Monte Carlo Methods - a special topics course

Tom Kennedy

April 27, 2016

Contents

Chapter 1

Introduction

A Monte Carlo method is a compuational method that uses random numbers to compute (estimate) some quantity of interest. Very often the quantity we want to compute is the mean of some random variable. Or we might want to compute some function of several means of random variables, e.g., the ratio of two means. Or we might want to compute the distribution of our random variable. Although Monte Carlo is a probabilistic method, it is often applied to problems that do not have any randomness in them. A classic example of this is using Monte Carlo to compute high dimensional integrals.

Monte Carlo methods may be divided into two types. In the first type we generate independent samples of the random variable. This is usually called direct, simple or crude Monte Carlo. The latter two terms are rather misleading. We will refer to these types of methods as direct Monte Carlo. In this type of Monte carlo the samples X_i that we generate are an i.i.d. sequence. So the strong law of large numbers tells us that the average of the X_i, i.e., the sample mean $\frac{1}{n}\sum_{i=1}^{n} X_i$, will converge to the mean of X as $n \to \infty$. Furthermore the central limit theorem tells us a lot about the error in our computation.

The second type of methods are Markov Chain Monte Carlo (MCMC) methods. These methods construct a Markov Chain whose stationary distribution is the probability measure we want to simulate. We then generate samples of the distribution by running the Markov Chain. As the chain runs we compute the value X_n of our random variable at teach time step n. The samples X_n are not independent, but there are theorems that tell us the sample mean still converges to the mean of our random variable.

We give some examples.

Example - Integration: Suppose we want to compute a definite integral over an interval like

$$I = \int_a^b f(x)\,dx \tag{1.1}$$

Let X be a random variable that is uniformly distributed on $[a, b]$. Then the expected value of $f(X)$ is

$$E[X] = \frac{1}{|b-a|}\int_a^b f(x)\,dx \tag{1.2}$$

So $I = (b-a)E[X]$. If we generate n independent sample of X, call them $X_1, X_2, \cdots, X_n$, then the law of large numbers says that

$$\frac{1}{n}\sum_{i=1}^{n} f(X_n) \tag{1.3}$$

conveges to $E[X]$. The error in this method goes to zero with n as $1/\sqrt{n}$. If f is smooth, simple numerical integration methods like Simpson's rule do much better - $1/n^4$ for Simpson. However, the rate of convergence of such methods is worse in higher dimensions. In higher dimension, Monte Carlo with its slow rate of convergence may actually be faster. And if the integrand is not smooth Monte Carlo may be the best method, especially if simplicity is important.

Example - more integration: Suppose we want to evaluate the integral

$$\int_0^{10} e^{-x^2} \tag{1.4}$$

We could follow the example above. However, e^{-x^2} is essentially zero on a large part of the interval $[0, 10]$. So for most of our samples X_i, $f(X_i)$ is essentially zero. This looks inefficent. A large part of our computation time is spent just adding zero to our total. We can improve the efficiency by a technique that is called importance sampling. We rewrite the integral we want to compute as

$$\int_0^{10} \frac{e^{-x^2}}{ce^{-x}} ce^{-x}\,dx \tag{1.5}$$

where the constant c is chosen so that $\int_0^{10} ce^{-x}\,dx = 1$. Let $g(x) = e^{-x^2}/ce^{-x}$. If X is a random variable with density ce^{-x}, then the expected value $E[g(X)]$ is the integral we want to compute. As we will learn, it is quite easy to use uniform random numbers on $[0, 1]$ to generate samples of X with the desired density. (It is not so easy to generate samples with density proportional to e^{-x^2} on $[0, 10]$.)

Example - yet more integration: Another example of the need for importance sampling is the following. Suppose we want to compute a d-dimensional integral over some region D. If D

is a parallelepiped, it is easy to generate a random vector uniformly distributed over D. But if D is a more complicated shape this can be quite difficult. One approach is to find a parallelepiped R that contains D. We then generate random vectors that are uniformly distributed on R, but we reject them when the vector is outside of D. How efficient this is depends on the ratio of the volume of D to that of R. Even if it is very difficult to generate uniform samples from D, it may be possible to find a domain D' which contains D and whose volume is not too much bigger than that of D. Then generating a uniform sample from D' and rejecting it if it is not in D can be a much more effecient method that generating a uniform sample from a parallelepiped that contains D.

Example - shortest path in a network: By a network we mean a collection of nodes and a collection of edges that connect two nodes. (Given two nodes there need not be an edge between them). For each edge there is a random variable that give the time it takes to traverse that edge. There random variables are taken to be independent. We fix a starting node and an ending node in the graph. The random variable we want to study is the minimum total time it takes to get from the starting node to the ending node. By minimum we mean the minimum over all possible paths in the network from the starting node to the ending node. For very small networks you work out the expected value of this random variable. But this analytic approach becomes intractable for every modest size networks. The Monte Carlo approach is to generate a bunch of samples of the network and for each sample network compute the minimum transit time. (This is easier said than done.) The average of these minima over the samples is then our estimate for the expected value we want.

Example - network connectivity or reliability We now consider a network but now it is random in a different way. We fix the nodes in the network. For each possible edge e, there is a parameter $p_e \in [0, 1]$. We include the edge in the graph with probability p_e. The edges are independent. Now we are interested in the probability that there is some path that connects the starting and ending nodes. Probabilties can be thought of as expected values and the strong law still provides the theoretical basis for a direct MC simulation. We can approximately compute the probability by generating a bunch of samples of the network and for each sample checking if the starting and ending nodes are connected. Our estimate for the probability of a connection is the number of samples that have a connection divided by the total number of samples.

Example - network connectivity with dependence In the previous example, if we let E_e be the event that the graph contains edge e, then these events are independent. In this example we again we fix the nodes in the network and make the edge configuration random. However, now the edges are not independent. There are many models that do this. Here is a relatively simple one. Let s_e be 0 if the edge e is not present, 1 if it is. So the graph is specified by the set of random variables s_e. Let s denote the collection $\{s_e\}$. We take the

probability density to be

$$P(s) = \frac{1}{Z} \exp(-\sum_e c_e s_e + \sum_{e,f} c_{e,f} s_e s_f) \tag{1.6}$$

where c_e and $c_{e,f}$ are parameters. The sum of pairs of edges e, f is only over $e \neq f$. The constant Z is determined by the requirement that this must be a probability measure. Even though the probability measure is quite explicit, there is no practical way to directly sample from it. In particular, computing Z is not feasible unless the number of nodes is small. But we can generate dependent samples of this distribution using a Markov Chain. Possible states are the possible edge configurations. If we are in state s the chain can transition to a new state s' which differs from s in only one edge. In other words, the transitions consist of either deleting an edge that is there or adding an edge that is not present. If we choose the transition probabilities for these transitions appropriately, the stationary distribution of the Markov chain will be P. So we can compute the probability that the starting and ending nodes are connected by running the chain.

Example - self-avoiding walks To be concrete we describe this in two dimensions on the square lattice. An N-step self-avoiding walk (SAW) is a nearest neighbor walk on the square lattice that does not visit a site more than once. We take all the N-step SAW's that start at the origin and put the uniform probability measure on this finite set. This is a really simple (to define) probability measure, but the number of such walks grows exponentially with N. So for large values of N there is no practical way to generate samples. We have to use a MCMC method.

Overview of the topics to be covered

Chapter 2 will introduce the basic idea of direct Monte Carlo and how one estimates the error involved. All Monte Carlo simulations require a source of randomness. Usually the starting point for this randomness is a pseudo-random number generator that produces numbers in $[0, 1]$ that look like they are uniformly distributed on the interval and independent. In chapter 3 we will look at the general structure of such generators and how one tests them. In chapter 4 we will study how you can use random numbers that are uniformly distributed on $[0, 1]$ to generate samples of a random variable with either a continuous or discrete distribution.

For all Monte Carlo methods the error in our estimate of the quantity we want to compute is of order $\sigma/\sqrt{n}$. In direct Monte Carlo the constant σ is just the standard deviation of the random variable whose mean we are computing. For MCMC σ is more involved. Obviously we can improve the estimate by increasing n, i.e., generating a larger number of samples. One can also attempt to reduce σ. This is known as variance reduction. We study some techniques for variance reduction in chapter 6. An important such technique that we touched on on some of the examples is importance sampling. All of chapter 7 is devoted to this topic.

In chapter 8 we will give a crash course on Markov chains, including the central idea of

MCMC. Chapter 9 will study some particular MCMC methods - the Gibbs sampler and the Metropolis-Hastings algorithm. In Chapter 10 we will look in more detail at the statistical analysis of what comes out of our Monte Carlo simulation. Further topics may include stochastic optimization, rare event simulation, perfect sampling, simulating annealing and who knows what.

Part I

Independent Monte Carlo

In chapter 2 we recall the strong law of large numbers and how it is the basis of Monte Carlo methods in which the samples are independent. Any Monte Carlo method needs a source of randomness. In chapter 3 we take a look at random number generators with the goal being to understand how they work at a somewhat abstract level and various tests for random number generators. These random number generators produce a real number that is uniformly distributed on $[0, 1]$. In chapter 4 we study how one uses such random numbers to generate samples of random variables and vectors, both continuous and discrete, with prescribed distributions.

Chapter 2

Basics of direct Monte Carlo

2.1 The probabilistic basis for direct MC

We start our study of Monte Carlo methods with what is usually called direct or simple Monte Carlo. We will refer to it as direct Monte Carlo.

We assume that our original problem can be put in the following form. There is a probability space (Ω, P) and a random variable X on it. The quantity we want to compute is the the mean of X which we denote by $\mu = E[X]$. (The sample space Ω is the set of possible outcomes. Subsets of Ω are called events, and the probability measure P a function that assigns a number between 0 and 1 to each event. Usually the probability measure is only defined on a σ-field $\mathcal{F}$, which is a sub-collection of the subsets of Ω, but we will not worry about this.) We emphasize that the original problem need not involve any randomness, even if it does the probability space we use for the Monte Carlo may not have been part of the original problem.

Let X_n be an independent, identically distributed (iid) sequence which has the same distribution as X. Recall that saying X_n and X are indentically distributed means that for all Borel sets B, $P(X_n \in B) = P(X \in B)$. A standard result in probability says that if they are equal for all B whic are intervals, then that is sufficient to insure they are equal for all Borel sets. The key theorem that underlies direct Monte Carlo is the Strong Law of Large Numbers.

Theorem 1 *Let X_n be an iid sequence such that $E[|X_n|] < \infty$. Let $\mu = E[X_n]$. Then*

$$P(\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} X_k = \mu) = 1 \tag{2.1}$$

The conclusion of the theorem is often written as $\lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \mu$ a.s. Here a.s. stands for almost surely.

Suppose we want to compute the probability of an event rather than the mean of a random variable. If E is the event, we can think of its probability $P(E)$ as the expected value of the indicator function of E, i.e., $P(E) = E[1_E]$. In this case the sample is bunch of 0's and $1's$ indicating whether or not the outcome was in E. The sample mean is the fraction of outcomes that were in E is usually denoted $\hat{p}_n$. In this situation the strong law says that $\hat{p}_n$ converges to $P(E)$ almost surely.

We have seen several examples of direct Monte Carlo in the introduction. The integration examples and the network examples with independent edges were all examples of direct Monte Carlo. Note that in the network example where we were concerned with connectivity, we were computing a probability. The network example in which the edges are not independent and the self-avoiding walk example cannot be studied with direct Monte Carlo. Unless the network or walk is really small, there is no practical way to generate samples from the probability distribution .

2.2 Error estimation

The strong law says that we can approximate μ by generating a sample $X_1, X_2, \cdots, X_n$ and then computing the sample mean

$$\hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n} X_k \tag{2.2}$$

Note that μ is a constant while $\hat{\mu}_n$ is a random variable. In the language of statistics, μ is a parameter and $\hat{\mu}_n$ is a statistic that estimates μ. We follow the notational convention of using a hat to denote the statistic that estimates the corresponding parameter.

The strong law tells us that $\hat{\mu}_n$ converges to μ but it does not tell us anything about how close $\hat{\mu}_n$ is to μ for a given value of μ. In any Monte Carlo simulation we do not actually let n go to infinity, we only use a (hopefully) large value of n. So it is crucial to address this question of how close our approximation is. $\hat{\mu}_n$ is a random variable. Since all the random variables X_i in our sample have mean μ, the mean of the sample mean is

$$E\hat{\mu}_n = \mu \tag{2.3}$$

In the language of statistics, μ_n is said to be an unbiased estimator of μ. We assume that the X_i have finite variance. Since they are identically distributed, they have the same variance

and we denote this common variance by σ. Since the X_i are independent, we have

$$var(\sum_{i=1}^{n} X_i) = n\sigma^2 \tag{2.4}$$

and so the variance of the sample mean is

$$var(\hat{\mu}_n) = \frac{1}{n^2} var(\sum_{i=1}^{n} X_i) = \frac{\sigma^2}{n} \tag{2.5}$$

Thus the difference of μ_n from μ should be of order $\sigma/\sqrt{n}$.

The $1/\sqrt{n}$ rate of convergence is rather slow. If you compute a one dimensional integral with Simpon's rule the rate of convergence is $1/n^4$. However, this rate of convergence requires some smoothness assumptions on the integrand. By contrast, we get the $1/\sqrt{n}$ rate of convergence in Monte Carlo without any assumptions other than a finite variance. While Simpson's rule is fourth order in one dimension, as one goes to higher dimensional integrals the rate of convergence gets worse as the dimension increases. In low dimensions with a smooth integrand, Monte Carlo is probably not the best method to use, but to compute high dimensional integrals or integrals with non-smooth integrands, Monte Carlo with it slow $1/\sqrt{n}$ convergence may be the best you can do.

The central limit theorem gives a more precise statemnt of how close $\hat{\mu_n}$ is to μ. Note that since $\hat{\mu}_n$ is random, even if n is very large, there is always some probability that $\hat{\mu}_n$ is not close to μ. The central limit theorem says

Theorem 2 *Let X_n be an iid sequence such that $E[|X_n|^2] < \infty$. Let $\mu = E[X_n]$ and let σ^2 be the variance of X_n, i.e., $\sigma^2 = E[X_n^2] - E[X_n]^2$. Then*

$$\frac{1}{\sigma\sqrt{n}} \sum_{k=1}^{n} (X_k - \mu) \tag{2.6}$$

converges in distribution to a standard normal random variable. This means that

$$\lim_{n\to\infty} P(a \le \frac{1}{\sigma\sqrt{n}} \sum_{k=1}^{n} (X_k - \mu) \le b) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx \tag{2.7}$$

In terms of the sample mean, the central limit theorem says that $(\hat{\mu}_n - \mu)\sqrt{n}/\sigma$ converges in distrbution to a standard normal distribution.

The statement of the central limit theorem involves σ^2. It is unlikely that we know σ^2 if we don't even know μ. So we must also use our sample to estimate σ^2. The usual estimator of

the variance σ^2 is the sample variance. It is typically denoted by s^2, but we will denote it by s_n^2 to emphasize that it depends on the sample size. It is defined to be

$$s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{\mu}_n)^2 \tag{2.8}$$

(s_n is defined to be $\sqrt{s_n^2}$.) A straightforward calculation show that $Es_n^2 = \sigma^2$, so s_n is an unbiased estimator of σ^2. This is the reason for the choice of $1/(n-1)$ as the normalization. It makes the estimator unbiased. An application of the strong law of large numbers shows that $s_n^2 \to \sigma^2$ a.s. Since $(\hat{\mu}_n - \mu)\sqrt{n}/\sigma$ converges in distribution to a standard normal distribution and that $s_n^2 \to \sigma^2$ converges almost surely to σ^2, a standard theorem in probability implies that $(\hat{\mu}_n - \mu)\sqrt{n}/s_n$ converges in distribution to a standard normal. (In statistics the theorem being used here is usually called Slutsky's theorem.) So we have the following variation on the central limit theorem.

Theorem 3 *Let X_n be an iid sequence such that $E[|X_n|^2] < \infty$. Let $\mu = E[X_n]$ and let σ^2 be the variance of X_n, i.e., $\sigma^2 = E[X_n^2] - E[X_n]^2$. Then*

$$\frac{(\mu_n - \mu)\sqrt{n}}{s_n} \tag{2.9}$$

converges in distribution to a standard normal random variable. This means that

$$\lim_{n\to\infty} P(a \le \frac{(\mu_n - \mu)\sqrt{n}}{s_n} \le b) = \int_a^b \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}\, dx \tag{2.10}$$

The central limit theorem can be used to construct confidence intervals for our estimate $\hat{\mu}_n$ for μ. We want to construct an interval of the form $[\hat{\mu}_n - \epsilon, \hat{\mu}_n + \epsilon]$ such that the probability μ is in this interval is $1 - \alpha$ where the confidence level $1 - \alpha$ is some number close to 1, e.g., 95%. Let Z be a random variable with the standard normal distribution. Note that μ belongs to $[\hat{\mu}_n - \epsilon, \hat{\mu}_n + \epsilon]$ if and only if $\hat{\mu}_n$ belongs to $[\mu - \epsilon, \mu + \epsilon]$. The central limit theorem says that

$$\begin{aligned} & P(\mu - \epsilon \le \hat{\mu}_n \le \mu + \epsilon) = P(-\epsilon \le \hat{\mu}_n - \mu \le \epsilon) \\ = \;& P(-\frac{\epsilon\sqrt{n}}{s_n} \le \frac{(\hat{\mu}_n - \mu)\sqrt{n}}{s_n} \le \frac{\epsilon\sqrt{n}}{s_n}) \approx P(-\frac{\epsilon\sqrt{n}}{s_n} \le Z \le \frac{\epsilon\sqrt{n}}{s_n}) \end{aligned} \tag{2.11}$$

Let z_c be the number such that $P(-z_c \le Z \le z_c) = 1 - \alpha$. Then we have $\frac{\epsilon\sqrt{n}}{s_n} = z_c$, i.e., $\epsilon = z_c s_n/\sqrt{n}$. Thus our confidence interval for μ is

$$\hat{\mu}_n \pm \frac{z_c s_n}{\sqrt{n}} \tag{2.12}$$

Stop - Wed, 1/20

Common choices for $1-\alpha$ are 95% and 99%, for which $z_c \approx 1.96$ and $z_c \approx 2.58$, respectively. The central limit theorem is only a limit statement about what happens as the sample size n goes to infinity. How fast the distribution in question converges to a normal distribution depends on the distribution of the original random variable X.

Suppose we are using direct Monte Carlo to compute the probability $p = P(E)$ of some event E. As discussed above we can think of this as computing the expected value of 1_E. The central limit theorem still applies, so we can construct confidence intervals for our estimate. The variance of 1_E is easily found to be $p(1-p)$. So we can use our estimate $\hat{p}_n$ of p to estimate the variance by $\hat{p}_n(1-\hat{p}_n)$ rather than s_n. Thus the confidence interval is

$$\hat{p}_n \pm \frac{z_c\sqrt{\hat{p}_n(1-\hat{p}_n)}}{\sqrt{n}} \tag{2.13}$$

This is one possible problem with the above. Suppose p is very small, so small that np is of order one. There is some chance that none of the outcomes in our sample will lie in the event E and so $\hat{p}_n = 0$. In this case the above confidence interval would be $[0,0]$. This is clearly nonsense. The problem with the above derivation of the confidence interval in this case is that if np is not reasonable large, the central limit theorem is not a good approximation. When $\hat{p}_n = 0$ a reasonable confidence interval can be obtained as follows. Obviously the confidence interval should be of the form $[0, p_0]$. Note that $P(\hat{p}_n = 0) = (1-p)^n$. If p is not small enough this is a small probability. But $\hat{p}_n$ did in fact equal 0. So we will choose p_0 so that $(1-p_0)^n = \alpha$, where $1-\alpha$ is the confidence level, e.g., 95%. This is the same as $n\ln(1-p_0) = \ln(\alpha)$. Since p_0 is small, $\ln(1-p_0) \approx -p_0$. So we let

$$p_0 = \frac{-\ln(\alpha)}{n} \tag{2.14}$$

With $\alpha = 5\%$, $-\ln(\alpha)$ is approximately 3, so the confidence interval is $[0, 3/n]$.

Another approach to treating $\hat{p}_n$ is the Agresti confidence interval. See Owen for a discussion.

2.3 Accuracy vs. computation time

We have seen that the error in a Monte Carlo computation is proportional to $\sigma/\sqrt{n}$. Obviously we can reduce the error by increasing the number of samples. Note, however, that

to reduce the error by half we must increase the number of samples by a factor of four. Another way to improve the accuracy is to reduce the variance σ^2. We will study this topic in detail in a later chapters.

It is important to keep in mind that from a practical point of view, what is important is not how many samples are needed to achieve a given level of accuracy, but rather how much CPU time is need to achieve that accuracy. Suppose we have two Monte Carlo methods that compute the same thing. They have variances σ_1^2 and σ_2^2. Let τ_1 and τ_2 be the CPU time needed by the methods to produce a single sample. Then with a fixed amount T of CPU time we can produce $N_i = T/\tau_i$ samples for the two methods. So the errors of our two methods with be

$$\frac{\sigma_i}{\sqrt{N_i}} = \frac{\sigma_i\sqrt{\tau_i}}{\sqrt{T}} \tag{2.15}$$

Thus the method with the smaller $\sigma_i^2\tau_i$ is the better method.

It is also important to keep in mind that the time needed to compute a sample typically consists of two parts. First we have to generate a sample ω from the probability space. Then we have to evaluate the random variable X on ω. In many applications this second step can be as time consuming (or more so) than the first step. As an illustration, consider the network reliability example from the introduction. To generate a sample, all we have to do is generate a uniformly distributed random number from $[0, 1]$ for each edge and compare that random number with p_e to decide if the edge is included in the network or not. This take a time proportional to the number of possible edges. Finding the shortest path in the resulting network can take much longer.

Many problems involve a size or scale. In the network examples there is the number of edges. In the self-avoiding walk we have the number of steps. In integration problems there is the dimension. One should pay attention to how the times required for different parts of the Monte Carlo simulation depend on n. In particular one should keep in mind that while one part of the computation may be the most time consuming for moderate values of n, for larger values of n another part of the computation may start to dominate.

2.4 How good is the confidence interval

Our confidence interval was chosen so that the probability that the confidence interval contains the mean μ is the given confidence level $1 - \alpha$. In working this out we used the central limit theorem which is only an approximation which gets better as n increases. If the distribution of X is normal, then for any n the distribution of μ_n is a well-studied distribution known as student's t distribution. One can use this distribution (with $n - 1$ degrees of

freedom) in place of the standard normal. Of course this is only a reasonble thing to do if the distribution of X is approximately normal. Unless n is pretty small, the effect of using student's t instead of normal is negligible. With a confidence level of 95%, the critical z from the normal distribution is 1.960 while the critical t value for $n = 100$ is 1.984 and for $n = 20$ is 2.086. So unless you are in the very usual situation of doing a Monte Carlo with a very small number of samples, there is no need to use students t.

Even if n is not small, one can still worry about how much error the central limit theorem approximation introduces, i.e., how close is $P(\hat{\mu}_n - \frac{z_c s_n}{\sqrt{n}} \leq \mu \leq \hat{\mu}_n + \frac{z_c s_n}{\sqrt{n}})$ to $1 - \alpha$? Typically it is off by something of order $1/n$, so this is not really an issue for large values of n.

If we want to be really paranoid and we have an a priori upper bound on σ^2, then we can use Chebyshev's inequality. It says that for any $\epsilon > 0$,

$$P(|\hat{\mu}_n - \mu| \geq \epsilon) \leq \frac{1}{\epsilon^2} E[(\hat{\mu}_n - \mu)^2] = \frac{1}{\epsilon^2} var(\hat{\mu}_n) = \frac{1}{n\epsilon^2}\sigma^2 \tag{2.16}$$

Suppose we know that $\sigma^2 \leq M$. Then the above is bounded by $M/(n\epsilon^2)$. If we set this equal to α, we get $\epsilon = \sqrt{\frac{M}{n\alpha}}$. So if we take the confidence interval to be

$$\hat{\mu}_n \pm \sqrt{\frac{M}{n\alpha}} \tag{2.17}$$

then Chebyshev insures that the probability μ is not in this confidence interval is at most α. Comparing with our central limit theorem confidence interval we see that $z_c s_n$ has been replaced by $\sqrt{M/\alpha}$. If M is close to σ^2, then s_n is close to $\sqrt{M}$ and so the effect is to replace z_c by $1/\sqrt{\alpha}$. For $\alpha = 5\%$ this amounts to replacing 1.96 by 4.47. So we can get a confidence interval for which we are certain that the probability the interval does not capture μ is at most 5%, but at the expense of a much wider confidence interval.

Finally, if one does not have a bound on σ^2, but the random variable X is known to always be in the interval $[a, b]$, then one can use Hoeffding's inequality in place of Chebyshev. See Owen for more on this.

2.5 Estimating a function of several means

Sometimes the quantity we want to compute is the quotient of two means

$$\theta = \frac{E[X]}{E[Y]} \tag{2.18}$$

Suppose we generate an independent samples $\omega_1, \cdots, \omega_n$ from our probability space, distributed according to P. We then let $X_i = X(\omega_i)$ and $Y_i = Y(\omega_i)$. We want to use the

sample $(X_1, Y_1), \cdots, (X_n, Y_n)$ to estimate θ. Note that in this approach X_i and Y_i are not independent. The natural estimator for θ is

$$\hat{\theta} = \frac{\overline{X}_n}{\overline{Y}_n} \tag{2.19}$$

Here we use the notation

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \overline{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{2.20}$$

for the two sample means. The nontrivial thing here is to find a confidence interval for our estimate. We do this using the Delta method.

There is nothing special about ratios. More generally we can consider a function of several means. So we assume we have a random vector $(X_1, X_2, \cdots, X_d)$ and we want to estimate a function of their means

$$\theta = f(E[X_1], \cdots, E[X_d]) \tag{2.21}$$

for some function f on R^d. Since we are using subscripts to indicate the components, we will now use superscripts to label our samples. We suppose we have n i.i.d. samples of our random vector. We denote them by $(X_1^i, \cdots, X_d^i)$ where $i = 1, 2, \cdots, n$. We let $\hat{\mu}_j^n$ be the sample mean of the jth component

$$\hat{\mu}_j^n = \frac{1}{n}\sum_{i=1}^{n} X_j^i \tag{2.22}$$

The natural estimator for θ is

$$\hat{\theta} = f(\hat{\mu}_1, \cdots, \hat{\mu_d}) \tag{2.23}$$

To get a confidence interval we need a multivariate version of the central limit theorem. We first recall a couple of definitions. The covariance of X and Y is
$cov(X, Y) = E[XY] - E[X]E[Y]$. Letting $\mu_x = E[X]$ and $\mu_y = E[Y]$, the covariance can also be written as

$$cov(X, Y) = E[(X - \mu_x)(Y - \mu_y)] \tag{2.24}$$

The correlation of X and Y is

$$\rho = \frac{cov(X, Y)}{\sigma_x \sigma_y} \tag{2.25}$$

where σ_x^2, σ_y^2 are the variance of X and Y. We let σ_j^2 be the variance of X^j, and let $\rho_{j,k}$ be the correlation of X_j and X_k. So the covariance of X_j and X_k is $\rho_{j,k}\sigma_j\sigma_k$. As $j, k = 1, 2, \cdots, d$ this gives a $d \times d$ matrix which we denote by Σ. It is called the covariance matrix.

Theorem 4 *Let $(X_1^n, \cdots, X_d^n)$ be an i.i.d. sequence of random vectors with finite variances. Let $\mu_j = E[X_j^n]$, let σ_j^2 be the variance of X_j^n and $\rho_{j,k}$ be their correlations. Then*

$$\frac{1}{\sqrt{n}} \sum_{k=1}^{n} (X_1^k - \mu_1, \cdots, X_d^k - \mu_d) \tag{2.26}$$

converges in distribution to a multivariate normal random variable with zero means and covariance matrix Σ.

The crucial fact we will need about a multivariate normal distribution is the following. Let $(Z_1, Z_2, \cdots, Z_d)$ be a multivariate normal distribution with zero mean and covariance matrix Σ. Then the linear combination $\sum_{j=1}^{d} c_j Z_j$ is a normal random variable with mean zero and variance equal to

$$\sum_{j,k=1}^{d} c_j c_k \Sigma_{j,k} \tag{2.27}$$

Now we turn to the delta method. The idea is simple. We just use a first order Taylor expansion of f about $(\mu_1, \cdots, \mu_d)$ and use the central limit theorem.

$$\hat{\theta} = f(\hat{\mu}_1, \cdots, \hat{\mu}_d) \approx f(\mu_1, \cdots, \mu_d) + \sum_{j=1}^{d} f_j(\mu_1, \cdots, \mu_d)(\hat{\mu_j} - \mu_j) \tag{2.28}$$

where f_j denotes the jth partial derivative of f. The mean of the right side is $f(\mu_1, \cdots, \mu_d)$. This says that to first order the estimator $\hat{\theta}$ is an unbiased estimator. It is not exactly unbiased. By looking at the second order Taylor expansion one can see that the bias is of order $1/n$. The central limit theorem says that the vector with components $\hat{\mu_j} - \mu_j$ has approximately a multivariate normal distribution with covariance matrix Σ/n. So the variance of $\hat{\theta}$ is

$$var(\hat{\theta}) = \frac{1}{n} \sum_{j,k=1}^{d} f_j(\mu_1, \cdots, \mu_d) f_k(\mu_1, \cdots, \mu_d) \Sigma_{j,k} \tag{2.29}$$

This can be written more succintly as $(\nabla f, \Sigma \nabla f)/n$

Before our application of the central limit theorem involved σ which we do not know. So we had to replace σ by s_n. Here there are several things in the above that we do not know: μ_i and Σ. We approximate $f_j(\mu_1, \cdots, \mu_d)$ by $f_j(\hat{\mu_1}, \cdots, \hat{\mu_d})$. We denote the resulting approximation of ∇f by $\hat{\nabla} f$. We approximate Σ by $\hat{\Sigma}$ where the entries are

$$\hat{\Sigma}_{j,k} = \frac{1}{n} \sum_{i=1}^{n} (X_j^i - \hat{\mu}_j)(X_k^i - \hat{\mu}_k) \tag{2.30}$$

So our estimate for the variance of $\hat{\theta}$ is $(\hat{\nabla} f, \hat{\Sigma}\hat{\nabla} f)/n$. Thus the confidence interval is

$$\hat{\theta} \pm \frac{z_c}{\sqrt{n}} \sqrt{(\hat{\nabla} f, \hat{\Sigma}\hat{\nabla} f)} \tag{2.31}$$

We can now return to the problem of estimating the ratio $\theta = E[X]/E[Y]$. So $n = 2$ and $f(x, y) = x/y$. Some computation leads to the follows. The variance of $\hat{\theta}$ is approximately

$$\frac{1}{n}(\hat{\nabla} f, \hat{\Sigma}\hat{\nabla} f) = \frac{1}{n} \frac{\sum_{i=1}^{n} (Y_i - \hat{\theta} X_i)^2}{n\overline{X}^2} \tag{2.32}$$

where $\overline{X}$ and $\overline{Y}$ are the sample means for X and Y and $\hat{\theta} = \overline{X}/\overline{Y}$.

2.6 References

Most books on Monte Carlo include the topics in this section. Our treatment follows Owen closely. Fishman's *A first course in Monte Carlo* has some nice examples, one of which we have used (networks).

Chapter 3

Pseudo-random numbers generators

3.1 Basics of pseudo-random numbers generators

Most Monte Carlo simulations do not use true randomness. It is not so easy to generate truly random numbers. Instead, pseudo-random numbers are usually used. The goal of this chapter is to provide a basic understanding of how pseudo-random number generators work, provide a few examples and study how one can empirically test such generators. The goal here is not to learn how to write your own random number generator. I do not recommend writing your own generator. I also do not recommend blindly using whatever generator comes in the software package your are using. From now on we will refer to pseudo random number generators simply as random number generators (RNG).

The typical structure of a random number generator is as follows. There is a finite set S of states, and a function $f : S \to S$. There is an output space U, and an output function $g : S \to U$. We will always take the output space to be $(0, 1)$. The generator is given an initial value S_0 for the state, called the seed. The seed is typically provided by the user. Then a sequence of random numbers is generated by defining

$$\begin{aligned} S_n &= f(S_{n-1}), \quad n = 1, 2, 3, \cdots \\ U_n &= g(S_n) \end{aligned} \tag{3.1}$$

Note that there is nothing random here. If we generate a sequence of numbers with this procedure and then generate another sequence using the same seed, the two sequences will be identical. Since the state space is finite, S_n must eventually return to a state it was in some time earlier. The smallest integer p such that for some state the function f returns to that state after p iterations is called the period of the generator. Obviously, longer periods are better than short periods. But a long period by itself certainly does insure a good random

number generator.

We list some of the properties we want in our generator. Of couse we want it to produce an i.i.d. sequence with the uniform distribution on $(0, 1)$. That by itself is a rather abstract mathematical requirement; the first two properties below make it more practical.

1. **Pass empirical statistical tests** These are tests where you generate a long sequence of random numbers and then perform various statistical tests to test the hypothesis that the numbers are uniformly distributed on $[0, 1]$ and are independent.

2. **Mathematical basis** There is a lot of mathematical theory behind these random number generators (at least some of them) including properties that should produce a good random number generator. Obviously, we want a large period, but there are more subtle issues.

3. **Fast (and not a lot of memory)** Most Monte Carlo simulations require a huge number of random numbers. You may want to generate a large number of samples, and the generation of each sample often involves calling the random number generator many times. So the RNG needs to be fast. With most generators memory is not really an issue, although some can take up a lot of cache.

4. **Multiple streams** It is easy to use parallel computation for Monte Carlo. But this requires running multiple copies of your random number generator. So you need to be sure that these different streams are independent of each other.

5. **Practical concerns** Ease of installation and ease of seeding. Some random number generators are quite short and only take a few lines of code. Others are considerably more complicated. Some like the Mersenne twister require a rather large seed. Many of the better generators use 64 bit integers. So be sure you are working on a 64 bit system and type your variables appropriately.

6. **Reproducibility** For debugging and testing purposes you want to be able to generate the same stream of random numbers repeatedly. For any random number generator of the form we are considering this is easy - just start with the same seed.

3.2 Some examples

We do not attempt to give all the different types of generators. We discuss a few different types with some specific examples.

3.2.1 Linear congruential generators

The state space is $\{0, 1, 2, \cdots, m-1\}$ where m is a positive integer.

$$f(X) = aX + c \bmod m \tag{3.2}$$

where mod m means we do the arithmetic mod m. The constants a and c are integers and there is no loss of generality to take them in $\{0, \cdots, m-1\}$. For the output function we can take

$$g(X) = \frac{X}{m} \tag{3.3}$$

The quality of this generator depends on the choice of the constants a and c. There is a lot of mathematics behind how these constants should be chosen which we will not go into.

Example: Lewis, Goodman, and Miller, proposed the choice $a = 7^5 = 16807, c = 0$ and and $m = 2^{31} - 1 = 2147483647$. It has period $2^{31} - 2$.

drand48(): This is a linear congrutial generator which uses $m = 48$,
$a = 5DEECE66D_{16} = 273673163155_8$, $c = B_{16} = 13_8$. It is part of the standard C library. It is not recommended.

A bunch of examples of (not necessarily good) LCG is at :

random.mat.sbg.ac.at/results/karl/server/node4.html

3.2.2 Multiple-recursive generators

We take the state space to be $\{0, 1, 2, \cdots, m-1\}^k$ and define the function f by

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \cdots a_k X_{n-k}) \bmod m \tag{3.4}$$

The notation is inconsistent here. Before X_n was the state at time n. Now the state at time n is $(X_n, X_{n-1}, \cdots, X_{n-k+1})$. Then the output is $U_n = X_n/m$. With suitable choices of m and a_i we can get a period of $m^k - 1$.

There are more complicated linear generators, e.g., matrix multiplicative recursive generators, generalized feedback shift registers, and twisted generalized feedback shift registers, but we do not discuss them. A widely used example of the latter is the Mersenne twister, MT19937, invented by Matsumoto and Nishimura. It is implemented in MATLAB and SPSS. It has a very large period, $2^{19937} - 1$. Code for it can be found at

http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/emt.html

3.2.3 Combining generators

A common trick in designing random number generators is to combine several not especially good random number generator. An example is the Wichman-Hill generator which combines three linear congruential generators. The state space is
$\{0, 1, 2 \cdots, m_1 - 1\} \times \{0, 1, 2 \cdots, m_2 - 1\} \times \{0, 1, 2 \cdots, m_3 - 1\}$. We denote the state at step n by (X_n, Y_n, Z_n). Then the generator is

$$\begin{aligned} X_n &= 171 X_{n-1} \bmod m_1 \\ Y_n &= 172 Y_{n-1} \bmod m_2 \\ Z_n &= 170 Z_{n-1} \bmod m_3 \end{aligned} \tag{3.5}$$

with $m_1 = 30269, m_2 = 30307, m_3 = 30323$. The output function is

$$U_n = \frac{X_n}{m_1} + \frac{Y_n}{m_2} + \frac{Z_n}{m_3} \bmod 1 \tag{3.6}$$

The period is approximately 7^{12}, large but not large enough for large scale Monte Carlo.

We can also combine multiple-recursive generators. L'Ecuyer's MRG32k3a is an example which employs two MRGs of order 3:

$$\begin{aligned} X_n &= (1403580 X_{n-2} - 810728 X_{n-3}) \bmod m_1 \\ Y_n &= (527612 Y_{n-1} - 1370589 Y_{n-3}) \bmod m_2 \end{aligned} \tag{3.7}$$

with $m_1 = 2^{32} - 209$, $m_2 = 2^{32} - 22853$. The output function is

$$U_t = \begin{cases} \frac{X_n - Y_n + m_1}{m_1 + 1}, & \text{if } X_n \le Y_n, \\ \frac{X_n - Y_n}{m_1 + 1}, & \text{if } X_n > Y_n, \end{cases} \tag{3.8}$$

The period is approximately 3×10^{57}. This generator is implemented in MATLAB.

Some very simple and fast RNG Marsagalia's KISS generators. He posted some of these in the forum:

http://www.thecodingforums.com/threads/64-bit-kiss-rngs.673657/

Some more examples like this can be found in the article by David Jones (reference at end of section).

Stop - Mon, 1/25

3.3 A little statistics

We briefly explain two statistical tests - χ^2 goodness of fit and the Kolmogorov-Smirnov goodness of fit tests. These are worth discussing not just because there are used in testing random number generators but also because they are used to analyze what comes out of our Monte Carlo simulation.

The χ^2 distribution: Let $Z_1, Z_2, \cdots Z_k$ be independent random variables, each of which has a standard normal distribution. Let

$$X = \sum_{i=1}^{k} Z_i^2 \tag{3.9}$$

The distribution of X is called the χ^2 distribution. Obviously, it depends on the single integer parameter k. k is called the "degrees of freedom." The density of X can be explicitly found:

$$f_X(x) = cx^{k/2-1}e^{-x/2}, x > 0 \tag{3.10}$$

The multinomial distribution: This is a discrete distribution. Fix an integer k and probabilities $p_1, p_2, \cdots p_k$ which sum to one. Let $X_1, X_2, \cdots, X_n$ be independent random variables with values in $\{1, 2, \cdots, k\}$ and $P(X_j = l) = p_l$. Let O_j be the number of $X_1, X_2, \cdots, X_n$ that equal j. (O stands for observed, this is the observed number of times j occurs.) There is an explicit formula for the joint distribution of $O_1, O_2, \cdots, O_k$, but we will not need it. We will refer to the parameter n as the number of trials. Note that the mean of O_j is $p_j n$.

Suppose we have a random variable X with values in $\{1, 2, \cdots, k\}$. We want to test the hypothesis that X has a prescribed distribution $P(X = j) = p_j$. We generate n independent samples X_i of X, and let O_j be the number that take on the value j as we did above. We let $e_j = np_j$, the mean of O_j under the null hypothesis, and consider the statistic

$$V = \sum_{j=1}^{k} \frac{(O_j - e_j)^2}{e_j} \tag{3.11}$$

Theorem 5 *If X has the distribution $P(X = j) = p_j$, then as $n \to \infty$ the distribution of V defined above converges to the χ^2 distribution with $k - 1$ degrees of freedom.*

Any software package with statistics (or any decent calculator for that matter) can compute the χ^2 distribution. As a general rule of thumb, the number of observations in each "bin" or "class" should be at least 5.

The Kolmogorov-Smirnov goodness of fit test is a test that a random variable has a particular continuous distribution. We start with an easy fact:

Theorem 6 *Let X be a random variable with a continuous distribution function $F(x)$. We assume that F is strictly increasing on the range of X. Let $U = F(X)$. Then the random variable U is uniformly distributed on $[0, 1]$.*

Proof: Since F is strictly increasing on the range of X. So we can define an inverse F^{-1} that maps $(0, 1)$ to the range of X. For $0 < t < 1$,

$$P(U \leq t) = P(F(X) \leq t) = P(X \leq F^{-1}(t)) = F(F^{-1}(t)) = t \tag{3.12}$$

QED.

