Y_1 and Y_2 as two independent draws from $N(0, 1)$.

Accept-Reject algorithms

The basic idea of the accept-reject methods is by considering one-dimensional probability density as the marginal density of higher-dimensional probability density, i.e., if random variable (X_1, X_2) has probability density, $p(x_1, x_2)$, then the probability density of X_1 is $\int_{-\infty}^{\infty} p(x_1, x_2)dx_2$.

Suppose we want to generate a random variable X with probability density $p(x)$. We assume $x \in [a, b]$, $\int_a^b p(x)dx = 1$ and $p(x) \leq d$, and define the set,

$$A \doteq \{(x, y)|x \in [a, b], y \in [0, p(x)]\}.$$

Let

$$(X, Y) \sim U(A),$$

which has the density $\chi_A(x, y)$. Then the marginal probability of the X component is,

$$\int_0^d \chi_A(x, y)dy = \int_0^{p(x)} 1dy = p(x).$$

Algorithm:

1. Generate $X_i \sim U[a, b]$
2. Generate $Y_i \sim U[0, d]$
3. Accept X_i if $0 \leq Y_i < p(X_i)$,
4. or reject it and go to 1.

This is a special case of the following algorithm.

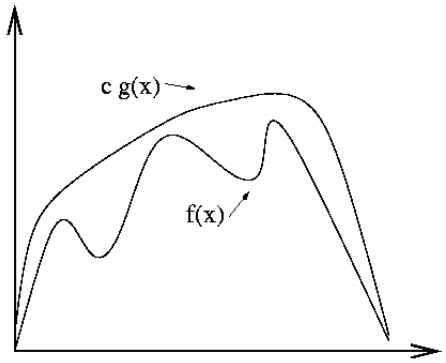


Figure 3.1: Accept-Reject sampling

If g is another density function and we know how to sample from $g(x)$, then suppose $e(x) = cg(x)$ for $c \geq 1$ be an envelope of $p(x)$, we have the reject sampling algorithm:

1. Generate $X \sim g$
2. Generate $Y \sim U[0,1]$
3. Accept X if $Y < p(X)/e(X)$,
4. or reject it and go to 1.

Proof. The distribution function,

$$P(X \leq x) = P\left(X \leq x | Y \leq \frac{p(X)}{e(X)}\right) = \frac{P\left(X \leq x, Y \leq \frac{p(X)}{e(X)}\right)}{P\left(Y \leq \frac{p(X)}{e(X)}\right)}$$

$$= \frac{\int_{-\infty}^{x} \int_{0}^{p(z)/e(z)} dug(z)dz}{\int_{-\infty}^{\infty} \int_{0}^{p(z)/e(z)} dug(z)dz} = \int_{-\infty}^{x} p(z)dz.$$

3.4 Importance sampling

Example. Suppose we are calculating the probability, p, that a Cauchy $C(0,1)$ variable is larger than 2,

$$p = \int_{2}^{\infty} \frac{1}{\pi(1+x^2)} dx,$$

when p is evaluated through the empirical average,

$$\hat{p}_1 = \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}_{X_j > 2}$$

of an iid sample $X_1, \cdots, X_N \sim C(0,1)$, the variance of this estimate is $p(1-p)/N = 0.127/N$ since $p = 0.15$. (The variance is evaluated as following: let $f(x)$ is the pdf of $C(0,1)$, then $E[(\mathbb{I}_{X_j>2})^2] = \int (\mathbb{I}_{x>2})^2 f(x)dx = \int_2^\infty f(x)dx = p$. Also, $E[(\mathbb{I}_{X_j>2})]^2 = [\int_2^\infty f(x)dx]^2 = p^2$. And thus the variance of each sample is $p - p^2$.)

This variance can be reduced by taking into account the symmetric nature of $C(0,1)$, since the average,

$$\hat{p}^2 = \frac{1}{2N} \sum_{j=1}^{N} \mathbb{I}_{|X_j|>2}$$

has variance $p(1-2p)/2N = 0.052/N$.

Again, if we do the following transform,

$$p = \frac{1}{2} - \int_0^2 \frac{1}{\pi(1+x^2)} dx$$

which is considered to be the expectation of $h(X) = 2/\pi(1+X^2)$, where $X \sim U[0,2]$. Then we have

$$\hat{p}_3 = \frac{1}{2} - \frac{1}{N}\sum_{j=1}^{N} h(U_j)$$

for $U_j \sim U[0,2]$. The variance of this method is $(E[h^2] - E[h]^2)/N = 0.0285/N$.

Moreover, from the definition of p, it can be rewritten as,

$$p = \int_0^{1/2} \frac{y^{-2}}{\pi(1+y^{-2})} dy = \int_0^{1/2} \frac{1}{\pi(1+y^2)} dy$$

which can be considered the expectation of $h(Y)/4$. We have another evaluation,

$$\hat{p}_4 = \frac{1}{4N}\sum_{j=1}^{N} h(Y_j)$$

where $Y_j \sim U[0, \frac{1}{2}]$. A simple calculation shows the variance is $0.95 \times 10^{-4}/N$.

The idea of the *importance sampling* is as follows. If we rewrite $I(f)$ as,

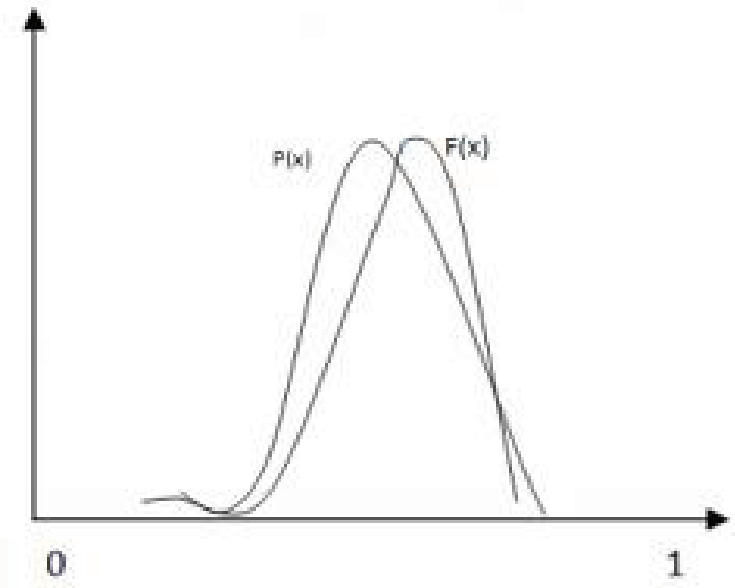
$$I(f) = \int f(x)dx = \int \frac{f(x)}{p(x)} p(x)dx,$$

where $p(x)$ is a pdf with $p(x) > 0$. Then we have an expression of approximation:

$$I(f) \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(Y_i)}{p(Y_i)}$$

where Y_i is an iid sample with $p(y)$ as the probability density.

The method is called importance sampling because it is based on so-called importance function $p(x)$ which reflects the important portion of the integrand, and although it would be more accurate to call it "weighted sampling".



Theoretically, we have variances,

$$Var_X(f) = \int [f - I(f)]^2 dx = \int f^2 dx - I^2(f),$$

$$Var_Y\left(\frac{f}{p}\right) = \int \left(\frac{f}{p}\right)^2 p dy - I^2(f) = \int \frac{f^2}{p} dy - I^2(f).$$

If we find a suitable $p(y)$ such that $\int \frac{f^2}{p} dy < \int f^2 dx$, then the variance will be reduced. In particular, if $p(x)$ happens to equal $f(x)/I(f)$, then $Var_Y\left(\frac{f}{p}\right) = 0$. But to have this $p(x)$, we have to know the exact value of $I(f)$, so it is impossible.

Example. Consider $\Omega = [-1, 1] \times [-1, 1]$ and to compute the integral $\mu = \int\int_\Omega f(x, y) dx dy$, with

$$f(x, y) = 0.5e^{-90(x-0.5)^2 - 45(y+0.1)^4} + e^{-45(x+0.4)^2 - 60(y-0.5)^2}$$

we have $\mu = 0.1258$.

If we take samples $(X_1, Y_1), \cdots, (X_N, Y_N) \sim U(\Omega)$, we can compute the integral by the original MC. When $N = 2500$, a typical computation yields $\hat{\mu} = 0.1307$.

If we use the importance sampling with

$$p(x, y) \propto 0.5e^{-90(x-0.5)^2-10(y+0.1)^2} + e^{-45(x+0.4)^2-60(y-0.5)^2}$$

for $(x, y) \in \Omega$, which is a truncated mixed Gaussian distribution

$$0.46N\left[\begin{pmatrix} 0.5 \\ -0.1 \end{pmatrix}, \begin{pmatrix} \frac{1}{180} & 0 \\ 0 & \frac{1}{20} \end{pmatrix}\right] + 0.54N\left[\begin{pmatrix} -0.4 \\ 0.5 \end{pmatrix}, \begin{pmatrix} \frac{1}{90} & 0 \\ 0 & \frac{1}{120} \end{pmatrix}\right]$$

We can do the following sampling: (1) Flip a nonuniform coin with probability 0.46 for Head, (2) If Head, sample a random variable from the first Gaussian distribution, or from the second one. Compute $\omega_i = \frac{f(X_i,Y_i)}{p(X_i,Y_i)}$ (if the sample locates outside Ω, let $\omega_i = 0$). (3) Evaluate the mean value. When $N = 2500$, the estimator is 0.1259.

3.5 Project 2. Random Number Generators and MC Integration

We will implement the MSG

$$X_{n+1} = aX_n \mod m$$

by the Schrage's algorithm. The idea is based on an approximate factorization of m,

$$m = aq + r, \quad i.e., \ \ q = [m/a], \ r = (m \bmod a).$$

For any z, there is a factorization $z = q[z/q] + (z \mod q)$, then

$$az \mod m = \{a(q[z/q] + (z \mod q)) + r[z/q] - r[z/q]\} \mod m$$

that is,

$$az \mod m = \begin{cases} a(z \mod q) - r[z/q] & \text{if it is} \geq 0, \\ a(z \mod q) - r[z/q] + m & \text{otherwise.} \end{cases}$$

Numerical experiments and output your results:

1. Implement the MSG by the Schrage's algorithm for the parameters of either $m_1 = 2147483563$ and $a_1 = 40014$, or $m_2 = 2147483399$ and $a_2 = 40692$. Plot the histogram of a sequence of random numbers of $U[0, 1]$, $X = (X_1, \cdots, X_N)$ using "hist" command. Also use "plot(X,'.')" to see if they are randomly distributed.

2. Implement the Bays-Durham shuffle algorithm by combining two generators of the two sets of parameters in previous item.

3. Let $\Omega = [-1, 1] \times [-1, 1]$. Compute the integral $\mu = \int_\Omega f(x, y)dxdy$, with

$$f(x, y) = 0.5e^{-90(x-0.5)^2-45(y+0.1)^4} + e^{-45(x+0.4)^2-60(y-0.5)^2}.$$

Compare the result of the MC integration with the accurate solution $\mu = 0.125844$ with the increase number of sample pairs (X_i, Y_i), $i = 1, \cdots, N$, and analyze the results.

4. Discuss the results you obtained.

Importance sampling. If we use the importance sampling with $p(x, y)$ being a truncated mixed Gaussian distribution

$$0.46N\left[\begin{pmatrix}0.5\\-0.1\end{pmatrix},\begin{pmatrix}\frac{1}{180}&0\\0&\frac{1}{20}\end{pmatrix}\right]+0.54N\left[\begin{pmatrix}-0.4\\0.5\end{pmatrix},\begin{pmatrix}\frac{1}{90}&0\\0&\frac{1}{120}\end{pmatrix}\right]$$

We can do the following sampling:
(1) Flip a nonuniform coin with probability 0.46 for Head,
(2) If Head, sample a random variable from the first Gaussian distribution, or from the second one. Compute $\omega_i = \frac{f(X_i,Y_i)}{p(X_i,Y_i)}$ (if the sample locates outside Ω, let $\omega_i = 0$).
(3) Evaluate the mean value.

Numerical experiments and output your results:

1. Implement the MC by sampling from uniform distribution $U([-1, 1] \times [-1, 1])$. Plot a figure to show the error with the increase of N, where $N = 500, 1000, \cdots, 10000$.

2. Implement the importance sampling MC by sampling from the mixed Gaussian distribution. Plot a figure to show the error with the increase of N, where $N = 500, 1000, \cdots, 10000$. (Hint: you can use two one-dimensional Gaussian distributions to obtain the two-dimensional distribution)

3. Discuss the results you obtained.

Chapter 4

Numerical Differentiation and Integration

4.1 Numerical differentiation

Consider the Taylor expansion of a function $f(x)$ around x_j:

$$f(x_j + h) = f_j + f'(x_j)h + \frac{f''(x_j)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_j)}{n!}h^n + \cdots,$$

and similarly,

$$f(x_j - h) = f_j - f'(x_j)h + \frac{f''(x_j)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_j)}{n!}(-h)^n + \cdots.$$

Approximation of the first order derivative. By the method of undetermined coefficients, the first-order derivative around x_j can be approximated by the linear combination of values of nearby knots, yielding the following three difference formulas:

$$\begin{aligned} f'(x_j) &= \frac{f_{j+1} - f_j}{h} + O(h), \\ f'(x_j) &= \frac{f_j - f_{j-1}}{h} + O(h), \\ f'(x_j) &= \frac{f_{j+1} - f_{j-1}}{2h} + O(h^2), \end{aligned}$$

called the *forward, backward, and central differences.*

The formulas are obtained by truncating the Taylor series by considering h is a small quantity. The **truncation error** can be easily computed by using the Taylor expansion. For the central difference, it is

$$\frac{f_{j+1} - f_{j-1}}{2h} = f'(x_j) + \frac{f'''(x_j)}{3!}h^2 + \frac{f^{(5)}(x_j)}{5!}h^4 + \cdots .$$

Also, as we have the difference scheme,

$$\frac{f_{j+2} - f_{j-2}}{4h} = f'(x_j) + 4\frac{f'''(x_j)}{3!}h^2 + O(h^5).$$

If we cancel the $f'''(x_j)$ terms, we obtain (five-point formula),

$$f'(x_j) = \frac{f_{j-2} - 8f_{j-1} + f_{j+1} - f_{j+2}}{12h} + O(h^4).$$

Approximation of the second-order derivative

Three-point formula:

$$f''(x_j) = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} + O(h^2);$$

five-point formula:

$$f''(x_j) = \frac{-f_{j-2} + 16f_{j-1} - 30f_i + 16f_{j+1} - f_{j+2}}{12h^2} + O(h^4).$$

Nonuniform data points

If the data points are nonuniform, then by using the Taylor expansion formulas again, we have the following difference approximations:

$$f'_j = \frac{h_{j-1}^2 f_{j+1} + (h_j^2 - h_{j-1}^2)f_j - h_j^2 f_{j-1}}{h_j h_{j-1}(h_j + h_{j-1})} + O(h^2), \quad h = \max(h_j, h_{j-1}),$$
$$f''_j = \frac{2[h_{j-1}f_{j+1} - (h_j + h_{j-1})f_j + h_j f_{j-1}]}{h_j h_{j-1}(h_j + h_{j-1})} + O(h).$$

It should be noted that the second-order difference has the accuracy **an order lower** than the case of uniform data points.

