Stochastic methods and algorithms.

Stochastic methods and their algorithms become indispensable in designing iterative/recursive procedures with randomness, and they are often viewed as discrete-time stochastic processes. We explore various ideas of stochastic approach and relate them to stochastic methods. The lecture presentation materials are interspersed with demonstrations in R, and tentative coverage of topics is listed at

`vps63.heliohost.us/e-math/MCMC`

for anyone to review and reproduce them. R is a programming language and environment for statistical methods and visualizations, and can be downloaded at

`cran.r-project.org`

and executed in your own machine.

Code execution in R.

Each “command” in R is executed in an interactive manner, known as “interpretor,” and requested in a form of “function.” For example, “`demo(graphics)`” shows a demonstration of graphics.

```
demo("graphics")
```

Functions can be prepared as external files (usually with extension ".r" or ".R"), and sourced in advance with the command

```
source("[script filename]")
```

Your external files must be found in the working directory and recognized by R. You can always change the working directory from R via [File]→[Change dir...]. Alternatively you can set the working directory by the command

```
setwd("[pathname]")
```

A probabilistic model.

The foundation of probabilistic model involves experiments, events, and probability of an event. A numerical outcome X of an experiment is called a **random variable (r.v.)**. We conventionally denote random variables by uppercase letters $X, Y, Z, U, V, \dots$ from the end of alphabet. A **discrete random variable** is a random variable taking its value on a finite set $\{a_1, a_2, \dots, a_n\}$ of real numbers (usually integers), or on a countably infinite set $\{a_1, a_2, a_3, \dots\}$. The statement such as "$X = a_i$" is an event, and the probability of the event $\{X = a_i\}$ is denoted by $P(X = a_i)$. The function

$$\pi(a_i) = P(X = a_i)$$

over the possible values of X, say $a_1, a_2, \dots$, is called a **frequency function**, or a **probability mass function**.

Continuous random variables.

A **continuous random variable** is a random variable whose possible values are real values such as 78.6, 5.7, 10.24, and so on. An example of continuous random variable is typically a measurement such as temperature, height, diameter of metal cylinder, etc. In what follows, a random variable X always means a "continuous" random variable, unless it is specifically indicated as discrete. Unlike discrete random variables the probability $P(X = t)$ typically vanishes. Instead, how X is distributed over the real numbers is completely characterized by the **cumulative distribution function (cdf)**. The cdf

$$\Pi(t) = P(X \le t).$$

represents the probability that the random variable X is less than or equal to t. Then we often say that "the random variable X is distributed as Π."

Probability density functions (pdf).

It is often the case that the probability of a random variable X being in any particular range of values is given by the area under a curve over that range of values. Such a curve is called a **probability density function (pdf)**, and denoted by $\pi(x)$, sharing the same symbol π with frequency function. Then the probability that "$a \le X \le b$" can be expressed as

$$P(a \le X \le b) = \int_a^b \pi(x)dx.$$

In particular we can find the following relationship between cdf and pdf:

$$\Pi(t) = P(X \le t) = \int_{-\infty}^t \pi(x)dx.$$

Consequently we obtain $\dfrac{d}{dt}\Pi(t) = \pi(t)$.

Joint distributions.

Consider a pair (X, Y) of continuous random variables. A **joint density function** $\pi(x, y)$ is a nonnegative function satisfying

$$P(X \le a, Y \le b) = \int_{-\infty}^{a} \int_{-\infty}^{b} \pi(x, y)\, dy\, dx$$

and it is used to compute probabilities constructed from the random variables X and Y simultaneously. Assuming the **marginal density function**

$$\pi_Y(y) = \int_{-\infty}^{\infty} \pi(x, y)\, dx$$

is strictly positive, we can define the **conditional density function** $\pi(x|y)$ given $Y = y$ by

$$\pi(x|y) = \frac{\pi(x, y)}{\pi_Y(y)}$$

Normal distributions.

A **normal distribution** is represented by a family of distributions which have the same general shape, sometimes described as "bell shaped." The normal density function

$$f(x;\mu,\sigma) := \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \quad -\infty < x < \infty,$$

is determined by two parameters μ and σ and by the exponential function $\exp(u) = e^u$. The parameter μ, called a **mean** (or, location parameter), provides the center of distribution. The parameter σ is a **standard deviation** (or, a scale parameter); small values of σ lead to high peaks but sharp drops. Larger values of σ lead to flatter densities. The shorthand notation

$$X \sim N(\mu, \sigma^2)$$

is often used to express that X is a normal random variable with parameter (μ, σ^2) with **variance** σ^2.

Uniform distributions.