Now fix a distribution with continuous distribution function (CDF) $F(x)$. (This test does not apply to discrete distributions.) We have a random variable and we want to test the null hypothesis that the CDF of X is $F(x)$. Let $X_1, X_2, \cdots, X_n$ be our sample. Let $X_{(1)}, X_{(2)}, \cdots, X_{(n)}$ be the sample arranged in increasing order. (The so-called order statistics.) And let $U_{(i)} = F(X_{(i)})$. Note that $U_{(1)}, U_{(2)}, \cdots, U_{(n)}$ are just $U_1, U_2, \cdots, U_n$ arranged in increasing order where $U_i = F(X_i)$. Under the null hypothesis we expect the $U_{(i)}$ to be roughly uniformly distributed on the interval $[0, 1]$. In particular $U_{(i)}$ should be roughly close to $(i - 1/2)/n$. (The difference should be on the order of $1/\sqrt{n}$.)

$$\begin{aligned} D &= \frac{1}{2n} + \max_{1 \leq i \leq n} |F(X_{(i)}) - \frac{i - \frac{1}{2}}{n}| && (3.13) \\ &= \frac{1}{2n} + \max_{1 \leq i \leq n} |U_{(i)} - \frac{i - \frac{1}{2}}{n}| && (3.14) \end{aligned}$$

Note that if the null hypothesis is true, then the U_i are uniform on $[0, 1]$ and so the distribution of D does not depend on $F(x)$. It only depends on n. There is a formula for the distribution of D involving an infinite series. More important, software will happily compute p-values for this statistic for you.

3.4 Tests for pseudo-random numbers generators

In the following we let U_n be the sequence of random numbers from our generator. We want to test the null hypothesis that they are independent and each U_n is uniformly distributed on $[0, 1]$ In the following the null hypothesis will always mean the hypothesis that the U_n are i.i.d. and uniform on $[0, 1]$.

3.4.1 Equidistribution tests

One way to test that the U_n are uniformly distributed is to just use the Kolmogorov-Smirnov test. Here $F(x) = x$. Note that this does not test the independence of the U_n at all.

Another way to test the uniform distribution is the following. Fix a positive integer m. Let

$$Y_n = \lfloor mU_n \rfloor \tag{3.15}$$

where $\lfloor x \rfloor$ is the floor function which just rounds x down to the nearest integer. So Y_n takes on the values $0, 1, 2, \cdots, m-1$. Under the null hypothesis the Y_n are independent and uniformly distributed on $\{0, 1, \cdots, m-1\}$. So we can do a χ^2 test using V above, where O_j is the number of times Y_i equals $j+1$. Note that this only tests that the U_n are uniformly distributed on $[0, 1]$.

To test the independence (as well as the uniformity) we can do the following. Fix another integer d. Use the random number generator to generate random d-tuples of numbers: $(U^1, U^2, \cdots, U^d)$. Generate n such d-tuples, call them $(U_i^1, U_i^2, \cdots, U_i^d)$ where $i = 1, 2, \cdots, n$. (So we call the RNG nd times.)

$$Y_i^j = \left\lfloor mU_i^j \right\rfloor \tag{3.16}$$

Then the $(Y_i^1, Y_i^2, \cdots, Y_i^d)$ should be uniformly distributed over $\{0, 1, 2, \cdots, m-1\}^d$. We can test this with χ^2 test. This tests the independence to some extent, but it only tests if d consecutive calls to the RNG are independent. Note that the number of cells in the χ^2 test is m^d, so d cannot be too large.

3.4.2 Gap tests

Fix an interval $(\alpha, \beta) \subset (0, 1)$. Let T_n be the times when the random number is in (α, β). Let Z_n be the gaps between these times, i.e., $Z_n = T_n - T_{n-1}$. (We take $T_0 = 0$.) If the random numbers are i.i.d. and uniform, then the Z_n should be i.i.d. with a geometric distribution with parameter $p = \beta - \alpha$. We can test this with a χ^2 test. Fix an integer r and take the classes to be $Z = 0, Z = 1, \cdots, Z = r-1$ and $Z \geq r$.

A special case is $(\alpha, \beta) = (0, 1/2)$ which is called runs above the mean. With $(\alpha, \beta) = (1/2, 1)$ it is called runs below the mean.

3.4.3 Permutation tests

Use the RNG to generate random d-tuples of numbers: $(U^1, U^2, \cdots, U^d)$. For each d-tuple let π be the permutation which puts them in increasing order, i.e., $U^{\pi(1)} < U^{\pi(2)} < \cdots < U^{\pi(d)}$. Under the null hypothesis that the random numbers are i.i.d. uniform, the permutations will have the uniform distribution on the set of $d!$ permutations. We can test this with a χ^2 test.

3.4.4 Rank of binary matrix test

This one looks pretty crazy, but it has been useful in showing some RNG's that pass some of the simpler tests are not good.

Convert the i.i.d. sequence U_n to an i.i.d. sequence B_n which only takes on the values 0 and 1 with probability 1/2. For example let B_n be 0 if $U_n < 1/2$, $B_n = 1$ if $U_n \geq 1/2$. Fix integers r and c with $r \leq c$. Group the B_n into groups of size rc and use each group to form an $r \times c$ matrix. Then compute the rank of this matrix using arithmetic mod 2. There is an explicit formula for the distribution of this rank under the null hypothesis. (See Kroese review article for the formula.) We can then do a χ^2 test. For example, take $r = c = 32$. The rank can be at most 32. Note that it is very unlikely to get a rank a lot less than 32. So you don't want to take classes running over all possible values of the rank. With $r = c = 32$ we could take the classes to be $R \leq 30$,$R = 31$, and $R = 32$, where R is the rank.

3.4.5 Two-stage or second order tests

Consider one of the above tests (or many others). Let T be the test statistic. We assume that the distribution of T under the null hypothesis is known. The simple test is to generate a sample, compute the value of T and then compute its p-value. (Explain what a p-value is). A small p value, e.g., less than 5% means we got a value of T that is pretty unlikely if the null hypothesis is true. So we reject the null hypothesis (and so conclude our RNG is suspect). Now suppose we repeat this 50 times. So we get 50 values $T_1, T_2, \cdots T_{50}$ of the test statistic. We then get 50 p-values. Even if the null hypothesis is true, we expect to get a few p-values less than 5%. So we shouldn't reject the null hypothesis if we do. But we can use our 50 values of T to carry out a second order test. If the null hypothesis is true, then $T_1, T_2, \cdots T_{50}$ should be a sample from the known distribution of the test statistic. We can test this with the KS statistic and compute a single p-value which tests if the T_i do follow the known distribution.

3.4.6 Test suites

We end our discussion of testing with a few of the well known packages for testing a RNG.

George Marsaglia developed a suite of 15 tests which he called the diehard suite. Marsaglia passed away in 2011, but google will happily find the diehard suite. Robert Brown at Duke expanded the suite and named it dieharder:

http://www.phy.duke.edu/ rgb/General/dieharder.php

A comprehensive set of tests is the TestU01 software library by LEcuyer and Simard:

http://www.iro.umontreal.ca/ lecuyer/myftp/papers/testu01.pdf

Another important use of random numbers is in cryptography. What makes a good RNG for this (unpredictability) is not exactly the same as what make a good one for Monte Carlo. In particular, speed is not so important for some crytography applications. So in looking at test suites you should pay attention to what they are testing for.

3.5 Seeding the random number generator

How you seed the random number generator is important.

For parallel computation we certainly don't want to use the same seed.

MT and bad seeds

The seed of a RNG is a state for the RNG and so is typically much larger than a single random number. For the MT the seed requires ?? So it is nice to have an automated way to generate seeds.

A cheap way to generate a seed is to use some output from the operating system and then run it through a hash function. For example, on a unix system

ps -aux | md5sum

will produce a somewhat random 128 bit number. (The output is in hexadecimal, so it is 32 characters long.)

A more sophisticated approach on unix system is to use /dev/urandom. This uses "noise" in the computer to generate a random number. Note that it tests how much entropy has accumulated since it was last called and waits if there is not enough. So this is too slow to be

used for your primary RNG. On a unix system

od -vAn -N4 -tu4 < /dev/urandom

will give you some sort of random integer. (Need to find out just what this returns.)

3.6 Practical advice

- Don't use built in RNG unless you know what it is and how it has been tested.
- Design your code so that it is easy to change the RNG you use.
- Use two different RNG and compare your results. This is not a waste of time as you can always combine the two sets of data. Or if you do not want to go to the trouble of a second generator, try things like using only every fifth random number from the generator.
- Don't use too many random numbers from your generator compared to your period. There are many ad hoc rules of thumb : use at most $P/1000$, $\sqrt{P}/200$ or even $P^{1/3}$.

3.7 References

I have followed the chapter on random number generation in Kroese's review article and also taken some from "Good Practice in (Pseudo) Random Number Generation for Bioinformatics Applications" by David Jones. It is a nice practical discussion of picking a RNG. It can be found at

http://www0.cs.ucl.ac.uk/staff/d.jones/GoodPracticeRNG.pdf

For an interesting example of how supposedly good RNG lead to wrong results, see

Monte Carlo simulations: Hidden errors from "good" random number generators Alan M. Ferrenberg, D. P. Landau, and Y. Joanna Wong, Phys. Rev. Lett. 69, 3382 (1992).

Stop - Wed, 1/27

Chapter 4

Generating non-uniform random variables

4.1 Inversion

We saw in the last chapter that if the CDF is strictly increasing, then $F(X)$ has a uniform distribution. Conversely, it is easy to show in this case that if U is uniformly distributed on $[0,1]$ then $F^{-1}(U)$ has the distribution $F(x)$. For this we do not need that the CDF is strictly increasing. In this case the usual inverse function need not be defined. We define

$$F^{-1}(u) = \inf\{x : F(x) \geq u\} \tag{4.1}$$

for $0 < u < 1$. (really should not denote this by F^{-1} since it is not the inverse of F.)

Theorem 7 *Let $F(x)$ be a CDF. Define F^{-1} as above. Let U be uniformly distributed on $[0,1]$. Then the CDF of $F^{-1}(U)$ is $F(x)$.*

Proof: Consider the set $\{x : F(x) \geq U\}$. Since F is non-decreasing it must be of the form (c, ∞) or $[c, \infty)$. The right continuity of F implies it must be $[c, \infty)$. We need to compute $P(F^{-1}(U) \leq t)$. We claim that $F^{-1}(U) \leq t$ if and only if $U \leq F(t)$. (This is not quite as trivial as the notation makes it look since F^{-1} is not really the inverse function for F.) The claim will complete the proof since the probability that $U \leq F(t)$ is just $F(t)$.

First suppose $F^{-1}(U) \leq t$. Then t must belong to the set $\{x : F(x) \geq U\}$. So $F(t) \geq U$. Now suppose that $U \leq F(t)$. Then t belongs to $\{x : F(x) \geq U\}$. So t is greater than or equal to the inf of this set, i.e., $t \geq F^{-1}(U)$.

QED

In principle this allows us to generate samples of any distribution. The catch is that it may be difficult to compute F^{-1}. Note that this works for any CDF, continuous or discrete.

Example (exponential) The CDF of the exponential is

$$F(x) = 1 - \exp(-\lambda x) \tag{4.2}$$

and so

$$F^{-1}(u) = -\frac{\ln(1-u)}{\lambda} \tag{4.3}$$

So if U is uniform on $[0,1]$, then $-\ln(1-U)/\lambda$ has the exponential distribution with parameter λ. Note that if U is uniform on $[0,1]$ then so is $1-U$. So we can also use $-\ln(U)/\lambda$ to generate the exponential distribution.

Example - discrete distributions Let $x_1 < x_2 < \cdots < x_n$ be the values and p_i their probabilities. The CDF is piecewise constant and F^{-1} only takes on the values $x_1, x_2, \cdots, x_n$. The value of $F^{-1}(U)$ is x_k where k is the integer such that

$$\sum_{i=1}^{k-1} p_i < U \leq \sum_{i=1}^{k} p_i \tag{4.4}$$

This corresponds to the obvious way to generate a discrete distribution.

To implement this we need to find the k that satisfies the above. We can do this by searching starting at $k = 1$ and computing the partial sums as we go. This takes a time $O(n)$.

If we are going to generate a random variable with this same set of p_i many times we can do better. To implement this we first set up a table with the partial sums above. This takes a time $O(n)$, but we do this only once. Then when we want to generate a sample from the distribution we have to find the k that satisfies the above. If we use a bisection search we can reduce the time to $O(\ln(n))$. **Explain bisection** So there is a one time cost that is $O(n)$ and then the cost per sample is $O(\ln(n))$.

If we have a discrete distribution with an infinite number of values we can still do inversion. Doing this by starting at $k = 1$ and searching for the k that satisfies the above is straightforward. How long does this take on average? Is there a version of bisection that does better?

Example - Normal distribution At first glance it looks like we cannot use the inversion method for the normal distribution since we cannot explicitly compute the CDF and so cannot compute its inverse. Nonetheless, there are very fast methods for computing F^{-1} to high accuracy. See Owen for more details.

Scaling and shifting For many families of random variables there is a parameter that just corresponds to scaling the random variable, i.e., multiplying by a constant, or a parameter that corresponds to shifting the random variable, i.e., adding a constant.

For example consider the exponential random variable which has density

$$f(x) = \lambda \exp(-\lambda x), x \geq 0 \tag{4.5}$$

If Y is exponential with parameter 1 and we let $X = \lambda Y$ then X is exponential with the above density.

The normal density with mean μ and variance σ^2 is

$$f(x) = c \exp(-\frac{1}{2}(x-\mu)^2/\sigma^2) \tag{4.6}$$

If Z is a standard normal then $X = \sigma Z + \mu$ will have the above density.

The above can be thought of as applying an affine transformation to our random variable. If $g(x)$ is an increasing function and we let $Y = g(X)$ then the density of Y is related to that of X by

$$f_Y(y) = \frac{f_X(g^{-1}(y))}{g'(g^{-1}(y))} \tag{4.7}$$

4.2 Acceptance-rejection

The method of acceptance-rejection sampling is also called rejection sampling or accept-reject. We discuss the case of continuous random variables. There is a similar method for discrete random varibles. Suppose we want to simulate a random variable with density $f(x)$. We assume there is another density $g(x)$ and a constant c such that

$$f(x) \leq cg(x) \tag{4.8}$$

Note that by integrating this equation we see that c must be greater than 1. (It can equal 1 only if $f(x) = g(x)$.) We also assume it is relatively easy to generate samples with density g. The algorithm is as follows.

repeat

$Y \sim g$

$U \sim U(0,1)$

until $U \leq f(Y)/(cg(Y))$

$X = Y$

output X

The notation $Y \sim g$ means we generate a random variable with the density g. $U(0,1)$ denotes the uniform distribution on $[0,1]$, and $U \sim U(0,1)$ means we generate a random varible U with this distribution. Note that the test $U \leq f(Y)/(cg(Y))$ means that we accept Y with probability $f(Y)/(cg(Y))$. Otherwise we reject Y and try again.

Theorem 8 *With the reject-accept algorithm above, the probability density of X is given by $f(x)$.*

Review the partition theorem for continuous RV's

The acceptance-rejection algorithm can take multiple attempts to get a value of X that we accept. So we should worry about how long this takes. The number of tries is random; in fact the number of tries has a geometric distribution. Let p be the parameter for this geometric distribution, i.e., the probability we accept on a particular attempt. Given $Y = y$, the probability we accept is $f(y)/(cg(y))$. So

$$p = \int P(accept|Y = y)g(y)\,dy = \int \frac{f(y)}{cg(y)}g(y)\,dy = \frac{1}{c} \tag{4.9}$$

where "accept" is the event that we accept the first try. Note that the closer c is to 1, the closer the acceptance probability is to 100%. The mean of a geometric is $1/p$, so the mean number of tries is c.

Proof: We will compute the CDF of X and show that we get $P(X \leq t) = \int_{-\infty}^{t} f(y)dy$.

$$P(X \leq t) = P(\{X \leq t\} \cap A_1) + P(\{X \leq t\} \cap A_1^c) \tag{4.10}$$

where A_1 is the event that we accept the first try. For the second term we use

$$P(\{X \leq t\} \cap A_1^c) = P(X \leq t|A_1^c)P(A_1^c) \tag{4.11}$$

If we reject on the first try, then we just repeat the process. So $P(X \leq t|A_1^c) = P(X \leq t)$.

For the first term we use the continuous form of the partition theorem. Let Y_1 be the Y random variable on the first attempt. We condition on the value of Y_1:

$$P(\{X \leq t\} \cap A_1) = \int P(\{X \leq t\} \cap A_1|Y_1 = y)g(y)dy \tag{4.12}$$

Note that $P(\{X \leq t\} \cap A_1 | Y_1 = y) = P(\{Y \leq t\} \cap A_1 | Y_1 = y)$. This is zero if $y > t$ and $P(A_1|Y_1 = y)$ if $y \leq t$. Since $P(A_1|Y_1 = y) = f(y)/(cg(y))$, we get

$$P(\{X \leq t\} \cap A_1) = \int_{-\infty}^{t} \frac{f(y)}{cg(y)} g(y)dy = \frac{1}{c}\int_{-\infty}^{t} f(y)dy \tag{4.13}$$

So we have shown

$$P(X \leq t) = \frac{1}{c}\int_{-\infty}^{t} f(y)dy + (1 - \frac{1}{c})P(X \leq t) \tag{4.14}$$

Solving for $P(X \leq t)$ we find it equals $\int_{-\infty}^{t} f(y)dy$. **QED**

Example: Suppose we want to generate samples from the standard normal distribution. So

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2) \tag{4.15}$$

We use

$$g(x) = \frac{1}{\pi}\frac{1}{1+x^2} \tag{4.16}$$

the Cauchy distribution. Let

$$c = \sup_x \frac{f(x)}{g(x)} = \sup_x \sqrt{\frac{\pi}{2}} \exp(-x^2/2)(1+x^2) \tag{4.17}$$

It is obvious that c is finite. A little calculus shows the sup occurs at $x = \pm 1$. So we get $c = \sqrt{\frac{\pi}{2}} e^{-1/2} 2 \approx 1.52$. Note that it is easy to sample from the Cauchy distribution. This method will accept about 66 % of the time.

Exercise Consider the normal distribution again, but now we let $g(x) = \frac{1}{2}\exp(-|x|/2)$. Show that this works for the acceptance-rejection method and compute the acceptance probability.

There is a nice geometric interpretation of this method. First a fact from probability.

Theorem 9 *Let $f(x)$ be a density function for a continuous random variable. Let A be the region in the plane given by*

$$A = \{(x, y) : 0 \leq y \leq f(x)\} \tag{4.18}$$

So A is the area under the graph of $f(x)$. Let (X, Y) have the uniform distribution on A. Then the marginal distribution of X is $f(x)$.

Proof: Note that A has area 1. So the joint density function is just the indicator function of A. To get the marginal density of X at x, call it $f_X(x)$, we integrate out y.

$$f_X(x) = \int 1_A(x,y)dy = f(x) \tag{4.19}$$

QED

The theorem goes the other way: if we generate X using the density $f(x)$ and then pick Y uniformly from $[0, f(X)]$, then (X, Y) will be uniformly distributed on A. To see this note that we are making

$$f_{Y|X}(y|x) = \frac{1}{f(x)} 1_{0 \le y \le f(x)} \tag{4.20}$$

So the joint density will be

$$f_{X,Y}(x,y) = f_{Y|X}(y|x)f(x) = 1_{0 \le y \le f(x)} = 1_A \tag{4.21}$$

Now consider the acceptance-rejection method. Let A be area under the graph of $f(x)$ and let B be the area under the graph of $cg(x)$. By our assumption, A is contained in B. Note that B is the area under the graph of $g(x)$ stretched in the vertical direction by a factor of c. So if we sample Y and sample U from the uniform distribution on $[0, 1]$, then $(Y, Ucg(x))$ will be uniformly distributed on B. (Notation is confusing.) The process of accepting it only if $U \le f(Y)/(cg(Y))$ means that we accept the point in B only if it is in A

We have considered continuous random variables but the acceptance-rejection works for discrete random variables as well. If we want to simulate a discrete RV X with pdf $f_X(x)$, i.e., $f_X(x) = P(X = x)$ then we look for another discrete RV Y with pdf $f_Y(x)$ that we know how to simulate. If there is a constant c such that $f(x) \le cg(x)$, then we are in business. The algorithm is essentially identical to the continuous case, and the proof that it works is the same as the proof in the continuous case with integrals replaced by sums.

4.3 Alias method for discrete distribution

This is a method for generating a sample from a discrete distribution with a finite number of values. Let n be the number of values. The method takes a time $O(n)$ to set up, but once it is set up the time to generate a sample is $O(1)$. It is no loss of generality to take the values to be $1, 2, \cdots, n$. Let p_i be the probability of i.

Proposition 1 *Given $p_i > 0$ with $\sum_{i=1}^n p_i = 1$, we can find $q_i \in [0,1]$ for $i = 1, \cdots, n$ and functions $u(i), l(i) : \{1, 2, \cdots, n\} \to \{1, 2, \cdots, n\}$ such that for each value i, we have*

$$\frac{1}{n} \sum_{k:l(k)=i} q_k + \frac{1}{n} \sum_{k:u(k)=i} (1 - q_k) = p_i \tag{4.22}$$

DRAW A PICTURE.

The algorithm is then as follows. Pick k uniformly from $\{1, 2, \cdots\}$ and pick U uniformly from $[0, 1]$. If $U \leq q_k$ we return $l(k)$. If $U > q_k$ we return $u(k)$. The equation in the proposition insures that the probability we will return i is p_i.

Proof: It is convenient to let $P_i = np_i$. So the sum of the P_i is n and we want

$$\sum_{k:l(k)=i} q_k + \sum_{k:u(k)=i} (1 - q_k) = P_i \tag{4.23}$$

If the P_i all equal 1 the solution is trivial. Otherwise we can find k, m such that $P_k < 1$ and $P_m > 1$. Define $q_1 = P_k$, $l(1) = k$, $u(1) = m$. This takes care of all of the "probability" for value k. For value m it takes care of only some of the "probability", $1 - q_1$ to be precise. Now let $P'_1, \cdots, P'_{n-1}$ be $P_1, \cdots, P_n$ with P_k deleted and P_m replaced by
$P_m - (1 - q_1) = P_m + P_k - 1$. Note that the sum of the P'_i for $i = 1$ to $n - 1$ is $n - 1$. So we can apply induction to define $q_2, \cdots, q_n$ and define $l()$ and $u()$ on $2, 3, \cdots, n$. Note that $l(i)$ and $u(i)$ will not take on the value k for $i \geq 2$.

QED

Another method for fast simulation of a discrete RV is the method of guide tables. They are discussed in Owen's book.

Stop - Mon, 2/1

4.4 Tricks for specific distributions

For specific distributions there are sometimes clever tricks for simulating them. We give a few of them for distributions that are frequently used.

4.4.1 Box-Muller

Suppose we want to generate samples from the standard normal distribution. We can use inversion, but then we have the nontrivial problem of inverting the CDF. We can also use acceptance-rejection as we saw. Another method is called the Box-Muller method. It it based on the following observation. Let X, Y be independent, both with the standard normal distribution. So the joint density is

$$\frac{1}{2\pi}\exp(-(x^2+y^2)/2) \tag{4.24}$$

We change to polar coordinates, i.e., we define two new random variables Θ and R by

$$X = R\cos(\Theta) \tag{4.25}$$
$$Y = R\sin(\Theta) \tag{4.26}$$

Then the joint density of R, Θ is $\exp(-r^2/2)r/(2\pi)$. This shows that R and Θ are independent. Θ is uniform on $[0, 2\pi]$ and R has density $r\exp(-r^2/2)$.

We can go the other way; this is the Box Mueller algorithm. Generate Θ and R with these densities. It is trivial to generate Θ. For R we can use inversion since the density function is explicitly integrable. Define X and Y as above and they will be independent, each with a standard normal distribution.

4.4.2 Geometric

We can use inversion to simulate the geometric distribution. If the mean is large this can be slow. Here is another method. Recall that N has the geometric distribution with parameter $p \in (0,1)$ if $P(N=k) = (1-p)^{k-1}p$. Now let X have an exponential distribution with parameter λ. So the density is

$$f(x) = \exp(-\lambda x)\lambda \tag{4.27}$$

So

$$P(k-1 \le X < k) = \int_{k-1}^{k} \exp(-\lambda x)\lambda\, dx = \exp(-(k-1)\lambda) - \exp(-k\lambda) \tag{4.28}$$
$$= \exp(-(k-1)\lambda)[1-\exp(-\lambda)] \tag{4.29}$$

So if we take $p = 1-\exp(-\lambda)$, then X round down to the nearest integer has the geometric distribution. In other words $N = \lfloor X \rfloor$ has a geometric distribution.

4.4.3 Random permutations

We are interested in the uniform distribution on the set of permutations on $\{1, 2, \cdots, n\}$. If n is fairly small we treat this as a discrete random variable with a finite set of values and use techniques we have discussed. But the number of factorials is $n!$ for this will be impractical for even modest values of n. Here is an algorithm that generates a sample in time $O(n)$. The idea is simple. First randomly pick $\pi(i)$. It is just uniform on $\{1, 2, \cdots, n\}$. Now pick $\pi(2)$. It will be uniform on $\{1, 2, \cdots, n\}$ with $\pi(1)$ removed. Then $\pi(3)$ will be uniform on $\{1, 2, \cdots, n\}$ with $\pi(1)$ and $\pi(2)$ removed. The slightly tricky part is keeping track of the integers which are not yet in the range of π. In the following $(k_1, \cdots, k_n)$ will be the integers which are not yet in the range.

Initialize $(k_1, \cdots, k_n) = (1, 2, \cdots n)$
For $i = 1$ to n
    Generate I uniformly from $\{1, 2, \cdots, n - i + 1\}$
    Set $X_i = k_I$
    Set $k_I = k_{n-i+1}$
Return $(X_1, \cdots, X_n)$

4.4.4 RV's that are sums of simpler RV's

Suppose we want to simulate a binomial RV X with n trials. If X_i takes on the values $0, 1$ with probabilities $1 - p, p$ and $X_1, \cdots, X_n$ are independent, then $\sum_{i=1}^{n} X_i$ has a binomial distribution. The X_i are trivial to simulate. Of course if n is large this will not be a fast method.

Similarly the χ^2 distribution with k degrees of freedom is given by

$$\sum_{i=1}^{k} Z_i^2 \tag{4.30}$$

where the Z_i are independent standard normals.

4.4.5 Mixtures

A mixture means that we have random variables $X_1, X_2, \cdots, X_n$ and probabilities $p_1, p_2, \cdots, p_n$. Let I be a random variable with $P(I = i) = p_i$, independent of the X_i. Then we let $X = X_I$. The CDF of X is

$$F_X(x) = \sum_{i=1}^{n} p_i F_{X_i}(x) \tag{4.31}$$

If the X_i are all absolutely continuous with densities $f_{X_i}(x)$ then X is absolutely continuous with densities

$$f_X(x) = \sum_{i=1}^{n} p_i f_{X_i}(x) \tag{4.32}$$

If we can sample the X_i then it should be obvious how we sample X.

4.5 Generating multidimensional random variables

We now consider the case of generating samples of a random vector $(X_1, \cdots, X_d)$. If the random vector $(X_1, \cdots, X_d)$ is discrete with a finite set of values, then sampling from it is no different than sampling from a random variable X that only takes on a finite set of values. So we can use the techniques of the previous sections. Hence we will focus on continuous RV's in this section.

If the components X_i are independent, then this reduces the problem we have considered in the previous sections. This section considers what to do when the components are dependent.

4.5.1 Composition method or conditioning

For simplicity of notation we consider the case of $d = 2$. We want to generate jointly continuous random variables with density $f(x, y)$. Compute $f_X(x)$ by integrating out y. Define $f_{Y|X}(y|x)$ in the usual way. Then we generate X according to $f_X(x)$ and then generate Y according to $f_{Y|X}(y|x)$. Then (X, Y) will have the desired joint distribution. Several things can do wrong. Computing $f_X(x)$ requires doing an integral which may not be explicitly computable. And we need to be able to generate samples from $f_X(x)$ and $f_{Y|X}(y|x)$.

4.5.2 Acceptance-rejection

The acceptance-rejection method generalizes immediately to random vectors. We consider the case of a jointly continuous random vector. Suppose we want to sample from the density $f(x_1, \cdots, x_d)$. If $g(x_1, \cdots, x_d)$ is a density that we can sample from and there is a constant c such that

$$f(x_1, \cdots, x_d) \leq c g(x_1, \cdots, x_d) \tag{4.33}$$

then we generate a random vector $\vec{Y} = (Y_1, \cdots, Y_n)$ according to g and we generate a uniform RV U on $[0, 1]$. We accept $\vec{Y}$ if $U \leq f(Y_1, \cdots, Y_d)/(cg(Y_1, \cdots, Y_d))$. If we reject we repeat.

This is useful only if we can sample from g. If g is the density of independent random variables this may be doable. Note that g corresponds to independent components if (and only if) $g(x_1, \cdots, x_d)$ factors into a product of functions $g_i(x_i)$. So it is natural to seek an upper bound on $f(x_1, \cdots, x_d)$ of this form.

If we want to generate the uniform distribution on some subset S of R^d and we can enclose S in a hypercube, then we can do this with acceptance rejection. The acceptance ratio will be the volume of S divided by the volume of the hypercube. If this is unacceptably small we can try to do better by enclosing S in geometry that "fits" better but for which we can still uniformly sample from the geometry.

4.5.3 The multivariate normal

The random vector $(X_1, \cdots, X_d)$ has a multivariate normal distribution with means $(\mu_1, \cdots, \mu_d)$ and covariance matrix Σ if its density is

$$f(x_1, \cdots, x_d) = c \exp(-\frac{1}{2}((\vec{x} - \vec{\mu}), \Sigma^{-1}(\vec{x} - \vec{\mu})) \tag{4.34}$$

where c is a normalizing constant. This distribution is easily simulated using the following.

Proposition 2 *Let $Z_1, \cdots, Z_d$ be independent standard normal RV's. Let*

$$\vec{X} = \Sigma^{1/2}\vec{Z} + \vec{\mu} \tag{4.35}$$

Then $\vec{X}$ has the multivariate normal distribution above and $\Sigma^{1/2}$ is the matrix square root of Σ.

Idea of proof: Diagonalize Σ and do a change of variables.

4.5.4 Affine transformations

Let $\vec{X} = (X_1, \cdots, X_d)$ be an absolutely continuous random vector with density $f_{\vec{X}}(x_1, \cdots, x_d)$. Let $\vec{Z} = A\vec{X} + \vec{b}$ where A is a d by d matrix. Then $\vec{Z} = (Z_1, \cdots, Z_d)$ is an absolutely continuous random vector with density

$$f_{\vec{Z}}(z_1, \cdots, z_d) = \frac{1}{|det(A)|} f_{\vec{X}}(A^{-1}(\vec{z} - \vec{b})) \tag{4.36}$$

4.5.5 Uniform point in a sphere, on a sphere

Suppose we want to generate a point in R^d that is uniformly distributed in the ball of radius 1. We can do this by acceptance rejection. We generate a point uniformly on the hypercube $[-1,1]^d$ and accept it if it is inside the ball. The acceptance ratio depends on the dimension d and goes to zero as d goes to infinity. So for large d we would like a better method. We would also like to be able to generate points uniformly on the sphere, i.e., the boundary of the ball. (By ball I mean $\{(x_1, \cdots, x_d) : \sum_i x_i^2 \leq 1\}$ and by sphere I mean $\{(x_1, \cdots, x_d) : \sum_i x_i^2 = 1\}$.

Note that generating a point uniformly inside the ball and generating a point uniformly on the sphere are closely related. We can see the relation by thinking of hyper-spherical coordinates. If $\vec{X}$ is uniformly distributed inside the ball, then $\vec{X}/||\vec{X}||$ will be uniformly distributed on the sphere. Conversely, if $\vec{X}$ is uniformly distributed on the sphere and R is an independent, continuous RV with density r^{n-1}/n on $[0,1]$, then $R\vec{X}$ will be uniformly distributed inside the ball.

So we only need a method to generate the uniform distribution on the sphere. Let $Z_1, Z_2, \cdots, Z_d$ be independent, standard normal random variables. Then their joint density is $c \exp(-\frac{1}{2}\sum_{i=1}^d z_i^2)$. Note that it is rotationally invariant. So $\vec{Z}/||\vec{Z}||$ will be uniformly distributed on the sphere.

Chapter 5

Variance reduction

The error in a direct Monte Carlo simulation goes as $\sigma/\sqrt{n}$. So there are two ways we can reduce the error. Run the simulation for a longer time, i.e., increase n or find a different formulation of the Monte Carlo that has a smaller σ. Methods that do the latter are know as variance reduction.

5.1 Antithetic variables

If X and Y are independent, then $var(X+Y) = var(X) + var(Y)$. If they are not independent the covariance enters. Letting μ_X, μ_Y denote the means of X and Y, we have

$$\begin{aligned} var(X+Y) &= E[(X+Y)^2] - (\mu_X + \mu_Y)^2 && (5.1)\\ &= E[X^2] - \mu_X^2 + E[Y^2] - \mu_Y^2 + 2(E[XY] - \mu_X\mu_Y) && (5.2)\\ &= var(X) + var(Y) + 2cov(X,Y) && (5.3) \end{aligned}$$

The covariance is $cov(X,Y) = \rho_{X,Y}\sigma_X\sigma_Y$ and ρ lies between -1 and 1. If ρ is negative the variance of $X+Y$ is smaller than the sum of their variances. Antithetic variables take advantage of this fact.

Definition 1 *Random variables X, Y on the same probability space are antithetic if they have the same distribution and their covariance is negative.*

Suppose we want to compute $\mu = E[X]$ and we can find another random variable Y such that X, Y is an antithetic pair. So $E[Y]$ is also equal to μ. Our Monte Carlo algorithm is as

follows. We generate n indendent samples $\omega_1, \cdots, \omega_n$ from the probability space and let $X_i = X(\omega_i)$ and $Y_i = Y(\omega_i)$. Our estimator for μ is then

$$\hat{\mu}_n = \frac{1}{2n} \sum_{i=1}^{n} (X_i + Y_i) \tag{5.4}$$

Obvious the mean of $\hat{\mu}_n$ is μ, so this is an unbiased estimator. Let ρ denote the correlation of X and Y. Since they have the same distribution, they have the same variance. We denote it by σ^2. So the variance of our estimator is

$$\begin{aligned} var(\hat{\mu}_n) &= \frac{1}{(2n)^2} n \; var(X_1 + Y_2) && (5.5) \\ &= \frac{1}{2n} \sigma^2 (1 + \rho) && (5.6) \end{aligned}$$

To compare this with direct Monte Carlo just using X we have to pay attention to the times involved. Recall that the relevant quantity for the quality of our MC is $\sigma^2 \tau$, where τ is the time to generate a sample.

For direct MC with just X we have to generate an ω and then evaluate X on it. For our antithetic MC we have to generate an ω and then evaluate both X and Y on it. Let τ_ω be the time required to generate an ω. We assume it takes the same time to evalute Y that it does to evaluate Y. Call that time τ_e. (e for evaluation.) Then the original MC takes time $\tau_\omega + \tau_e$, while the antithetic MC takes time $\tau_\omega + 2\tau_e$. So we need to compare $\sigma^2(\tau_\omega + \tau_e)$ for the original MC with $\sigma^2 \frac{1}{2}(1 + \rho)(\tau_\omega + 2\tau_e)$ So the antithetic is better if

$$\frac{1}{2}(1 + \rho)(\tau_\omega + 2\tau_e) < \tau_\omega + \tau_e \tag{5.7}$$

If τ_ω is neglible compared to τ_e then this simplifies to $\rho < 0$. On the other hand, if τ_e is neglible compared to τ_ω then this simplifies to $\rho < 1$ which is always true unless $Y = X$. Note that the advantage of the antithetic MC will be large only if ρ is close to -1.

If we want to find a confindence interval for our estimate, we need the variance of the antithetic estimator. We could use the calculations above. But this requires estimating ρ. We can avoid this by the following approach. Let $Z = (X + Y)/2$, and let $Z_i = (X_i + Y_i)/2$. We can think of our antithetic Monte Carlo as just generating n samples of Z. Then we compute the sample variance of the sample $Z_1, \cdots, Z_n$ and just do a straightforward confidence interval.

Of course this is only useful if we can find antithetic pairs. We start with a trivial example. We want to compute

$$\mu = \int_0^1 f(x) dx = E[f(U)] \tag{5.8}$$

where U is a uniform random variable on $[0,1]$. So we are computing the mean of $X = f(U)$. Suppose f is an increasing function. Then it might be helpful to balance a value of U in $[0,1/2]$ with its "reflection" $1-U$. So take $Y = F(1-U)$. This has the same distribution (and hence the same mean) as X since $1-U$ is uniform on $[0,1]$. A fancy way to say this is that the uniform probability measure on $[0,1]$ is invariant under the map $1 \to 1-x$ on $[0,1]$.

So we consider the following general set-up. We assume there is a map $R : \Omega \to \Omega$ under which the probability measure is invariant, i.e., $P = P \circ R$. We define $Y(\omega) = X(R\omega)$. Then Y has the same distribution as X and hence the same mean. We need to study the correlation of X and Y to see if this will be useful. Define

$$X_e(\omega) = \frac{1}{2}[X(\omega) + X(R\omega)], \tag{5.9}$$

$$X_o(\omega) = \frac{1}{2}[X(\omega) - X(R\omega)] \tag{5.10}$$

Then $X = X_o + X_e$, and we can think of this as a decomposition of X into its even and odd parts with respect to R. (Note $X_e(R\omega) = X_e(\omega)$, $X_o(R\omega) = -X_o(\omega)$.) The invariance of P under R implies that

$$E[X_e] = \mu, \quad E[X_o] = 0, \quad E[X_e X_o] = 0 \tag{5.11}$$

Thus X_e and X_o are uncorrelated. This is weaker than being independent, but it does imply the variance of their sum is the sum of their variances. So if we let σ_e^2 and σ_o^2 be the variances of X_e and X_o, then $\sigma^2 = \sigma_e^2 + \sigma_o^2$, where σ is the variance of X. Note that the variance of Y is also equal to σ^2. A little calculation shows that ρ, the correlation between X and Y, is given by

$$\rho = \frac{\sigma_e^2 - \sigma_o^2}{\sigma_e^2 + \sigma_o^2} \tag{5.12}$$

Thus Y will be a good antithetic variable if σ_e is small compared to σ_o. This will happen if X is close to being an odd function with respect to R.

Literature seems to say that if $X = f(U)$ where U is a vector of i.i.d. uniform on $[0,1]$ and f is increasing then $Y = f(1-U)$ is a good antithetic RV where in $1-U$, 1 means the vector with 1's in all the components.

Stop - Wed, 2/3

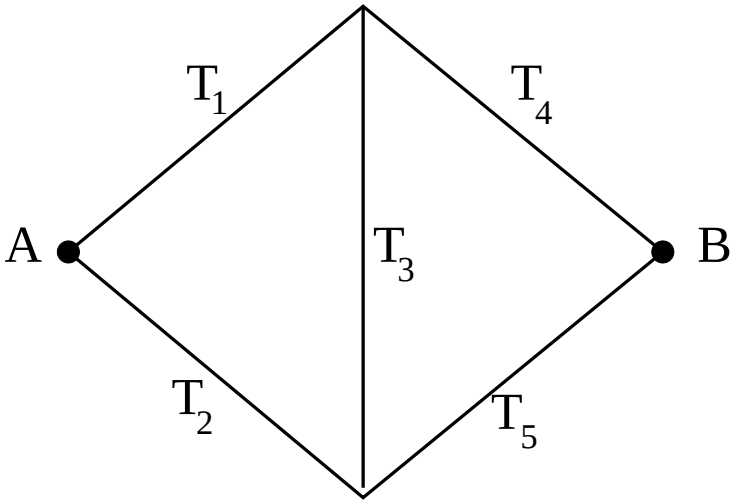

Figure 5.1: Network example. We seek the quickest path from A to B.

Network example from Kroese The times T_i are independent and uniformly distributed but with different ranges:
T_1 uniform on $[0, 1]$
T_2 uniform on $[0, 2]$
T_3 uniform on $[0, 3]$
T_4 uniform on $[0, 1]$
T_5 uniform on $[0, 2]$
The network is small enough that you can find the mean time of the quickest path analytically. It is

$$\mu = \frac{1339}{1440} \approx 0.92986 \tag{5.13}$$

Let U_1, U_2, U_3, U_4, U_5 be independent, uniform on $[0, 1]$. Then we can let $T_1 = U_1, T_2 = 2 * U_2$, etc. And the quickest time can be written as a function $X = h(U_1, U_2, U_3, U_4, U_5)$. We let $Y = h(1 - U_1, 1 - U_2, 1 - U_3, 1 - U_4, 1 - U_5)$, and then $Z = (X + Y)/2$. Our antithetic estimator is

$$\frac{1}{n}\sum_{i=1}^{n} Z_i = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(X_i + Y_i) \tag{5.14}$$

where X_i and Y_i use the same U vector. Some simulation shows that without using the antithetic pair, the variance of X is approximately 0.158. Another simulation using the antithetic pair shows that the variance of Z is approximately 0.0183. The error is proportional to the square root of the variance, so the error is reduced by a factor of $\sqrt{0.158/0.0183}$. But we must keep in mind that it takes twice as long to generate a sample of

Z as it does a sample of X. So for a fixed amount of CPU time we will have half as many samples of Z which means a factor of $\sqrt{2}$ for the error. So the true reduction in the error is a factor of $\sqrt{0.158/(2 * 0.0183)}$ which is just over 2. But keep in mind that to reduce the error by a factor of 2 by increasing the number of samples would require generating 4 times as many samples. So if we think in terms of how long it takes to reach a given error level, then the antithetic method has reduced the computation time by a factor of 4.