4.2 Numerical integration: Newton-Cotes formulas

The construction of numerical integration is also based on the polynomial interpolation. The numerical integration is to discretize the integral into the form of

$$I = \int_a^b f(x)dx \approx \sum c_k f_k$$

for some notes $\{x_k\}$, and coefficients $\{c_k\}$. We will discuss how to determine the coefficients by the trapezoidal rule, Simpson's method, and Gauss quadrature.

Let us first recall how the interpolation function is used to construct difference schemes. For given three points x_{j-1}, x_j and x_{j+1} with equally spaced grid size h, the Lagrange interpolation is as follows,

$$P_2(x) = f_{j-1}\frac{(x-x_j)(x-x_{j+1})}{2h^2} + f_j\frac{(x-x_{j-1})(x-x_{j+1})}{-h^2} + f_{j+1}\frac{(x-x_{j-1})(x-x_j)}{2h^2}.$$

And obviously,

$$P_2'(x_j) = f_{j-1}\frac{-h}{2h^2} + f_j\frac{h-h}{-h^2} + f_{j+1}\frac{h}{2h^2} = \frac{f_{j+1}-f_{j-1}}{2h},$$

see the central difference scheme. Also, for the forward difference,

$$P_1(x) = f_j\frac{(x-x_{j+1})}{-h} + f_{j+1}\frac{(x-x_j)}{h},$$

and thus

$$P_1'(x_j) = \frac{f_{j+1}-f_j}{h}.$$

Trapezoidal rule

We first consider the integral $\int_{x_j}^{x_{j+1}} f(x)dx$, with $x_{j+1} - x_j = h$. By approximating the function f with the linear interpolation of two end points, we have,

$$P_1(x) = f_j\frac{(x-x_{j+1})}{-h} + f_{j+1}\frac{(x-x_j)}{h}.$$

We then approximate the integral by the integral of P_1:

$$\int_{x_j}^{x_{j+1}} P_1(x)dx = h\left(\frac{1}{2}f_j + \frac{1}{2}f_{j+1}\right).$$

For the integral defined on interval (a, b), we will use the composite trapezoidal rule. It divides the interval into many subintervals, and applies the trapezoidal rule to each subinterval. Let the nodes be $x_j = a + jh$ for $j = 0, 1, \cdots, N$, and $h = (b - a)/N$. Then,

$$\int_a^b f(x)dx = \sum_{j=0}^{N}\int_{x_j}^{x_{j+1}} f(x)dx \approx \sum_{j=0}^{N} h\left(\frac{1}{2}f_j + \frac{1}{2}f_{j+1}\right).$$

By a simplification, we obtain the composite trapezoidal rule,

$$\int_a^b f(x)dx \approx I_N = h\left[\frac{1}{2}(f_0 + f_N) + \sum_{j=1}^{N} f_j\right].$$

Error of the approximation. To determine the number of divisions. For the elementary formula on (x_j, x_{j+1}), we have a number ξ in this interval, such that,

$$\int_{x_j}^{x_{j+1}} [f(x) - P_1(x)]dx = \int_{x_j}^{x_{j+1}} \frac{f''(\xi)}{2!}(x - x_j)(x - x_{j+1})dx = -\frac{h^3}{12}f''(\xi).$$

Then for the composite trapezoidal rule, we have the following theorem.

Theorem. If f has continuous second order derivative on $[a, b]$, then there is an $\xi \in (a, b)$, such that

$$\int_a^b f(x)dx - I_N = -\frac{h^2(b - a)}{12}f''(\xi).$$

From this Theorem, if we know some bounds for $|f''(x) \leq C$ and expect the error less than a small number ε, we will have

$$\frac{(b - a)^3 C}{12N^2} \leq 10^{-8}.$$

Simpson's method

It is natural to use higher-order polynomials to approximate the integration. The Simpson's method is derived from a quadratic polynomial interpolating three points. The idea is the same.

Consider the interval (x_j, x_{j+2}), with grid size h. Then we have approximation,

$$\int_{x_j}^{x_{j+2}} f(x)dx \approx \int_{x_j}^{x_{j+2}} P_2(x)dx,$$

where

$$P_2(x) = f_j \frac{(x-x_{j+1})(x-x_{j+2})}{2h^2} + f_{j+1}\frac{(x-x_j)(x-x_{j+2})}{-h^2} + f_{j+2}\frac{(x-x_j)(x-x_{j+1})}{2h^2}.$$

This gives rise to the following elementary Simpson's rule:

$$\int_{x_j}^{x_{j+2}} f(x)dx \approx h\left[\frac{1}{3}f_j + \frac{4}{3}f_{j+1} + \frac{1}{3}f_{j+2}\right].$$

Thus we have the composite Simpson's rule:

$$\int_a^b f(x)dx \approx h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \cdots + \frac{2}{3}f_{N-2} + \frac{4}{3}f_{N-1} + \frac{1}{3}f_N\right],$$

for an **even** n.

Error estimate. $\int_{x_j}^{x_{j+2}}[f(x) - P_2(x)]dx \approx -\frac{h^5}{90}f^{(4)}(\xi)$.

We have the following theorem:

Theorem. If f has a continuous fourth order derivative on (a, b), then $\exists \xi \in (a, b)$, such that,

$$\int_a^b f(x)dx - h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \cdots + \frac{2}{3}f_{N-2} + \frac{4}{3}f_{N-1} + \frac{1}{3}f_N\right] = -\frac{(b-a)^5}{180N^4}f^{(4)}(\xi),$$

where n is even.

Note. We gain one more order of degree for the numerical integration due to the symmetry.

What happens if we have an odd number of subintervals, say, N odd? One solution is to isolate the last slice and we then have

$$\int_{x_{N-1}}^{x_N} P_2(x)dx = \frac{h}{12}(-f_{N-2} + 8f_{N-1} + 5f_N).$$

Any problem of this approximation?

4.3 Euler-Maclaurin formula

Definition. The Bernoulli polynomials $B_n(x)$ are recursively defined by: $B_0(x) = 1$, $B_n' = B_{n-1}$ with the normalization condition $\int_0^1 B_n(x)dx = 0,\ n = 1, 2, \cdots$. The rational number $b_n = n!B_n(0)$ $(n = 0, 1, \cdots)$ is called the Bernoulli number.

Lemma. The Bernoulli polynomials have the symmetry property:

$$B_n(x) = (-1)^n B_n(1-x),\ \ x \in \mathbb{R},\ n = 0, 1, \cdots$$

Proof. Obviously, it holds for $n = 0$. Assume it holds for some $n \geq 0$, then integrating both sides, we have,

$$B_{n+1}(x) = (-1)^{n+1} B_{n+1}(1-x) + \beta_{n+1}$$

for some constant β_{n+1}. The normalization condition implies the constant is zero, and the conclusion is proven.

From the Proof, we have $B_{2m+1}(0) = -B_{2m+1}(1)$, and thus $B_{2m+1}(0) = B_{2m+1}(1) = 0$.

Definition. The periodic extension of the Bernoulli polynomials, $\tilde{B}_n(x)$; i.e., $\tilde{B}_n(x) = B_n(x)$, for $0 \leq x \leq 1$ and $\tilde{B}_{n+1}(x) = B_n(x)$.

Theorem. Let $f \in C^m[a, b], m \geq 2$, and $T_h(f) = \frac{1}{2}\sum_{k=0}^{n-1}[f(x_k) + f(x_{k+1})]$. Then we have,

$$\int_a^b f(x)dx = T_n(f) - \sum_{j=1}^{[m/2]} \frac{b_{2j}h^{2j}}{(2j)!}[f^{(2j-1)}(b) - f^{(2j-1)}(a)] + (-h)^m \int_a^b \tilde{B}_m(\frac{x-a}{h}) f^{(m)}(x)dx$$

where $[m/2]$ denotes the largest integer smaller than or equal to $m/2$.

Proof. Let $g \in C^m[0,1]$. Then by $m-1$ partial integrations and using $B_n(0) = (-1)^n B_n(1)$ $(n = 2, 3, \cdots)$, we find that,

$$\begin{aligned}\int_0^1 B_1(z)g'(z)dz = & \sum_{j=2}^{m}(-1)^j B_j(0)[g^{(j-1)}(1) - g^{(j-1)}(0)] \\ & - (-1)^m \int_0^1 B_m(z)g^{(m)}(z)dz\end{aligned}$$

Combining this with the partial integration,

$$\int_0^1 B_1(z)g'(z)dz = \frac{1}{2}[g(1) + g(0)] - \int_0^1 g(z)dz$$

and observing $B_{2m+1}(0) = 0, m = 1, 2, \cdots$, leads us to,

$$\begin{aligned}\int_0^1 g(z)dz = & \frac{1}{2}[g(1) + g(0)] - \sum_{j=1}^{[m/2]} \frac{b_{2j}}{(2j)!}[g^{(2j-1)}(1) - g^{(2j-1)}(0)] \\ & +(-1)^m \int_0^1 B_m(z)g^{(m)}(z)dz\end{aligned}$$

Now we substitute $x = x_k + hz$ and $g(z) = f(x_k + hz)$ to obtain,

$$\begin{aligned}\int_{x_k}^{x_{k+1}} f(x)dx = & \frac{h}{2}[f(x_k) + f(x_{k+1})] - \sum_{j=1}^{[m/2]} \frac{b_{2j}h^{2j}}{(2j)!}[f^{(2j-1)}(x_{k+1}) - f^{(2j-1)}(x_k)] \\ & +(-h)^m \int_{x_k}^{x_{k+1}} B_m(\frac{x-a}{h})f^{(m)}(x)dx\end{aligned}$$

Finally, we sum this equation for $k = 0, \cdots, n-1$ to arrive at the Euler-Maclaurin expansion.

The Euler-Maclaurin formula illustrates that, if $f(x)$ is periodic, the trapezoidal rule is spectrally convergent.

4.4 Romberg integration

By the *Euler-Maclaurin expansion*, we have the expression,

$$\int_a^b f(x)dx = T_1(h) + \sum_{k=1}^{\infty} c_{2k}^{(1)} h^{2k},$$

where $T_1(h)$ is the composite Trapezoidal formula of space step-size h, and $c_{2k}^{(1)}$ are constants. Then by Richardson extrapolation, we have,

$$\int_a^b f(x)dx = T_2(h) + \sum_{k=1}^{\infty} c_{2k}^{(2)} h^{2k},$$

where

$$T_2(h) = \frac{T_1(h/2) - 4^{-1}T_1(h)}{1 - 4^{-1}}.$$

The Richardson extrapolation is recursively used, we get the *Romberg quadratures*:

$$T_{k+1}(h) = \frac{T_k(h/2) - 4^{-k}T_k(h)}{1 - 4^{-k}}, \;\; k = 1, 2, \cdots .$$

4.5 Gauss-Legendre quadrature

We have studied the trapezoidal, Simpson and other Newton-Cotes rules, which can be obtained by the so-called Romberg quadrature. The basic idea of constructing such integration rule is by using the Lagrange polynomials for *equidistant* nodes. The obtained quadrature rule with $n+1$ nodes is exact for the integrand of $1, x, \cdots, x^n$. We call the quadrature is of algebraic accuracy of order n ($n+1$ for even n). For example, the trapezoidal rule ($n = 1$) is with two nodes, so it has the algebraic accuracy of order 1. While the Simpson rule ($n = 2$) is with three nodes, and its algebraic accuracy is order 3.

If we choose the nodes in a different way, we can get higher algebraic accuracy with fixed number of nodes. Suppose we have the quadrature,

$$\int_a^b \rho(x)f(x)dx = \sum_{k=1}^{n} A_k f(x_k).$$

The optimal choice of the nodes is determined by solving the following algebraic equations using the basis functions:

$$\sum_{k=1}^{n} A_k x_k^j = \int_a^b \rho(x)x^j dx, \;\; j = 0, 1, \cdots, 2n-1,$$

where the unknowns are A_k and x_k for $k = 1, \cdots, n$. If this system is solvable, we can obtain the algebraic accuracy of at least order $2n-1$!

Example 1. For example, suppose $[a, b] = [-1, 1]$, when $n = 1$ and $\rho(x) = 1$, we have $A_1 = 2$ and $A_1 x_1 = 0$. This is the mid-point rule.

Example 2. When $n = 2$, we have

$$\begin{aligned} &A_1 + A_2 = 2, \\ &A_1 x_1 + A_2 x_2 = 0, \\ &A_1 x_1^2 + A_2 x_2^2 = 2/3, \\ &A_1 x_1^3 + A_2 x_2^3 = 0. \end{aligned}$$

The solution of this system is $A_1 = A_2 = 1$, $x_1 = -1/\sqrt{3}$ and $x_2 = 1/\sqrt{3}$. It can be verified that the equation is not exact for x^4, and so the order of algebraic accuracy is 3.

