Monte Carlo Strategies In Scientific Computing

Jun S. Liu
Department of Statistics
Harvard University

June 13, 2001
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 $\pi$ as

$$\hat{\pi} = \lim_{N \to \infty} \frac{2l}{Np_ND},$$

which will “converge” to $\pi$ as $N$ increases to infinity. Numerous investigators actually used this setting to estimate $\pi$. 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 (Gemai and Gemai 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 method-
ologies 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 — Yugu Chen, Lingyu Chen, Chiara Sabatti, Tanya Logvinenko, Zhaohui Qin and Jini Zhang — have contributed in many ways to the development of this book. Ms. Heden Tombroupolos 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
Preface
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 .......................... 26
  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 ............................... 33
    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 ....................... 49
2.8 Problems ....................................................... 51

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 ...................................... 70
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 ....................................................... 77

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

   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 ..... 169
  8.3 Transformation Group and Generalized Gibbs ..... 171
  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 .............................................. 181

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 .............................. 197
  9.6 Application of HMC in Statistics .................. 198
    9.6.1 Indirect observation model .................... 199
    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 ....................... 222

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 237
   11.5.4 Bayesian neural network training 239
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 257
      12.5.2 Poincaré inequality 257
      12.5.3 Example: Simple random walk on a graph 259
      12.5.4 Cheeger's inequality 261
   12.6 Functional Analysis for Markov Chains 263
      12.6.1 Forward and backward operators 264
      12.6.2 Convergence rate of Markov chains 266
      12.6.3 Maximal correlation 267
   12.7 Behavior of the Averages 269

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
xvi  Contents

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 . . . . . . . . . . . 300
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 . . . . . . . . . . . . . . . . 308
A.4  The Expectation-Maximization Algorithm . . . . . . . . . 310

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

$$I = \int_D g(x) \, dx,$$

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

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

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,

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

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

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

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

\[
\hat{I}_m = \frac{1}{m} \left\{ g(b_1) + \cdots + g(b_m) \right\},
\]

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 \( x^{(1)}, \ldots, x^{(m)} \) 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 \( \sigma^2 \), which measures how “uniform” the function \( g \) is in region \( D \), can be formidable 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 \( x^{(1)}, \ldots, x^{(m)} \) from a nonuniform distribution \( \pi(x) \) that puts more probability mass on “important” parts of the state space \( D \). One can estimate integral \( I \) as

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

which has a variance \( \sigma^2_\pi = \text{var}_\pi \left\{ g(x)/\pi(x) \right\} \). In the most fortunate case, we may choose \( \pi(x) \propto g(x) \) 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” \( \pi \) which will explore more in regions where the value of \( g \) is high. In such a situation, generating random draws from \( \pi \) can be a challenging problem.

Demands for sampling from a nonuniform distribution \( \pi \) are also seen from another set of problems in bioinformatics, computational chemistry, physics, structural biology, statistics, etc. In these problems, the desired probability distribution \( \pi(x) \) of a complex system, where \( x \) 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 
\[
\pi(x) = Z(T) e^{-\frac{h(x)}{kT}},
\]
where \( k \) is the Boltzmann constant, \( T \) is the system’s temperature, \( h(x) \) 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 \( E_x[g(x)] \) for a suitable function \( g \). In Bayesian statistical inference, \( \mathbf{x} \) often represents the joint configuration of missing data and parameter values and \( \pi(x) \) 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, \( h(x) \), defined on a possibly complex configuration space. The problem is equivalent to finding the maximum of another function, \( q_T(x) = e^{-\frac{h(x)}{T}} \) (as long as \( T > 0 \)). In the case when \( q_T(x) \) is integrable for all \( T > 0 \), which is most common in practice, we can make up a family of probability distributions:

\[
\pi_T(x) \propto e^{-\frac{h(x)}{T}}, \quad T > 0.
\]

If we can sample from \( \pi_T(x) \) when \( T \) is sufficiently small, resulting random draws will most likely be located in the vicinity of the global minimum of \( h(x) \). 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 \( \pi(x) \), 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 \( \pi \) 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 \( \pi \), 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 \( p(x) \), which differs from, but should be similar to, the target distribution \( \pi \). 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 \( \pi \) 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 1964, 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 recast 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

\[ \pi(x) = \frac{1}{Z} e^{-\frac{U(x)}{kT}}, \]

where \( x \) is a particular configuration of a physical system, \( U(x) \) 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, \( \mathcal{L} = \{(i, j), i = 1, \ldots, N; j = 1, \ldots, N\} \), as shown in Figure 1.1.