5.2 Control variates

Consider the bridge example again. The times T_1 and T_4 are both uniformly distributed on $[0, 1]$ while the other three times are uniformly distributed on larger intervals. So we expect that the quickest path will be the path through bonds 1 and 4 with fairly high probability. In other words, if we let X be the minimum time for the full network and let $Y = T_1 + T_4$, then Y will be equal to X with high probability. Note that we know the mean of Y. Can we take advantage of this to improve our Monte Carlo method? Let ν be the known mean of Y. (Of course, $\nu = 1/2 + 1/2 = 1$.) Then $\mu = E[X - (Y - \nu)]$. So we can do a Monte Carlo simulation of $X - (Y - \nu)$ rather than X. The hope is that $X - (Y - \nu)$ has a smaller variance since it equals ν most of the time.

The general setup is as follows. We want to compute $\mu = E[X]$. We have another random variable Y on the same probability space and we know its mean. Call it $\nu = E[Y]$. (Note that μ is unknown, ν is known.) We generate a random sample $\omega_1, \omega_2, \cdots, \omega_n$ and evaluate X and Y on them. So let $X_i = X(\omega_i)$ and $Y_i = Y(\omega_i)$. Now define

$$\hat{l}_n = \frac{1}{n} \sum_{i=1}^{n} [X_i - \alpha(Y_i - \nu)] \tag{5.15}$$

where α is a parameter. In our discussion of the brigde example we took $\alpha = 1$, but now we allow a more general choice of the new estimator. Note that $E[\hat{l}_n] = \mu$, i.e., for any choice of α this is an unbiased estimator of μ.

Let ρ denote the correlation of X and Y. Let σ_X^2 and σ_Y^2 be the variances of X and Y. The variance of our estimator is

$$var(\hat{l}_n) = \frac{1}{n} var(X - \alpha Y) \tag{5.16}$$

We have

$$var(X - \alpha Y) = \sigma_X^2 + \alpha^2 \sigma_Y^2 - 2\alpha\rho\sigma_X\sigma_Y \tag{5.17}$$

We can use any α we want, so we choose α to minimize this variance. The minimizing α is given by

$$\alpha_0 = \frac{\rho\sigma_X}{\sigma_Y} = \frac{cov(X,Y)}{\sigma_Y^2} \tag{5.18}$$

in which case

$$var(X - \alpha_0 Y) = \sigma_X^2(1-\rho^2) \tag{5.19}$$

So the variance of our estimator is $\sigma_X^2(1-\rho^2)/n$. So we have reduced the variance by a factor of $1-\rho^2$. So the method works well if ρ is close to 1 or -1.

Note that to compute the optimal α we need to know σ_X, σ_Y, and ρ. We might know σ_Y, but we almost certainly will not know σ_X or ρ. So we have to use our sample to estimate them. We estimate σ_X^2 (and σ_Y^2 if needed) with the usual sample variance. And we estimate ρ with

$$\hat{\rho}_n = \frac{1}{n}\sum_{i=1}^{n}[X_iY_i - \overline{X}_n\nu] \tag{5.20}$$

where $\overline{X}_n$ is the samples mean of X.

In the above we have assumed that the mean of Y is known exactly. Even if we do not know it exactly, but just have a good approximation we can still use the above. If it is much faster to compute the control RV Y than the original RV X, then we could use a preliminary Monte Carlo to compute a good approximation to the mean of Y and then do the above Monte Carlo to get the mean of X.

We look at the last paragraph in more detail and think of what we are doing as following. We want to compute $E[X]$. We have another random variable Y such that we think the varince of $X-Y$ is small. We write X as $(X-Y)+Y$ and try to compute the mean of X by computing the means of $X-Y$ and Y separately. Let σ_x^2, σ_Y^2 be the variances of X and Y. Let σ_{X-Y} be the variance of $X-Y$, which hopefully is small. Let τ_X and τ_Y be the time it take to generate samples of X and Y. We assume we have a fixed amount T of CPU time. If we do ordinary MC to compute $E[X]$, then we can compute T/τ_X samples and the square of the error will be $\sigma_X^2\tau_X/T$.

Now suppose we do two independent Monte Carlo simulations to compute the means of $X-Y$ and Y. For the $X-Y$ simulation we generate n_1 samples and for the Y simulation we generate n_2 samples. These numbers are constrained by $n_1(\tau_X+\tau_Y)+n_2\tau_Y = T$. We assume that τ_Y is much smaller than τ_X and replace this constraint by $n_1\tau_X + n_2\tau_Y = T$. Since our two Monte Carlos are independent, the square of the error is the sum of the squares of the errors of the two Monte Carlos, i.e.,

$$\frac{\sigma_{X-Y}^2}{n_1} + \frac{\sigma_Y^2}{n_2} \tag{5.21}$$

Now we minimize this as a function of n_1 and n_2 subject to the constraint. (Use Lagrange multiplier or just use the constraint to solve for n_2 in terms of n_1 and turn it into a one variable minimization problem.) You find that the optimal choice of n_1, n_2 is

$$n_1 = \frac{T\sigma_{X-Y}}{\sqrt{\tau_X}(\sigma_{X-Y}\sqrt{\tau_X} + \sigma_Y\sqrt{\tau_Y})}, \tag{5.22}$$

$$n_2 = \frac{T\sigma_Y}{\sqrt{\tau_Y}(\sigma_{X-Y}\sqrt{\tau_X} + \sigma_Y\sqrt{\tau_Y})} \tag{5.23}$$

which gives a squared error of

$$\frac{1}{T}(\sigma_{X-Y}\sqrt{\tau_X} + \sigma_Y\sqrt{\tau_Y})^2 \tag{5.24}$$

If σ_{X-Y} is small compared to σ_X and σ_Y and τ_Y is small compared to τ_X, then we see this is a big improvement over ordinary MC.

Network example We return to our network example. For the control variable we use $Y = T_1 + T_4$. So we do a Monte Carlo simulation of $Z = X + (Y - \nu)$ where $\nu = E[Y] = 1$. We find that the variance of Z is approximately 0.0413. As noted earlier the variance of X is approximately 0.158. So the control variate approach reduced the variance by a factor of approximately 3.8. This corresponds to a reduction in the error by a factor of $\sqrt{3.8}$. If we want a fixed error level then the use of a control variate reduces the computation time by a factor of 3.8.

Note that you do not have to know what α you want to use before you do the MC run. You can compute the sample means and sample variances for X and Y separately as well as an estimator for ρ. Then at the end of the run you can use the sample variances and estimator for ρ to compute an estimator for the best α. Note, however, that if you do this your α now depends on all the samples and so the samples $X_i - \alpha Y_i$ are not independent. So the usual method of deriving a confidence interval is not legit. If you really want to worry about this see section 4.2 of Fishman's *Monte Carlo: Concepts, Algorithms, and Applications*. I would be surprised if it matters unless n is small.

It is possible to use more than one control variable. Let $\vec{Y} = (Y^1, \cdots, Y^d)$ be vector of random variables. We assume we know their means $\nu^i = E[Y^i]$. Then our estimator is

$$\hat{l}_n = \frac{1}{n}\sum_{i=1}^{n}[X_i - (\vec{\alpha}, \vec{Y}_i - \vec{\nu})] \tag{5.25}$$

where $\vec{\alpha}$ is a vector of parameters and $(\vec{\alpha}, \vec{Y} - \vec{\nu})$ denotes the inner product of that vector and $\vec{Y} - \vec{\nu}$. To find the optimal α we need to minimize the variance of $X - (\vec{\alpha}, Y)$.

$$var(X - (\vec{\alpha}, Y)) = cov(X - (\vec{\alpha}, Y), X - (\vec{\alpha}, Y)) \tag{5.26}$$

$$= var(X, X) - 2\sum_{i=1}^{2}\alpha_i cov(X, Y_i) + \sum_{i,j=1}^{2}\alpha_i\alpha_j cov(Y_i, Y_j) \tag{5.27}$$

Let Σ be the matrix with entries $cov(Y_i, Y_j)$, i.e., the covariance matrix of the control variates. Let $\vec{C}$ be the vector of covariances of X and the Y_i. Then the above is

$$var(X,X) - 2\sum_{i=1}^{2} \alpha_i C_i + \sum_{i,j=1}^{2} \alpha_i \alpha_j \Sigma_{i,j} = var(X,X) - 2(\vec{\alpha}, \vec{C}) + (\vec{\alpha}, \Sigma\vec{\alpha}) \tag{5.28}$$

Optimal α is

$$\vec{\alpha}_0 = \Sigma^{-1}\vec{C} \tag{5.29}$$

5.3 Stratified sampling

To motivate stratified sampling consider the following simple example. We want to compute

$$I = \int_0^1 f(x)\, dx \tag{5.30}$$

On the interval $[0, 1/2]$ the function $f(x)$ is nearly constant, but on the interval $[1/2, 1]$ the function varies significantly. Suppose we wanted to use Monte Carlo to compute the two integrals

$$\int_0^{1/2} f(x)\, dx, \quad \int_{1/2}^1 f(x)\, dx \tag{5.31}$$

The Monte Carlo for the first integral will have a much smaller variance than the Monte Carlo for the second integral. So it would make more sense to spend more time on the second integral, i.e., generate more samples for the second integral. However, under the usual Monte Carlo we would randomly sample from $[0, 1]$ and so would get approximately the same number of X_i in $[0, 1/2]$ and in $[1/2, 1]$. The idea of stratified sampling is to divide the probability space into several regions and do a Monte Carlo for each region.

We now turn to the general setting. As always we let Ω be the probability space, the set of possible outcomes. We partition it into a finite number of subsets Ω_j, $j = 1, 2, \cdots, J$. So

$$\Omega = \cup_{j=1}^{J} \Omega_j, \quad \Omega_j \cap \Omega_k = \emptyset \; if \; j \neq k \tag{5.32}$$

We let $p_j = P(\Omega_j)$ and let P_j be $P(\cdot|\Omega_j)$, the probability measure P conditioned on Ω_j. We assume that the probabilities p_j are known and that we can generate samples from the conditional probability measures $P(\cdot|\Omega_j)$. The sets Ω_j are called the strata. Note that the partition theorem says that

$$P(\cdot) = \sum_{j=1}^{J} p_j P(\cdots|\Omega_j), \tag{5.33}$$

$$E[X] = \sum_{j=1}^{J} p_j E[X|\Omega_j] \tag{5.34}$$

We are trying to compute $\mu = E[X]$. We will generate samples in each strata, and the number of samples from each strata need not be the same. So let n_j, $j = 1, 2, \cdots, J$ be the number of samples from strata j. Let $X_i^j, i = 1, 2, \cdots, n_j$ be the samples from the jth strata. Then our estimator for μ is

$$\hat{\mu} = \sum_{j=1}^{J} \frac{p_j}{n_j} \sum_{i=1}^{n_j} X_i^j \tag{5.35}$$

Note that the expected value of X_i^j is $E[X|\Omega_j]$. So the mean of $\hat{\mu}$ is

$$E[\hat{\mu}] = \sum_{j=1}^{J} p_j E[X|\Omega_j] = E[X] \tag{5.36}$$

where the last equality follows from the partition theorem.

Let

$$\mu_j = E[X|\Omega_j], \quad \sigma_j^2 = E[X^2|\Omega_j] - \mu_j^2 \tag{5.37}$$

The quantity σ_j^2 is often denoted $var(X|\Omega_j)$. It is the variance of X if we use $P(\cdot|\Omega_j)$ as the probability measure instead of $P(\cdot)$.

We write out our estimator as

$$\hat{\mu} = \sum_{j=1}^{J} p_j \hat{\mu}_j, \quad \hat{\mu}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} X_i^j \tag{5.38}$$

We assume that we generate sample from different strata in an independent fashion. So the random variables $\hat{\mu}_1, \hat{\mu}_2, \cdots, \hat{\mu}_J$ are independent. The variance of $\hat{\mu}_j$ is σ_j^2. So we have

$$var(\hat{\mu}) = \sum_{j=1}^{J} p_j^2 \frac{\sigma_j^2}{n_j} \tag{5.39}$$

Stop - Mon, 2/8

How do we choose the n_j? One possible choice is *proportional allocation*. Letting N denote the total number of samples we will generate, we take $n_j = p_j N$. (Of course, we have to round this to the nearest integer.) This gives

$$var(\hat{\mu}) = \frac{1}{N} \sum_{j=1}^{J} p_j \sigma_j^2 \tag{5.40}$$

To compare this with the variance of ordinary MC, we do a little computation.

$$\begin{aligned}
\sum_{j=1}^{J} p_j \sigma_j^2 &= \sum_{j=1}^{J} p_j E[(X-\mu_j)^2|\Omega_j] && (5.41)\\
&= \sum_{j=1}^{J} p_j E[((X-\mu)+(\mu-\mu_j))^2|\Omega_j] && (5.42)\\
&= \sum_{j=1}^{J} p_j \left[E[((X-\mu)^2|\Omega_j] + (\mu-\mu_j))^2 + 2(\mu-\mu_j)E[X-\mu|\Omega_j\right] && (5.43)
\end{aligned}$$

Note that $\sigma^2 = \sum_j p_j E[(X-\mu)^2|\Omega_j]$, and $E[X-\mu|\Omega_j] = \mu_j - \mu$. So the above reduces to

$$\sum_{j=1}^{J} p_j \sigma_j^2 = \sigma^2 - \sum_{j=1}^{J} p_j (\mu-\mu_j)^2 \tag{5.44}$$

So using proportional allocation the variance of the Monte Carlo using strata is smaller than the variance of plain Monte Carlo unless the μ_j are all equal to μ We also see that we should try to choose the strata so that the means within the strata are far from the overall mean.

But is proportional allocation optimal? Recall our motivating example. We probably should sample more in the Ω_j with higher variance. We can find the optimal choice of n_j. We want to minimize $var(\hat{\mu})$ subject to the constraint that the total number of samples is fixed, i.e.,

$$\sum_{j=1}^{J} n_j = N \tag{5.45}$$

This is a straightforward Lagrange multiplier problem. Let

$$\begin{aligned}
f(n_1, n_2, \cdots, n_J) &= \sum_{j=1}^{J} p_j^2 \frac{\sigma_j^2}{n_j}, && (5.46)\\
g(n_1, n_2, \cdots, n_J) &= \sum_{j=1}^{J} n_j && (5.47)
\end{aligned}$$

We want to minimize f subject to $g = N$. The minimizer will satify

$$\nabla f(n_1, n_2, \cdots, n_J) = -\lambda \nabla g(n_1, n_2, \cdots, n_J) \tag{5.48}$$

for some λ. So for $j = 1, 2, \cdots, J$

$$-p_j^2 \frac{\sigma_j^2}{n_j^2} = -\lambda \tag{5.49}$$

Solving for n_j,

$$n_j = \frac{1}{\sqrt{\lambda}} p_j \sigma_j \tag{5.50}$$

Thus

$$n_j = c p_j \sigma_j \tag{5.51}$$

where the constant c is chosen to make the sum of the n_j sum to N. So $c = N/\sum_j p_j \sigma_j$. This choice of n_j makes the variance of the estimate

$$var(\hat{\mu}) = \frac{1}{N} [\sum_{j=1}^{J} p_j \sigma_j]^2 \tag{5.52}$$

Note that the Cauchy Schwarz inequality implies this is less than or equal to the variance we get using proportional allocation and it is equal only if the σ_j are all the same. Of course, to implement this optimal choice we need to know all the σ_j whereas the proportional allocation does not depend on the σ_j.

Suppose the time to generate a sample depends on the strata. Let τ_j be the time to generate a sample in the jth strata. Then if we fixed the amount of computation time to be T, we have $\sum_j n_j \tau_j = T$. Then using a Lagrange multiplier to find the optimal n_j we find that n_j should be proportional to $p_j \sigma_j / \sqrt{\tau_j}$.

Example: rainfall example from Owen.

Example: Network example. Partition each uniform time into 4 intervals, so we get 4^5 strata.

5.4 Conditioning

To motivate this method we first review a bit of probability. Let X be a random variable, $\vec{Z}$ a random vector. We assume X has finite variance. We let $E[X|\vec{Z}]$ denote the conditional expectation of X where the conditioning is on the σ-field generated by $\vec{Z}$. Note that $E[X|\vec{Z}]$ is a random variable. We assume denote its variance by $var(E[X|\vec{Z}])$.

We define the conditional variance of X to be

$$var[X|\vec{Z}] = E[X^2|\vec{Z}] - (E[X|\vec{Z}])^2 \tag{5.53}$$

Note that $var[X|\vec{Z}]$ is a random variable. There is a conditional Cauchy Schwarz inequality which says that this random variable is always non-negative. A simple calculation gives

$$var(X) = E[var[X|\vec{Z}]] + var(E[X|\vec{Z}]) \tag{5.54}$$

In particular this shows that $E[X|\vec{Z}]$ has smaller variance than X.

The conditional expectation $E[X|\vec{Z}]$ is a function of $\vec{Z}$. There is a Borel-measurable function $h : R^d \to R$ such that $X = h(\vec{Z})$. Now suppose that it is possible to explicitly compute $E[X|\vec{Z}]$, i.e., we can explicitly find the function h. Suppose also that we can generate samples of $\vec{Z}$. One of the properties of conditional expectation is that the expected value of the conditional expectation is just the expected value of X. So we have

$$\mu = E[X] = E[E[X|\vec{Z}]] = E[h(\vec{Z})] \tag{5.55}$$

So we have the following Monte Carlo algorithm. Generate samples $\vec{Z}_1, \cdots, \vec{Z}_n$ of $\vec{Z}$. Compute $h(\vec{Z}_1), \cdots, h(\vec{Z}_n)$. Then the estimator is

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} h(\vec{Z}_i) \tag{5.56}$$

Of course the non-trivial thing is to find a random vector $\vec{Z}$ for which we can explicitly compute $E[X|\vec{Z}]$.

Example: Let $X_1, \cdots, X_d$ be independent random variables with exponential distributions and $E[X_i] = 1/\lambda_i$. We want to compute the probability that the largest of the d random variables is X_1, i.e., we want to compute

$$\mu = P(X_i < X_1, i = 2, \cdots, d) \tag{5.57}$$

We are particularily interested in the case that λ_1 is large compared to the other λ_i. In this case X_1 is usually small compared to the other X_i, so so the the probability μ will be tiny and very hard to compute accurately with an ordinary MC.

Suppose we condition on X_1. Then keeping in mind that the X_i are independent, we have

$$\begin{aligned} P(X_i < X_1, i = 2, \cdots, d|X_1 = x_1) &= P(X_i < x_1, i = 2, \cdots, d|X_1 = x_1) && (5.58)\\ &= P(X_i < x_1, i = 2, \cdots, d) && (5.59)\\ &= \prod_{i=2}^{d} P(X_i < x_1) && (5.60) \end{aligned}$$

Note that if X has an exponential distribution with mean λ, then $P(X < x) = 1 - e^{-\lambda x}$. Define $F(x) = 1 - e^{-x}$. So $P(X < x) = F(\lambda x)$. Then the above becomes

$$P(X_i < X_1, i = 2, \cdots, d | X_1 = x_1) = \prod_{i=2}^{d} F(\lambda_i x_1) \tag{5.61}$$

Now the probability we want is

$$\mu = P(X_i < X_1, i = 2, \cdots, d) = E[P(X_i < X_1, i = 2, \cdots, d | X_1)] = E[\prod_{i=2}^{d} F(\lambda_i X_1)] \tag{5.62}$$

If λ_1 is large compared to the other λ_i, then $\lambda_i X_1$ will typically be small and so the random variable in the expectation is very small. But that is ok. It is much better that trying to do MC on an indicator function that is 0 with very high probability.

Example In the network example, take $\vec{Z} = (T_1, T_2, T_3)$. It is possible, but not trivial, to compute $E[X|T_1, T_2, T_3]$ where X is the minimum time to get from A to B.

Stop - Wed, 2/10

Chapter 6

Importance sampling

6.1 The basics

To movtivate our discussion consider the following situation. We want to use Monte Carlo to compute $\mu = E[X]$. There is an event E such that $P(E)$ is small but X is small outside of E. When we run the usual Monte Carlo algorithm the vast majority of our samples of X will be outside E. But outside of E, X is close to zero. Only rarely will we get a sample in E where X is not small.

Most of the time we think of our problem as trying to compute the mean of some random variable X. For importance sampling we need a little more structure. We assume that the random variable we want to compute the mean of is of the form $f(\vec{X})$ where $\vec{X}$ is a random vector. We will assume that the joint distribution of $\vec{X}$ is absolutely continous and let $p(\vec{x})$ be the density. (Everything we will do also works for the case where the random vector $\vec{X}$ is discrete.) So we focus on computing

$$Ef(\vec{X}) = \int f(\vec{x})p(\vec{x})dx \tag{6.1}$$

Sometimes people restrict the region of integration to some subset D of R^d. (Owen does this.) We can (and will) instead just take $p(x) = 0$ outside of D and take the region of integration to be R^d.

The idea of importance sampling is to rewrite the mean as follows. Let $q(x)$ be another probability density on R^d such that $q(x) = 0$ implies $f(x)p(x) = 0$. Then

$$\mu = \int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)}\, q(x)\, dx \tag{6.2}$$

We can write the last expression as

$$E_q \left[\frac{f(\vec{X})p(\vec{X})}{q(\vec{X})} \right] \tag{6.3}$$

where E_q is the expectation for a probability measure for which the distribution of $\vec{X}$ is $q(x)$ rather than $p(x)$. The density $p(x)$ is called the *nominal or target distribution* , $q(x)$ the *importance or proposal distribution* and $p(x)/q(x)$ the *likelihood ratio*. Note that we assumed that $f(x)p(x) = 0$ whenever $q(x) = 0$. Note that we do not have to have $p(x) = 0$ for all x where $q(x) = 0$.

The importance sampling algorithm is then as follows. Generate samples $\vec{X}_1, \cdots, \vec{X}_n$ according to the distribution $q(x)$. Then the estimator for μ is

$$\hat{\mu}_q = \frac{1}{n} \sum_{i=1}^{n} \frac{f(\vec{X}_i)p(\vec{X}_i)}{q(\vec{X}_i)} \tag{6.4}$$

Of course this is doable only if $f(x)p(x)/q(x)$ is computable.

Theorem 10 *$\hat{\mu}_q$ is an unbaised estimator of μ, i.e., $E_q\hat{\mu}_q = \mu$. Its variance is σ_q^2/n where*

$$\sigma_q^2 = \int \frac{f(x)^2 p(x)^2}{q(x)} dx - \mu^2 = \int \frac{(f(x)p(x) - \mu q(x))^2}{q(x)} dx \tag{6.5}$$

Proof: Straightforward. **QED**

We can think of this importance sampling Monte Carlo algorithm as just ordinary Monte Carlo applied to $E_q[f(\vec{X})p(\vec{X})/q(\vec{X})]$. So a natural estimator for the variance is

$$\hat{\sigma}_q^2 = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{f(\vec{X}_i)p(\vec{X}_i)}{q(\vec{X}_i)} - \hat{\mu}_q \right]^2 \tag{6.6}$$

What is the optimal choice of the importance distribution $q(x)$? Looking at the theorem we see that if we let $q(x) = f(x)p(x)/\mu$, then the variance will be zero. This is a legitimate probability density if $f(x) \geq 0$. Of course we cannot really do this since it would require knowing μ. But this gives us a strategy. We would like to find a density $q(x)$ which is close to being proportional to $f(x)p(x)$.

What if $f(x)$ is not positive? Then we will show that the variance is minimized by taking $q(x)$ to be proportional to $|f(x)|p(x)$.

Theorem 11 *Let $\overline{q}(x) = |f(x)|p(x)/c$ where c is the constant that makes this a probability density. Then for any probability density $q(x)$ we have $\sigma_{\overline{q}} \leq \sigma_q$*

Proof: Note that $c = \int |f(x)|q(x)dx$.

$$\begin{aligned}
\sigma_{\overline{q}} - \mu^2 &= \int \frac{f(x)^2 p(x)^2}{\overline{q}(x)} dx && (6.7)\\
&= c \int |f(x)|p(x)dx && (6.8)\\
&= \left(\int |f(x)|p(x)dx \right)^2 && (6.9)\\
&= \left(\int \frac{|f(x)|p(x)}{q(x)} q(x)dx \right)^2 && (6.10)\\
&\leq \int \frac{f(x)^2 p(x)^2}{q(x)^2} q(x)dx && (6.11)\\
&= \int \frac{f(x)^2 p(x)^2}{q(x)} dx && (6.12)\\
&= \sigma_q - \mu^2 && (6.13)
\end{aligned}$$

where we have used the Cauchy Schwarz inequality with respect to the probaility measure $q(x)dx$. (One factor is the function 1.) **QED**.

Since we do not know $\int f(x)p(x)dx$, we probably do not know $\int |f(x)|p(x)dx$ either. So the optimal sampling density given in the theorem is not realizable. But again, it gives us a strategy. We want a sampling density which is approximately proportional to $|f(x)|p(x)$.

Big warning: Even if the original $f(\vec{X})$ has finite variance, there is no guarantee that σ_q will be finite. Discuss heavy tails and light tails.

How the sampling distribution should be chosen depends very much on the particular problem. Nonetheless there are some general ideas which we illustrate with some trivial examples.

If the function $f(x)$ is unbounded then ordinary Monte Carlo may have a large variance, possibly even infinite. We may be able to use importance sampling to turn a problem with an unbounded random variable into a problem with a bounded random variable.

Example We want to compute the integral

$$I = \int_0^1 x^{-\alpha} e^{-x} \, dx \tag{6.14}$$

where $0 < \alpha < 1$. So the integral is finite, but the integrand is unbounded. We take $f(x) = x^{-\alpha}e^{-x}$ and the nominal distribution is the uniform distribution on $[0, 1]$. Note that f will have infinite variance if $\alpha \leq -1/2$.

We take the sampling distribution to be

$$q(x) = \frac{1}{1-\alpha}x^{-\alpha} \tag{6.15}$$

on $[0, 1]$. This can be sampled using inversion. We have

$$f(x)\frac{p(x)}{q(x)} = e^{-x}(1-\alpha) \tag{6.16}$$

So we do a Monte Carlo simulation of $E_q[e^{-X}(1-\alpha)]$ where X has distribution q. Note that $e^{-X}(1-\alpha)$ is a bounded random variable.

The second general idea we illustrate involves rare-event simulation. This refers to the situation where you want to compute the probabily of an event when that probability is very small.

Example: Let Z have a standard normal distribution. We want to compute $P(Z \geq 4)$. We could do this by a Monte Carlo simulation. We generate a bunch of samples of Z and count how many satisfy $Z \geq 4$. The problem is that there won't be very many (probably zero). If $p = P(Z \geq 4)$, then the variance of $1_{Z\geq 4}$ is $p(1-p) \approx p$. So the error with n samples is of order $\sqrt{p/n}$. So this is small, but it will be small compared to p only if n is huge.

Our nominal distribution is

$$p(x) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}x^2) \tag{6.17}$$

We take the sampling distribution to be

$$q(x) = \begin{cases} e^{-(x-4)}, & \text{if } x \geq 4, \\ 0, & \text{if } x < 4, \end{cases} \tag{6.18}$$

The sampling distribution is an exponential shifted to the right by 4. In other words, if Y has an exponential distribution with mean 1, then $Y + 4$ has the distribution q. The probability we want to compute is

$$p = \int 1_{x\geq 4}p(x)\,dx \tag{6.19}$$

$$= \int 1_{x\geq 4}\frac{p(x)}{q(x)}q(x)\,dx \tag{6.20}$$

The likehood ratio is

$$w(x) = \frac{p(x)}{q(x)} = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2 + x - 4) \tag{6.21}$$

On $[4, \infty)$ this function is decreasing. So its maximum is at 4 where its value is $\exp(-8)/\sqrt{2\pi}$ which is really small. The variance is no bigger than the second moment which is bounded by this number squared. This is $\exp(-16)/2\pi$. Compare this with the variance of ordinary MC which saw was of the order of p which is on the order of $\exp(-8)$. So the decrease in the variance is huge.

Example We return to the network example, following Kroese's review article. Let $U_1, U_2, \cdots, U_5$ be independent and uniform on $[0, 1]$. Let T_i be U_i multiplied by the approriate constant to give the desired distribution for the times T_i. We want to estimate the mean of $f(U_1, \cdots, U_5)$ where f is the minimum time. The nominal density is $p(u) = 1$ on $[0, 1]^5$. For our sampling density we take

$$g(u) = \prod_{i=1}^{5} \nu_i u_i^{\nu_i - 1} \tag{6.22}$$

where the ν_i are parameters. (This is a special case of the beta distribution.) Note that $\nu_i = 1$ gives the nominal distribution p. There is no obvious choice for the ν_i. Kroese finds that with $\nu = (1.3, 1.1, 1.1, 1.3, 1.1)$ the variance is reduced by roughly a factor of 2.

We have discussed importance sampling in the setting where we want to estimate $E[f(\vec{X})]$ and $\vec{X}$ is jointly absolutely continuous. Everything we have done works if $\vec{X}$ is a discrete RV. For this discussion I will drop the vector notation. So suppose we want to compute $\mu = E[f(X)]$ where X is discrete with probability mass function $p(x)$, i.e., $p(x) = P(X = x)$. If $q(x)$ is another discrete distribution such that $q(x) = 0$ implies $f(x)p(x) = 0$, then we have

$$\mu = E[f(X)] = \sum_x f(x)p(x) = \sum_x \frac{f(x)p(x)}{q(x)} q(x) = E_q[\frac{f(x)p(x)}{q(x)}] \tag{6.23}$$

where E_q means expectation with repect to $q(x)$.

Example - union counting problem (from Fishman)
We have a finite set which we will take to just be $\{1, 2, \cdots, r\}$ and will call Ω. We also have a collection $S_j, j = 1, \cdots, m$ of subsets of Ω. We know r, the cardinality of Ω and the cardinalities $|S_j|$ of all the given subsets. Througout this example we use $|\quad|$ to denote the cardinality of a set. We want to compute $l = |U|$ where

$$U = \cup_{j=1}^{m} S_j \tag{6.24}$$

We assume that r and l are huge so that we cannot do this explicitly by finding all the elements in the union. We can do this by a straightforward Monte Carlo if two conditions are

met. First, we can sample from the uniform distribution on Ω. Second, given an $\omega \in \Omega$ we can determine if $\omega \in S_j$ in a reasonable amount of time. The MC algorithm is then to generate a large number, n, of samples ω_i from the uniform distribution on Ω and let X be the number that are in the union U. Our estimator is then rX/n. We are computing $E_p[f(\omega)]$, where

$$f(\omega) = r1_{\omega \in U} \tag{6.25}$$

We are assuming r and n are both large, but suppose r/n is small. Then this will be an inefficient MC method.

For our importance sampling algorithm, define $s(\omega)$ to be the number of subsets S_j that contain ω, i.e.,

$$s(\omega) = |\{j : \omega \in S_j\}| \tag{6.26}$$

and let $s = \sum_\omega s(\omega)$. Note that $s = \sum_j |S_j|$. The importance distribution is taken to be

$$q(\omega) = \frac{s(\omega)}{s} \tag{6.27}$$

The likelihood ratio is just

$$\frac{p(\omega)}{q(\omega)} = \frac{s}{rs(\omega)} \tag{6.28}$$

Note that $q(\omega)$ is zero when $f(\omega)$ is zero. So $f(\omega)q(\omega)/p(\omega)$ is just $\frac{s}{s(\omega)}$. We then do a Monte Carlo to estimate

$$E_q[\frac{s}{s(\omega)}] \tag{6.29}$$

However, is it really feasible to sample from the q distribution? Since l is huge a direct attempt to sample from it may be impossible. We make two assumptions. We assume we know $|S_j|$ for all the subsets, and we assume that for each j, we can sample from the uniform distribution on S_j. Then we can sample from q as follows. First generate a random $J \in \{1, 2, \cdots, m\}$ with

$$P(J = j) = \frac{|S_j|}{\sum_{i=1}^m |S_i|} \tag{6.30}$$

Then sample ω from the uniform distribution on S_J. To see that this gives the desired density $q()$, first note that if ω is not in $\cup_i S_i$, then there is no chance of picking ω. If ω is in the union, then

$$P(\omega) = \sum_{j=1}^m P(\omega|J = j)P(J = j) = \sum_{j:\omega \in S_j} \frac{1}{|S_j|} \frac{|S_j|}{\sum_{i=1}^m |S_i|} = \frac{s(\omega)}{s} \tag{6.31}$$

Fishman does a pretty complete study of the variance for this importance sampling algorithm. Here we will just note the following. The variance will not depend on n. So if n, r are huge but r/n is small then the importance sampling algorithm will certainly do better than the simple Monte Carlo of just sampling uniformly from Ω.

Stop - Mon, 2/15

6.2 Self-normalized importance sampling

In many problems the density we want to sample from is only known up to an unknown constant, i.e., $p(x) = c_p p_0(x)$ where $p_0(x)$ is known, but c_p is not. Of course c_p is determined by the requirement that the integral of $p(x)$ be 1, but we may not be able to compute the integral. Suppose we are in this situation and we have another density $q(x)$ that we can sample from. It is also possible that $q(x)$ is only known up to a constant, i.e., $q(x) = c_q q_0(x)$ were $q_0(x)$ is known but c_q is not known.

The idea of self-normalizing is based on

$$\begin{aligned}
\int f(x)p(x)dx &= \int \frac{f(x)p(x)}{q(x)} q(x)dx && (6.32)\\
&= \frac{\int \frac{f(x)p(x)}{q(x)} q(x)dx}{\int \frac{p(x)}{q(x)} q(x)dx} && (6.33)\\
&= \frac{\int \frac{f(x)p_0(x)}{q_0(x)} q(x)dx}{\int \frac{p_0(x)}{q_0(x)} q(x)dx} && (6.34)\\
&= \frac{\int f(x)w(x)q(x)dx}{\int w(x)q(x)dx} && (6.35)\\
&= \frac{E_q[f(x)w(x)]}{E_q[w(x)]} && (6.36)
\end{aligned}$$

where $w(x) = p_0(x)/q_0(x)$ is a known function.

The self-normalized importance sampling algorithm is as follows. We generate samples

$\vec{X}_1, \cdots, \vec{X}_n$ according to the distribution $q(x)$. Our estimator for $\mu = \int f(x)p(x)dx$ is

$$\hat{\mu} = \frac{\sum_{i=1}^n f(\vec{X}_i)w(\vec{X}_i)}{\sum_{i=1}^n w(\vec{X}_i)} \tag{6.37}$$

Theorem 12 hypotheses *The estimator $\hat{\mu}$ converges to μ with probability* 1.

Proof: Note that

$$\hat{\mu} = \frac{\frac{1}{n}\sum_{i=1}^n f(\vec{X}_i)w(\vec{X}_i)}{\frac{1}{n}\sum_{i=1}^n w(\vec{X}_i)} = \frac{\frac{1}{n}\sum_{i=1}^n f(\vec{X}_i)\frac{c_q}{c_p}\frac{p(\vec{X}_i)}{q(\vec{X}_i)}}{\frac{1}{n}\sum_{i=1}^n \frac{c_q}{c_p}\frac{p(\vec{X}_i)}{q(\vec{X}_i)}} = \frac{\frac{1}{n}\sum_{i=1}^n f(\vec{X}_i)\frac{p(\vec{X}_i)}{q(\vec{X}_i)}}{\frac{1}{n}\sum_{i=1}^n \frac{p(\vec{X}_i)}{q(\vec{X}_i)}} \tag{6.38}$$

Now apply the strong law of large number to the numerator and denominator separately. Remember that $\vec{X}$ is sampled from $q(x)$, so the numerator converges to $\int f(x)p(x)dx = \mu$. The denominator converges to $\int p(x)dx = 1$. **QED**

It should be noted that the expected value of $\hat{\mu}$ is not exactly μ. The estimator is slightly biased.

To find a confidence interval for self normalized importance sampling we need to compute the variance of $\hat{\mu}$. We already did this using the delta method. In $\hat{\mu}$ the numerator is the sample mean for fw and the denominator is the sample mean for w. Plugging this into our result from the delta method we find that an estimator for the variance of $\hat{\mu}$ is

$$\frac{\sum_{i=1}^n w(\vec{X}_i)^2(f(\vec{X}_i) - \hat{\mu})^2}{(\sum_{i=1}^n w(\vec{X}_i))^2} \tag{6.39}$$

If we let $w_i = w(\vec{X}_i)/\sum_{j=1}^n w(\vec{X}_j)$, then this is just

$$\sum_{i=1}^n w_i^2(f(\vec{X}_i) - \hat{\mu})^2 \tag{6.40}$$

In ordinary Monte Carlo all of our samples contribute with equal weight. In importance sampling we give them different weights. The total weight of the weights is $\sum_{i=1}^n w_i$. It is possible that most of this weight is concentrated in a just a few weights. If this happens we expect the important sampling Monte Carlo will have large error. We might hope that when this happens our estimate of the variance will be large and so this will alert us to the problem. However, our estimate of the variance σ_q uses the same set of weights, so it may not be accruate when this happens.

Another way to check if we are getting grossly imbalanced weights is to compute an effective sample size. Consider the following toy problem. Let $w_1, \cdots, w_n$ be constants (not random).

Let $Z_1, \cdots, Z_n$ be i.i.d. random variables with common variance σ^2. An estimator for the mean of the Z_i is

$$\hat{\mu} = \frac{\sum_{i=1}^n w_i Z_i}{\sum_{i=1}^n w_i} \tag{6.41}$$

The variance of $\hat{\mu}$ is

$$var(\hat{\mu}) = \sigma^2 \frac{\sum_{i=1}^n w_i^2}{(\sum_{i=1}^n w_i)^2} \tag{6.42}$$

Now define the number of effective samples n_e to be the number of independent samples we would need to get the same variance if we did not use the weights. In this case the variance is σ^2/n_e. So

$$n_e = \frac{(\sum_{i=1}^n w_i)^2}{\sum_{i=1}^n w_i^2} \tag{6.43}$$

As an example, suppose that k of the w_i equal 1 and the rest are zero. The a trivial calculation shows $n_e = k$.

Note that this definition of the effective sample size only involves the weights. It does not take f into account. One can also define an effective sample size that depends on f. See Owen.

6.3 Variance minimization and exponential tilting

Rather than consider all possbile choices for the sampling distribution $q(x)$, one strategy is to restrict the set of $q(x)$ we consider to some family of distributions and minimize the variance σ_q over this family. So we assume we have a family of distributions $p(x, \theta)$ where θ parameterizes the family. Here x is multidimensional and so is θ, but the dimensions need not be the same. We let θ_0 be the parameter value that corresponds to our nominal distribution. So $p(x) = p(x, \theta_0)$. The weighting function is

$$w(x, \theta) = \frac{p(x, \theta_0)}{p(x, \theta)} \tag{6.44}$$

The importance sampling algorithm is based on

$$\mu = E_{\theta_0}[f(\vec{X})] = \int f(x) p(x, \theta_0) dx = \int f(x) w(x, \theta) p(x, \theta) dx = E_\theta[f(\vec{X}) w(\vec{X}, \theta)] \tag{6.45}$$

The variance for this is

$$\sigma^2(\theta) = \int f(x)^2 w(x, \theta)^2 p(x, \theta) dx - \mu^2 \tag{6.46}$$

We want to minimize this as a function of θ. One approach would be for each different value of θ we run a MC simulation where we sample from $p(x,\theta)$ and use these samples to estimate $\sigma^2(\theta)$. This is quite expensive since it involves a simulation for every value of θ we need to consider.

A faster approach to search for the best θ is the following. Rewrite the variance as

$$\sigma^2(\theta) = \int f(x)^2 w(x,\theta) p(x,\theta_0) dx - \mu^2 = E_{\theta_0}[f(\vec{X})^2 w(\vec{X},\theta)] - \mu^2 \tag{6.47}$$

Now we run a single MC simulation where we sample from $p(x,\theta_0)$. Let $\vec{X_1}, \vec{X_2}, \cdots, \vec{X_m}$ be the samples. We then use the following to estimate σ_θ:

$$\hat{\sigma_0(\theta)}^2 = \frac{1}{m}\sum_{i=1}^{m} f(\vec{X_i})^2 w(\vec{X_i},\theta) - \mu^2 \tag{6.48}$$

The subscript 0 on the estimator is to remind us that we used a sample from $p(x,\theta_0)$ rather than $p(x,\theta)$ in the estimation. We then use our favorite numerical optimization method for minimizing a function of several variables to find the minimum of this as a function of θ. Let θ^* be the optimal value.

