

Monte Carlo Strategies In Scientific Computing

Jun S. Liu
Department of Statistics
Harvard University

March 20, 2002

To my wife Wei

Preface

An early experiment that conceives the basic idea of Monte Carlo computation is known as “Buffon’s needle” (Dörrie 1965), first stated by Georges Louis Leclerc Comte de Buffon in 1777. In this well-known experiment, one throws a needle of length l onto a flat surface with a grid of parallel lines with spacing D ($D > l$). It is easy to compute that, under ideal conditions, the chance that the needle will intersect one of the lines is $2l/\pi D$. Thus, if we let p_N be the proportion of “intersects” in N throws, we can have an estimate of π as

$$\hat{\pi} = \lim_{N \rightarrow \infty} \frac{2l}{p_N D},$$

which will “converge” to π as N increases to infinity. Numerous investigators actually used this setting to estimate π . The idea of simulating random processes so as to help evaluate certain quantities of interest is now an essential part of scientific computing.

A systematic use of the Monte Carlo method for real scientific problems appeared in the early days of electronic computing (1945-55) and accompanied the development of the world’s first programmable “super” computer, MANIAC (Mathematical Analyzer, Numerical Integrator and Computer), at Los Alamos during World War II. In order to make a good use of these fast computing machines, scientists (Stanislaw Ulam, John von Neumann, Nicholas Metropolis, Enrico Fermi, etc.) invented a statistical sampling-based method for solving numerical problems concerning random neutron diffusion in fissile material in atomic bomb designs and for estimating eigenvalues of the Schrödinger equation. The basic idea underlying the method was first brought up by Ulam and deliberated between him

and von Neumann in a car when they drove together from Los Alamos to Lamy. Allegedly, Nick Metropolis coined the name “Monte Carlo,” which played an essential role in popularizing the method.

In the early 1950s, statistical physicists (N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller) introduced a Markov-chain-based dynamic Monte Carlo method for the simulation of simple fluids. This method was later extended to cover more and more complex physical systems, including spin glass models, harmonic crystal, polymer models, etc. In the 1980s, statisticians and computer scientists developed Monte-Carlo-based methods for a wide variety of tasks such as combinatorial optimizations, nonparametric statistical inference (e.g., jackknife and bootstrap), likelihood computation with missing observations, statistical genetics analysis, Bayesian modeling and computations, and others. In the 1990s, the method began to play an important role in computational biology and was used to solve problems in sequence motif identification and the analysis of complex pedigree. Now, the list of application areas of Monte Carlo methods includes biology (Leach 1996, Karplus and Petsko 1990, Lawrence, Altschul, Boguski, Liu, Neuwald and Wootton 1993), chemistry (Alder and Wainwright 1959), computer science (Kirkpatrick, Gelatt and Vecchi 1983), economics and finance (Gouriéroux and Monfort 1997); engineering (Geman and Geman 1984), material science (Frenkel and Smit 1996), physics (Metropolis, Rosenbluth, Rosenbluth, Teller and Teller 1953, Goodman and Sokal 1989, Marinari and Parisi 1992), statistics (Efron 1979, Gelfand and Smith 1990, Rubin 1987, Tanner and Wong 1987), and many others. Among all Monte Carlo methods, *Markov chain Monte Carlo* (MCMC) provides an enormous scope for dealing with very complicated stochastic systems and has been the central pillar in the study of macromolecules and other physical systems. Recently, the MCMC methodology has drawn much attention from statisticians because the method enables them to entertain more sophisticated and realistic statistical models.

Being attracted by the extreme flexibility and power of the Monte Carlo method, many researchers in different scientific areas have contributed to its development. However, because a substantial amount of domain-specific knowledge is required in order to understand problems in any of these fields, communications among researchers in these fields are very limited. Many efforts have been devoted to the reinvention of techniques that have been developed in other fields. It is therefore desirable to develop a relatively general framework in which scientists in every field — e.g., theoretical chemists, statistical physicists, structural biologists, statisticians, econometricians, and computer scientists — can compare their Monte Carlo techniques and learn from each other. For a large number of scientists and engineers who employ Monte Carlo simulation and related global optimization techniques (such as simulated annealing) as an essential tool in their work, there is also a need to keep up to date with recent advances in Monte Carlo methodologies and to understand the nature and connection of various proposed

methods. The aim of this book is to provide a self-contained, unified, and up-to-date treatment of the Monte Carlo method.

This book is intended to serve three audiences: researchers specializing in the study of Monte Carlo algorithms; scientists who are interested in using advanced Monte Carlo techniques; and graduate students in statistics, computational biology, and computer sciences who want to learn about Monte Carlo computations. The prerequisites for understanding most of the methods described in this book are rather minimal: a one-semester course on probability theory (Pitman 1993) and a one-semester course on theoretical statistics (Rice 1994), both at the undergraduate level. However, it would be more desirable if the reader has some background in a specific scientific field such as artificial intelligence, computational biology, computer vision, engineering, or Bayesian statistics in which heavy computations are involved. This book is most suitable for a second-year graduate-level course on Monte Carlo methods, with an emphasis on their relevance to scientific and statistical research.

The author is most grateful to his mentor and friend Wing Hung Wong for his many important suggestions, his overwhelming passion for Monte Carlo and scientific problems, and his continuous encouragement. The author is also grateful to Persi Diaconis for teaching him many things including Markov chain theory, group theory, and nonparametric Bayes methods, to both Susan Holmes and Persi for numerous enlightening conversations on Markov chain Monte Carlo and other related problems, to Donald B. Rubin for insights on the missing data formulation and the Bayesian thinking, to Jonathan Goodman for helpful comments on multigrid Monte Carlo, to Yingnian Wu and Songchun Zhu for their materials on pattern simulations and thoughts on conditional sampling, to Faming Liang for his supply of many examples and figures, and to Minghui Chen and David van Dyk for helpful comments. Several former graduate students in the statistics departments of Stanford and Harvard universities — Yuguo Chen, Lingyu Chen, Chiara Sabatti, Tanya Logvinenko, Zhaohui Qin and Juni Zhang — have contributed in many ways to the development of this book. Ms. Helen Tombropoulos has provided editorial assistance to the author both for this book and for many articles published earlier. Finally, the author is greatly indebted to his wife Wei for her love and her continuous support of his research activities these years. Part of the book was written when the author was on the faculty of the Statistics Department of Stanford University. This work was also partially supported by the National Science Foundation Grants DMS-9803649 and DMS-0094613.

Cambridge, Massachusetts
March 2001

Jun Liu

Contents

Preface	vii
1 Introduction and Examples	1
1.1 The Need of Monte Carlo Techniques	1
1.2 Scope and Outline of the Book	3
1.3 Computations in Statistical Physics	7
1.4 Molecular Structure Simulation	9
1.5 Bioinformatics: Finding Weak Repetitive Patterns	10
1.6 Nonlinear Dynamic System: Target Tracking	14
1.7 Hypothesis Testing for Astronomical Observations	16
1.8 Bayesian Inference of Multilevel Models	18
1.9 Monte Carlo and Missing Data Problems	19
2 Basic Principles: Rejection, Weighting, and Others	23
2.1 Generating Simple Random Variables	23
2.2 The Rejection Method	24
2.3 Variance Reduction Methods	25
2.4 Exact Methods for Chain-Structured Models	28
2.4.1 Dynamic programming	29
2.4.2 Exact simulation	30
2.5 Importance Sampling and Weighted Sample	31
2.5.1 An example	31
2.5.2 The basic idea	32
2.5.3 The “rule of thumb” for importance sampling	34