![Figure 1.1. A configuration of the Ising model on a 32 x 32 grid space, denoted as \( \mathcal{L} \), with a temperature slightly higher than the critical temperature.](image)

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 \( x_\sigma \) which is either \( +1 \) or \( -1 \). A configuration of the whole system is then \( x = \{ x_\sigma, \sigma \in \mathcal{L} \} \), whose potential energy is defined as

\[ U(x) = -J \sum_{\sigma \sim \sigma'} x_\sigma x_{\sigma'} + \sum_\sigma h_\sigma x_\sigma, \]

where the symbol \( \sigma \sim \sigma' \) means that they are a neighboring pair, \( J \) is called the interaction strength, and \( h_\sigma \) the external magnetic field.
Several important quantities regarding a physical system are often of interest. First, the internal energy is defined as

\[ \langle U \rangle = E_\pi \{ U(x) \} . \]

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 \( U(x) \) with respect to \( \pi \).” They are, of course, the same thing and are equal to the integral

\[ I = \int_D U(x) \pi(x) dx , \]

where \( D \) is the set of all possible configurations of \( x \). 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 \( \pi(x) \).

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

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

and the system’s entropy is

\[ S = \langle \langle U \rangle - F \rangle / T \rangle . \]

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 $\mathbb{R}^{3k}$: $x = \{x_i; i = 1, \ldots, k\}$, 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

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

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.

![Graph](image)

**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

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

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

$$\text{bond terms} = \sum_{bonds} \frac{k_i}{2}(r_i - r_{i,0})^2 + \sum_{angles} \frac{k_i}{2}(\theta_i - \theta_{i,0})^2 + \sum_{torsions} v(\phi_i),$$
where $l_i$ is the bond length, $\theta_i$ is the bond angle\(^1\), and $\omega_i$ is the torsion angle\(^2\). The torsional term $v(\omega)$ has the form

$$v(\omega) = \frac{V_\omega}{2}(1 + \cos(n\omega - \gamma)).$$  \hspace{1cm} (1.3)

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

![Water molecule](image)

**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\(_2\)O, 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 $x_0$ that minimized $U(x)$].

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.

---

\(^1\)The *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.

\(^2\)The *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).
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.](image)

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 \(R_1, \ldots, R_K\), 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](image.png)

**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 \(\theta_i\), \(i = 1, \ldots, w\); letters elsewhere follow a common background multinomial model with parameter \(\theta_0\), where \(\theta_i\) 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 \(\theta_0\); a residue observed at position \(i\) within the motif pattern is generated from probability vector \(\theta_i\). [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”

\(^3\)A 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 $k = 1, \ldots , K$) is a randomly chosen position of the $k$th sequence. For $t = 1, \ldots , m$, 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

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

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 $P(a_k | \ldots )$ 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:**

\[
\begin{pmatrix}
  v_{t,1} \\
  v_{t,2}
\end{pmatrix}
= \begin{pmatrix}
  v_{t,1,1} \\
  v_{t,1,2}
\end{pmatrix}
+ \begin{pmatrix}
  \epsilon_{t,1} \\
  \epsilon_{t,2}
\end{pmatrix},
\]

\[
\begin{pmatrix}
  s_{t,1} \\
  s_{t,2}
\end{pmatrix}
= \begin{pmatrix}
  s_{t,1,1} \\
  s_{t,1,2}
\end{pmatrix}
+ \begin{pmatrix}
  v_{t,1,1} \\
  v_{t,1,2}
\end{pmatrix}
+ \frac{1}{2} \begin{pmatrix}
  \epsilon_{t,1} \\
  \epsilon_{t,2}
\end{pmatrix},
\]

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 \(\mathcal{N}(0, \sigma^2_s I)\). 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 \(\mathcal{N}(0, \sigma^2_o I)\).

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

\[
x_t = G x_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2_s A);
\]

\[
y_t = H x_t + e_t, \quad e_t \sim \mathcal{N}(0, \sigma^2_o I),
\]
where 

\[ G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T. \]

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, \(z_t = \{z_{t,1}, \ldots, z_{t,k_t}\}\), in a 2-D detection region with area \(\Delta\) in which the number of false signals follow a spatial Poisson process with rate \(\lambda\). The set \(z_t\) includes the true measurement \(y_t = (y_{t,1}, y_{t,2})^T\) with probability \(p_t\). 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 (Geb 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

\[(z_i, m_i, a_i, b_i), \quad i = 1, \ldots, n,\]
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

$$y_i = 19.894 - 2.300 \frac{m_i}{2.5} + 2 \log(Z_i - \sqrt{Z_i}) - \frac{1}{2} \log(Z_i),$$

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 $T = t(y^*)$, repeatedly, where $y^*=(y_1^*, \ldots, y_n^*)$ 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 $y^*$ 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 \((y_1, \ldots, y_n)\) 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 \(\sigma(i) = i, \ i = 1, \ldots, n\). At each iteration step, we do the following:

- Randomly pick a pair of elements, \(i\) and \(j\), say, and propose a new permutation \(\sigma'\) which differs from \(\sigma\) 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 \((m=1,000,000, \text{say})\) of iterations of the above steps, the latter \(m_0\) \((m_0=900,000, \text{say})\) 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 $Y_i = \arctan(L_i/200)$ follows a Gaussian distribution with mean $\theta_i$ 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 $\theta_i$:

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

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:

$$Y_i \mid \theta_i \overset{i.i.d.}{\sim} f_i(y_i \mid \theta_i),$$

$$\theta_i \overset{i.i.d.}{\sim} G(\theta \mid \lambda),$$

for $i = 1, \ldots, k$. This model is often referred to as a “hierarchical model.” Of interest in this model are the estimation of all the unknown parameters, $\theta_i$ and $\lambda$, 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 $\lambda$.