A random variable U is called a **uniform** random variable on $[a, b]$, when U takes any real number in the interval $[a, b]$ equally likely. The pdf of U is given by

$$f(u; a, b) = \begin{cases} 1/(b-a) & \text{if } a \le u \le b; \\ 0 & \text{otherwise [that is, if } u < a \text{ or } b < u]. \end{cases}$$

Then we obtain the cdf $F(t)$ as follows.

$$F(t; a, b) = \int_{-\infty}^{t} f(u)du = \begin{cases} 0 & \text{if } t < a; \\ \frac{t-a}{b-a} & \text{if } a \le t \le b; \\ 1 & \text{if } t > b. \end{cases}$$

Beta distributions.

Let $X_1, \ldots, X_n$ be independent uniform random variables on $[0, 1]$. Then we can consider the k-th order statistic $X_{(k)}$, that is, the k-th smallest value among the observed values of $X_1, \ldots, X_n$. The pdf of the random variable $X_{(k)}$ becomes

$$f_k(x) = \frac{n!}{(k-1)!(n-k)!} x^{k-1}(1-x)^{n-k}, \quad 0 \le x \le 1,$$

which is known as the **beta distribution** with parameters $\alpha_1 = k$ and $\alpha_2 = n - k + 1$. In general, the beta density function

$$f(x; \alpha_1, \alpha_2) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} x^{\alpha_1 - 1}(1-x)^{\alpha_2 - 1}$$

is defined for $\alpha_1 > 0$ and $\alpha_2 > 0$, where we have $\Gamma(n+1) = n!$ for nonnegative integer n.

R code: Plotting density functions.

The function "`dnorm(x,mean,sd)`" returns the normal density value at x, where x can be a single point or a collection of points. Provided x, the graph of the density $f(x)$ is visualized by "`plot(x,f(x))`"

```
x = seq(0, 10, by=0.1)
plot(x, dnorm(x, mean=4.5, sd=1.8), type="l", main="Normal
Density")
```

The function "`dbeta(x,shape1,shape2)`" returns the beta density value, where the parameters α_1 and α_2 correspond to `shape1` and `shape2`.

```
x = seq(0, 1, by=0.02)
plot(x, dbeta(x, shape1=4, shape2=8), type="l", main="Beta
Density")
```

Normal mixture distributions.

Let $f(x;\mu,\sigma)$ be the pdf of normal population distribution with mean μ and standard deviation (SD) σ. Then mixture distributions arise when a population contains distinct subpopulations. Given two normal population distributions with distinct pairs (μ_1,σ_1) and (μ_2,σ_2) of parameters and a weight parameter $0<\gamma<1$, the "normal mixture" density is defined as

$$g(x;\mu_1,\sigma_1,\mu_2,\sigma_2,\gamma)=\gamma f(x;\mu_1,\sigma_1)+(1-\gamma)f(x;\mu_1,\sigma_1)$$

The mixture of two normal distributions will be a bimodal distribution, and the two peaks may be approximated by the respective mean μ_1 and μ_2. Generally given k normal distributions $f(x;\mu_i,\sigma_i)$ with their respective weigth parameters γ_i, $i=1,\ldots,k$, the normal mixture density can be extended to

$$\sum_{i=1}^{k}\gamma_i f(x;\mu_i,\sigma_i)$$

when $\sum_{i=1}^{k}\gamma_i=1$. These mixture distributions exhibit multimodality (i.e., the presence of local maxima), and they become less tractable.

R code: Plotting a normal mixture.

A new function "`nm(x,p)`" is introduced in the external file "`nm.r`" which define the new function "`nm`" as follows.

```
nm = function(x, p=c(0.3, 0.2, 0.8, 0.1, 0.4)){
   p[5] * dnorm(x, p[1], p[2]) + (1-p[5]) * dnorm(x, p[3], p[4])
}
```

It has an array of parameters, `p=` $(\mu_1, \sigma_1, \mu_2, \sigma_2, \gamma)$, and the default value `p=(0.3,0.2,0.8,0.1,0.4)` is used unless alternative value is specified.

```
source("nm.r")
x = seq(0,1,by=0.01)
p = c(0.2, 0.2, 0.7, 0.1, 0.4)
plot(x, nm(x,p), type="l", xlab="x", ylab="PDF",
main="Normal Mixture")
```

Monte Carlo integration.

When the pdf $\pi(x)$ of target distribution is completely known, their statistical characteristics can be obtained in the form of integration

$$E[h(X)] = \int h(x)\pi(x)\,dx$$

such as mean $E[X]$ and variance $E[(X - E[X])^2]$. The integration is interpretted as the **expectation** (**expected value** or **mean**) of the random variable $h(X)$. The objective of Monte Carlo integration is to understand the statistical characteristics of the "target" density function $\pi(x)$ by sampling a random variable X from $\pi(x)$ repeatedly, which allows us to approximate the above integration as an average value of the repeated measurements of $h(X)$.