Now we return to the original problem of estimating μ. We generate samples of $\vec{X}$ according to the distribution $p(x,\theta^*)$. We then let

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} f(\vec{X_i}) w(\vec{X_i},\theta^*) \tag{6.49}$$

The variance of this estimator is $\sigma^2_{\theta^*}/n$. Our estimator for the variance $\sigma^2_{\theta^*}$ is

$$\hat{\sigma^2(\theta^*)} = \frac{1}{m}\sum_{i=1}^{m} f(\vec{X_i})^2 w(\vec{X_i},\theta)^2 \tag{6.50}$$

The above algorithm can fail completely if the distribution $p(x,\theta_0)$ is too far from a good sampling distribution. We illustrate this with an example.

Example: We return to an earlier example. Z is a standard normal RV and we want to compute $P(Z > 4)$. We take for our family the normal distributions with variance 1 and mean θ. So

$$p(x,\theta) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x-\theta)^2) \tag{6.51}$$

and the nominal density $p(x)$ is $p(x,0)$.

MORE MORE MORE MORE

We can fix the problem above as follows. Instead of doing our single MC run by sampling from $p(x, \theta_0)$, we sample from $p(x, \theta_r)$ that θ_r, our "reference" θ, is our best guess for a good choice of θ. Rewrite the variance as

$$\sigma^2(\theta) = \int f(x)^2 w(x,\theta)^2 p(x,\theta) dx - \mu^2 \tag{6.52}$$

$$= \int f(x)^2 \frac{p(x,\theta_0)^2}{p(x,\theta)p(x,\theta_r)} p(x,\theta_r) dx - \mu^2 \tag{6.53}$$

We then generate samples $\vec{X}_1, \cdots, \vec{X}_n$ from $p(x, \theta_r)$. Our estimator for the variance $\sigma(\theta)^2$ is then

$$\hat{\sigma_r^2}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{f(\vec{X}_i)^2 p(\vec{X}_i, \theta_0)^2}{p(\vec{X}_i, \theta) p(\vec{X}_i, \theta_r)} - \hat{\mu}^2 \tag{6.54}$$

For several well-known classes of distributions the ratio $p(x)/q(x)$ takes a simple form. An exponential family is a family such that

$$p(x;\theta) = \exp((\eta(\theta), T(x)) - A(x) - C(\theta)) \tag{6.55}$$

for functions $\eta(\theta), T(X), A(x), C(\theta)$. The following are examples. A multivariate normal with a fixed covariance matrix is an exponential distribution where the means are the parameters. The possion distribution is an exponential family where the parameter is the usual (one-dimensional) λ. If we fixed the number of trials, then the binominal distribution is an exponential family with parameter p. The gamma distribution is also an exponential family. In many cases the weight function just reduces to $\exp((\theta, x))$. Even if $p(x)$ does not come from an exponential family we can still look for a proposal density of the form

$$q(x) = \frac{1}{Z(\theta)} \exp((\theta, x)) p(x) \tag{6.56}$$

where $Z(\theta)$ is just the normalizing constant. Importance sampling in this case is often called exponential tilting.

Example Comment on network example.

Stop - Wed, Feb 17

6.4 Processes

Now suppose that instead of a random vector we have a stochastic process $X_1, X_2, X_3, \cdots$. We will let X stand for $X_1, X_2, X_3, \cdots$. We want to estimate the mean of a function of the process $\mu = f(X)$. It doesn't make sense to try to give a probability density for the full infinite process. Instead we specify it through conditional densities:
$p_1(x_1), p_2(x_2|x_1), p_3(x_3|x_1, x_2), \cdots, p_n(x_n|x_1, x_2, \cdots, x_{n-1}), \cdots$. Note that it is immediate from the definition of conditional density that

$$\begin{aligned} p(x_1, x_2, \cdots, x_n) &= p_n(x_n|x_1, x_2, \cdots, x_{n-1}) p_{n-1}(x_{n-1}|x_1, x_2, \cdots, x_{n-2}) && (6.57) \\ &\cdots \quad p_3(x_3|x_1, x_2) p_2(x_2|x_1) p_1(x_1) && (6.58) \end{aligned}$$

We specify the proposal density in the same way:

$$\begin{aligned} q(x_1, x_2, \cdots, x_n) &= q_n(x_n|x_1, x_2, \cdots, x_{n-1}) q_{n-1}(x_{n-1}|x_1, x_2, \cdots, x_{n-2}) && (6.59) \\ &\cdots \quad q_3(x_3|x_1, x_2) q_2(x_2|x_1) q_1(x_1) && (6.60) \end{aligned}$$

So the likehood function is

$$w(x) = \prod_{n \geq 1} \frac{p_n(x_n|x_1, x_2, \cdots, x_{n-1})}{q_n(x_n|x_1, x_2, \cdots, x_{n-1})} \tag{6.61}$$

An infinite product raises convergence questions. But in applications f typically either depends on a fixed, finite number of the X_i or f depends on a finite but random number of the X_i. So suppose that f only depends on $X_1, \cdots, X_M$ where M may be random. To be more precise we assume that there is a random variable M taking values in the non-negative integers such that if we are given that $M = m$, then $f(X_1, X_2, \cdots)$ only depends on $X_1, \cdots, X_m$. So we can write

$$f(X_1, X_2, \cdots) = \sum_{m=1}^{\infty} 1_{M=m} \, f_m(X_1, \cdots, X_m) \tag{6.62}$$

We also assume that M is a stopping time. This means that the event $M = m$ only depends on $X_1, \cdots, X_m$. Now we define

$$w(x) = \sum_{m=1}^{\infty} 1_{M=m}(x_1, \cdots, x_m) \prod_{n=1}^{m} \frac{p_n(x_n|x_1, x_2, \cdots, x_{n-1})}{q_n(x_n|x_1, x_2, \cdots, x_{n-1})} \tag{6.63}$$

Example - random walk exit: This follows an example in Owens. Let ξ_i be an i.i.d. sequence of random variables. Let $X_0 = 0$ and

$$X_n = \sum_{i=1}^{n} \xi_i \tag{6.64}$$

In probability this is called a random walk. It starts at 0. Now fix an interval (a, b) with $0 \in (a, b)$. We run the run until it exits this interval and then ask whether it exited to the right or the left. So we let

$$M = \inf\{n : X_n \geq b \quad or \quad X_n < a\} \tag{6.65}$$

So the stopping condition is $X_M \geq b$ or $X_M \leq a$. Then we want to compute $\mu = P(X_M \geq b)$. We are particularily interested in the case where $E\xi_i < 0$. So the walk drifts to the left on average and the probability μ will be small if b is relatively large.

We take the walk to have steps with a normal distribution with variance 1 and mean -1. So the walk drifts to the left. We take $(a, b) = (-5, 10)$. We run the walk until is exits this interval and want to compute the probability it exits to the right. This is a very small probability. So the ξ_i are independent normal random variables with variance 1 and mean -1.

The conditional densities that determine the nominal distribution are given by

$$p(x_n|x_1, \cdots, x_{n-1}) = p(x_n|x_{n-1}) = f_{\xi_n}(x_n - x_{n-1}) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_n - x_{n-1} - \theta_0)^2) \tag{6.66}$$

In our example we take $\theta_0 = -1$. **MORE** **Explain how we sample this** A Monte Carlo simulation with no importance sampling with 10^6 samples produced no samples that exited to the right. So it gives the useless estimate $\hat{p} = 0$.

For the sampling distribution we take a random walk whose step distribution is normal with variance 1 and mean θ. So

$$q(x_n|x_1, \cdots, x_{n-1}) = q(x_n|x_{n-1}) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x_n - x_{n-1} - \theta)^2) \tag{6.67}$$

The weight factors are then

$$w_n(x_1, \cdots, x_n) = \exp((\theta_0 - \theta)(x_n - x_{n-1}) - \frac{1}{2}\theta_0^2 + \frac{1}{2}\theta^2) \tag{6.68}$$

With no idea of how to choose θ, we try $\theta = 0$ and find with 10^6 samples

$$p = 6.74 \times 10^{-10} \pm 0.33 \times 10^{-10} \tag{6.69}$$

The confidence intervals is rather large, so we do a longer run with 10^7 samples and find

$$p = 6.53 \times 10^{-10} \pm 0.098 \times 10^{-10} \tag{6.70}$$

The choice of $\theta = 0$ is far from optimal. More on this in a homework problem.

Part II

Markov Chain Monte Carlo

Chapter 7

Markov chain background

A stochastic process is a family of random variables $\{X_t\}$ indexed by a varaible t which we will think of as time. Time can be discrete or continuous. We will only consider the case of discrete time since that is what is relevant for MCMC. So we consider a sequence of random variables X_n indexed by a non-negative integer. The set of possible values of the random variables X_n is called the state space. It could be finite, countable or an uncountable set like R or R^d. The intuitive definition of a Markov process is that if you know $X_n = x$, then the probability distrbution of where you go next, i.e., of X_{n+1} only depends on x, not on how the process got to x. Loosely speaking, this says that the future (time $n+1$) depends on the past (times $1, 2, \cdots, n$) only through the present (time n).

We start with Markov chains with finite and then countable state spaces. For these sections I am extracting material from Durrett's *Essentials of Stochastic Processes*. Then for the section on uncountable state spaces I follow chapter 6 in the Robert and Casella book. We note that our study of Markov processes will be somewhat unbalanced since we are focusing on just the things we need for MCMC.

7.1 Finite state space

The state space is the set of possible values of the random variables X_n. In this section we study the case of finite S. A Markov chain is specified by giving a collection of transition probabilities $p(x, y)$ where $x, y \in S$. $p(x, y)$ is the probability of jumping to state y given that we are presently in state x. So if we keep x fixed and sum over y we must get 1.

Definition 2 *A transition function $p(x, y)$ is a non-negative function on $S \times S$ such that*

$$\sum_{y \in S} p(x, y) = 1 \tag{7.1}$$

To completely specify the Markov chain we also need to give the initial distribution of the chain, i.e., the distribution of X_0.

Definition 3 *Let S be a finite set and $p(x, y)$ a transition function for S. Let $\pi_0(x)$ be a probability distribuion on S. Then the Markov chain corresponding to initial distribution π_0 and transition probabilities $p(x, y)$ is the stochastic process X_n such that*

$$P(X_0 = x) = \pi_0(x), \tag{7.2}$$
$$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \cdots, X_2 = x_2, X_1 = x_1) = p(x_n, x_{n+1}) \tag{7.3}$$

Note that the last equation implies that

$$P(X_{n+1} = x_{n+1} | X_n = x_n) = p(x_n, x_{n+1}) \tag{7.4}$$

A hard core mathematician would feel compelled to prove at this point that such a process exists. This is not that easy (it requires an extension theorem), and we will not do it. We do note that if we want to sample the process, i.e., generate a sample path for X_n, this is straightforward. Sample X_0 according to the distribution π_0. Then sample X_1 according to the distribution $p(X_0, \cdot)$. Continue, sample X_n according to the distribution $p(X_{n-1}, \cdot)$.

The following is an easy consequence of the equations above

Proposition 3 *For any states $x_0, x_1, \cdots, x_n$,*

$$P(X_0 = x_0, X_1 = x_1, \cdots, X_n = x_n) = p(x_0, x_1)p(x_1, x_2) \cdots p(x_{n-1}, x_n) \tag{7.5}$$

The transition matrix gives the transition probabilities for one time step. We consider the transition probabilities for longer times. Let $p^m(x, y) = P(X_{n+m} = y | X_n = x)$.

Proposition 4 *(Chapman-Kolmogorov equation)*

$$p^{m+k}(x, y) = \sum_z p^m(x, z)p^k(z, y) \tag{7.6}$$

If we think of $p^n(x, y)$ as the elements of a matrix, then this matrix is just p raised to the n where p is the matrix with matrix elements $p(x, y)$. If $\pi_n(x) = P(X_n = x)$ and we think of π_n as a row vector, then $\pi_n = \pi_0 p^n$.

We are primarily interested in the long time behavior of our chain. We impose some conditions on the chain for this study to rule out chains that we do not care about for MCMC. It is possible that the state space decomposes into several subsets with no transitions between the subsets. Or there could be subsets which have transitions out of the subset but not into it. **Give some examples of these** To eliminate these sorts of chains we make the following definition. The definition says that a chain is irreducible if it is possible to transition from any state to any other state in some finite number of steps.

Definition 4 *A Markov chain is irreducible if for every $x, y \in S$ there is an integer n and states $x_0, x_1, \cdots x_n$ such that $x = x_0$ and $y = x_n$ and $p(x_{i-1}, x_i) > 0$ for $i = 1, \cdots, n$. In other words, there is an integer n such that $p^n(x, y) > 0$*

Define $T_x = \min\{n \geq 1 : X_n = y\}$. This is the time of the first return (after time 0) to state x. Let P_x denote the probability measure when $X_0 = x$. A state is recurrent if $P_x(T_x < \infty) = 1$. So if we start in x we will eventually return to x. If this probability is less than 1 we say the state is transient. It can be shown that if a finite state Markov chain is irreducible, then every state x is recurrent. Finite state Markov chains can have transient states, but only if they are not irreducible.

We need to rule out one more type of chain. **Give example of periodic chain.**

Definition 5 *Let $x \in S$. The period of x is the greatest common division of the set of integers n such that $p^n(x, y) > 0$.*

Theorem 13 *In an irreducible chain all the states have the same period.*

Definition 6 *An irreducible chain is aperiodic if the common period of the states is* 1.

Note that if there is a state x such that $p(x, x) > 0$, then the period of x is 1. So if we have an irreducible chain with a state x such that $p(x, x) > 0$ then the chain is aperiodic. The condition $p(x, x) > 0$ says that if you are in state x there there is nonzero probabiity that you stay in state x for the next time step. In many applications of Markov Chains to Monte Carlo there are states with this property, and so they are aperiodic.

An irreducible, aperiodic Markov chain has nice long time behavior. It is determined by the stationary distribution.

Definition 7 *A distribution $\pi(x)$ on S is stationary if $\pi P = \pi$, i.e.,*

$$\sum_{y \in S} \pi(y) p(y, x) = \pi(x) \tag{7.7}$$

In words, a distribution is stationary if it is invariant under the time evolution. If we take the initial distribution to be the stationary distribution, than for all n the distribution of X_n is the stationary distribution. Note that the definition says that π is a left eigenvector of the transition matrix with eigenvalue 1. It may seem a little strange to be working with left eigenvectors rather than the usual right eigenvectors, but this is just a consequence of the convention that $P(X_{n+1} = y|X_n = x)$ is $p(x, y)$ rather than $p(y, x)$.

Theorem 14 *An irreducible Markov chain has a unique stationary distribution π. Furthermore, for all states x, $\pi(x) > 0$.*

Idea of proof: Eq. (7.1) implies that the constant vector is a right eigenvector of the transition matrix with eigenvalue 1. So there must exist a left eigenvector with eigenvalue 1. To see that it can be chosen to have all positive entries and is unique one can use the Perron-Frobenius theorem. Durrett has a nice proof.

Stop - Mon, 2/22

In many problems there is a short-cut for finding the stationary distribution.

Definition 8 *A state π is said to satisfy detailed balance if for all states x, y*

$$\pi(x)p(x, y) = \pi(y)p(y, x) \tag{7.8}$$

Note there are no sums in this equation.

Proposition 5 *If a distribution π satisfies detailed balance, then π is a stationary distribution.*

Proof: Sum the above equation on y. **QED**

The converse is very false. **Example**

There are two types of convergence theorems. In the first type we start the chain in some initial distribution and ask what happens to the distribution of X_n as $n \to \infty$.

Theorem 15 *Let $p(x, y)$ be the transition matrix of an irreducible, aperiodic finite state Markov chain. Then for all states x, y,*

$$\lim_{n\to\infty} p^n(x, y) = \pi(y) \tag{7.9}$$

For any initial distribution π_o, the distribution π_n of X_n converges to the stationary distribution π.

The second type of convergence theorem is not a statement about distributions. It is a statement that involves a single sample path of the process.

Theorem 16 *Consider an irreducible, finite state Markov chain. Let $f(x)$ be a function on the state space, and let π be the stationary distribution. Then for any initial distribution,*

$$P(\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(X_k) = \sum_x f(x)\pi(x)) = 1 \tag{7.10}$$

The second convergence theorem is the one that is relevant to MCMC. It says that if we want to compute the expected value of f in the probability measure π and π is the stationary distribution of some Markov chain, then we can run the chain for a long time and compute the long time average of $f(X_k)$ to get an approximation to the expected value of f.

7.2 Countable state space

Much of the finite state stuff carries over immediately. In particular the Chapman-Komogorov eq. and the fact that the n step transition matrix is the n power of the transition matrix. (Note that we now have infinite matrices.) The definition of irreducible and the period of a state is the same. And in an irreducible Markov chain, all states have the same period.

Recall that $T_x = \min\{n \geq 1 : X_n = x\}$, the time of the first return to state x. And a state is recurrent if $P_x(T_x < \infty) = 1$. There are two big changes when we go to infinite but countable state spaces. First, there can be transient states even if the chain is irreducible. Second, irreducible chains need not have stationary distibutions when they are recurrent. The definition of recurrence needs to be refined.

In a finite state Markov chain the expected value $E_x[T_x]$ is always finite for a recurrent state. But in an infinite chain, it can be infinite. If $E_x[T_x] < \infty$ we say the state is **positive recurrent.** If $E_x[T_x] = \infty$ but $P_x(T_x < \infty) = 1$, we say the state is **null recurrent.** States that are neither null or positive recurrent are said to be **transient**.

Theorem 17 *In an irreducible chain either*
(i) All states are transient
(ii) All states are null recurrent
(iii) All states are positive recurrent

MORE Example We consider a random walk on the non-negative integers. Let $0 < p < 1$. The walk jumps right with probability p, left with probability $1 - p$. If it is at the origin it jumps right with probability 1. Chain is positive recurrent if $p < 1/2$, null recurrent if $p = 1/2$ and transient if $p > 1/2$.

Theorem 18 *For an irreducible Markov chain with countable state space the following are equivalent.*

(i) All states are positive recurrent.

(ii) There is a stationary distribution π. (It is a distribution, so in particular $\sum_x \pi(x) < \infty$.)

The two big convergence theorems of the previous section hold if we add the hypothesis that there is a stationary distribution.

Theorem 19 *Let $p(x, y)$ be the transition matrix of an irreducible, aperiodic countable state Markov chain which has a stationary distribution. Then for all states x, y,*

$$\lim_{n\to\infty} p^n(x, y) = \pi(y) \tag{7.11}$$

For any initial distribution π_0, the distribution π_n of X_n converges to the stationary distribution π in the total variation norm.

In the setting of a discrete (finite or countable) state space, the total variation norm is just an l^1 norm:

$$||p^n(x, \cdot) = \pi(\cdot)||_{TV} = \sum_y |p^n(x, y) = \pi(y)| \tag{7.12}$$

Theorem 20 *Consider an irreducible, countable state Markov chain which has a stationary distribution π. Let $f(x)$ be a function on the state space such that $\sum_x |f(x)|\pi(x) < \infty$. Then for any initial distribution,*

$$P(\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(X_k) = \sum_x f(x)\pi(x)) = 1 \tag{7.13}$$

The definition of detailed balance is just the same as in the finite state space case. And we have

Proposition 6 *If a distribution π satisfies detailed balance, then π is a stationary distribution.*

7.3 General state space

For finite or countable state spaces the transition kernel is a function on $S \times S$. Since $\sum_y p(x,y) = 1$, if we fix x we can think of $p(x,\cdot)$ as a probability measure. For $A \subset S$, define

$$K(x,A) = \sum_{y \in A} p(x,y) \tag{7.14}$$

So $K(x,A)$ is the probability we jump to some state in the set A given that we are currently in state x.

Definition 9 *Let S be a set and $\mathcal{S}$ a σ-field on S. The set S is called the state space. A transition kernel K is a function from $S \times \mathcal{S}$ into $[0,1]$ such that*

(i) For all $x \in S$, $K(x,\cdot)$ is a probability measure on $(S,\mathcal{S})$.

(ii) For all $A \in \mathcal{S}$, $K(\cdot,A)$ is a measurable function on S.

If S is finite or countable, then the transition function we considered in the previous section is just given by $p(x,y) = K(x,\{y\})$

Suppose the state space is a subset of R^d and for all x, the measure $K(x,\cdot)$ is absolutely continuous with respect to Lebesgue measure. So there is a non-negative function $k(x,y)$ such that

$$K(x,A) = \int_A k(x,y)\,dy \tag{7.15}$$

for $A \in \mathcal{S}$. In this case we will refer to $k(x,y)$ as the transition function. Note that it must satisfy

$$\int_S k(x,y)dy = 1, \quad \forall x \tag{7.16}$$

Note: Robert and Casella write the density for the transition kernel as $K(x,y)$ rather then $k(x,y)$.

There are other situations in which there is a density of sorts. Something like the following will come up when we look at Gibbs samplers. To keep the notation simpler we consider two dimensions. Suppose the state space is R^2 or a subset of it. We denote out states by (x, y) and denote the Markov process by (X_n, Y_n). We consider transition kernels that describe the following. We flip a fair coin to decide whether we change the first component or the second component. If we are changing the first component then the new state (X_{n+1}, Y_{n+1}) is (X_n, Y_{n+1}) where the distribution of Y_{n+1} is absolutely continuous with respect to 1d Lebesgue measure with a density that depends on (X_n, Y_n). And if we are changing the second component, ... So we have two functions $k_1(x, y; z)$ and $k_2(x, y; z)$ such that

$$K((x_0, y_0), \cdot) = \frac{1}{2}[\delta_{x,x_0} \times k_2(x_0, y_0; y)dy + k_2(x_0, y_0; x)dx \times \delta_{y,y_0}] \tag{7.17}$$

where we let (x, y) be the variables for the measure in $K((x_0, y_0), \cdot)$.

Definition 10 *Let K be a transition matrix on the state space $(S, \mathcal{S})$. Let μ be a probability measure on $(S, \mathcal{S})$. A sequence of random variables $X_0, X_1, X_2, \cdots$ is a Markov process with transition kernel K and initial distribution μ if for all $k = 0, 1, 2, \cdots$,*

$$P(X_{k+1} \in A | X_0, X_1, \cdots, X_k) = \int_A K(X_k, dx) \tag{7.18}$$

and the distribution of X_0 is μ.

It follows immediately from the definition that

$$P(X_{k+1} \in A | X_k) = \int_A K(X_k, dx) \tag{7.19}$$

Notation: The above equation is sometimes written as (Robert and Casella do this)

$$P(X_{k+1} \in A | x_0, x_1, \cdots, x_k) = P(X_{k+1} \in A | x_k) = \int_A K(x_k, dx) \tag{7.20}$$

This should be taken to mean

$$P(X_{k+1} \in A | X_0 = x_0, X_1 = x_1, \cdots, X_k = x_k) = P(X_{k+1} \in A | X_k = x_k) = \int_A K(x_k, dx) \tag{7.21}$$

The probability measure for the process depends on the transition kernel and the initial distribution. Typically the kernel is kept fixed, but we may consider varying the initial distribution. So we let P_μ denote the probability measure for initial distribution μ. We denote

the corresponding expectation by E_μ. The fact that such a Markov process exists is quite non-trivial.

Example (Random walk) Let ξ_n be an iid sequence of RV's, and let

$$X_n = \sum_{i=1}^{n} \xi_i \tag{7.22}$$

Since $X_{n+1} = X_n + \xi_{n+1}$, $K(x, \cdot) = \mu_{\xi_{n+1}+x}$ where $\mu_{\xi_{n+1}+x}$ denotes the distribution measure of the RV $\xi_{n+1} + x$.

Example (AR(1)) Let ϵ_n be an iid sequence. For example they could be standard normal RV's. Let θ be a real constant. Then define

$$X_n = \theta X_{n-1} + \epsilon_n \tag{7.23}$$

We take $X_0 = 0$. The transition kernel is $K(x, \cdot) = \mu_{\epsilon_{n+1}+\theta x}$.

Stop - Wed, 2/24

Let $K^n(\cdot, \cdot)$ be the function on $S \times \mathcal{S}$ given by

$$K^n(x, A) = P(X_n \in A | X_0 = x) \tag{7.24}$$

These are the n step transition kernels. We now consider the Chapman Komogorov equations.

Notation remark: Let $f(x)$ be a function on X and μ a measure on X. The integral of f with respect to μ is denoted in several ways:

$$\int_X f d\mu = \int_X f(x) d\mu = \int_X f(x) d\mu(x) = \int_X f(x) \mu(dx) \tag{7.25}$$

The last one is most commonly seen in probability rather than analysis.

Proposition 7 *Let m, n be positive integers and $A \in \mathcal{S}$. Then*

$$K^{n+m}(x, A) = \int_S K^n(y, A) K^m(x, dy) \tag{7.26}$$

for $x \in S$ and $A \in \mathcal{S}$.

The finite dimensional distributions are completely determined by μ and K. Let $A_0, A_1, \cdots, A_n \in \mathcal{S}$.

$$P(X_0 \in A_0) = \int_{A_0} K(X_0, A_1)\mu(dx_0) \tag{7.27}$$

$$P(X_0 \in A_0, X_1 \in A_1, \cdots, X_n \in A_n) = \int_{A_0} d\mu(x_0) \int_{A_1} K(x_0, dx_1) \tag{7.28}$$

$$\cdots \int_{A_n} K(x_{n-1}, dx_n) \tag{7.29}$$

Theorem 21 *(weak Markov property) Let $h(x_1, x_2, \cdots)$ be a reasonable function. Then for any initial distribution and any positive integer k,*

$$E_\mu[h(X_{k+1}, X_{x+2}, \cdots)|X_0, X_1, \cdots, X_k] = E_{X_k}[h(X_1, X_2, \cdots)] \tag{7.30}$$

Note that if we take $h(x_1, \cdots) = 1_{x_1 \in A}$ then the above equation becomes the definition of a Markov process.

The definition of irreducible for discrete state space does not work for general state space.

Definition 11 *Let ϕ be a non-zero measure on the state space. A Markov chain is ϕ-irreducible if for every $A \in \mathcal{S}$ with $\phi(A) > 0$ and every $x \in S$ there is a positive integer n such that $K^n(x, A) > 0$.*

Some caution with the meaning of this def. Consider a finite chain on $\{1, \cdots, n\}$ which is irreducible. Now add one more state $n+1$ but the only new transitions are from $n+1$ to $\{1, 2, \cdots, n\}$. So there are no transitions from the original n states to state $n+1$. This is not an irreducbile chain. $n+1$ is a transient state. But if $\phi(n+1) = 0$, this new chain is ϕ-irreducible.

Example: We return to the AR(1) example. Take the ϵ_n to be standard normal. Then we can take ϕ to be Lebesgue measure on the real line. Argue the example is ϕ irreducible. Now suppose ϵ_n is uniform on $[-1, 1]$ and $\theta > 1$. Argue the chain is not ϕ irreducible.

Now suppose $\theta > 1$ and ϵ_n is uniformly distributed on $[-1, 1]$. Start the chain at $X_0 = 0$. Argue it is not ϕ irreducible where ϕ is Lebesgue measure.

Now suppose ϵ_n is absolutely continuous with respect to Lebesgue measure and the density is positive everywhere. Then it is easy to see the chain is ϕ irreducible when ϕ is Lebesgue measure.

Given a set $A \in \mathcal{S}$ we let τ_A be the time the chain first enters A, i.e.,

$$\tau_A = \inf\{n \geq 1 : X_n \in A\} \tag{7.31}$$

And we let η_A be the number of times the chain visits A, i.e.,

$$\eta_A = \sum_{n=1}^{\infty} 1_A(X_n) \tag{7.32}$$

Note that η_A can be infinite.

Definition 12 *Let X_n be a ψ irreducible Markov chain. The chain is recurrent if for all $A \in \mathcal{S}$ with $\psi(A) > 0$ we have $E_x[\eta_A] = \infty$ for all $x \in A$.*

In the discrete case if a state is recurrent, then the probability we return to the state is 1. Once we have returned the probability we will return again is still 1, and so on. So with probability one we will return infinitely many times. So $\eta_A = 1$ with probability one. The above definition is weaker than this. In particular we will have $E_x[\eta_A] = \infty$ if $P_x(\eta_A = \infty)$ is non-zero but less than 1. To rule out some pathologies we will need a stronger notion of recurrent for our convergence theorems.

Definition 13 *The chain is Harris recurrent if there is a measure ψ such that for $A \in \mathcal{S}$ with $\psi(A) > 0$ and all $x \in A$ we have $P_x[\tau_A < \infty] = 1$ for all $x \in A$.*

Definition 14 *A σ-finite measure π is invariant for a Markov chain with transition kernel K such that*

$$\pi(B) = \int_S K(x, B)\pi(dx), \quad \forall B \in \mathcal{S} \tag{7.33}$$

(Note that we do not require that it be a probability measure or even that it is a finite measure). If there is an invariant measure which is a probability measure then we say the chain is positive recurrent.

Note: Robert and Casella say just "positive" instead of "positive recurrent."

Theorem 22 *Every recurrent chain has an invariant σ-finite measure. It is unique up to a multiplicative constant.*

Example: For random walk Lebesgue measure is an invariant measure.

Example: Consider the AR(1) example when the ϵ_n have a standard normal distribution. We look for a stationary distribution with a normal distribution with mean μ and variance σ^2. If X_n is $N(\mu, \sigma^2)$ then $X_{n+1} = \theta X_n + \epsilon_n$ is $N(\theta\mu, \theta^2\sigma^2 + 1)$. So it will be stationary only if $\mu = \theta\mu$ and $\sigma^2 = \theta^2\sigma^2 + 1$. This is possible only if $|\theta| < 1$, in which case $\mu = 0$ and $\sigma^2 = 1/(1 - \theta^2)$.

Proposition 8 *If the chain is positive recurrent then it is recurrent.*

A recurrent chain that is not positive recurrent is called null recurrent.

There is an analog of detailed balance if the transition kernel is given by a density, i.e., the state space is a subset of R^d and for all x

$$K(x, A) = \int_A k(x, y)\, dy \tag{7.34}$$

for $A \in \mathcal{S}$.

Definition 15 *A chain for which the transition kernel is given by a density satisfies detailed balance if there is a non-negative function $\pi(x)$ on S such that*

$$\pi(y)k(y, x) = \pi(x)k(x, y), \quad \forall x, y \in S \tag{7.35}$$

Proposition 9 *If the chain satisfies detailed balance then π is a stationary measure.*

Proof: Integrate the detailed balance equation over y with respect to Lebesgue measure. **QED**

As we already note there will be MCMC algorithms in which the transition kernel is not given by a density but is given by a lower dimensional density. There is an analog of detailed balance in this case.

Markov chains with continuous state space can still be periodic. We give a trivial example.

Example: Let $S = [0, 1] \cup [2, 3]$. Define $K(x, \cdot)$ to be the uniform measure on $[2, 3]$ if $x \in [0, 1]$ and the uniform measure on $[0, 1]$ if $x \in [2, 3]$. Clearly if we start the chain in $[0, 1]$, then after n steps it will be somewhere in $[0, 1]$ if n is even and somewhere in $[2, 3]$ if n is odd.

The definition of period for a general state space is a bit technical and we will skip it.

As in the previous section there are two types of convergence theorems.

Theorem 23 *If the chain is Haris positive and aperiodic then for every initial distbution μ,*

$$\lim_{n\to\infty} ||K^n(x,\cdot) - \pi||_{TV} = 0 \tag{7.36}$$

where $||\quad||_{TV}$ is the total variation norm.

Explain the total variation norm

Theorem 24 *(Ergodic theorem) Suppose that the Markov chain has an invariant measure π. Then the following two statements are equivalent.*
(i) The chain is Harris recurrent.
(ii) For all $f, g \in L^1(\pi)$ with $\int_S g(x)d\pi(x) \neq 0$ we have

$$\lim_{n\to\infty} \frac{\frac{1}{n}\sum_{k=1}^n f(X_k)}{\frac{1}{n}\sum_{k=1}^n g(X_k)} = \frac{\int_S f(x)d\pi(x)}{\int_S g(x)d\pi(x)} \tag{7.37}$$

Corollary If the Markov chain is Harris recurrent and has an invariant probability measure π, then for all $f \in L^1(\pi)$ we have

$$\lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^n f(X_k) = \int_S f(x)d\pi(x) \tag{7.38}$$

Chapter 8

Markov chain Monte Carlo

8.1 The key idea of MCMC

We start with a state space S and a probability density $\pi(x)$ on it. Our goal is to come up with a Markov chain on this state space that has $\pi(x)$ as its invariant distribution. If the chain is recurrent, then the ergodic theorem says that we can compute (approximately) the expected value of a function $F(x)$ on the state space by running the chain for along time and taking the long time average of $F(x)$ along the sequence of states that we generate. We start with two very simple examples to illustrate the idea of MCMC. One is discrete, one continuous.

Example: Fix an integer k and let S be the set of permutations with on $\{1, 2, \cdots, k\}$. Let π be the uniform measure on S. We want to construct a Markov chain on S with π as the stationary measure. (There are many ways to do this.) Our algorithm is as follows. We pick two integers $i, j \in \{1, 2, \cdots, k\}$. The choice is random with the uniform distribution on the set of k^2 possibilities. Let σ_{ij} be the permutation that interchanges i and j and leaves the other elements fixed. Then if σ is the state at time n, the state at time $n + 1$ is $\sigma_{ij} \circ \sigma$.

Show that it satisfies detailed balance.

Show it is irreducible.

Remark: This example illustrates the following observation. If $p(x, y)$ is symmetric, i.e., $p(x, y) = p(y, x)$, then the stationary distribution is the uniform distribution.

Example: The state space is the real line. Let $\pi(x)$ be the density of the standard normal.

We want to cook up a Markov chain with this as the stationary distribution. We take

$$k(x,y) = c(\sigma)\exp(-\frac{1}{2}(y-\frac{1}{2}x)^2/\sigma^2) \tag{8.1}$$

Show that $\pi(x)$ is the stationary distribution if $\sigma^2 = 3/4$.

8.2 The Metropolis-Hasting algorithm

We want to generate samples from a distribution

$$\pi(x) = \frac{1}{Z}p(x) \tag{8.2}$$

where $x \in X$. The set X could be a subset of R^d in which case $\pi(x)$ is a density and the measure we want to sample from is $\pi(x)$ times Lebesgue measure on R^d. Or X could be finite or countable in which case the distribution is discrete and the measure we want to sample from assigns probability $\pi(x)$ to x. The function $p(x)$ is known and Z is a constant which normalizes it to make it a probability distribution. Z may be unknown.

Let $q(x,y)$ be some transition function for a Markov chain with state space S. If S is discrete then $q(x,y)$ is a transition probability, while if S is continuous it is a transition probability density. We will refer to q as the proposal density or distribution. $q(x,y)$ is often written as $q(y|x)$. We assume that $q(x,y) = 0$ if and only if $q(y,x) = 0$ We define a function called the acceptance probability by

$$\alpha(x,y) = \min\{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\} \tag{8.3}$$

Since $\pi(y)/\pi(x) = p(y)/p(x)$, the possibly unknown constant Z is not needed to compute the acceptance probability. Note that $\alpha(x,y)$ is always in $[0,1]$. If one of the terms in the denominator above is zero, we define $\alpha(x,y)$ to be zero. It really doesn't matter how we define $\alpha(x,y)$ in this case. Explain.

Then we define a Monte Carlo chain as follows.

Metropolis-Hasting algorithm *Suppose the chain is in state X_n at time n. We generate Y from the distribution $q(y|X_n)$. Next generate U from the uniform distribution on $[0,1]$. If $U \leq \alpha(X_n, Y)$ then we set $X_{n+1} = Y$. Otherwise we set $X_{n+1} = X_n$.*

If the proposal distribution is symmetric, meaning that $q(y,x) = q(x,y)$, then the acceptance probability function simplifies to

$$\alpha(x,y) = \min\{\frac{\pi(y)}{\pi(x)}, 1\} \tag{8.4}$$

This is sometimes called just the Metropolis algorithm. This case was studied by Metropolis. Hasting generalized to non-symmetric q.

The above description of the algorithm implicitly defines the transition kernel for the Markov chain. We make it more explicit. In the discrete case for $x \neq y$ the transition probability is

$$p(x, y) = q(x, y)\alpha(x, y) \tag{8.5}$$

and

$$p(x, x) = q(x, x)\alpha(x, x) + \sum_y [1 - \alpha(x, y)]q(x, y) \tag{8.6}$$

The first term comes from accepting the proposed state x and the second term (with the sum on y) comes from proposing y and rejecting it.

In the continuous case the transition kernel $K(x, \cdot)$ is a mixture of a continuous measure and a point mass.

$$K(x, A) = \int_A \alpha(x, y)q(x, y)\, dy + 1_{x\in A} \int_S [1 - \alpha(x, y)]q(x, y)\, dy \tag{8.7}$$

Remark: This sort of looks like acceptance-rejection. We generate a proposed state Y and accept it with probability $\alpha(X_n, Y)$, reject it otherwise. But it is not the same as the acceptance-rejection algorithm. One crucial difference is that in the acceptance-rejection algorithm when we reject a proposed value the number of samples does not increase. In Metropolis-Hasting when we reject Y the chain still takes a time step. When this happens there are two consecutive states in $X_0, X_1, X_2, \cdots$ that are the same. So in the time average

$$\frac{1}{n} \sum_{k=1}^{n} f(X_k) \tag{8.8}$$

there can be terms that are the same.

Theorem 25 *$\pi(x)$ is the stationary distribution of the Markov chain of the Metropolis-Hasting algorithm.*

Proof: We will eventually consider the discrete and continuous cases separately, but first we prove the following crucial identity which holds in both cases.

$$\pi(x)\alpha(x, y)q(x, y) = \pi(y)\alpha(y, x)q(y, x), \quad \forall x, y \in S \tag{8.9}$$

To prove it we consider two cases: $\alpha(x,y) = 1$ and $\alpha(y,x) = 1$. (It is possible these cases overlap, but that does not affect the proof.) The cases are identical. So we assume $\alpha(x,y) = 1$. Then

$$\alpha(y,x) = \frac{\pi(x)q(x,y)}{\pi(y)q(y,x)} \tag{8.10}$$

The above identity follows.

Now consider the discrete case, i.e., the state space is finite or countable. In this case $\pi(x)$ is a probability mass function. The transition function is

$$p(x,y) = \alpha(x,y)q(x,y) \tag{8.11}$$

We prove that $\pi(x)$ is the stationary distribution by showing it satisfies detailed balance. Let x and y be distinct states. (If $x = y$ it is trivial to verify the detailed balance equation.) So we must show $\pi(x)p(x,y) = \pi(y)p(y,x)$. This is immediate from (8.9).

Now consider the continuous case. So the state space is a subspace of R^d and $\pi(x)$ is a density with respect to Lebesgue measure on R^d. Note that the transition kernel is now a mix of a continuous and discrete measure. So trying to use detailed balance is problematic. We just verify the stationary equation. So let $A \subset R^d$. We must show

$$\int_A \pi(x)dx = \int K(x,A)\pi(x)dx \tag{8.12}$$

The right side is the sum of two terms - one is from when we accept the proposed new state and one from when we reject it. The acceptance term is

$$\int \left[\int_A \alpha(x,y)q(x,y)dy\right] \pi(x)dx \tag{8.13}$$

Given that we are in state x and that the proposed state is y, the probability of rejecting the proposed state is $1 - \alpha(x,y)$. So the probability we stay in x given that we are in x is

$$\int [1 - \alpha(x,y)]q(x,y)dy \tag{8.14}$$

So the rejection term is

$$\int_A \pi(x) \left[\int [1 - \alpha(x,y)]q(x,y)dy\right] dx \tag{8.15}$$

Note that the integral over x is only over A since when we reject we stay in the same state. So the only way to end up in A is to have started in A. Since $\int q(x,y)dy = 1$, the above equals

$$\int_A \pi(x)dx - \int_A \pi(x) \left[\int \alpha(x,y)]q(x,y)dy\right] dx \tag{8.16}$$

So we need to show

$$\int \left[\int_A \alpha(x,y) q(x,y) dy \right] \pi(x) dx = \int_A \pi(x) \left[\int \alpha(x,y)] q(x,y) dy \right] dx \tag{8.17}$$

In the right side we do a change of variables to interchange x and y. So we need to show

$$\int \left[\int_A \alpha(x,y) q(x,y) dy \right] \pi(x) dx = \int_A \pi(y) \left[\int \alpha(y,x)] q(y,x) dx \right] dy \tag{8.18}$$

If we integrate (8.9) over $x \in X$ and $y \in A$ we get the above. **QED**

Stop - Mon, 3/9

To apply our convergence theorem for Markov chains we need to know that the chain is irreducible and if the state space is continuous that it is Harris recurrent.

Consider the discrete case. We can assume that $\pi(x) > 0$ for all x. (Any states with $\pi(x) = 0$ can be deleted from the state space.) Given states x and y we need to show there are states $x = x_0, x_1, \cdots, x_{n-1}, x_n = y$ such that $\alpha(x_i, x_{i+1}) q(x_i, x_{i+1}) > 0$. If $q(x_i, x_{i+1}) > 0$ then $\alpha(x_i, x_{i+1}) > 0$. So it is enough to find states so that $q(x_i, x_{i+1}) > 0$. In other words we need to check that the transition function $q(x,y)$ is irreducible.