2.5.4	Concept of the weighted sample	36
2.5.5	Marginalization in importance sampling	37
2.5.6	Example: Solving a linear system	38
2.5.7	Example: A Bayesian missing data problem	40
2.6	Advanced Importance Sampling Techniques	42
2.6.1	Adaptive importance sampling	42
2.6.2	Rejection and weighting	43
2.6.3	Sequential importance sampling	46
2.6.4	Rejection control in sequential importance sampling	48
2.7	Application of SIS in Population Genetics	48
2.8	Problems	52
3	Theory of Sequential Monte Carlo	53
3.1	Early Developments: Growing a Polymer	55
3.1.1	A simple model of polymer: Self-avoid walk	55
3.1.2	Growing a polymer on the square lattice	56
3.1.3	Limitations of the growth method	59
3.2	Sequential Imputation for Statistical Missing Data Problems	60
3.2.1	Likelihood computation	60
3.2.2	Bayesian computation	62
3.3	Nonlinear Filtering	64
3.4	A General Framework	67
3.4.1	The choice of the sampling distribution	69
3.4.2	Normalizing constant	69
3.4.3	Pruning, enrichment, and resampling	71
3.4.4	More about resampling	72
3.4.5	Partial rejection control	75
3.4.6	Marginalization, look-ahead, and delayed estimate	76
3.5	Problems	76
4	Sequential Monte Carlo in Action	79
4.1	Some Biological Problems	79
4.1.1	Molecular Simulation	79
4.1.2	Inference in population genetics	81
4.1.3	Finding motif patterns in DNA sequences	84
4.2	Approximating Permanents	90
4.3	Counting 0-1 Tables with Fixed Margins	92
4.4	Bayesian Missing Data Problems	94
4.4.1	Murray's data	94
4.4.2	Nonparametric Bayes analysis of binomial data	95
4.5	Problems in Signal Processing	98
4.5.1	Target tracking in clutter and mixture Kalman filter	98
4.5.2	Digital signal extraction in fading channels	102
4.6	Problems	103

5	Metropolis Algorithm and Beyond	105
5.1	The Metropolis Algorithm	106
5.2	Mathematical Formulation and Hastings's Generalization	111
5.3	Why Does the Metropolis Algorithm Work?	112
5.4	Some Special Algorithms	114
5.4.1	Random-walk Metropolis	114
5.4.2	Metropolized independence sampler	115
5.4.3	Configurational bias Monte Carlo	116
5.5	Multipoint Metropolis Methods	117
5.5.1	Multiple independent proposals	118
5.5.2	Correlated multipoint proposals	120
5.6	Reversible Jumping Rule	122
5.7	Dynamic Weighting	124
5.8	Output Analysis and Algorithm Efficiency	125
5.9	Problems	127
6	The Gibbs Sampler	129
6.1	Gibbs Sampling Algorithms	129
6.2	Illustrative Examples	131
6.3	Some Special Samplers	133
6.3.1	Slice sampler	133
6.3.2	Metropolized Gibbs sampler	133
6.3.3	Hit-and-run algorithm	134
6.4	Data Augmentation Algorithm	135
6.4.1	Bayesian missing data problem	135
6.4.2	The original DA algorithm	136
6.4.3	Connection with the Gibbs sampler	137
6.4.4	An example: Hierarchical Bayes model	138
6.5	Finding Repetitive Motifs in Biological Sequences	139
6.5.1	A Gibbs sampler for detecting subtle motifs	140
6.5.2	Alignment and classification	141
6.6	Covariance Structures of the Gibbs Sampler	143
6.6.1	Data Augmentation	143
6.6.2	Autocovariances for the random-scan Gibbs sampler	144
6.6.3	More efficient use of Monte Carlo samples	146
6.7	Collapsing and Grouping in a Gibbs Sampler	146
6.8	Problems	151
7	Cluster Algorithms for the Ising Model	153
7.1	Ising and Potts Model Revisit	153
7.2	The Swendsen-Wang Algorithm as Data Augmentation	154
7.3	Convergence Analysis and Generalization	155
7.4	The Modification by Wolff	157
7.5	Further Generalization	157
7.6	Discussion	158

7.7	Problems	159
8	General Conditional Sampling	161
8.1	Partial Resampling	161
8.2	Case Studies for Partial Resampling	163
8.2.1	Gaussian random field model	163
8.2.2	Texture synthesis	165
8.2.3	Inference with multivariate t-distribution	168
8.3	Transformation Group and Generalized Gibbs	170
8.4	Application: Parameter Expansion for Data Augmentation	174
8.5	Some Examples in Bayesian Inference	176
8.5.1	Probit regression	176
8.5.2	Monte Carlo bridging for stochastic differential equation	178
8.6	Problems	179
9	Molecular Dynamics and Hybrid Monte Carlo	183
9.1	Basics of Newtonian Mechanics	184
9.2	Molecular Dynamics Simulation	185
9.3	Hybrid Monte Carlo	189
9.4	Algorithms Related to HMC	192
9.4.1	Langevin-Euler moves	192
9.4.2	Generalized hybrid Monte Carlo	193
9.4.3	Surrogate transition method	194
9.5	Multipoint Strategies for Hybrid Monte Carlo	195
9.5.1	Neal's window method	195
9.5.2	Multipoint method	196
9.6	Application of HMC in Statistics	198
9.6.1	Indirect observation model	198
9.6.2	Estimation in the stochastic volatility model	201
10	Multilevel Sampling and Optimization Methods	205
10.1	Umbrella Sampling	206
10.2	Simulated Annealing	209
10.3	Simulated Tempering	210
10.4	Parallel Tempering	212
10.5	Generalized Ensemble Simulation	215
10.5.1	Multicanonical sampling	216
10.5.2	The $1/k$ -ensemble method	217
10.5.3	Comparison of algorithms	218
10.6	Tempering with Dynamic Weighting	219
10.6.1	Ising model simulation at sub-critical temperature	221
10.6.2	Neural network training	221
11	Population-Based Monte Carlo Methods	225

11.1	Adaptive Direction Sampling: Snooker Algorithm	226
11.2	Conjugate Gradient Monte Carlo	227
11.3	Evolutionary Monte Carlo	228
11.3.1	Evolutionary movements in binary-coded space	230
11.3.2	Evolutionary movements in continuous space	231
11.4	Some Further Thoughts	233
11.5	Numerical Examples	235
11.5.1	Simulating from a bimodal distribution	235
11.5.2	Comparing algorithms for a multimodal example	236
11.5.3	Variable selection with binary-coded EMC	238
11.5.4	Bayesian neural network training	240
11.6	Problems	242
12	Markov Chains and Their Convergence	245
12.1	Basic Properties of a Markov Chain	245
12.1.1	Chapman-Kolmogorov equation	247
12.1.2	Convergence to stationarity	248
12.2	Coupling Method for Card Shuffling	250
12.2.1	Random-to-top shuffling	250
12.2.2	Riffle shuffling	251
12.3	Convergence Theorem for Finite-State Markov Chains	252
12.4	Coupling Method for General Markov Chain	254
12.5	Geometric Inequalities	256
12.5.1	Basic setup	256
12.5.2	Poincaré inequality	257
12.5.3	Example: Simple random walk on a graph	259
12.5.4	Cheeger's inequality	260
12.6	Functional Analysis for Markov Chains	263
12.6.1	Forward and backward operators	263
12.6.2	Convergence rate of Markov chains	265
12.6.3	Maximal correlation	266
12.7	Behavior of the Averages	268
13	Selected Theoretical Topics	271
13.1	MCMC Convergence and Convergence Diagnostics	271
13.2	Iterative Conditional Sampling	273
13.2.1	Data augmentation	273
13.2.2	Random-scan Gibbs sampler	275
13.3	Comparison of Metropolis-Type Algorithms	277
13.3.1	Peskun's ordering	277
13.3.2	Comparing schemes using Peskun's ordering	279
13.4	Eigenvalue Analysis for the Independence Sampler	281
13.5	Perfect Simulation	284
13.6	A Theory for Dynamic Weighting	287
13.6.1	Definitions	287

13.6.2	Weight behavior under different scenarios	288
13.6.3	Estimation with weighted samples	291
13.6.4	A simulation study	292
A	Basics in Probability and Statistics	295
A.1	Basic Probability Theory	295
A.1.1	Experiments, events, and probability	295
A.1.2	Univariate random variables and their properties	296
A.1.3	Multivariate random variable	298
A.1.4	Convergence of random variables	299
A.2	Statistical Modeling and Inference	301
A.2.1	Parametric statistical modeling	301
A.2.2	Frequentist approach to statistical inference	302
A.2.3	Bayesian methodology	304
A.3	Bayes Procedure and Missing Data Formalism	306
A.3.1	The joint and posterior distributions	306
A.3.2	The missing data problem	307
A.4	The Expectation-Maximization Algorithm	309
	References	313
	Author Index	333
	Subject Index	338

1

Introduction and Examples

1.1 The Need of Monte Carlo Techniques

An essential part of many scientific problems is the computation of integral

$$I = \int_D g(\mathbf{x}) d\mathbf{x},$$

where D is often a region in a high-dimensional space and $g(\mathbf{x})$ is the target function of interest. If we can draw independent and identically distributed (i.i.d.) random samples $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}$ uniformly from D (by a computer), an approximation to I can be obtained as

$$\hat{I}_m = \frac{1}{m} \{g(\mathbf{x}^{(1)}) + \dots + g(\mathbf{x}^{(m)})\}.$$

The *law of large numbers* states that the average of many independent random variables with common mean and finite variances tends to stabilize at their common mean (see the Appendix); that is,

$$\lim_{m \rightarrow \infty} \hat{I}_m = I, \quad \text{with probability 1.}$$

Its convergence rate can be assessed by the *central limit theorem* (CLT):

$$\sqrt{m}(\hat{I}_m - I) \rightarrow N(0, \sigma^2), \quad \text{in distribution,}$$

where $\sigma^2 = \text{var}\{g(\mathbf{x})\}$. Hence, the “error term” of this Monte Carlo approximation is $O(m^{-1/2})$, regardless of the dimensionality of \mathbf{x} . This basic

setting underlies the potential role of the Monte Carlo methodology in science and statistics.