In practice, Monte Carlo integration begins with simulation by drawing a large number $X_1, X_2, \ldots, X_n$ of random variables and by taking their average

$$M_n = \frac{1}{n} \sum_{i=1}^{n} h(X_i).$$

If $X_1, X_2, \ldots, X_n$ are independent copies of the random variable X from the pdf $\pi(x)$ then the empirical average M_n converges to the expected value

$$\mu_h = E[h(X)] = \int h(x)\pi(x)\,dx$$

Here $h(X_1), h(X_2), \ldots, h(X_n)$ become **independent and identically distributed (iid)**, and the convergence property of the empirical average M_n is known as the **law of large numbers**. It guarantees for large n that the empirical average M_n is very close to the expected value μ_h with high probability.

Normal approximation.

Provided iid sample $X_1, X_2, \dots$ having the common pdf $\pi(x)$, the empirical average

$$M_n = \frac{1}{n} \sum_{i=1}^{n} h(X_i)$$

approximates the expected value μ_h. The **central limit theorem** gives a quantitative statement on how close M_n is to μ_h: If the size n is adequately large then M_n has approximately the normal distribution with mean μ_h and variance σ_h^2/n, where σ_h^2 is the variance of $h(X)$, obtained by

$$\sigma_h^2 = E[(h(X) - \mu_h)^2] = \int (h(x) - \mu_h)^2 \pi(x)\, dx$$

A general rule for "adequately large" n is $n \geq 100$, but it is often good for much smaller n.

Confidence interval.

The value $\sigma_h/\sqrt{n}$ is called the **Monte Carlo standard error**, where σ_h^2 itself is estimated by the Monte Carlo integration

$$\hat{\sigma}_h^2 = \frac{1}{n-1}\sum_{i=1}^{n}[h(X_i) - M_n]^2$$

Together with the central limit theorem we obtain the $(1-\alpha)\%$ confidence interval for μ_h by

$$\left(M_n - z_{\alpha/2}\frac{\hat{\sigma}_h}{\sqrt{n}},\ M_n + z_{\alpha/2}\frac{\hat{\sigma}_h}{\sqrt{n}},\ \right),$$

where z_α denotes the critical point for standard normal distribution satisfying $P(Z > z_\alpha) = \alpha$ with standard normal random variable Z. Technically the critical point should be obtained from t-distribution with $(n-1)$ degrees of freedom, but the effect is negligible in practice.

Sampling algorithm via probability inverse transform.

If $\pi(x)$ is the density function on the real line $\mathbb{R}$, we can introduce the cumulative distribution function (cdf)

$$\Pi(z) = \int_{-\infty}^{z} \pi(x)\, dx = P(X \leq z)$$

Furthermore, we can define the **quantile function** by

$$\Pi^{-1}(u) := \sup\{z : \Pi(z) < u\}$$

It is an inverse function in the sense that

$$\Pi^{-1}(u) \leq z \text{ if and only if } u \leq \Pi(z),$$

where the equality holds when Π^{-1} is continuous at z.

Sampling Algorithm:

1. Generate an uniform random variable $\mathbf{U}$ on $(0, 1]$.
2. Return the value $X = \Pi^{-1}(\mathbf{U})$.

It generates a random variable X distributed as the pdf $\pi(x)$.

Rejection sampling algorithm.

Suppose that $\pi(x)$ is the density function on a general space (e.g., $\mathbb{R}^n$), and that there is a way to sample from a different density $\rho(x)$. If there is some positive value c satisfying

$$\pi(x) \leq c\rho(x) \quad \text{for all } x,$$

then we can also sample from the target density $\pi(x)$.

REJECTION SAMPLING ALGORITHM:

1. Generate a random variable X from $\rho(x)$.
2. Generate an uniform random variable $\mathbf{U}$ on $[0, 1)$.
3. Accept X if $\mathbf{U} \leq \dfrac{\pi(X)}{c\rho(X)}$; otherwise, reject it.

It returns the value X only if it accepts, and consequently, the returned value X is distributed as π.

How does it work?

For the sake of simplicity we assume that X is discrete. Then the probability that the algorithm successfully returns $X = x$ is given by

$$\begin{aligned} P\left(X = x, U \leq \frac{\pi(x)}{c\rho(x)}\right) &= P\left(U \leq \frac{\pi(x)}{c\rho(x)} \,\middle|\, X = x\right) P(X = x) \\ &= \left(\frac{\pi(x)}{c\rho(x)}\right) \rho(x) = \frac{\pi(x)}{c} \end{aligned}$$

Thus, the probability of acceptance becomes

$$P\left(U \leq \frac{\pi(X)}{c\rho(X)}\right) = \frac{1}{c}$$

Therefore, having accepted $X = x$ in the algorithm it is distributed as

$$P\left(X = x \,\middle|\, U \leq \frac{\pi(X)}{c\rho(X)}\right) = \pi(x)$$

Search algorithm.