Legendre polynomials:

Let us first define Legendre polynomials:

$$L_0 = 1, L_1 = x, \;\text{ and }\; (n+1)L_{n+1} = (2n+1)xL_1 - nL_{n-1}, \;\; n = 2, 3, \cdots$$

which satisfy $L_n(1) = 1$. Here are some of them:

$$\begin{aligned} L_2 &= \frac{3x^2}{2} - \frac{1}{2}, \\ L_3 &= \frac{5x^3}{2} - \frac{3x}{2}, \\ L_4 &= \frac{35x^4}{8} - \frac{15x^2}{4} + \frac{3}{8}, \\ L_5 &= \frac{63x^5}{8} - \frac{35x^3}{4} + \frac{15x}{8}. \end{aligned}$$

The Legendre polynomials are orthogonal,

$$\int_{-1}^{1} L_i(x)L_j(x)dx = 0, \quad \text{when } \ i \neq j.$$

The N-point Gauss-Legendre quadrature formula is given by,

$$\int_{-1}^{1} f(x)dx \approx \sum_{j=0}^{N-1} c_j f_j,$$

where $x_0, \cdots, x_{N-1}$ are zeros of the Legendre polynomials of degree N, and c_j is given by,

$$c_j = \int_{-1}^{1} \prod_{l=0,\cdots,N-1}^{l\neq j} \frac{x - x_l}{x_j - x_l} dx.$$

For $N = 1$, we have

$$\int_{-1}^{1} f(x)dx \approx 2f(0).$$

For $N = 2$, we have

$$\int_{-1}^{1} f(x)dx \approx f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right).$$

For $N = 3$, we have

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9} f\left(-\sqrt{0.6}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{0.6}\right).$$

For the integral on $[a, b]$ instead of $[-1, 1]$, the Gauss quadrature can be achieved by a transformation:

$$t = a + \frac{b-a}{2}(x+1).$$

The convergence order. Let us compare the following three-point formulas:

Composite trapezoidal rule: $\int_{-1}^{1} f(x)dx \approx \frac{1}{2}f(-1) + f(0) + \frac{1}{2}f(1)$,

Simpson's rule: $\int_{-1}^{1} f(x)dx \approx \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1)$,

and Gauss quadrature: $\int_{-1}^{1} f(x)dx \approx \frac{5}{9}f\left(-\sqrt{0.6}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{0.6}\right)$.

The composite trapezoidal rule is exact for any polynomial of degree less than 1. The Simpson's rule is exact for any polynomial of degree less than 3. The 3-point Gauss quadrature is exact for any polynomial of degree less than 5! We can see the Gauss quadrature is the most accurate method, as these three methods require the same amount of computation. In general, we have the following theorem:

Theorem. The N-point Gauss-Legendre formula is exact when $f(x)$ is any polynomial of degree less then or equal to $2N-1$.

The sketch of the proof is as follows.

1. For any polynomial $f(x)$ with degree $2N-1$, we can rewrite it as,

$$f(x) = h(x)L_N(x) + r(x),$$

where $h(x)$ and $r(x)$ are polynomials with degree less than or equal to $N-1$.

2. Let $P_{N-1}(x)$ interpolates (x_j, f_j) for $j = 0, \cdots, N-1$, then $P_{N-1}(x) = r(x)$.

3. If we integrate the equation $f(x)$, we have,

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} h(x)L_N(x)dx + \int_{-1}^{1} r(x)dx = \int_{-1}^{1} P_{N-1}(x)dx.$$

Thus,

$$\int_{-1}^{1} f(x)dx = \sum_{j=0}^{N-1} c_j f(x_j).$$

4.6 Singular integration

Transformation method:

Let us consider the singular integration, $\int_0^1 e^x/\sqrt{x}dx$ which has a singular point at 0. We can use the transformation $u = \sqrt{x}$ to become a Riemann integral.

Another example is $\int_0^1 x^{999}e^x dx$ which is also hard to calculate, although there is no singularity. We can use the transformation $u = x^{1000}$.

Elimination of singularity

A general technique of solving

$$\int_a^b f(x)dx = \int_a^b \frac{g(x)}{(x-a)^p}dx$$

where $g(x)$ is smooth is as follows. The integral converges if and only if $0 < p < 1$.

We can construct the Taylor expansion of $g(x)$,

$$P_4(x) = g(a) + g'(a)(x-a) + \cdots + \frac{g^(4)(a)}{4!}(x-a)^4,$$

and then we can write

$$\int_a^b f(x)dx = \int_a^b \frac{g(x) - P_4(x)}{(x-a)^p}dx + \int_a^b \frac{P_4(x)}{(x-a)^p}dx$$

The second term can be exactly determined. and We can approximated the first term with the Simpson rule.

4.7 Project 3. Alzheimer's disease

Example 5.1 in Computational Statistics by G. H. Givens and J. A. Hoeting.

Calculate the integral Eq. (5.7) by the composite Trapezoidal rule and the composite Gauss quadrature (3-point formula). Analyze and discuss your results.

Chapter 5

Interpolation, Approximation, Regression and Matrix Factorization

Weierstrass approximation theorem. Suppose that f is defined and continuous on $[a, b]$. For each $\varepsilon > 0$, there exists a polynomial $P(x)$ such that

$$|f(x) - P(x)| \leq \varepsilon, \quad \text{for all } x \in [a, b].$$

The proof can be found in most texts on real analysis.

5.1 Polynomial interpolation

Given $(x_j, y_j), j = 0, 1, \cdots, n$ and $y_j = f(x_j)$, find a polynomial $P_n(x)$, with degree $\leq n$, such that $P_n(x_j) = y_j$. $P_n(x)$ is called the **interpolation polynomial** of function these data set.

Uniqueness: Suppose $P_n(x) = a_0 + a_1 x + \cdots + a_n x^n \in \mathbb{P}_n$ where $\mathbb{P}_n$ is the **linear space** composed of real polynomial with order less than or equal to n, then,

$$\begin{cases} a_0 + a_1 x_0 + \cdots + a_n x_0^n = y_0, \\ a_0 + a_1 x_1 + \cdots + a_n x_1^n = y_1, \\ \qquad \cdots, \\ a_0 + a_1 x_n + \cdots + a_n x_n^n = y_n. \end{cases}$$

The coefficient matrix

$$A = \begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ & & \cdots & \\ 1 & x_n & \cdots & x_n^n \end{pmatrix},$$

is a Vandermonde matrix, i.e., we have

$$\det A = \prod_{0 \le i < j \le n} (x_j - x_i) \neq 0.$$

So the solution of the linear system is unique.

Lagrange interpolation

Lagrange interpolation is an explicit formula of polynomial $P_n(x)$ with a set of different basis functions.

For $n = 1$, given (x_0, y_0) and (x_1, y_1), then the straight line through these two points can be written as,

$$P_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0}.$$

For $n = 2$, one more point (x_2, y_2), then

$$P_2(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}.$$

(plot a figure for the Lagrange basis.)

In general,

$$P_n(x) = \sum_{j=0}^{n} y_j \prod_{k=0, k \neq j}^{n} \frac{(x - x_k)}{(x_j - x_k)}.$$

Basis functions $l_j(x) = \prod_{k=0, k \neq j}^{n} \frac{(x - x_k)}{(x_j - x_k)}$ satisfies,

$$\begin{cases} l_j(x_j) = 1, \\ l_j(x_k) = 0, \quad k \neq j. \end{cases}$$

Thus,

$$P_n(x_j) = y_0 l_0(x_j) + y_1 l_1(x_j) + \cdots + y_n l_n(x_j) = y_j.$$

Error estimate. Let f be a function of x with continuous $(n+1)$-th derivatives. $P_n(x)$ is the n-th order interpolation function of knots $(x_j, f(x_j)), j = 0, \cdots, n$. Then, for each x, there exists $\xi \in [x_0, x_n]$, such that

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^{n} (x - x_j).$$

Proof. As $R_n(x_j) = 0$ for $j = 0, \cdots, n$, then the remainder function can be written as,

$$R_n(x) = K(x)(x - x_0) \cdots (x - x_n).$$

Let us fix x so that $x \neq x_j$ for any j, then consider a new function of t,

$$\phi(t) = f(t) - P_n(t) - K(x)(t - x_0) \cdots (t - x_n).$$

We can easily see:

$$\phi(x) = 0, \quad \text{and}, \quad \phi(x_j) = 0, j = 0, \cdots, n,$$

i.e., function $\phi(t)$ has $n + 2$ distinct zeros. From Rolle's theorem, if a function has two zeros, there must be a zero for its derivative between the two zeros. If a function has three distinct zeros, then its first derivative has two zeros, and its second derivative has one zero. In general, since $\phi(t)$ has $(n + 2)$ zeros, its $(n + 1)$-th derivative has at least one zero, i.e., there exists an $\xi \in [x_0, x_n]$ such that $\phi^{(n+1)}(\xi) = 0$. Then a simple calculation yields,

$$\phi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - K(x)(n + 1)! = 0.$$

So $K(x) = f^{(n+1)}(\xi)/(n + 1)!$ which leads to the theorem.

Divided differences and Newton interpolation

Suppose $P_n(x)$ is the interpolation polynomial. The divided differences of f are used to express $P_n(x)$ in the form

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0) \cdots (x - x_{n-1}).$$

When $n = 1$, the linear interpolation can be rewritten as,

$$P_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0);$$

When $n = 2$, it is,

$$P_2(x) = P_1(x) + a_2(x - x_0)(x - x_1)$$

where

$$a_2 = \frac{\frac{f(x_2)-f(x_0)}{x_2-x_0} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_1}.$$

Definition: Zeroth divided difference: $f[x_i] = f(x_i)$

First divided difference: $f[x_i, x_{i+1}] = \frac{f[x_{i+1}]-f[x_i]}{x_{i+1}-x_i}$;

Second divided difference: $f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1},x_{i+2}]-f[x_i,x_{i+1}]}{x_{i+2}-x_i}$;

In general, k-th divided difference: $f[x_i, x_{i+1}, \cdots, x_{i+k}] = \frac{f[x_{i+1},\cdots,x_{i+k-1},x_{i+k}]-f[x_i,\cdots,x_{i+k-2},x_{i+}}{x_{i+k}-x_i}$
Then we have theorem:

Theorem. Let a_k be the k-th divided difference $a_k = f[x_0, x_1, \cdots, x_k]$, and $a_0 = f(x_0)$, then,

$$P_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}).$$

Hemite interpolation

Suppose that we are given $n + 1$ distinct points $x_0, \cdots, x_n \in [a, b]$ and nonnegative $m_0, \cdots, m_n$ with $m = \max\{m_0, \cdots, m_n\}$. The **osculating interpolation polynomial** approximating a function $f \in C^m[a, b]$ at x_i is the polynomial of least degree with the property that it agrees with the function and all its derivatives of order less than or equal to m_i at x_i. The degree of this osculating polynomial is at most,

$$M = \sum_{i=0}^{n} m_i + n$$

because the number of conditions to be satisfied is $M + 1$.

Definition. The osculating polynomial approximating f is the polynomial $P(x)$ such that

$$\frac{d^k P(x_i)}{dx^k} = \frac{d^k f(x_i)}{dx^k}$$

for each $i = 0, \cdots, n$ and $k = 0, 1, \cdots, m_i$.

The case when $m_i = 1$ for each i gives the **Hemite polynomials.**

Runge's phenomenon.

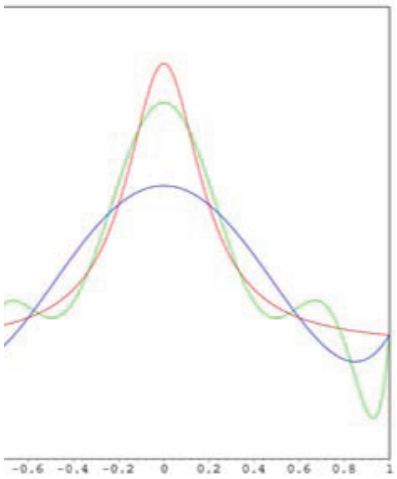


Figure 5.1: Runge's phenomenon.

A trivial choice of uniformly spaced nodes could lead to divergence. For example,

$$R(x) = \frac{1}{1 + 25x^2}, \qquad x \in [-1, 1].$$

This is because $\prod_{j=0}^{n}(x - x_j)$ is very large near two ends and thus very sensitive to the higher order derivatives, which is the famous Runge phenomenon found by C. Runge 1901.

A direct calculation of the first derivatives at the end $x = 1$ is

$$|f'(1)| \approx 0.075, \quad |f''(1)| \approx 0.211.$$

The magnitude of higher order derivatives of the Runge function get even larger.

Weierstrass theorem tells us there must be a polynomial approximate to $f(x)$. Does it conflict with the Runge phenomenon? One way to resolve it is to use the zeros of the Chebyshev polynomial as the nodes.

To resolve this problem, one could take zeros of Chebyshev polynomials as the interpolation nodes.

Besides the Runge phenomenon, high order interpolation polynomials are numerically unstable. It is natural that in order to compute $P_n(x) = (x - a)^n$ for a large n, a small perturbation in x will lead to a large error in $P_n(x)$.

5.2 Piecewise polynomial interpolation

Piecewise linear and Hermite polynomials

Piecewise linear polynomial. Linear polynomial in each cell $[x_j, x_{j+1}]$. Truncation error is $O(h^2)$, but only C^0 continuity.

Piecewise Hermite interpolation. The interpolating function are given knot values and derivative values. For the cubic Hemite interpolation, suppose $y_j = f(x_j), m_j = f'(x_j)$, then the constructed polynomial $H_3(x)$ satisfies

$$H_3(x_j) = y_j, \;\; H_3(x_{j+1}) = y_{j+1}, \;\; H_3'(x_j) = m_j, \;\; H_3'(x_{j+1}) = m_{j+1}.$$

How to construct the basis functions similar to the Lagrange interpolation?

$$H_3(x) = \alpha_j(x)y_j + \alpha_{j+1}(x)y_{j+1} + \beta_j(x)m_j + \beta_{j+1}(x)m_{j+1}$$

where the basis functions satisfy the properties that

$$\alpha_j(x_j) = 1, \;\; \alpha_{j+1}(x_{j+1}) = 1, \;\; \beta_j'(x_j) = 1, \;\; \beta_{j+1}'(x_{j+1}) = 1,$$

and zero at the knots and the 1st derivatives otherwise. Let's look $\alpha_j(x)$ which can be defined by,

$$\alpha_j(x) = (ax + b)\left(\frac{x - x_{j+1}}{x_j - x_{j+1}}\right)^2$$

which implies $\alpha_j(x_{j+1}) = \alpha_j'(x_{j+1}) = 0$. Now using $\alpha_j(x_j) = 1$ and $\alpha_j'(x_j) = 0$ yields,

$$\alpha_j(x) = \left(1 + 2\frac{x - x_j}{x_{j+1} - x_j}\right)\left(\frac{x - x_{j+1}}{x_j - x_{j+1}}\right)^2.$$

Similarly,

$$\alpha_{j+1}(x) = \left(1 + 2\frac{x - x_{j+1}}{x_j - x_{j+1}}\right)\left(\frac{x - x_j}{x_{j+1} - x_j}\right)^2,$$

$$\beta_j(x) = (x - x_j)\left(\frac{x - x_{j+1}}{x_j - x_{j+1}}\right)^2,$$

$$\beta_{j+1}(x) = (x - x_{j+1})\left(\frac{x - x_j}{x_{j+1} - x_j}\right)^2.$$

Now in the piecewise Hermite approximation, the coefficients can be determined.