In the simplest case when $D = [0, 1]$ and $I = \int_0^1 g(x)dx$, one can approximate I by

$$\tilde{I}_m = \frac{1}{m} \{g(b_1) + \cdots + g(b_m)\},$$

where $b_j = j/m$. This method can be called the *Riemann approximation*. When g is reasonably smooth, the Riemann approximation gives us an error rate of $O(m^{-1})$, better than that of the Monte Carlo method. More sophisticated methods such as Simpson's rule and the Newton-Cotes rules give better numerical approximations (Thisted 1988). However, a fatal defect of these deterministic methods is that they do not scale well as the dimensionality of D increases. For example, in a 10-dimensional space with $D = [0, 1]^{10}$, we will have to evaluate $O(m^{10})$ grid points in order to achieve an accuracy of $O(m^{-1})$ in the Riemann approximation of I . In contrast, the naive Monte Carlo approach, which draws $x^{(1)}, \dots, x^{(m)}$ uniformly from D , has an error rate $O(m^{-1/2})$ regardless of the dimensionality of D , at least theoretically.

Although the "error rate" of a Monte Carlo integration scheme remains the same in high-dimensional problems, two intrinsic difficulties arise: (a) when the region D is large in high-dimensional space, the variance σ^2 , which measures how "uniform" the function g is in region D , can be formidably large; (b) one may not be able to produce uniform random samples in an arbitrary region D . To overcome these difficulties, researchers often employ the idea of *importance sampling* in which one generates random samples $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}$ from a nonuniform distribution $\pi(\mathbf{x})$ that puts more probability mass on "important" parts of the state space D . One can estimate integral I as

$$\hat{I} = \frac{1}{m} \sum_{j=1}^m \frac{g(\mathbf{x}^{(j)})}{\pi(\mathbf{x}^{(j)})},$$

which has a variance $\sigma_\pi^2 = \text{var}_\pi \{g(\mathbf{x})/\pi(\mathbf{x})\}$. In the most fortunate case, we may choose $\pi(\mathbf{x}) \propto g(\mathbf{x})$ when g is non-negative and I is finite, which results in an exact estimate of I . But in no known application of the Monte Carlo method has this "luckiest situation" ever occurred. More realistically, we may hope to find a good "candidate" π which will explore more in regions where the value of g is high. In such a situation, generating random draws from π can be a challenging problem.

Demands for sampling from a nonuniform distribution π are also seen from another set of problems in bioinformatics, computational chemistry, physics, structural biology, statistics, etc. In these problems, the desired probability distribution $\pi(\mathbf{x})$ of a complex system, where \mathbf{x} is often called a *configuration* of the system, arises from basic laws in physics and statistical inference. For example, in the study of a macromolecule, \mathbf{x} may represent the *structure* of a molecule in the form of three-dimensional coordinates

of all the atoms in the molecule. The target probability distribution is defined by the Boltzmann distribution $\pi(\mathbf{x}) = Z(T)e^{-h(\mathbf{x})/kT}$, where k is the Boltzmann constant, T is the system's temperature, $h(\mathbf{x})$ is the energy function, and $Z(T)$ is the *partition function* which is difficult to compute. Scientists are often interested in certain “average characteristics” of the system, many of which can be expressed mathematically as $E_\pi[g(\mathbf{x})]$ for a suitable function g . In Bayesian statistical inference, \mathbf{x} often represents the joint configuration of missing data and parameter values and $\pi(\mathbf{x})$ is usually the posterior distribution of these variables. One has to integrate out nuisance parameters and the missing data so as to make a proper inference on the parameter of interest and to make valid predictions for future observations. These tasks can, once again, be expressed as computing the expectation of a function of the configuration space.

Sometimes, an optimization problem can also be formulated as a Monte Carlo sampling problem. Suppose we are interested in finding the minimum of a target function, $h(\mathbf{x})$, defined on a possibly complex configuration space. The problem is equivalent to finding the maximum of another function, $q_T(\mathbf{x}) = e^{-h(\mathbf{x})/T}$ (as long as $T > 0$). In the case when $q_T(\mathbf{x})$ is integrable for all $T > 0$, which is most common in practice, we can make up a family of probability distributions:

$$\pi_T(\mathbf{x}) \propto e^{-h(\mathbf{x})/T}, \quad T > 0.$$

If we can sample from $\pi_T(\mathbf{x})$ when T is sufficiently small, resulting random draws will most likely be located in the vicinity of the global minimum of $h(\mathbf{x})$. This consideration is the basis of the well-known simulated annealing algorithm (Kirkpatrick et al. 1983) and is also key to the *tempering* techniques for designing more efficient Monte Carlo algorithms (Chapter 10).

1.2 Scope and Outline of the Book

A fundamental step in all Monte Carlo methods is to generate (pseudo-) random samples from a probability distribution function $\pi(\mathbf{x})$, often known only up to a normalizing constant. The variable of interest \mathbf{x} usually takes value in \mathbb{R}^k , but occasionally can take value in other spaces such as a permutation or transformation group (Diaconis 1988, Liu and Wu 1999). In most applications, directly generating independent samples from the target distribution π is not feasible. It is often the case that either the generated samples have to be dependent or the distribution used to generate the samples is different from π , or both. The rejection method (von Neumann 1951), importance sampling (Marshall 1956), and sampling-importance-resampling (SIR) (Rubin 1987) are schemes that make use of samples generated from a *trial distribution* $p(\mathbf{x})$, which differs from, but should be similar to, the target distribution π . The Metropolis algorithm (Metropolis et al. 1953)

which, together with Hastings's (1970) generalizations, serves as the basic building block of Markov chain Monte Carlo (MCMC), is the one that generates dependent samples from a Markov chain with π as its equilibrium distribution. In other words, MCMC is essentially a Monte Carlo integration procedure in which the random samples are produced by evolving a Markov chain.

Because of the great potential of Monte Carlo methodology, various techniques have been developed by researchers in their respective fields. Recent advances in Monte Carlo techniques include the cluster method, data augmentation, parameter expansion, multicanonical sampling, multigrid Monte Carlo (MGMC), umbrella sampling, density-scaling Monte Carlo, simulated tempering, parallel tempering, hybrid Monte Carlo (HMC), multiple try Metropolis (MTM), sequential Monte Carlo, particle filtering, etc. There is also a trend in moving toward a population-based approach. These advances in one way or another were all motivated by the need to sample from very complex probability distributions for which the standard Metropolis method tends to be trapped in a local "energy" well. Many of these methods are related, and some are even identical. For example, the configurational bias Monte Carlo (Siepmann and Frenkel 1992) is equivalent to a sequential importance sampling combined with a Metropolized independence sampler (Chapters 2 & 3); the exchange Monte Carlo (Hukushima and Nemoto 1996) is reminiscent of *parallel tempering* (Geyer 1991); the multiple-try Metropolis (Liu, Liang and Wong 2000) generalizes a method described by Frenkel and Smit (1996); the parameter expansion (Liu and Wu 1999) recently developed is a special case of the *partial resampling* technique (Goodman and Sokal 1989); and the bootstrap filter and sequential imputation (Gordon, Salmond and Smith 1993, Kong, Liu and Wong 1994) can be traced back to the "growth method" (Hammersley and Morton 1954, Rosenbluth and Rosenbluth 1955). By providing a systematic account of these methods, this book focuses on the following aspects: understanding the properties and characteristics of these methods, revealing their connections and differences, comparing their performances and proposing generalizations, and demonstrating their use in scientific and statistical problems.

