4
Sequential Monte Carlo in Action

The previous chapter outlines a general Monte Carlo framework based on the sequential buildup strategy. Several essential elements are (a) the choice of the trial densities, (b) the resampling method, (c) the marginalization strategy, and (d) the rejection control. This chapter will illustrate how these generic strategies are applied to various application problems.

4.1 Some Biological Problems

4.1.1 Molecular Simulation

Simulating molecular structures is one of the most important and challenging scientific problems. Starting with those simple SAW model for long-chain polymers (Section 3.1), chemists and structural biologists have developed numerous lattice-based models (and also more complicated models) for predicting native structures of important macromolecules, such as protein molecules. Here, we offer the reader a glimpse of this huge area.

The most well-known open problem in structural biology and biophysics is the the so-called *protein folding problem* in which one is required to predict the three-dimensional fold shape of a protein molecule based only on its sequence of amino acids. One simple model to imitate the real protein folding process is a 2-D or 3-D lattice-bead model. These models have identical structures as the SAW model described in Section 3.1, but they usually use a more complicated function for interactive energy between the "beads" on the lattice. In protein language, these beads correspond

to amino acid residues, which are of 20 different types (i.e., a 20-letter alphabet).

One of the simplest models used by chemists (Unger and Moult 1993) has only two different kinds of beads, white and black, corresponding to hydrophilic and hydrophobic residues, respectively. An example of such a simple bead sequence of length 36 is

WWWBBWWBBWWWWWBBBBBBBWWBBWWWWBBWWBWW

It is of interest to find out its most "favorable fold" in a 2-D lattice space. To define what is meant by a favorable fold, we define an energy function for each configuration of this bead sequence:

$$U_n(\mathbf{x}_n) = - \sum_{|i-j|>1} c(x_i, x_j),$$

where $c(x_i, x_j) = 1$ if x_i and x_j are non-bonding neighbors and the identities of beads i and j are both black (hydrophobic), and $c(x_i, x_j) = 0$ otherwise. Clearly, this simple model favors the close packing of hydrophobic residues and, to certain extent, mimics some aspects of a real protein fold.[1]

With a target distribution

$$\pi_n(\mathbf{x}_n) \propto \exp\{-U_n(\mathbf{x}_n)/2\},$$

we applied the SIS, with various modifications, including resampling, rejection control, and one-step-look-ahead, to all the examples in Unger and Moult (1993). In light of a recent result of Bastolla, Frauenkron, Gerstner, Grassberger and Nadler (1998), it is not surprising that we were able to find the same or better minimum energy state than Unger and Moult (1993). In fact, Bastolla et al. (1998) applied the SIS method with pruning and enrichment modifications to the examples of Unger and Moult and many others and achieved some excellent results. Our results are comparable to theirs. Figure 4.1 (b) shows the best configuration we found for a 60-mer chain: it has 36 favorable contacts. In comparison, the best configuration found by Unger and Moult has only 34 favorable contacts. An additional experiment we did was to repeat the modified SIS simulation of this 60-mers (each took about 6 minutes) for 125 times, from which we obtained an estimate of the normalizing constant $\log(Z) = 89.65$ and the mean squared extension $E_\pi(R^2) \approx 157$.

To further investigate the effect of resampling and partial rejection controls, we applied the SIS to an earlier example of simulating SAWs in which

[1] Because most proteins are surrounded by water, these macromolecules tend to pack hydrophobic residues in the center of their fold and leave those hydrophilic ones on the surface to interact with water molecules.

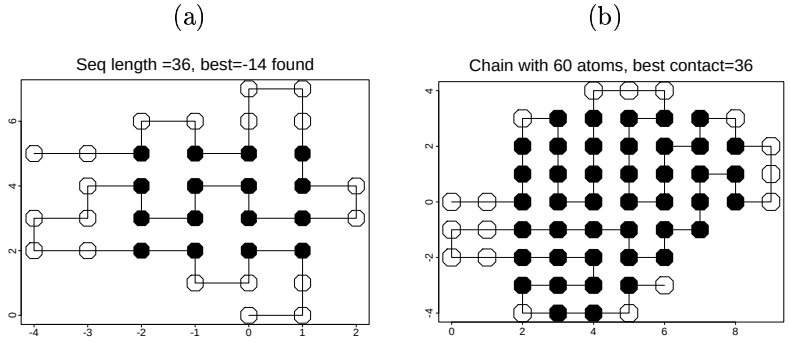

FIGURE 4.1. Configurations of two black-white bead chains with length 36 and 60, respectively. Configuration (a) has the minimum energy (verified by Unger and Moult), whereas (b) has a total favorable contacts of 36, the optimal one we have obtained. Its true minimum energy is unknown.

many accurate results have been obtained. The two-step-look-ahead strategy can increase simulation efficiency greatly, as shown by our example. We simulated a chain with $N = 100$ particles (99 links) and estimated the log-partition function, $\log Z_n \approx 96.39 \pm 0.037$. A numerical approximation formula gives

$$\log Z_n \approx (N-1) * \log(2.6385) + \left(\frac{43}{32} - 1\right) \log(N-1) - \log(4) = 96.24.$$

In all these simulations, either the original growth method (Rosenbluth and Rosenbluth 1955, Kong et al. 1994), which does not involve resampling and rejection control, or the bootstrap filtering method (Gordon et al. 1993), which resamples at every step, fails.

4.1.2 Inference in population genetics

In Section 2.7, we introduced a simple demographic model (also can be use for inferring phylogeny) for inferring relationships among different species based on comparisons of homologous DNA segments. The computational method described there is due to Griffiths and Tavare (1994) and can be seen as a special SIS method. The general method of resampling discussed in Section 3.4.4 can be applied to improve such computation (Chen and Liu 2000b).

As with many SIS applications, the trial distribution for simulating the evolutionary history $\mathcal{H}$ (Section 2.7) has the form

$$g(\mathcal{H}) = \prod_{t=1}^{k} g_t(H_{-t} \mid H_{-t+1}).$$

We define the *current weight* (for $t \leq k$) for this trial density:

$$w_{-t} = \frac{p_\theta(H_{-t+1}|H_{-t}) \cdots p_\theta(H_0 \mid H_{-1})}{g_t(H_{-t}|H_{-t+1}) \cdots g_1(H_{-1} \mid H_0)} \equiv w_{-t+1} \frac{p_\theta(H_{-t+1}|H_{-t})}{g_t(H_{-t}|H_{-t+1})}.$$

The final weight is then $w = w_{-k}\ p_\theta(H_{-k}) p_\theta(\text{stop} \mid H_0)$, where the last term is same as in (2.24).

In a parallel implementation of SIS, we first generate m samples from $q_\theta(H_{-1}|H_0)$, and then recursively generate $\{H_{-t}^{(1)}, \ldots, H_{-t}^{(m)}\}$, called the *current sample*, for $t = 2, 3, \ldots$, until coalescence in all m processes. Along with producing the current sample, we could also monitor the current weight and incur resampling steps at any time $-t$ when the coefficient of variation in $\{w_{-t}^{(1)}, \ldots, w_{-t}^{(m)}\}$ exceeds a threshold B. In resampling, one produces a new current sample by drawing with replacement from $\{H_{-t}^{(1)}, \ldots, H_{-t}^{(m)}\}$ according to probability $\propto \{w_{-t}^{(1)}, \ldots, w_{-t}^{(m)}\}$. The weight for each new sample after resampling is set as the *sample average* of the $w_{-t}^{(j)}$ so as to ensure that at the end we obtain a proper estimate of the likelihood function.

We note, however, that resampling among $\{H_{-t}^{(1)}, \ldots, H_{-t}^{(m)}\}$ is inefficient because these samples differ greatly in their coalescence speeds. Those $H_{-t}^{(j)}$ that have fast coalescence speeds (small population sizes) often have small current weights, but large final weights. Resampling among the $H_{-t}^{(m)}$ actually prunes away many "good" samples. To address this problem, we propose to conduct resampling at the same *coalescence time* instead of the same sequential sampling time. In other words, we wait until all the m processes reach the same population size, say, i, and then resample from $\{H_{-i_1}^{(1)}, \ldots, H_{-i_m}^{(m)}\}$, where $i_j = \min\{t : \ |H_{-t}^{(j)}| = i\}$. (Here, $|H_{-i}|$ denotes the population size of that generation.) Although early histories, $H_{-s}^{(j)}$, for $s < i_j$, are not needed in sequential sampling (due to a Markovian structure), we still need to keep all the early histories of those processes that survive the resampling.