Cubic spline interpolation

The spline polynomial is a piecewise polynomial to interpolate the data set $\{(x_j, f_j), j = 0, 1, \cdots, n\}$, which may gain both the accuracy of high order polynomials and the stability of low order polynomial (without Runge's phenomenon).

Let $S(x) \in C^2[a, b]$ with $a = x_0$ and $b = x_n$. If $S_j(x) = S(x)$ in each interval $[x_j, x_{j+1}]$ for $j = 0, \cdots, n-1$ is a cubic polynomial, then $S(x)$ is called the **cubic spline function**.

Clearly, there are four coefficients to be determined in each interval of n intervals, so totally $4n$ unknowns to be determined. The continuity conditions

$$S_j(x_{j+1}) = S_{j+1}(x_{j+1}),\ S_j'(x_{j+1}) = S_{j+1}'(x_{j+1}),\ S_j''(x_{j+1}) = S_{j+1}''(x_{j+1}),$$

for $j = 0, \cdots, n-2$, provide $3(n-1)$ equations. The interpolation conditions $S(x_j) = f_j$ give $n+1$ equations. Two addition equations are required

to determine the function $S(x)$, which are provided by two addition boundary conditions at two ends.

The natural spline is given by the choices of $S_0''(x_0) = 0$ and $S_{n-1}''(x_n) = 0$. Other choices could set the first or second order derivatives be given values.

To construct the cubic spline, we let $M_j = S''(x_j)$, and we start with the linear function $S_j''(x)$ which can be obtained by the Lagrange interpolation of two points (x_j, M_j) and (x_{j+1}, M_{j+1}), expressed by,

$$S_j''(x) = \frac{1}{h_j}[-M_j(x - x_{j+1}) + M_{j+1}(x - x_j)],$$

where $h_j = x_{j+1} - x_j$.

Integrating this equation twice, and using conditions $S_j(x_j) = f_j$ and $S_j(x_{j+1}) = f_{j+1}$, we obtain the expression of the cubic function:

$$S_j(x) = M_j\frac{(x_{j+1} - x)^3}{6h_j} + M_{j+1}\frac{(x - x_j)^3}{6h_j} + \left(f_j - \frac{M_j h_j^2}{6}\right)\frac{x_{j+1} - x}{h_j} + \left(f_{j+1} - \frac{M_{j+1}h_j^2}{6}\right)\frac{x}{}$$

provided M_j for $j = 0, 1, \cdots, n-1$.

Now consider how to determine M_j. Notice we did not use the continuity condition of the first derivative $S'(x)$, in interval $[x_j, x_{j+1}]$, which is given by,

$$S_j'(x) = -M_j\frac{(x_{j+1} - x)^2}{2h_j} + M_{j+1}\frac{(x - x_j)^2}{2h_j} + \frac{f_{j+1} - f_j}{h_j} - \frac{M_{j+1} - M_j}{6}h_j.$$

We have

$$S_j'(x_j) = -\frac{h_j}{3}M_j - \frac{h_j}{6}M_{j+1} + \frac{f_{j+1} - f_j}{h_j}.$$

And similarly, using the cubic polynomial in interval $[x_{j-1}, x_j]$ gives,

$$S_{j-1}'(x_j) = \frac{h_{j-1}}{6}M_{j-1} + \frac{h_{j-1}}{3}M_j + \frac{f_j - f_{j-1}}{h_{j-1}}.$$

Then we obtain an expression of the first order continuity,

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = d_j, \quad \text{for } j = 1, 2, \cdots, n-1,$$

where

$$\mu_j = \frac{h_{j-1}}{h_{j-1}+h_j}, \quad \lambda_j = \frac{h_j}{h_{j-1}+h_j}, \text{ and } d_j = 6f[x_{j-1}, x_j, x_{j+1}],$$

for $j = 1, 2, \cdots, n-1$.

For natural boundary conditions, $M_0 = M_n = 0$ are fixed. We can write the above equations in a matrix form

$$\begin{pmatrix} 2 & \lambda_1 & 0 & \cdots & \cdots & 0 \\ \mu_2 & 2 & \lambda_2 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \mu_{n-2} & 2 & \lambda_{n-2} \\ 0 & \cdots & \cdots & 0 & \mu_{n-1} & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}.$$

5.3 Polynomial approximation

Let $P_0(x) \in \mathbb{P}_n$ satisfy $||P_0(x) - f(x)|| = \min_{P \in \mathbb{P}_n} ||P(x) - f(x)||$. Then $P_0(x)$ is the best approximation under the norm $||\cdot||$. We have

$$L^\infty\text{norm} : ||f||_\infty = \max_{[a,b]} |f(x)|;$$

$$L^1\text{norm} : ||f||_1 = \int_a^b |f(x)|dx;$$

$$L^p\text{norm} : ||f||_p = \left(\int_a^b |f(x)|^p dx \right)^{1/p}.$$

If using L^∞ norm, it is called the **best uniform approximation**; if L^2 norm, it is the **best square approximation**.

Least squares approximation

Find function $P_m(x)$ which approximates the data set $\{(x_j, f(x_j)), j = 0, \cdots, n\}$, with $m < n$, and

$$P_m(x) = \sum_{k=0}^{m} a_k x^k.$$

Square error function: $I(a_1, \cdots, a_n) = \sum_{j=0}^{n} [P_m(x_j) - f(x_j)]^2$, which is a discrete L^2 norm.

The least squares approximation is obtained by minimizing the above error function, through,

$$\frac{\partial I}{\partial a_l} = 0, \quad \text{and,} \quad \frac{\partial^2 I}{\partial a_l^2} > 0, \quad l = 0, \cdots, m.$$

We have,

$$\frac{\partial I}{\partial a_l} = 2 \sum_{j=0}^{n} \left[\sum_{k=0}^{m} a_k x_j^k - f(x_j) \right] x_j^l = 0.$$

Hence,

$$\sum_{j=0}^{n} \sum_{k=0}^{m} a_k x_j^k x_j^l = \sum_{j=0}^{n} f(x_j) x_j^l, \quad \text{for } j = 0, 1, \cdots, m.$$

In matrix form, let,

$$A = \begin{pmatrix} 1 & x_0 & x_0^2 \cdots & x_0^m \\ 1 & x_1 & x_1^2 \cdots & x_1^m \\ & & \cdots & \\ 1 & x_n & x_n^2 \cdots & x_n^m \end{pmatrix}_{n \times m},$$

then the system can be rewritten as,

$$(A^T A)\mathbf{a} = A^T \mathbf{f}.$$

Best square approximation

Definition. Find $P^*(x) \in \mathbb{P}_k$, s.t.,

$$\int_a^b |P^*(x) - f(x)|^2 dx = \min_{P \in \mathbb{P}_k} \int_a^b |P(x) - f(x)|^2 dx.$$

The problem is called the **best square approximation.**

Let $P(x) = c_0\varphi_0(x) + \cdots + c_k\varphi_k(x)$ where $\{\varphi_0(x), \cdots, \varphi_k(x)\}$ is a group of basis function. Let $I(c_0, \cdots, c_k) = \int_a^b |P^*(x) - f(x)|^2 dx$, then the minimization problem is equivalent to: Find $(c_0^*, \cdots, c_k^*)$, such that,

$$I(c_0^*, \cdots, c_k^*) = \min_{c_0, \cdots, c_k} I(c_0, \cdots, c_k).$$

Naturally, we have $\frac{\partial I}{\partial c_j} = 0, \quad j = 0, 1, \cdots, k$, which leads us to,

$$\sum_{i=0}^{k} \left[\int_a^b \varphi_i(x)\varphi_j(x)dx \right] c_i = \int_a^b f(x)\varphi_j(x)dx, \quad j = 0, 1, \cdots, k.$$

Orthogonal polynomials

Definition. Let $\rho(x) \geq 0$ for $x \in [a, b]$, which satisfies that for continuous $g(x) \geq 0$, $\int_a^b \rho(x)g(x)dx = 0$ implies $g(x) = 0, x \in [a, b]$. Then for any $f, g \in C[a, b]$, we call

$$(f, g) = \int_a^b \rho(x)f(x)g(x)dx$$

the weighted inner product of f and g. We call $||f||_2 = \sqrt{(f, f)}$ the weighted 2 norm, and $\rho(x)$ the weight function.

Definition. We call $f, g \in C[a, b]$ orthogonal, if their product $(f, g) = 0$.

Using the inner product, we can write the coefficient matrix of the equations if last section as,

$$A = \begin{pmatrix} (\varphi_0, \varphi_0) & (\varphi_1, \varphi_0) \cdots & (\varphi_k, \varphi_0) \\ (\varphi_0, \varphi_1) & (\varphi_1, \varphi_1) \cdots & (\varphi_k, \varphi_1) \\ & \cdots & \\ (\varphi_0, \varphi_k) & (\varphi_1, \varphi_k) \cdots & (\varphi_k, \varphi_k) \end{pmatrix},$$

with the weight function $\rho(x) = 1$.

Definition. Let $\mathbb{P}[a, b]$ be the linear space composed of all real polynomials. Suppose $\{\varphi_0(x), \varphi_1(x), \cdots\}$ are linearly independent. If

$$\int_a^b \rho(x)\varphi_j(x)\varphi_l(x)dx = 0, \quad \forall\, j \neq l,$$

then $\{\varphi_0(x), \varphi_1(x), \cdots\}$ is called the **orthogonal polynomial series** with weight $\rho(x)$ in interval $[a, b]$.

Gram-Schmidt orthogonalization for the basis function $\{1, x, x^2, \cdots\}$:

$$\begin{cases} \varphi_0(x) = 1, \\ \varphi_{j+1}(x) = x^{j+1} - \sum_{i=0}^{j} \dfrac{(x^{j+1}, \varphi_i(x))}{(\varphi_i(x), \varphi_i(x))}\varphi_i(x), \quad j = 0, 1, 2, \cdots \end{cases}$$

(1). Legendre polynomials ($[a, b] = [-1, 1]$, $\rho(x) = 1$):

$$\begin{cases} \varphi_0(x) = 1, \ \ \varphi_1(x) = x, \\ \varphi_{j+1}(x) = \frac{2j+1}{j+1}x\varphi_j(x) - \frac{k}{k+1}\varphi_{j-1}(x), \quad j = 1, 2, \cdots \end{cases}$$

(2). Chebyshev polynomials ($[a, b] = [-1, 1]$, $\rho(x) = 1/\sqrt{1 - x^2}$):

$$\begin{cases} T_0(x) = 1, \ \ T_1(x) = x, \\ T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x), \quad j = 1, 2, \cdots \end{cases}$$

(3). Lagurre polynomials ($[a, b] = [-1, \infty)$, $\rho(x) = \exp(-x)$):

$$\begin{cases} Q_0(x) = 1, \ \ Q_1(x) = 1 - x, \\ Q_{j+1}(x) = (1 + 2j - x)Q_j(x) - j^2Q_{j-1}(x), \quad j = 1, 2, \cdots \end{cases}$$

(4). Hermite polynomials ($(a, b) = (-\infty, \infty)$, $\rho(x) = \exp(x^2)$):

$$\begin{cases} H_0(x) = 1, \ \ H_1(x) = 2x, \\ H_{j+1}(x) = 2xH_j(x) - 2jH_{j-1}(x), \quad j = 1, 2, \cdots \end{cases}$$

Best uniform approximation

Chebyshev polynomial: $T_n(x) = \cos(n \arccos x)$, for $x \in [-1, 1]$, which is defined recursively by,

$$T_0(x) = 1, \ \ T_1(x) = x, \ \ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

The Chebyshev polynomials are orthogonal with respect to the weight $(1 - x^2)^{-1/2}$.

The zeros of the Chebyshev polynomial are $\cos(\frac{\pi}{2}\frac{2k-1}{n}),\ \ k = 1, 2, \cdots, n.$

The resulting interpolation polynomial with the zeros as nodes minimizes the problem of Runge's phenomenon and provides an approximation that is close to the polynomial of best approximation to a continuous function under the maximum norm.

Theorem. If $P(x)$ is the interpolating polynomial of degree at most n with nodes at the roots of $T_{n+1}(x)$, then,

$$\max_{x\in[-1,1]} |f(x) - P(x)| \leq \frac{1}{2^n(n+1)!} \max_{x\in[-1,1]} |f^{(n+1)}(x)|,$$

for $f \in C^{n+1}[-1, 1]$.

5.4 Linear and nonlinear regression

Linear regression model can be written as,

$$\begin{cases} y_1 = \beta_0 + \beta_1 x_{11} + \cdots + \beta_m x_{1m} + \varepsilon_1, \\ \quad \cdots \qquad \cdots \\ y_n = \beta_0 + \beta_1 x_{n1} + \cdots + \beta_m x_{nm} + \varepsilon_n, \end{cases}$$

where $\mathbf{X}_i$ are n measurements, $\mathbf{Y}$ are predictions, $n > m$, and $\varepsilon_i \sim N(0, \sigma^2)$ are errors.

In matrix form, the model is,

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon, \quad \varepsilon \sim N_n(0, \sigma^2 \mathbf{I}_n).$$

Solving least square problems by QR factorization

The least square estimate: To find $\hat{\beta}$, s.t.,

$$||\mathbf{Y} - \mathbf{X}\hat{\beta}||^2 = \min_{\beta} ||\mathbf{Y} - \mathbf{X}\beta||^2$$

where $||\cdot||$ is the vectorial L_2 norm, and $R(\hat{\beta}) = ||\mathbf{Y} - \mathbf{X}\hat{\beta}||$ is called the residue of the model.

QR factorizaiton is to decompose a matrix $\mathbf{A}$ into the product of an orthonormal matrix $\mathbf{Q}$ and an upper triangular $\mathbf{R}$ matrix, $\mathbf{A} = \mathbf{QR}$.

The least square problem is to solve

$$\min_{\beta} ||\mathbf{X}\beta - \mathbf{Y}||^2$$

with $\mathbf{X}$ being a $n \times m$ matrix. With the QR factorization, $\mathbf{X} = \mathbf{QR}$, we can write

$$||\mathbf{X}\beta - \mathbf{Y}|| = ||\mathbf{R}\beta - \mathbf{Q}^T\mathbf{Y}||$$

where $\mathbf{R} = (\mathbf{R}_1, \mathbf{0}_{(n-m)\times m})^T$ and $\mathbf{R}_1$ is an $m \times m$ upper triangle matrix. At once, we have

$$||\mathbf{X}\beta - \mathbf{Y}|| = (||\mathbf{R_1}\beta - \mathbf{c_1}||^2 + ||\mathbf{c_2}||^2)^{1/2}.$$

Clearly, the minimum of $||\mathbf{X}\beta - \mathbf{Y}||^2$ is $||\mathbf{c_2}||^2$, and the solution the least square problem is that of $\mathbf{R_1}\beta = \mathbf{c_1}$.