The remaining part of this chapter presents motivating examples in statistical physics, molecular simulation, bioinformatics, dynamic system analysis, statistical hypothesis testing, Bayesian inference for hierarchical models, and other statistical missing data problems.

Chapter 2 covers basic Monte Carlo techniques including the inversion method, rejection sampling, antithetic sampling, control variate method, stratified sampling, importance sampling, and the exact sampling method for chain-structured models. The last method is usually not covered by the standard Monte Carlo books but is becoming increasingly important in modern statistical analysis, artificial intelligence, and computational biology. Special attention is given to methods related to importance sampling

(e.g., that for solving linear equations, for phylogenetic analysis, and for Bayesian inference with missing data).

Chapter 3 explains in detail the origin and the theoretical framework of sequential Monte Carlo. Started with the Monte Carlo treatment of a self-avoid random walk by Morton, Hammersley, Rosenbluth, and Rosenbluth, in the 1950s, this chapter shows the reader an important common structure in seemingly unrelated problems, such as polymer simulation and Bayesian missing data problems. A general methodology built upon sequential importance sampling, resampling (or pruning and enrichment), and rejection sampling is described to generalize the methods used in the polymer simulation and Bayesian missing data problems. Chapter 4 illustrates how sequential Monte Carlo methods can be used in different problems such as molecular simulation, population genetics, computational biology, non-parametric Bayes analysis, approximating permanents, target tracking, and digital communications.

The later chapters focus primarily on Markov chain based dynamic Monte Carlo strategies. Chapter 5 introduces the basic building block of almost all Markov chain Monte Carlo strategies — the Metropolis-Hastings transition rule. A few recent generalizations of the rule, such as the multipoint rule, the reversible jumping rule, and the dynamic weighting rule are described so that the reader can be equipped with a full array of basic tools in designing a MCMC sampler. The basic method for analyzing efficiency of a MCMC algorithm is described at the end of this chapter.

Chapters 6-8 analyze and generalize another main class of Monte Carlo Markov chains — those built upon iterative sampling from conditional distributions. A prominent special case is the Gibbs sampler, which is now a standard tool for statistical computing. Data augmentation, which was originally designed for solving statistical missing data problems, is recasted as a strategy for improving the ease of computation and the convergence speed of a MCMC sampler. The cluster algorithm for the Ising model (Swendsen and Wang 1987) is treated under this unified view. Another important generalization is the view of partial resampling (Goodman and Sokal 1989), which can be used to conduct more global moves based on a transformation group formulation (Liu and Sabatti 2000, Liu and Wu 1999). The analytical form of the required conditional distributions in this general setting, as well as its applications in Gaussian random field, texture modeling, probit regression, and stochastic differential equations, are given in Chapter 8.

Starting with the basic Newtonian mechanics, Chapter 9 introduces the method of hybrid Monte Carlo (HMC), a means to construct Markov chain moves by evolving Hamiltonian equations. This method reveals the close connection between Monte Carlo and molecular dynamics algorithms, the latter having been one of the most widely used tools in structural biology and theoretical chemistry. A few strategies for improving the efficiency of an HMC or MC algorithm are discussed; these include the surrogate transition method, the window method, and the multipoint method. We also want to

draw the reader's attention to unconventional uses of HMC, especially its application in statistical problems.

Chapters 10 and 11 discuss a few recent developments for efficient Monte Carlo sampling. Incidentally, many of these new methods rely on the idea of running multiple Monte Carlo Markov chains in parallel. Mechanisms that enable communications among the multiple chains are incorporated into the samplers so as to speed up their exploration of the configuration space. These techniques can be grouped into three main classes: temperature-based methods (simulated tempering, parallel tempering, and simulated annealing), reweighting methods (umbrella sampling, multicanonical sampling, and $1/k$ -ensemble method), and evolution-based methods (adaptive direction sampling, conjugate gradient Monte Carlo, and evolutionary Monte Carlo). Some of these approaches can be combined so as to produce a more efficient sampler. For many important scientific problems, these new techniques are indispensable.

Chapter 12 provides a basic theory for the general Markov chain and a few analytical techniques for studying its convergence rate. The basic theory includes the Chapman-Kolmogorov equation and the geometric convergence theorem for finite-state Markov chains. For advanced techniques, we show how to use the coupling method, the Poincaré inequalities, and Cheeger's inequality to bound the second largest eigenvalue of the Markov chain transition matrix and how to use the basic functional analysis tools to get a qualitative understanding of the Markov chain's convergence.

Chapter 13 selects a few theoretical topics in the analysis of Monte Carlo Markov chain. The chapter starts with short discussion of the general convergence issue in MCMC sampling and proceeds to an analysis of the covariance structures of data augmentation and the random-scan Gibbs sampler, showing that these structures can be used to gain insight into the design of an efficient Gibbs sampler. Peskun's theorem is described and used to compare different Metropolis samplers. A complete eigenstructure analysis for the Metropolized independence sampler is provided. One of the most exciting recent advances in Markov chain theory, the so-called *perfect simulation*, is briefly described and the related literature mentioned. Finally, a theoretical analysis of the dynamic weighting method is given.

In the Appendix, we outline the basics of the probability theory and statistical inference procedures. The interested reader can also find there a rather short description of the popular expectation-maximization (EM) algorithm (Dempster, Laird and Rubin 1977) and a brief discussion of its property.

1.3 Computations in Statistical Physics

Scientists are often interested in simulating from a *Boltzmann distribution* (or Gibbs distribution) which is of the form

$$\pi(\mathbf{x}) = \frac{1}{Z} e^{-U(\mathbf{x})/kT}, \quad (1.1)$$

where \mathbf{x} is a particular configuration of a physical system, $U(\mathbf{x})$ is its potential energy, T is the temperature, and k is the Boltzmann constant. The function $Z = Z(T)$ is called the *partition function* (also called the normalizing constant in non-physics literature).

The Ising model serves to model the behavior of a magnet and is perhaps the best known and the most thoroughly researched model in statistical physics. The intuition behind the model is that the magnetism of a piece of material is the collective contribution of dipole moments of many atomic spins within the material. A simple 2-D Ising model places these atomic spins on a $N \times N$ lattice space, $\mathcal{L} = \{(i, j), i = 1, \dots, N; j = 1, \dots, N\}$, as shown in Figure 1.1.

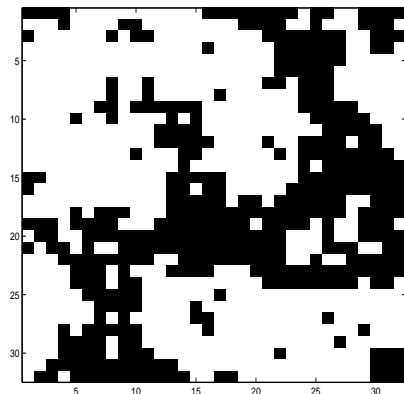


FIGURE 1.1. A configuration of the Ising model on a 32×32 grid space, denoted as \mathcal{L} , with a temperature slightly higher than the critical temperature.

In the model, each site $\sigma \in \mathcal{L}$ hosts a particle that has either a positive or a negative spin. Abstractly, the state of each particle can be represented by a random variable x_σ which is either $+1$ or -1 . A configuration of the whole system is then $\mathbf{x} = \{x_\sigma, \sigma \in \mathcal{L}\}$, whose potential energy is defined as

$$U(\mathbf{x}) = -J \sum_{\sigma \sim \sigma'} x_\sigma x_{\sigma'} + \sum_{\sigma} h_\sigma x_\sigma, \quad (1.2)$$

where the symbol $\sigma \sim \sigma'$ means that they are a neighboring pair, J is called the interaction strength, and h_σ the external magnetic field.

Several important quantities regarding a physical system are often of interest. First, the *internal energy* is defined as

$$\langle U \rangle = E_\pi\{U(\mathbf{x})\}.$$

The notation on the left-hand side is frequently used by physicists to refer to the “state average” of the potential energy, whereas the right-hand side notation is employed mostly by mathematicians and statisticians and is read as “the mathematical expectation of $U(\mathbf{x})$ with respect to π .” They are, of course, the same thing and are equal to the integral

$$I = \int_D U(\mathbf{x})\pi(\mathbf{x})d\mathbf{x},$$

where D is the set of all possible configurations of \mathbf{x} . Clearly, estimating $\langle U \rangle$ based on random samples drawn uniformly from D is disastrous. A much better estimate would have been resulted if we could simulate random draws from the Boltzmann distribution $\pi(\mathbf{x})$.