Now consider the continuous case. We want to show that the chain is π-irreducible. We start with a trivial observation. If $q(x,y) > 0$ for all x, y, then the chain is π-irreducible since for any x and any set A with $\int_A \pi(x) dx > 0$, the probability that if we start in x and reach A in just one step will be non-zero. This condition is too restrictive for many cases. Here is a more general sufficient condition.

Proposition 10 *Suppose that the state space S is connected in the following sense. Given $\delta > 0$ and $x, y \in S$ there exists states $y_0 = x, y_1, \cdots, y_{n-1}, y_n = y$ such that $|y_i - y_{i-1}| < \delta$ for $i = 1, 2, \cdots, n$ and the sets $B_\delta(y_i) \cap S$ have non-zero Lebesgue measure for $i = 0, 1, 2, \cdots, n$. Assume there is an $\epsilon > 0$ such that $|x - y| < \epsilon$ implies $q(x,y) > 0$. Then the Metropolis-Hasting chain is irreducible with respect to Lebesgue measure on S.*

Proof: Let $x_0 \in S$ and let $A \subset S$ have non-zero Lebesgue measure. Pick $y_n \in A$ such that the set $B_\delta(y_n) \cap A$ has non-zero Lebesgue measure. (This is possible since A has non-zero Lebesgue measure.) Let $y_1, y_2, \cdots, y_{n-1}$ be states as in the above sense of connectedness with

$\delta = \epsilon/3$. We will show $K^n(x_0, A) > 0$. We do this by only considering trajectories $x_0, x_1, x_2, \cdots, x_n$ such that $|x_i - y_i| < \delta$ for $i = 1, \cdots, n$. We further require $x_n \in A$. And finally we only consider trajectories for which all the proposed jumps were accepted. The probability of this set of trajectories is given by the integral of

$$q(x_0, x_1)\alpha(x_0, x_1)q(x_1, x_2)\alpha(x_1, x_2)\cdots q(x_{n-1}, x_n)\alpha(x_{n-1}, x_n) \tag{8.19}$$

where the region of integration is given by the constraints $|x_i - y_i| < \delta$ for $i = 1, 2, \cdots, n-1$ and $x_n \in B_\delta(y_n) \cap A$. Since $|y_{i-1} - y_i| < \delta$ the triangle inequality implies $|x_{i-1} - x_i| < 3\delta = \epsilon$. So we have $q(x_{i-1}, x_i) > 0$. Note that $q(x_{i-1}, x_i) > 0$ implies $\alpha(x_{i-1}, x_i) > 0$. So the integrand is strictly positive in the integral. The integral is over a set of non-zero Legesgue measure, so the integral is non-zero. **QED**

Finally we have

Proposition 11 *If the Metropolis-Harris chain is π-irreducible then it is Harris recurrent.*

A proof can be found in Robert and Casella. This is lemma 7.3 in their book in section 7.3.2.

We do not need to know the chain is aperiodic for our main use of it, but it is worth considering. If $U > \alpha(X_n, Y)$ then we stay in the same state. This happens with probability $1 - \alpha(X_n, Y)$. So as long as $\alpha(x, y) < 1$ on a set with non-zero probability (meaning what ???), the chain will be aperiodic. If $\alpha(x, y) = 1$ for all x, y, then $\pi(y)q(y, x) = \pi(x)q(x, y)$. But this just says that π satisfies detailed balance for the transition function q. So we would not be doing Metropolis-Hasting anyway. In this case we would need to study q to see if it was aperiodic.

Example (normal distribution): Want to generate samples of standard normal. Given $X_n = x$, the proposal distribution is the uniform distribution on $[x - 1, x + 1]$. So

$$q(x, y) = \begin{cases} \frac{1}{2} & \text{if } |x - y| \leq 1, \\ 0, & \text{if } |x - y| > 1, \end{cases} \tag{8.20}$$

We have

$$\alpha(x, y) = \min\{\frac{\pi(y)}{\pi(x)}, 1\} = \frac{1}{2}\min\{\exp(-\frac{1}{2}y^2 + \frac{1}{2}x^2), 1\} \tag{8.21}$$

$$= \frac{1}{2}\begin{cases} \exp(-\frac{1}{2}y^2 + \frac{1}{2}x^2) & \text{if } |x| < |y|, \\ 1 & \text{if } |x| \geq |y| \end{cases} \tag{8.22}$$

Example (permutations): Consider permutations σ of $\{1, 2, \cdots, k\}$. A permutation is a bijective function on $\{1, 2, \cdots, k\}$. Instead of considering the uniform distribution on the set

of permutations as we did in an earlier example we consider a general probability measure. We write it in the form

$$\pi(\sigma) = \frac{1}{Z}\exp(w(\sigma)) \tag{8.23}$$

$w(\sigma)$ can be any function on permutation and can take on positive and negative values. An example of a possible w is the following. Let $s(\sigma)$ be the number of elements that are fixed by σ. Then let $w(\sigma) = \alpha s(\sigma)$. So depending on the sign of α we either favor or disfavor permutations that fix a lot of elements. The proposal distribution is to choose two distinct i, j uniformly and multiply the current permuation by the tranposition (i, j).

$$\alpha(\sigma, \sigma') = \min\{\frac{\pi(\sigma')}{\pi(\sigma)}, 1\} = \min\{\exp(w(\sigma') - w(\sigma)), 1\} \tag{8.24}$$

$$\tag{8.25}$$

Note that we only need to compute the change in $w()$. For "local" w this is a relatively cheap computation.

Example (Ising model): Fix a finite subset Λ of the lattice Z^d. At each site $i \in \Lambda$ there is a "spin" σ_i which takes on the values ± 1. The collection $\sigma = \{\sigma_i\}_{i\in\Lambda}$ is called a spin configuration and is a state for our system. The state space is $\{-1, 1\}^\Lambda$. The Hamiltonian $H(\sigma)$ is a function of configurations. The simplest H is the nearest neighbor H:

$$H(\sigma) = \sum_{<ij>} \sigma_i \sigma_j \tag{8.26}$$

We then define a probability measure on the spin configurations by

$$\pi(\sigma) = \frac{1}{Z}\exp(-\beta H(\sigma)) \tag{8.27}$$

The proposal distribution is defined as follows. We pick a site i uniformly from Λ. Then we flip the spin at i, i.e., we replace σ_i by $-\sigma_i$. So we only propose transitions between configurations that only only differ in one site. So $q(\sigma, \sigma') = 1/|\Lambda|$ when the spin configurations differ at exactly on site and it is zero otherwise. For two such configurations σ and σ' the acceptance probability is

$$\alpha(\sigma, \sigma') = \min\{\frac{\pi(\sigma')}{\pi(\sigma)}, 1\} = \min\{\exp(-\beta[H(\sigma') - H(\sigma)], 1\} \tag{8.28}$$

Note that there is lots on cancellation in the differece of the two Hamiltonians. This computation takes a time that does not depend on the size of Λ.

Example: QFT

8.3 The independence sampler

The independence sampler is a special case of the Metropolis-Hasting algorithm. In the independence sampler the proposal distribution does not depend on x, i.e., $q(x,y) = g(y)$. So the acceptance probability becomes

$$\alpha(x,y) = \min\{\frac{\pi(y)g(x)}{\pi(x)g(y)}, 1\} \tag{8.29}$$

Suppose that there is a constant C such that $\pi(x) \leq Cg(x)$. In this setting we could do the acceptance-rejection algorithm. It will generate independent samples of $\pi(x)$ and the acceptance rate will be $1/C$. By contrast the independence sampler will generate dependent samples of $\pi(x)$.

Stop - Wed, 3/9

Proposition 12 *Consider the independence sampler with proposal distribution $g(x)$ and stationary distribution $\pi(x)$. Suppose there is a constant C such that $\pi(x) \leq Cg(x)$ for all $x \in S$. Let $\pi_0(x)$ be any initial distibution and let $\pi_n(x)$ be the distribution at time n. Then*

$$||\pi_n - \pi||_{TV} \leq 2(1 - \frac{1}{C})^n \tag{8.30}$$

Proof: We will only consider the case that the initial distribution is absolutely continuous with respect to Lebesgue measure. So the initial distribution is $\pi(x)dx$. Note that in this case the subsequent distributions π_n will be absolutely continuous with respect to Lebesgue measure. **Explain this.**

For convenience let $\epsilon = \frac{1}{C}$. So our bound can be rewritten as $g(x) \geq \epsilon\pi(x)$. Since $q(x,y) = g(y)$, we have

$$\begin{aligned} \alpha(x,y)q(x,y) &= \min\{\frac{\pi(y)g(x)}{\pi(x)g(y)}, 1\}g(y) = \min\{\frac{\pi(y)g(x)}{\pi(x)}, g(y)\} & (8.31) \\ &\geq \min\{\frac{\pi(y)\epsilon\pi(x)}{\pi(x)}, \epsilon\pi(y)\} = \epsilon\pi(y) & (8.32) \end{aligned}$$

Let $\rho(x)$ be a probability density. The transition kernel takes it to another density, and we will denote this new density by $K\rho$. So K is a linear operator on integrable functions on R^d.

Let P be the linear operator which maps a probability density $\rho(x)$ to the probability density $\pi(x)$. So P is a projection. For a general integrable function

$$(P\rho)(x) = \pi(x) \int_X \rho(y) dy \tag{8.33}$$

Our previous bound shows that for any probability density ρ, $K\rho - \epsilon P\rho$ is a non-negative function. Its integral is $1 - \epsilon$. So if we define another linear operator by

$$R = \frac{1}{1-\epsilon}[K - \epsilon P] \tag{8.34}$$

then R will map a probability density into another probability density. Note that $K = \epsilon P + (1-\epsilon)R$. A straightforward induction argument shows that

$$K^n = \sum_{k=1}^{n} K^{n-k} P (1-\epsilon)^{k-1} R^{k-1} + (1-\epsilon)^n R^n \tag{8.35}$$

Note that $P\rho = \pi$ for any probability distribution ρ, and $K\pi = \pi$. So $K^j P\rho = \pi$ for any j. So for any initial distribution π_0,

$$\pi_n = K^n \pi_0 = \pi \sum_{k=1}^{n} (1-\epsilon)^{k-1} + (1-\epsilon)^n R^n \pi_0 = [1 - (1-\epsilon)^n]\pi + (1-\epsilon)^n R^n \pi_0 \tag{8.36}$$

So

$$\pi_n - \pi = -(1-\epsilon)^n \pi + (1-\epsilon)^n R^n \pi_0 \tag{8.37}$$

Since $R^n \pi_0$ is a probability density, the L^1 norm of the above is bounded by $2(1-\epsilon)^n$. **QED**

8.4 The Gibbs sampler

In this section change notation and let $f(x)$ denote the distribution we want to sample from, in place of our previous notation $\pi(x)$.

We first consider the two-stage Gibbs sampler. We assume that the elements of the state space are of the form (x, y). The probability distribution we want to sample from is $f(x, y)$. Recall that the marginal distributions of X and Y are given by

$$f_X(x) = \int f(x,y)\, dy, \quad f_Y(y) = \int f(x,y)\, dx \tag{8.38}$$

and the conditional distributions of X given Y and Y given X are

$$f_{Y|X}(y|x) = \frac{f(x,y)}{f_X(x)}, \quad f_{X|Y}(x|y) = \frac{f(x,y)}{f_Y(y)} \tag{8.39}$$

Note that if we only know $f(x,y)$ up to an overall constant, we can still compute $f_{Y|X}(y|x)$ and $f_{X|Y}(x|y)$.

The two-stage Gibbs sampler *Given that we are in state (X_n, Y_n), we first generate Y_{n+1} from the distribution $f_{Y|X}(\cdot|X_n)$. Then we generate X_{n+1} from the distribution $f_{X|Y}(\cdot|Y_{n+1})$. These two stages make up one time step for the Markov chain. So the transition kernel is*

$$K(x,y;x',y') = f_{Y|X}(y'|x) f_{X|Y}(x'|y') \tag{8.40}$$

Proposition 13 *$f(x,y)$ is the stationary distribution of the two-stage Gibbs sampler.*

Proof: We just show that $Kf = f$.

$$\begin{aligned}
(Kf)(x',y') &= \int\int f(x,y) K(x,y;x',y') dxdy && (8.41)\\
&= \int\int f(x,y) f_{Y|X}(y'|x) f_{X|Y}(x'|y') dxdy && (8.42)\\
&= \int\int f(x,y) \frac{f(x,y')}{f_X(x)} \frac{f(x',y')}{f_Y(y')} dxdy && (8.43)\\
&= \int f_X(x) \frac{f(x,y')}{f_X(x)} \frac{f(x',y')}{f_Y(y')} dx && (8.44)\\
&= \int f(x,y') \frac{f(x',y')}{f_Y(y')} dx && (8.45)\\
&= f(x',y') && (8.46)
\end{aligned}$$

QED

Remark: The two-stage Gibbs sampler does not satisfy detailed balance in general.

Example: (Bivariate normal) We consider the bivariate normal (X,Y) with joint density

$$f(x,y) = c \exp(-\frac{1}{2}x^2 - \frac{1}{2}y^2 - \alpha xy) \tag{8.47}$$

where α is a parameter related to the correlation of X and Y. Argue that

$$f_{Y|X}(y|x) = c(x) \exp(-\frac{1}{2}(y+\alpha x)^2) \tag{8.48}$$

So for the first stage in the Gibbs sampler, we generate Y_{n+1} from a standard normal distribution with mean $-\alpha X_n$. We have

$$f_{X|Y}(x|y) = c(y) \exp(-\frac{1}{2}(x+\alpha y)^2) \tag{8.49}$$

So for the second stage, we generate X_{n+1} from a standard normal distribution with mean $-\alpha Y_{n+1}$.

We now consider the multi-stage Gibbs sampler. Now suppose that the points in the state space are of the form $x = (x_1, x_2, \cdots, x_d)$. We need to consider the conditional distribuion of X_i given all the other X_j. To keep the notation under control we will write

$$f_{X_i|X_1,\cdots,X_{i-1},X_{i+1},\cdots,X_d}(x_i|x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_d) = f_i(x_i|x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_d) \quad (8.50)$$

Again we emphasize that we can compute these conditional distributions even if we only know $f(x_1,\cdots,x_d)$ up to an unknown constant.

Multi-stage Gibbs sampler: *The algorithm has d stages and proceeds as follows.*
(1) Given $(X_1^n,\cdots,X_d^n)$ *we sample* X_1^{n+1} *from* $f_1(\cdot|X_2^n,\cdots,X_d^n)$.
(2) Then we sample X_2^{n+1} *from* $f_2(\cdot|X_1^{n+1},X_3^n,\cdots,X_d^n)$.
(j) Continuing we sample X_j^{n+1} *from* $f_j(\cdot|X_1^{n+1},\cdot,X_{j-1}^{n+1},X_{j+1}^n,\cdots,X_d^n$.
(p) In the last step we sample X_d^{n+1} *from* $f_d(\cdot|X_1^{n+1},\cdots X_{d-1}^{n+1})$.

Before we show that the stationary distribution of this algorithm is f, we consider some variations of the algorithm. Let K_j be the transition kernel corresponding to the jth step of the multi-stage Gibbs sampler. So

$$K_j(x_1,x_2,\cdots,x_p;x_1',x_2',\cdots,x_d') = f_j(x_j'|x_1,x_2,\cdots,x_{j-1},x_{j+1},\cdots,x_p)\prod_{i=1:i\neq j}^{p}\delta(x_i-x_i') \quad (8.51)$$

If we think of K_j as a linear operator, then the multi-stage Gibbs sampler is $K_dK_{d-1}\cdots K_2K_1$.

Here is another Gibbs sampler which for lack of a standard name we will call the randomized Gibbs sampler. Fix some probability distribution p_i on $\{1,2,\cdots,d\}$. Given that we are in state $(X_1^n,\cdots,X_d^n)$, we first pick $i\in\{1,2,\cdots,d\}$ according to this distribution. Then we sample X_i^{n+1} from $f_i(\cdot|X_1^n,\cdot,X_{i-1}^n,X_{i+1}^n,\cdots,X_d^n)$. For $l\neq i$, $X_l^{n+1}=X_l^n$. The transition kernel for this algorithm is

$$K = \sum_{i=1}^{d} p_iK_i \quad (8.52)$$

Proposition 14 *$f(x_1,x_2,\cdots,x_d)$ is the stationary distribution of the multi-stage Gibbs sampler and of the randomized Gibbs sample for any choice of the distribution p_i.*

Stop - Mon, 3/21

Proof: We only need to show that for all j, $K_j f = f$. So we compute:

$$
\begin{aligned}
& (K_j f)(x'_1, x'_2, \cdots, x'_d) && (8.53)\\
= & \int \cdots \int f(x_1, x_2, \cdots, x_d) K_j(x_1, x_2, \cdots, x_d; x'_1, x'_2, \cdots, x'_d) dx_1 dx_2 \cdots dx_d && (8.54)\\
= & \int \cdots \int f(x_1, x_2, \cdots, x_d) f_j(x'_j | x_1, x_2, \cdots, x_{j-1}, x_{j+1}, \cdots, x_d) \prod_{i=1:i\neq j}^{d} \delta(x_i - x'_i) && (8.55)\\
= & f_j(x'_j | x'_1, x'_2, \cdots, x'_{j-1}, x'_{j+1}, \cdots, x'_d) \int f(x'_1, x'_2, \cdots, x'_{j-1}, x_j, x'_{j+1}, x'_d) dx_j && (8.56)\\
= & f(x'_1, x'_2, \cdots, x'_d) && (8.57)
\end{aligned}
$$

Note that the random stage Gibbs sampler has f as the stationary distrbution for any choice of the p_i. We need to take all $p_i > 0$ to have any chance that the chain is irreducible. The simplest choice for the p_i is to use the uniform distribution on $\{1, 2, \cdots, d\}$. Why would we do anything else? In the following example we will see a case where we might want to use a non-uniform distribution.

Caution: There are lots of variations on the Gibbs sampler, but one should be careful. Here is one that does not work.

WRONG two-stage Gibbs sampler: Given that we are in state (X_n, Y_n), we first generate Y_{n+1} from the distribution $f_{Y|X}(\cdot|X_n)$. Then we generate X_{n+1} from the distribution $f_{X|Y}(\cdot|Y_n)$. These two stages make up one time step for the Markov chain. So the transition kernel is

$$K(x, y; x', y') = f_{Y|X}(y'|x) f_{X|Y}(x'|y) \tag{8.58}$$

Note that the difference with the correct two-stage Gibbs sampler is that we we generate X_{n+1} from $f_{X|Y}(\cdot|Y_n)$ rather than $f_{X|Y}(\cdot|Y_{n+1})$.

Here is an example to illustrate how the above algorithm is wrong. Take $f(x, y)$ to be the uniform distribution on the three points $(0, 0), (0, 1), (1, 0)$. Explain this.

Remark: The d-stage Gibbs sampler requires that the states have the structure $(x_1, x_2, \cdots, x_d)$. However this does mean that the state space has to be a subset of R^d. Some of the x_i could be vectors or even something stranger.

Example: We consider the Ising model that we considered in a previous example. The integer d is not the number of dimensions. It is the number of sites in Λ. For $j \in \Lambda$, f_j is the

conditional distribution of σ_j given the values of all the other spins. We compute this in the usual way (joint density over marginal) to get

$$f_j(\sigma_j|\sigma_{\Lambda\setminus j}) = \frac{\exp(-\beta H(\sigma))}{\sum_{s_j}\exp(-\beta H(\hat{\sigma}))} \tag{8.59}$$

where s_j is summed over just $-1, 1$ and $\hat{\sigma}$ equals σ_i for all sites $i \neq j$ and equals s_j at site j. The algorithm applies to any H, but there are some nice cancellations if H is "local." We illustrate this by considering the nearest neighbor H. Any term in H that does not involve site j cancels in the numerator and the denominator. The result is just

$$f_j(\sigma_j|\sigma_{\Lambda\setminus j}) = \frac{\exp(-\beta\sigma_j \sum_{k:|k-j|=1}\sigma_k)}{\exp(-\beta\sum_{k:|k-j|=1}\sigma_k) + \exp(\beta\sum_{k:|k-j|=1}\sigma_k)} \tag{8.60}$$

So computing f_j takes a time that is $O(1)$, independent of the size of Λ. But just how fast the algorithm mixes depends very much on the size of Λ and on β. For the multi-stage algorithm each time steps take a time of order $|\Lambda|$. For the random-stage algorithm each time step only takes a time $O(1)$, but it will take $O(|\Lambda|)$ times steps before we have changed a significant fraction of the spins.

Now suppose we want to compute the expected value of $F(\sigma)$ in the Ising model and $F(\sigma)$ only depends on a few spins near the center of Λ. Then we may want to choose the distribution p_i so that the sites near the center have higher probability than the sites that are not near the center.

Remark: As the example above shows, d is not always the "dimension" of the model.

Example: (loosely based on example 6.6 in Rubenstein and Kroese, p. 177) For $i = 1, 2, \cdots, d$, let $p_i(x_i)$ be a discrete probability function on the non-negative integers. If $X_1, X_2, \cdots, X_d$ were independent with these distributions, then the joint distribution would be just the product of the $p_i(x_i)$. This is trivial to simulate. We are interested in something else. Fix a positive integer m. We restrict the sample sample to the d-tuples of non-negative integers $x_1, x_2, \cdots, x_d$ such that $\sum_{i=1}^d x_i = m$. We can think of this as the conditional distribution of $X_1, \cdots, X_d$ given that $\sum_i X_i = m$. So we want to simulate the joint pdf given by

$$f(x_1, \cdots, x_d) = \frac{1}{Z}\prod_{i=1}^d p_i(x_i) \tag{8.61}$$

when $\sum x_i = m$ and $f() = 0$ otherwise. The constant Z is defined by ... Since $X_d = m - \sum_{i=1}^{d-1} X_i$, we can work with just $X_1, X_2, \cdots, X_{d-1}$. Their joint distribution is

$$f(x_1, \cdots, x_{d-1}) = \frac{1}{Z}\, p_d(m - \sum_{i=1}^{d-1} x_i)\, \prod_{i=1}^{d-1} p_i(x_i) \tag{8.62}$$

for $x_1, \cdots, x_{d-1}$ whose sum is less than or equal to m. Then their sum is greater than m, $f(x_1, \cdots, x_{d-1}) = 0$. All we need to run the Gibbs sampler are the conditional distributions of X_j given the other X_i. They are given by

$$f_j(x_j|x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_{d-1}) \propto f_j(x_j) f_d(m - \sum_{i=1}^{d-1} x_i) \tag{8.63}$$

If we let $m' = m - \sum_{i=1:i\neq j}^{d-1} x_i$, then the right side is equal to $f_j(x_j) f_d(m' - x_j)$. We need to compute the constant to normalize this, but that takes only a single sum on x_j. (And for some f_j, f_d can be done explicitly.)

Irreducibility: brief discussion of irreducibility for the Gibbs sample. Gap in notes here.

8.5 Slice sampler

The slice sampler is in some sense a special case of the Gibbs sampler. Suppose we want to sample from $f(x)$ where x ranges over X. We consider a new distribution: the new state space is a subspace of $X \times R$, namely,

$$S = \{(x, u) : 0 \leq u \leq f(x)\} \tag{8.64}$$

This can be thought of as the area under the graph of f. The new distribution is the uniform measure on S. The key observation is that with this distribution on (X, U), the marginal distribution of X is $f(x)$. So if we can construct a Markov chain (X_n, U_n) with the uniform measure on S as its stationary measure then we can just look at X_n and long time averages of random variables on X will converge to their expectation with respect to $f(x)$. We use the two-stage Gibbs sampler. So we need the conditional distributions $f_{U|X}(u|x)$ and $f_{X|U}(x|u)$. They are both uniform. More precisely, the distribution of U given $X = x$ is uniform on $[0, f(x)]$, and the distribution of X given $U = u$ is uniform on $\{x : u \leq f(x)\}$.

Slice sampler (single slice) *Given that we are in state (X_n, U_n), we first generate U_{n+1} from the uniform distribution on $[0, f(X_n)]$. Then we generate X_{n+1} from the uniform distribution on $\{x : U_{n+1} \leq f(x)\}$.*

Remark: Suppose that the density we wish to simulate is given by $cf(x)$ where c is an unknown constant. We can still take

$$S = \{(x, u) : 0 \leq u \leq f(x)\} \tag{8.65}$$

and put the uniform distribution on S. The density function is $\frac{1}{c} 1_S$. The constant c is unknown, but that will not matter. The marginal density of X is still $f(x)$.

Example: Let $f(x) = ce^{-x^2/2}$, the standard normal. Given (X_n, U_n) it is trivial to sample U_{n+1} uniformly from $[0, f(X_n)]$. Next we need to sample X_{n+1} uniformly from $\{x : U_{n+1} \leq f(x)\}$. This set is just the interval $[-a, a]$ where a is given by $U_{n+1} = f(a)$. We can trivially solve for a.

The first step of the slice sampler, generating U_{n+1}, is always easy. The second step, generating X_{n+1}, may not be feasible at all since the set $\{x : U_{n+1} \leq f(x)\}$ may be very complicated. For example suppose we want to sample

$$f(x) = c\frac{1}{1+x^2}\exp(-x^2/2) \tag{8.66}$$

The set will be an interval, but finding the endpoints requires solving an equation like $\exp(-x^2/2)/(1+x^2) = u$. This could be done numerically, but the set could be even more complicated. There is a generalization that may work even when this second step is not feasible for the single slice sampler.

Assume that $f(x)$ can be written in the form

$$f(x) = \prod_{i=1}^{d} f_i(x) \tag{8.67}$$

where the $f_i(x)$ are non-negative but need not be probability densities. We then introduce a new random variable (sometimes called auxillary variables) for each f_i. So the new state space is a subspace of $X \times R^d$ and is given by

$$S = \{(x, u_1, u_2, \cdots, u_d) : 0 \leq u_i \leq f_i(x),\ i = 1, 2, \cdots, d\} \tag{8.68}$$

We use the uniform distribution on S. The key observation is that if we integrate out $u_1, u_2, \cdots, u_d$, we just get $f(x)$. So the marginal distribution of X will be $f(x)$. For the Markov chain we use the $d+1$ dimensional Gibbs sample.

Example: Let

$$f(x) = c\frac{1}{1+x^2}\exp(-x^2/2) \tag{8.69}$$

Let

$$f_1(x) = \frac{1}{1+x^2}, \quad f_2(x) = \exp(-x^2/2) \tag{8.70}$$

Note that we are dropping the c. We sample U_1^{n+1} uniformly from $[0, f_1(X^n)]$. Then we sample U_2^{n+1} uniformly from $[0, f_2(X^n)]$. Finally we need to sample X^{n+1} uniformly from

$$\{x : U_1^{n+1} \leq f_1(x),\ U_2^{n+1} \leq f_2(x)\} \tag{8.71}$$

This set is just an interval with endpoints that are easily computed.

Stop - Wed, March 23

The slice sampler can be used when the initial distribution $f(x)$ is discrete as the next example shows.

Example: Consider the density $f(x) = c\exp(-\alpha x^2)$ where $\alpha > 0$ and $x = 0, 1, 2, \cdots$. Note that the constant c cannot be computed analytically. We would have to compute it numerically. For the slice sampler we can just drop c. The first stage is to generate U_{n+1} uniformly from $[0, f(X_n)]$. Then we generate X_{n+1} uniformly from the set of non-negative integers k such that $U_{n+1} \leq f(k)$.

8.6 Bayesian statistics and MCMC

We start with a triviality which is often called Bayes rule. Given two random variables (which can be random vectors), we have

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}, \quad f_{Y|X}(y|x) = \frac{f_{X,Y}(x,y)}{f_X(x)} \tag{8.72}$$

So

$$f_{Y|X}(y|x) = \frac{f_{X|Y}(x|y)f_Y(y)}{f_X(x)} \tag{8.73}$$

This is often written as

$$f_{Y|X}(y|x) \propto f_{X|Y}(x|y)f_Y(y) \tag{8.74}$$

with the understanding that the constant of proportionality depends on x. In Bayesian statistics it is often written in the more abbreviated form

$$f(y|x) \propto f(x|y)f(y) \tag{8.75}$$

"This particular style of notation is typical in Bayesian analysis and can be of great descriptive value, despite its apparent ambiguity" - Rubinstein and Kroese.

Now suppose we have a probability distribution for x, which is typically a vector, that depends on some parameters $\theta = (\theta_1, \cdots, \theta_d)$. Often the vector x is a sample $x_1, x_2, \cdots, x_n$ that comes from performing some experiment n times. We don't know θ. In statistics we want to use the value of x that results from our experiment to estimate the unknown parameters θ. The Bayesian statistician puts a probability distribution on θ, $f(\theta)$, that is supposed to encode all the information we have about how likely we think different values of θ are **before** we do the experiment. $f(\theta)$ is called the *prior distribution*. Now we do the experiment, and so we have a particular value for x. We want to replace the prior distribution on θ by a distribution that incorporates the knowledge of x. The natural distribution is $f(\theta|x)$. This is called the *posterior* distribution of θ. By Bayes rule

$$f(\theta|x) \propto f(x|\theta)f(\theta) \tag{8.76}$$

where the constant of proportionality depends on x. The conditional density $f(x|\theta)$ is called the *likelihood*. We typically know this function quite explicitly. For example, if $f(x|\theta)$ comes from independent repetitions of the same experiment, then

$$f(x|\theta) = f(x_1, x_2, \cdots, x_n|\theta) = \prod_{i=1}^{n} f_X(x_i|\theta) \tag{8.77}$$

where $f_X(x|\theta)$ is the distribution of X for one performance of the experiment. So Bayes rule says

$$f(\theta|x) \propto \left[\prod_{i=1}^{n} f_X(x_i|\theta)\right] f(\theta) \tag{8.78}$$

Given the data x this gives the joint distribution of the parameters $\theta_1, \cdots, \theta_d$. To run a Gibbs sampler we need the conditional distribution of each θ_i given the other θ_j, $j \neq i$. The constant of proportionality in Bayes rule is often impossible to compute analytically, but this does not matter for the Gibbs sampler.

Example : We have a coin with probability θ of getting heads. However, we do not know θ. We flip it n times, let $X_1, X_2, \cdots, X_n$ be 1 for heads, 0 for tails. If we are given a value for θ, then the distribution of $X_1, X_2, \cdots, X_n$ is just

$$f(x|\theta) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} = \theta^s(1-\theta)^{n-s} \tag{8.79}$$

where x is short for $x_1, x_2, \cdots, x_n$ and s is defined to be $\sum_{i=1}^{n} x_i$. If we have no idea what θ is, a reasonable choice for the prior distribution for θ is to make it uniform on $[0, 1]$. Now suppose we flip the coin n times and use the resulting "data" $x_1, \cdots, x_n$ to find a better distribution for θ that incorporates this new information, i.e., find the posterior distribution. The posterior is given by

$$f(\theta|x) \propto f(x|\theta)f(\theta) = \theta^s(1-\theta)^{n-s}1_{[0,1]}(\theta) \tag{8.80}$$

where $s = \sum_{i=1}^n X_i$. If n is large then s will be large too and this density will be sharply peaked around s/n.

In this example above the formula for the posterior is quite simple and in particular it is trivial to compute the normalizing constant. In many actual applications this is not the case. Often θ is multidimensional and so just computing the normalizing constant requires doing a multidimensional integral which may not be tractable. We still want to be able to generate samples from the posterior. For example we might want to compute the mean of θ from the posterior and maybe find confidence interval for it. We can try to use MCMC, in particular the Gibbs sampler, to do this.

Example: This is similar to the coin example above but with more parameters. We have a die with probabilities $\theta_1, \theta_2, \cdots, \theta_6$ of getting $1, 2, \cdots, 6$. So the sum of the θ_i must be 1. We role the die n times and let $x_1, x_2, \cdots, x_n$ be the numbers we get. So the x_i take values in $\{1, 2, 3, 4, 5, 6\}$. Putting a prior distribution on the θ_i is a little tricky since we have the constraint that they must sum to 1. Here is one approach. We would like to assume the die is close to being fair and we have no prior reason to think that a particular number is more likely than any other number. Take $\phi_1, \cdots, \phi_6$ to be independent and identically distributed with distribution $g(\phi)$ where ϕ is peaked around $1/6$. So the joint distribution of the ϕ_i is $\prod_i g(\phi_i)$. Then we just set $\theta_i = \phi_i / \sum_j \phi_j$. We now think of the ϕ_i as the parameters.

We have

$$f(x|\theta) = \prod_{i=1}^n \theta_{x_i} = \prod_{j=1}^6 \theta_j^{n_j} \tag{8.81}$$

and so

$$f(x|\phi) = [\sum_{j=1}^6 \phi_j]^{-n} \prod_{j=1}^6 \phi_j^{n_j} \tag{8.82}$$

where n_j is the number of x_i equal to j. We have used the fact that $\sum_{j=1}^6 n_j = n$. So Bayes rule says

$$f(\phi_1, \cdots, \phi_6|x) \propto [\sum_{j=1}^6 \phi_j]^{-n} \prod_{j=1}^6 [\phi_j^{n_j} g(\phi_j)] \tag{8.83}$$

We would like to compute things like the expected value of each θ_i. This would give us an idea of how unfair the die is and just how it is "loaded". We do this by generating samples of $(\phi_1, \cdots, \phi_6)$. We can use the Gibbs sampler. We need the conditional distribution of each ϕ_i given the other ϕ. Up to a normalization constant this is

$$[\phi_i + \Phi]^{-n} \phi_i^{n_i} g(\phi_i) \tag{8.84}$$

where $\Phi = \sum_{j \neq i} \phi_j$.

Example: Zero-inflated poisson process - handbook p. 235.

Review Poisson proccesses, Poisson RV's, and gamma distribution

$Gamma(w, \lambda)$ has pdf

$$f(x) = \frac{\lambda^w}{\Gamma(w)} x^{w-1} e^{-\lambda x} \tag{8.85}$$

Hierarchical models: Suppose we have a parameter λ which we take to be random. For example it could have a $Gamma(\alpha, \beta)$ distribution. Now we go one step further and make β random, say with a $Gamma(\gamma, \delta)$ distribution. So

$$f(\lambda|\beta) = Gamma(\alpha, \beta), \tag{8.86}$$
$$f(\beta) = Gamma(\gamma, \delta) \tag{8.87}$$

and so the prior is

$$f(\lambda, \beta) = f(\lambda|\beta) f(\beta) = \cdots \tag{8.88}$$

Example The following example appears in so many books and articles it is ridiculuous. But it is still a nice example. A nuclear power plant has 10 pumps that can fail. The data consists of an observation time t_i and the number of failures x_i for each pump that have occured by time t_i.

A natural model for the times at which a single pump fail is a Poisson process with parameter λ. We only observe the process at a single time, and the number of failures that have occured by that time is a Possion random variable with parameter λt_i. One model would be to assume that all the pumps have the same failure rate, i.e., the same λ. This is an unrealistic assumption. Instead we assume that each pump has its own failure rate λ_i. The λ_i are assumed to be random and independent, but with a common distribution. We take this common distribution to be $Gamma(\alpha, \beta)$ where α is a fixed value but β is random with distribution $Gamma(\gamma, \delta)$. γ and δ are numbers. The parameters θ here are $\lambda_1, \cdots, \lambda_{10}, \beta$. From now on we write $\lambda_1, \cdots, \lambda_{10}$ as λ. Note that

$$f(\lambda, \beta) = f(\lambda|\beta) f(\beta) \tag{8.89}$$

and

$$f(x|\lambda, \beta) = f(x|\lambda) \tag{8.90}$$

pump	1	2	3	4	5	6	7	8	9	10
Number failures	5	1	5	14	3	19	1	1	4	22
observation time	94.32	15.72	62.88	125.76	5.24	31.44	1.05	1.05	2.10	10.48

Table 8.1:

and we have

$$f(x|\lambda) = \prod_{i=1}^{10} \left[\frac{(\lambda_i t_i)^{x_i}}{x_i!} e^{-\lambda_i t_i} \right] \tag{8.91}$$

Bayes rule says

$$\begin{aligned} f(\lambda, \beta|x) &\propto f(x|\lambda, \beta) f(\lambda, \beta) && (8.92)\\ &= f(x|\lambda, \beta) f(\lambda|\beta) f(\beta) && (8.93)\\ &= \prod_{i=1}^{10} \left[\frac{(\lambda_i t_i)^{x_i}}{x_i!} e^{-\lambda_i t_i} \right] \prod_{i=1}^{10} Gamma(\lambda_i|\alpha, \beta) Gamma(\beta|\gamma, \delta) && (8.94)\\ &= \prod_{i=1}^{10} \left[\frac{(\lambda_i t_i)^{x_i}}{x_i!} e^{-\lambda_i t_i} \right] \prod_{i=1}^{10} \left[\frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\lambda_i \beta} \right] \frac{\delta^\gamma}{\Gamma(\gamma)} \beta^{\gamma-1} e^{-\delta\beta} && (8.95)\\ &\propto \prod_{i=1}^{10} \left[\lambda_i^{x_i+\alpha-1} e^{-\lambda_i(t_i+\beta)} \right] \beta^{10\alpha+\gamma-1} e^{-\delta\beta} && (8.96) \end{aligned}$$

where the constant of proportionality depends on the x_i and t_i and on the constants γ, δ.

We want to compute the posterior distribution of the parameters. We are particularly interested in the mean of the distribution of the λ_i, i.e., the mean of $Gamma(\alpha, \beta)$. The mean of this gamma distribution with fixed α, β is α/β. So we need to compute the mean of α/β over the posterior distribution. We can write this as a ratio of high dimensional (ten or eleven) integrals, but that is hard to compute. So we use the Gibbs sampler to sample λ, β from the posterior. Note that this is an 11 dimensional sampler. So we need the conditional distributions of each λ_i and of β.

$$\lambda_i|\beta, t_i, x_i \sim Gamma(x_i + \alpha, t_i + \beta), \tag{8.97}$$

$$\beta|\lambda \sim Gamma(\gamma + 10\alpha, \delta + \sum_{i=1}^{10} \lambda_i) \tag{8.98}$$

Chapter 9

Convergence and error estimation for MCMC

References:

Robert, Casella - chapter 12

chapter 8 of "handbook" - primarily on statitical analysis

Fishman chap 6

9.1 Introduction - sources of errors

When we considered direct Monte Carlo simulations, the estimator for the mean we were computing was a sum of independent random variables since the samples were independent. So we could compute the variance of our estimator by using the fact that the variance of a sum of independent random variables is the sum of their variances. In MCMC the samples are not independent, and so things are not so simple. We need to figure out how to put error bars on our estimate, i.e., estimate the variance of our estimator.

There is a completely different source of error in MCMC that has no analog in direct MC. If we start the Markov chain in a state which is very atypical for the stationary distribution, then we need to run the chain for some amount of time before it will move to the typical states for the stationary distribution. This preliminary part of the simulation run goes under a variety of names: burn-in, initialization, thermalization. We should not use the samples generated during this burn-in period in our estimate of the mean we want to compute. The

preceeding was quite vague. What is meant by a state being typical or atypical for the stationary distribution?

There is another potential source of error. We should be sure that our Markov chain is irreducible, but even if it is, it may take a very long time to explore some parts of the state space. This problem is sometimes called "missing mass."

We illustrate these sources of errors with some very simple examples. One way to visualize the convergence of our MCMC is a plot of the evolution of the chain, i.e, X_n vs n.

Example: We return to an example we considered when we looked at Metropolis-Hasting. We want to generate samples of a standard normal. Given $X_n = x$, the proposal distribution is the uniform distribution on $[x - \epsilon, x + \epsilon]$, where ϵ is a parameter. So

$$q(x, y) = \begin{cases} \frac{1}{2\epsilon} & \text{if } |x - y| \leq \epsilon, \\ 0, & \text{if } |x - y| > \epsilon, \end{cases} \tag{9.1}$$

(When we looked at this example before, we just considered $\epsilon = 1$.) We have

$$\alpha(x, y) = \min\{\frac{\pi(y)}{\pi(x)}, 1\} = \min\{\exp(-\frac{1}{2}y^2 + \frac{1}{2}x^2), 1\} \tag{9.2}$$

$$= \begin{cases} \exp(-\frac{1}{2}y^2 + \frac{1}{2}x^2) & \text{if } |x| < |y|, \\ 1 & \text{if } |x| \geq |y| \end{cases} \tag{9.3}$$

First consider what the evolution plot for direct MC would look like. So we just generate sample of X_n from the normal distribution. The evolution plot is shown in figure 9.1. Note that there is not really any evolution here. X_{n+1} has nothing to do with X_n.

The next evolution plot (figure 9.2) is for the Metropolis-Hasting algorithm with $X_0 = 0.$ and $\epsilon = 1.0$. The algorithm works well with this initial condition and proposal distribution. The samples are correlated, but the Markov chain mixes well.

The next evolution plot (figure 9.3) is for the Metropolis-Hasting algorithm with $X_0 = 0.$ and $\epsilon = 0.1$. The algorithm does not mix as well with this proposal distribution. The state moves rather slowly through the state space and there is very strong correlation between the samples over long times.