The new resampling procedure can be implemented as follows. Suppose m parallel coalescence processes have been started as in Section 2.7. Then, for $i = n-1, \ldots, 1$:

- for $j = 1, \ldots, m$, we run the jth process until its population size first reaches i; denote this time as $-i_j$;
- compute the coefficient of variation for $w_{-i_1}^{(1)}, \ldots, w_{-i_m}^{(m)}$; name this number CV_i;
- do resampling among $\{H_{-i_1}^{(1)}, \ldots, H_{-i_m}^{(m)}\}$ when CV_i is greater than a threshold, otherwise continue the usual sequential sampling;
- the weight for each new sample after resampling is set as the sample average of the $w_{-i_j}^{(j)}$.

At the end of this procedure, we produce a sample of histories $\mathcal{H}^{(*j)}$, $j = 1, \ldots, m$, and their associated weights $w^{(*j)}$, $j = 1, \ldots, m$. As in a standard SIS, we use the sample average of these weights,

$$\frac{1}{m}\left(w^{(*1)} + \cdots + w^{(*m)}\right),$$

to estimate the likelihood function $p_\theta(H_0)$. If we are interested in estimating $p_{\theta'}(H_0)$ for $\theta' \neq \theta$, we can use a modified weighted average

$$\hat{p}_{\theta'}(H_0) = \frac{1}{m}\left[w^{(*1)}\frac{p_{\theta'}(\mathcal{H}^{(*1)})}{p_\theta(\mathcal{H}^{(*1)})} + \cdots + w^{(*m)}\frac{p_{\theta'}(\mathcal{H}^{(*m)})}{p_\theta(\mathcal{H}^{(*m)})}\right].$$

Chen and Liu (2000b) applied this modified resampling step to the trial distribution described in Section 2.7. To compare with other approaches, the new method was implemented for a numerical example in Stephens and Donnelly (2000) (also treated in Section 2.7). With sample size $m = 10,000$ and bound $B = 4$, two resampling steps were incurred. We repeated this exercise five times and the results were rather similar. The extra computational cost was negligible. Figure 4.2(b) displays the likelihood curves estimated from five independent replications of our method. For comparison, Figure 4.2(a) shows the results (with $m = 10,000$) obtained by the plain SIS method discussed in Section 2.7. Figure 4.2(b) is almost indistinguishable from Figure 3(b) of Stephens and Donnelly (2000), which is produced by using a more efficient trial function. We note that the resampling modification described in this section can also be applied to their method.

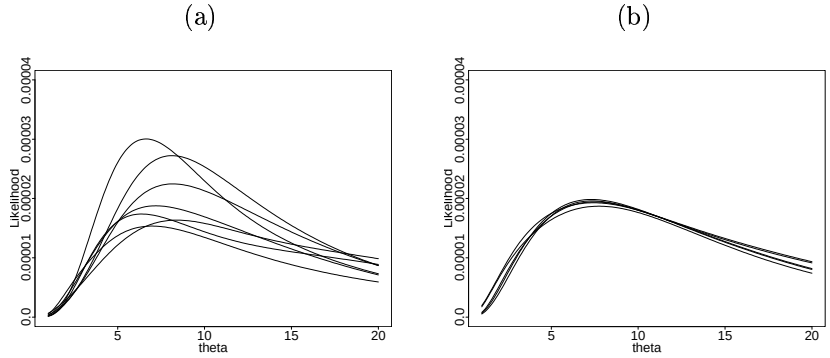

FIGURE 4.2. The estimated likelihood curve for a small dataset in Stephens and Donnelly (2000). (a) Seven independent runs of the plain SIS method of Griffiths and Tavare; (b) five independent runs of SIS with resampling.

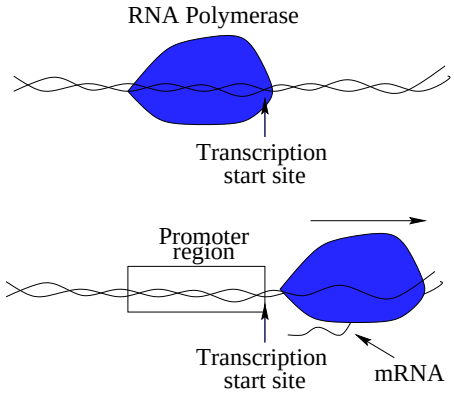

FIGURE 4.3. Cartoon illustration of gene transcription. Top figure: RNA polymerase binds to the promoter region and is about to start the transcription. Bottom figure: transcription is started when the RNA polymerase zips along the genome resulting in messenger RNA.

4.1.3 Finding motif patterns in DNA sequences

As we have briefly discussed in Section 1.5, all the hereditary information of a living thing is stored in its genome, which can be represented by linear sequences of letters from a four-letter alphabet (A, T, G, and C). Segments of the genome of 100 to 10,000 DNA bases long, called genes, code for proteins, which are responsible for almost all functions of life. An entire genome containing millions or billions of DNA bases and may encode tens of thousands of proteins. The process of transforming the information contained in a gene into its product (protein) is fairly complicated. First of all, the information of a gene has to be *transcribed* (copied) from the genome to messenger RNA (mRNA) by a molecular complex known as *RNA polymerase*. Then, these mRNAs are translated to proteins via *transfer RNAs* (tRNA). Free RNA polymerase molecules collide with the chromosomes at random positions, sliding along it but sticking only weakly to most DNA segments. When it meets a specific DNA sequence called the *promoter*, which signals the start of RNA synthesis, the polymerase binds tightly and starts the transcription. The promoter for a bacteria gene is located just "upstream" (at -10 and -35) of the transcriptional start site of the gene (Alberts, Bray, Lewis, Raff, Roberts and Watson 1994). Note that the transcriptional start site is different from the start of the gene (located further downstream).

It is an amazing fact that every single cell of an organism carries a complete copy of the full genome. However, diverse cells within an individual differ drastically (for example, cells in one's eyes and cells on one's skin have completely different forms and functions). The process by which cells acquire special characteristics, called differentiation, occurs because the ex-

pression of genes is regulated by various mechanisms. In single-cell organisms, genes are regulated so as to allow the cell to respond to environmental changes. A particular important form of gene regulation is achieved by proteins (often called *regulatory proteins*) bound to sites within or close to the promoter of a gene located in its upstream (5′ end) non-coding regions. which then alters the rate of RNA polymerase binding to a transcriptional starting site. A protein bound to a site within the promoter region will prevent RNA polymerase from starting transcription. If the regulatory protein binds further upstream, it may enhance gene expression by attracting RNA polymerases. The locations on the genome where the regulatory proteins bind are called the regulatory *binding sites* and these sites are composed of one or more short DNA segments of 20 base pairs long (some are even shorter).

cole1	taatgtttgtgctggtttttgtggcatcgggcgagaatagcgcgtggtgtgaaagactgtttttttgatcgttttcacaaaaatggaagtccacagtcttgacag
ecoarabop	gacaaaaacgcgtaacaaaagtgtctataatcacggcagaaaagtccacattgattatttgcacggcgtcacactttgctatgccatagcatttttatccataag
ecobglr1	acaaatcccaataacttaatttattgggatttgttatatataactttataaattcctaaaattacacaaagttaataactgtgagcatggtcatatttttatcaat
ecocrp	cacaaagcgaaagctatgctaaaacagtcaggatgctacagtaatacattgatgtactgcatgtatgcaaaggacgtcacattaccgtgcagtacagttgatagc
ecocya	acggtgctacacttgtatgtagcgcatctttctttacggtcaatcagcatggtgttaaattgatcacgttttagaccatttttttcgtcgtgaaactaaaaaaacc
ecodecop	agtgaattatttgaaccagatcgcattacagtgatgcaaacttgtaagtagatttccttaatttgtgatgtgtatcgaagtgtgttgcggagtagatgttagaata
ecogale	gcgcataaaaaacggctaaattcttgtgtaaacgattccactaatttattccatgtcacactttttcgcatctttgttatgctatggttatttcataccatataagcc
ecoilvbpr	gctccggcgggggttttttgttatctgcaattcagtacaaaacgtgatcaacccctcaattttccctttgctgaaaaattttccattgtctcccctgtaaagctgt
ecolac	aacgcaattaatgtgagttagctcactcattaggcaccccaggctttacactttatgcttccggctcgtatgttgtgtggaattgtgagcggataacaatttcac
ecomale	acattaccgccaattctgtaacagagatcacacaaagcgacggtggggcgtaggggcaaggaggatggaaagaggttgccgtataaagaaactagagtccgttta
ecomalk	ggaggaggcgggaggatgagaacacggcttctgtgaactaaaccgaggtcatgtaaggaatttcgtgatgttgcttgcaaaaatcgtggcgatttttatgtgcgca
ecomalt	gatcagcgtcgttttaggtgagttgttaataaagatttggaattgtgacacagtgcaaatttcagacacataaaaaaacgtcatcgcttgcattagaaaggtttct
ecoompa	gctgacaaaaaagattaaacatacctttatacaagacttttttttcatatgcctgacggagttcacacttgtaagttttcaactacgttgtagactttacatcgcc
ecotnaa	ttttttaaacattaaaattcttacgtaatttataatctttaaaaaaagcatttaatattgctccccgaacgattgtgattcgattcacatttaaacaatttcaga
ecouxu1	cccatgagagtgaaattgttgtgatgtggtttaacccaattagaattcgggattgacatgtcttaccaaaaggtagaacttatacgccatctcatccgatgcaagc
pbr-p4	ctggcttaactatgcggcatcagagcagattgtactgagagtgcaccatatgcggtgtgaaataccgcacagatgcgtaaggagaaaataccgcatcaggcgctc
trn9cat	ctgtgacggaagatcacttcgcagaataaataaatcctggtgtccctgttgataccgggaagccctgggccaacttttggcgaaaatgagacgttgatcggcacg
(tdc)	gatttttatactttaacttgttgatatttaaaggtatttaattgtaataacgatactctggaaagtattgaaagttaatttgtgagtggtcgcacatatcctgtt

TABLE 4.1. A dataset of 18 DNA segments, each of 105 base pairs long, taken from the upstream noncoding regions of 18 genes of *E. coli*. This dataset was created by Stormo and Hartzell (1989) who used a greedy sequential buildup strategy to find the binding sites.

The cyclic receptor protein (CRP) is a positive control factor necessary for the expression of catabolite repressible genes. Table 4.1 shows a set of DNA segments, each 105 bases long, cut from upstream non-coding regions of 18 genes of *E. coli*. It is known that there is at least one CRP-binding site in each of the 18 segments and the location of these binding sites have been experimentally determined (Stormo and Hartzell 1989). The width of

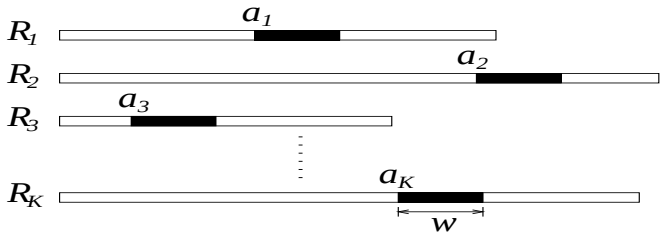

FIGURE 4.4. A schematic plot for finding subtle sites in multiple sequences. The blackened segment in each sequence represents the "common pattern" expected from these sequences; both its location and its content are unknown.

the binding motif is decided at $w = 20$. So this dataset allows one to test the ability of various new algorithms.

We can abstract the motif finding problem into a simpler form, as depicted by Figure 4.4: We are given K sequences, $\boldsymbol{R} = (R_1, \ldots, R_K)$, of letters from an alphabet of size d ($d = 20$ for proteins and 4 for DNAs), and we search within them for a "conserved" pattern of length w, as illustrated by the blackened region in the figure. In this model, we assume that every sequence R_k has exactly one binding site and their locations are called the *alignment variable* and denoted as $A = (a_1, \ldots, a_K)$.

Table 4.2 displays a typical alignment of a common pattern, corresponding to the CRP-binding sites, found from the CRP data in Table 4.1. Each sequence segment corresponds to a blackened part in Figure 4.4. The locations of these sites have been experimentally determined (Stormo and Hartzell 1989). Hence, the "true" value of the alignment variable A is $A_0 = (64, 58, \ldots, 81)$. Clearly, these patterns are not exactly conserved. Biologists also showed that there are multiple motif sites in several of the sequence segments. For example, the 20th positions of the first and second sequences (cole 1 and ecoarabop) are all real binding sites. Hence, the assumption that each sequence has exactly one motif segment is not very realistic. A few other models have been developed to account for this limitation (Liu, Neuwald and Lawrence 1995, Neuwald, Liu and Lawrence 1995).

A common approach in literature (Stormo and Hartzell 1989, Lawrence and Reilly 1990, Lawrence et al. 1993, Krogh, Brown, Mian, Sjolander and Haussler 1994, Liu 1994a, Liu, Neuwald and Lawrence 1995, Durbin, Eddy, Krogh and Mitchison 1998) is to assume that the specificity of the motif can be represented as an unknown matrix with w columns (a first-order model). Column j in the matrix represents the base-type preference for the jth position in the motif. For example, we see from Table 4.2 that the first position of the motif prefers T, and the second prefers both T and G, and so on. Thus, a pattern matrix can be thought of as a $4 \times w$ matrix in our example, in which each column shows the preference of nucleotide base

Gene's name	Motif Start	Binding Site
cole1	64	TTTGATCGTTTTCACAAAA
ecoarabop	58	TTTGCACGGCGTCACACTT
ecobglr1	79	TGTGAGCATGGTCATATTT
ecocrp	66	TGCAAAGGACGTCACATTA
ecocya	53	TGTTAAATTGATCACGTTT
ecodecop	10	TTTGAACCAGATCGCATTA
ecogale	45	TTTATTCCATGTCACACTT
ecoilvbpr	42	CGTGATCAACCCCTCAATT
ecolac	12	TGTGAGTTAGCTCACTCAT
ecomale	17	TGTAACAGAGATCACACAA
ecomalk	64	CGTGATGTTGCTTGCAAAA
ecomalt	44	TGTGACACAGTGCAAATTC
ecoompa	51	CCTGACGGAGTTCACACTT
ecotnaa	74	TGTGATTCGATTCACATTT
ecouxu1	20	TGTGATGTGGTTAACCCAA
pbr-p4	56	TGTGAAATACCGCACAGAT
(tdc)	81	TGTGAGTGGTCGCACATAT

TABLE 4.2. The alignment of a common motif in upstream regions of the 17 *E. coli* genes. These sites have been determined experimentally. The original data consisting of 18 segments were produced by Stormo and Hartzell (1989) and displayed in Table 4.1. One of the sequences is eliminated from the table because its binding site contains an extra insertion compared with others.

types, in terms of total counts of A, T, G, and C, for the corresponding position in the motif.

As stated in Stormo and Hartzell (1989), the problem of identifying the binding sites from a collection of unaligned sequences can be viewed as finding the best A that gives us the maximal "mutual similarity." This similarity among multiple DNA segments can be represented by the "information content"

$$I_A = \sum_{j=1}^{w} \sum_{b=A}^{T} f_{j,b} \log \frac{f_{j,b}}{p_b}, \tag{4.1}$$

where $f_{j,b}$ is the observed frequency of base b at the jth position of the site, and p_b is the fraction of base b in an appropriate background (e.g., the whole genome or the whole dataset in consideration). With the goal of optimizing (4.1), Stormo and Hartzell (1989) proposed the first effective method to search for conserved patterns in multiple sequences. Their method as outlined can be seen as a greedy sequential method:

Stormo-Hartzell Algorithm

1. Each of the *w-words* (k-long substring) of the first sequence, R_1, are considered as a possible motif pattern — they are indeed equally likely to be a binding site without further information. In the CRP example, one forms 86 "matrices" to represent possible motif patterns.

2. The next sequence on the list is added to the analysis. All the matrices formed previously (say, N of them) are paired with all possible w-

words in the new sequence, and a "similarity score" is computed for each pair. There are a total of $N \times 86$ possible pairs for the CRP example.

3. The top N best scored pairs are kept, from which a new set of N "matrices" is formed. For each kept pair, the new matrix is formed by adding the site (w-word) found in the new sequence to the matrix it paired with.

4. Repeat the previous two steps until all sequences have been processed.

The foregoing algorithm can be understood from a sequential imputation viewpoint. Step 2 in the algorithm is, in fact, a predictive sampling step as in (1.4), $p(a_t \mid a_1, \ldots, a_{t-1}, R_1, \ldots, R_k)$. But instead of random sampling, they chose the most probable one according to this distribution. We now describe how the problem can be treated by a proper statistical model and how the computation can be completed by a sequential Monte Carlo method.

The matrix model can be more concisely stated as a *product multinomial* model (Liu, Neuwald and Lawrence 1995). Let $\boldsymbol{\theta}_0 = (\theta_{01}, \ldots, \theta_{0d})^T$ be the probability (column) vector describing the residue frequencies outside a motif ($d = 4$ in our example). For notational simplicity, we just use numbers $1, 2, \ldots, d$, instead of the actual names of the DNA base pairs or amino acid residues, to represent the letters in the alphabet. We let $\boldsymbol{\theta}_j$, $j = 1, \ldots, w$, represent the frequency of each base at the jth position of the motif. Then, $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_w]$, called a *product-multinomial* model (Liu, Neuwald and Lawrence 1995), is equivalent to the pattern matrix we described previously by words. It is also called a *profile* matrix in literature. By treating the alignment variable $A = (a_1, \ldots, a_K)$ as missing data, we can now write down a simple statistical model:

$$\begin{aligned} p(\boldsymbol{R} \mid \boldsymbol{\theta}_0, \boldsymbol{\Theta}, A) &= \boldsymbol{\theta}_0^{\mathbf{h}(\mathbf{R}_{\{A\}^c})} \prod_{j=1}^{w} \boldsymbol{\theta}_j^{\mathbf{h}(\mathbf{R}_{\mathbf{A}+\mathbf{j}-\mathbf{1}})} \\ &= \boldsymbol{\theta}_0^{\mathbf{h}(\mathbf{R})} \prod_{j=1}^{w} \left(\frac{\boldsymbol{\theta}_j}{\boldsymbol{\theta}_0}\right)^{\mathbf{h}(\mathbf{R}_{\mathbf{A}+\mathbf{j}-\mathbf{1}})} . \end{aligned} \tag{4.2}$$

In the expression, we use $A_{[-k]}$ to denote all of A but a_k, $A + l = \{a_1 + l, \ldots, a_K + l\}$ to denote the set of l-shifted positions of A, and $\{A\} = \{a_k + j - 1 : k = 1, \ldots, K,\ j = 1, \ldots, w\}$ to represent the set of residue indices occupied by the motif elements with alignment variable A. For any set C of indices, $\boldsymbol{R}_C$ represents the collection of the residues indexed by elements of C. For example, given any alignment variable A, we have $\boldsymbol{R}_{\{A\}} = \{r_{k,a_k+j-1} : \text{ for } j = 1, \ldots, w; \quad k = 1, \ldots, K\}$. The counting function $\boldsymbol{h}(\,)$, whose argument is a set of residues, counts how many of each letter types in a set of protein residues or DNA base pairs). For example, if

$R = \{AATCCCTG\}$ is an oligonucleotide sequence, we obtain that $\boldsymbol{h}(R) = (2, 2, 1, 3)$ for letter types A, T, G, and C. It is much simpler to state the complete-data model (4.2) in words: Given the alignment variable $\boldsymbol{A}$, the part of the dataset $\boldsymbol{R}$ outside the motif elements are like i.i.d. realizations from a multinomial model $\boldsymbol{\theta}_0$; and each position in a binding site follows a different multinomial model $\boldsymbol{\theta}_j$. Our task is to make inference on A, $\boldsymbol{\theta}_0$, and the motif matrix $\boldsymbol{\Theta}$. In order to take a Bayesian approach, we let the prior distribution for A be uniform on all allowable configurations; let the prior for $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_j$ be Dirichlet$(N_0\alpha_{0,1}, \ldots, N_0\alpha_{0,4})$, where the α are base frequencies in the genome and the *pseudo-counts* N_0 were chosen as $\sqrt{K}$ in many of our examples. More details on this model can be found elsewhere (Liu 1994a, Liu, Neuwald and Lawrence 1995). A Gibbs sampling scheme for this problem will be shown in Section 6.5. Here, we will describe a SIS method to impute A.

Let $A_t = (a_1, \ldots, a_t)$ and let $\boldsymbol{R}_t = \{R_1, \ldots, R_t\}$ be the collection of the first t sequences. Suppose we have imputed multiple copies of the alignment variable, $A_{t-1}^{(1)}, \ldots, A_{t-1}^{(m)}$, with respective weights $w_{t-1}^{(1)}, \ldots, w_{t-1}^{(m)}$. Then, with the new sequence R_t, we can update the weight as

$$w_t^{(j)} = w_{t-1}^{(j)} p(R_t \mid A_{t-1}^{(j)}, \boldsymbol{R}_{t-1}), \tag{4.3}$$

where the predictive probability can be computed:

$$p(R_t \mid A_{t-1}^{(j)}, \boldsymbol{R}_{t-1}) = \frac{1}{l_t - w + 1} \sum_{i=1}^{l_t - w + 1} p(R_t, a_t = i \mid A_{t-1}^{(j)}, \boldsymbol{R}_{t-1}).$$

Since $(A_{t-1}^{(j)}, \boldsymbol{R}_{t-1})$ determines an estimate of the pattern matrix $\boldsymbol{\Theta}$, the above predictive probability is just the likelihood of R_t under the jth estimated pattern matrix, given that it contains a binding site. Simultaneously with the weight updating, we can impute the motif locations for the new sequence; that is, $a_t^{(j)}$ is drawn from

$$P(a_t^{(j)} = i \mid A_{t-1}^{(j)}, \boldsymbol{R}_t) \propto p(R_t, a_t = i \mid A_{t-1}^{(j)}, \boldsymbol{R}_{t-1}). \tag{4.4}$$

When the importance weight $w_t^{(j)}$ is too skewed, we *resample* among the alignment matrices with probability proportional to the $w_t^{(j)}$ (Section 3.4.4). One can also use a different set of weights to do resampling (Liu, Chen and Logvinenko 2000). For better efficiency, the resampling step should take place right after the weight updating step (4.3) and *before* the sampling step (4.4). One can see that the resampling and predict sampling steps play a similar role as Steps 2 and 3 of the Stormo-Hartzell algorithm.

4.2 Approximating Permanents

Let $A = (a_{ij})_{n\times n}$ be a 0-1 matrix (also called the restriction matrix), where each entry a_{ij} is either 0 or 1. Let Π be the set of all permutaions of $\{1,\ldots,n\}$. A useful representation of a permutation σ is

$$\begin{pmatrix} 1 & 2 & \cdots & n \\ \sigma(1) & \sigma(2) & \cdots & \sigma(n) \end{pmatrix},$$

where $\sigma(i)$ record the new position of label i after the permutation. For any $\sigma \in \Pi$, we define

$$a(\sigma) = \prod_{i=1}^{n} a_{i\sigma(i)}.$$

Following the notations in Diaconis et al. (2001), we let S_A be the set of all "permitted" permutations under A:

$$S_A = \{\sigma : \ a(\sigma) = 1\}.$$

The *permanent* of A is defined as

$$\text{perm}(A) = \sum_{\sigma\in\Pi}\prod_{i=1}^{n} a_{i\sigma(i)} \equiv |S_A|, \tag{4.5}$$

There are many statistical applications that are related to permutations and the computation of the permanents (Diaconis et al. 2001). For example, for the test of independence for the astronomy data in Section 1.7, we want to approximate the tail probability of a test statistics under the uniform distribution of all "permitted" permutations. This task requires us to simulate from the uniform distribution on S_A. Sometimes one might also be interested in the total number of permitted permutations — the permanent perm(A). Approximating the permanent has been a great undertaking in recent years among computer scientists (Jerrum and Sinclair 1989).

A naive Monte Carlo method for approximating perm(A) is to generate random permutations $\sigma_1,\ldots,\sigma_N$ *uniformly* and to estimate

$$\text{perm}(A) \approx n!\frac{\sum_{j=1}^{N} a(\sigma_j)}{N}.$$

However, this method becomes very inefficient when the number of zeros in A is relatively large. Clearly, if one of the entries in the product in $a_{1\sigma(1)}$, $\ldots$, $a_{n\sigma(n)}$ is zero, then $a(\sigma) = 0$. Thus, we want to design a Monte Carlo algorithm that samples only on those permutations whose $a_{i\sigma(i)}$ terms are all none-zero. This goal can be achieved by the following recursive strategy proposed in (Chen and Liu 2001).

Let r_i, $i = 1,\ldots,n$, and c_j, $j = 1,\ldots,n$, be the row sums and the column sums of A, respectively. For the first column, we draw an entry [i.e., $\sigma(1)$]

among all those positions with $a_{i1} = 1$. However, we do not want to sample $\sigma(1)$ uniformly among all the c_1 positions. Note that for any permutation $\sigma \in \Pi$, if $\sigma(1) = s$, then none of the $\sigma(i)$, $i = 2, \ldots, n$ can be equal to s. Thus, we should prefer to use those nonzero entries that correspond to relatively small row sums (i.e., small r_s). More precisely, we sample $\sigma(1)$ from the distribution

$$P[\sigma(1) = s] \propto \frac{1}{r_s - 1}, \tag{4.6}$$

for s in the set $S_1 = \{s : \ a_{s1} = 1\}$. If one row sum, say, r_s equals one for some $s \in S_1$, then $q[\sigma(1) = s] = 1$. If we can find $s \neq s'$, both in S_1, such that $r_s = r_{s'} = 1$, then there is no allowable permutation; thus, $\text{perm}(A) = 0$.

After sampling the first column, or $\sigma(1)$ equivalently, we can update our "working matrix" by setting all the entries in the first column and the $\sigma(1)$th row to 0's and updating the corresponding row sums and column sums. Then, we proceed to the next column, applying the same sampling strategy. If this procedure can be carried out recursively to the last column, we obtain an "allowable" permutation σ whose sampling distribution is

$$P[\sigma = (s_1, s_2, \ldots s_n)] = \frac{(r_{(0)s_1} - 1)^{-1}}{D_{(0)}} \frac{(r_{(1)s_2} - 1)^{-1}}{D_{(1)}} \cdots \frac{(r_{(n-1)s_n} - 1)^{-1}}{D_{(n-1)}},$$

where $r_{(k)s_{k+1}}$ is the kth modification of the s_{k+1}th row sum (after resetting the first k columns)and the denominator is

$$D_{(k)} = \sum_{i:\ a^{(k)}_{i(k+1)}=1} (r_{(k)i} - 1)^{-1},$$

where $a^{(k)}_{ij}$ is the (i,j)th entry of the kth modified restriction matrix (i.e., after resetting the first k columns and the rows $s_1, \ldots, s_k$ to 0).

The weight of each generated permutation σ is updated recursively as

$$w_{k+1}(s_1, \ldots, s_{k+1}) = \begin{cases} w_k(r_{(k)s_{k+1}} - 1)D_{(k)} & \text{if } D_{(k)} < \infty \\ w_k & \text{otherwise,} \end{cases} \tag{4.7}$$

If at some stage it is impossible to proceed to produce a valid permutation, we assign a weight 0 to this unsuccessful trial. After performing m such trials of sequential sampling, we obtained m weighted permutations, $(\sigma^{(j)}, w^{(j)})$, for $j = 1, \ldots, m$, including unsuccessful ones (corresponding to a weight of 0), we can estimate the permanent of A as

$$\widehat{\text{perm}}(A) = \frac{w^{(1)} + \cdots + w^{(m)}}{m}.$$

Other statistical tasks such as the hypothesis testing can also be accommodated.

This simple method was shown very efficient in a number of simulation examples we have tested on. We started with an 8×8 matrix of which we know the true answer. The cv^2 of our procedure, which approximates $\text{perm}(A)$ accurately, was about 0.3. We then simulated a random matrix of 50×50. The cv^2 of our SIS procedure for this case only increased moderately to 0.5. It took about half a second to generate one permutation on a Sun Ultra 60 workstation. When we increased the matrix size to 100×100, the cv^2 was still impressively small, only about 0.7.

It should be noted that the methods of Kuznetsov (1996) and Beichl and Sullivan (1999) can all be viewed as SIS samplers. The computational and conceptual complexity of our approach is similar to that of Kuznetsov (1996). But our method tends to be more efficient. Compared with Beichl and Sullivan, our approach is much simpler, yet more efficient.

4.3 Counting 0-1 Tables with Fixed Margins

The problem of counting 0-1 tables with fixed margins was introduced in Section 3.4.2. It is a long-standing problem in the fields of applied mathematics and computer science and has been subject to active research for many years. In a recent work, Chakraborty, Chen, Diaconis, Holmes and Liu (2001) developed a number of techniques for solving the problem. These include a MCMC method, an exact counting method, and an SIS method. Here, we focus on their SIS approach.

Briefly, the SIS method begins by filling in columns (or rows) of the $m \times n$ table from left to right sequentially. After the first $t-1$ columns are filled, the row sums are updated and the tth column is filled in by sampling c_t of its m possible positions to put in 1's. These c_t positions are sampled according to a distribution that is related to the vector of the updated row sums. For example, the probability that positions $i_1, \ldots, i_{c_t}$ of the tth column are sampled can be proportional to

$$\left(\frac{r_{i_1}}{n - r_{i_1}} \times \cdots \times \frac{r_{i_{c_t}}}{n - r_{i_{c_t}}} \right)^{1+\delta}. \tag{4.8}$$

More choices of the sampling distribution and other mathematical details can be found in Chen (2001). This sequential sampling method gives us a rather accurate estimate for fairly large tables.

To test the method, they examined Darwin's finch data (Sanderson 2000), which, in the form of an "occurrence matrix," recorded the presence and absence for each of the 13 species of Galápagos finch in 17 islands of an east archipelago. This data is shown as in Table 4.3.

The problem of couting the number of 0-1 tables that have the same marginal sums as the finch data was first raised by Susan Holmes in Stanford. Based on 1000 sequentially simulated tables, the SIS method esti-

0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	14
1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	0	13
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	14
0	0	1	1	1	0	0	1	0	1	0	1	1	0	1	1	1	10
1	1	1	0	1	1	1	1	1	1	0	1	0	1	1	0	0	12
0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	2
0	0	1	1	1	1	1	1	1	0	0	1	0	1	1	0	0	10
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	0	0	10
0	0	1	1	1	1	1	1	1	1	0	1	0	1	1	0	0	11
0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0	6
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	3	10	9	9	7	8	9	7	8	2	9	3	6	8	2	2	

TABLE 4.3. The finch dataset records the occurrences of 13 species(rows) of Galápagos finch in 17 islands (columns) of an east Pacific archipelago. Ecologists are interested in testing if the occurrence pattern is a random draw from the uniform distribution of all such patterns.

mated that the total count of the 0-1 tables with the given margins is $(6.72 \pm .02) \times 10^{16}$. The computation took 12 minutes on a Pentium 400 machine. With more computing time, 10^8 tables were generated and the total number of the tables was estimated as 6.715×10^{16}. The coefficient of variation of the importance weights (defined as the sample variance divided by the square of the sample mean) was around 0.7. The "truth" they obtained by using an exact-counting algorithm is $6.71...\times 10^{16}$.

To push further for the algorithm, we simulated a 50×50 random table and recorded its marginal sums. The row sums are 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 6, 6, 7, 8, 8, 9, 10, 11, 12, 12, 12, 12, 12, 13, 14, 14, 15, 16, 18, 19, respectively, and the column sums are 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 8, 8, 8, 8, 9, 9, 9, 10, 10, 11, 12, 13, 13, 13, 14, 14, 14, respectively. The column sums were ordered from the largest to the smallest and the columns were sampled sequentially according to (4.8) with $\delta = 0$. This gave us a cv^2 of around 0.2. Based on 1,000 samples, which took about 5 minutes to generate, we estimated that the total number of 0-1 tables with these marginal sums is $(8.9 \pm 0.1) \times 10^{242}$. Based on 10,000 samples, the estimate was improved to 8.78×10^{242} with standard error 0.05×10^{242}. This example shows that the SIS method is still very efficient even for large tables.

4.4 Bayesian Missing Data Problems

4.4.1 Murray's data

Let us revisit the Gaussian missing data problem described in Section 2.5.7. The previous approach was based on the plain importance sampling in which the trial density was chosen as the posterior distribution based on the first four complete observations. Here, we consider an SIS approach.

Recall that the covariance matrix is assigned the Jeffreys non-informative prior (2.13) The posterior distribution of Σ given complete data is

$$p(\mathcal{Z}|\text{complete data}) \propto \mathcal{Z}^{\frac{3}{2}-1} \exp\left\{-\frac{1}{2}\,\text{tr}[\mathcal{Z}\cdot S]\right\},$$

where $S = (s_{ij})_{2\times 2}$ is the uncorrected sum of squares matrix and $\mathcal{Z} = \Sigma^{-1}$.

Let the complete data be $y_1, \ldots, y_{12}$, where $y_t = (y_{t,1}, y_{t,2})$ for $t = 1, \ldots, 12$. Thus, the $y_{t,2}$ are missing for observations 5 to 8 and the $y_{t,1}$ are missing for observations 9 to 12. The predictive distribution of y_{t+1} given $\mathbf{y}_t = (y_1, \ldots, y_t)$ is

$$y_{t+1}|\mathbf{y}_t \sim t_2\left(0,\ \frac{S_t}{t-1},\ t-1\right),$$

where $S_t = (s_{ij})$, $s_{ij} = \sum_{s=1}^{t} y_{s,i}y_{s,j}$, and t_2 is the bivariate t-distribution. It follows that, conditional on $\mathbf{y}_t = (y_1, \ldots, y_t)$, the marginal and conditional distributions are

$$\begin{aligned} y_{t+1,1} \mid \mathbf{y}_t &\sim t_1\left(0, \frac{s_{11}}{t-1}, t-1\right), \\ y_{t+1,2} \mid \mathbf{y}_t, y_{t+1,1} &\sim t_1\left[\frac{s_{12}}{s_{11}}y_{t+1,1},\ \frac{|S_t|}{ts_{11}}\left(1+\frac{y_{t+1,1}^2}{s_{11}}\right),\ t\right], \end{aligned}$$

respectively. Similar results can be obtained for conditional distributions $[y_{t+1,2} \mid \mathbf{y}_t]$ and $[y_{t+1,1} \mid \mathbf{y}_t, y_{t+1,2}]$. Based on these distributional results, Steps A and B of the sequential imputation (Section 3.2.2) can both be easily implemented. Figure 4.5 gives the exact posterior distribution of ρ and the approximated posterior distribution based on sequential imputation. The approximation is a weighted mixture of $m = 1000$ complete-data posterior distributions.

The complete-data posterior distribution of ρ is still difficult to compute since it has the form

$$p(\rho \mid \mathbf{y}) \propto (1-\rho^2)^{\frac{\nu-2}{2}} \int_0^\infty \omega^{-1}\left(\omega + \frac{1}{\omega} - 2\rho r\right)^{-(\nu+1)} d\omega, \qquad (4.9)$$

where $\mathbf{y}$ is the completed data after the missing part being imputed, r is the sample correlation coefficient, and $\nu = n - 2 = 10$. To avoid numerical

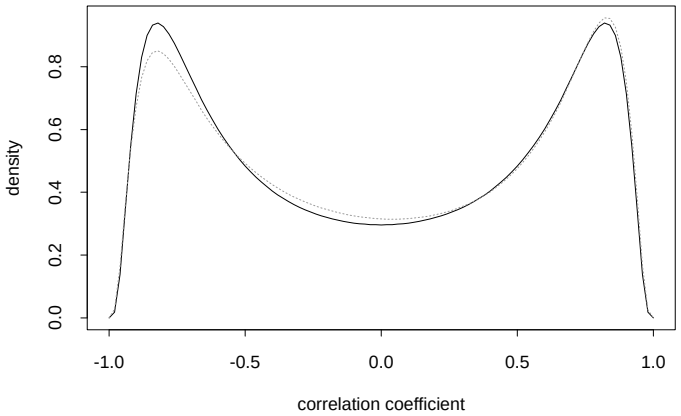

FIGURE 4.5. An approximation to the posterior distribution of the sample correlation coefficient ρ for Murray's data. Solid line: true density; dashed line: approximation.

integration in (4.9), we can use the algorithm of Odell and Feiveson (1966) to generate observations from the inverse Wishart distribution, which gives rise to a sample from $p(\rho \mid \mathbf{y})$. Note that we do not have to draw from an inverse Wishart distribution *during* sequential imputation.

4.4.2 Nonparametric Bayes analysis of binomial data

Kong et al. (1994) and Liu (1996b) considered the following *hierarchical* binomial model:

$$\begin{aligned} y_t &\sim \text{Binomial}(l_t, \zeta_t) \\ \zeta_t &\overset{\text{i.i.d.}}{\sim} F, \quad 1 \le t \le n. \end{aligned}$$

Our interests are in drawing inference about both F and the ζ_t's based on the observed data y_t. As a concrete example, consider a dataset (Figure 4.6) of n thumbtacks randomly drawn from a certain population. Here, ζ_t can be interpreted as the inherent probability for the tth tack to point up when being flicked. The observed data for the tth tack is y_t, the number of times the tack landed point up out of a total of l_t flicks. The data in Figure 4.6 was generated by Beckett and Diaconis (1994), of whom each flicked 16 different thumbtacks on 10 different surfaces. For each person-tack-surface combination (a total of $2 \times 16 \times 10$ scenarios), the experiment was repeated 9 times. For simplicity, we treat the data as though they came from 320 different tacks and each being flicked 9 times independently. A histogram of the data (the y_i) is shown in Figure 4.6.

Different from the method outlined in Section 1.8, we take a *nonparametric* Bayes approach by assigning a *Dirichlet process* prior, $\mathcal{D}(\alpha)$, to the

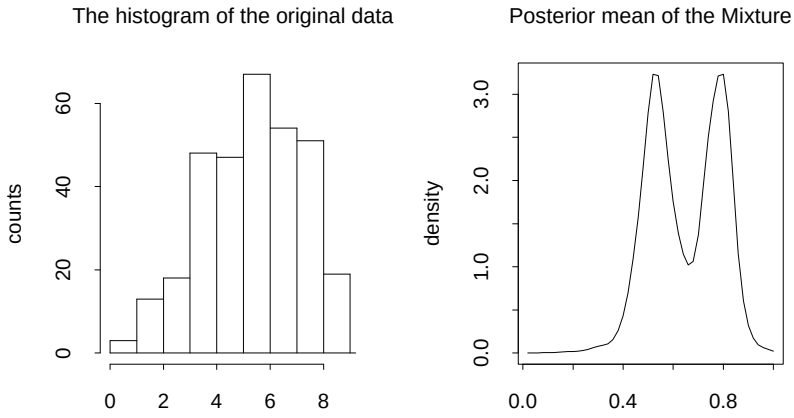

FIGURE 4.6. Left: The histogram of the tack data produced by Beckett and Diaconis (1994). Right: The predictive distribution of ζ for a future tack (which is also the posterior mean of F).

infinite-dimensional parameter F. The hyperparameter α is a finite positive measure on interval $[0,1]$. We define the norm $\|\alpha\|$ as the total measure of the support of F. Note that $\mathcal{D}(\alpha)$ is a probability measure on $\mathcal{P}$, the set of all probabilities on the Borel sets of $[0,1]$. Readers not familiar with this special class of Dirichlet process distribution and nonparametric Bayes inference are referred to Ferguson (1974). In the tacks example, we may be interested in estimating ζ_t for tack t, given its performance in 9 independent flips, or in obtaining the posterior mean of the unknown distribution F [i.e., $E(F \mid \zeta_1, \ldots, \zeta_n)$].

For simplicity, suppose α is the uniform measure on $[0,1]$ (hence, the norm $\|\alpha\| = 1$). Then, a key result (Ferguson 1974) is that conditional on $\zeta_1, \ldots, \zeta_{t-1}$, the posterior distribution of F is again a Dirichlet process, $\mathcal{D}(\alpha_{t-1})$ with $\alpha_{t-1} = \alpha + \sum_{i=1}^{t-1} \delta_{\zeta_i}$, where δ_{ζ_i} is a delta measure with mass 1 located at ζ_s. Thus, the predictive distribution for the next thumbtack, ζ_t, is

$$\zeta_t | \zeta_1, \ldots, \zeta_{t-1} \sim \frac{1}{t} \left(\alpha + \sum_{i=1}^{t-1} \delta_{\zeta_i} \right), \tag{4.10}$$

which should be interpreted as a probabilistic mixture of α and delta measures concentrated at the ζ_i's (Antoniak 1974). Note that the posterior mean of F, $E(F \mid \zeta_1, \ldots, \zeta_t)$, is the same as (4.10). It is easy to see from (4.10) that

$$\begin{aligned} [\zeta_t | \zeta_1, \ldots, \zeta_{t-1}, y_t] \;=\; & \frac{1}{Z} \Bigg[B(y_t + 1, l_t - y_t + 1) \mathrm{Beta}(y_t + 1, l_t - y_t + 1) \\ & + \sum_{i=1}^{t-1} \zeta_i^{y_t} (1 - \zeta_i)^{l_t - y_t} \delta_{\zeta_i} \Bigg] \end{aligned} \tag{4.11}$$

where

$$B(y_t+1, l_t-y_t+1) = \int_0^1 \zeta^{y_t}(1-\zeta)^{l_t-y_t}d\zeta = \frac{\Gamma(y_t+1)\Gamma(l_t-y_t+1)}{\Gamma(l_t+2)}$$

is the Beta function, $\text{Beta}(\cdot,\cdot)$ is the standard Beta distribution, and

$$Z = B(y_t+1, l_t-y_t+1) + \sum_{i=1}^{t-1} \zeta_i^{y_t}(1-\zeta_i)^{l_t-y_t}$$

is the normalizing constant. Note that (4.11) is a mixture of a Beta distribution and discrete point masses. From (4.10), we also get

$$p(y_t|\zeta_1,\ldots,\zeta_{t-1}) = \frac{1}{t}B(y_t+1, l_t-y_t+1) + \frac{1}{t}\sum_{i=1}^{t-1} \zeta_i^{y_t}(1-\zeta_i)^{l_t-y_t},$$

which is the term needed for updating the importance sampling weights. Hence, both Steps A and B of sequential imputation can be easily implemented (Liu 1996b).

Note that a direct application of the data augmentation algorithm (Section 6.4) is difficult because sampling F from

$$[F \mid \zeta_1,\ldots,\zeta_n] = \mathcal{D}(\alpha + \delta_{\zeta_1} + \cdots + \delta_{\zeta_n})$$

is infeasible. [Some approximations exist; see Doss (1994)]. Escobar (1994) described a "collapsed" Gibbs sampling algorithm (Section 6.7) in which the sampling of F is not needed. His method also takes advantage of the simplicity of the predictive distributions (4.10) and (4.11). For a related problem where the $\zeta_t's$ are ordered, Gelfand and Kuo (1991) used a similar idea to avoid sampling the infinite-dimensional parameter F.

The sequential imputation procedure derived from using (4.10) and (4.11) was applied to the thumbtack data with α being uniform and $\|\alpha\| = 1$. Figure 4.6 displays the posterior mean of the unknown density function F, $E(F \mid \mathbf{y})$. Liu (1996b) studied the sensitivity of this result to the prior assumption and presented a method to estimate the norm of α.

A more sophisticated "second-generation" SIS procedure for nonparametric Bayes problems was proposed by MacEachern et al. (1999). They noticed that when sampling ζ_t from (4.11), either a new ζ is produced from the Beta distribution or a previous ζ is drawn. As a consequence, the ζ_t forms natural clusters because of t)the Dirichlet process assumption on F. The plain SIS procedure tends to fix the location of these clusters at a relatively early stage, which is inefficient because an early cluster location may be invalidated by the subsequent new observations. The new SIS method introduces the *cluster* indicator variable I_t, which tells us whether ζ_t should be in one of the clusters formed by $\zeta_1,\ldots,\zeta_{t-1}$ or form a new cluster. Conditional on the indicators $I_1,\ldots,I_n$, one can integrate out all

the ζ_t. The new sequential importance sampler is then constructed on the space of $(I_1, \ldots, I_n)$. When applied to the tack data under the same prior setting, the new method yields a substantial improvement over the plain SIS described earlier in this section (a cv^2 of 11 for the new method versus 43 for the old one).

4.5 Problems in Signal Processing

4.5.1 Target tracking in clutter and mixture Kalman filter

Tracking a target in clutter is of interest to engineers and computer scientists. The problem has received much attention recently since the proposal of the bootstrap filter (Avitzour 1995, Gordon et al. 1995). Here, we use the simple one-dimensional tracking problem in Avitzour (1995) to show how the SIS methods can be applied in this area.

Avitzour (1995) modeled the tracking problem as a state-space model with the state variable $x_t=(x_{t,1}, x_{t,2})$, where $x_{t,1}$ is the location of the target on a straight line and $x_{t,2}$ is the target velocity. The x_t evolve in the following way:

$$\begin{aligned} x_{t,1} &= x_{t-1,1} + x_{t-1,2} + \frac{1}{2} w_t, \\ x_{t,2} &= x_{t-1,2} + w_t, \end{aligned}$$

where the noise term w_t are i.i.d. and follow distribution $N(0, q^2)$. If we could identify the object (without confusing it with others) at all times, then our observation would have been the object's location $x_{t,1}$ plus a small random noise. In other words, we observe

$$z_t = x_{t,1} + v_t,$$

where the v_t are i.i.d. Gaussian noises with distribution $N(0, r^2)$. If all the z_t are directly observable, then the tracking problem can be solved satisfactorily by using a Kalman filter (Bar-Shalom and Fortmann 1988).

When confusing objects are present in the detection window, however, our observation at time t becomes y_t, which is a vector of length m_t, where m_t is the total number of observed objects in the window and each component of y_t represents an object's position. Among these m_t measured locations, at most one corresponds to the true target we are interested in tracking (i.e., is equal to z_t). The occurrence of the confusing objects is assumed to follow a Poisson process with rate α. We further assume that there is only a probability $p_d < 1$ for the observation window to actually include the target's location (z_t) in y_t. Therefore, if the range of the detection window is Δ, the distribution of m_t is Bernoulli(p_d)+Poisson$(\lambda\Delta)$ and the false signals are uniformly distributed in the detection region. By

introducing an indicator variable I_t,

$$I_t = \begin{cases} 0 & \text{if the target object is not in the detection range} \\ k & \text{if the } k\text{th object corresponds to the target,} \end{cases}$$

we can formulate this problem as a state-space model; that is, when $I_t = 0$,

$$p(y_t \mid x_t, I_t = 0) = \Delta^{-m_t} \frac{(\lambda\Delta)^{m_t}}{m_t!} e^{-\lambda\Delta} = \frac{\lambda^{m_t}}{m_t!} e^{-\lambda\Delta};$$

and when $I_t = k$,

$$p(y_t \mid x_t, I_t = k) = \frac{\lambda^{m_t - 1}}{(m_t - 1)!} e^{-\lambda\Delta} \frac{1}{\sqrt{2\pi}r} \exp\left\{ -\frac{(y_{t,k} - x_t)^2}{2r^2} \right\}.$$

Since *a priori* $P(I_t = 0) = 1 - p_d$ and $P(I_t = k) = p_d/m_t$, we have

$$f_t(y_t, I_t \mid x_t) \propto \begin{cases} (1 - p_d)\lambda & \text{if } I_t = 0; \\ p_d(2\pi r^2)^{-1/2} \exp\left[-\dfrac{(y_{t,k} - x_t)^2}{2r^2} \right] & \text{otherwise.} \end{cases}$$

Since I_t is not observable, we need to sum out the I_t to obtain the *observation distribution* $f_t(y_t \mid x_t)$.

Using $q(x_t \mid x_{t-1})$ to denote the state evolution relationship, we can obtain a sequence of auxiliary distributions, $\pi_t(x_1, \ldots, x_t)$, as in (3.8):

$$\pi_t(\mathbf{x}_t) \propto f_t(y_t \mid x_t) q(x_t \mid x_{t-1}) \pi_{t-1}(\mathbf{x}_{t-1})$$

and

$$\pi_t(x_t) \propto \int f_t(y_t \mid x_t) q(x_t \mid x_{t-1}) \pi_{t-1}(x_{t-1}) dx_{t-1}.$$

The second equation holds because of the Markovian structure among the x_t. With this formulation, the current position of the target can be estimated as $E_{\pi_t}(x_t)$.

Conditional on the current value of x_{t-1}, we can easily simulate x_t from the state equation $q(\,)$. Given a sampled value of x_t, the computation of its weight, $f_t(y_t \mid x_t)$, is also easy. Thus, the *bootstrap filter* (Section 3.3) can be easily applied to this problem. However, if we pay a little more attention to the model's special structure, we can come up with a much more efficient algorithm.

Let $\mathbf{\Lambda}_t = (I_1, \ldots, I_t)$ be called a *trajectory* up to time t. We note that an important feature of our tracking model is that if we know the values of the trajectory $\mathbf{\Lambda}_t$ of the target, the tracking system becomes linear and Gaussian and the computation of the Bayes estimator, $E(x_t \mid y_1, \ldots, y_t)$, can be achieved *exactly* by a standard Kalman filter. Therefore, conditional on $\mathbf{\Lambda}_t$, we can integrate out $\mathbf{x}_t$ exactly. This feature enables us to design a SIS system only on the reduced space of $\mathbf{\Lambda}_t$ (Liu and Chen 1998, Chen

and Liu 2000a). This approach is a vivid demonstration of the power of *marginalization* technique (Section 3.4.6). Since this new algorithm takes the form of mixing over a number of Kalman filters, we call the method *mixture Kalman filter* (MKF). Chen and Liu (2000a) give more details. Briefly, a general MKF algorithm can be stated as follows.

Suppose at time $t-1$, we have sampled m trajectories $\{\mathbf{\Lambda}_{t-1}^{(1)}, \ldots, \mathbf{\Lambda}_{t-1}^{(m)}\}$ with weights $w_{t-1}^{(1)}, \ldots, w_{t-1}^{(m)}$. For each trajectory $\mathbf{\Lambda}_{t-1}^{(j)}$, we can compute via a Kalman filter the mean vector $\boldsymbol{\mu}_{t-1}^{(j)}$ and the covariance matrix $\Sigma_{t-1}^{(j)}$ for the target. These are the sufficient statistics because the system is linear and Gaussian given $\mathbf{\Lambda}_{t-1}$. We denote $KF_{t-1}^{(j)} = (\boldsymbol{\mu}_{t-1}^{(j)}, \Sigma_{t-1}^{(j)})$. At time t, we run the following MKF updates: For $j = 1, \ldots, m$,

- generate $I_t^{(j)}$ from a trial distribution $g(I_t \mid \mathbf{\Lambda}_{t-1}^{(j)}, KF_{t-1}^{(j)}, y_t)$;
- conditional on each $\{KF_{t-1}^{(j)}, y_t, I_t^{(j)}\}$, obtain $KF_{t+1}^{(j)}$ by a one-step Kalman filter (Chen and Liu 2000a);
- update the new weight as $w_t^{(j)} = w_{t-1}^{(j)} \times u_t^{(j)}$, where

$$u_t^{(j)} = \frac{p(\mathbf{\Lambda}_{t-1}^{(j)}, I_t^{(j)} \mid \mathbf{y}_t)}{p(\mathbf{\Lambda}_{t-1}^{(j)} \mid \mathbf{y}_{t-1}) g(I_t^{(j)} \mid \boldsymbol{\lambda}_{t-1}^{(j)}, KF_{t-1}^{(j)}, y_t)};$$

- if the coefficient of variation of the w_t exceeds a threshold value, we resample a new set of KF_t from $\{KF_t^{(1)}, \ldots, KF_t^{(m)}\}$ with probability proportional to the weights $w_t^{(j)}$.

Smith and Winter (1978) proposed a deterministic filtering method, called *split-track filter* (STF), which has a similar flavor to the MKF we just outlined. In STF, one always keeps m trajectories of the latent indicators. At a future time step, it evaluates the likelihoods of all possible propagations from the m trajectories kept at the previous step, then finds and keeps the m new trajectories with the highest likelihood values. In contrast, our MKF selects these trajectories randomly, according to the weights (which is the predictive likelihood value), and uses the associated weights to measure how good each trajectory is. The important step of resampling is naturally built into MKF, which can overcome some weaknesses of STF. More sophisticated sampling and estimation methods can also be incorporated.

Figure 4.7 shows the plots of tracking errors (estimated location $-$ true location) of 50 simulated runs, with $r^2 = 1.0$, $q^2 = 0.1$, $p_d = 0.9$, and $\lambda = 0.1$. These parameter combination is slightly different from that of Avitzour (1995), with smaller clutter density but larger state equation variance. With their configuration, the results are similar, but the differences between different procedures are smaller. Five hundred streams (m=500)

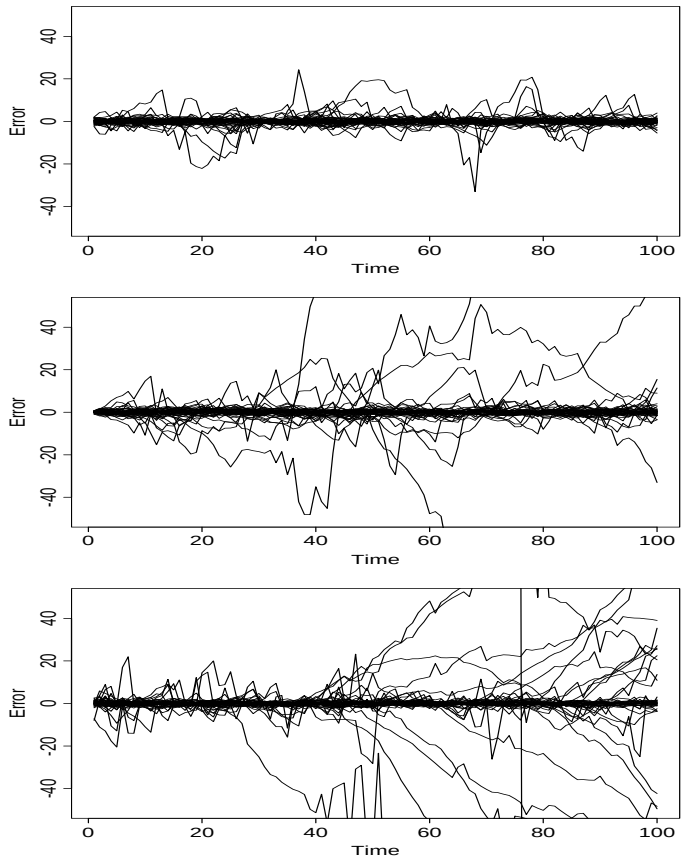

FIGURE 4.7. The tracking errors of 50 runs of the MKF (top), a sequential importance sampler (middle), and the split-track filter (bottom) for a simulated target moving system.

were used, with resampling done at every step. The top part of Figure 4.7 resulted from using the MKF method, and the middle part shows the result from using a slightly improved bootstrap filter (Avitzour 1995). We also implemented the split-track filter for this problem, which, at each step, saves the 500 trajectories with the highest likelihood values (bottom part). The generality of the MKF method is more thoroughly explored in Chen and Liu (2000a).

4.5.2 Digital signal extraction in fading channels

Many mobile communication channels can be modeled as Rayleigh flat-fading channels, which have the following form:

$$\begin{array}{rl} \text{State equations:} & \left\{ \begin{array}{rcl} \mathbf{x}_t & = & F\mathbf{x}_{t-1} + Ww_t \\ \alpha_t & = & G\mathbf{x}_t \\ s_t & \sim & p(\cdot \mid s_{t-1}), \end{array} \right. \\ \text{Observation equation:} & y_t = \alpha_t s_t + Vv_t, \end{array}$$

where s_t are the input digital signals (symbols), y_t are the received complex signals, and α_t are the unobserved (changing) fading coefficients. Both w_t and v_t are complex Gaussian with identity covariance matrices. It is important to note that this model has some similarity to the tracking problem; that is, given the input signals s_t, the system is linear in $\mathbf{x}_t$ and y_t. Therefore, we can design a MKF which focuses solely on the s_t (with $\mathbf{x}_t$ integrated out). The algorithmic detail is similar to the MKF updating steps described in the previous subsection and will be omitted here. Readers interested in more details and related applications are referred to Chen and Liu (2000a) and Chen, Wang and Liu (2000).

Consider a special example in which input signals are binary (i.e., $s_t = -1$ or $+1$). The fading coefficient takes complex values, with independent real and imaginary parts following the same state equation. Simulation were done with the following configurations:

$$F = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.94 & -2.88 & 2.94 \end{pmatrix}; \quad G' = 10^{-4} \begin{pmatrix} 0.04 \\ 0.11 \\ 0.11 \\ 0.04 \end{pmatrix}; \quad W = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix};$$

and $V = r$. That is, both of the real and the imaginary parts of α_t follow an ARMA(3,3) process

$$\begin{array}{rl} & \alpha_t - 0.94\alpha_{t-1} + 2.88\alpha_{t-2} - 2.94\alpha_{t-3} \\ = & 0.04e_t + 0.11e_{t-1} + 0.11e_{t-2} + 0.04e_{t-3} \end{array}$$

where $e_t \sim N(0, 0.01^2)$. In the communication literature, this is called a (low-pass) Butterworth filter of order 3 with cutoff frequency 0.01. It is normalized to have a stationary variance 1.

We are interested in estimating the differential code $d_t = s_t s_{t-1}$. Figure 4.8 shows the bit error rate of different signal-to-noise ratios (SNR), using a form of the MKF, the differential detection $\hat{d}_t = \text{sgn}[\text{real}(y_t y_{t-1}^*)]$ and a lower bound. The lower bound is obtained using the true fading coefficients α_t and $\hat{d}_t = \text{sgn}[\text{real}(\alpha_t^* y_t y_{t-1}^* \alpha_{t-1})]$. The Monte Carlo sample size m was 100 for the MKF. We also include the result of a delayed estimation, in which s_t is estimated using the samples $s_t^{(j)}$ generated by MKF and the

weight $w_{t+1}^{(j)}$ at time $t+1$ (Liu and Chen 1998). This delayed estimation is able to utilize the substantial information contained in the future information y_{t+1}, hence more accurate, due to the strong memory in the fading channel.

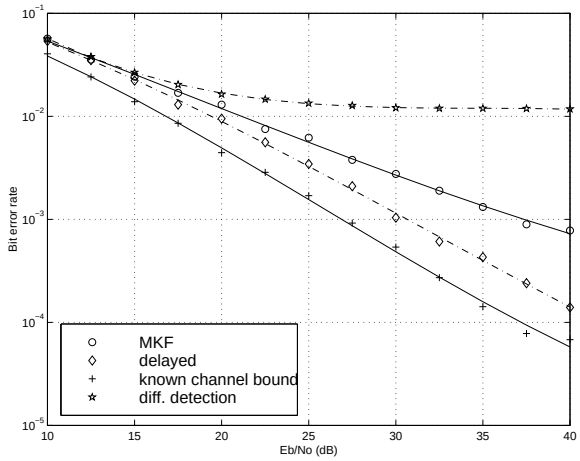

FIGURE 4.8. The bit error rate of extracting differential binary signals from a fading channel using MKF and differential detection. A lower bound that assumes the exact knowledge of the fading coefficients is also shown.

We can see that the simple differential detection works very well in low-SNR cases and no significant improvement can be expected. However, it has an apparent bit error rate floor for high-SNR cases. The MKF managed to break that floor, by using the structure of the fading coefficients.

4.6 Problems

1. Prove that the method described in Section 4.2 is valid. In particular, show that the weight updating strategy is correct. That is, the average of the weights indeed provide us an unbiased estimate on perm(A).

2. Describe how to use the SIS method of Section 4.2 to conduct the hypothesis test of independence for the astronomy data in Section 1.7.

3. Test to see whether you can find a better sampling distribution than (4.6) for approximating permanents. One possible choice is $P(\sigma(1) = s) \propto (r_s - 1)^{-\alpha}$ for some $\alpha > 0$. But it might be possible to use a more complex function of all the r_j.

4. Study the "scanning future" method (Meirovitch 1982, Meirovitch 1985). Investigate its potential for the nonlinear filtering problem and the molecular structural optimization problem.

5. Investigate further the difference between the method of "split-track filter" and the mixture Kalman filter.