Consider a nonnegative objective function $f(x)$ on the interval $[a, b]$, and the optimization problem to find

$$x^* = \text{argmax}\{f(x) : a \leq x \leq b\}$$

Our search algorithm employs a random propagation strategy.

Search Algorithm:

1. Choose the initial state X_0 at time $t = 0$.
2. Suppose that $X_t = x$ at time t. Then pick the next move Δx randomly and set $y = x + \Delta x$.
3. If $a \leq y \leq b$ and $f(y) > f(x)$ then set $X_{t+1} = y$.
4. Otherwise, stay $X_{t+1} = x$ at time $t + 1$.

The algorithm will not necessarily find the global maximum, but may instead converge to a local maximum.

Random ascending Search algorithm.

In general for multimodal functions it is common to reach a local maximum instead of global maximum, which is heavily dependent on the initial state x_0. Here we can relax a strict ascent criteria and allow some moves to descend in addition to the randomness for the next move Δx.

Random Ascending Search Algorithm:

1. Choose the initial state X_0 at time $t = 0$.
2. Suppose that $X_t = x$ at time t. Then pick the next move Δx randomly and set $y = x + \Delta x$.
3. Generate a uniform random variable U on $[0, 1]$
4. If $f(y)/f(x) > U$ then set $X_{t+1} = y$.
5. Otherwise, stay $X_{t+1} = x$ at time $t + 1$.

A random ascension can avoid getting stuck in a local maximum. But what are stopping criteria for the algorithm?

R code: Running search algorithm.

Consider a normal mixture density function as the objective function of choice. Download R source files “nm.r” and “search.r” into your own machine, and see how the search algorithm seeks the global maximum.

- Choose a different standard deviation σ for random steps $\Delta x \sim N(0, \sigma^2)$. The value “`sigma=0.05`” is given by default.
- Use a different parameter $p = (\mu_1, \sigma_1, \mu_2, \sigma_2, \gamma)$ for the objective function.

```
source("nm.r")
source("search.r")
search(ff=nm, x0=0, sigma=0.05, run.time=500, random=F,
p=c(0.2, 0.2, 0.7, 0.1, 0.4))
```

R code: Random trajectories of search algorithm.

See how the evolution of search produces different trajectories.

- Save the trajectory data from $t = 0$, and plot the trajectory of moves in time series.
- Run with (`random=T`) or without (`random=F`) random ascending strategy.
- Choose a different initial state x_0 or a different running time N.

```
par(mfrow=c(2,1))
N = 500
xx = search(ff=nm, x0=0, run.time=N, random=T,
p=c(0.2, 0.2, 0.7, 0.1, 0.4), save.time=0)
plot(0:N, xx, type="l", main="Trajectory", xlab="time",
ylab="state")
```

R code: A long run behavior.

See a long run behavior of search moves, and investigate how they are distributed.

- Save the trajectory data from $t = 1000$ (`save.time=1000`) until $N = 2000$ (`run.time=2000`).
- Construct the histogram of data, and use a different parameter $p = (\mu_1, \sigma_1, \mu_2, \sigma_2, \gamma)$.

```
par(mfrow=c(2,1))
xx = search(ff=nm, x0=0, run.time=2000, save.time=1000,
random=T, p=c(0.2, 0.2, 0.7, 0.1, 0.4))
hist(xx, freq=F, breaks=seq(0,1,by=0.05), col="blue",
main="Trajectory", xlab="state")
```

Activities and report questions.

1. Install R and produce beta distributions for $(\alpha_1, \alpha_2) = (1, 10)$, $(5, 6)$ and $(10, 1)$. Describe the shape of each distribution. What does each distribution represent? Explain differences among them.
2. In the sampling algorithm via probability inverse transform prove that $P(\Pi^{-1}(\mathbf{U}) \leq z) = \Pi(z)$. Then conclude that the algorithm generates a sample X exactly distributed as Π.
3. In the rejection sampling algorithm we continue to assume that X is discrete. Observe that

$$P\left(X = x, U \leq \frac{\pi(X)}{c\rho(X)}\right) = P\left(X = x, U \leq \frac{\pi(x)}{c\rho(x)}\right)$$

   and show the detailed derivation of

$$P(X = x | X \text{ is accepted}) = \pi(x)$$

   Then conclude that the algorithm generates a sample X exactly distributed as π when X is accepted.