If we let $\beta = 1/kT$, an interesting relationship between the internal energy and the *partition function* can be derived:

$$\frac{\partial \log Z}{\partial \beta} = -\langle U \rangle.$$

This implies that we can use Monte Carlo methods to estimate the temperature derivative of the partition function, which can then be used to estimate the partition function itself (up to a multiplicative constant). The *free energy* of the system, defined as $F = -kT \log Z$, can also be estimated, up to an additive constant. To date, problems related to the estimation of partition functions of various probabilistic systems still present a significant challenge to researchers in different fields (Meng and Wong 1996, Chen and Shao 1997).

The *specific heat* of the system is defined as

$$C = \frac{\partial \langle U \rangle}{\partial T} = \frac{1}{kT^2} \text{var}_\pi U(\mathbf{x}),$$

and the system’s *entropy* is

$$S = (\langle U \rangle - F)/T.$$

For the Ising model, one is also interested in the *mean magnetization per spin*, defined as

$$\langle m \rangle = E_\pi \left\{ \frac{1}{N^2} \left| \sum_{\sigma \in S} x_\sigma \right| \right\},$$

which can, again, be estimated by Monte Carlo sample averages. Generally, many physical quantities of interest correspond to taking expectations with respect to the Boltzmann distribution and can be estimated by Monte Carlo simulations.

1.4 Molecular Structure Simulation

Simple Liquids Model. In this model, the configuration space is a compact subset in \mathbb{R}^{3k} : $\mathbf{x} = \{x_i; i = 1, \dots, k\}$, where $x_i = (x_{i1}, x_{i2}, x_{i3})^T$ represents the position vector of the i th particle. A simple energy function is of the form

$$U(\mathbf{x}) = \sum_{\text{all } i,j} \Phi(|x_i - x_j|) = \sum_{\text{all } i,j} \Phi(r_{ij}),$$

where

$$\Phi(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right]$$

is called the *Lennard-Jones pair potential*. Its shape is depicted in Figure 1.2.

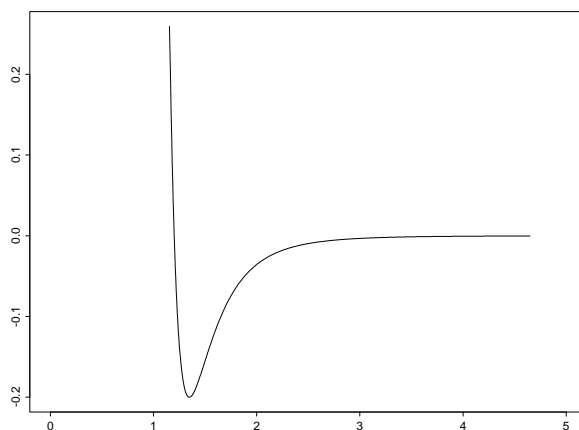


FIGURE 1.2. A plot of the Lennard-Jones function.

A more complicated model for macromolecules, which is widely used in protein structural simulations (Creighton 1993), has a potential energy function of the form

$$U(\mathbf{x}) = \sum_{\text{bonds}} (\text{bond terms}) + \sum_{i,j} \left\{ \Phi(r_{ij}) + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \right\},$$

where the last term on the right-hand side represents electrostatic interaction between two atoms. In macromolecule simulations, one often uses three bond terms that have the form

$$\text{bond terms} = \sum_{\text{bonds}} \frac{k_i}{2} (l_i - l_{i,0})^2 + \sum_{\text{angles}} \frac{k_i}{2} (\theta_i - \theta_{i,0})^2 + \sum_{\text{torsions}} v(\omega_i),$$

where l_i is the bond length, θ_i is the *bond angle*¹, and ω_i is the *torsion angle*². The torsional term $v(\omega)$ has the form

$$v(\omega) = \frac{V_n}{2}(1 + \cos(n\omega - \gamma)). \quad (1.3)$$

Also see Leach (1996) for more details. A water molecule is shown in Figure 1.3.

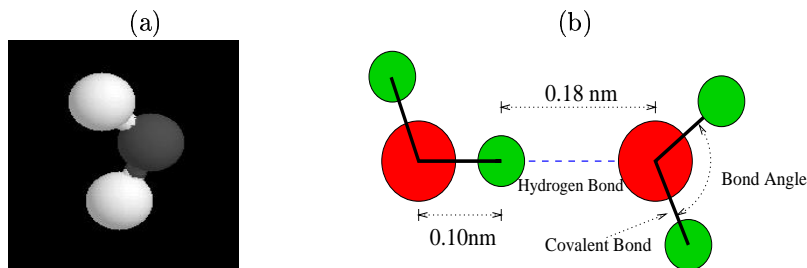


FIGURE 1.3. (a) A water molecule generated by Rasmol, a molecular visualization software (<http://www.umass.edu/microbio/rasmol/>). There are two covalent bonds in H_2O , one between each hydrogen atom and the oxygen atom. (b) A schematic plot of the interactions between two water molecules.

Figure 1.4 shows how a protein molecule interacts with the double-helix structure of a DNA molecule at the atomic level. This interaction is important for gene regulation. In this class of problems, one is often interested in seeing the likely structures of a stable macromolecule and estimating several basic physical quantities such as free energy and specific heat. In protein folding problems, one is sometimes more interested in finding the “minimal-energy” configuration of the system [i.e., finding the configuration \mathbf{x}_0 that minimized $U(\mathbf{x})$].

1.5 Bioinformatics: Finding Weak Repetitive Patterns

The linear biopolymers, DNA, RNA, and proteins, are the three central molecular building blocks of life. DNA is an information storage molecule.

¹The *bond angle* is defined as the angle between the lines connecting the nucleus of one atom to the nuclei of two other atoms that are bonded to it.

²The *torsion angle* is the angle of rotation about the bond B-C in a series of bonded atoms A-B-C-D needed to make all the four atoms be on the same plane (remember that they are in a three-dimensional space). The positive sense is clockwise. If the torsion angle is 180° , the four atoms lie in a planar zigzag (Z-shaped).

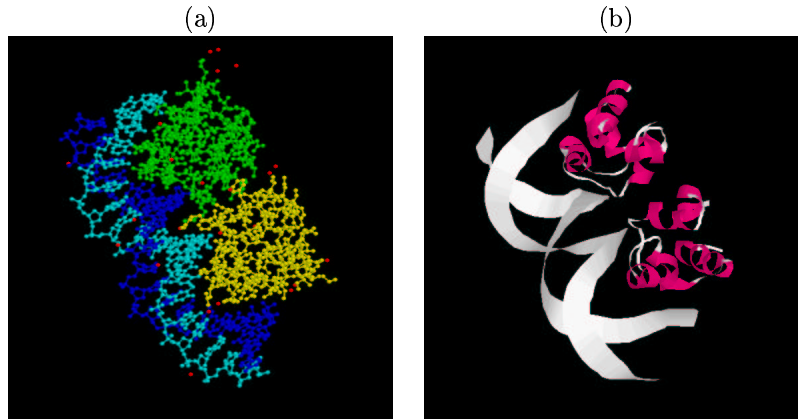


FIGURE 1.4. (a) A ball-and-stick plot of the interaction between the regulatory protein 3CRO of bacteriophage 434 and the DNA segment to which it binds. (b) The same structure as in (a), but expressed by a ribbon representation widely used in the protein structure modeling community.

All of the hereditary information of an individual organism is contained in its genome, which consists of sequences of the four DNA bases (nucleotides), A, T, C, and G. RNA has a wide variety of roles, including a small but important set of functions. Proteins, which are chains of 20 different amino acid residues, are the action molecules of life, being responsible for nearly all of the functions of all living beings and forming many of life's structures. All protein sequences are coded by segments of the genome called genes. The universal genetic code is used to translate triplets of DNA bases, called *codons*, to the 20-letter alphabet of proteins (Campbell 1999). For example, codons “CCA” and “CCG” are both translated into the amino acid “Proline” (abbreviated as Pro or P). How genetic information flows from DNA to RNA and then to protein is regarded as the central paradigm of molecular biology.