Solving least square problems by Cholesky factorization

The Cholesky factorization is of the form, $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is a lower triangular matrix, and $\mathbf{A}$ must be a symmetric and positive definite matrix.

In the least square problem, the solution can be written as,

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}.$$

By the Cholesky factorization, $\mathbf{X}^T\mathbf{X} = \mathbf{L}\mathbf{L}^T$, the solution is

$$\hat{\beta} = (\mathbf{L}^T)^{-1}\mathbf{L}^{-1}\mathbf{X}^T\mathbf{Y}.$$

Since matrix $\mathbf{L}$ is triangular, the product of its inverse matrix and a vector can be easily done.

Nonlinear regression

A nonlinear regression model can be often written as,

$$\mathbf{Y} = \mathbf{\Phi}(\mathbf{X}, \beta) + \varepsilon, \quad \varepsilon \sim N_n(0, \sigma^2\mathbf{I}_n),$$

and we need find the solution of nonlinear least square problems,

$$||\mathbf{Y} - \mathbf{\Phi}(\mathbf{X}, \hat{\beta})||^2 = \min_{\beta} ||\mathbf{Y} - \mathbf{\Phi}(\mathbf{X}, \beta)||^2.$$

This is an optimization problem.

5.5 Matrix factorization

Most of statistical computation contents are related to matrix computation. This is then the object of this section to introduce matrix factorization algorithms, including LU, Cholesky, and QR factorizations.

LU factorization

For any $n \times n$ matrix A, in general, to solve the linear system, we decompose it into a product of a lower-triangular matrix L and and a upper-triangular matrix U, the so-called LU decomposition,

$$A = LU,$$

with $L_{ij} = 0$ for $i < j$ and $U_{ij} = 0$ for $i > j$.

If we choose $U_{ii} = 1$, it is called the Crout factorization. If we choose $L_{ii} = 1$ it is the Doolittle factorization.

If we can write $\mathbf{A} = \mathbf{LU}$, where $\mathbf{L}$ and $\mathbf{U}$ are a lower and upper triangular matrices, respectively, the solution of $\mathbf{Ax} = \mathbf{b}$ is solved by two steps: (1) solve $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$; and (2) solve $\mathbf{Ux} = \mathbf{y}$ for $\mathbf{x}$.

Doolittle decomposition:

$$\mathbf{LU} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix}.$$

Computational procedure is as follows:
1. $u_{1i} = a_{1i}, \;\; l_{i1} = a_{i1}/u_{11}, \;\; i = 2, 3, \cdots, n$
Calculate r-th row of U, and r-th column of L for $r = 2, \cdots, n$ by
2. $u_{ri} = a_{ri} - \sum_{k=1}^{r-1} l_{rk} u_{ki}, \quad i = r, r+1, \cdots, n;$
3. $l_{ir} = (a_{ir} - \sum_{k=1}^{r-1} l_{ik} u_{kr})/u_{rr}, \quad i = r+1, \cdots, n \;\; (r \neq n).$

The back-substitution step is straightforward, and not presented here.

For example,

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 2 \\ 3 & 1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 14 \\ 18 \\ 20 \end{pmatrix}$$

we have LU decomposition

$$LU = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -5 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & -4 \\ 0 & 0 & -24 \end{pmatrix}$$

solved by,

$$Ly = (14, 18, 20)^T, \;\; \text{gives} \;\; y = (14, -10, -72)^T$$

$$Ux = (14, -10, -72)^T, \text{ gives } y = (1, 2, 3)^T.$$

Factorization of symmetric matrix

If $\mathbf{A}$ symmetric, $\mathbf{U}$ can be written as

$$\mathbf{U} = \mathbf{D}\mathbf{L}^T = \begin{pmatrix} u_{11} & & & \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{pmatrix} \begin{pmatrix} 1 & \frac{u_{12}}{u_{11}} & \cdots & \frac{u_{2n}}{u_{11}} \\ & 1 & \cdots & \frac{u_{2n}}{u_{22}} \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix}.$$

Theorem. For a symmetric matrix A, if all the order principal minor determinants (determinants of principal sub-matrices) are non-zero, then the factorzation $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ is unique.

If we further assume the matrix is positive definite, D can be decomposed by

$$\mathbf{D} = \mathbf{D}^{\frac{1}{2}}\mathbf{D}^{\frac{1}{2}} = \begin{pmatrix} \sqrt{d_1} & & & \\ & \sqrt{d_2} & & \\ & & \ddots & \\ & & & \sqrt{d_n} \end{pmatrix} \begin{pmatrix} \sqrt{d_1} & & & \\ & \sqrt{d_2} & & \\ & & \ddots & \\ & & & \sqrt{d_n} \end{pmatrix}.$$

We have the following theorem.

Theorem (Cholesky factorization). If symmetric matrix $\mathbf{A}$ is positive definite, there exists a nonsingular lower triangular matrix $\mathbf{L}$ such that $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. If we restrict the diagonal elements positive, the factorization is unique.

Tridiagonal matrix

To solve $Ax = f$, we can first solve $Ly = f$, then $Ux = y$.

For a tridiagonal matrix, the decomposition is extremely simple and of the

form (Crout factorization),

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_1 & & & & & \\ r_2 & \alpha_2 & & & & \\ & \ddots & \ddots & & & \\ & & & r_{n-1} & \alpha_{n-1} & \\ & & & & r_n & \alpha_n \end{pmatrix} \begin{pmatrix} 1 & \beta_1 & & & & \\ & 1 & \beta_2 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & \beta_{n-1} \\ & & & & & 1 \end{pmatrix},$$

where, obviously, the matrix identities satisfy,

$$\begin{cases} b_1 = \alpha_1, \;\; c_1 = \alpha_1\beta_1, \\ a_i = r_i, \;\; b_i = r_i\beta_{i-1} + \alpha_i, \;\; i = 2, 3, \cdots, n, \\ c_i = \alpha_i\beta_i, \;\; i = 2, 3, \cdots, n-1. \end{cases}$$

The **chasing method** to solve the tridiagonal matrix is composed of three steps:

1. Compute $\{\beta_i\}$ recursively,

$$\beta_1 = c_1/b_1, \;\text{ and }\; \beta_i = c_i/(b_i - a_i\beta_{i-1}), \;\text{ for }\; i = 2, 3, \cdots, n-1;$$

2. Solve $Ly = f$,

$$y_1 = f_1/b_1, \;\text{ and }\; y_i = (f_i - a_i y_{i-1})/(b_i - a_i\beta_{i-1}), \;\text{ for }\; i = 2, 3, \cdots, n;$$

3. Solve $Ux = y$,

$$x_n = y_n, \;\text{ and }\; x_i = y_i - \beta_i x_{i+1}, \;\text{ for }\; i = n-1, n-2, \cdots, 2, 1.$$

Householder QR factorization

We describe the QR decomposition to solve the linear system $\mathbf{Ax} = \mathbf{b}$.

Definition. A factorization $\mathbf{A} = \mathbf{QR}$ where $\mathbf{Q}$ is an orthogonal matrix, and $\mathbf{R}$ is an upper triangular matrix, is called a QR decomposition.

A matrix $\mathbf{Q}$ is **orthogonal** (sometimes also called orthonormal matrix) of $Q^TQ = I$. (We are considering the real matrix. For the matrix correspondence, it is called unitary if $Q^*Q = I$.)

To find the QR decomposition, we need Householder reflections (transformations).

Definition. (Householder matrix) Let $\mathbf{w} \in R^n$, with $\mathbf{w}^T\mathbf{w} = 1$, i.e., $\mathbf{w}$ is a unit vector. Then the $n \times n$ matrix $\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T$ is called a Householder matrix (Householder transform.)

Theorem. If $\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T$ is a Householder matrix, then $\mathbf{H}$ is symmetric and orthogonal.

The symmetry is obvious. Let us consider the orthogonality. We have

$$H^TH = (I - 2ww^T)(I - 2ww^T) = I - 4ww^T + 4w(w^Tw)w^T = I.$$

Geometrically a Householder matrix corresponds to reflection across the plane through the origin orthogonal to w, say, the super plane $w^Tx = 0$. To see this, for any x, we write $x = ww^Tx + y$, with a component y orthogonal to w, and the component ww^T in the w-direction. Then we obtain $Hx = x - 2ww^Tx = -ww^Tx + y$; i.e., Hx has the opposite component $-ww^T$ in the w-direction and the same component y orthogonal to the super plane. Because of this property, H is also called the Householder reflection matrix.

Theorem. For any vectors $\mathbf{x}$ and $\mathbf{y}$ in R^n, if they satisfy $||\mathbf{x}||_2 = ||\mathbf{y}||_2$, then there exists a Householder transformation $\mathbf{H}$, such that $\mathbf{y} = \mathbf{H}\mathbf{x}$.

The proof of this theorem is simple. Just let $\mathbf{w} = \frac{\mathbf{x}-\mathbf{y}}{||\mathbf{x}-\mathbf{y}||_2}$, then

$$\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T = \mathbf{I} - 2\frac{(\mathbf{x}-\mathbf{y})}{||\mathbf{x}-\mathbf{y}||^2}(\mathbf{x}^T - \mathbf{y}^T).$$

Therefore,

$$\mathbf{H}\mathbf{x} = \mathbf{x} - 2\frac{(\mathbf{x}-\mathbf{y})}{||\mathbf{x}-\mathbf{y}||^2}(\mathbf{x}^T - \mathbf{y}^T)\mathbf{x} = \mathbf{x} - 2\frac{(\mathbf{x}^T\mathbf{x} - \mathbf{y}^T\mathbf{x})}{||\mathbf{x}-\mathbf{y}||^2}(\mathbf{x}-\mathbf{y}),$$

and

$$||\mathbf{x}-\mathbf{y}||^2 = (\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y}) = 2(\mathbf{x}^T\mathbf{x} - \mathbf{y}^T\mathbf{x}),$$

where we use the property $||\mathbf{x}||_2 = ||\mathbf{y}||_2$. So

$$\mathbf{Hx} = \mathbf{x} - (\mathbf{x} - \mathbf{y}) = \mathbf{y}.$$

For a given column vector $\mathbf{a} = (a_1, a_2, \cdots, a_n)^T$, we can find a Householder matrix $\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T$ such that,

$$\mathbf{Ha} = \begin{pmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where

$$\sigma = \pm||\mathbf{a}||, \quad \mathbf{w} = \frac{\mathbf{v}}{||\mathbf{v}||}, \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} a_1 - \sigma \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

This leads to,

$$\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T = \mathbf{I} - \frac{2}{\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T.$$

QR factorization. The basic idea of the QR decomposition of matrix $\mathbf{A}$ using Householder reflections is to construct a sequence of Householder matrices such that,

$$\mathbf{H_1 A} = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix},$$

$$\mathbf{H_2 H_1 A} = \begin{pmatrix} * & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{pmatrix},$$

and if we assume $\mathbf{A}$ is $m \times n$ with $m > n$, then

$$\mathbf{H_n H_{n-1}} \cdots \mathbf{H_1 A} = \mathbf{R} = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ & \ddots & \ddots & \vdots \\ & & \ddots & * \\ & & & 0 \\ & & & \vdots \end{pmatrix}.$$

This can be written as $\mathbf{A} = \mathbf{QR}$, where

$$\mathbf{Q} = \mathbf{H_n H_{n-1}} \cdots \mathbf{H_1}^{-1} = \mathbf{H_1 H_2} \cdots \mathbf{H_n},$$

due to the symmetric and orthogonal properties of the Householder transformations.

If the first column of $\mathbf{A}$ is $(a_{11}, a_{21}, \cdots, a_{n1})^T$, we choose

$$r_1 = \pm\sqrt{a_{11}^2 + \cdots + a_{n1}^2}$$

(choose the opposite sign of a_{11}), and let

$$\mathbf{v_1} = \begin{pmatrix} a_{11} - r_1 \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}, \quad \text{and,} \mathbf{H_1} = \mathbf{I} - \frac{2}{\mathbf{v_1^T v_1}} \mathbf{v_1 v_1^T}.$$

If the second column of $\mathbf{H_1 A}$ is $(a_{12}, a_{22}, \cdots, a_{n2})^T$, we choose

$$r_2 = \pm\sqrt{a_{22}^2 + a_{32}^2 + \cdots + a_{n2}^2}$$

(the sign of r_2 is opposite to that of a_{22}), and let

$$\mathbf{v_2} = \begin{pmatrix} 0 \\ a_{22} - r_2 \\ a_{32} \\ \vdots \\ a_{n2} \end{pmatrix}, \quad \text{and,} \mathbf{H_2} = \mathbf{I} - \frac{2}{\mathbf{v_2^T v_2}} \mathbf{v_2 v_2^T}.$$

To efficiently evaluate $\mathbf{H_1 A}$, we can use the following procedure:
(1) calculate $\beta_0 = 2/\mathbf{v_1^T v_1}$;
(2) for $j = 2, 3, \cdots, n$, i) calculate $\beta_1 = \mathbf{v}_1^T \omega$ where ω is the jth column of $\mathbf{A}$, ii) set $\beta = \beta_0 \beta_1$, and iii) set the jth column of $\mathbf{H}_1\mathbf{A}$ as $\omega - \beta \mathbf{v}_1$.

The evaluation of β and $\omega - \beta \mathbf{v}_1$ requires $2n$ operations, respectively. The total number of multiplication operations for the 1st step is about $2n^2$. The total number of the QR factorization of an $n \times n$ matrix is thus,

$$2n^2 + 2(n-1)^2 + \cdots + 2 \cdot 2^2 \approx \frac{2}{3} n^3.$$

Iteration for linear system

For the point of view of iteration, we can construct a sequence $x^{(k+1)} = Bx^{(k)} + f, \ \ k = 0, 1, 2$ and the iteration is convergent to x^*, then x^* satisfies $x^* = Bx^* + f$.

For example, let $D, -L$ and $-U$ be the diagonal, lower triangular, and upper triangular matrices, $A = D - L - U$. We obtain $Dx = (L+U)x + b$. Thus, we have the form of $x = Bx + f$ with

$$B = D^{-1}(L+U), \quad \text{and} \ \ f = D^{-1}b.$$

We call B is the iteration matrix of the **Jacobi iteration.**

The iteration scheme of the Jacobi iteration is

$$Dx^{(k+1)} = (L+U)x^{(k)} + b.$$

The Gauss-Seidel iteration is a little bit different, where

$$B = (D-L)^{-1}U, \quad \text{and} f = (D-L)^{-1}b.$$

The iteration scheme is

$$Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b.$$

In component form, it is

$$a_{ii}x_i^{(k+1)} = -\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i, \ \ i = 1, 2, \cdots$$

Example. Solve the following equations with GS iteration:

$$\begin{aligned} 8x_1 - 3x_2 + 2x_3 &= 20 \\ 4x_1 + 11x_2 - x_3 &= 33 \\ 6x_1 + 3x_2 + 12x_3 &= 36. \end{aligned}$$

with $x^{(0)} = (0, 0, 0)^T$. The solution is $(3, 2, 1)$.

Convergence. The sufficient and necessary condition for the convergence of the Jacobi or Gauss-Seidel iteration is that the spectral radius of the iteration matrix, $\rho(B) < 1$, where $B = D^{-1}(L + U)$ for the Jacobi, and $B = (D - L)^{-1}U$ for the Gauss-Seidel. The main reason is that $\rho(B) < 1$ implies $\lim_{k\to\infty} B^k = 0$.

Theorem. Let $x^{(n+1)} = Bx^{(n)} + f$ be an iteration scheme of $x = Bx + f$. If there is some norm such that $||B|| = q < 1$, then:
(1) For any $x^{(0)}$, $\lim_{k\to\infty} = x^*$, and $x^* = Bx^* + f$;
(2) $||x^* - x^{(k)}|| \leq q^k ||x^* - x^{(0)}||$.

Definition (Diagonally dominant matrix). $A = (a_{ij})_{n\times n}$.
(1) We call A is **strictly diagonally dominant matrix** if

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, 2, \cdots, n$$

(2) A is weakly diagonally dominant matrix if

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, 2, \cdots, n$$

and at least one inequality holds.

If A is a strictly DD matrix, then A is nonsingular. Otherwise, $\det A = 0$, i.e., $\exists x \neq 0$, s.t., $Ax = 0$. Then

$$\sum_{j=1}^{n} a_{ij} x_j = 0.$$

Let $|x_k| = \max |x_j|$, then

$$|a_{kk}x_k| = |\sum_{j=1,j\neq k}^{n} a_{kj}x_j| \leq \sum_{j=1,j\neq k}^{n} |a_{kj}||x_j| \leq |x_k| \sum_{j=1,j\neq k}^{n} |a_{kj}|$$

So $|a_{kk}| \leq \sum_{j=1,j\neq k}^{n} |a_{kj}|$. Contradiction!

Definition (Reducible and irreducible matrix). If there exists a permutation matrix P such that

$$P^T AP = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

where A_{11} is $r \times r$ square matrix, A_{22} is $(n-r) \times (n-r)$ matrix, and $1 \leq r < n$, then, A is **reducible matrix**, or if there does not exist such a permutation matrix, A is **irreducible matrix.**

Examples. A is a tridiagonal matrix. Is A a reducible matrix?

If $B = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}$, is B a reducible matrix?

Theorem. Solve $Ax = b$.
If A is a strictly diagonally dominant matrix, then the Jacobi and Gauss-Seidel iterations are both convergent.
If A is weakly diagonally dominant and irreducible, the Jacobi and Gauss-Seidel iterations are convergent.

Proof. Let us consider the Gauss-Seidel, and the case of A being a strictly diagonally dominant matrix. The iteration matrix $B = (D-L)^{-1}U$, so $\det(\lambda I - B) = \det(\lambda I - (D-L)^{-1}U) = \det((D-L)^{-1}) \det(\lambda(D-L) - U)$. The eigenvalues of B is the roots of $\det(\lambda(D-L) - U) = 0$. Let $C = \lambda(D-L) - U$. We just need to prove that the following contradiction: when $\lambda \geq 1$ the determinant of C does not equal 0. This can be reached

that, if $\lambda \geq 1$,

$$C = \lambda(D-L) - U = \begin{pmatrix} \lambda a_{11} & a_{12} \cdots a_{1n} \\ \lambda a_{21} & \lambda a_{22} \cdots a_{2n} \\ \cdots & \\ \lambda a_{n1} & \lambda a_{n2} \cdots \lambda a_{nn} \end{pmatrix}$$

and by using the properties that A is strictly diagonally dominant,

$$|c_{ii}| = |\lambda a_{ii}| > |\lambda|(\sum_{j=1}^{i-1}|a_{ij}| + \sum_{j=i+1}^{n}|a_{ij}|) \geq \sum_{j=1}^{i-1}|\lambda a_{ij}| + \sum_{j=i+1}^{n}|a_{ij}| = \sum_{j=1, j\neq i}^{n}|c_{ij}|$$

for $i = 1, 2, \cdots, n$, then C is also strictly diagonally dominant, and thus $\det(C) \neq 0$. Contradiction!

Successive over-relaxation (SOR) iteration

The successive over-relaxation (SOR) is a variant of the Gauss-Seidel method for solving a linear system of equations, resulting in faster convergence.

We introduce a parameter ω, called the relaxation factor, and write $(D - L - U)x = b$ as

$$\left(\frac{1}{\omega}D - L\right)x + \left((1 - \frac{1}{\omega})D - U\right)x = b.$$

Then we have the iteration scheme $x = Bx + f$ where

$$B = (D - \omega L)^{-1}\left((1-\omega)D + \omega U\right), \quad \text{and} \quad f = \omega(D - \omega L)^{-1}b.$$

The iteration scheme of the SOR iteration is as follows,

$$Dx^{(k+1)} = Dx^{(k)} + \omega(b + Lx^{(k+1)} + Ux^{(k)} - Dx^{(k)}),$$

and the computational procedure is,

$$a_{ii}x_i^{(k+1)} = a_{ii}x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \cdots$$

When $\omega = 1$, the SOR is the Gauss-Seidel iteration.
If $\omega > 1$, it is the over relaxation method. If $\omega < 1$, it is the low relaxation method.

The choice of ω is not easy, depends upon the properties of the matrix. Usually, $\omega > 1$ is used to o speedup convergence of a slow-converging process, while values of $\omega < 1$ are often used to help establish convergence of a diverging iterative process.

5.6 Project 4. Regression analysis

1. Solve the least square estimate by QR factorization in Matlab.

2. Do regression analysis with your subroutine for a model you like. You could analyze your own data or pick an example from the website and repeat their analysis:
http://www.neas-seminars.com/discussions/Forum175.aspx

Chapter 6

Markov Chain Monte Carlo

6.1 Background

The key issue of the MC algorithm is to sample iid samples from a probability distribution $\pi(x)$ where x could be a very highly dimensional variables. This section devotes to the MC methods based on Markov chains, say, the Metropolis algorithm, published by Metropolis, Rosenbluth, Rosenbluth, Teller and Teller in 1953.

Suppose we want to calculate the integral $\int f(x)\pi(x)dx$ where $\pi(x) = Z^{-1}e^{-\beta H(x)}$ be the object distribution we studied, and the normalization constant or the partition function Z is often unknown. In principle, $Z = \int e^{-\beta H(x)}dx$ could be known, but it is difficult to compute it, usually more difficult than obtaining a sample from π.

Example. The Ising model is a mathematical model of ferromagnetism in statistical mechanics. The model consists of discrete variables called spins that can be in one of two states. The spins are arranged in a lattice or graph, and each spin interacts only with its nearest neighbors. The goal is to find phase changes in the Ising model, as a simplified form of phase changes in real substances. The two-dimensional square lattice Ising model is one of the simplest statistical models to show a phase change.

Consider the 1d Ising model, composed of M sites, each locates an electron. $\sigma_i = \pm 1$ represents the spin-up and spin-down of the ith electron.

Then $\sigma = (\sigma_1, \cdots, \sigma_M)$ represents a microstate of the system. Clearly, there are 2^M possible states.

In the statistical physics, if there is no external field, the thermal macroscopic quantities can be obtained by a so-called ensemble average, for example, the internal energy,

$$U = \sum_{\sigma} \frac{e^{-\beta H(\sigma)}}{Z_M} H(\sigma) \doteq < H(\sigma) >$$

where the sum runs all the microstates, and $\beta = (k_B T)^{-1}$, and $< H(\sigma) >$ represents the average of the energy function $H(\sigma)$ under the probability density $\frac{e^{-\beta H(\sigma)}}{Z_M}$. The partition function is given by,

$$Z_M = \sum_{\sigma} e^{-\beta H(\sigma)},$$

and the energy function reads,

$$H(\sigma) = -J \sum_{i} \sigma_i \sigma_{i+1}$$

When $J > 1$, the interaction is called ferromagnetic, or it is called antiferromagnetic. A ferromagnetic interaction tends to align spins, and an antiferromagnetic tends to antialign them. The Ising model, strictly speaking, requires ferromagnetic interaction; the general model may be called the nonferromagnetic Ising model.

The special heat of the system is defined by,

$$C = \frac{\partial < H(\sigma) >}{\partial T} = \frac{\beta}{T} Var_{\pi}[H(\sigma)]$$

6.2 Properties of a Markov chain

A Markov chain is a random process with the Markov property, i.e. the property, simply said, that the next state depends only on the current state and not on the past. It is a Markov model, named for Andrey Markov, for

a particular type of Markov process in which the process can only be in a finite or countable number of states. Markov chains are useful as tools for statistical modeling in almost all fields of modern applied mathematics. (from wiki)

Definition. Let $X_0, X_1, \cdots$ be a sequence of random variables in a countable set of the state space S. The sequence is called a Markov chain if it satisfies the Markov property,

$$P(X_{t+1} = y | X_0 = x_0, X_1 = x_1, \cdots, X_t = x_t) = P(X_{t+1} = y | X_t = x_t),$$

namely, given the present state, the future and past states are independent.

Example. Random walk on a straight line. Let Z_n be iid random variables with Bernoulli distribution, i.e., $P(Z_i = 1) = 1 - P(Z_i = -1) = p$. Let $X_0 = 0$, and $X_t = X_{t-1} + Z_t$, then $X_t, t = 0, 1, \cdots$ forms a Markov chain which represents a random walk on the number line, at each step, with probability p to move to the right. When $t \to \infty$, X_t will drift to $\pm\infty$ direction, depending on the value of p.

Center-biased random walk. $P_{move\ left} = 0.5 + \frac{x}{2(1+|x|)}$. Then the probabilities of a move to the left at positions $x = -2, -1, 0, 1, 2$ are given by $\frac{1}{6}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{5}{6}$ respectively. Since the probabilities depend only on the current position and not on any prior positions, this biased random walk satisfies the definition of a Markov chain.

If the transition probability $P(X_{t+1} = y | X_t = x)$ is time-homogeneous, then we have a transition function $A(x, y)$ which defines a **transition matrix** $\mathbf{P} = (p_{ij})_{N_t \times N_t}$ where N_t is the number of all the microstates. The transition matrix is a **stochastic matrix**, which has the following properties,

(1). $p_{ij} \geq 0, \ \ i, j = 1, 2, \cdots, N_t$

(2). $\sum_{j=1}^{N_t} p_{ij} = 1, \ i = 1, 2, \cdots, N_t$, ($\sum_y A(x, y) = 1$.)

The condition (2) implies matrix $\mathbf{P}$ has an eigenvalue 1, and right eigenvector $(1, 1, \cdots, 1)^T$. From the Gerschgorin circle theorem, 1 is the spectral radius.

A very simple weather model. The probabilities of weather conditions, given the weather on the preceding day, can be represented by a transition matrix:

$$\mathbf{P} = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$$

The matrix P represents the weather model in which a sunny day is 90% likely to be followed by another sunny day, and a rainy day is 50% likely to be followed by another rainy day. The columns can be labelled "sunny" and "rainy" respectively, and the rows can be labelled in the same order.

Let $x_0 = (1, 0)$, then the weather on day 1 can be predicted by,

$$x_1 = x_0 P = (0.9, 0.1)$$

$$x_2 = x_1 P = (0.86, 0.14)$$

Generally, $x_n = x_{n-1} P = x_0 P^n$.

Predictions for the weather on more distant days are increasingly inaccurate and tend towards a steady state vector. This vector represents the probabilities of sunny and rainy weather on all days, and is independent of the initial weather. Let us consider,

$$\pi = \lim_{n \to \infty} x_n$$

which implies, if convergence,

$$\pi = \pi P$$

This makes it an eigenvector (with eigenvalue 1) of P. We have

$$0 = \pi(P - I) = \pi \begin{pmatrix} -0.1 & 0.1 \\ 0.5 & -0.5 \end{pmatrix},$$

i.e., $-0.1\pi_1 + 0.5\pi_2 = 0$, leading to

$$\pi = (0.833, 0.167).$$

Definition. If vector π satisfies $\pi = \pi\mathbf{P}$, that is, it is a left eigenvector of the matrix with eigenvalue 1, then $\pi$is called the **stationary probability vector** of the Markov chain.

Recall Definition (Reducible and irreducible matrix). If there exists a permutation matrix P such that

$$P^T AP = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

where A_{11} is $r \times r$ square matrix, A_{22} is $(N-r) \times (N-r)$ matrix, and $1 \leq r < N$, then, A is reducible matrix, or if there does not exist such a permutation matrix, A is irreducible matrix.

Perron-Frobenius theorem. If matrix $A \in R^{n \times n}$ is nonnegative (all elements) and irreducible, then
(1) $\lambda_1 = \rho(A) > 0$ is a single eigenvalue;
(2) There exists an eigenvector corresponding λ_1 such that all components are positive.
(3) There does not exist nonnegative eigenvector corresponding to other eigenvalues.

The Perron-Frobenius theorem states that the stationary distribution uniquely exists. Such a stochastic matrix is also called **ergodic.**

Definition. A Markov chain is said to show **detailed balance** if the transition matrix, P, obeys

$$\pi_i p_{ij} = \pi_j p_{ji}$$

between each pair of states i and j in the state space.

Proposition. The sufficient condition for the stationary distribution, π, is that, it satisfies the detailed balance condition.

Proof. We have

$$(\pi P)_j = \sum_{i=1}^{N_t} \pi_i p_{ij} = \sum_{i=1}^{N_t} \pi_j p_{ji} = \pi_j.$$

and thus we have $\pi P = \pi$.

This complete the proof.

6.3 Metropolis algorithm

Let us now turn to the Gibbs distribution π, where the σ state component is

$$\frac{1}{Z}e^{-\beta H(\sigma)} \doteq \pi(\sigma).$$

Then, π is a stationary distribution, but we do not know the transition matrix P as it has N_t^2 unknowns. But we have the detailed balance condition, which yields,

$$\frac{P(\sigma' \to \sigma)}{P(\sigma \to \sigma')} = \frac{\pi(\sigma)}{\pi(\sigma')} = e^{-\beta[H(\sigma)-H(\sigma')]}.$$

That is, by introducing the detailed balance condition, we can determine the ratio of the elements of the matrix.

The objective of the Metropolis algorithm is to generate a Markov chain, $\sigma^{(1)}, \sigma^{(2)}, \cdots, \sigma^{(n)}$, so that we can use the ensemble average. In order to satisfy the detailed balance condition, the Metropolis generates the sequence as follows,
(1) Create a candidate state σ';
(2) Judge to accept it or not. If accept, $\sigma^{(n+1)} = \sigma'$; or $\sigma^{(n+1)} = \sigma^{(n)}$.

Step (1).

There are usually two straightforward methods to generate the candidate state: i) randomly choose one from the left $N_t - 1$ states; ii) randomly choose one site from the M sites, and change its spin state.

Usually, the second method is more efficient.

Mathematically, step (1) defines a proposal matrix $\mathbf{T}$. For method i),

$$T(\sigma, \sigma') = \begin{cases} \frac{1}{N_t - 1}, & \sigma \neq \sigma', \\ 0, & \sigma = \sigma'. \end{cases}$$

For method ii), it is,

$$T(\sigma, \sigma') = \begin{cases} \frac{1}{M}, & \sigma, \sigma' \text{ are different at only one site,} \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, $\mathbf{T}$ is symmetric and irreducible.

Step (2). Accept or reject?

The rule is to use the detailed balance condition, which is the core of the Metropolis algorithm.

First, we compute the ratio,

$$R = \frac{\pi(\sigma')}{\pi(\sigma)} = e^{-\beta \Delta H},$$

where $\Delta H = H(\sigma') - H(\sigma)$.
If $\Delta H < 0$, say the energy of state σ' is lower than that of state σ, then $R > 1$ and σ' is accepted.
If $\Delta H > 0$, then $R < 1$, σ' is accepted with probability R, i.e., we have a probability to jump to high energy state.

Mathematically, we can write down the transition matrix $\mathbf{P}$ with identities as follows:

$$P(\sigma, \sigma') = T(\sigma, \sigma') \min\left\{\frac{\pi(\sigma')}{\pi(\sigma)}, 1\right\}.$$

It is not difficult to verify the detailed balance condition for $\mathbf{P}$,

$$\pi(\sigma)P(\sigma, \sigma') = T(\sigma, \sigma') \min\{\pi(\sigma), \pi(\sigma')\} = T(\sigma', \sigma) \min\{\pi(\sigma'), \pi(\sigma)\} = \pi(\sigma')P(\sigma', \sigma$$

Finally, in order to obtain states $\sigma^{(1)}, \sigma^{(2)}, \cdots$, we have,
Metropolis algorithm:
1. Generate σ' from $\sigma^{(n)}$
2. Define $\Delta H(\sigma) = H(\sigma') - H(\sigma)$ and compute $A = \min\{1, e^{-\beta \Delta H(\sigma)}\}$
3. Generate a random number $r \sim U[0, 1]$
4. If $r \leq A$, then $\sigma^{(n+1)} = \sigma'$; or $\sigma^{(n+1)} = \sigma^{(n)}$.

Primitivity. It can be verified that, the Markov chain corresponding to the Metropolis algorithm is a primitive Markov chain. (The positivity of the diagonal holds:

$$P(\sigma, \sigma) = 1 - \sum_{\sigma', \sigma \text{ are different at one site}} P(\sigma, \sigma')$$

where $P(\sigma, \sigma') = \frac{1}{M} \min\{1, R\} \leq \frac{1}{M}$, and thus $P(\sigma, \sigma) \geq \frac{1}{M}$).

6.4 Hastings's generalization

The Metropolis algorithm prescribes a transition rule for a Markov chain. It uses a symmetric proposal function $T(x, y)$ to suggest a possible move and employs an acceptance-rejection rule to "thin it down". Hastings extended the algorithm to the case when T is not necessarily symmetric. The only serious restriction on the proposal function is that

$$T(x, y) > 0 \Longleftrightarrow T(y, x) > 0.$$

Metropolis-Hastings algorithm. Given current state $x^{(n)}$:

1. Draw y from the proposal distribution $T(x^{(n)}, y)$.

2. Draw $U \sim$Uniform[0,1] and update,

$$x^{(n+1)} = \begin{cases} y, & \text{if } U \leq r(x^{(n)}, y) \\ x^{(n)}, & \text{otherwise} \end{cases}$$

where Metropolis et al. (1953) and Hastings (1970) suggested using

$$r(x, y) = \min \left\{ 1, \frac{\pi(y)T(y, x)}{\pi(x)T(x, y)} \right\}.$$

When $T(x, y) = T(y, x)$, the algorithm is identical to the original Metropolis algorithm.

Baker (1965) suggested another acceptance function:

$$r_B(x, y) = \frac{\pi(y)T(y, x)}{\pi(y)T(y, x) + \pi(x)T(x, y)}.$$

A more general formula is given by Charles Stein:

$$r(x, y) = \frac{\delta(x, y)}{\pi(x)T(x, y)},$$

where $\delta(x, y)$ is any symmetric function that makes $r(x, y) \leq 1$ for all x and y.

If a rejection function is used, then for any $y \neq x$, the actual transition probability is,

$$P(x,y) = T(x,y)r(x,y) = T(x,y)\frac{\delta(x,y)}{\pi(x)T(x,y)} = \pi(x)^{-1}\delta(x,y).$$

Because $\delta(x,y) = \delta(y,x)$, we have that $\pi(x)P(x,y) = \pi(y)P(y,x)$. This implies the Markov chain is reversible and has π as its invariant distribution.

6.5 Special algorithms

Random-walk Metropolis.

Suppose the target distribution $\pi(x)$ is defined on the d-dimensional Euclidean space. A natural "perturbation" of the current configuration is the addition of a random "error"; that is, the next candidate position is proposed as $x' = x^{(n)} + \varepsilon_n$, where $\varepsilon_n \sim g_\sigma(\cdot)$ is independent and identically distributed for different n, and σ represents the range of the proposal exploration is controlled by user. Often, a spherical Gaussian distribution $N(0, \sigma^2 I)$ is chosen.

Given the current state $x^{(n)}$, the random walk Metropolis algorithm iterates the following steps:

1. Draw $\varepsilon \sim g_\sigma$ and set $x' = x^{(n)} + \varepsilon$

2. Simulate $u \sim U[0,1]$ and update

$$x^{(n+1)} = \begin{cases} y & \text{if } u \leq \frac{\pi(x')}{\pi(x^{(n)})} \\ x^{(n)} & \text{otherwise.} \end{cases}$$

Metropolized independence sampler

A very special choice of the proposal transition function $T(x,y)$ is an independent trial density $g(y)$; that is, the proposal move y is generated from $g(\cdot)$ independent of the previous state $x^{(n)}$.

MIS scheme: given the current state $x^{(n)}$,

1. Draw $y \sim g(y)$

2. Simulate $u \sim U[0,1]$ and let

$$x^{(n+1)} = \begin{cases} y & \text{if } u \leq \frac{w(y)}{w(x^{(n)})} \\ x^{(n)} & \text{otherwise.} \end{cases}$$

where $w(x) = \pi(x)/g(x)$ is the usual importance sampling weight.

The efficiency of MIS depends on how close the trial density $g(y)$ is to the target $\pi(y)$. To ensure robust performance, it is advisable to let $g(\cdot)$ be a relatively long-tailed distribution. The idea is useful in many Bayesian computations in which each conditional density can be approximated reasonably well by a Gaussian distribution. To accommodate irregular tail behaviors, it is essential to use a long-tailed t-distribution as $g(x)$.

Multiple-try Metropolis

Often the Metropolis-Hastings algorithm leads to a small step-size in the proposal transition, resulting in exceedingly slow movement of the corresponding Markov chain, whereas a large step-size will result in very low acceptance rate. The mixing rate of the algorithm would be very slow. The multiple independent proposals enables a MCMC sampler to make large step-size jumps without lowering the acceptance rate.

Suppose $T(x,y)$ is an arbitrary proposal transition function and $\delta(x,y)$ is symmetric and non-negative. Define

$$w(x,y) = \pi(x)T(x,y)\lambda(x,y)$$

where $\lambda(x,y)$ is non-negative symmetric function in x and y that can be chosen by the user.

Suppose the current state is $x^{(n)} = x$, then a MTM transition is defined as follows:

1. Draw k independent trial proposals, $y_1, \cdots, y_k$ from $T(x,\cdot)$. Compute $w(y_j, x)$ for $j = 1, \cdots, k$

2. Select y among the trial set $\{y_1, \cdots, y_k\}$ with probability proposal to $w(y_j, x)$. Then produce a reference set by drawing $x_1^*, \cdots, x_{k-1}^*$ from the

distribution $T(y,\cdot)$. Let $x_k^* = x$.

3. Accept y with probability,

$$\tau_g = \min\left\{1, \frac{w(y_1,x)+\cdots+w(y_k,x)}{w(x_1^*,y)+\cdots+w(x_k^*,y)}\right\}$$

and reject it with probability $1-\tau_g$. The quantity τ_g is called the generalized MH ratio.

When $T(x,y)$ is symmetric, for example, one can choose $\lambda(x,y) = T^{-1}(x,y)$. Then $w(x,y) = \pi(x)$. In this case, the MTM algorithm is simplified as the *orientational bias* Monte Carlo in the field of molecular simulation.

6.6 Simulated annealing

Traveling salesman problem (TSP)

The Travelling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once. (from wiki)

Let the distance connecting i and j be $l_{ij} = l_{ji}$. Mathematically, this problem is described by,

$$\min_{x\in X}\left\{H(x) = \sum_{i=1}^{N-1} l_{x_i x_{i+1}}\right\}$$

where $X = \{(x_1, x_2, \cdots, x_N)$ which is a sequence of $1, 2, \cdots, N\}$, and $H(x)$ is called the path function.

There are $\frac{(N-1)!}{2}$ possible paths. It is a typical NP problem, where the state number increases exponentially with N.

Image smoothing problem

A picture has J pixels, each with 256 colors, denoted by,

$$X = \{(x_1, x_2, \cdots, x_J), \;\; x_i \in \{0, 1, \cdots, 255\}\}.$$

The smoothness of an image is defined by,

$$H(\mathbf{x}) \doteq \alpha \sum_{<s,t>} (x_s - x_t)^2, \;\; \alpha > 0, \;\; \mathbf{x} \in X,$$

where $\sum_{<s,t>}$ represents the sum of the nearest pairs of pixels.

For a given image $\mathbf{y} \in X$, define,

$$H(\mathbf{x}|\mathbf{y}) = \alpha \sum_{<s,t>} (x_s - x_t)^2 + \frac{1}{2\sigma^2} \sum_{s} (x_s - y_s)^2.$$

In practice, $\mathbf{y}$ is a polluted image with noises and defects. To obtain a smoother image, we can solve the following optimization problem:

$$\min_{\mathbf{x} \in X} H(\mathbf{x}|\mathbf{y}).$$

The number of possible states is 256^J, which is not applicable for traditional optimization method.

Simulated annealing

The SA algorithm was proposed by Kirkpatrick et al. in 1983, and has been widely used in many problems.

In condensed physics, the annealing is a thermal process that produces conditions by heating to above the recrystallization temperature and maintaining a suitable temperature, and then cooling, in order to reach the low energy state.

Let us consider the optimization problem,

$$\min_{x \in X} H(x),$$

where $H(x)$ has a set of global minimum points $M = \{x_0 | H(x_0) = \min_{x \in X} H(x)\}$. Let us introduce parameter $\beta > 0$, and define,

$$\Pi^\beta(x) = \frac{1}{Z_\beta} e^{-\beta H(x)}, \qquad Z_\beta = \sum_{x \in X} e^{-\beta H(x)}.$$

Clearly, $\Pi^\beta(x)$ is a probability density in X, the corresponding distribution is called the $\Pi^\beta(x)$ distribution.

The main theorem of the simulated annealing is the following:

Theorem. $\lim_{\beta\to+\infty}\Pi^{\beta}(x)=\begin{cases}\frac{1}{|M|}, & x\in M\\ 0, & \text{otherwise,}\end{cases}$

and when β is sufficiently large, then for any $x\in M$, $\Pi^{\beta}(x)$ is monotonically increasing function of β; and for $x\notin M$, $\Pi^{\beta}(x)$ is monotonically decreasing function of β.

Proof. Let $m=\min_{x\in X}H(x)$. Then

$$\begin{aligned}\Pi^{\beta}(x) &= \frac{e^{-\beta[H(x)-m]}}{\sum\limits_{z:H(z)=m}e^{-\beta[H(x)-m]}+\sum\limits_{z:H(z)>m}e^{-\beta[H(x)-m]}}\\ &\to\begin{cases}\frac{1}{|M|}, & x\in M\\ 0, & \text{otherwise,}\end{cases}\qquad(\beta\to+\infty).\end{aligned}$$

If $x\in M$, then

$$\Pi^{\beta}(x)=\frac{1}{|M|+\sum\limits_{z:H(z)>m}e^{-\beta[H(x)-m]}}.$$

Obviously, because of $\beta>0$, it is an increasing function of β.

If $x\notin M$, then

$$\begin{aligned}\frac{\partial\Pi^{\beta}(x)}{\partial\beta} &= \frac{e^{-\beta[H(x)-m]}[m-H(x)]\tilde{Z}_{\beta}}{\tilde{Z}_{\beta}^{2}}-\frac{e^{-\beta[H(x)-m]}\sum_{z\in X}e^{-\beta[H(z)-m]}[m-H(z)]}{\tilde{Z}_{\beta}^{2}}\\ &= \frac{e^{-\beta[H(x)-m]}}{\tilde{Z}_{\beta}^{2}}\left[(m-H(x))\tilde{Z}_{\beta}-\sum_{z\in X}e^{-\beta[H(z)-m]}[m-H(z)]\right]\end{aligned}$$

where

$$\tilde{Z}_{\beta}=\sum_{z\in X}e^{-\beta[H(z)-m]}.$$

Noticing that

$$\lim_{\beta\to+\infty}\left\{(m-H(x))\tilde{Z}_{\beta}-\sum_{z\in X}e^{-\beta[H(z)-m]}[m-H(z)]\right\}=|M|[m-H(x)]\le 0,$$

we complete the proof.

The theorem states if we can construct a chain with $\Pi^\beta(x)$ distribution, then if we increase β, then at $\beta = \infty$ the random number will jump among the minimum points. Physically, the energy minimum corresponds to a state of perfect crystals, and a local minimum means the structure includes defects.

Algorithm:
1. Initialize a state $x^{(0)}$ and β_0;
2. For each β_k, run the Metropolis sampling to obtain N_k samples from $\Pi^\beta(x)$;
3. Update β_k to β_{k+1}.

Define the Metropolis sampling of the simulated annealing,

$$P^\beta(x,y) = \begin{cases} T(x,y)\min\left\{\frac{\Pi^\beta(y)}{\Pi^\beta(x)}, 1\right\}, & x \neq y, \\ 1 - \sum_{z\neq x} P^\beta(x,z), & x = y. \end{cases}$$

Fundamental theorem of simulated annealing. Let X be a finite set, $H(x)$ is a non-trivial function, T is a symmetric and irreducible matrix. If $\beta_n \leq C\ln n$ (annealing speed), where constant C depends on T and function H, then for any initial distribution ν, we have,

$$\lim_{n\to+\infty} ||\nu P^{\beta_1}\cdots P^{\beta_n} - \Pi^\infty|| = 0,$$

where $\Pi^\infty = \lim_{\beta\to+\infty}\Pi^\beta(x)$.

Attractive tool for global optimization; widely used in many applications, but with slow convergence.

Implementation of simulated annealing

6.7 Project 5: Energy of Crystalized Particles on the Unit Sphere

This is a SIAM problem from:

http://www.siam.org/journals/categories/04-003.php

In this project, we study a structure of 100 charges on a spherical surface by the Monte Carlo simulations with simulated annealing. We consider a sphere of radius 1 with N unit point charges confined on the surface. Suppose the locations of these charges are $\mathbf{r}_j,\ j = 1, \cdots, N$, where $|\mathbf{r}_j| = 1$, then the electrostatic energy, i.e., the Hamiltonian, of the system is the combination of all pairs of Coulomb repulsion,

$$H = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}.$$

We calculate the minimum energy by Monte Carlo simulated annealing and understand the crystal structure at minimum energy. Discuss the simulated results.

Chapter 7

Optimization and EM Optimization

This chapter will discuss the optimization methods [1], followed with the EM optimization methods.

7.1 Steepest descent method

Direction of the steepest descent. Consider the following problem without constraint,

$$\min f(x), \quad x \in \mathbb{R}^n$$

where $\nabla f(x)$ are continuous. We hope to find the steepest descent direction from a starting point.

The direction derivative of $f(x)$ along direction d is equal to the product of its gradient and the direction,

$$\partial_d f(x) = \nabla f(x)^T d$$

To find the direction of the steepest descent, we solve the following optimization problem,

$$\min \nabla f(x)^T d, \quad \text{s.t.,} \ \|d\| \leq 1.$$

By the Schwartz inequality,

$$|\nabla f(x)^T d| \leq \|\nabla f(x)^T\| \|d\| \leq \|\nabla f(x)^T\|$$

[1] Baolin Chen, Optimization Theory and Algorithm, Tsinghua University Press

and then,

$$\nabla f(x)^T d \geq -\|\nabla f(x)^T\|.$$

We see the minimization is obtained when the equality holds, i.e.,

$$d = -\frac{\nabla f(x)}{\|\nabla f(x)^T\|}.$$

The negative gradient direction is the direction of the steepest descent.

The method of the steepest descent is the following iteration,

$$x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}$$

where $d^{(k)}$ is the searching direction starting from $x^{(k)}$, i.e., $d^{(k)} = -\nabla f(x^{(k)})$, and λ_k satisfies,

$$f(x^{(k)} + \lambda_k d^{(k)}) = \min_{\lambda \geq 0} f(x^{(k)} + \lambda d^{(k)}).$$

Steps:

1. Given initial $x^{(1)}$, and error tolerance ε, and set $k = 1$;

2. Compute $d^{(k)} = -\nabla f(x^{(k)})$;

3. If $\|d^{(k)}\| \leq \varepsilon$, stop the iteration; or compute λ_k;

4. Let $x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}$, and set $k := k + 1$, goto 2.

Example. Solve

$$\min \quad f(x) = 2x_1^2 + x_2^2$$

given initial data $x^{(1)} = (1, 1)^T$ and $\varepsilon = 0.1$.

Convergence. Suppose $f(x)$ has continuous second derivatives and $\bar{x}$ is a local extrema. The Hessian matrix $\nabla^2 f(\bar{x})$ has the smallest eigenvalue $a > 0$, and the largest eigenvalue A, then the sequence $\{x^{(k)}\}$ converges to $\bar{x}$ with convergence rate,

$$\left(\frac{A-a}{A+a}\right)^2.$$

Let $r = A/a$ be the condition number of the symmetric positive definite matrix $\nabla^2 f(\bar{x})$, then the convergence rate is $\left(\frac{r-1}{r+1}\right)^2$. This implies a fast convergence for a small condition number.

Explanation. Let $\varphi(\lambda) = f(x^{(k)} + \lambda d^{(k)})$, then

$$\varphi'(\lambda_k) = \nabla f(x^{(k)} + \lambda_k d^{(k)})^T d^{(k)} = {d^{(k+1)}}^T d^{(k)} = 0.$$

We see $d^{(k+1)}$ and $d^{(k)}$ are orthogonal. This property will lead to a sawtooth phenomenon during the iteration.

7.2 Newton's method

We have discussed the Newton's method in the Chapter of numerical algebra. To make a short overview, the iteration of the Newton's method is as follows,

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

Since $d^{(k)} = \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)})$ doesn't have to be the descent direction, the Newton's method may not converge. Similar to the steepest descent method, the **Damped Newton's method** fixes the flaw by adding a 1D searching with the iteration,

$$x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}$$

where the step length satisfies,

$$f(x^{(k)} + \lambda_k d^{(k)}) = \min_{\lambda} f(x^{(k)} + \lambda d^{(k)})$$

7.3 Conjugate gradient method

The key in the unconstraint optimization methods is the searching direction. The conjugate direction is an important concept, which is the basis of the conjugate gradient method.

Definition. Suppose A is $n \times n$ symmetric and positive definite matrix. We two vectors $d^{(1)}$ and $d^{(2)}$ are conjugate with respect to A, if they satisfy,

$${d^{(1)}}^T A d^{(2)} = 0.$$

We see the geometric meaning of the conjugate directions by considering the quadratic function,

$$f(x) = \frac{1}{2}(x - \bar{x})^T A(x - \bar{x})$$

where $\bar{x}$ is the minimum, i.e., $f(\bar{x}) = 0$. Let $x^{(1)}$ is a point on a contour surface. Then the normal direction is

$$\nabla f(x^{(1)}) = A(x^{(1)} - \bar{x}).$$

Suppose $d^{(1)}$ is a tangential vector at $x^{(1)}$, then $d^{(1)}$ and $\nabla f(x^{(1)})$ are orthogonal. Let's denote $d^{(2)} = x^{(1)} - \bar{x}$, then we find that $d^{(1)}$ and $d^{(2)}$ are conjugate with respect to A,

$${d^{(1)}}^T A d^{(2)} = 0.$$

And we find that $\bar{x}$ is at the direction of $d^{(2)}$.

The algorithm. We discuss the algorithm by the problem,

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

where A is symmetric and positive definite. The algorithm is as follows.

1. Given initial $x^{(1)}$, compute the gradient of the objective function, $g_1 = \nabla f(x^{(1)})$. Let $d^{(1)} = -g_1$, and set $k = 1$;

2. Suppose we know $x^{(k)}$ and $d^{(k)}$. We find λ_k to obtain $x^{(k+1)} = x^{(k)} + \lambda_k d^{(k)}$ which satisfies,

$$f(x^{(k)} + \lambda_k d^{(k)}) = \min f(x^{(k)} + \lambda d^{(k)}).$$

This can be obtained explicitly, $\lambda_k = -\frac{g_k^T d^{(k)}}{{d^{(k)}}^T A d^{(k)}}$.

3. Compute the gradient of $f(x)$ at $x^{(k+1)}$, $g_{k+1} = \nabla f(x^{(k+1)})$. If $\|g_{k+1}\| = 0$, stop the iteration. Or find the next searching direction $d^{(k+1)}$, s.t., $d^{(k)}$ and $d^{(k+1)}$ are conjugate with respect to A. Suppose,

$$d^{(k+1)} = -g_{k+1} + \beta_k d^{(k)}.$$

Multiplying both sides by $d^{(k)}A$, and letting,

$$d^{(k)}Ad^{(k+1)} = -d^{(k)}Ag_{k+1} + \beta_k d^{(k)}Ad^{(k)} = 0,$$

we find the solution of β_k,

$$\beta_k = \frac{d^{(k)}Ag_{k+1}}{d^{(k)}Ad^{(k)}},$$

and thus $d^{(k+1)}$;

4. Set $k := k + 1$, and goto step 2.

Theorem. The iteration will be terminated in $m \leq n$ steps. And for all i $(1 \leq i \leq m)$, the following relations hold:

1. ${d^{(i)}}^T Ad^{(j)} = 0$, for $j = 1, 2, \cdots, i-1$;

2. ${g_i}^T g_j = 0$, for $j = 1, 2, \cdots, i-1$;

3. ${g_i}^T d^{(i)} = -g_i^T g_i$.

7.4 Penalty function method

Consider the minimization problem with constraints,

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } g_i(x) \geq 0, \quad i = 1, \cdots, m, \\ &\qquad h_j(x) = 0, \quad j = 1, \cdots, l. \end{aligned}$$

Exterior penalty function.

For equality constraints, define the auxiliary function,

$$F_1(x, \sigma) = f(x) + \sigma \sum_{j=1}^{l} h_j^2(x)$$

for $\sigma \gg 0$.

For inequality constraints, define the auxiliary function,

$$F_1(x, \sigma) = f(x) + \sigma \sum_{i=1}^{m} [\max\{0, -g_i(x)\}]^2,$$

for $\sigma \gg 0$.

In general, we could define the auxiliary function,

$$F(x, \sigma) = f(x) + \sigma P(x),$$

to transform a constraint optimization into an unconstraint problem. Usually, $\sigma P(x)$ is called the penalty term, σ is the penalty factor, and $F(x, \sigma)$ is the penalty function.

Example 1. Solving the following problem,

$$\begin{aligned} &\min x \\ &\text{s.t. } x - 2 \geq 0. \end{aligned}$$

Example 2. Solving the following nonlinear problem,

$$\begin{aligned} &\min f(x) = (x_1 - 1)^2 + x_2^2 \\ &\text{s.t. } g(x) = x_2 - 1 \geq 0. \end{aligned}$$

Exterior penalty function algorithm:

1. Given initial $x^{(0)}$, penalty factor σ_1, coefficient $c > 1$, error tolerance ε, and set $k = 1$.

2. Given initial $x^{(k-1)}$, solve the unconstraint problem $\min\ f(x) + \sigma_k P(x)$ to find the minimum point $x^{(k)}$

3. If $\sigma_k P(x^{(k)}) < \varepsilon$, stop and $x^{(k)}$ is obtained; or let $\sigma_{k+1} = c\sigma_k$ and set $k := k + 1$, go to step 2.

Convergence.

(1) $F(x^{(k)}, \sigma_k) \leq F(x^{(k+1)}, \sigma_{k+1})$;

(2) $P(x^{(k)}) \geq P(x^{(k+1)})$;

(3) $f(x^{(k)}) \geq f(x^{(k+1)})$.

Interior penalty function. For inequality constraints,

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } g_i(x) \geq 0,\ \ i = 1, \cdots, m \end{aligned}$$

define the feasible domain $S = \{x|g_i(x) \geq 0, i = 1, \cdots, m\}$. To remain the iterative points inside the feasible domain, we define the barrier function,

$$G(x, r) = f(x) + rB(x)$$

where $B(x)$ is continuous, and $B(x) \to +\infty$ when x tends to the domain boundary.

Interior penalty function algorithm:

1. Given initial $x^{(0)} \in \text{int } S$, barrier factor r_1, coefficient $\beta \in (0, 1)$, error tolerance ε, and set $k = 1$.

2. Given initial $x^{(k-1)}$, minimize the barrier function $G(x, r) = f(x) + r_k B(x)$, to find the minimum point $x^{(k)}$

3. If $r_k B(x^{(k)}) < \varepsilon$, stop and $x^{(k)}$ is obtained; or let $r_{k+1} = \beta r_k$ and set $k := k + 1$, go to step 2.

Example. Solving the following nonlinear problem,

$$\begin{array}{l} \min f(x) = \frac{1}{12}(x_1 + 1)^3 + x_2 \\ \text{s.t. } x_1 - 1 \geq 0 \text{ and } x_2 \geq 0. \end{array}$$

7.5 EM optimization

Suppose random variables X produce observed data, and missing or unobserved data are from random variables Z. We expect $Y = (X, Z)$ produce complete data. Given the observed data x, we hope to maximize some likelihood function $L(\theta|x)$. Usually, it will be easier to use the densities of $Y|\theta$ and $Z|(x, \theta)$ for handling the likelihood function.

Suppose $f_X(x|\theta)$ and $f_Y(y|\theta)$ are the densities of the observed and the complete data. The conditional density of the missing data is,

$$f_{Z|X}(z|x, \theta) = f_Y(y|\theta)/f_X(x|\theta).$$

The EM algorithm is to maximize $L(\theta|x)$. Define the joint log-likelihood

expectation for given observed data,

$$\begin{aligned} Q(\theta|\theta^{(t)}) &= E\{\log L(\theta|Y)|x, \theta^{(t)}\} \\ &= E\{\log f_Y(Y|\theta)|x, \theta^{(t)}\} \\ &= \int [\log f_Y(Y|\theta)] f_{Z|X}(z|x, \theta^{(t)}) dz. \end{aligned}$$

EM starts from $\theta^{(0)}$ and alternately performs the expectation and maximization steps.

1. E step: compute $Q(\theta|\theta^{(t)})$;

2. M step: maximize $Q(\theta|\theta^{(t)})$ to find $\theta^{(t+1)}$).