The next evolution plot (figure 9.4) is for the Metropolis-Hasting algorithm with $X_0 = 0.$ and $\epsilon = 100.$. This algorithm does very poorly. Almost all of the proposed jumps take the chain to a state with very low probability and so are rejected. So the chain stays stuck in the state it is in for many time steps. This is seen in the flat regions in the plot.

The next evolution plot (figure 9.5) is for the Metropolis-Hasting algorithm with $X_0 = 10.$ and $\epsilon = 0.1$. Note that this initial value is far outside the range of typical values of X. This

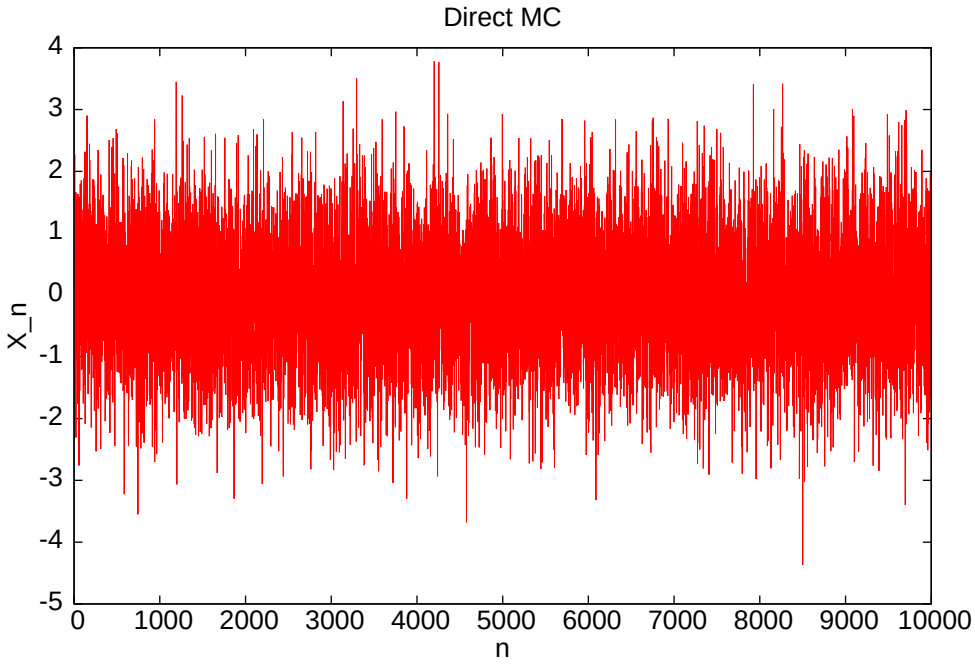

Figure 9.1: Direct Monte Carlo for the normal distribution.

algorithm has a significant burn-in period. It takes a couple thousand time steps before the chain is in a typical state. We need to discard the samples from this burn-in phase.

Now we change the distribution. We consider a mixture or two normal distributions. They both have variance 1; one is centered at 3 and one at -3. So the density is given by

$$f(x) \propto \exp(-\frac{1}{2}(x-3)^2) + \exp(-\frac{1}{2}(x+3)^2) \tag{9.4}$$

We use the same proposal distribution as before. The evolution plot for the Metropolis-Hasting algorithm for this distribution with $X_0 = 0.$ and $\epsilon = 1.0$ is shown in figure 9.6.

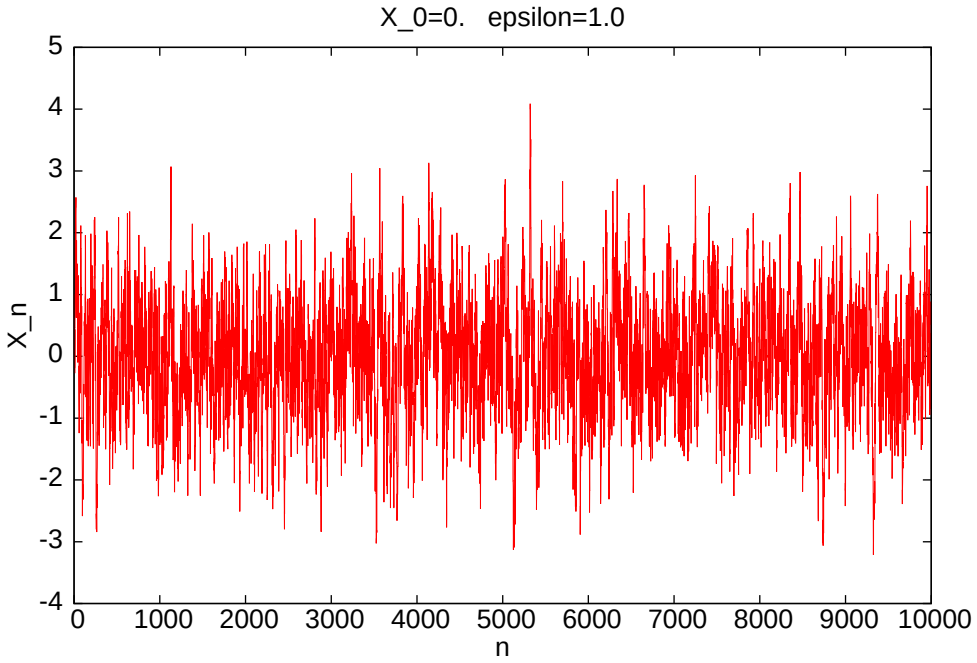

Figure 9.2: Metropolis Hasting for the normal distribution with $X_0 = 0$ and $\epsilon = 1.0$.

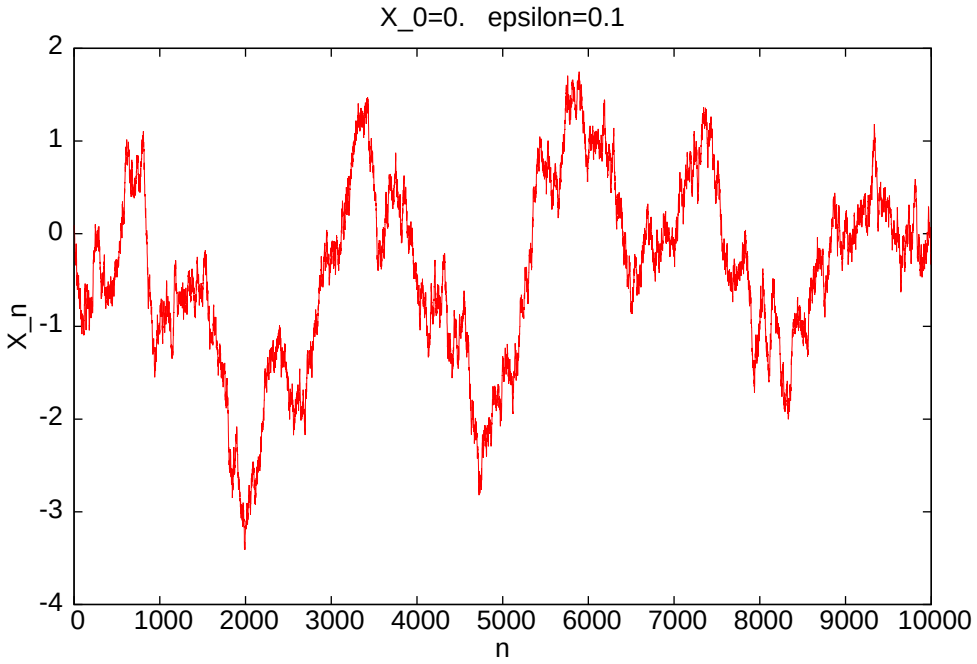

Figure 9.3: Metropolis Hasting for the normal distribution with $X_0 = 0$ and $\epsilon = 0.1$.

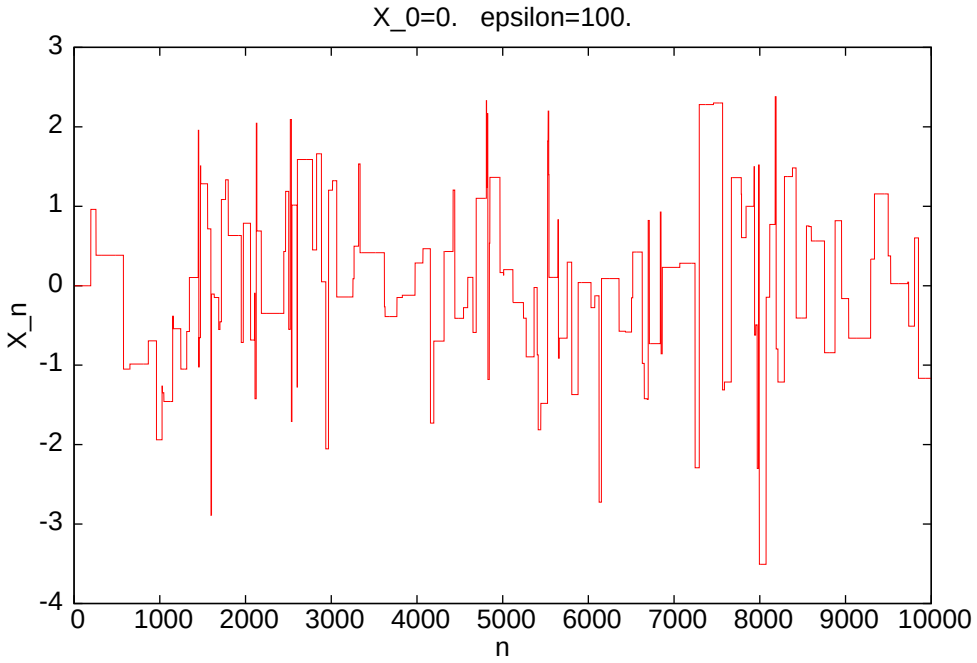

Figure 9.4: Metropolis Hasting for the normal distribution with $X_0 = 0$ and $\epsilon = 100$..

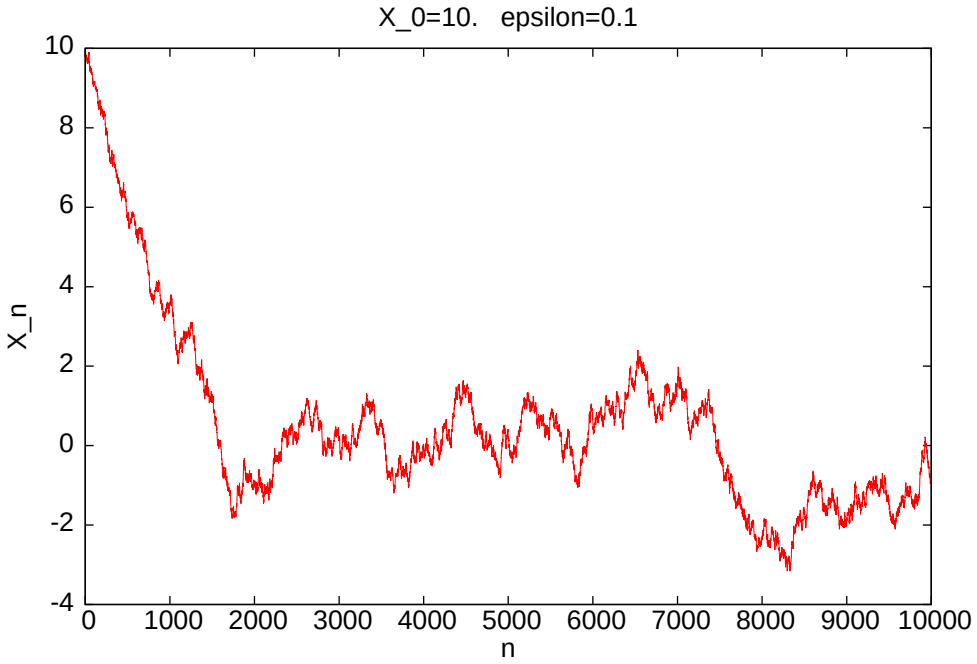

Figure 9.5:

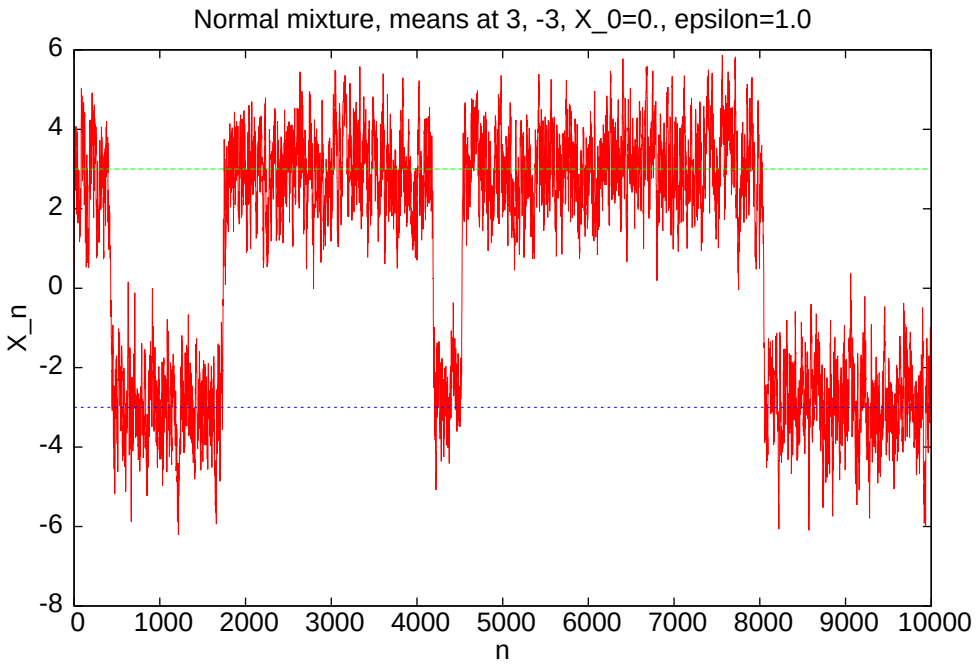
Normal mixture, means at 3, -3, X_0=0., epsilon=1.0
X_n
n
6
4
2
0
-2
-4
-6
-8
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Figure 9.6:

9.2 The variance for correlated samples

We recall a few probability facts. The covarance of random variable X and Y is

$$cov(X,Y) = E[XY] - E[X]E[Y] = E[(X-\mu_X)(Y-\mu_Y)] \tag{9.5}$$

This is a bi-linear form. Also note that $cov(X,X) = var(X)$. Thus for any random variables $Y_1, \cdots, Y_N$,

$$var(\sum_{i=1}^{N} Y_i) = \frac{1}{N^2} \sum_{i,j=1}^{N} cov(X_i, X_j) \tag{9.6}$$

A stochastic process X_n is said to be stationary if for all positive integers m and t, the joint distribution of $(X_{1+t}, X_{2+t}, \cdots, X_{m+t})$ is independent of t. In particular, $cov(X_i, X_j)$ will only depends on $|i-j|$. It is not hard to show that if we start a Markov chain in the stationary distribution, then we will get a stationary process. If the initial distribution is not the stationary distribution, then the chain will not be a stationary process. However, if the chain is irreducible, has a stationary distribution and is aperiodic, then the distribution of X_n will converges to that of the stationary distribution. So if we only look at the chain at long times it will be approximately a stationary process.

As before let

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} f(X_n) \tag{9.7}$$

be our estimator for the mean of $f(X)$. Its variance is

$$Var(\hat{\mu}) = \frac{1}{N^2} \sum_{k,n=1}^{N} cov(f(X_k), f(X_n)) \tag{9.8}$$

In the following we do a bit of hand-waving and make some unjustifiable assumptions. If the initial state X_0 is chosen according to the stationary distribution (which is typically impossible in practice) then these assumptions are justified. But there are certainly situations where they are not. Let $\sigma^2(f)$ be the variance of $f(X)$ in the stationary distribution. If N is large, then for most terms in the sum k and n are large. So the distribution of X_k and X_n should be close to the stationary distribution. So the variance of $f(X_k)$ should be close to $\sigma(f)^2$. So

$$\begin{aligned} cov(f(X_k), f(X_n)) &= [var(f(X_k))var(f(X_n))]^{1/2} cor(f(X_k), f(X_n)) & (9.9) \\ &\approx \sigma^2(f)\, cor(f(X_k), f(X_n)) & (9.10) \end{aligned}$$

where $cor(Y,Z)$ is the correlation coefficient for Y and Z. Thus

$$Var(\hat{\mu}) = \frac{\sigma(f)^2}{N^2} \sum_{k=1}^{N} \sum_{n=1}^{N} cor(f(X_k), f(X_n)) \tag{9.11}$$

Fix a k and think of $cor(f(X_k), f(X_n))$ as a function of n. This function will usually decay as n moves away from k. So for most values of k we can make the approximation

$$\sum_{n=1}^{N} cor(f(X_k), f(X_n)) \approx \sum_{n=-\infty}^{\infty} cor(f(X_k), f(X_{k+n})) = 1 + 2\sum_{n=1}^{\infty} cor(f(X_k), f(X_{k+n})) \quad (9.12)$$

Finally we assume that this quantity is is essentially independent of k. This quantity is called the autocorrelation time of $f(X)$. We denote it by $\tau(f)$. So

$$\tau(f) = 1 + 2\sum_{n=1}^{\infty} cor(f(X_0), f(X_n)) \quad (9.13)$$

Explain why it is a "time" by looking at correlation that decays exponentially with a time scale. We now have

$$Var(\hat{\mu}) \approx \frac{\sigma(f)^2}{N^2} \sum_{k=1}^{N} \tau(f) = \frac{\sigma(f)^2 \tau(f)}{N} \quad (9.14)$$

If the samples were independent the variance would be $\sigma(f)^2/N$. So our result says that this variance for independent samples is increased by a factor of $\tau(f)$. Another way to interpret this result is that the "effective" number of samples is $N/\tau(f)$.

To use this result we need to compute $\tau(f)$. We can use the simulation run to do this. We approximate the infinite sum in the expression for $\tau(f)$ by truncating it at M. So we need to estimate

$$1 + 2\sum_{n=1}^{\infty} cor(f(X_k), f(X_{k+n})) \quad (9.15)$$

$$\hat{\tau(f)} = 1 + 2\frac{1}{N-M} \sum_{k=1}^{N-M} \sum_{n=1}^{M} cor(f(X_k), f(X_{k+n})) \quad (9.16)$$

Of course we do not typically have any a priori idea of how big M should be. So we need to first try to get an idea of how fast the correlation function decays.

MORE Help Help Help

Once we have an estimate for the variance of our estimator of $E[f(X)]$, we can find a confidence interval, i.e. error bars, in the usual way.

9.3 Variance via batched means

Esimating $\tau(f)$ can be tricky. In this section we give a quick and dirty method for putting error bars on our estimator that does not require computing the correlation time $\tau(f)$. Let N

be the number of MC time steps that we have run the simulation for. We pick an integer l and put the samples in batches of length l. Let $b = N/l$. So b is the number of batches. So the first batch is $X_1, \cdots, X_l$, the second batch is $X_{l+1}, \cdots, X_{2l}$, and so on. If l is large compared to the autocorrelation time of $f(X)$, then if we pick X_i and X_j from two different batches then for most choices of i and j, $f(X_i)$ and $f(X_j)$ will be almost independent. So if we form estimators for the batches, $j = 1, 2, \cdots, b$,

$$\hat{\mu}_j = \frac{1}{l} \sum_{i=1}^{b} f(X_{(j-1)l+i}) \tag{9.17}$$

then the $\hat{\mu}_1, \cdots, \hat{\mu}_b$ will be almost independent. Note that

$$\hat{\mu} = \frac{1}{b} \sum_{i=1}^{b} \hat{\mu}_i \tag{9.18}$$

So

$$var(\hat{\mu}) \approx \frac{1}{b^2} \sum_{i=1}^{b} var(\hat{\mu}_i) \tag{9.19}$$

The batches should have essentially the same distribution, so $var(\hat{\mu}_i)$ should be essentially independent of i. We estimate this common variance using the sample variance of $\hat{\mu}_i$, $i = 1, 2, \cdots, b$. Denote it by s_l^2. Note that this batch variance depends very much on the choice of l. Then our estimator for the variance of $\hat{\mu}$ is

$$var(\hat{\mu}) \approx \frac{s_l^2}{b} \tag{9.20}$$

How do we choose the batch size? The number of batches b is N/l. We need b to be reasonably large (say 100, certainly at least 10) since we estimate the variance of the batches using a sample of size b. So l should be at most 1/10 of N. We also need l to be large compared to the autocorrelation time. If N is large enough, there will be a range of l where these two criteria are both met. So for l in this range the estimates we get for the variance of $\hat{\mu}$ will be essentially the same. So in practice we compute the above estimate of $var(\hat{\mu})$ for a range of l and look for a range over which it is constant. If there is no such range, then we shouldn't use batched means. If this happens we should be suspicious of the MC simulation itself since this indicates the number of MC time steps is not a large multiple of the autocorrelation time.

Stop - Wed, 4/6

We return to our hand-waving argument that the batch means should be almost independent if the batch size is large compared to the autocorrelation time and make this argument more quantitative.

The variance of $\hat{\mu}$ is given exactly by

$$var(\hat{\mu}) = \frac{1}{N^2} \sum_{i,j=1}^{N} cov(f(X_i), f(X_j)) \tag{9.21}$$

Using batched means we approximate this by

$$\frac{1}{b^2} \sum_{i=1}^{b} var(\hat{\mu}_i) \tag{9.22}$$

This is equal to

$$\frac{1}{N^2} \sum_{(i,j) \in S} cov(f(X_i), f(X_j)) \tag{9.23}$$

where S is the set of pairs (i, j) such that i and j are in the same batch. So the difference between the true variance of $\hat{\mu}$ and what we get using batched means is

$$\frac{1}{N^2} \sum_{(i,j) \notin S} cov(f(X_i), f(X_j)) \tag{9.24}$$

Keep in mind that the variance of $\hat{\mu}$ is of order $1/N$. So showing this error is small means showing it is small compared to $1/N$. We fix the number b of batches, let $N = bl$ and let $l \to \infty$. We will show that in this limit the error times N goes to zero. So we consider

$$\frac{1}{N} \sum_{(i,j) \notin S} cov(f(X_i), f(X_j)) \tag{9.25}$$

Define $C()$ by $C(|i - j|) = cov(f(X_i), f(X_j))$. Then the above can be bounded by

$$\frac{2b}{N} \sum_{k=1}^{l} \sum_{i=1}^{\infty} |C(i + k)| \tag{9.26}$$

Let

$$T(k) = \sum_{i=1}^{\infty} |C(i + k)| = \sum_{i=k+1}^{\infty} |C(i)| \tag{9.27}$$

We assume that $C(i)$ is absolutely summable. So $T(k)$ goes to zero as $k \to \infty$. Using the fact that $l = N/b$, we have that the above equals

$$\frac{2}{l} \sum_{k=1}^{l} T(k) \tag{9.28}$$

which goes to zero by the analysis fact that if $\lim_{k \to \infty} a_k = 0$ then $\lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} a_k = 0$.

9.4 Subsampling

Until now we have always estimated the mean of $f(X)$ with the estimator

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} f(X_i) \tag{9.29}$$

Subsampling (sometimes called lagging) means that we estimate the mean with

$$\hat{\mu}_{sub} = \frac{1}{N/l} \sum_{i=1}^{N/l} f(X_{il}) \tag{9.30}$$

where l is the subsampling time. In other words we only evaluate f every l time steps.

One reason to do this is that if l is relatively large compared to the autocorrelation time, then the X_{il} will be essentially independent and we can estimate the variance of our estimator just as we did for direct Monte Carlo where the samples where independent. So the variance of the subsampling estimator will be approximately $\sigma^2/(N/l)$, where σ^2 is the variance of $f(X)$.

Intuitively one might expect if we compare two estimators, one with subsampling and one without, for the same number of MC steps, then the estimator without subsampling will do better. More precisely, we expect the variance of $\hat{\mu}$ to be smaller than the variance of $\hat{\mu}_{sub}$ if we use the same N in both. The following proposition makes this rigorous.

Proposition 15 *Define $\hat{\mu}$ as above and define*

$$\hat{\mu}_j = \frac{1}{N/l} \sum_{i=0}^{N/l-1} f(X_{il+j}) \tag{9.31}$$

Letting $\sigma(?)$ denote the standard deviation of ?, we have

$$\sigma(\hat{\mu}) \leq \frac{1}{l} \sum_{j=1}^{l} \sigma(\hat{\mu}_j) \tag{9.32}$$

Typically the variances of the $\hat{\mu_j}$ will be essentially the same and so morally the proposition says that the variance of $\hat{\mu}$ is no larger than the variance of $\hat{\mu}_{sub}$.

Proof: The crucial fact used in the proof is that for any RV's X, Y,

$$|cov(X,Y)| \leq [var(X)\, var(Y)]^{1/2} \tag{9.33}$$

which follows from the Cauchy Schwarz inequality. Note that

$$\hat{\mu} = \frac{1}{l} \sum_{j=1}^{l} \hat{\mu}_j \tag{9.34}$$

So

$$\begin{aligned} var(\hat{\mu}) &= \frac{1}{l^2} \sum_{i=1}^{l} \sum_{j=1}^{l} cov(\hat{\mu}_i, \hat{\mu}_j) && (9.35) \\ &\leq \frac{1}{l^2} \sum_{i=1}^{l} \sum_{j=1}^{l} [var(\hat{\mu}_i) var(\hat{\mu}_j)]^{1/2} && (9.36) \\ &= \frac{1}{l^2} \sum_{i=1}^{l} \sum_{j=1}^{l} \sigma(\hat{\mu}_i) \sigma(\hat{\mu}_j) && (9.37) \\ &= \left[\frac{1}{l} \sum_{i=1}^{l} \sigma(\hat{\mu}_i) \right]^2 && (9.38) \end{aligned}$$

Taking square roots the result follows. **QED**.

The proposition does not imply that it is never beneficial to subsample. Computing f for every time step in the Markov chain takes more CPU time than subsampling with the same number of time steps. So there is a subtle trade-off between the increase in speed as we subsample less frequently and the increase in the variance. Choosing l so large that it is much larger than the autocorrelation time is not necessarily the optimal thing to do. Ideally we would like to choose l to minimize the error we get with a fixed amount of CPU time. Finding this optimal choice of l is not trivial.

Factor of two hueristic: Actually finding the optimal amount choice of l is non-trivial. We give a crude method that at worst will require twice as much CPU time as the optimal choice of l will. Let τ_f be time to compute $f(X_i)$ and let τ_X be the time to perform one time step, i.e., compute X_{i+1} given X_i. We take l to be τ_f/τ_X (rounded to an integer). Note that this will make the time spent on computing the X_i equal to the time spent evaluating f on the X_i. Now we argue that this is worse than by the optimal choice by at most a factor of two in CPU time.

Let l_{opt} be the optimal choice of l. First consider the case that $l < l_{opt}$, i.e., we are evaluating f more often than in the optimal case. Now compare a simulation with N time steps with our crude choice of l with a simulation with N time steps with the optimal choice l_{opt}. They have the same number of MC steps, but the simulation using l is sampled more often and so is at least as accurate as the simulation using l_{opt}. The simulation using l takes more CPU time, but at most the extra time is the time spent evaluating f and this is half of the total CPU time. So the simulation using l takes at most twice as long as the one using l_{opt}.

Now consider the case that $l > l_{opt}$. Now we compare two simulations that each evaluate f a total of M times. Since the evaluations using l are more widely spaced in time, they will be less correlated than those for the simulation using l_{opt}. So the simulation using l will be at least as accurate as the simulation using l_{opt}. The two simulations evaluate f the same number of times. The simulation using l requires more MC steps. But the total time it spends computing the X_i is equal to the total time it spends evaluating f. So the total CPU is twice the time spent evaluating f. But the time spent evaluating f is the same for the two simulations, and so is less that the total time the simulation using l uses.

9.5 Burn-in or initialization

In this section we consider the error resulting from the fact that we start the chain in an arbitrary state X_0 which may be atypical is some sense. We need to run the chain for some number T of time steps and discard those times steps, i.e., we estimate the mean using

$$\frac{1}{N-T} \sum_{i=T+1}^{N} f(X_i) \tag{9.39}$$

This goes under a variety of names: initilization, burn-in, convergence to stationarity, stationarization, thermalization.

We start with a trivial, but sometimes very relevant comment. Suppose we have a direct MC algorithm that can generate a sample from the distribution we are trying to simulate, but it is really slow. As long as it is possible to generate one sample in a not unreasonable amount of time, we can use this algorithm to generate the inital state X_0. Even if the algorithm that directly samples from π takes a thousand times as much time as one step for the MCMC algorithm, it may still be useful as a way to initialize the MCMC algorithm. When we can do this we eliminate the burn-in or initialization issue altogether.

In the example of burn-in at the start of this chapter, $f(X)$ was just X. When we started the chain in $X_0 = 10$, this can be thought of as starting the chain in a state where the value of $f(X)$ is atypical. However, we should emphasize that starting in a state with a typical value of $f(X)$ does not mean we will not need a burn-in period. There can be atypical X for which $f(X)$ is typical as the following example shows.

Example: We want to simulate a nearest neighbor, symmetric random walk in one dimension with n steps. This is easily done by direct Monte Carlo. Instead we consider the following MCMC. This is a 1d version of the pivot algorithm. This 1d RW can be thought of as a sequence of n steps which we will denote by $+1$ and -1. So a state $(s_1, s_2, \cdots, s_n)$ is a string of n $+1$'s and -1's. Let $f(s_1, \cdots, s_n) = \sum_{i=1}^{n} s_i$, i.e., the terminal point of the random

walk. The probability measure we want is the uniform measure - each state has probability $1/2^n$. The pivot algorithmn is as follows. Pick an integer j from $0, 1, 2, \cdots, n-1$ with the uniform distribution. Leave s_i unchanged for $i \leq j$ and replace s_i by $-s_i$ for $i > j$. Show this statisfies detailed balance.

In the following plots the number of steps in the walk is always $L = 1,000,000$. All simulations are run for 10,000 time steps. The initial state and the RV we look at varies. Note that all state have probability 2^{-n}, so no state is atypical in the sense of having unusually small probability.

In the first plot, figure 9.7, we start with the state that is $n/2$ +'s, followed by $n/2$ −'s. The RV plotted is the distance of the endpoint to the origin. The signed distance to the endpoint is approximately normal with mean zero and variance L. So the range of this RV is roughly $[0, 2\sqrt{L}]$. In the initial state the RV is 0. This is a typical value for this RV. There is a significant burn-in period.

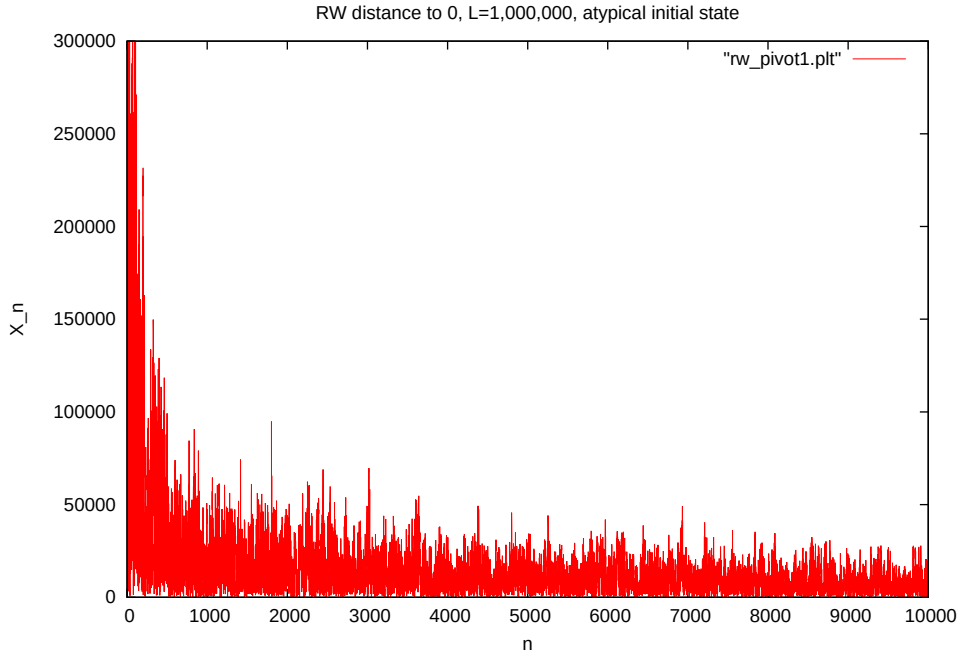

Figure 9.7: MCMC simulation of random walk with $L = 1,000,000$ steps. Random variable plotted is distance from endpoint of walk to 0. Initial state is a walk that goes right for $500,000$ steps, then left for $500,000$ steps. MC is run for 10,000 time steps. There is a significant burn-in period.

In the second plot, figure 9.8, the initial state is a random walk generated by just running a direct MC. The RV is still the distance from the endpoint to the origin. Note the difference in vertical scale between this figure and the preceding figure. This MCMC seems to be working

well.

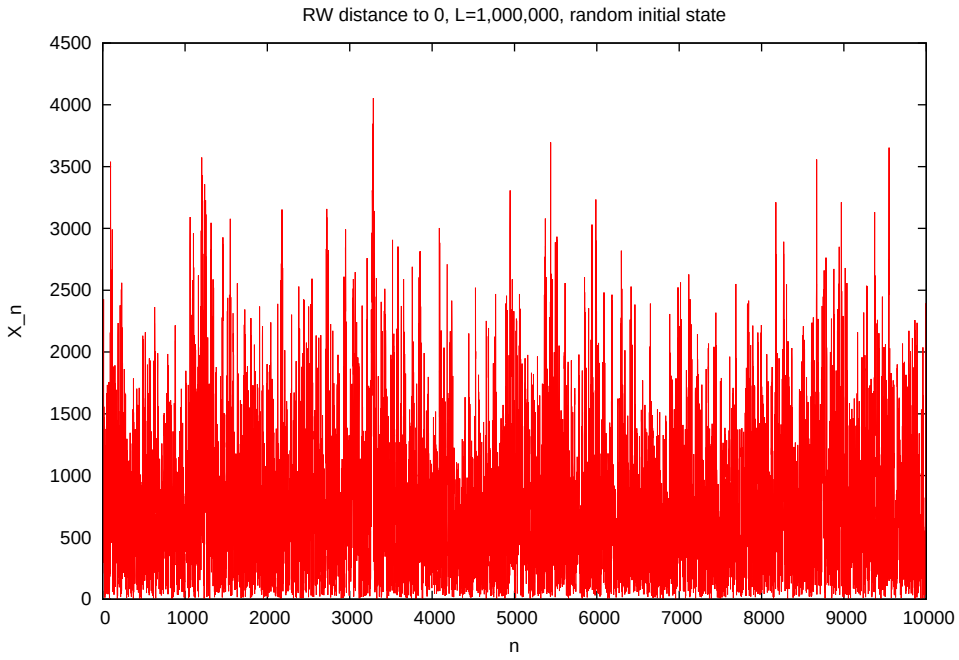

Figure 9.8: MCMC simulation of random walk with $L = 1,000,000$ steps. Random variable plotted is distance from endpoint of walk to 0. Initial state is a random walk generated by direct sampling. MC is run for 10,000 time steps.

In the third plot, figure 9.9, the random variable is the distance the walk travels over the time interval $[L/2, L/2 + 10,000]$. The initial state is again a random walk generated by just running a direct MC. For this RV the autocorrelation time is large.

There are two lessons to be learned from this example. One is the subtlety of initilization. The other is that there can be very different time scales in our MCMC. The autocorrelation time of the RV that is the distance to the origin of the endpoint appears to be not very large. By contrast, the autocorrelation time for the other RV is rather large. We might expect that the exponential decay time for $cov(f(X_i), f(X_j))$ for the first RV will be not too large while this time for the second RV will be large. However, it is quitely like that for the first RV there is some small coupling to this "slow mode." So even for the first RV the true exponential decay time for the covariance may actually be quite large. Note that there are even slower modes in this system. For example, consider the RV that is just the distance travelled over the time interal $[L/2, L/2 + 2]$.

A lot of theoretical discussions of burn-in go as follows. Suppose we start the chain in some

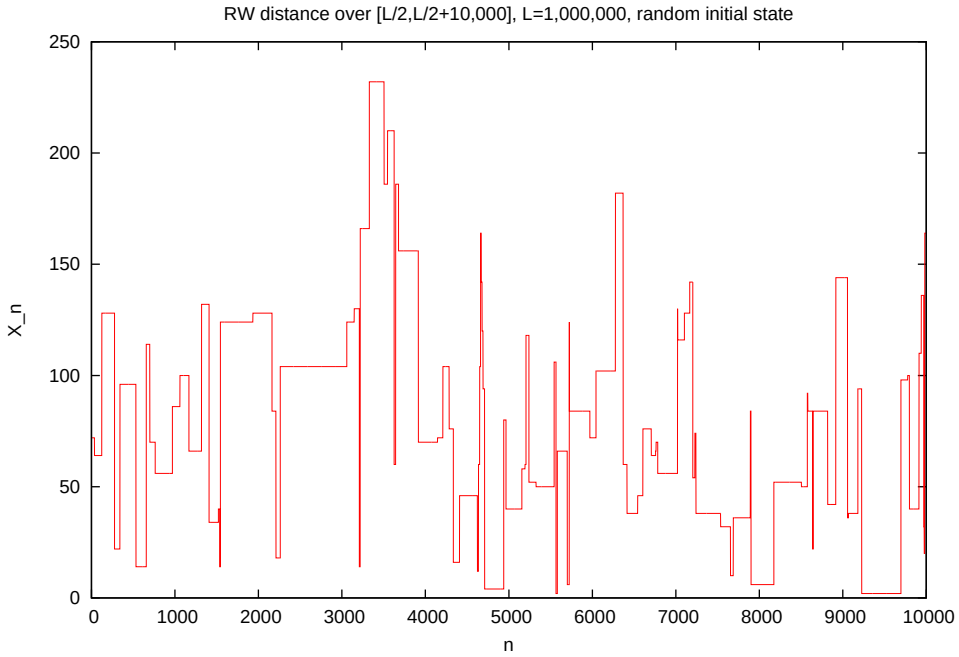

Figure 9.9: MCMC simulation of random walk with $L = 1,000,000$ steps. Random variable plotted is distance walk travels over the time interval $[L/2, L/2 + 10,000]$. Initial state is a random walk generated by direct sampling.

distribution π_0 and let π_n be the distribution of X_n. Suppose we have a bound of the form

$$||\pi_n - \pi||_{TV} \leq C \exp(-n/\tau) \tag{9.40}$$

Then we can choose the number of samples T to discard by solving for T in $C \exp(-T/\tau) \leq \epsilon$ where ϵ is some small number. The problem is than we rarely have a bound of the above form, and in the rare cases when we do, the τ may be far from optimal. So we need a practical test for convergence to stationarity.

Stop - Wed, 4/6

If the distribution of the initial state X_0 is the stationary (so the chain is stationary), then the two samples

$$f(X_1), f(X_2), \cdots, f(X_N), \quad \textit{and} \quad f(X_{N+1}), f(X_{N+2}), \cdots, f(X_{2N}) \tag{9.41}$$

have the same distribution.

To use this observation to test for converges to stationarity, we let T be the number of samples we will discard for burn-in purposes and compare the two samples

$$f(X_{T+1}), f(X_{T+2}), \cdots, f(X_{T+N}), \quad \textit{and} \quad f(X_{T+N+1}), f(X_{T+N+2}), \cdots, f(X_{T+2N}) \tag{9.42}$$

A crude graphical test is just to plot histograms of these two samples and see if they are obviously different.

For a quantitative test we can use the Kolmogorov-Smirnov statistic. This is a slightly different version of KS from the one we saw before when we tested if a sample came from a specific distribution with a given CDF F. We first forget about the Markov chain and consider a simpler situation. Let $X_1, X_2, \cdots X_{2N}$ be i.i.d. We think of this as two samples: $X_1, X_2, \cdots X_N$ and $X_{N+1}, X_{N+2}, \cdots X_{2N}$. We form their empirical CDF's:

$$F_1(x) = \frac{1}{N} \sum_{i=1}^{N} 1_{X_i \leq x}, \tag{9.43}$$

$$F_2(x) = \frac{1}{N} \sum_{i=N+1}^{2N} 1_{X_i \leq x} \tag{9.44}$$

Then we let

$$K = \sup_x |F_1(x) - F_2(x)| \tag{9.45}$$

Note that K is a random variable (a statistic). As $N \to \infty$, the distribution of $\sqrt{N}K$ converges. The limiting distribution has CDF

$$R(x) = 1 - \sum_{k=1}^{\infty} (-1)^{k-1} \exp(-2k^2 x^2) \tag{9.46}$$

This sum converges quickly and so $R(x)$ is easily computed numerically. It is useful to know the 95% cutoff. We have $R(1.36) = 0.95$. So for large N, $P(\sqrt{N}K \leq 1.36) \approx 0.95$.