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

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

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:

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

The Bayes estimator of $\theta_i$ is its posterior mean:

$$\hat{\theta}_i = E(\theta_i \mid Y) = \int \pi(\theta, \mu, \sigma^2) d\theta_i \mid Y d\mu d\sigma^2,$$

where $\theta_i$ is all but the $i$th component of $\theta$. 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 $\theta$ 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 $f(\mathbf{y} | \theta)$, known up to a finite-dimensional parameter $\theta$. Two of the most popular approaches for inferring $\theta$ are the maximum likelihood estimation (MLE) method and the Bayes method. In the MLE, the unknown parameter $\theta$ is estimated by the $\hat{\theta}$ that maximizes $f(\mathbf{y} | \theta)$, which is also called the likelihood. Computationally, this becomes an optimization problem. In the Bayesian methods, one relies on the posterior distribution of $\theta$, $p(\theta | \mathbf{y})$, 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 $\theta_i$ 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 $f(\mathbf{y} | \theta)$ 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 $\mathbf{y}_{\text{obs}}$, is observed, and the remaining part, $\mathbf{y}_{\text{mis}}$, is missing. Let $\mathbf{y} = (\mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}})$. The joint posterior distribution of $\mathbf{y}_{\text{mis}}$ and $\theta$ is

$$\pi(\theta, \mathbf{y}_{\text{mis}}) \propto f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} | \theta) f_0(\theta),$$

and, marginally, the observed-data posterior is

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

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

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

Note that the form of $\pi$ 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).
1. Introduction and Examples
2

Basic Principles: Rejection, Weighting, and Others

2.1 Generating Simple Random Variables

To generate random variables that follow a general probability distribution function $\pi$, we need first to generate random variables uniformly distributed in $[0,1]$. These random variables are often called \textit{random numbers} for simplicity. However, this “simple-sounding” task is not easily achievable on a computer. But even if it were possible, it might not be desirable to use authentic random numbers because of the need to debug computer programs. In debugging a program, we often have to repeat the same computation many times; this require us to reproduce the same sequence of random numbers repeatedly. What becomes an accepted alternative in the community of scientific computing is to generate \textit{pseudo-random} numbers. More formally, we can define a \textit{uniform pseudo-random number generator} as an algorithm which, starting from an initial value $u_0$ (i.e., the \textit{seed}), produces a sequence $(u_i) = (D^i(u_0))$ of values in $[0,1]$. For all $n$, the values $(u_1, \ldots, u_n)$ should reproduce the behavior of an i.i.d. sample $(V_1, \ldots, V_n)$ of uniform random variables. A few very good pseudo-random number generators are available; we refer the reader to Marsaglia and Zaman (1993) and Knuth (1997) for further reference. Consequently, we assume from now on that uniform random variables can be satisfactorily produced on the computer. The following simple lemma enables us to produce nonuniform random variables. Its proof is left as an exercise for the reader.
Lemma 2.1.1 Suppose $U \sim \text{Uniform}[0,1]$ and $F$ is a one-dimensional cumulative distribution function (cdf). Then, $X = F^{-1}(U)$ has the distribution $F$. Here we define $F^{-1}(u) = \inf\{x; F(x) \geq u\}$.

This lemma suggests to us an explicit way (i.e., the inversion method) of generating a one-dimensional random variable when its cdf is available. However, because many distributions (e.g., Gaussian) do not have a closed-form cdf, it is often difficult to directly apply the above inversion procedure. For distributions with nice mathematical properties, special methods are often available for drawing random samples from them. For example, a fast way of generating standard Gaussian random variables is to use the property that a standard bivariate Gaussian random vector $(X, Y)$ (with zero mean and identity covariance matrix) can be generated by first uniformly choosing an angle in $\mathbb{R}^2$ (two-dimensional Euclidean space) and then generating the square distance from an Exponential distribution (Devroye 1986). A Beta random variable can be constructed as the ratio $X_1/(X_1 + X_2)$, where $X_1$ and $X_2$ are independent Gamma random variables. For mathematically less convenient distributions, von Neumann (1951) proposed a very general algorithm, the rejection method, which can — at least in principle — be applied to draw from any probability distribution with a density function given up to a normalizing constant, regardless of dimensions.

2.2 The Rejection Method

Suppose $l(x) = c\pi(x)$ is computable, where $\pi$ is a probability distribution function or density function and $c$ is unknown. If we can find a sampling distribution $g(x)$ and “covering constant” $M$ so that the envelope property [i.e., $Mg(x) \geq l(x)$] is satisfied for all $x$, then we can apply the following procedure.

Rejection sampling [von Neumann (1951)]:

(a) Draw a sample $x$ from $g(\cdot)$ and compute the ratio

$$r = \frac{l(x)}{Mg(x)} \leq 1.$$

(b) Flip a coin with success probability $r$;

- if the head turns up, we accept and return the $x$;
- otherwise, we reject the $x$ and go back to (a).

The accepted sample follows the target distribution $\pi$.

To show that the foregoing procedure is correct, we let $I$ be the indicator function so that $I = 1$ if the sample $X$ drawn from $g(\cdot)$ is accepted, and
2.2 The Rejection Method

$I = 0$, otherwise. Then, we observe that

$$P(I = 1) = \int P(I = 1 \mid X = x)g(x)dx = \int \frac{c\pi(x)}{Mg(x)}g(x)dx = \frac{c}{M}. $$

Hence,

$$p(x \mid I = 1) = \frac{c\pi(x)}{Mg(x)}g(x)/P(I = 1) = \pi(x).$$

Because the expected number of “operations” for obtaining one accepted sample is $M$, the key to a successful application of the algorithm is to find a good trial distribution $g(x)$ which gives rise to a small $M$. It is usually very difficult to apply the simple rejection method for a high-dimensional Monte Carlo simulation problem.

**Example:** *Truncated Gaussian distribution.* Suppose we want to draw random samples from $\pi(x) \propto \phi(x)I_{\{x > c\}}$, where $\phi(x)$ is the standard normal density and $I$ is the indicator function. A simple strategy can be applied when $c < 0$: We continue to generate random samples from a standard Gaussian distribution until a sample satisfying $X > c$ is obtained. In the worst case, the efficiency of this method is 50%.

For $c > 0$, especially when $c$ is large, the above strategy is very inefficient. We can use the rejection method with an exponential distribution as the envelope function. Suppose the density of this exponential distribution has the form $\lambda_0e^{-\lambda_0x}$. We want to find the smallest constant $b$ such that

$$\frac{\phi(x + c)}{1 - \Phi(c)} \leq b\lambda_0e^{-\lambda_0x}, \quad \forall \, x \geq 0.$$

The optimal choice of $b$ is

$$b = \exp\{(\lambda_0^2 - 2\lambda_0c)/2\}/\sqrt{2\pi\lambda_0(1 - \Phi(c))}. $$

The acceptance rate for using the posited exponential distribution as the envelope function is then $1/b$. To achieve the minimum rejection rate, we further find that the best choice for $\lambda_0$ is

$$\lambda_0 = (c + \sqrt{c^2 + 4})/2.$$

With this choice of $\lambda_0$ and $b$, we can implement the rejection method. The rejection rate for this scheme decreases as $c$ increases, and this rate becomes very small for moderate to large $c$. For example, for $c = 0, 1$, and 2, the rejection rates are 0.24, 0.12, and 0.07, respectively.
2.3 Variance Reduction Methods

Here we briefly describe a few techniques commonly used for variance reduction in Monte Carlo computations. More detailed descriptions can be found in standard Monte Carlo books (Hammersley and Handscomb 1964, Rubinstein 1981).

**Stratified Sampling.** It is a powerful and commonly used technique in population survey and is also very useful in Monte Carlo computations. Mathematically, this method can be viewed as a special importance sampling method with its trial density \( g(x) \) constructed as a piecewise constant function. Suppose we are interested in estimating \( \int_X f(x)dx \). If possible, we want to break the region \( X \) into the union of \( k \) disjoint subregions, \( D_1, \ldots, D_k \), so that within each subregion, the function \( f(x) \) is relatively “homogeneous” (e.g., close to being a constant). Then, we can spend \( m_i \) random samples, \( X^{(i,1)}, \ldots, X^{(i,m_i)} \), in the subregion \( D_i \), and approximate the subregional integral \( \int_{D_i} f(x)dx \) by

\[
\hat{\mu}_i = \frac{1}{m_i} [f(X^{(i,1)}) + \cdots + f(X^{(i,m_i)})].
\]

The overall integral \( \mu \) can be approximated by

\[
\hat{\mu} = \hat{\mu}_1 + \cdots + \hat{\mu}_k,
\]

whose variance is easily calculated as

\[
\text{var}(\hat{\mu}) = \frac{\sigma_1^2}{m_1} + \cdots + \frac{\sigma_k^2}{m_k},
\]

where \( \sigma_i^2 \) is the variation of \( f(x) \) in region \( D_i \). In contrast, if we use all of the \( m = m_1 + \cdots + m_k \) samples to do a plain uniform sampling in the region \( X \), the variance of the estimate would be \( \sigma^2/n \), with \( \sigma^2 \) being the overall variation of \( f(x) \) in \( X \).

Clearly, if we fail to have relatively homogeneous \( f(x) \) in each region \( D_i \) (in other words, if \( \sigma_i^2 \) is not much different from \( \sigma^2 \)), stratified sampling actually makes the computation less accurate than a plain Monte Carlo. The moral is this: There is no free lunch, and one needs to think carefully before adopting any advanced techniques.

**Control Variates Method.** In this method, one uses a control variate \( C \), which is correlated with the sample \( X \), to produce a better estimate. Suppose the estimation of \( \mu = E(X) \) is of interest and \( \mu_C = E(C) \) is known. Then, we can construct Monte Carlo samples of the form

\[
X(b) = X + b(C - \mu_C),
\]

which have the same mean as \( X \), but a new variance

\[
\text{var}(X(b)) = \text{var}(X) - 2bcov(X, C) + b^2\text{var}(C),
\]
If the computation of \( \text{cov}(X, C) \) and \( \text{var}(C) \) is easy, then we can let \( b = \text{cov}(X, C)/\text{var}(C) \), in which case

\[
\text{var}\{X(b)\} = (1 - \rho^2_{XC})\text{var}(X) < \text{var}(X).
\]

Another situation is when we know only that \( E(C) \) is equal to \( \mu \). Then, we can form \( X(b) = bX + (1 - b)C \). It is easy to show that if \( C \) is correlated with \( X \), we can always choose a proper \( b \) so that \( X(b) \) has a smaller variance than \( X \). Extensions to more than one control variate are also useful in Monte Carlo computations, but are omitted in this book.

**Antithetic Variates Method.** This method is due to Hammersley and Morton (1956), where they describe a way of producing negatively correlated samples. Suppose \( U \) is the random number used in the production of a sample \( X \) that follows a distribution with \( \text{cdf} \ F [\text{i.e., } X = F^{-1}(U)] \) according to Lemma2.1.1. Then, \( X' = F^{-1}(1-U) \) also follows distribution \( F \). More generally, if \( g \) is a monotonic function, then

\[
\{g(u_1) - g(u_2)\} \{g(1 - u_1) - g(1 - u_2)\} \leq 0
\]

for any \( u_1, u_2 \in [0,1] \). For two independent uniform random variables \( U_1 \) and \( U_2 \) (in fact, it is only required that the two are i.i.d. with symmetric density in \([0,1]\)), we have

\[
E[\{g(U_1) - g(U_2)\} \{g(1 - U_1) - g(1 - U_2)\}] = \text{cov}(X, X') \leq 0,
\]

where \( X = g(U) \) and \( X' = g(1 - U) \). Thus, \( \text{var}((X + X')/2) \leq \text{var}(X)/2 \), implying that using the pair \( X \) and \( X' \) is better than using two independent Monte Carlo draws for estimating \( E(X) \).

**Rao-Blackwellization.** This method reflects a basic principle (or rule of thumb) in Monte Carlo computation: One should carry out analytical computation as much as possible. The problem can be formulated as follows: Suppose we have drawn independent samples \( x^{(1)}, \ldots, x^{(m)} \) from the target distribution \( \pi(x) \) and are interested in evaluating \( I = E_x h(x) \). A straightforward estimator is

\[
\hat{I} = \frac{1}{m} \left\{ h(x^{(1)}) + \cdots + h(x^{(m)}) \right\}.
\]

Suppose, in addition, that \( x \) can be decomposed into two parts \( (x_1, x_2) \) and that the conditional expectation \( E[h(x) \mid x_2] \) can be carried out analytically. An alternative estimator of \( I \) is

\[
\tilde{I} = \frac{1}{m} \left\{ E[h(x) \mid x_2^{(1)}] + \cdots + E[h(x) \mid x_2^{(m)}] \right\}.
\]
Clearly, both $\hat{I}$ and $\tilde{I}$ are unbiased\(^1\) because of the simple fact that

$$E_{\pi} h(x) = E_{\pi} [E(h(x) \mid x_2)].$$

If the computational effort for obtaining the two estimates are the same, then $\tilde{I}$ should be preferred because

$$\text{var} \{ h(x) \} = \text{var} \{ E[h(x) \mid x_2] \} + \text{var} \{ h(x) \mid x_2 \},$$

which implies that

$$\text{var}(\tilde{I}) = \frac{\text{var} \{ h(x) \}}{m} \geq \frac{\text{var} \{ E[h(x) \mid x_2] \}}{m} = \text{var}(\hat{I}).$$

In statistics, $\tilde{I}$ is often called the “histogram estimator” or the empirical estimator and $\hat{I}$ the “mixture estimator.” Statisticians find that by conditioning an inferior estimator on the value of sufficient statistics, one can obtain the optimal estimator. This procedure is often referred to as Rao-Blackwellization (Bickel and Doksum 2000). Some other uses of Rao-Blackwellization in Monte Carlo estimations can be found in Casella and Robert (1996). More discussions on this issue can be found in Section 2.5.5 and Chapter 6.

### 2.4 Exact Methods for Chain-Structured Models

An important probability distribution used in many applications has the following form:

$$\pi(x) \propto \exp \left\{-\sum_{i=1}^{d} h_i(x_{i-1}, x_i)\right\}, \quad (2.1)$$

where $x = (x_0, x_1, \ldots, x_d)$. This type of model can be seen as having a “Markovian structure” because the conditional distribution $\pi(x_i \mid x_{i-1} \tilde{q})$, where $x_{i-1} \tilde{q} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$, depends only on the two neighboring variables $x_{i-1}$ and $x_{i+1}$; that is,

$$\pi(x_i \mid x_{i-1} \tilde{q}) \propto \exp \left\{-h(x_{i-1}, x_i) - h(x_i, x_{i+1})\right\}.$$

The unobserved state variables in a state-space model (Section 1.6) can clearly be represented in this form, which can also be depicted by the following graph:

\(^1\)An estimator $\hat{\mu}$ is called an unbiased estimator of $\mu$ if $E_{\mu} \hat{\mu} = \mu$. In words, this means that the average behavior of $\hat{\mu}$ is “on target.”
2.4 Exact Methods for Chain-Structured Models

When \( x_0, \ldots, x_d \) are discrete random variables taking values in a finite set \( S = \{s_1, \ldots, s_k\} \), this structure is often referred to as the hidden Markov model (HMM) and we can do many things with it. First, the “dynamic programming” (DP) method (Bellman 1957) can be used to find the global maximum of \( \pi(x) \) and its maximizer \( \hat{x} \) with \( O(dk^2) \) operations. Second, an algorithm of the same order as the DP exists for finding the marginal distribution of each \( x_i \) and drawing “exact” random samples from \( \pi(x) \). Clearly, these exact algorithms are only practical when \( k \), the number of distinctive values that \( x_i \) can take, is not too large.

2.4.1 Dynamic programming

Suppose each \( x_i \) in \( x \) only takes values in set \( S = \{s_1, \ldots, s_k\} \). Maximizing the distribution \( \pi(x) \) in (2.1) is equivalent to minimizing its exponent

\[
H(x) = h_1(x_0, x_1) + \cdots + h_d(x_{d-1}, x_d).
\]

A recursive procedure can be carried out:

- Define
  \[
  m_1(x) = \min_{s_i \in S} h_1(s_i, x), \text{ for } x = s_1, \ldots, s_k.
  \]

- Recursively compute the function
  \[
  m_l(x) = \min_{s_i \in S} \{m_{l-1}(s_i) + h_l(s_i, x)\}, \text{ for } x = s_1, \ldots, s_k.
  \]

- The optimal value \( H(x) \) is attained by \( \min_{s \in S} m_d(s) \).

It is not difficult to see that the minimum of \( m_1(x) \) is equal to the minimum of \( h_1(x_0, x_1) \). By induction, one can easily argue that

\[
\min_{s \in S} m_1(x) = \min_{x_0, \ldots, x_1 \in S} [h_1(x_0, x_1) + \cdots + h_l(x_{l-1}, x_l)].
\]

Thus, the above procedure indeed minimizes the target function \( H(x) \).

To find out which \( x \) gives rise to the global minimum of \( H(x) \), we can trace backward as follows:

- Let \( \hat{x}_d \) be the minimizer of \( m_d(x) \); that is,
  \[
  \hat{x}_d = \arg \min_{s_i \in S} m_d(s_i).
  \]

Break ties arbitrarily.
2. Basic Principles: Rejection, Weighting, and Others

- For \( t = d - 1, d - 2, \ldots, 1 \), we let
  \[
  \hat{x}_t = \arg \min_{s_i \in S} \{ m_t(s_i) + h_{t+1}(s_i, \hat{x}_{t+1}) \}.
  \]

Break ties arbitrarily.

Configuration \( \hat{x} = (\hat{x}_1, \ldots, \hat{x}_d) \) obtained by this method is the minimizer of \( H(x) \).

2.4.2 Exact simulation

The first step for simulating from \( \pi(x) \) in (2.1) is to draw \( x_d \) from its marginal distribution. This requires us to marginalize \( x_1, \ldots, x_{d-1} \) in the joint distribution \( \pi(x) \). After we have drawn \( x_d \), we can work our way backward recursively; that is, sampling \( x_d \) from \( \pi(x_d | x_{d-1}) \); \( x_{d-1} \) from \( \pi(x_{d-1} | x_{d-2}) \); and so on. The principle behind the marginalization step is based on the observation that the overall summation can be decomposed into recursive steps; that is,

\[
Z = \sum_x \exp\{-H(x)\} = \sum_{x_d} \left[ \cdots \left[ \sum_{x_1} \left\{ \sum_{x_0} e^{h_1(x_0, x_1)} \right\} e^{h_2(x_1, x_2)} \right] \cdots \right].
\]

More precisely, the following recursions similar to those in DP can be carried out by a computer:

- Define \( V_1(x) = \sum_{x_0 \in S} e^{h_1(x_0, x)} \).
- Compute recursively for \( t = 2, \ldots, d \):
  \[
  V_t(x) = \sum_{y \in S} V_{t-1}(y) e^{h_t(y, x)}. \tag{2.2}
  \]

Then, the partition function is \( Z = \sum_{x \in S} V_d(x) \) and the marginal distribution of \( x_d \) is \( \pi(x_d) = V_d(x_d) / Z \). To simulate \( x \) from \( \pi \), we can do the following:

- Draw \( x_d \) from \( S \) with probability \( V_d(x_d) / Z \);
- For \( t = d - 1, d - 2, \ldots, 1 \), we draw \( x_t \) from distribution
  \[
  p_t(x_t) = \frac{V_t(x_t) e^{h_{t+1}(x_t, x_{t+1})}}{\sum_{y \in S} V_{t-1}(y) e^{h_{t+1}(y, x_{t+1})}}.
  \]

The random sample \( x = (x_1, \ldots, x_d) \) obtained in this way follows the distribution \( \pi(x) \).
As an example, we can use the forward-recursion formula (2.2) to compute the partition function for a one-dimensional Ising model

\[
\pi(x) = Z^{-1} \exp\{\beta(x_0 x_1 + \cdots + x_{d-1} x_d)\},
\]

where \( x_i \) takes value in \( S = \{-1, 1\} \). First, we have

\[
V_i(x) = e^{\beta x} + e^{-\beta x} \equiv e^{\beta} + e^{-\beta},
\]

which is a constant. Applying the recursion, we easily obtain that

\[
V_i(x) = (e^{\beta} + e^{-\beta})^i
\]

and \( Z = 2(e^{\beta} + e^{-\beta})^d \). Details for an exact simulation from this distribution are left to the reader.

An important feature of the target distribution \( \pi(x) \) treated in this section is that it can be written as

\[
\pi(x) \propto \exp\left\{ - \sum_{C \in \mathcal{C}} h(x_C) \right\},
\]

where \( \mathcal{C} \) is the set of some subsets of \( \{1, \ldots, d\} \) and \( x_C = (x_i, i \in C) \). In Lauritzen and Spiegelhalter (1988), each subset \( C \) in \( \mathcal{C} \) is called a “clique.” Any two subsets \( C_1 \) and \( C_2 \) are said to “connected” if they share at least one common component. Any probability distribution that possesses this dependency structure is termed a graphical model. Our model in this section can be seen as a special graphical model in which the set of cliques is \( \mathcal{C} = \{C_1, \ldots, C_d\} \), where \( C_i = \{i-1, i\} \). When \( \mathcal{C} \) forms a “tree” and each clique does not have too many vertices (components), one can derive efficient algorithms for optimization and exact simulation similar to the algorithms described in this section (Lauritzen and Spiegelhalter 1988). There does not seem to be a common name for this sampling method. Some people call it the peeling algorithm because this method was first developed for a genetic linkage problem (Cannings, Thompson and Skolnick 1978). Some others refer it as the forward-summation-backward-sampling method, a rather awkward name. We prefer to use the name propagation method for the reason that both the forward and the backward steps can be thought of as propagating information along the chain graph.

### 2.5 Importance Sampling and Weighted Sample

#### 2.5.1 An example

Suppose we wish to evaluate the quantity

\[
\theta = \int_X h(x)\pi(x)dx = E_\pi[h(X)],
\]
where the support of the random variable $X$ is denoted as $\mathcal{X}$ and $h(x) \geq 0$. In standard numerical methods, we discretize the domain $\mathcal{X}$ by regular grids, evaluate $h(x)\pi(x)$ on each of the grid points, and then use the Riemann sum as an approximation. Consider the target function given by Figure 2.1(a):

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

where $(x, y) \in [-1, 1] \times [-1, 1]$. More than two-thirds of computing time are wasted on evaluating those grid points on which the function is virtually zero. It is easy to imagine that the situation deteriorates very rapidly as the dimension of space $\mathcal{X}$ increases.

![Figure 2.1](image_url)

**FIGURE 2.1.** (a) The target function whose integral is of interest. (b) A possible trial distribution $g(x, y)$ for applying importance sampling.

By taking $m$ random samples, $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, uniformly in $[-1, 1] \times [-1, 1]$, we implemented a vanilla Monte Carlo algorithm to estimate the integral $\mu = \int \int f(x, y)dx dy$. Because the density for the sampling distribution is a constant, 1/4, in the region, the estimate of the integral was produced as

$$\hat{\mu} = \frac{4}{m} \left\{ f^{(1)} + \cdots + f^{(m)} \right\},$$

where $f^{(i)} = f(x^{(i)}, y^{(i)})$. With $m=2,500$, we obtained $\hat{\mu} = 0.1307$, with a standard deviation 0.009, which was estimated by

$$\text{std}(\hat{\mu}) = \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{m} (f_i - \hat{\mu})^2}.$$

Clearly, this vanilla Monte Carlo algorithm suffers a similar problem as its deterministic counterpart: It wastes a lot of effort in evaluating random samples located in regions where the function value is almost zero. Although the theoretical convergence rate is $m^{-1/2}$ for essentially all Monte Carlo methods, it is the constant in front of this rate that makes a huge difference in a real problem.
2.5.2 The basic idea

The importance sampling idea (Marshall 1956) suggests that one should focus on the region(s) of “importance” so as to save computational resources. Although it may not seem so important in the toy example shown earlier, the idea of biasing toward “importance” regions of the sample space becomes essential for Monte Carlo computation with high-dimensional models such as those in statistical physics, molecular simulation, and Bayesian statistics. In high-dimensional problems, the region in which the target function is meaningfully nonzero compared with the whole space \( \mathcal{X} \) is just like a needle compared with a haystack. Vanilla Monte Carlo schemes (e.g., sampling uniformly from a regular region) are bound to fail in these problems.

Suppose one is interested in evaluating

\[
\mu = E_\pi \{ h(x) \} = \int h(x) \pi(x) dx.
\]

The following procedure is a simple form of the importance sampling algorithm:

(a) Draw \( x^{(1)}, \ldots, x^{(m)} \) from a trial distribution \( g(\cdot) \).

(b) Calculate the importance weight

\[
w^{(j)} = \frac{\pi(x^{(j)})}{g(x^{(j)})}, \quad \text{for } j = 1, \ldots, m.
\]

(c) Approximate \( \mu \) by

\[
\hat{\mu} = \frac{\sum_{j=1}^{m} w^{(j)} h(x^{(j)})}{\sum_{j=1}^{m} w^{(j)}}.
\]

Thus, in order to make the estimation error small, one wants to choose \( g(x) \) as “close” in shape to \( \pi(x) h(x) \) as possible. A major advantage of using (2.3) instead of the unbiased estimate,

\[
\tilde{\mu} = \frac{1}{m} \left\{ w^{(1)} h(x^{(1)}) + \cdots + w^{(m)} h(x^{(m)}) \right\},
\]

is that in using the former, we need only to know the ratio \( \pi(x)/g(x) \) up to a multiplicative constant; whereas in the latter, the ratio needs to be known exactly. Additionally, although inducing a small bias, (2.3) often has a smaller mean squared error than the unbiased one \( \hat{\mu} \).

Another scenario for resorting to importance sampling is when we want to generate i.i.d. random samples from \( \pi \) but doing so directly is infeasible. In this case, we may generate random samples from a different, but similar, trivial distribution \( g(\cdot) \), and then correct the bias by using the importance sampling procedure. Similar to the rejection method, a successful
application of importance sampling in this case requires that the sampling distribution \( g \) is reasonably close to \( \pi \); in particular, that \( g \) has a longer tail than \( \pi \). Note that finding a good trial distribution \( g \) can be a major — and sometimes impossible — undertaking in high-dimensional problems.

Alternatively, we can opt for correlated samples produced by running a Markov chain whose stationary distribution is \( \pi \). This methodology is referred to as Markov chain Monte Carlo (MCMC) throughout the book. A very general recipe for designing a proper Markov chain was first proposed by Metropolis et al. (1953) and has been subject to active research in the past few decades. We will discuss this class of methods in the latter part of this book (Chapters 5–11).

Let us illustrate the importance sampling method with the toy example shown in Figure 2.1. After a visual examination of function \( f(x,y) \), we decided to use a distribution \( g(x,y) \), which is of the form

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

with \((x,y) \in [-1,1] \times [-1,1]\). This is a truncated mixture of Gaussian distributions:

\[
0.46\mathcal{N}\left(\begin{pmatrix} 0.5 \\ -0.1 \end{pmatrix}, \begin{pmatrix} 0.180 & 0 \\ 0 & 0.20 \end{pmatrix}\right) + 0.54\mathcal{N}\left(\begin{pmatrix} -0.4 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0.180 & 0 \\ 0 & 0.120 \end{pmatrix}\right)
\]

We can sample from this mixture distribution by a two-step procedure: (a) a biased coin (with probability 0.464 of showing heads) is first tossed; (b) if the head turns up, we draw a random vector from the first Gaussian distribution, otherwise, we draw from the second Gaussian distribution. The integral can then be estimated by averaging the ratios \( w(x,y) = f(x,y)/g(x,y) \), with \( w = 0 \) when \((x,y)\) falls out of the region \( \mathcal{X} \). With \( m=2500 \), our estimate of \( \mu \) is 0.1259, with a standard error 0.0005.

### 2.5.3 The “rule of thumb” for importance sampling

Importance sampling suggests estimating \( \mu = E_\pi\{h(x)\} \) by first generating independent samples \( x^{(1)}, \ldots, x^{(m)} \) from an easy-to-sample trial distribution, \( g(\cdot) \), and then correcting the bias by incorporating the importance weight \( w^{(j)} \propto \pi(x^{(j)})/g(x^{(j)}) \) in estimation using either (2.3) or 2.4. By properly choosing \( g(\cdot) \), one can reduce the variance of the estimate substantially. A good candidate for \( g(\cdot) \) is one that is close to the shape of \( h(x)\pi(x) \). Therefore, the importance sampling method can be super-efficient; that is, the resulting variance of \( \hat{\mu} \) can be smaller than that obtained using independent samples from \( \pi \). The method is generalized to the case of, say, evaluating \( E_\pi\{h(x)\} \) when sampling from \( \pi(\cdot) \) directly is difficult but generating from \( g(\cdot) \) and computing the importance ratio \( w(x) = \pi(x)/g(x) \) (up to a multiplicative constant) are easy. The efficiency of such a method is then difficult to measure. A useful “rule of thumb” is to use the effective