The human genome and many other genome sequencing projects have resulted in rapidly growing and publicly available databases of DNA and protein sequences (e.g., <http://www.ncbi.nlm.nih.gov>). The data in these databases are sequences of letters using d -letter ($d = 4$ for DNA or $d = 20$ for proteins) alphabets without punctuation or space characters. One of the most interesting questions scientists are concerned with is how to get any useful biological information from “looking” at these databases. This task is often termed “data mining” for other types of data. The recent announcement of the near-completion of the human genome makes this interesting question more an urgent task for all interested scientists. However, “mining” a biopolymer database is noticeably different from mining other types of databases because (i) many sophisticated structures have

been built in well-organized biopolymer databases [one can take a look at NCBI's GeneBank (whose web address is <http://www.ncbi.nlm.nih.gov>) to have a rough idea], (ii) there is an enormous amount of biological knowledge, and (iii) fundamental laws in physics and chemistry can be applied. Consequently, more sophisticated mathematical/statistical models are often critical in developing a “mining” strategy.

One important problem in analyzing biological sequence data is to find patterns shared by multiple protein or DNA sequences. It is closely related to the task of *local sequence alignment*. The fact that a common pattern appears in several otherwise dissimilar protein sequences often indicates that these proteins may be functionally or structurally related. For example, in Figure 1.4(b), one can see that a helical part of protein 3CRO is inside the major groove of a DNA double-helix structure. This helical part of the protein, often referred to as the *helix-turn-helix* motif, plays an important role in the binding of 3CRO to a DNA segment and turns out to be a rather conserved part in a large family of proteins responsible for gene regulation. In another example, as shown in Figure 1.5, the helix in the light color is the pattern shared by a number of dinucleotide binding proteins; this helix segment is important in interacting with the ligand, dinucleotide. This pattern was discovered by a stochastic search algorithm, the *Gibbs motif sampler* (Liu, Neuwald and Lawrence 1995) to be described later.

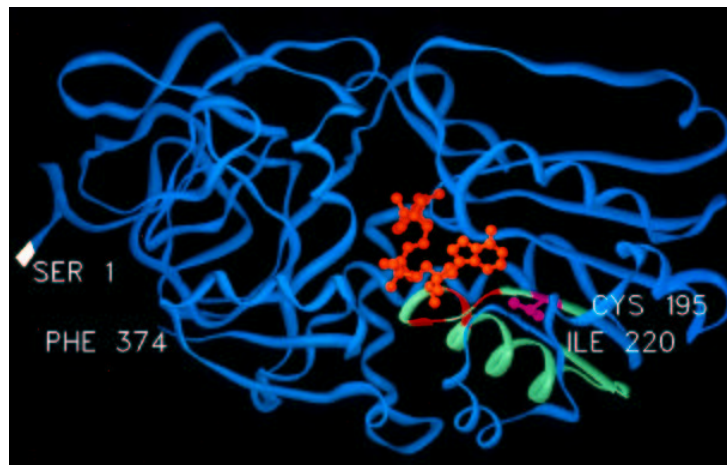


FIGURE 1.5. A ribbon model of the trace of the backbone chain of a dinucleotide binding protein, ADHE. The helical segment in light color (residues 195-220) corresponds to the pattern common to a number of such proteins. The cofactor (NAD) is represented by a stick-and-ball model.

Common patterns in DNA sequences also have important biological implications. For example, the existence of a common short sequence motif

in the upstream untranslated DNA regions (5' UTR) of a set of candidate genes may suggest that these genes are regulated in a similar fashion. This motif pattern, whose occurrences are often very close to the start of a gene (less than a few hundreds base pairs), may correspond to the sites bound by a common regulatory protein, called the transcription factor,³ which regulates the expression level of these genes (Stormo and Hartzell 1989, Lawrence and Reilly 1990, Lawrence et al. 1993, Liu 1994a). These short patterns are often called *binding motifs*. In Figure 1.4(b), the segment of DNA (double-helix structure) that are interacting with protein 3CRO is a *binding site* whose pattern is conserved among the 5' UTRs of a number of related genes.

The above pattern discovery problem can be abstracted as follows: Given a set of K sequences R_1, \dots, R_K , we seek within each sequence mutually similar segments of a specified width w . An analogy is that each R_i is a sentence and our task is to find some “word” (or words) that is most “common” to all the sentences in consideration. If this word occurs in each sequence exactly without any misspellings, one can find it without too much difficulties. But in the biology world, there is seldom anything as good as “exact” matches, implying that we have to describe both the common pattern and the remaining parts of the sequences probabilistically. The simplest statistical model is the *block-motif* model as depicted in Figure 1.6:

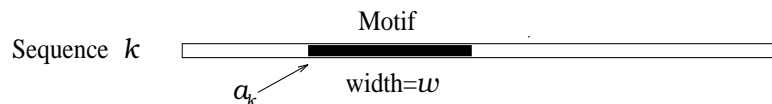


FIGURE 1.6. A schematic plot of the block-motif model used for our pattern finding.

In this model, the letter at the i th position within the “motif” (pattern) is assumed to be drawn independently from a multinomial distribution with parameter θ_i , $i = 1, \dots, w$; letters elsewhere follow a common background multinomial model with parameter θ_0 , where θ_i is a probability vector of length d , the size of the letter alphabet ($d = 4$ for DNA and $d = 20$ for protein). In other words, the residues (or base pairs) we see outside the motif pattern are treated as i.i.d. observations from the multinomial distribution with parameter θ_0 ; a residue observed at position i within the motif pattern is generated from probability vector θ_i . [It is possible to use a more sophisticated background model to improve the algorithm’s efficiency, see Liu, Brutlag and Liu (1995) and McCue et al. (2001).] What makes this problem difficult is that we do not know the location of the “word”

³A transcription factor is a protein which binds to certain site of a DNA molecule to either enhance or repress the gene expression.

(pattern) in each sequence. Based on this simple statistical model and the Gibbs sampling principle, Liu (1994a) and Lawrence et al. (1993) derived the following simple, yet effective, Monte Carlo algorithm (Chapter 6).

In what they called the *site sampler*, the motif locations (sites) are initialized at random; that is, position $a_k^{(0)}$ (for $k = 1, \dots, K$) is a randomly chosen position of the k th sequence. For $t = 1, \dots, m$, the algorithm iterates the following steps:

- Select a sequence, say the k th sequence, either deterministically or at random.
- Draw a new motif location a_k according to the predictive distribution

$$P(a_k | a_1^{(t)}, \dots, a_{k-1}^{(t)}, a_{k+1}^{(t)}, \dots, a_K^{(t)}) \quad (1.4)$$

and update the current motif location $a_k^{(t)}$ to $a_k^{(t+1)} = a_k$.

- Let $a_j^{(t+1)} = a_j^{(t)}$ for $j \neq k$.

Although there are many choices of the predictive distribution function used for updating the alignment [e.g., one can let $P(a_k | \dots)$ be proportional to certain fitness measure of the segment indexed by a_k to the current multinomial profile resulting from the $a_j^{(t)}$, $j \neq k$], those that make the foregoing iteration consistent have to be the ones derived from a complete Bayesian statistical model. More detailed derivations will be given in Chapter 6.

1.6 Nonlinear Dynamic System: Target Tracking

Dynamic modeling is widely used in areas such as computer vision, economical data analysis, feedback control systems, mobile communication, radar or sonar surveillance systems, etc. An important problem in such a dynamical system is the on-line (instantaneously in real time) processing of information (such as estimation and prediction) regarding the system characteristics. These tasks are generally termed “filtering” in engineering and statistical literature. A main challenge to researchers in these fields is to find efficient filtering algorithms.

Target tracking in a clutter environment as shown in Figure 1.7 is a typical example of dynamic modeling. To facilitate tracking, one often uses a linear Gaussian *state-space model* [also called *dynamic linear model* by West and Harrison (1989)] to describe the movement of the target object. A typical 2-D tracking model is as follows.

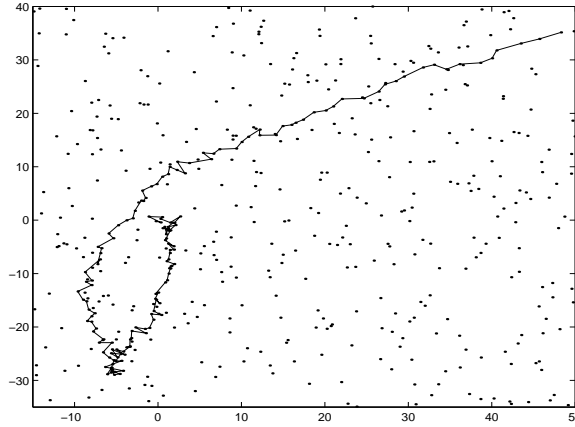


FIGURE 1.7. A simulation of the target tracking problem in a 2-D space. The dots connected by the line represent the signals (y_t) generated by the true positions of the target; the dots elsewhere represent the nearby confusing objects (simulated from a Poisson point process with rate 8%).