Now return to MCMC. We cannot directly use the above since the samples of our Markov chain are correlated. So we must subsample. Let l be large compared to the autocorrelation time for f, i.e., $\tau(f)$. Then we compare the two samples

$$f(X_{T+l}), f(X_{T+2l}), \cdots, f(X_{T+Nl}), \quad \textit{and} \quad f(X_{T+Nl+l}), f(X_{T+Nl+2l}, \cdots, f(X_{T+2Nl}) \tag{9.47}$$

Finally, we end this section by noting that it never hurts (much) to throw away the first 10% of your simulation run.

9.6 Autocorrelation times and related times

This section is based in large part on the article "The Pivot Algorithm: A Highly Efficient Monte Carlo Method for the Self-Avoiding Walk" by Madras and Sokal, Journal of Statistical Physics, **50**, 109-186 (1988), sections 2.2.

We assume the Markov chain is discrete with a finite state space. Not all of the following statement extend to the case of continuous state space or infinite discrete state space. We also assume our chain is irrecducible and aperiodic.

In this section we study three different times that can be defined for an MCMC and how they are related. We have already seen one autocorrelation time defined by (9.13). In this section we will refer to this as an "integrated autocorrelation time" and write it as $\tau_{int,f}$. We recall the formula for it:

$$\tau_{int,f} = 1 + 2\sum_{n=1}^{\infty} cor(f(X_0), f(X_n)) \tag{9.48}$$

This time is important since it enters our formula for the variance of our estimator $\hat{\mu}$. That variance is given approximatly by $\sigma(f)^2\tau_{int,f}/N$. Since the variance for direct Monte Carlo is $\sigma(f)^2/N$, we can interpret this formula as saying that in an MCMC the number of effectively independent samples is $N/\tau(f)$. (The definition of $\tau_{int,f}$ in Madras and Sokal differs by a factor of 2.) In the definition of $\tau_{int,f}$ we assume that X_0 is distributed according to the stationary distribution. So the Markov chain is stationary. (We can imagine that we have run the chain for a sufficiently long burn-in period to achieve this. In that case X_0 is the state after the burn-in period.)

The second time we will consider is closely related. We follow the terminology of Madras and Sokal. For a function f on the state space we define the unnormalized autocorrelation function of f to be

$$C_f(t) = E[f(X_s)f(X_{s+t})] - \mu_f^2, \quad where \tag{9.49}$$

$$\mu_f = E[f(X_s)] \tag{9.50}$$

Note that $C_f(0)$ is the variance of $f(X_t)$. We define the normalized autocorrelation function to be

$$\rho_f(t) = \frac{C_f(t)}{C_f(0)} \tag{9.51}$$

With our assumptions on the Markov chain $\rho_f(t)$ will decay exponentially with t. We define the *exponential autocorrelation time* for f by

$$\tau_{exp,f} = \limsup_{t\to\infty} \frac{t}{-log|\rho_f(t)|} \tag{9.52}$$

We define an exponential autocorrelation time for the entire chain by

$$\tau_{exp} = \sup_f \tau_{exp,f} \tag{9.53}$$

With our assumptions on the Markov chain, τ_f will be finite, but it can be infinite when the state space is finite.

The two times we have considered so far are properties of the stationary Markov chain. The third time characterizes how long it takes the Markov chain to converge to the stationary distribution starting from an arbitrary initial condition. With our assumptions on the chain this convergence will be exponentially fast. We define τ_{conv} to be the slowest convergence rate we see when we consider all initial distributions. More precisely we define

$$\tau_{conv} = \sup_{\pi_0} \limsup_{t \to \infty} \frac{t}{-\log(||\pi_t - \pi||_{TV})} \tag{9.54}$$

Here π_0 is the distribution of X_0, π_t is the distribution of X_t and π is the stationary distribution.

The following two propositions almost say that $\tau_{exp} = \tau_{conv}$.

Proposition 16 *Suppose there are constants c and τ such that for all initial π_0,*

$$||\pi_t - \pi||_{TV} \leq ce^{-t/\tau} \tag{9.55}$$

Then $\tau_{exp} \leq \tau$.

Remark: This almost says $\tau_{exp} \leq \tau_{conv}$. It does not exactly say this since our definition of τ_{conv} does not quite imply the bound (9.55) holds with $\tau = \tau_{conv}$.

Proof: If we apply the hypothesis to the initial condition where π_0 is concentrated on the single state x, we see that

$$||p^t(x, \cdot) - \pi(\cdot)||_1 \leq ce^{-t/\tau} \tag{9.56}$$

Now let f be a function on the state space with zero mean in the stationary distribution. Then using $\sum_y f(y)\pi(y) = 0$, we have

$$\begin{aligned}
|E[f(X_0)f(X_t)]| &= |\sum_{x,y} f(x)\pi(x)p^t(x,y)f(y)| && (9.57)\\
&= |\sum_{x,y} f(x)\pi(x)f(y)[p^t(x,y) - \pi(y)]| && (9.58)\\
&\leq \sum_x |f(x)|\pi(x)||f||_\infty ||p^t(x,\cdot) - \pi(\cdot)||_1 && (9.59)\\
&\leq \sum_x |f(x)|\pi(x)||f||_\infty ce^{-t/\tau} && (9.60)
\end{aligned}$$

The propostion follows. **QED**.

Proposition 17 *Suppose there are constants c and τ such that*

$$Cov(g(X_0), f(X_t)) \leq ce^{-t/\tau}||f||_\infty||g||_\infty \tag{9.61}$$

for all functions f and g on the state space. Then $\tau_{conv} \leq \tau$.

Remark: This almost says $\tau_{conv} \leq \tau_{exp}$. It does not exactly say this since our definition of τ_{exp} does not quite imply the bound (9.61) holds with $\tau = \tau_{exp}$.

Proof: The total variation norm can be computed by

$$||\pi_t - \pi||_{TV} \sup_{f:||f||_\infty \leq 1} \sum_x f(x)[\pi_t(x) - \pi(x)] \tag{9.62}$$

Let $g(x) = \pi_0(x)/\pi(x)$. (Note that $\pi(x) > 0$ for all x.) The expected value of g is the stationary distribution is

$$Eg(X_t) = \sum_x g(x)\pi(x) = \sum_x \pi_0(x) = 1 \tag{9.63}$$

So

$$\begin{aligned} Cov(g(X_0), f(X_t)) &= \sum_{x,y} g(x)f(y)\pi(x)p^t(x,y) - E[g(X_0)]E[f(X_t)] && (9.64)\\ &= \sum_{x,y} \pi_0(x)f(y)p^t(x,y) - E[f(X_t)] && (9.65)\\ &= \sum_y \pi_t(y)f(y) - \sum_x \pi(x)f(x) && (9.66) \end{aligned}$$

Since this is bounded by $ce^{-t/\tau}||f||_\infty||g||_\infty = ce^{-t/\tau}||g||_\infty$, the proposition follows. **QED**.

Recall that the stationary distribution is a left eigenvector of the transition matrix $p(x,y)$, and the constant vector is a right eigenvector. (Both have eigenvalue 1.) In general $p(x,y)$ is not symmetric, so there need not be a complete set of left or right eigenvectors. However, if the chain satisfies detailed balance then there is, as we now show. We rewrite the detailed balance condition as

$$\pi(x)^{1/2}p(x,y)\pi(y)^{-1/2} = \pi(y)^{1/2}p(x,y)\pi(y)^{-1/2} \tag{9.67}$$

So if we define

$$\hat{p}(x,y) = \pi(x)^{1/2}p(x,y)\pi(y)^{-1/2} \tag{9.68}$$

then $\hat{p}$ is symmetric. (p is self-adjoint on $l^2(\pi)$.) If we let S be the linear operator on functions on the state space that is just multiplication by $\pi^{1/2}$, then

$$\hat{p} = SpS^{-1} \tag{9.69}$$

Let $\hat{e}_k$ be a complete set of eigenvectors for it. So $\hat{p}\hat{e}_k = \lambda_k \hat{e}_k$. Note that the $\hat{e}_k$ are orthonormal.

Now let $f(x)$ be a function on the state space. It suffices to consider functions which have zero mean in the stationary distribution. So the covariance is just $E[f(X_0)f(X_t)]$. To compute this expected value we need the joint distribution of X_0, X_t. It is $\pi(x_0)p^t(x_0, x_t)$. So

$$E[f(X_0)f(X_t)] = \sum_{x_0,x_t} \pi(x_0)p^t(x_0,x_t)f(x_0)f(x_t) = \sum_{x_0} f(x_0)\pi(x_0) \sum_{x_t} p^t(x_0,x_t)f(x_t) \tag{9.70}$$

We can write this as

$$(f\pi, p^t f) = (f\pi, (S^{-1}\hat{p}S)^t f) = (f\pi, S^{-1}\hat{p}^t S f) = (S^{-1}\pi f, \hat{p}^t S f) = (f\pi^{1/2}, \hat{p}^t \pi^{1/2} f) \tag{9.71}$$

Using the spectral decompostion of $\hat{p}$ this is

$$\sum_k (f\pi^{1/2}, \hat{e}_k)\lambda_k^t (\hat{e}_k, \pi^{1/2} f) = \sum_k c_k^2 \lambda_k^t \tag{9.72}$$

with $c_k = (f\pi^{1/2}, \hat{e}_k)$.

The right eigenvector of p with eigenvalue 1 is just the constant vector. So $\pi^{1/2}$ is the eigenvector of $\hat{p}$ with eigenvalue 1. So

$$c_1 = \sum_x \pi(x)^{1/2}\pi(x)^{1/2} f(x) = 0 \tag{9.73}$$

since the mean of f in the stationary distribution is zero. Let λ be the maximum of the absolute values of the eigenvalues not equal to 1. So for $k \geq 2$, $|\lambda_k^t| \leq \lambda$. So

$$E[f(X_0)f(X_t)] \leq \lambda^t \sum_{k \geq 2} c_k^2 \tag{9.74}$$

Note that $var(f(X_0)) = \sum_k c_k^2$. So

$$\tau_{int,f} = 1 + 2\sum_{t=1}^{\infty} cor(f(X_0), f(X_t)) \leq 1 + 2\sum_{t=1}^{\infty} \lambda^t = 1 + 2\frac{\lambda}{1-\lambda} = \frac{1+\lambda}{1-\lambda} \leq = \frac{2}{1-\lambda} \tag{9.75}$$

Stop - Mon, 4/11

Proposition 18 *Assume that p is diagonalizable. Let λ be the maximum of the absolute values of the eigenvalues not equal to 1. Then τ_{conv} is given by $\lambda = \exp(-1/\tau_{conv})$.*

Proof: Let S be an invertible matrix that diagonalizes p. So $p = SDS^{-1}$ where D is diagonal with entries λ_k. We order things so that the first eigenvalue is 1. We have $\pi p = \pi$. So $\pi SD = \pi S$. So πS is an eigenvector of D with eigenvalue 1. So it must be $(1, 0, \cdots, 0)$. So $\pi = (1, 0, \cdots, 0)S^{-1}$. This says that the first row of S^{-1} is π. If we let $\vec{1}$ be the column vector $(1, 1, \cdots)^T$, then we know $p\vec{1} = \vec{1}$. So $DS^{-1}\vec{1} = S^{-1}\vec{1}$. This says $S^{-1}\vec{1}$ is the eigenvector of D with eigenvalue 1, so it is $(1, 0, \cdots, 0)^T$. So $\vec{1} = S(1, 0, \cdots, 0)^T$. We now have

$$\begin{aligned} \pi_t &= \pi_0 p^t = \pi_0 (SDS^{-1})^t = \pi_0 S D^t S^{-1} = \pi_0 S diag(\lambda_1^t, \cdots, \lambda^t) S^{-1} && (9.76) \\ &= \pi_0 S diag(1, 0, 0 \cdots, 0) S^{-1} + \pi_0 S diag(0, \lambda_2^t, \cdots, \lambda^t) S^{-1} && (9.77) \end{aligned}$$

Since $\vec{1} = S(1, 0, \cdots, 0)^T$, $Sdiag(1, 0, \cdots)$ is the matrix with 1's in the first column and 0's elsewhere. So $\pi_0 Sdiag(1, 0, 0 \cdots, 0)$ is just $(1, 0, \cdots, 0)^T$ Thus $\pi_0 Sdiag(1, 0, 0 \cdots, 0)S^{-1} = \pi$. The second term in the above goes to zero like λ^t. **QED**

Now consider the bound $\tau_{int,f} \leq \frac{2}{1-\lambda}$ which we derived when the chain satistifed detailed balance. Using the previous proposition, this becomes

$$\tau_{int,f} \leq \frac{2}{1 - \exp(-1/\tau_{conv})} \tag{9.78}$$

If τ_{conv} is large, then the right side is approximately $2\tau_{conv}$.

Madras and Sokal carry out a detailed analysis for the pivot algorithm for a random walk in two dimensions. Then show that τ_{exp} is $O(N)$. For "global" random variables such as the distance from the origin to the end of the walk, $\tau_{f,int}$ is $O(\log(N))$. But for very local observables, such as the angle between adjacent steps on the walk, $\tau_{f,int}$ is $O(N)$.

9.7 Missing mass or bottlenecks

The problem we briefly consider in this section is an MCMC in which the state space can be partitioned into two (or more) subsets $S = S_1 \cup S_2$ such that although the probabilities of getting from S_1 to S_2 and from S_2 to S_1 are not zero, they are very small.

The worst scenario is that we start the chain in a state in one of the subsets, say S_1, and it never leaves that subset. Then our long time sample only samples S_1. Without any apriori knowledge of the distribution we want to simulate, there is really no way to know if we are

failing to sample a representative portion of the distribution. "You only know what you have seen."

The slightly less worse scenario is that the chain does occasionally make transitions between S_1 and S_2. In this case there is no missing mass, but the autocorrelation time will be huge. In particular it will be impossible to accurately estimate the probabilities of S_1 and S_2.

It is worth looking carefully at the argument that your MCMC is irreducible to see if you can see some possible "bottlenecks".

If a direct MC is possible, although slow, we can do a preliminary direct MC simulation to get a rough idea of where the mass is.

Missing mass is not just the problem that you fail completely to visit parts of the state space, there is also the problem that you do eventually visit all of the state space, but it takes a long time to get from some modes to other modes. One way to test for this is to run multiple MCMC simulations with different initial conditions.

In the worst scenario when the chain is stuck in one of the two subsets, it may appear that there is a relatively short autocorrelation time and the burn-in time is reasonable. So our estimate of the autocorrelation time or the burn-in time will not indicate any problem. However, when the chain does occassionally make transitions between modes, we will see a very long autocorrelation time and a very slow convergence to stationary. So our tests for these sources of errors will hopefully alert us to the bottleneck.

Part III

Further topics

Chapter 10

Optimization

In this chapter we consider a very different kind of problem. Until now our prototypical problem is to compute the expected value of some random variable. We now consider minimization problems. For example we might have a purely non-random problem: find the mininum of some function $H(x)$ where x ranges over X, and find the minimizer. Or we might want to minimize some function which is the mean of some random variable.

10.1 Simulated annealing

We consider the problem of minimizing $H(x)$, a non-random problem a priori. We will look at simulated annealing which is especially useful in situations where $H(x)$ has local minima which cause many of the standard minimization methods like steepest descent to get "stuck." Explain the name.

The basic idea is to study the probability measure

$$\frac{1}{Z}e^{-\beta H(x)} \tag{10.1}$$

on X and let $\beta \to \infty$. In this limit this probability measure should be concentrated on the minimizing x or x's. If we just set β to a large value and pick an initial state at random, then the algorithm can get stuck near a local minimum near this initial value. The key idea behind simulated annealing is to start with a small value of β and then slowly increase β as we run the MCMC algorithm. In physics, β is propotional to the inverse of the temperature. So letting β go to infinity means the temperature goes to zero. So this is often called "cooling."

One thing we need to specify for the algorithm is how fast we increase β. A standard choice is to let $\beta = \rho^n$ where n is the time in the MC simulation and ρ is a parameter that is just

slightly bigger than 1. There is no obvious way to choose ρ. One should try different values and diferent choices of the initial condition. One can then compare the final state for all the MCMC runs and take the one with the smallest H.

So the algorithm looks like this:

1. Pick an initial state X_0, an inital $\beta = \beta_0$, and a cooling rate ρ.

2. For $i = 1, \cdots, N$, let $\beta_i = \rho\beta_{i+1}$ and use some MCMC algorithm with β_i to generate the next state X_i.

3. The final state X_N is our approximation to the minimizer.

We could also compute $H(X_i)$ at each time step and keep track of which X_i is best so far. It is possible that along the way we get a state X_i which is better than the final X_N. If it is expensive to compute $H()$ this may not be worth doing.

Example: We start with a trivial example that illustrates how the method avoids getting stuck in local minima.

$$H(x) = sin^2(\pi x) + \alpha x^2 \tag{10.2}$$

where $\alpha > 0$ is a parameter. Obviously the minimum is at $x = 0$, and there are local minima at the integers. If α is small some of these local minima are very close to the true minima. In our simulations we take $\alpha = 0.1$. With this value of α the local minima near 0 are pretty close to the global minima.

We use Metropolis-Hastings with a proposal distribution that is just uniform on $[X_n - \epsilon, X_n + \epsilon]$. We take $\epsilon = 0.2$ and $X_0 = 10.$. We start with $\beta_0 = 0.1$ and raise β by the rule $\beta_n = \rho\beta_{n-1}$ where we will try several different ρ. We consider three different choices of ρ. For each choice we run four simulations. The simulations are run until β reaches 100. Note that this correpsonds to very different numbers of time steps. We use a logarithmic scale on the horizontal axis in the plots. Since $\log(\beta) = n\log(\rho)$ this is a linear scale for the number of Monte Carlo steps.

In the first figure, figure 10.1, we use $\rho = 1.005$. The cooling is too rapid in these simulations and so the chain gets stuck in local minima that are not the global minima.

In the second figure, figure 10.2, we run four simulations with $\rho = 1.0001$. This does better than the previous simulation. One of the four simulations finds the global min while the other three get stuck in an adjacent min.

In the third figure, figure 10.3, we take with $\rho = 1.0000001$. Three of the four runs find the global min, but one still gets stuck in a local min.

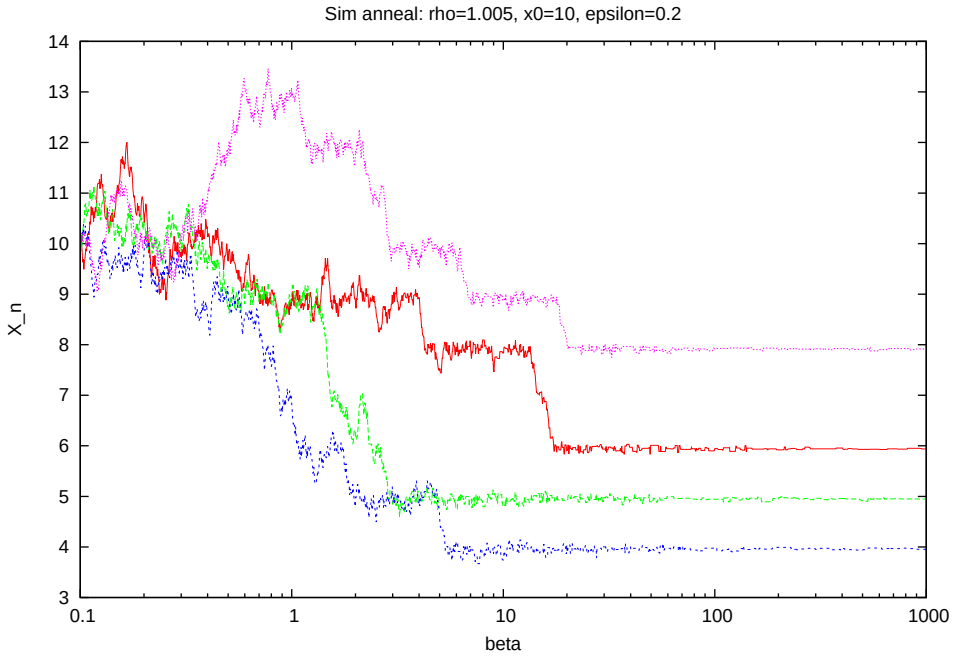

Figure 10.1: Simulated annealing $\rho = 1.005$.

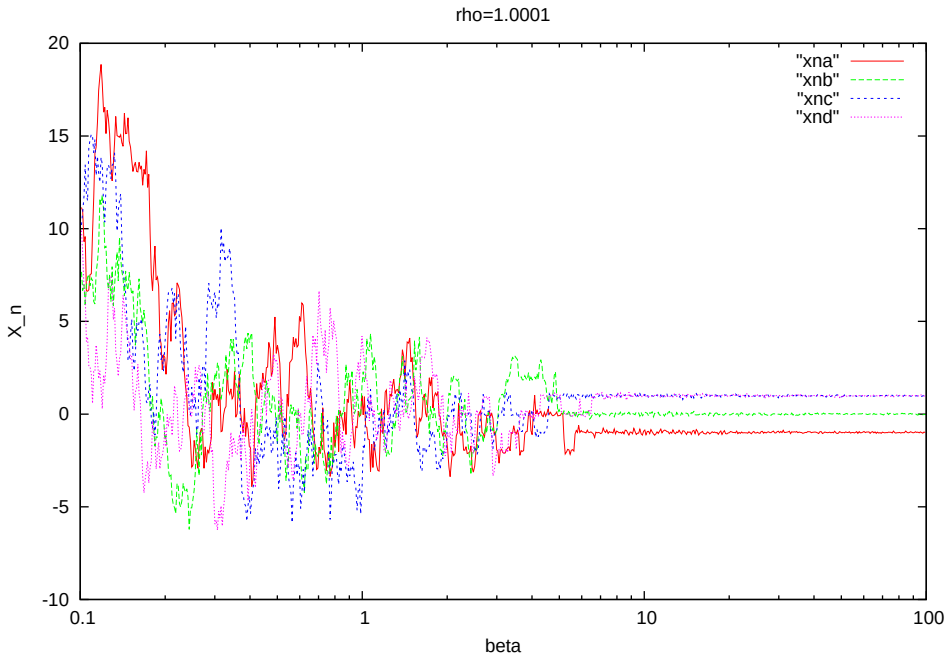

Figure 10.2: Simulated annealing $\rho = 1.0001$.

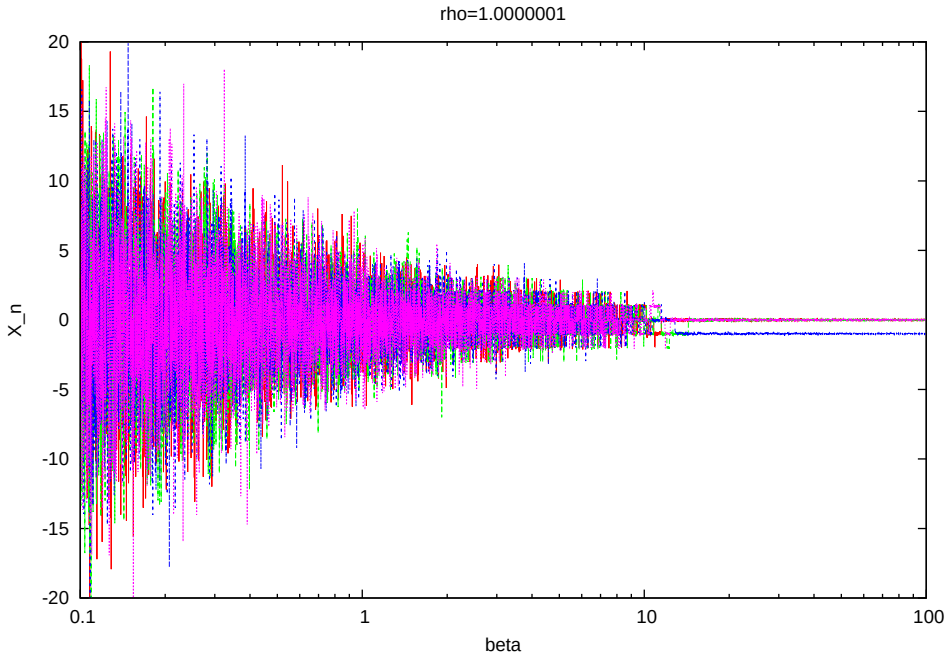

Figure 10.3: Simulated annealing $\rho = 1.0000001$.

Example: Simulated annealing can be used for discrete problems as well. Here is a famous example - the travelling salesperson problem. We have n cities labelled $1, 2, \cdots, n$. The "cost" of travelling from i to j is $c(i, j)$. We assume this is symmetric. The salesperson must start in some city and make a circuit that visits every city exactly once, ending back in the starting city. We call this a tour. So a tour is a permuation π of $1, 2, \cdots, n$. We start in $\pi(1)$, go to $\pi(2)$, then to $\pi(3)$, and so on to $\pi(n)$ and finally back to $\pi(1)$. The cost of the tour is

$$H(\pi) = \sum_{i=1}^{n-1} c(\pi(i), \pi(i+1)) + c(\pi(n), \pi(1)) \tag{10.3}$$

We want to find the tour that minimizes this cost.

We can use simulated annealing with the Metropolis Hastings algorithm. We need a proposal distribution. Here is one. Let π be the current tour. We pick two distinct cities i, j uniformly at random. We interchange these two cities and reverse the original tour between i and j. Do example.

Example: We are designing integrated circuit chips. We have a very large number of "circuits". There are too many to get them all onto one chip. We have to use two chips. Certain circuits need to be connected to other circuits. These connections are "expensive" is the two circuits are on different chips. So we want to decide how to put the circuits on the two chips to minimize the number of connections that must be made between the two chips. Let m be the number of circuits. Let a_{ij} equal 1 if there has to be a connection between circuits i and j, and equal 0 if no connection is needed. It is convenient to encode the board that circuit i is on with a variable that takes on the values -1 and 1. Call the boards A and B. We let $x_i = 1$ if circuit i is on board A and $x_i = -1$ if circuit i is on board B. Note that $|x_i - x_j|$ is 1 if the two circuits are on different boards and $|x_i - x_j|$ is 0 if the two circuits are on the same board. So the total number of connections needed between the two boards is

$$\frac{1}{2} \sum_{i,j} a_{i,j} |x_i - x_j| \tag{10.4}$$

(We set $a_{ii} = 0$.) We want to choose x_i to minimize this quantity. This problem has a trivail solution - put all the circuits on the same board. We need to incorporate the constraint that this is not allowed. Note that $|\sum_i x_i|$ gives the difference in the number of circuits on the two boards. We don't need this to be exactly zero, but it should not be two large. One model would be to add a constraint that this quantity is at most ?? Another approach is to add a penalty term to the function we want to minimize:

$$\frac{1}{2} \sum_{i,j} a_{i,j} |x_i - x_j| + \alpha [\sum_{i=1}^{m} x_i]^2 \tag{10.5}$$

where α is a parameter. There is no obvious choice for α. Say something about how α should be chosen.

There are two MCMC algorithms that can be easily implemented. We could use Metropolis-Hastings. The proposal distribution is to pick a circuit i at random (uniformly). Then we change x_i to $-x_i$. The other algorithm would be a Gibbs sampler. Note that the "dimension" is the number of circuits. Explain how each step of the MCMC only involves local computations.

10.2 Estimating derivative

We recall the network example from Kroese. The times T_i are independent and uniformly distributed but with different ranges: T_i uniform on $[0, \theta_i]$. Let U_1, U_2, U_3, U_4, U_5 be independent, uniform on $[0, 1]$. Then we can let $T_i = \theta_i U_i$. The random variable we want to compute the mean of is the minimum over all paths from A to B of the total time to traverse that path.

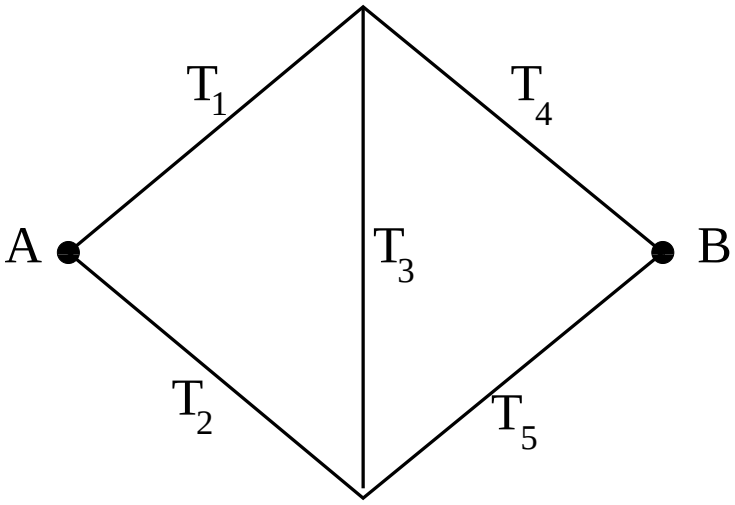

Figure 10.4: Network example. We seek the quickest path from A to B.

In the first figure we take
$\theta_2 = 2$,
$\theta_3 = 3$,
$\theta_4 = 1$,
$\theta_5 = 2$
and plot the mininum as a function of θ_1 using 10^6 samples. Two methods are used. For one we use different random numbers to generate the 10^6 samples for every choice of θ_1. For the other we use the same random numbers for different θ_1. The figure clearly shows that using "common" random numbers produces a much smoother curve. We should emphasize that the values of the min computed using the common random numbers are not any more accurate

than those computed using independent random numbers. It is just that the error go in the same direction.

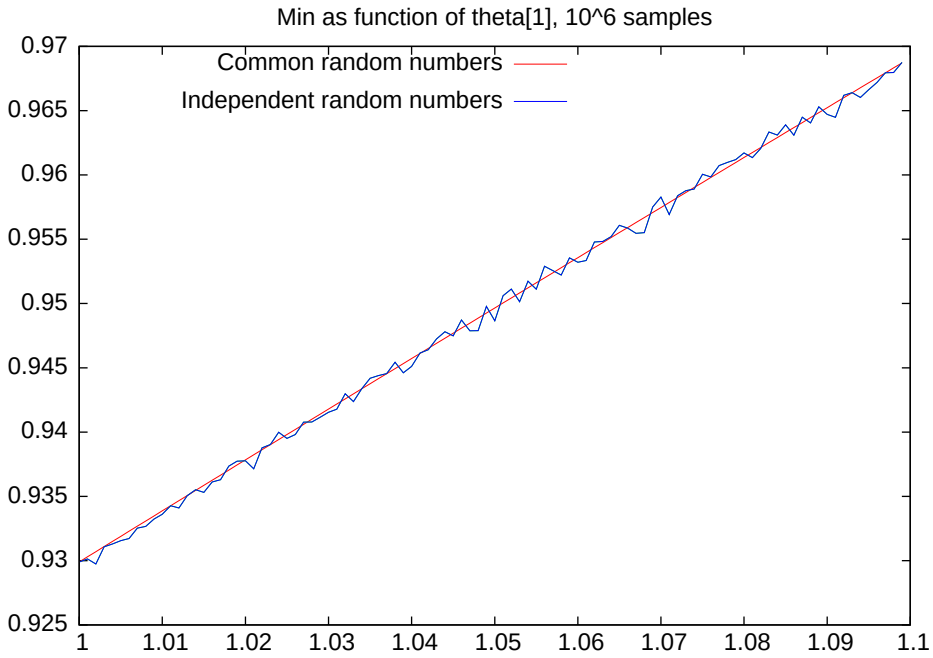

Figure 10.5: Network example. Plot of minimum as function of theta[1] using independent random numbers and common random numbers.

In figure (10.6) we compute a central difference approximation at $\theta_1 = 1$. We plot the approximation as a function of δ. The independent plots use independent samples of the random numbers. Two plots are shown, one using 10^6 samples, one with 10^7 samples. The other plot uses common random numbers. Clearly the common random number approach works much better. Starting around $\delta = 0.1$ the common random number curve starts to bend upwards, reflecting the error coming from the use of the central difference approximation. In these simulations we generate the random T_i by generating $U_1, \cdots, U_5$ uniformly in $[0, 1]$. Then we set $T_i = \theta_i U_i$.

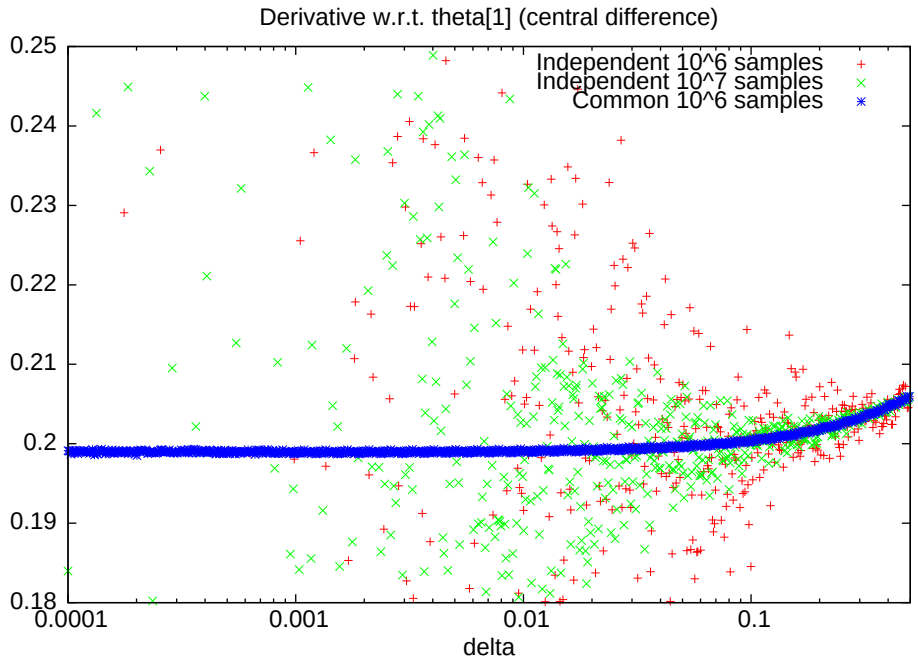

Figure 10.6: Network example. Plot of central difference approximation of derivative of minimum with respect to theta[1] using independent random numbers and common random numbers.

In figure (10.7) we continue to compute a central difference approximation to the derivative. The red and green plots are as in the previous figure. For the green plot we generate the T_i in a different way. We generate T_i using acceptance-rejection as follows. For each i we generate a random number in $[0, 5]$ and accept it when it is in $[0, \theta_i]$. We use common random numbers in the sense that we reset the random number generator to the original seed ...

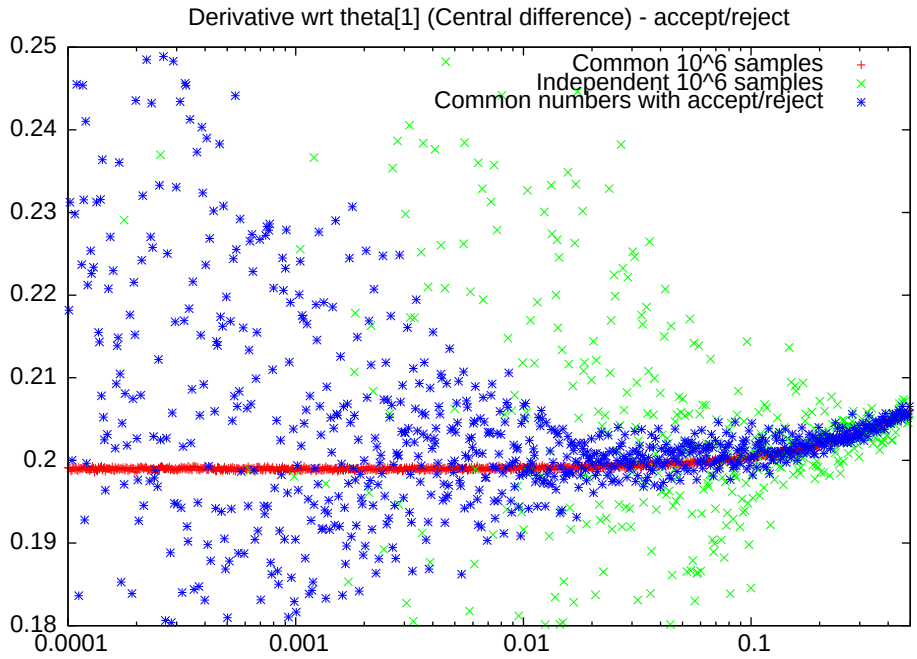

Figure 10.7: Network example. Plot of central difference approximation of derivative of minimum with respect to theta[1] using independent random numbers, good common random numbers and common numbers using accept/reject.

Finally in figure (10.8) we attempt to fix the accept-rejection approach. The idea is that common random numbers do not work well here since the number of attempts needed to accept can be different for $\theta_1 - \delta/2$ and $\theta_1 + \delta/2$. So we always generate 100 random numbers for each acceptance-rejection. Hopefully this keeps the common random numbers in sync. However, the result show in the figure is not really any better than the acceptance-rejection in figure (10.7).

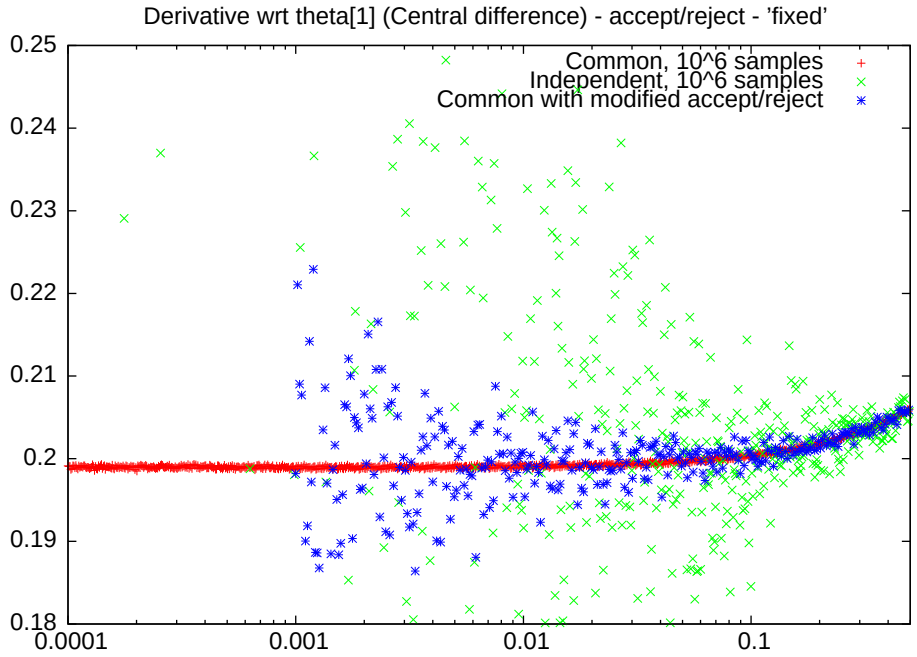

Figure 10.8: Network example. Plot of central difference approximation of derivative of minimum with respect to theta[1] using independent random numbers,good common random numbers and attemp to fix accept/reject.

10.3 Stochastic gradient descent

Chapter 11

Further topics

11.1 Computing the distribution - CDF's

Until now we have focused on computing the mean $\mu = E[X]$ of the RV X. Often we want to know the distribution of the RV. Both direct Monte Carlo and MCMC generates samples of X, and so give us information on the distribution. Most of this section will consider the case where X is a real-valued random variable, not a random vector. We briefly consider random vectors at the end of this section.

In this section we are only concerned with a non-parametric estimation of the density $f(x)$. Another approach would be a parametric approach in which we assume that $f(x)$ belongs to some multi-parameter family of distributions (normal, gamma, ...) and then estimate the parameters.

We can get a quick look at the density function $f(x)$ by plotting a histogram of our sample of X. We normalize the histogram so that the area of a rectangle is equal to the fraction of samples that fall in that bin. So the total area in the histogram is 1. Then the histogram is an approximation to $f(x)$. The histogram is easy and great for some purposes. The obvious drawbacks: it does not give a smooth function as the estimate of the density, it depends on the bin width chosen and on where we start the first bin. The dependence on the bin width is very similar to the dependence of kernel density estimation on the bandwidth which we will look at in some detail later.

Another approach is to compute the cumulative distribution function (CDF):

$$F(t) = P(X \leq t) \tag{11.1}$$

Given a sample $X_1, X_2, \cdots, X_N$, the natural estimator for the CDF is the empirical CDF

defined by

$$\hat{F}_N(t) = \frac{1}{N}\sum_{i=1}^{N} 1_{X_i \le t} \tag{11.2}$$

Note that $F(t) = E[1_{X \le t}]$, so we can think of computing the CDF as computing the means of the one parameter family of RV's $1_{X \le t}$. The usual estimator for a mean is the sample mean. For the RV $1_{X \le t}$, the sample is $1_{X_1 \le t}, 1_{X_2 \le t}, \cdots, 1_{X_N \le t}$, so this sample mean is just $\hat{F}_N(t)$. In particular we can estimate the variance of $\hat{F}_N(t)$ and put error bars on it.