State Equation:

$$\begin{aligned} \begin{pmatrix} v_{t,1} \\ v_{t,2} \end{pmatrix} &= \begin{pmatrix} v_{t-1,1} \\ v_{t-1,2} \end{pmatrix} + \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}, \\ \begin{pmatrix} s_{t,1} \\ s_{t,2} \end{pmatrix} &= \begin{pmatrix} s_{t-1,1} \\ s_{t-1,2} \end{pmatrix} + \begin{pmatrix} v_{t-1,1} \\ v_{t-1,2} \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}, \end{aligned}$$

where $s_t = (s_{t,1}, s_{t,2})^T$ is the object's position vector at time t and $v_t = (v_{t,1}, v_{t,2})^T$ is its current velocity vector. The state equation innovation $\epsilon_t = (\epsilon_{t,1}, \epsilon_{t,2})^T$ is distributed as $\mathcal{N}(\mathbf{0}, \sigma_a^2 \mathbf{I})$. This model says that the speed (vector) of the object evolves like a Gaussian random walk and the position of the object follows the change of its speed. We assume, however, that a noisy version of the object's true position $(y_{t,1}, y_{t,2})^T$ is observable.

Observation Equation:

$$\begin{pmatrix} y_{t,1} \\ y_{t,2} \end{pmatrix} = \begin{pmatrix} s_{t,1} \\ s_{t,2} \end{pmatrix} + \begin{pmatrix} e_{t,1} \\ e_{t,2} \end{pmatrix},$$

where the observation noise $e_t = (e_{t,1}, e_{t,2})^T$ follows $\mathcal{N}(\mathbf{0}, \sigma_b^2 \mathbf{I})$.

By writing $x_t = (s_{t,1}, s_{t,2}, v_{t,1}, v_{t,2})^T$ and $y_t = (y_{t,1}, y_{t,2})^T$, we can rewrite the foregoing system more briefly as

$$\begin{aligned} x_t &= Gx_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \sigma_a^2 A); \\ y_t &= Hx_t + e_t, \quad e_t \sim \mathcal{N}(\mathbf{0}, \sigma_b^2 \mathbf{I}), \end{aligned}$$

where

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} \frac{1}{4} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{pmatrix}, \quad \text{and} \quad H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T.$$

Mathematically, the tracking task is accomplished by *on-line* estimation of the object's position, $(s_{t,1}, s_{t,2})$, based on all information available until time t . If the y_t are always observable at time t (i.e., there is no clutter), this estimation task can be achieved rather efficiently via the *Kalman filter* (Kalman 1960) because of the linear Gaussian structures being employed.

If we consider a clutter environment, however, then at time t we observe instead a clutter of points, $z_t = \{z_{t,1}, \dots, z_{t,k_t}\}$, in a 2-D detection region with area Δ in which the number of false signals follow a spatial Poisson process with rate λ . The set z_t includes the true measurement $y_t = (y_{t,1}, y_{t,2})^T$ with probability p_d . Other z 's are treated as uniform within the detection range. This model for tracking in clutter is no longer a linear Gaussian system. To date, there has not been a universally effective algorithm for dealing with nonlinear and non-Gaussian systems. Depending on the features of individual problems, some generalizations of the Kalman filter can be effective. A few well-known generalizations are the extended Kalman filters (Gelb 1974), the Gaussian sum filters (Anderson and Moore 1979), and the iterated extended Kalman filters (Jazwinski 1970). Most of these methods are based on local linear approximations of the nonlinear system. More recently, researchers are attracted to a new class of filtering methods based on the sequential Monte Carlo approach. In Section 4.5.1, we will show that the method based on sequential Monte Carlo performs very well for the target tracking problem.

1.7 Hypothesis Testing for Astronomical Observations

Efron and Petrosian (1999) described an interesting problem in astronomy. Figure 1.8 shows us a set of doubly truncated astronomical data in which log-luminosities y_i (or the boundary point of its truncation interval) is plotted against redshifts z_i for $n = 210$ quasars. In other words, due to experimental constraint, we are able to observe y_i when it is within a known interval R_i depending on z_i , otherwise we only know that y_i is outside of the interval R_i . Two questions are of interest: (a) Are the y_i independent of the z_i ? (b) Assuming independence, can we estimate the marginal distribution of the y_i ?

The real data actually consist of independently collected quadruplets

$$(z_i, m_i, a_i, b_i), \quad i = 1, \dots, n,$$

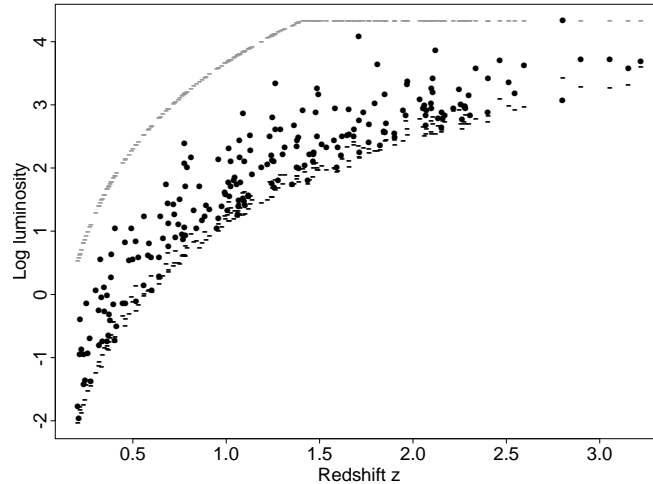


FIGURE 1.8. Doubly truncated data from an astronomical study of Efron and Petrosian (1999). Points represent (redshift, log-luminosity) for 210 quasars. Luminosity subject to upper and lower truncation is indicated by “—” in the figure.

where a_i and b_i are lower and upper truncation limits on m_i , respectively, and m_i is the apparent magnitude of the i th quasar. Quasars with apparent magnitude above b_i were too dim to produce usable redshift observations. The lower limit a_i was used to avoid confusion with nonquasar stellar objects. In the dataset, $a_i = 16.08$ for all i , and b_i is between 18.494 and 18.934. Since further quasars appear dimmer (and have larger values of m_i), one needs to use Hubble’s law, which states that distance is proportional to redshift, to transform m_i into a luminosity measurement that should be independent of distance. The plotted y_i is thus derived as

$$y_i = 19.894 - 2.303 \frac{m_i}{2.5} + 2 \log(Z_i - \sqrt{Z_i}) - \frac{1}{2} \log(Z_i),$$

where $Z_i = 1 + z_i$ [more details are given in Efron and Petrosian (1999)].

To answer question (a), a simple permutation test was considered by Efron and Petrosian (1999). In other words, if we could observe the y_i without truncation, then under the null hypothesis that the y_i and z_i are independent, the permuted y_i should “look” the same as the old y_i . Thus, if we permute the y_i many times and compute the test statistic, say $T = t(\mathbf{y}^*)$, repeatedly, where $\mathbf{y}^* = (y_1^*, \dots, y_n^*)$ is a permutation of the original y ’s, we can obtain the “null distribution” of T : that is, if the independence hypothesis were true, we would expect T to behave like what we see in the repeated permutations. Because of truncations, however, the \mathbf{y}^* is observable only if all the $y_i^* \in R_i$ (i.e., satisfying the double truncation requirement).

Let \mathcal{Y} be the set of all permutations that satisfy the truncation requirement. Then to compute the p -values for the statistical test, we need to

be able to generate permutations of (y_1, \dots, y_n) uniformly in \mathcal{Y} . In order to achieve this sampling with a reasonable amount of computing time, one needs to employ the *Metropolis algorithm* (Chapter 5). Briefly, to obtain an approximately uniform sample from the set of all allowable permutations \mathcal{Y} , we can start with the identity permutation $\sigma(i) = i$, $i = 1, \dots, n$. At each iteration step, we do the following:

- Randomly pick a pair of elements, i and j , say, and propose a new permutation σ' which differs from σ only by transposing $\sigma(i)$ and $\sigma(j)$.
- If the new permutation resulting from the transposition is still in \mathcal{Y} , then we accept the new configuration; otherwise, we stay put.

The validity of this algorithm can be understood in part by the Markov invariance property explained in Chapter 5. Diaconis, Graham and Holmes (2001) showed that the space \mathcal{Y} is “connected” by the foregoing Markov moves. Thus, the equilibrium state of the algorithm follows the target distribution (i.e., uniform in \mathcal{Y}). In other words, if we carry out a large number ($m=1,000,000$, say) of iterations of the above steps, the latter m_0 ($m_0=900,000$, say) correlated permutations may be regarded as being drawn uniformly from \mathcal{Y} . These samples can be used to estimate the null distribution of T . An importance sampling approach can also be applied to achieve the same goal.

1.8 Bayesian Inference of Multilevel Models

A basic regression model used for the prediction of a student’s first-year average (FYA) from the Law School Aptitude Test (LSAT) score and undergraduate grade point average (UGPA) is

$$\widehat{\text{FYA}} \propto \text{LSAT} + (?) \times \text{UGPA}.$$

However, the important question for each law school is how to choose its own multiplier of UGPA in this equation (Rubin 1980). In the past, a number of law schools simply chose 200 as the multiplier. But due to recent “grade inflation,” some recommended smaller multipliers such as 130. Many law schools favored estimating the multiplier empirically, using recent data from attending students. Because different law schools may have a different education effect on its students, we do not expect that the multipliers used by different schools to be the same. On the other hand, one may also feel that all law schools have certain common characteristics and the multiplier used in one school should provide information for other schools to determine their own.

The dataset in Rubin (1980) consisted of records from 82 law schools. A standard linear regression was carried out for each school to obtain an initial estimate of the multiplier, \hat{L}_i . Then Rubin (1980) proceeded by assuming that $Y_i = \arctan(\hat{L}_i/200)$ follows a Gaussian distribution with mean θ_i and variance s_i^2 , where s_i^2 is estimated from the i th law school's past record. A hierarchical structure is further imposed on the θ_i :

$$\theta_i \stackrel{\text{i.i.d.}}{\sim} N(\mu, \sigma^2).$$

The $\hat{\theta}_i$ estimated from this model can be used to calculate the multiplier [i.e., $\hat{L}_i = 200 \tan(\hat{\theta}_i)$] for the i th school in its FYA prediction.

A more general form of the model Rubin used can be stated as follows:

$$Y_i | \theta_i \stackrel{\text{i.i.d.}}{\sim} f_i(y_i | \theta_i), \quad (1.5)$$

$$\theta_i \stackrel{\text{i.i.d.}}{\sim} G(\theta | \lambda), \quad (1.6)$$

for $i = 1, \dots, k$. This model is often referred to as a “hierarchical model.” Of interest in this model are the estimation of all the unknown parameters, θ_i and λ , and the quantification of uncertainties in these estimates. A *hierarchical Bayes model* is completed by giving a prior distribution $f_0(\mu, \sigma^2)$ to the hyper-parameter λ .

With notation $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{82})$, we can write down the joint distribution of the data and the parameters in Rubin's study:

$$p(Y, \boldsymbol{\theta}, \mu, \sigma^2) = f_0(\mu, \sigma) \prod_{i=1}^{82} \phi(Y_i; \theta_i, s_i) \phi(\theta_i; \mu, \sigma),$$

where $\phi(x; \mu, \sigma)$ is the density function for $N(\mu, \sigma^2)$. By the Bayes theorem, we derive the *posterior distribution* of all the unknown variables:

$$\pi(\boldsymbol{\theta}, \mu, \sigma^2) = \frac{p(Y, \boldsymbol{\theta}, \mu, \sigma^2)}{p(Y)} \propto p(Y, \boldsymbol{\theta}, \mu, \sigma^2).$$

The *Bayes estimator* of θ_i is its posterior mean:

$$\hat{\theta}_i = E(\theta_i | Y) = \int \pi(\boldsymbol{\theta}, \mu, \sigma^2) d\boldsymbol{\theta}_{[-i]} d\mu d\sigma^2,$$

where $\boldsymbol{\theta}_{[-i]}$ is all but the i th component of $\boldsymbol{\theta}$. This quantity is not analytically available. Its numerical approximation needs high-dimensional integration and can be most effectively solved by Monte Carlo techniques (Gelfand and Smith 1990).

1.9 Monte Carlo and Missing Data Problems

A central theme of statistics is to infer unobserved parameters from observed data. Let $\boldsymbol{\theta}$ be the parameter vector of interest. In a *parametric*

inference problem, the observed vector \mathbf{y} is regarded as the realized value of a random vector whose distribution is $f(\mathbf{y} \mid \boldsymbol{\theta})$, known up to a finite-dimensional parameter $\boldsymbol{\theta}$. Two of the most popular approaches for inferring $\boldsymbol{\theta}$ are the *maximum likelihood estimation* (MLE) method and the *Bayes* method. In the MLE, the unknown parameter $\boldsymbol{\theta}$ is estimated by the $\hat{\boldsymbol{\theta}}$ that maximizes $f(\mathbf{y} \mid \boldsymbol{\theta})$, which is also called the *likelihood*. Computationally, this becomes an optimization problem. In the Bayesian methods, one relies on the posterior distribution of $\boldsymbol{\theta}$, $p(\boldsymbol{\theta} \mid \mathbf{y})$, to make inferential statement, which can be formulated as a high-dimensional integration problem. (See Section A.2 of the Appendix for more details.)

Many statistical problems do not naturally suggest “nice” models enabling simple analytical solutions to the posterior calculation. However, a tractable structure can often be obtained if some auxiliary parts are augmented to the system. In the field of statistics, these auxiliary components can often be viewed as “missing data” (Tanner and Wong 1987). For example, the hierarchical model discussed in the previous section can be treated as a “missing data problem” in which the individual effects θ_i are viewed as missing data. In a state-space model (Sections 1.6, 3.3, and 4.5), the unobserved state variable x_t can be viewed as missing data. In the motif alignment problem of biological sequence analysis (Section 1.5), the pattern location a_k in each sequence can be treated as missing data.

A Bayesian missing data problem can be formulated as follows: Suppose the “complete-data” model $f(\mathbf{y} \mid \boldsymbol{\theta})$ has a nice clean form from which we can obtain the analytical form of the posterior distribution. However, only part of \mathbf{y} , denoted as \mathbf{y}_{obs} , is observed, and the remaining part, \mathbf{y}_{mis} , is missing. Let $\mathbf{y} = (\mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}})$. The *joint* posterior distribution of \mathbf{y}_{mis} and $\boldsymbol{\theta}$ is

$$\pi(\boldsymbol{\theta}, \mathbf{y}_{\text{mis}}) \propto f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta}), \quad (1.7)$$

and, marginally, the *observed-data posterior* is

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{\text{obs}}) = \pi(\boldsymbol{\theta}) = \int \pi(\boldsymbol{\theta}, \mathbf{y}_{\text{mis}}) d\mathbf{y}_{\text{mis}}.$$

If we can draw Monte Carlo samples $(\boldsymbol{\theta}^{(1)}, \mathbf{y}_{\text{mis}}^{(1)}), \dots, (\boldsymbol{\theta}^{(m)}, \mathbf{y}_{\text{mis}}^{(m)})$ from the joint posterior distribution π , then the histogram based on $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}$ can serve as an approximation to the observed-data posterior distribution $p(\boldsymbol{\theta} \mid \mathbf{y}_{\text{obs}})$. Furthermore, if we are interested in estimation some posterior expectation of $\boldsymbol{\theta}$, say $E\{h(\boldsymbol{\theta}) \mid \mathbf{y}_{\text{obs}}\}$, we can estimate it by

$$\hat{h} = \frac{1}{m} \left[h(\boldsymbol{\theta}^{(1)}) + \dots + h(\boldsymbol{\theta}^{(m)}) \right].$$

Note that the form of π in (1.7) is no different, at least in principle, from that in (1.1). Hence, statisticians and physicists are indeed facing a similar computational problem.

To a broader scientific audience, the concept of “missing data” is perhaps a little odd, for many scientists may not believe that they have any missing data. In the most general and abstract form, the “missing data” can refer to any augmented *component* of the probabilistic system under consideration and the inclusion of this component often results in a simpler structure and easier computation (there are more examples to demonstrate this need in the later chapters). However, this component needs to be marginalized (integrated) out in the final inference. Indeed, it is the scientist’s desire to marginalize part or whole of a probabilistic system under investigation that has been the main impetus to the development of Monte Carlo techniques (Gilks, Richardson and Spiegelhalter 1998).