Note that for a fixed t, $1_{X \le t}$ is a Bernoulli trial. It is 1 with probability $F(t)$ and 0 with probability $1 - F(t)$. If we are doing direct Monte Carlo, then the samples are independent and so the variance of $\hat{F}(t)$ for N samples is

$$var(\hat{F}_N(t)) = \frac{\sigma^2}{N} \tag{11.3}$$

where σ^2 is the variance of $1_{X \le t}$. We can estimate this variance by the sample variance of $1_{X_1 \le t}, 1_{X_2 \le t}, \cdots, 1_{X_N \le t}$. Note that since $1_{X \le t}$ only takes on the values 0 and 1, a trivial calculation shows the variance is $F(t)[1 - F(t)]$. So we can also estimate the variance σ^2 by $\hat{F}_N(t)[1 - \hat{F}_N(t)]$. A little calculation shows this is the same as using the sample variance up to a factor of $N/(N-1)$.

Assume that $F(t)$ is continuous and strictly increasing (on the range of X.) Recall that if we let $U_i = F(X_i)$, then the U_i are i.i.d. with uniform distribution on $[0, 1]$. Let

$$\hat{G}_N(u) = \frac{1}{N}\sum_{i=1}^{N} 1_{U_i \le u} \tag{11.4}$$

This is empirical CDF for the U_i and is sometimes called the reduced empirical CDF for the X_i. Note that it does not depend on F. Note that the CDF of a uniform random variable U on $[0, 1]$ is just $G(u) = P(U \le u) = u$. The Kolmogorov-Smirnov statistic is

$$D_N = \sup_t |\hat{F}_N(t) - F(t)| = \sup_u |\hat{G}_N(u) - u| \tag{11.5}$$

We collect some facts in the following propostion

Proposition 19 *If the samples come from a direct Monte Carlo then*

1. *$N\hat{F}_N(t)$ has a binomial distribution with $p = F(t)$. The central limit theorem implies that for a fixed t, $\sqrt{N}(\hat{F}_N(t) - F(t))$ converges in distribution to a normal distribution with mean zero and variance $F(t)(1 - F(t))$.*

2. *The law of large numbers immediately implies that for each t, the random variables $\hat{F}_N(t)$ converge almost surely to $F(t)$. The Glivenko-Cantelli theorem gives a much stronger result. It says that with probability one, the convergence is uniform in t.*

3. *For $x > 0$*

$$\lim_{N\to\infty} P(\sqrt{N}D_N \leq x) = \sum_{k=-\infty}^{\infty} (-1)^k e^{-2(kx)^2} \tag{11.6}$$

If our samples come from an MCMC, then the samples are not independent. It is still true that $\hat{F}_N(t)$ is the sample mean of $1_{X\leq t}$. So we can use the techniques for error bars for MCMC (e.g., batched means) to put error bars on $\hat{F}_N(t)$.

11.2 Computing the distribution - Kernel density estimation

We follow chapter 8 of the Handbook.

Given a sample $X_1, X_2, \cdots, X_N$ we want an estimator of the density $f(x)$. The crude idea is to put mass $1/N$ at each X_i and then smear it out a little to get a smooth function. More precisely, we take a symmetric function $K(x)$ which is non-negative and has integral 1. This function is called the *kernel density*. It is helpful to think of the "spread" of this function being of order 1. We also have a parameter $h > 0$, called the *bandwidth*. The function $\frac{1}{h}K((x-c)/h)$ has integral 1, is centered at c and has width of order h. The kernel density estimator is then

$$\hat{f}(x,h) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{h} K(\frac{x - X_i}{h}) \tag{11.7}$$

We have to choose both the kernel density and the bandwidth. The choice of the kernel density is not so crucial and a natural choice for the kernel density is the standard normal density. With this choice

$$\hat{f}(x,h) = \frac{1}{N}\sum_{i=1}^{N} \phi(x, X_i, h) \tag{11.8}$$

where

$$\phi(x,\mu,h) = \frac{1}{h\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2h^2}) \tag{11.9}$$

The choice of the bandwidth is the crucial choice.

We need a criterion for the optimal choice of the bandwidth. A widely studied choice is the mean integrated square error (MISE):

$$MISE(h) = E\int [\hat{f}(x,h) - f(x)]^2\, dx \tag{11.10}$$

Another choice would be

$$E\int |\hat{f}(x,h) - f(x)|\, dx \tag{11.11}$$

The MISE has the advantage that we can compute things for it. A straightforward computation shows

$$MISE(h) = \int [E\hat{f}(x,h) - f(x)]^2\, dx + \int Var(\hat{f}(x,h))\, dx \tag{11.12}$$

In the first term, $E\hat{f}(x,h) - f(x)$ is the pointwise bias in the estimator. In the second term the integrand is a pointwise variance of the estimator.

We expect that the optimal h should go to zero as $N \to \infty$, and we also expect it goes to zero more slowly than $1/N$. One might expect that it goes like N^p for some p between 0 and 1, but it is not obvious what p should be. We will find an approximation to $MISE(h)$ for small h.

For the first term in (11.12) we first use the fact that $\hat{f}(x,h)$ is a sum of identically distributed terms. So

$$\begin{aligned} E[\hat{f}(x,h)] - f(x) &= E[\phi(x,X,h)] - f(x) && (11.13)\\ &= \frac{1}{h\sqrt{2\pi}}\int e^{-(x-u)^2/2h^2} f(u)\, du - f(x) && (11.14)\\ &= \frac{1}{h\sqrt{2\pi}}\int e^{-(x-u)^2/2h^2}[f(u) - f(x)]\, du && (11.15)\\ & && (11.16) \end{aligned}$$

In the integral u is close to x so we do a Taylor expansion:

$$\begin{aligned} & \frac{1}{h\sqrt{2\pi}}\int e^{-(x-u)^2/2t}[f(u) - f(x)]\, du && (11.17)\\ \approx\ & \frac{1}{h\sqrt{2\pi}}\int e^{-(x-u)^2/2h^2}[f'(x)(u-x) + \frac{1}{2}f''(x)(u-x)^2]\, du && (11.18)\\ =\ & \frac{1}{2}f''(x)h^2 && (11.19) \end{aligned}$$

Squaring this and integrating over x gives $\frac{1}{4}h^4||f''||_2^2$. where

$$||f''||_2^2 = \int (f''(x))^2\, dx \tag{11.20}$$

For the second term in (11.12) we use the fact that $\hat{f}(x,h)$ is a sum of i.i.d. terms. So

$$Var(\hat{f}(x,h)) = \frac{1}{N} Var(\phi(x,X,h)) \tag{11.21}$$

To compute $Var(\phi(x,X,h))$ we first compute the second moment:

$$\frac{1}{h^2 2\pi} \int \exp(-\frac{(x-u)^2}{h^2})\, f(u)\, du \tag{11.22}$$

We then need to integrate this over x. The result is $\frac{1}{h2\sqrt{\pi}}$. Next we compute the first moment:

$$\frac{1}{h\sqrt{2\pi}} \int \exp(-\frac{(x-u)^2}{2h^2})\, f(u)\, du \tag{11.23}$$

We must square this and then integrate the result over x:

$$\frac{1}{h^2 2\pi} \int dx \left[\int \exp(-\frac{(x-u)^2}{2h^2}) f(u) du \right]^2 \tag{11.24}$$

Do the x integral first and we see that it will give approximately $h1_{|u-v|\leq ch}$. This leads to the term being of order 1. Note that the second moment was proportional to $1/h$ which is large. So the term from the first moment squared is neglible compared to the second moment term. So the second term in (11.12) becomes $\frac{1}{h2\sqrt{\pi}}$.

Thus for small h we have

$$MISE(h) \approx \frac{1}{4} h^4 ||f''||_2^2 + \frac{1}{2Nh\sqrt{\pi}} \tag{11.25}$$

Minimizing this as a function of h we find the optimal choice of bandwidth is

$$h^* = \left(\frac{1}{2N\sqrt{\pi}||f''||_2^2} \right)^{1/5} \tag{11.26}$$

Of course the problem with the above is that we need $||f''||_2^2$ when f is precisely the function we are trying to estimate. A crude approach is the Gaussian rule of thumb. We pretend like f has a normal distribution with mean $\hat{\mu}$ and variance $\hat{\sigma}^2$. The computation of $||f''||_2^2$ for a normal density is straightforward but a bit tedious. Obviously it does not depend on the mean of the normal. We will argue that the dependence on the standard deviation σ must be of the form $c\sigma^{-5}$. Let $f_{\sigma,\mu}(x)$ be the density of the normal with mean μ and variance σ^2. Then

$$f_{\sigma,\mu}(x) = \frac{1}{\sigma} f_{1,0}(\frac{x-\mu}{\sigma}) \tag{11.27}$$

So

$$f''_{\sigma,\mu}(x) = \frac{1}{\sigma^3} f''_{1,0}(\frac{x-\mu}{\sigma}) \tag{11.28}$$

So

$$\begin{aligned} ||f''_{\mu,\sigma}||_2^2 &= \frac{1}{\sigma^6} \int [f''_{1,0}(\frac{x-\mu}{\sigma})]^2 \, dx & (11.29) \\ &= \frac{1}{\sigma^5} \int [f''_{1,0}(u)]^2 \, du & (11.30) \end{aligned}$$

With a bit of work (which we skip) one can compute the last integral. The result is that the Gaussian rule of thumb gives the following choice of bandwidth:

$$h_G = \left(\frac{4}{3N}\right)^{1/5} \hat{\sigma} \approx 1.06 \hat{\sigma} N^{-1/5} \tag{11.31}$$

In the figures we show some results of kernel density estimation. The distribution is a mixture of two normals. Each normal has variance 1. They are centered at $\pm c$ with $c = 2$ in the first four figures and $c = 10$ in the last figure. The mixture is given by taking the normal at $-c$ with probabiliity $1/3$ and the normal at $+c$ with probability $2/3$. In all the cases we used $10,000$ samples.

In each figure we estimate the variance of the distribution using the sample variance. We then use the Gaussian rule of thumb to compute the h_G. This is labelled "optimal" in the figures. We also do kernel density estimation with
$h = h_G/16, h_G/8, h_G/4, h_G/2, h_G * 2, h_G * 4, h_G * 8, h_G * 16$.

Figure 11.1 uses values of $h_G/16$ and $h_G/8$. The estimated $\hat{f}$ follows f pretty well but with significant fluctuation.

Figure 11.2 uses values of $h_G/4$ and $h_G/2$. There values do well. In fact, $h_G/2$ does better than h_G, shown in the next figure.

Figure 11.3 uses values of h_G and $2h_G$. The value $2h_G$ does rather poorly, and even the optimal value of h_G is significanty different from the true f.

Figure 11.4 uses values of $4h_G$ and $8h_G$. These values are way too large. The estimated f does not even resemble the true f.

In figure 11.5 the centers of the two normals are quite far apart, ± 10. The figure shows the kernel density estimation with the optimal choice of h from the Gaussian rule of thumb and the estimator with h equal to the optimal value divided by 8. Clearly the latter does much better. The Gaussian rule of thumb fails badly here. Note that as we move the centers of the two Gaussians apart the variance grows, so the Gaussian rule of thumb for the optimal h increases. But the optimal h should be independent of the separation of the two Gaussians.

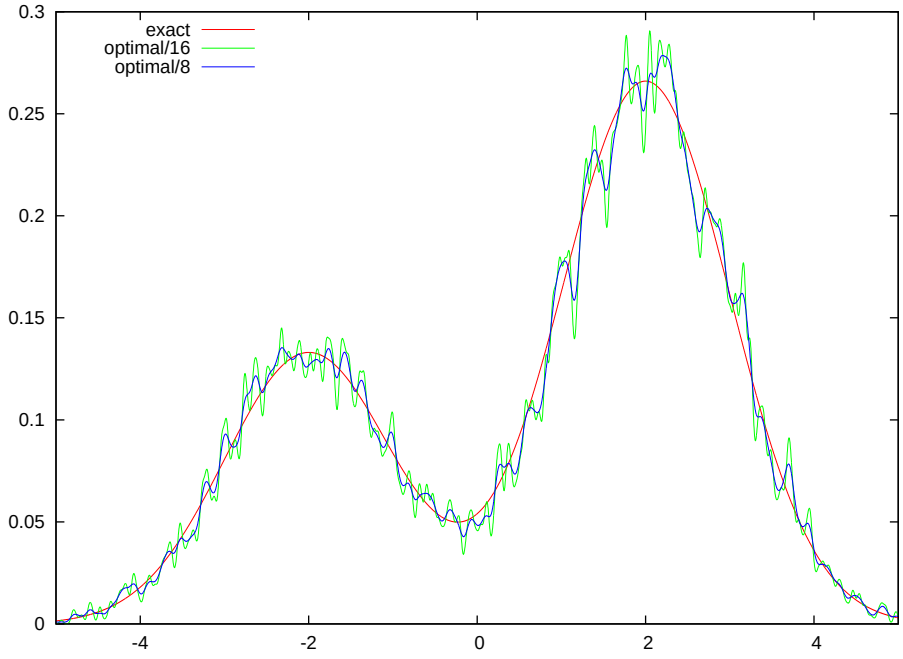

Figure 11.1:

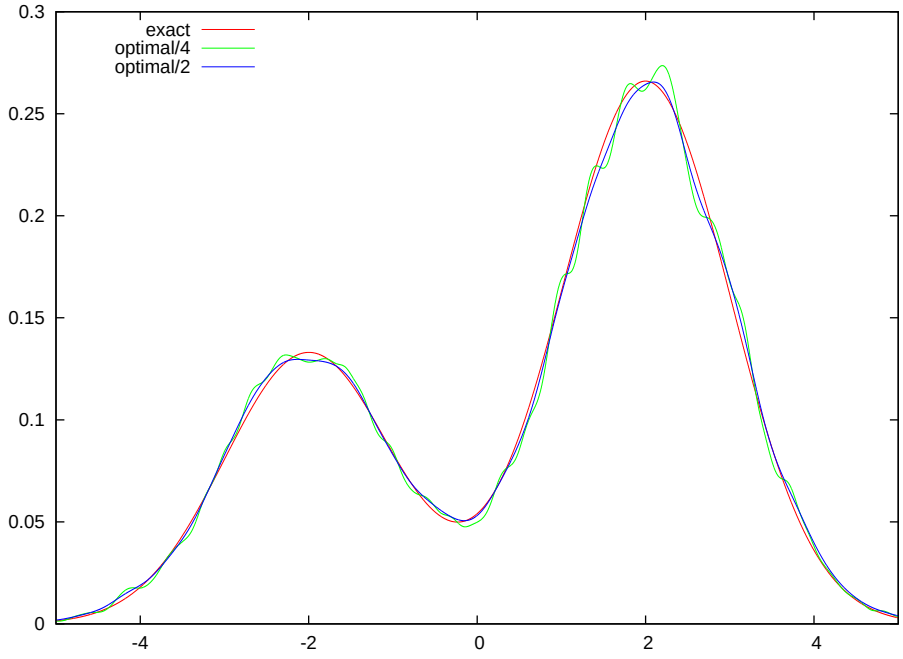

Figure 11.2:

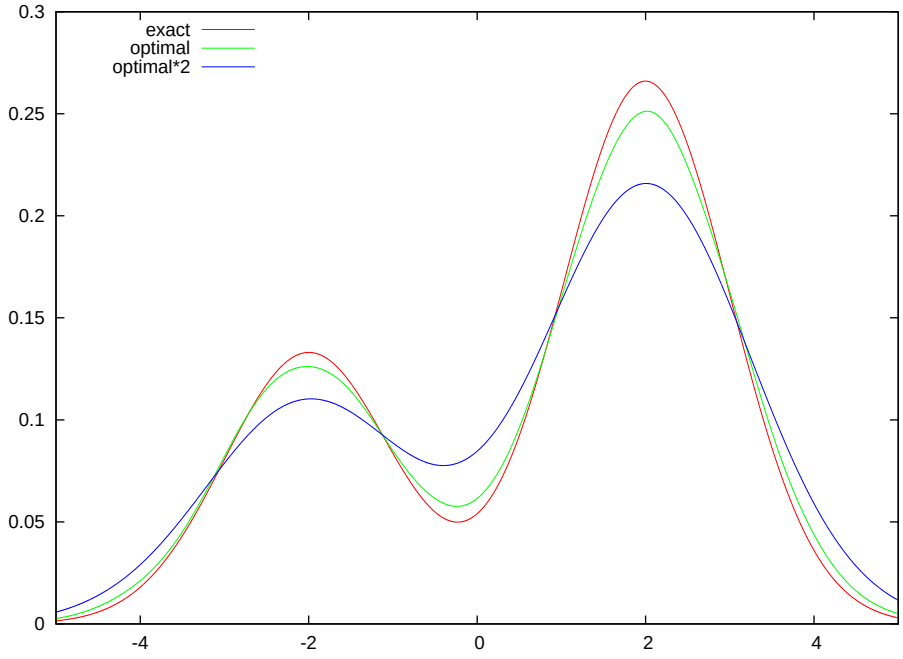

Figure 11.3:

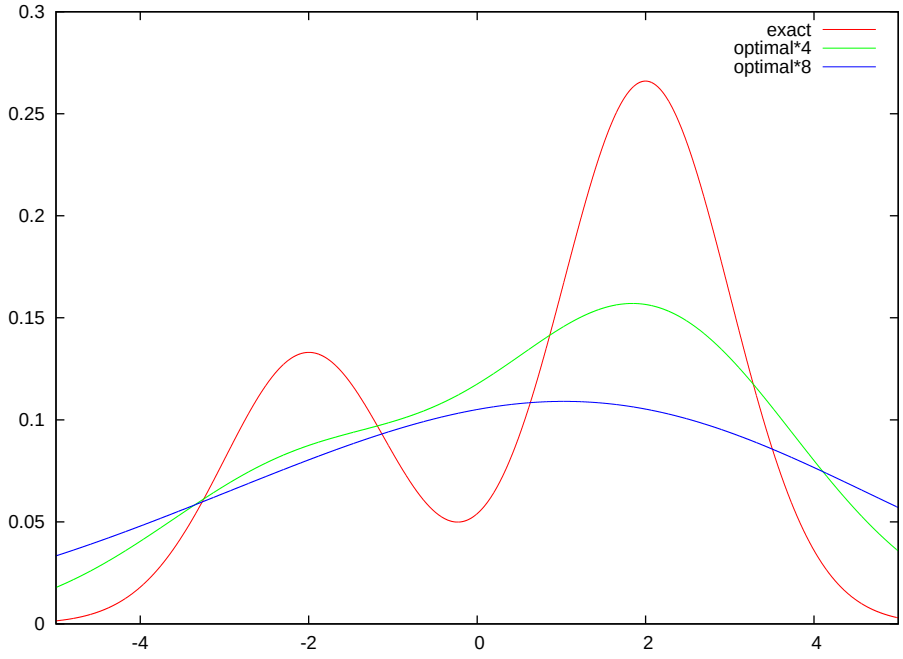

Figure 11.4:

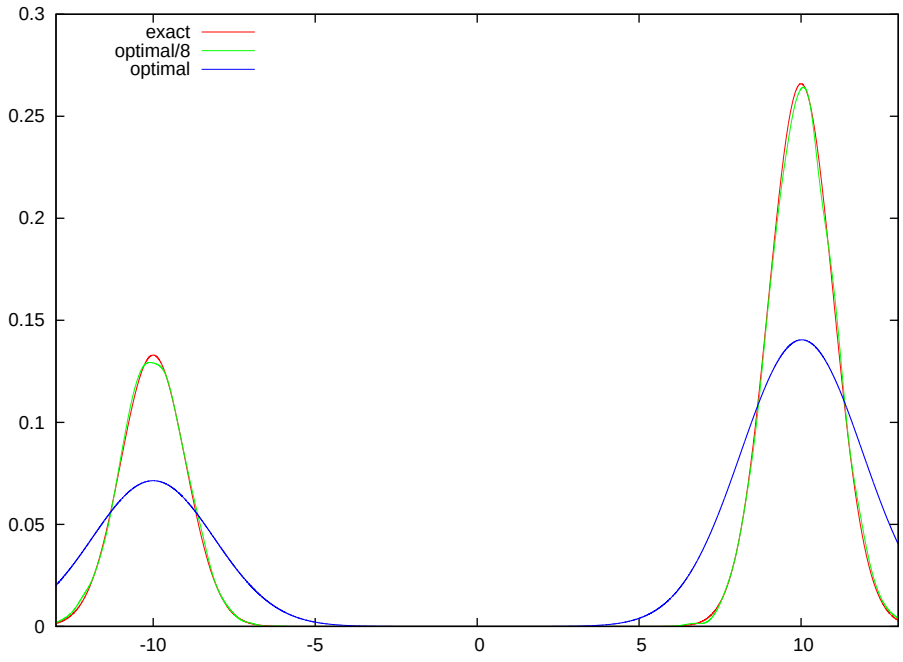

Figure 11.5:

We now consider multivariate kernel density estimation, i.e., estimating the density function for a random vector with dimension d. The obvious generalization of the estimator is

$$\hat{f}(x,h) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{h^d} K(\frac{x - X_i}{h}) \tag{11.32}$$

where $K(x)$ is now a non-negative function on R^d with integral 1.

There are some new issues in the multi-variate case. The different components of X may have vastly different scales. The kernel function needs to take this into account. This problem can occur in a more subtle way. Consider a two dimensional distribution in which the support of the density is a long narrow ellipsoid at an angle of 45 degrees with respect to the coordinates axes.

The problem of the components living on different scales can be dealt with by "pre-scaling." We rescale each component of the data so that they are on roughly the same scale. Then we do the kernel density estimation and then we rescale the resulting estimator.

The more subtle problem can be dealt with by "pre-whitening." Note that the covariance matrix of the data is far from the identity. We find a linear transformation of the data so that the covariance matrix of the new data is approximately the identity.

One possible multivariate kernel is to use a product form.

$$K(x,X,h) = \prod_{j=1}^{d} \frac{1}{h_j} k(\frac{x_j - X_j}{h_j}) \tag{11.33}$$

where $k(x)$ is a univariate kernel density. We can use a different h_j for each direction to account for different scale for the different directions. The h_j can be found as in the 1-d case.

11.3 Sequential monte carlo

We follow section 14.1 of the Handbook. For rigorous proof see the article "Particle Filters - A Theoretical Perspective" in *Sequential Monte Carlo Methods in Practice*, A. Doucet et al. (eds.).

11.3.1 Review of weighted importance sampling

We first review some things.

We want to compute $\mu = E[f(\vec{X})]$. Let $p(\vec{X})$ be the density of $\vec{X}$. We cannot sample from $p(\vec{X})$. But we can sample from $q(\vec{X})$ which is close to p in some sense. The importance sampling algorithm is as follows. Generate samples $\vec{X}_1, \cdots, \vec{X}_n$ according to the distribution $q(x)$. Then the estimator for μ is

$$\hat{\mu}_q = \frac{1}{n} \sum_{i=1}^{n} \frac{f(\vec{X}_i) p(\vec{X}_i)}{q(\vec{X}_i)} \tag{11.34}$$

We can think of this importance sampling Monte Carlo algorithm as just ordinary Monte Carlo applied to $E_q[f(\vec{X})p(\vec{X})/q(\vec{X})]$. $\hat{\mu}_q$ is an unbaised estimator of μ, i.e., $E_q\hat{\mu}_q = \mu$, and it converges to μ with probability one.

Suppose $p(x) = c_p p_0(x)$ where $p_0(x)$ is known, but c_p is unknown. And suppose we can sample from $q(x)$, but $q(x) = c_q q_0(x)$ where $q_0(x)$ is known and c_q is unknown. Then we can still do self-normalized or weighted importance sampling. The key observation is

$$\int f(x)p(x)dx = \frac{E_q[f(x)w(x)]}{E_q[w(x)]} \tag{11.35}$$

where $w(x) = p_0(x)/q_0(x)$ is a known function.

The self-normalized importance sampling algorithm is as follows. We generate samples $\vec{X}_1, \cdots, \vec{X}_n$ according to the distribution $q(x)$. Our estimator for $\mu = \int f(x)p(x)dx$ is

$$\hat{\mu}_{WI} = \frac{\sum_{i=1}^{n} f(\vec{X}_i) w(\vec{X}_i)}{\sum_{i=1}^{n} w(\vec{X}_i)} \tag{11.36}$$

where the WI subscript indicates this is the estimator coming from weighted importance sampling.

One way in which weighted importance sampling can do poorly is that the weighted are unbalanced, i.e., most of them are very small and only a few contribute to the overall weight. One measure of this is the effective sample size given by

$$\frac{(\sum_i w_i)^2}{\sum_i w_i^2} \tag{11.37}$$

where $w_i = w(\vec{X}_i)$.

11.3.2 Resampling

Before we discuss sequential monte carlo, we first consider the "resampling" that is part of the algorithm.

We want to compute $\mu = E[f(\vec{X})]$ using weighted importance sampling as above. We generate N samples $\vec{X}_1, \cdots, \vec{X}_N$ according to the density $q(x)$ and then approximate μ by

$$\hat{\mu}_{WI} = \sum_{i=1}^{N} p_i f(\vec{X}_i) \tag{11.38}$$

where p_i are the normalized importance sampling weights:

$$p_i = \frac{w_i}{\sum_{j=1}^{n} w_j} \tag{11.39}$$

Note that we can think of this as the integral of f with respect to the measure

$$\sum_{i=1}^{N} p_i \delta_{X_i}(x) \tag{11.40}$$

Resampling means that we replace this measure by

$$\frac{1}{N} \sum_{i=1}^{N} N_i \delta_{X_i}(x) \tag{11.41}$$

where the N_i are non-negative integers whose sum is N. So the new estimator for μ is

$$\hat{\mu}_R = \frac{1}{N} \sum_{i=1}^{N} N_i f(X_i) \tag{11.42}$$

There are different methods for choosing the N_i.

The simplest method is to draw N independent samples from the discrete density (11.40). This means the joint distribution of $N_1, N_2, \cdots, N_N$ is a multinomial distribution with N trials and probabilities $p_1, p_2, \cdots, p_N$. Note that many of the N_i will be zero. We would like to know that $\hat{\mu}_R$ converges to μ as $N \to \infty$. This is subtle. Note that the N_i are not independent.

We first show it is an unbiased estimator.

$$E[\hat{\mu}_R] = \frac{1}{N} \sum_{i=1}^{N} E[N_i f(X_i)] \tag{11.43}$$

We compute using the tower property:

$$\begin{aligned} E[N_i f(X_i)] &= E[E[N_i f(X_i) | X_1, \cdots, X_N]] & (11.44) \\ &= E[f(X_i) E[N_i | X_1, \cdots, X_N]] = E[f(X_i) N p_i] & (11.45) \end{aligned}$$

So

$$E[\hat{\mu}_R] = \sum_{i=1}^{N} E[f(X_i) p_i] = \mu \tag{11.46}$$

Next we show that for bounded functions f, the estimator converges to μ in the L^2 sense (and hence in probability). It converges a.s., but we do not prove this. See the article by Cristan. We already know that the estimator from weighted importance sampling converges to μ with probability one, i.e., $\hat{\mu}_{WI} \to \mu$ with probability one. Since f is bounded the bounded convergence theorem implies we also have convergence in L^2. So it suffices to show $\hat{\mu}_R - \hat{\mu}_{WI}$ converges to 0 in L^2, i.e., we need to show $E[(\hat{\mu}_R - \hat{\mu}_{WI})^2] \to 0$. Note that

$$\hat{\mu}_R - \hat{\mu}_{WI} = \frac{1}{N} \sum_{i=1}^{N} f(X_i)(N_i - p_i N) \tag{11.47}$$

We use conditioning again:

$$E[(\hat{\mu}_R - \hat{\mu}_{WI})^2] = E[E[(\hat{\mu}_R - \hat{\mu}_{WI})^2 | X_1, \cdots, X_N]] \tag{11.48}$$

We have

$$E[(\hat{\mu}_R - \hat{\mu}_{WI})^2 | X_1, \cdots, X_N]] \tag{11.49}$$

$$= \frac{1}{N^2} \sum_{i,j=1}^{N} f(X_i) f(X_j) E[(N_i - p_i N)(N_j - p_N) | X_1, \cdots, X_N] \tag{11.50}$$

Conditioned on $X_1, X_2, \cdots, X_N$, the p_i are constant and the N_i follow a multinomial distribution. So we can compute

$$E[(N_i - p_i N)(N_j - p_N) | X_1, \cdots, X_N] = -N p_i p_j \tag{11.51}$$

So

$$|E[(\hat{\mu}_R - \hat{\mu}_{WI})^2 | X_1, \cdots, X_N]]| \leq \frac{1}{N} \sum_{i,j=1}^{N} |f(X_i) f(X_j)| p_i p_j \leq \frac{||f||_\infty^2}{N} \tag{11.52}$$

Thus

$$E[(\hat{\mu}_R - \hat{\mu}_{WI})^2] = E[E[(\hat{\mu}_R - \hat{\mu}_{WI})^2 | X_1, \cdots, X_N]] \tag{11.53}$$

$$\leq E[|E[(\hat{\mu}_R - \hat{\mu}_{WI})^2 | X_1, \cdots, X_N]|] \leq \frac{||f||_\infty^2}{N} \to 0 \tag{11.54}$$

There are other ways to define the N_i. The definition above is not ideal because it introduces a fair amount of variance into the problem. Suppose we have a relatively small set of indices for which $p_i N$ is relatively large and the rest of the $p_i N$ are close to zero. The above procedure will replace the one copy for index i with N_i copies where the mean of N_i is $p_i N$ but the standard deviation is of order $\sqrt{N_i}$. It might be better to take the number of copies to be $p_i N$ rounded to the nearest integer. The following algorithm does something in this spirit.

Stratified resampling Let n_i be the largest integer less than or equal to $p_i N$. Note that the sum of the n_i cannot exceed N. Create c_i copies of X_i. We are still short $N_r = N - \sum_i n_i$ samples. Draw a sample of size N_r from $\{1, 2, \cdots, N\}$ uniformly and without replacement. Add these X_i to the previous ones. Finally, if it makes you feel better you can do a random permutation of our sample of size N. (What's the point?)

11.3.3 sequential MC

Now suppose that instead of a random vector we have a stochastic process $X_1, X_2, X_3, \cdots$. We will let X stand for $X_1, X_2, X_3, \cdots$. We want to estimate the mean of a function of the process $\mu = f(X)$. It doesn't make sense to try to give a probability density for the full infinite process. Instead we specify it through conditional densities:
$p_1(x_1), p_2(x_2|x_1), p_3(x_3|x_1, x_2), \cdots, p_n(x_n|x_1, x_2, \cdots, x_{n-1}), \cdots$. Note that it is immediate from the definition of conditional density that

$$\begin{aligned} p(x_1, x_2, \cdots, x_n) &= p_n(x_n|x_1, x_2, \cdots, x_{n-1}) p_{n-1}(x_{n-1}|x_1, x_2, \cdots, x_{n-2}) && (11.55) \\ &\cdots \quad p_3(x_3|x_1, x_2) p_2(x_2|x_1) p_1(x_1) && (11.56) \end{aligned}$$

We specify the proposal density in the same way:

$$\begin{aligned} q(x_1, x_2, \cdots, x_n) &= q_n(x_n|x_1, x_2, \cdots, x_{n-1}) q_{n-1}(x_{n-1}|x_1, x_2, \cdots, x_{n-2}) && (11.57) \\ &\cdots \quad q_3(x_3|x_1, x_2) q_2(x_2|x_1) q_1(x_1) && (11.58) \end{aligned}$$

So the likehood function is

$$w(x) = \prod_{n \geq 1} \frac{p_n(x_n|x_1, x_2, \cdots, x_{n-1})}{q_n(x_n|x_1, x_2, \cdots, x_{n-1})} \tag{11.59}$$

An infinite product raises convergence questions. But in applications f typically either depends on a fixed, finite number of the X_i or f depends on a finite but random number of the X_i. So suppose that f only depends on $X_1, \cdots, X_M$ where M may be random. To be more precise we assume that there is a random variable M taking values in the non-negative integers such that if we are given that $M = m$, then $f(X_1, X_2, \cdots)$ only depends on $X_1, \cdots, X_m$. So we can write

$$f(X_1, X_2, \cdots) = \sum_{m=1}^{\infty} 1_{M=m} \, f_m(X_1, \cdots, X_m) \tag{11.60}$$

We also assume that M is a stopping time. This means that the event $M = m$ only depends on $X_1, \cdots, X_m$. Now we define

$$w(x) = \sum_{m=1}^{\infty} 1_{M=m}(x_1, \cdots, x_m) \prod_{n=1}^{m} \frac{p_n(x_n|x_1, x_2, \cdots, x_{n-1})}{q_n(x_n|x_1, x_2, \cdots, x_{n-1})} \tag{11.61}$$

MORE Explain the potential problem that the weights can degenerate to the point that most are zero.

The final step is to modify the above by resampling at each time step. So the sequential Monte Carlo algorithm is as follows. Throughout N will be the number of samples. They are often called "particles." (There are variants where the number of particles changes with time, but we only consider an algorithm where the number stays constant.)

1. Initialize. Given N iid samples $X_1^1, \cdot, X_1^N$ from $q_1()$. The subscript 1 means $t = 1$.

2. Importance sampling. Given $X_{1:t-1}^1, \cdots, X_{1:t-1}^N$, generate (independently) Y_t^j from $q_t(\cdot|X_{1:t-1}^j)$. The Y_t^j are conditionally independent given $X_{1:t-1}^1, \cdots, X_{1:t-1}^N$, but not independent. Compute the weights

$$w_{t,j} = \frac{p_t(Y_t^j|X_{1:t-1}^j)}{q_t(Y_t^j|X_{1:t-1}^j)} \tag{11.62}$$

and then let $p_{t,j}$ be the normalized weights:

$$p_{t,j} = \frac{w_{t,j}}{\sum_{i=1}^N w_{t,j}} \tag{11.63}$$

3. Resample. Let $Z_{1:t}^j = (X_{1:t-1}^j, Y_t^j)$. Generate $X_{1:t}^j$ by independently sampling (N times) from the discrete distribution which has the values $Z_{1:t}^j$ with probability $p_{t,j}$. So we are drawing N independent samples from the mixture

$$\sum_{j=1}^N p_{t,j}\delta_{Z_{1:t}^j} \tag{11.64}$$

We do steps 2 and 3 for $t = 2, \cdots, T$ where T could be a random stopping time.

Example - random walk: We look at a one-dimensional random walk which only takes steps of ± 1. The probability of going right is p. For a proposal distribution we use a symmetric random walk which goes right or left with probability 1/2. We run the walk for 100 time steps. The random variable we study is the position of the walk at the end of the 100 steps. Note that we know μ exactly. It is $100(2p-1)$. We do two simulations, each with $2,000$ samples. One simulation is weighted importance sampling. The other is sequential MC using the multinomial resampling. Figure 11.6 shows the two estimators $\hat{\mu}_{WI}$ and $\hat{\mu}_R$ as a function of p, along with the exact result.

To see if the breakdown for the weighted importance sampling simulation somes from the weights becomes very unbalanced we plot the effective sample size as a function of time for

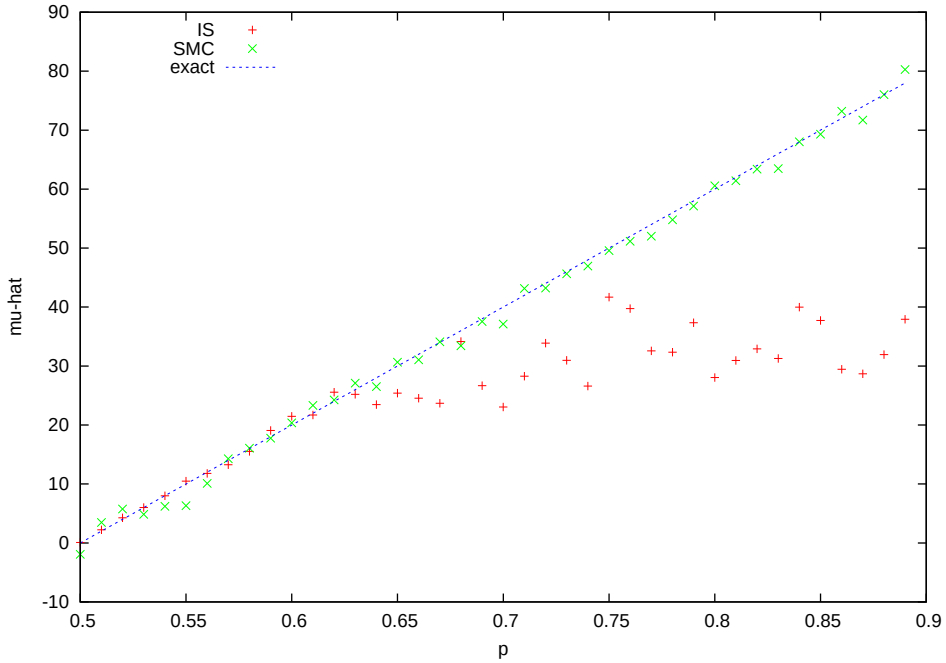

Figure 11.6:

several values of p in figure 11.7. The figure shows that for all values of p the effective sample size usually decreases with time. The rate of decrease gets larger as p moves away from 1/2.

In figure 11.8 we plot the effective sample size at time 100 as a function of p. Note that the value of p where the effective sample size becomes small corresponds with the value of p in figure 11.6 where the estimator $\hat{\mu}_R$ deviates significatly from μ.

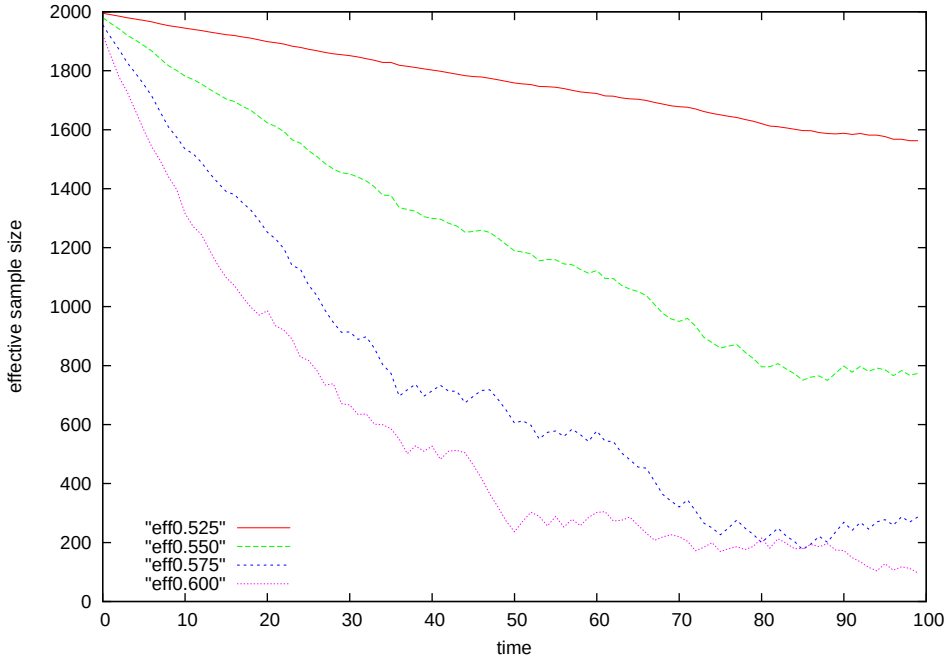

Figure 11.7:

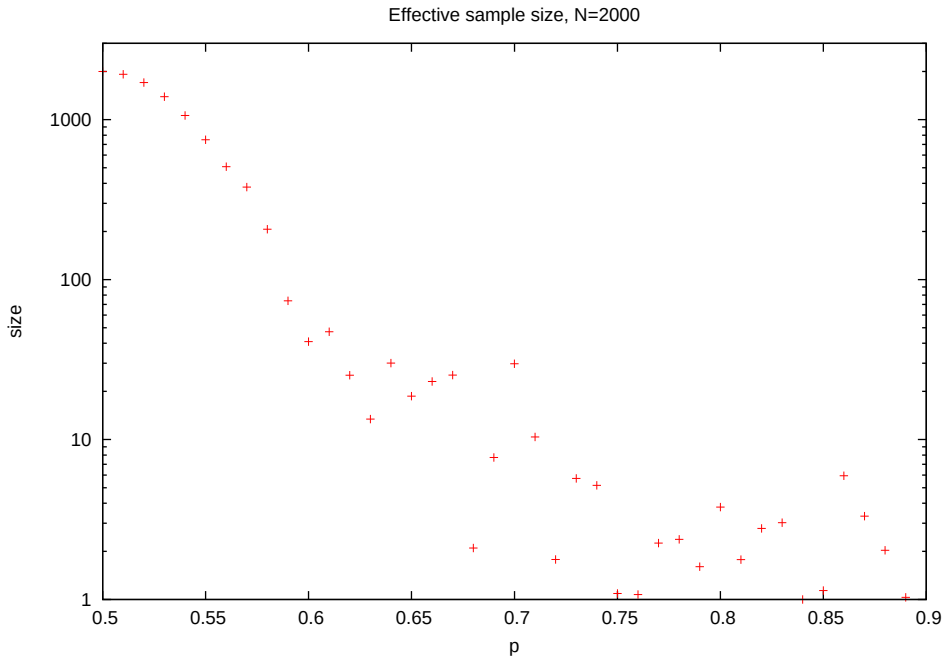

Figure 11.8: