Course Notes for STAT 8701[1]
Computational Statistical Methods

Galin L. Jones

School of Statistics

347 Ford Hall

galin@stat.umn.edu

Draft: April 17, 2007

[1]Acknowledgment: Some of these notes have been adapted from other sets of course notes created by Gary Oehlert and Charlie Geyer.

Contents

Chapter 1

Introduction to R

An Introduction to R claims that "R is an integrated suite of software facilities for data manipulation, calculation and graphical display." What this means is that we can use R in a variety of ways. For example, any computation that you want to do can be programmed in R. However, for most people the strength of R is that it has many statistical and graphical capabilities.

A major benefit of R is that it is free software (both in the sense of "free beer" and in the sense of "free speech") and may be used on a variety of platforms. R is similar to the commercial product S-Plus. Nearly everything you can do in R can be done in S-Plus and vice versa.

Rweb is a web interface to the R package. The Rweb commands are identical to R commands. If you are going to use R in any serious fashion, you should install it on your own computer or use the machines in the computer lab instead of using the web interface. Rweb is really only appropriate for teaching purposes. Rweb should not be used to do computing assignments for this course.

In the rest of this chapter we will cover some programming concepts focusing on just enough of the basics of R, C and LaTeX to produce a simple

R package. It is impossible to give a thorough introduction to even one of these topics in a single chapter. But maybe I can provide enough information to get a novice started. Learning R or C or LaTeX is a lot like learning statistics or anything else for that matter–you've got to use it to learn it. More substantial documentation for R can be found on-line at `http://www.r-project.org/index.html`. There are two main documents of interest there: *An Introduction to R* and *Writing R Extensions*. You can also find these documents on the course web page. If you are new to R you should look at this documentation as soon as possible. Finally, the main pieces of the sample code in this chapter can be found under "Examples" on the course web page.

1.1 Using R

The first thing to figure out is how to start R. For data analysis it is most common to use R interactively. Simply type `R` at the prompt and it will start. Throughout the shell prompt will be `>`. If you are an emacs user then use the commands C-u M-x R to start an interactive R session. To quit R just type `q()` at the prompt.

For programming projects it is more common to use the batch mode. To do this simply write an R program in your favorite text editor and save it as `myfile.R` then at the prompt type `R CMD BATCH myfile.R`.

1.2 Objects

Almost everything in R is an object. Objects can hold a collection of items and some can contain several different types of data.

- The most common object is a *vector*. These contain either numbers,

logicals or character strings. Another common object is a *matrix*. Both vectors and matrices contain only one type of item.

- *Data frames* look like a matrix but may have different item types in distinct columns.
- A *list* is an ordered sequence of objects which may contain any sort of object including another list.

Objects need to have a name. Names can be nearly any combination of letters, numbers and the period. At least one letter must appear before the first number, names are case sensitive and underscores are not allowed. Names are given via the assignment operator `->`. Thus, if at the prompt I type `d<-12` I will have created an object named `d` which is a numeric vector of length 1. We can also create logical vectors

```
> lv <-d > 34
> lv
[1] FALSE
```

and character vectors

```
> cv<-"x"
> cv
[1] "x"
```

R objects have modes and attributes.

```
> df<-data.frame(x=c("galin", "gators", "gophers"),y=rnorm(3))
> df
        x         y
1   galin -0.676810
2  gators  0.723919
```

```
3 gophers -2.105784
> mode(df)
[1] "list"
> length(df)
[1] 2
> attributes(df)
$names
[1] "x" "y"
$row.names
[1] "1" "2" "3"
$class
[1] "data.frame"
```

1.3 A Sample Interactive R Session

```
> library(MASS)
> data(geyser)
> names(geyser)
names(geyser)
[1] "waiting"  "duration"
> attach(geyser)
> summary(duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8333  2.0000  4.0000  3.4610  4.3830  5.4500
> mean(duration, trim=.10)
[1] 3.499239
```

1.4 Basic Programming

Before we look at programming in R and C, I have a few words about style. These comments apply to *every* program written for this course. Anytime you write a program you should strive for clarity and readability over cleverness. Extensive comments are a must as well as using variable names that make sense. That is, a variable named `iteration` is preferred to `i`. Also, a consistent indentation scheme should be followed rigorously.

1.4.1 How to Write a Bad R Program

We will have occasion to use R for more than simple data analysis. In particular, we will need to be able to write programs to perform sophisticated computations. R can do a lot of this but, unfortunately, it is awful for doing computations involving loops such as those encountered in Markov chain Monte Carlo (which we will see later in the course).

In *An Introduction to R* you can find the following statement: "Code that takes a 'whole object' view is likely to be faster in R." Lets see what this means in an example which computes the inner product of two vectors, `a` and `b`. First, a bad way then a better way.

```
for (i in 1:n){
 d[i] <-a[i]*b[i]
}
s<-0
for(i in 1:n){
 s<-s+d[i]
}
```

Now a better way

```
s<-sum(a*b)
```

1.4.2 Functions in R

A function is just a group of reusable commands. There are many existing R functions and it is easy (but not always helpful) to look at them.

```
> cos
.Primitive("cos")
> rnorm
function (n, mean = 0, sd = 1)
.Internal(rnorm(n, mean, sd))
<environment: namespace:stats>
```

`.Primitive` returns an entry point to an internally implemented function while `.Internal` performs a call to an internal code. This isn't something we will worry about in this course.

```
> summary
function (object, ...)
UseMethod("summary")
<environment: namespace:base>
```

So `summary` contains `UseMethod("summary")`. If we do `summary(df)` we will get to the `UseMethod` command. In this case R looks for the class of the first argument. Then it checks for a function `summary.data.frame`. If this exists it calls `summary.data.frame(df)`. If not it calls `summary.default(df)`. Thus we see that `summary` is a generic function.

We can write our own functions for use in an R program. This can be useful when you have occasion to do the same operation multiple times in a single program. Here is an example.

```
>#This function sums the squared elements of a vector.
> sum.sq<-function(u){
```

```
    ssq<-t(u)%*%u
    ssq
}
> sum.sq(1:10)
[1] 385
```

1.4.3 C Programming

As with everything else in this chapter, I do not aim to present a comprehensive introduction to the C language. But maybe I can get you started. Basically R is an interpreted language but C is compiled. That is, every time an R command is entered the R system parses the command into bits. It then acts on those bits. It has to do this every time a command is entered. On the other hand, compiled code is generally faster and more efficient than an interpreted language. In the rest of this section we will look at a simple C program and then cover the commands required to compile it and execute it as a standalone program. In the next section we will see how to use C within R.

In your favorite text editor put the following simple C program that asks the user to enter a number and then prints the square of that number.

```
/*[File: myfirst/myfirst.c]*/
/************************************************************
 *  myfirst -- program to print the square of an integer.   *
 *             This program is used to illustrate writing   *
 *             a C program.                                 *
 * Author: Galin Jones                                      *
 * Usage: Demonstration of a simple C program.              *
 * Last modified: 29/12/2005                                *
 ************************************************************/
```

```
#include <stdio.h>    /*contains function declarations*/

int main()            /*returns an integer*/
{
  double user_in;     /*user input*/

  /*Ask for an integer.*/
  printf("Enter a number: ");
  scanf("%lf", &user_in);

  /*Print the square of the number*/
  printf("The square is %f\n", user_in * user_in);

  return(0);          /*exit normally*/
}
```

Save this as `myfirst.c` then at the prompt type

```
> gcc -o myfirst myfirst.c -lm
> ./myfirst
Enter a number: 3
The square is 9.000000
```

Here `gcc` is the C compiler provided by the Free Software Foundation–see the Gnu link on the course web page. The `-o` switch tells the compiler that the program is to be called `myfirst` and the source file is `myfirst.c`. Other switches that are helpful include `-g` which turns on debugging and `-Wall` which turns on the warnings. (These will be helpful in more complicated programs.) The `-lm` switch is required since our function does mathematics. Thus you could also use

```
> gcc -g -Wall -o myfirst myfirst.c -lm
```

```
> ./myfirst
Enter a number: 3
The square is 9.000000
```

1.4.4 Calling C from R

A useful feature of R is that you can call C code within it. There are two ways to do this. One is with the command `.Call` and another is with the command `.C`. We will focus on `.Call`. Charlie Geyer has a web page (`http://www.stat.umn.edu/~charlie/rc/`) which covers using `.C`.

We start by writing a C function that sums the elements of a vector and saving it as `vecSum.c`

```
#include <R.h>
#include <Rinternals.h>
#include <Rmath.h>

SEXP vecSum(SEXP Rvec){
  int i, n;
  double *vec, value = 0;
  vec = REAL(Rvec);
  n = length(Rvec);
  for (i = 0; i < n; i++) value += vec[i];
  printf("The value is: %4.6f \n", value);
  return R_NilValue;
}
```

- `SEXP` stands for "S expression"
- If you don't want the function to return anything use `return R_NilValue;`

- The statement `vec = REAL(Rvec)` defines a pointer to the real part of `Rvec` which is useful so that we can type `vec[0]` instead of `REAL(Rvec)[0]`

Then at the prompt type `R CMD SHLIB vecSum.c` and it is ready to be used in R.

```
> dyn.load("vecSum.so")
> .Call("vecSum", rnorm(10))
The value is: -0.683804
NULL
```

We can use `vecSum.c` in R functions. For example, I find it easier to do error checking and type coercion in R than in C. So we could write an R function that wraps around the call to the C code.

```
vecSum <- function(vec){
  if (!is.vector(vec))
  stop("vec must be a vector")
  if (!is.real(vec)) vec <-as.real(vec)
  .Call("vecSum",vec)
}
```

1.5 Making an R Package

An R package is the standard way to collect and distribute R code. The *Writing R Extensions* document gives all of the gory details for making a package. Here is a streamlined version.

1. Make a directory with the same name as the package of interest. Ours will be `vecSum`. In this directory will be some files and further subdirectories.

2. We begin by creating the file `DESCRIPTION`.

```
Package: vecSum
Version: 1.0
Date:  2005-12-09
Title:  Example R package using vecSum
Author:  Galin L. Jones <galin@stat.umn.edu>
Maintainer:  Galin L. Jones <galin@stat.umn.edu>
Description:  This is a simple package that illustrates building a package.
License: GPL version 2 or newer
```

3. The next administrative file is `INDEX`.

```
vecSum  Adds the elements of a vector.
```

4. Now create 3 subdirectories: `R`, `src` and `man`.

5. In the `R` subdirectory we have the source of the R function.

```
vecSum <- function(vec){
  if (!is.vector(vec))
  stop("vec must be a vector")
  if (!is.real(vec)) vec <-as.real(vec)
  .Call("vecSum",vec)
}

.First.lib <- function(lib, pkg){
    library.dynam(pkg, pkg, lib)
  }
```

This is just the R function from the last section with another function `.First.lib` which gets called with the location of the library and the

package name. `library.dynam` has arguments of the object to load (without its suffix), the name of the package and the location of the library where the package is.

6. The subdirectory `src` contains the C source code. (An R package does not require C source code but it will often have it.) In this case it is in a file named vecSum.c.

```
#include <R.h>
#include <Rinternals.h>
#include <Rmath.h>

SEXP vecSum(SEXP Rvec){
  int i, n;
  double *vec, value = 0;
  vec = REAL(Rvec);
  n = length(Rvec);
  for (i = 0; i < n; i++) value += vec[i];
  printf("The value is: %4.6f \n", value);
  return R_NilValue;
}
```

7. The subdirectory `man` contains some documentation in a file ending with the suffix `.Rd`. This one is called vecSum.Rd.

```
\name{vecSum}
\alias{vecSum}
\keyword{arith}
\title{Sum of a vector}
\description{Compute the sum of the elements of a vector}
\usage{vecSum(x)}
\arguments{
```

```
  \item{x}{A numeric object, no missing values allowed.}
}
\value{Prints the result of adding the elements of a vector.}
\examples{
x<-1:10
vecSum(x)
}
```

8. Now check the library. In the directory that contains the package type `R CMD check vecSum`. This will generate a long list of checks. Make sure to fix any problems.

9. Install the package into a library. A library is a directory that holds packages. Move to the directory that holds the package (here ∼`/8701`) and type

   ```
   R CMD INSTALL -l ~/8701/myRlibray vecSum
   ```

   And, you're done!

To use the package from R:

```
> library("vecSum",lib.loc="~/8701/myRlibrary/")
> vecSum(rnorm(10))
The value is: -0.924766
NULL
```

1.6 Reproducible Research

One of the main points of scientific research is to share it. But sharing is not enough, it should also be reproducible–both by yourself and others.

(Believe it or not, traditional methods of scientific publication do not really encourage this.) What I mean is that even after several years have passed you (or anyone with access to your files) should be able to perfectly reproduce everything in a paper including figures and tables. Notice that simply providing comments in a computer program will not accomplish this; however, many researchers do not even do this which renders their programs useless.

1.6.1 LaTeX

LaTeX is a typesetting system that is used to produce scientific papers in many disciplines. Lets look at a simple LaTeXfile.

```
\documentclass[12pt]{article}

\usepackage{amsbsy,amsmath,amsthm,amssymb}

\usepackage[sort,longnamesfirst]{natbib}
\newcommand{\pcite}[1]{\citeauthor{#1}'s \citeyearpar{#1}}
\usepackage{geometry}
\geometry{hmargin=3cm,vmargin={2.25cm,2.25cm},nohead,footskip=0.5in}

\renewcommand{\baselinestretch}{1.66}
\setlength{\baselineskip}{0.3in} \setlength{\parskip}{.05in}

\usepackage[dvips]{changebar}

\begin{document}
Using \LaTeX, I plan to rule the world.  Care to join my cause?  The
key will be the equation
\[
a^{2} + b^{2} = c^{2} \; .
```

```
\]
A labeled equation is different
\begin{equation}
\label{eq:key}
\frac{\mu}{\sigma} \, | \, y \sim \text{N}(0.3, 10) \; .
\end{equation}
Then it can be easily referenced throughout the paper by using
\eqref{eq:key}.

\end{document}
```

Save this file as `myfile.tex` then at the prompt type `latex myfile.tex` or in emacs use C-c C-c. This will produce a `.dvi` file that can be viewed with xdvi or in emacs use C-c C-c again to see it. The result follows.

Using LaTeX, I plan to rule the world. Care to join my cause? The key will be the equation

$$a^2 + b^2 = c^2 \; .$$

A labeled equation is different

$$\frac{\mu}{\sigma} \, | \, y \sim \text{N}(0.3, 10) \; . \tag{1.1}$$

Then it can be easily referenced throughout the paper by using (1.1).

1.6.2 Sweave and Vignettes

`Sweave` is a framework for putting LaTeX code and R code together in the same document. The idea is that we create a single source file that will produce a document that has R code and its documentation along with the output of the R code including tables and figures. A *vignette* is just an `Sweave` file that illustrates an R package. (Once again Charlie Geyer has a nice web page with extensive examples; see `http://www.stat.umn.edu/~charlie/Sweave/`)

Here is a simple example of using `Sweave` with the `vecSum` package. We begin by creating a `.Rnw` file which looks like a LaTeX document with R code "chunks."

```
\documentclass[12pt]{article}

\usepackage{amsbsy,amsmath,amsthm,amssymb}

\usepackage[sort,longnamesfirst]{natbib}
\newcommand{\pcite}[1]{\citeauthor{#1}'s \citeyearpar{#1}}
\usepackage{geometry}
\geometry{hmargin=3cm,vmargin={2.25cm,2.25cm},nohead,footskip=0.5in}

\renewcommand{\baselinestretch}{1.66}
\setlength{\baselineskip}{0.3in} \setlength{\parskip}{.05in}

\usepackage[dvips]{changebar}

\begin{document}

\title{My First Sweave Document}
\author{Dr. Evil}
\date{December 25, 2525}
\maketitle

Here we consider a simple Swaeve document.
<<summary>>=
library(MASS)
data(geyser)
attach(geyser)
summary(waiting)
```

```
@

Lets see how to make a plot. First make something to plot.

<<data>>=
x <- 1:10
y <- rnorm(10)
out<-lm(y ~ x)
@

Then Figure~\ref{figpe1} is produced by the following code and appears on
p.~\pageref{figpe1}.

<<label=fig1plot,include=FALSE>>==
plot(x, y)
abline(out)
@
\begin{figure}
\begin{center}
<<label=fig1,fig=TRUE,echo=FALSE>>=
<<fig1plot>>
@
\end{center}
\caption{A simple plot.}
\label{figpe1}
\end{figure}
\end{document}
```

Save this file as `mysweave.Rnw`. Now go to the prompt and type

```
echo 'Sweave("mysweave.Rnw")' | R --vanilla --quiet
```

This will produce the following LaTeX file which can be treated like any other LaTeX file.

```
\documentclass[12pt]{article}

\usepackage{amsbsy,amsmath,amsthm,amssymb}

\usepackage[sort,longnamesfirst]{natbib}
\newcommand{\pcite}[1]{\citeauthor{#1}'s \citeyearpar{#1}}
\usepackage{geometry}
\geometry{hmargin=3cm,vmargin={2.25cm,2.25cm},nohead,footskip=0.5in}

\renewcommand{\baselinestretch}{1.66}
\setlength{\baselineskip}{0.3in} \setlength{\parskip}{.05in}

\usepackage[dvips]{changebar}

\usepackage{/APPS/32/lib/R/share/texmf/Sweave}
\begin{document}

\title{My First Sweave Document}
\author{Dr. Evil}
\date{December 25, 2525}
\maketitle

\begin{Schunk}
\begin{Sinput}
> library(MASS)
> data(geyser)
> attach(geyser)
> summary(waiting)
```

```
\end{Sinput}
\begin{Soutput}
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  43.00   59.00   76.00   72.31   83.00  108.00
\end{Soutput}
\end{Schunk}

Lets see how to make a plot. First make something to plot.

\begin{Schunk}
\begin{Sinput}
> x <- 1:10
> y <- rnorm(10)
> out <- lm(y ~ x)
\end{Sinput}
\end{Schunk}

Then Figure~\ref{figpe1} is produced by the following code and appears on
p.~\pageref{figpe1}.

\begin{Schunk}
\begin{Sinput}
> plot(x, y)
> abline(out)
\end{Sinput}
\end{Schunk}
\begin{figure}
\begin{center}
\includegraphics{mysweave-fig1}
\end{center}
```

```
\caption{A simple plot.}
\label{figpe1}
\end{figure}

\end{document}
```

There is one thing that a vignette requires that is not included in a typical `Sweave` document:

```
% \VignetteIndexEntry{vecSum Example}
```

which should be included before the `\begin{document}` command.

1.7 Numerical Preliminaries

Most people are vaguely aware that the number system used in a computer is not the same as the one we like to think about, namely $\mathbb{R}$. Sometimes this can cause major problems but, to the uninitiated, it often appears to be only a semantic distinction.

It's clear that not all of $\mathbb{R}$ can be represented in a computer–$\mathbb{R}$ is uncountable after all. Lets look at just how large (and small) computer numbers can be in a simple R session.

```
> 2^1023
[1] 8.988466e+307
> 2^1024
[1] Inf
> 2^-1074
[1] 4.940656e-324
> 2^-1075
[1] 0
```

We will generally let $F \subset \mathbb{R}$ denote the possible computer numbers.

The second thing that is important to keep in mind is that computer algorithms are also only approximate. For example, when we want $x \in \mathbb{R}$ we actually get $x^* \in F$. Moreover, if we try to compute a function $f(\cdot)$ we actually get $f^*(\cdot)$. For a function $f(\cdot)$ and an input x we say that $f(x)$ is *well-conditioned* if $f(x)$ is close to $f(u)$ whenever u is close to x and *ill-conditioned* otherwise. If $f(x)$ is ill-conditioned then $f^*(x^*)$ will be problematic.

The difference between $f^*(x^*)$ and $f(x^*)$ typically depends on the algorithm used to compute f^*. An algorithm is *stable* if

$$f^*(x) = f(u)$$

whenever u is near x. That is, the algorithm produces the exact answer to a nearby problem. An algorithm can matter a lot. Consider calculating the sample variance

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2$$

```
> x<-1:3
> mean(x^2) - (mean(x))^2
[1] 0.6666667
> x<-1:3 + 1e+5
> mean(x^2) - (mean(x))^2
[1] 0.666666
> x<-1:3 + 1e+10
> mean(x^2) - (mean(x))^2
[1] -16384
```

Note that the `var` function divides by $n-1$ rather than n and hence

```
> x<-1:3
> var(x)
[1] 1
```

Chapter 2

Optimality Conditions

2.1 Introduction to Optimization

Optimization is a central concern for statisticians. For example, a frequentist may want to use maximum likelihood while a Bayesian may want to find a posterior mode. In a course on mathematical statistics these tasks are done analytically. In the real world this is often impossible and hence we must approximate the quantity of interest. This is usually done via an iterative routine implemented on a computer.

For $n \geq 1$ let $\mathbb{R}^n$ be the set of n-tuples $x = (x_1, \ldots, x_n)$ of real numbers. Suppose we are given an *objective function* f along with a set of constraints C. The problem we are interested in solving is

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to } x \in C\,. \tag{2.1}$$

Note that if we want to maximize f we minimize $-f$. If $C = \mathbb{R}^n$ this is an *unconstrained* optimization problem while if $C \subset \mathbb{R}^n$ it is a *constrained* optimization problem.

Example 2.1.1. *Suppose we have measurements $y_1, \ldots, y_n$ at times $x_1, \ldots, x_n$*

and we want to fit the following model

$$g(x \mid \beta) = \beta_1 + \beta_2 x^3 \exp\{-(\beta_3 - x)^2\}$$

where $\beta = (\beta_1, \beta_2, \beta_3)^T$. *Define* $r_i(\beta) = y_i - g(x_i \mid \beta)$. *One method for obtaining an estimate of* β *is to solve the problem*

$$\min_{\beta \in \mathbb{R}^3} \sum_{i=1}^{n} r_i^2(\beta)$$

which is an unconstrained optimization problem known as nonlinear least squares.

For unconstrained optimization problems we will consider ways to identify a local minimum. That is, we will focus on finding points at which the objective function is smaller than at all other feasible points in its neighborhood. Global solutions to (2.1) when $C = \mathbb{R}^n$ are often desirable, however they are also often extremely difficult to identify and locate and, in fact, may not exist; see Figure 2.1. On the other hand, when $C \subset \mathbb{R}^n$, a global solution may be possible; again see Figure 2.1. We can be more precise about what constitutes a solution to (2.1).

Definition 2.1.1. *The point* $x^* \in \mathbb{R}^n$ *is a*

1. *global minimizer if* $f(x^*) \leq f(x)$ *for all* $x \in \mathbb{R}^n$
2. *local minimizer if there is a neighborhood* $N(x^*)$ *such that* $f(x^*) \leq f(x)$ *for all* $x \in N(x^*)$
3. *strict local minimizer if there is a neighborhood* $N(x^*)$ *such that* $f(x^*) < f(x)$ *for all* $x \in N(x^*)$ *and* $x \neq x^*$.
4. *isolated local minimizer if there is a neighborhood* $N(x^*)$ *such that* x^* *is the only local minimizer in* $N(x^*)$.

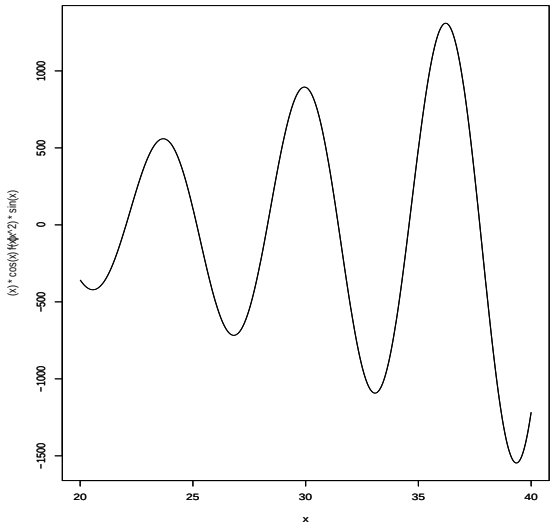

Figure 2.1: A function without a global minimum.

Example 2.1.2.

Suppose $f(x) = c$ *for* $c \in \mathbb{R}$ *then every point is a weak local minimizer.*

Suppose $f(x) = (x-1)^2$ *then* $x = 1$ *is a strict local minimizer.*

Strict local minimizers are not always isolated but isolated local minimizers are strict.

Definition 2.1.2. $S \subseteq \mathbb{R}^n$ *is a convex set if for* $x, y \in S$

$$\alpha x + (1-\alpha)y \in S \quad \forall\, \alpha \in [0,1]\,.$$

Definition 2.1.3. $f : \mathbb{R}^n \to \mathbb{R}$ *is convex if*

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \forall\, \alpha \in [0,1]\,.$$

The set of points lying above the graph of a convex function forms a convex set.

Convexity allows characterization of local and global minimizers.

Theorem 2.1.1. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is convex and that x^* is a local minimizer of f. Then x^* is a global minimizer of f.*

Proof. Suppose by way of contradiction that x^* is not a global minimizer of f. Then there exists $z \in \mathbb{R}^n$ such that $f(z) < f(x^*)$. Consider the line segment joining x^* and z, i.e.,

$$x = \alpha z + (1 - \alpha)x^* \quad \text{for some } \alpha \in (0, 1] .$$

By convexity

$$f(x) \leq \alpha f(z) + (1 - \alpha)f(x^*) < f(x^*) . \tag{2.2}$$

Since any $N(x^*)$ will contain at least one point at which (2.2) is satisfied this contradicts the assumption that x^* is a local minimizer. □

In the rest of this chapter we consider some generalizations of the familiar criteria that a necessary condition for a minimum of a function is that the first derivative is zero and a sufficient condition for a local minimum is that the first derivative is zero and the second derivative is positive. Since such conditions are rarely stated carefully, we begin with a review of differentiation theory that gives us the needed tools.

2.2 Differentiation

2.2.1 $\mathbb{R}^n$

Definition 2.2.1. *An inner product on $\mathbb{R}^n$ is a function $u : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ such that for all α, $\beta \in \mathbb{R}$ and all x, y, $z \in \mathbb{R}^n$ the following are satisfied*

1. *$u(x, x) \geq 0$ with equality if and only if $x = 0$,*

2. $u(x, y) = u(y, x)$,

3. $u(\alpha x + \beta y, z) = \alpha u(x, z) + \beta u(y, z)$.

We will denote an inner product by $\langle x, y\rangle = u(x, y)$. *The norm of* $x \in \mathbb{R}^n$ *is* $\|x\| = \sqrt{\langle x, x\rangle}$.

It is standard that $\mathbb{R}^n$ is an n-dimensional vector space, which we equip with the canonical inner product

$$\langle x, y\rangle = x^T y = x_1 y_1 + \cdots + x_n y_n, \qquad x, y \in \mathbb{R}^n$$

and the Euclidean norm

$$\|x\| = \sqrt{\langle x, x\rangle} = \sqrt{x^T x} = \sqrt{x_1^2 + \cdots + x_n^2}, \qquad x \in \mathbb{R}^n$$

and the associated metric

$$d(x, y) = \|x - y\|, \qquad x, y \in \mathbb{R}^n.$$

A *linear transformation* is a function $T : \mathbb{R}^n \to \mathbb{R}^m$ that satisfies the linearity property

$$T(ax + by) = aT(x) + bT(y), \qquad a, b \in \mathbb{R},\ x, y \in \mathbb{R}^n\ .$$

Every linear transformation can be represented by an $m \times n$ matrix, so $y = T(x)$ has two different interpretations. Thinking abstractly, y is the image of x under the mapping T. Thinking concretely, y is the result of the matrix multiplication Tx

$$y_i = \sum_{j=1}^{n} t_{ij} x_j$$

where the t_{ij} are the components of the matrix T. Following standard practice, we will write Tx for both interpretations, so the notation does not force either interpretation.

In the case $m = 1$, where the transformation maps into $\mathbb{R}$, a linear transformation is called a *linear functional*. Every linear functional is of the form $x \mapsto \langle x, y \rangle$ for some $y \in \mathbb{R}^n$.

In the case $m = n$, where the domain and codomain are the same, a linear transformation is called a *linear operator*.

A *bilinear form* on $\mathbb{R}^n$ is a function from $\mathbb{R}^n \times \mathbb{R}^n$ to $\mathbb{R}$ that is linear in both arguments. Every bilinear form is of the form $(x, y) \mapsto \langle x, Ay \rangle$ for some linear operator A. The same bilinear form can also be constructed using the adjoint operator A^* determined by

$$\langle x, Ay \rangle = \left\langle A^* x, y \right\rangle, \qquad x, y \in \mathbb{R}^n. \tag{2.3}$$

Comparing the sums for matrix multiplications and inner products written out explicitly, it is clear that the matrix representing A^* is the transpose of the matrix representing A.

A bilinear form b is *symmetric* if

$$b(x, y) = b(y, x), \qquad x, y \in \mathbb{R}^n.$$

From (2.3) it is clear that a bilinear form is symmetric if and only if the matrix A representing it satisfies $A = A^*$, in which case we say that A is symmetric. Note that the inner product is itself a symmetric bilinear form.

An operator B is *antisymmetric* if $B = -B^*$. Any operator A can be decomposed into symmetric and antisymmetric parts $A = A_s + A_a$, where

$$A_s = \tfrac{1}{2}(A + A^*) \tag{2.4a}$$
$$A_a = \tfrac{1}{2}(A - A^*) \tag{2.4b}$$

A *quadratic form* on $\mathbb{R}^n$ is a function from $\mathbb{R}^n \times \mathbb{R}^n$ having the form $x \mapsto \frac{1}{2} b(x, x)$ where b is a bilinear form. By symmetry of the inner product

$$\langle x, Ax \rangle = \langle Ax, x \rangle = \left\langle x, A^* x \right\rangle = \langle x, A_s x \rangle$$

where A_s is given by (2.4a). So to each bilinear form there corresponds a quadratic form, but the quadratic form only depends on the symmetric part of the operator inducing the bilinear form.

2.2.2 Little Oh Notation

A function $\psi : U \to \mathbb{R}^n$, where U is a neighborhood of zero in $\mathbb{R}^n$, satisfying

$$\lim_{x \to 0} \psi(x) = 0 \tag{2.5}$$

can also be described as being continuous at zero if we add the condition $\psi(0) = 0$. Such a function is said to be $o(1)$, read "little oh of one," written

$$\psi(x) = o(1) \ .$$

Saying that $\psi_1(x) = o(1)$ and $\psi_2(x) = o(1)$, does not mean, despite appearances, that $\psi_1 = \psi_2$ but only that both ψ_1 and ψ_2 are continuous at zero and $\psi_1(0) = \psi_2(0) = 0$. Little oh notation is a code that is not decoded according to the usual rules of mathematics.

Everything above also applies to a function $\psi : U \to \mathbb{R}^n$. It is said to be $o(1)$ if (2.5) holds or if it is continuous at zero with $\psi(0) = 0$.

More generally, given two functions f and g from a neighborhood U of zero in $\mathbb{R}^n$ to $\mathbb{R}^m$, we say that f is $o\big(g(x)\big)$, read "little oh of $g(x)$" if

$$f(x) = \|g(x)\|\psi(x) \tag{2.6}$$

for some $o(1)$ function ψ. The comment about little oh notation being an unusual code applies even more here. Argument is almost always clearer when (2.6) is used to put any little oh terms into ordinary mathematical notation.

2.2.3 Differentiation

Definitions

A function $f : U \to \mathbb{R}^m$, where U is a neighborhood of a point x in $\mathbb{R}^n$, is *differentiable* at x if there is a linear transformation $A : \mathbb{R}^n \to \mathbb{R}^m$ such that

$$f(x + y) = f(x) + Ay + o(\|y\|), \tag{2.7}$$

in which case we say that A is the *derivative of f at x* and write $f'(x)$ or $\nabla f(x)$ in place of A. As usual, we can think of the linear transformation A as an $m \times n$ matrix, if we like, and think of Ay as a matrix multiplication.

It is not obvious from the definition that at most one A satisfying (2.7) exists. It turns out that this is true, and we will prove it presently. That means the derivative, if it exists, is uniquely defined.

If we decode the little oh notation in (2.7) we get

$$f(x + y) = f(x) + Ay + \|y\|\psi(y),$$

for some $o(1)$ function ψ, hence

$$\frac{f(x + y) - f(x) - Ay}{\|y\|} = \psi(y),$$

or

$$\lim_{y \to 0} \frac{f(x + y) - f(x) - Ay}{\|y\|} = 0. \tag{2.8}$$

Often (2.8) given as the definition of differentiability: f is differentiable at x if and only if (2.8) holds for some linear transformation A, in which case $\nabla f(x) = A$.

The "vector space" notion of derivative introduced in this section, sometimes called the *Fréchet derivative* by those who like eponyms, is rather different from the ordinary derivative of a real-valued function of a single real variable. It is important to know that the differences arise because of

the *domain* of the function being multi-dimensional. They arise even for a real-valued function on $\mathbb{R}^n$. On the other hand, the *codomain* being multi-dimensional is rather trivial. A vector-valued function f can be thought of as a vector

$$f(x) = \big(f_1(x), \ldots, f_m(x)\big) \tag{2.9}$$

of real-valued functions. Since a sequence of vectors converges if and only if their components converge, it is obvious from the definitions that f is differentiable if and only if each f_i is differentiable and

$$\nabla f(x) = \big(\nabla f_1(x), \ldots, \nabla f_m(x)\big).$$

Directional and Partial Derivatives

If $f : U \to \mathbb{R}^m$ is a function, where U is a neighborhood of a point x in $\mathbb{R}^n$, then

$$f'(x; w) = \lim_{s \downarrow 0} \frac{f(x + sw) - f(x)}{s}$$

is called the *one-sided directional derivative* of f at x in the direction w if the limit exists. If f is actually differentiable at x, then

$$f(x + sw) - f(x) = sf'(x)w + s\|w\|\psi(sw)$$

for some $o(1)$ function ψ. Hence

$$f'(x; w) = \lim_{s \downarrow 0} \big[f'(x)w + \|w\|\psi(sw)\big] = f'(x)w$$

Hence all directional derivatives exist and

$$f'(x; w) = f'(x)w$$

Hence $w \mapsto f'(x; w)$ is a linear transformation, the same linear transformation as the derivative $f'(x)$.

This gives us the proof of the uniqueness of the derivative promised at the beginning of the preceding section. The directional derivatives $f(x; w)$

are uniquely defined, since limits are uniquely defined. Since the directional derivatives determine the derivative (if it exists), the derivative is also uniquely defined.

Thus if the derivative $f'(x)$ exists, then all of the directional derivatives $f'(x;w)$ exist and determine a linear transformation $w \mapsto f'(x;w)$. The converse is not true. All of the directional derivatives can exist and $w \mapsto f'(x;w)$ can be linear, but the function f not be differentiable.

One set of particularly interesting directional derivatives are those along directions parallel to coordinate axes. Let e_i denote the unit vector having all components zero except the i-th. If the coordinate functions of a vector-valued function f are given by (2.9), then $f_i'(x;e_j)$ is another notation for the partial derivative usually denoted $\partial f_i(x)/\partial x_j$. If we write down what it means to be the matrix representing $\nabla f(x)$, we see that it is the matrix of partial derivatives.

Thus, if a function is differentiable, the derivative is the matrix of partial derivatives. But, as the example above shows, the partial derivatives can all exist and the function not be differentiable.

Stronger assumptions must be imposed to infer differentiability from the properties of the partial derivatives. The family of functions $f : U \to \mathbb{R}^m$, where U is an open set in $\mathbb{R}^n$, that are continuously differentiable everywhere, meaning the map $x \mapsto \nabla f(x)$ is continuous, is denoted $C^1(U)$. It is a theorem of real analysis that if the partial derivatives exist and are continuous everywhere on some open set U, then $f \in C^1(U)$.

Second Derivatives

The space of all $m \times n$ matrices is an (mn)-dimensional vector space, which we can consider to be $\mathbb{R}^{mn}$. Hence if $\nabla f(y)$ exists for all y in a neighborhood U of x we can consider whether the map $\nabla f : U \to \mathbb{R}^{mn}$ is differentiable. Applying the definition, we see that ∇f is differentiable at x if there is a

linear function $A : \mathbb{R}^n \to \mathbb{R}^{mn}$ such that

$$\nabla f(x + y) = \nabla f(x) + Ay + o(\|y\|), \tag{2.10}$$

in which case we say that A is the *second derivative of f at x* and write it as $f''(x)$ or $\nabla^2 f(x)$. We also say f is *twice differentiable* at x.

Applying what we have already said about differentiation, we see that if f is twice differentiable, then the second partial derivatives must exist, and $\nabla^2 f(x)$ must be an (mn^2)-dimensional object with elements $\partial^2 f_i(x)/\partial x_j \partial x_k$.

What sort of object is a bit difficult to say. If we want to consider $\nabla f(x)$ a matrix, then $\nabla^2 f(x)$ must be a linear transformation that maps a vector y to a matrix $\nabla^2 f(x)y$. Abstractly, everything is simple, $m \times n$ matrices form a vector space, and $\nabla^2 f(x)$ maps $\mathbb{R}^n$ to that space. Considered concretely, things are a bit more complicated, the elements $\partial^2 f_i(x)/\partial x_j \partial x_k$ of the matrix representing $\nabla^2 f(x)$ have three indices, thus are most naturally considered to form a three-dimensional array, not a matrix.

Fortunately, we are most interested in first and second derivatives of scalar-valued ($\mathbb{R}$-valued) functions. Then the first derivative can be considered an n vector rather than a $1 \times n$ matrix, and we rewrite (2.7) as

$$f(x + y) = f(x) + \langle \nabla f(x), y \rangle + o(\|y\|).$$

Similarly, the second derivative can be considered an $n \times n$ matrix rather than a $1 \times n \times n$ array, and we rewrite (2.10) as

$$\nabla f(x + y) = \nabla f(x) + \nabla^2 f(x)y + o(\|y\|).$$

Now $\nabla^2 f(x)$ makes sense as a linear operator on $\mathbb{R}^n$ that maps the vector y to another vector $\nabla^2 f(x)y$, which can be added to other vectors like $\nabla f(x)$.

2.2.4 Taylor's Theorem

In Section 2.5 a more general version of the following theorem is addressed in some detail. However, the following should be sufficient for our present

purpose.

Theorem 2.2.1. *(Taylor's Theorem) Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $s \in \mathbb{R}^n$. Then f has a linear approximation for some $t \in (0, 1)$*

$$f(x + s) = f(x) + \langle s, \nabla f(x + ts) \rangle \ . \tag{2.11}$$

If f is twice differentiable then

$$\nabla f(x + s) = \nabla f(x) + \int_0^1 \nabla^2 f(x + ts)\, s\, dt \ . \tag{2.12}$$

and

$$f(x + s) = f(x) + \langle s, \nabla f(x + ts) \rangle + \frac{1}{2} \left\langle s, \nabla^2 f(x + ts) s \right\rangle \ . \tag{2.13}$$

for some $t \in (0, 1)$.

2.3 Unconstrained Optimization

Recall that the unconstrained optimization problem can be stated as

$$\min_{x \in \mathbb{R}^n} \ f(x) \ . \tag{2.14}$$

However, we will focus on the characterization of local minima. Recall that a linear operator $A : \mathbb{R}^n \to \mathbb{R}^n$ is *positive semi-definite* if

$$\langle y, Ay \rangle \geq 0, \qquad y \in \mathbb{R}^n$$

and *positive definite* if

$$\langle y, Ay \rangle > 0, \qquad y \in \mathbb{R}^n, \ y \neq 0.$$

We use the abbreviations $A \geq 0$ to indicate positive semi-definiteness and $A > 0$ to indicate positive definiteness.

Theorem 2.3.1. *(First-Order Necessary Conditions) Suppose $f : U \to \mathbb{R}$, where U is a neighborhood of a point x^* in $\mathbb{R}^n$, is continuously differentiable on U. If x^* is a local minimizer of f then*

$$\nabla f(x^*) = 0. \tag{2.15}$$

Proof. Suppose by way of contradiction that $\nabla f(x^*) \neq 0$. Set $s = -\nabla f(x^*)$ so that $s^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$. By continuity there exists M such that for all $0 \leq t \leq M$

$$s^T \nabla f(x^* + ts) < 0 \, .$$

Taylor's theorem says for any $0 < t^* \leq M$

$$f(x^* + t^* s) = f(x^*) + t^* \left\langle s, \nabla f(x^* + ts) \right\rangle$$

for some $0 < t < t^*$. Hence $f(x^* + t^* s) < f(x^*)$ for any $0 < t^* \leq M$. Since any neighborhood $N(x^*)$ will contain at least one point at which $f(x^* + t^* s) < f(x^*)$ for some t^* this contradicts the assumption that x^* is a local minimizer. □

Theorem 2.3.2. *(Second-Order Necessary Conditions) Suppose $f : U \to \mathbb{R}$, where U is a neighborhood of a point x^* in $\mathbb{R}^n$, and $\nabla^2 f$ is continuous on U. If x^* is a local minimizer of f then* (2.15) *holds and*

$$\nabla^2 f(x^*) \geq 0 \, . \tag{2.16}$$

Proof. We already know that (2.15) holds from the previous theorem. Suppose by way of contradiction that $\nabla^2 f(x^*)$ is not positive semi-definite. Then there is an $s \in \mathbb{R}^n$ such that $s^T \nabla^2 f(x^*) s < 0$. By continuity of $\nabla^2 f$ at x^* there exists $M > 0$ such that $s^T \nabla^2 f(x^* + ts) s < 0$ for all $0 \leq t \leq M$.Taylor's theorem says for any $0 < t^* \leq M$

$$f(x^* + t^* s) = f(x^*) + t^* \left\langle s, \nabla f(x^*) \right\rangle + \frac{1}{2}(t^*)^2 \left\langle s, \nabla^2 f(x^* + ts) s \right\rangle < f(x^*)$$

for some $0 < t < t^*$. Since any neighborhood $N(x^*)$ will contain at least one point at which $f(x^*+t^*s) < f(x^*)$ for some t^* this contradicts the assumption that x^* is a local minimizer. □

Theorem 2.3.3. *(Second-Order Sufficient Conditions) Suppose $f : U \to \mathbb{R}$, where U is a neighborhood of a point x^* in $\mathbb{R}^n$, and $\nabla^2 f$ is continuous on U. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) > 0$ then f has a strict local minimum at x.*

Proof. Since $\nabla^2 f(x^*) > 0$ and $\nabla^2 f$ is continuous there exists an $r > 0$ such that $\nabla^2 f(z) > 0$ for all $z \in B_r(x^*) = \{z \ : \ \|z - x^*\| < r\}$. If $s \neq 0$ and $\|s\| < r$ then $x^* + s \in B_r(x^*)$ and hence by Taylor's theorem

$$\begin{aligned} f(x^* + s) &= f(x^*) + \langle s, \nabla f(x^*) \rangle + \frac{1}{2} \left\langle s, \nabla^2 f(x^* + ts)s \right\rangle \\ &= f(x^*) + \frac{1}{2} \left\langle s, \nabla^2 f(x^* + ts)s \right\rangle \end{aligned}$$

for some $0 < t < 1$. Since $x^* + ts \in B_r(x^*)$ we have $s^T \nabla^2 f(x^* + ts)s > 0$ which implies $f(x^* + s) > f(x^*)$. The result follows. □

Theorem 2.3.4. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable. If x^* satisfies $\nabla f(x^*) = 0$ then x^* is a global minimizer of f.*

Proof. See Nocedal and Wright (1999, page 17). □

2.4 Constrained Optimization

Now consider the problem of minimizing a function f defined on some subset of $\mathbb{R}^n$ subject to the constraint that the solution lie in a closed set C. For simplicity we assume that f is defined on an open set containing C and is differentiable everywhere in its domain. The shorthand for our problem is

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to } x \in C \, .$$

Example 2.4.1. *Consider a simple example where the objective function has the form* $f(x) = x_1 + x_2 + x_3$ *and we want to restrict our attention to the region where* $4 - x_1^2 - x_2^2 - x_3^2 \geq 0$. *Then the feasible region consists of the ball centered at the origin with radius 2 and its interior.*

In the material on unconstrained optimization we saw that finding a global minimum is difficult in general; see Figure 2.1. However, there are times when adding constraints to the problem makes this possible; consider Figure 2.1 with the constraint that $30 \leq x \leq 35$.

An editorial note: much of the following material is based on the presentations given by Nocedal and Wright (1999) and Rockafellar and Wets (1998).

2.4.1 The Tangent Cone

A set $K \subset \mathbb{R}^n$ is a *cone* if it contains the origin and $x \in K$ implies $\lambda x \in K$ for all $\lambda \geq 0$

Example 2.4.2. *Define*

$$F = \{(x_1, x_2) \,:\, x_1 > 0,\, x_2 \geq 0\} \,.$$

Then F *is a cone in* $\mathbb{R}^2$.

The *tangent cone* of a set $C \subset \mathbb{R}^n$ at a point $x \in C$, denoted $T_C(x)$, is the set of all vectors v such that there exists a sequence $\tau_n \downarrow 0$ and a sequence x_n in C converging to x such that

$$\frac{x_n - x}{\tau_n} \to v$$

Example 2.4.3. *Suppose*

$$C = \{x \in \mathbb{R}^2 \,:\, x_1 \geq x_2^2,\, x_2 \geq x_1^2\} \,.$$

Then the tangent cone at the origin is

$$T_C((0,0)) = \{x \in \mathbb{R}^2 \,:\, x_1 \geq 0,\, x_2 \geq 0\}\,.$$

Theorem 2.4.1. *$T_C(x)$ is a closed cone.*

Proof. Exercise. □

Another way to think about $T_C(x)$ is that it consists of the set of *directions* from which sequences x_n in C approach x. If none of the x_n are equal to x, let

$$u_n = \frac{x_n - x}{\|x_n - x\|}.$$

The u_n are unit vectors, so by compactness of the unit sphere there are convergent subsequences $u_{n_k} \to u$. The tangent cone consists of all such u and the rays $\{\, \lambda u : \lambda \geq 0 \,\}$ generated by such u.

2.4.2 The Variational Inequality

Theorem 2.4.2. *If $f : U \to \mathbb{R}$, where U is a neighborhood of a point x in $\mathbb{R}^n$, is differentiable at x, then a necessary condition that f have a local minimum over a closed set C containing x is*

$$\langle \nabla f(x), v \rangle \geq 0, \qquad v \in T_C(x). \tag{2.17}$$

Proof. If $v \in T_C(x)$, there are $\tau_n \downarrow 0$ and x_n in C converging to x such that $(x_n - x)/\tau_n \to v$. Note that

$$f(x_n) = f(x_n + x - x) = f(x + (x_n - x))$$

so that the definition of differentiability says (with $y = x_n - x$))

$$f(x_n) - f(x) = \langle \nabla f(x), x_n - x \rangle + o\left(\|x_n - x\|\right).$$

By continuity of the norm

$$\frac{\|x_n - x\|}{\tau_n} \to \|v\|.$$

Hence for any real number $c > \|v\|$ there is an integer N such that

$$\|x_n - x\| \leq c\tau_n, \qquad n \geq N,$$

and

$$\frac{f(x_n) - f(x)}{\tau_n} = \left\langle \nabla f(x), \frac{x_n - x}{\tau_n} \right\rangle + \frac{o(\|x_n - x\|)}{\tau_n} \to \langle \nabla f(x), v \rangle$$

If f has a local minimum at x, then the left hand side is eventually greater than or equal to zero, hence so is the limit on the right. □

The *variational inequality* (2.17) is the generalization of (2.15) to inequality-constrained problems.

2.4.3 Polars

The *polar* of a cone K is the cone

$$K^* = \{\, v : \langle v, x \rangle \leq 0,\ x \in K \,\}.$$

K^* is always a closed convex cone, regardless of whether K is closed or convex (because it is the intersection of closed half-spaces).

Theorem 2.4.3. (The Double Polar Theorem) *If K is a closed convex cone then $K^{**} = K$. In general, K^{**} is the closed convex hull of K.*

2.4.4 Normal Cones

The *regular normal cone* of a set $C \subset \mathbb{R}^n$ at a point $x \in C$, denoted $\widehat{N}_C(x)$, is the set of all vectors v such that

$$\langle v, y - x \rangle \leq o(\|y - x\|), \qquad y \in C. \tag{2.18}$$

The elements of $\widehat{N}_C(x)$ are called *regular normals* to C at x.

The *normal cone* of a set $C \subset \mathbb{R}^n$ at a point $x \in C$, denoted $N_C(x)$, is the set of all vectors v such that there exists a sequence x_n in C converging to x and a sequence $v_n \to v$ with $v_n \in \widehat{N}_C(x_n)$. The elements of $N_C(x)$ are called *normals* to C at x.

For now, we are mostly interested in the regular normal cone. The normal cone will become interesting when we consider asymptotics. The regular normal cone is interesting because of its close connection with the tangent cone.

Theorem 2.4.4. $\widehat{N}_C(x) = T_C(x)^*$.

This gives an equivalent way of stating the variational inequality (2.17).

Corollary 2.4.5. *If f is defined in a neighborhood of x and differentiable at x, then a necessary condition that f have a local minimum over C at $x \in C$ is*

$$-\nabla f(x) \in \widehat{N}_C(x). \tag{2.19}$$

2.4.5 Lagrange Multipliers

We now consider the problem of minimizing a function f defined on a subset of a Euclidean space subject to a finite set of equality and inequality constraints

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g_i(x) = 0, \qquad i \in E \\ &\qquad\qquad\quad g_i(x) \le 0, \qquad i \in I \end{aligned} \tag{2.20}$$

where I and E are disjoint index sets, and the g_i are differentiable functions. This is a special case of the preceding set-up with

$$C = \{\, x : g_i(x) = 0,\ i \in E \text{ and } g_i(x) \le 0,\ i \in I \,\}. \tag{2.21}$$

C is a closed set because the g_i are continuous. A point x is said to be *feasible* if it satisfies the constraints so $x \in C$.

The method of *Lagrange multipliers* is a trick for converting constrained problems to simpler problems with additional variables, the Lagrange multipliers. Form the *Lagrangian function*

$$L(x) = f(x) + \sum_{i \in E \cup I} \lambda_i g_i(x), \tag{2.22}$$

the λ_i being the Lagrange multipliers.

Theorem 2.4.6. *A sufficient condition that x solve the problem (2.20) is that there exist Lagrange multipliers λ such that*

(a) *[minimization] x minimizes the Lagrangian (2.22).*

(b) *[primal feasibility] $g_i(x) = 0$, $i \in E$ and $g_i(x) \leq 0$, $i \in I$.*

(c) *[dual feasibility] $\lambda_i \geq 0$, $i \in I$.*

(d) *[complementary slackness] $\lambda_i g_i(x) = 0$, $i \in I$.*

Proof. Let y be any feasible point. By (b) and (d) $L(x) = f(x)$. By (a) $L(x) \leq L(y)$. By (b) and (c)

$$\sum_{i \in I} \lambda_i g_i(y) \leq 0 \quad \text{and} \quad \sum_{i \in E} \lambda_i g_i(y) = 0,$$

so $L(y) \leq f(y)$. Thus

$$f(x) = L(x) \leq L(y) \leq f(y).$$

□

Corollary 2.4.7. *If (a) in the theorem is changed to assert only that x is a local minimizer of the Lagrangian, then the conditions are sufficient for x to be a local minimizer of f over C.*

Proof. Consider the problem restricted to a neighborhood of x over which the Lagrangian achieves its minimum at x. □

A necessary condition for condition (a) of the theorem to hold is that the derivative be zero. Replacing (a) of the theorem by

(a) [zero gradient] $\nabla L(x) = \nabla f(x) + \sum_{i \in E \cup I} \lambda_i \nabla g_i(x) = 0$.

gives the so-called Kuhn-Tucker conditions Kuhn and Tucker (1951). Without further assumptions, this set of conditions is now neither necessary nor sufficient, not sufficient because $\nabla L(x) = 0$ does not guarantee even a local minimum and not necessary because no Lagrange multiplier vector λ need exist that makes the conditions hold.

2.4.6 Examples

Normal Means, Diagonal Covariance

Let x be a normal random vector with unknown mean vector μ and known covariance matrix Σ and precision matrix Σ^{-1}, which in this section are assumed to be diagonal and positive definite. We wish to estimate μ under the constraint $\mu_i \geq 0$, for all i. The estimation procedure is maximum likelihood, or equivalently, weighted least squares. The estimate is found by minimizing $\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$ with respect to μ subject to the constraints. The Lagrangian is

$$L(\mu) = \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) - \lambda^T\mu,$$

λ being the vector of Lagrange multipliers. The reason for the minus sign is that to plug into (2.22) we need to put the the constraints in the form $g_i(\mu) = -\mu_i \leq 0$. The Kuhn-Tucker conditions are thus

(a) $\nabla L(\mu) = -\Sigma^{-1}(x-\mu) - \lambda = 0$.

(b) $\mu_i \geq 0$, for all i.

(c) $\lambda_i \geq 0$, for all i.

(d) $\lambda_i \mu_i = 0$, for all i.

Complementary slackness requires $\lambda_i = 0$ or $\mu_i = 0$. If $\lambda_i = 0$ then $\mu_i = x_i$, and primal feasibility requires $x_i \geq 0$. If $\mu_i = 0$, then $0 \leq \lambda_i = -\sigma_{ii} x_i$, so $x_i \leq 0$. Thus we have our solution

$$\mu_i = \begin{cases} x_i, & x_i \geq 0 \\ 0, & x_i \leq 0 \end{cases}$$

Since the Lagrangian is a positive definite quadratic form its unique minimum is found where (a) holds, so this solution is a global minimum by Theorem 2.4.6.

Linear Regression

Let $y = X\beta + e$, where e is a vector of i. i. d. standard normal errors. We wish to estimate β by maximum likelihood (least squares) subject to the constraint $\beta \geq 0$. The Lagrangian is

$$L(\beta) = \frac{1}{2}(y - X\beta)'(y - X\beta) - \lambda'\beta,$$

and the Kuhn-Tucker conditions are

(a) $\nabla L(\beta) = -X'(y - X\beta) - \lambda = 0$.

(b) $\beta \geq 0$.

(c) $\lambda \geq 0$.

(d) $\lambda_i \beta_i = 0$, for all i.

Define $s = X'(y - X\beta)$ (this is the score, the gradient of the log likelihood). Then (a) becomes $s = -\lambda \le 0$. This is as far as we can go in reducing the Kuhn-Tucker conditions. There is no closed form solution. We are to find a β such that

$$s(\beta) = X'(y - X\beta) \le 0$$

satisfying complementary slackness: $s_i = 0$ or $\beta_i = 0$. Again because the Lagrangian is a positive definite quadratic function, any such solution is a global minimum.

Isotonic Regression

Suppose that we observe random variables

$$y_i = \mu_i + e_i$$

where μ_i are unknown parameters and the e_i are independent and and identically distributed mean zero normal "errors." The isotonic regression assumption is that the μ_i are ordered

$$\mu_1 \le \mu_2 \le \cdots \le \mu_n \tag{2.23}$$

For this example, we also assume that the error variance σ^2 is known, although the isotonic regression estimator is unchanged if σ^2 is an unknown parameter. As always with normal errors, maximum likelihood is least squares. The regression problem is to minimize

$$f(\mu) = \sum_{i=1}^{n} \tfrac{1}{2}(y_i - \mu_i)^2$$

subject to the $n - 1$ inequality constraints (2.23). Thus the Lagrangian is

$$L(\mu) = \sum_{i=1}^{n} \tfrac{1}{2}(y_i - \mu_i)^2 + \sum_{i=1}^{n-1} \lambda_i(\mu_i - \mu_{i+1})$$

Since this is a quadratic function of μ a zero of the Lagrangian is the global minimizer, and the Kuhn-Tucker conditions are sufficient conditions for a global minimizer in the isotonic regression problem.

The first K-T condition is

$$\frac{\partial L(\mu)}{\partial \mu_i} = \mu_i - y_i + \lambda_i - \lambda_{i-1} = 0, \qquad i = 1, \ldots, n \tag{2.24}$$

if we introduce $\lambda_0 = \lambda_n = 0$ to allow for the fact that the $i = 1$ and $i = n$ cases involve only one real Lagrange multiplier. We will refer to (2.24) as the "minimization" condition rather than the "zero gradient" condition because it does characterize the global minimum of the Lagrangian function.

Complementary slackness requires that either $\mu_i = \mu_{i+1}$ or $\lambda_i = 0$ for $i = 1, \ldots, n-1$. Consider a block of equal μ values of length l starting at $r+1$, that is,

$$\mu_r < \mu_{r+1} = \mu_{r+2} = \cdots = \mu_{r+l} < \mu_{r+l+1} \tag{2.25}$$

Complementary slackness implies $\lambda_r = \lambda_{r+l} = 0$. Hence

$$\sum_{i=r+1}^{r+l} \frac{\partial L(\mu)}{\partial \mu_i} = \sum_{i=r+1}^{r+l} (\mu_i - y_i)$$

the other λ's canceling because of the telescoping sum. Hence

$$\mu_{r+1} = \mu_{r+2} = \cdots = \mu_{r+l} = \frac{1}{l} \sum_{i=r+1}^{r+l} y_i \tag{2.26}$$

Thus the isotonic regression estimator is a step function with each step height being the average of the y values over the step. We will refer to (2.26) as the "minimization plus complementary slackness" condition because it encompasses those two of the four K-T conditions. The isotonic regression estimator must satisfy (2.26) for each step characterized by (2.25).

The hard part is now figuring out where to place the steps. To do that we need to use the other two K-T conditions. Primal feasibility requires that

the step heights increase. In order to apply dual feasibility we need to isolate the remaining Lagrange multipliers. Returning to the step characterized by (2.25), we now sum fewer terms of (2.24). For $k < l$

$$\sum_{i=r+1}^{r+k} \frac{\partial L(\mu)}{\partial \mu_i} = \sum_{i=r+1}^{r+k} (\mu_i - y_i) + \lambda_{r+k} = 0.$$

Hence dual feasibility requires

$$\sum_{i=r+1}^{r+k} (\mu_i - y_i) = -\lambda_{r+k} \leq 0, \qquad k = 1, \ldots, l-1,$$

or

$$\mu_{r+1} = \mu_{r+2} = \cdots = \mu_{r+l} \leq \frac{1}{k} \sum_{i=r+1}^{r+k} y_i, \qquad k = 1, \ldots, l-1, \tag{2.27}$$

We will refer to (2.27) as the "dual feasibility" condition for isotonic regression.

This gives us a complete characterization of the isotonic regression estimator: Step heights are determined by the minimization plus complementary slackness condition (2.26), if primal feasibility holds (step heights are increasing) and dual feasibility (2.27) holds for each step, then this is the unique solution to the isotonic regression problem.

The Pool Adjacent Violators Algorithm

How does one find the isotonic regression estimator? That is, how does one find steps such that the Kuhn-Tucker conditions are satisfied? An effective algorithm for solving the isotonic regression is the *pool adjacent violators algorithm* (PAVA). It works as follows.

1. [*initialization*] Start with any estimate μ that satisfies all the Kuhn-Tucker conditions *except* primal feasibility, for example, $\mu = y$.

2. [*pool adjacent violators*] If primal feasibility is not satisfied there are two adjacent steps that are not in increasing order: for some r, l, and m

$$\mu_r \neq \mu_{r+1} = \cdots = \mu_{r+l} > \mu_{r+l+1} = \cdots = \mu_{r+l+m} \neq \mu_{r+l+m+1}. \quad (2.28)$$

Combine the two steps into one satisfying (2.26), that is, redefine μ_{r+1}, $\ldots$, μ_{r+l+m} to be

$$\frac{1}{l+m} \sum_{i=r+1}^{r+l+m} y_i. \quad (2.29)$$

Repeat this step until primal feasibility is satisfied.

We claim that the PAV update (step 2 of the algorithm) preserves all of the K-T conditions except primal feasibility. Accepting that claim for the moment, we see that the algorithm must find the isotonic regression solution, because it terminates only if primal feasibility is also satisfied. It must terminate because the number of steps decreases in each execution of step 2, and if it does not terminate before collapsing down to a single step (all μ's equal) it must terminate then because that trivially satisfies primal feasibility.

So now we need to show that the PAV update preserves the other three K-T conditions, that is, it satisfies (2.26) and (2.27) for each (new) step after the pooling operation. Clearly, both these equations hold for the steps that are unchanged, and (2.26) is satisfied for the new step by definition of the pooling operation, so we only need to show dual feasibility for the new step. For this discussion, let μ_i denote the estimate at the beginning of the PAV update, and denote the height (2.29) of the new step by $\bar{\mu}$. Then, since the minimization plus complementary slackness condition is assumed to hold for the old estimate,

$$\bar{\mu} = \frac{l}{l+m}\mu_{r+l} + \frac{m}{l+m}\mu_{r+l+1}. \quad (2.30)$$

Thus

$$\mu_r \neq \mu_{r+1} = \cdots = \mu_{r+l} > \bar{\mu} > \mu_{r+l+1} = \cdots = \mu_{r+l+m} \neq \mu_{r+l+m+1}.$$

What we must show is dual feasibility for the new step, which is

$$\bar{\mu} \leq \frac{1}{k} \sum_{i=r+1}^{r+k} y_i, \qquad k = 1, \ldots l + m - 1. \tag{2.31}$$

Clearly (2.31) holds for $k < l$ because (2.27) implies

$$\bar{\mu} < \mu_{r+l} \leq \frac{1}{k} \sum_{i=r+1}^{r+k} y_i,$$

and by assumption (2.27) holds at the beginning of each PAV update. For $k \geq l$, we have

$$\begin{aligned} \frac{1}{k} \sum_{i=r+1}^{r+k} y_i &= \frac{l}{k} \mu_{r+l} + \frac{1}{k} \sum_{i=r+l+1}^{r+k} y_i \\ &\geq \frac{l}{k} \mu_{r+l} + \frac{k-l}{k} \mu_{r+l+1}, \end{aligned} \tag{2.32}$$

the equality being minimization plus complementary slackness for the old step from $r+1$ to $r+l$) and the inequality being dual feasibility for the old step from $r+l+1$ to $r+l+m$. If we think of the right hand side of (2.32) as a function of a continuous variable k, that is,

$$g(x) = \frac{l}{x} \mu_{r+l} + \frac{x-l}{x} \mu_{r+l+1}$$

then its derivative is

$$g'(x) = -\frac{l}{x^2} (\mu_{r+l} - \mu_{r+l+1}),$$

which is strictly negative by (2.28). Thus g is a decreasing function, and $g(l+m) = \bar{\mu}$ by (2.30), so the right hand side of (2.32), which is $g(k)$ is greater than $\bar{\mu}$ and that is dual feasibility for the new step.

2.4.7 Constraint Qualification

Let $A(x) \subset I$ denote the *active set* of constraints, those satisfied with equality at x

$$A(x) := \{\, i \in I : g_i(x) = 0 \,\} \ .$$

Define the set

$$K_C(x) := \{\, v : \langle \nabla g_i(x), v \rangle = 0,\ i \in E \text{ and } \langle \nabla g_i(x), v \rangle \leq 0,\ i \in A(x) \,\} \ . \tag{2.33}$$

Lemma 2.4.8. *$T_C(x) \subset K_C(x)$.*

Proof. Suppose $v \in T_C(x)$ so that there are $x_n \to x$ in C and $\tau_n \downarrow 0$ such that $(x_n - x)/\tau_n \to v$. Then by continuity of g_i for any $i \in A(x) \cup E$

$$\frac{g_i(x_n) - g_i(x)}{\tau_n} \to \langle \nabla g_i(x), v \rangle \, .$$

Since the left hand side is less than or equal zero for $i \in A(x)$ and equal to zero for $i \in E$, it follows that $v \in K_C(x)$. □

Theorem 2.4.9. *If all the constraints are linear $T_C(x) = K_C(x)$.*

Proof. Suppose $v \in K_C(x)$ and the constraints are linear, that is,

$$g_i(x) = \langle a_i, x \rangle + b_i$$

for some vector of scalars a_i and a scalar b_i for each i. We claim that there exists an $\epsilon > 0$ such that $x_\tau = x + \tau v \in C$, whenever $0 \leq \tau < \epsilon$. This proves $v \in T_C(x)$, because $(x_\tau - x)/\tau = v$ trivially converges to v as $\tau \downarrow 0$. Thus it only remains to prove the claim. There are three kinds of constraints to consider.

(*Equality Constraints*). Since g_i is an equality constraint

$$g_i(x_\tau) = \langle a_i, x \rangle + b_i + \tau \langle a_i, v \rangle = g_i(x) + \tau \langle a_i, v \rangle = \tau \langle a_i, v \rangle \ . \tag{2.34}$$

If $g_i(x) = \langle a_i, x\rangle + b_i$ then $\nabla g_i(x) = \nabla \langle a_i, x\rangle = a_i$. Since $v \in K_C(x)$ it follows that

$$\langle a_i, v\rangle = \langle \nabla g_i(x), v\rangle = 0$$

and hence

$$g_i(x_\tau) = g_i(x + \tau v) = 0, \qquad \tau \in \mathbb{R}$$

so that $x_\tau \in C$.

(*Active Inequality Constraints*). If g_i is an active inequality constraint, then (2.34) still holds. The first term on the right is again zero because g_i is active, and the second term is nonpositive if $\tau \geq 0$ by definition of $K_C(x)$. Hence for active inequality constraints

$$g_i(x_\tau) = g_i(x + \tau v) \leq 0, \qquad 0 \leq \tau.$$

(*Inactive Inequality Constraints*). If g_i is an inactive inequality constraint, then $g_i(x) < 0$, hence by continuity, there exists an $\epsilon_i > 0$ such that

$$g_i(x_\tau) = g_i(x + \tau v) < 0, \qquad 0 \leq \tau < \epsilon_i$$

Thus we see that if we take $\epsilon = \min_i \epsilon_i$, that x_τ satisfies all constraints when $0 \leq \tau < \epsilon$. That establishes the claim and the theorem. □

Theorem 2.4.10. *(Kuhn-Tucker constraint qualification) If $K_C(x) = T_C(x)$ and f is differentiable at x, then the Kuhn-Tucker conditions are necessary conditions for f to have a local minimum at x.*

Proof. If f has a local minimum at x then (2.19) holds. Consider the cone

$$W_C(x) = \{\, \lambda \nabla g_i(x) : i \in A(x) \text{ and } \lambda \geq 0 \text{ or } i \in E \text{ and } \lambda \in \mathbb{R} \,\}.$$

By assumption $K_C(x) = T_C(x)$, so $\widehat{N}_C(x) = K_C(x)^*$ by (2.33) $K_C(x) = W_C(x)^*$. Hence $\widehat{N}_C(x) = W_C(x)^{**}$. Thus by the double polar theorem $\widehat{N}_C(x)$

is the closed convex hull of $W_C(x)$, so there are λ_i, $i \in A(x) \cup E$, such that $\lambda_i \geq 0$, $i \in A(x)$ and

$$-\nabla f(x) = \sum_{i \in A(x) \cup E} \lambda_i \nabla g_i(x)$$

And this implies the Kuhn-Tucker conditions. □

We would now like to find some condition weaker than that of Theorem 2.4.9 that implies the Kuhn-Tucker constraint qualification $T_C(x) = K_C(x)$. The following theorem is taken from Rockafellar and Wets (1998).

Theorem 2.4.11. *The following two sets of conditions are equivalent. (First condition) The set*

$$\Lambda(x) = \left\{ \lambda \in \mathbb{R}^{I \cup E} : \lambda_i \geq 0,\ i \in I \text{ and } \lambda_i = 0,\ i \in I \setminus A(x) \right\}$$

contains no nonzero λ such that $\sum_i \lambda_i \nabla g_i(x) = 0$. (Second set of conditions)

(a) *The gradients $\nabla g_i(x)$, $i \in E$ are a linearly independent set of vectors.*

(b) *The set*

$$K'_C(x) = \{ v : \langle \nabla g_i(x), v \rangle = 0,\ i \in E \text{ and } \langle \nabla g_i(x), v \rangle < 0,\ i \in A(x) \}$$

is nonempty.

If either set of conditions hold, then $T_C(x) = K_C(x)$.

Proof. Since neither condition involves the inactive constraints, we may assume without loss of generality that $A(x) = I$ (there are no inactive constraints).

The first condition also implies the linear independence of the gradients of the equality constraints. Suppose without loss of generality that $E = \{1, \ldots, m\}$. Define a set of vectors b_i, $i = m+1, \ldots, n$, such that the set

$\nabla g_1(x)$, ..., $\nabla g_m(x)$, b_{m+1}, ..., b_n forms a basis for $\mathbb{R}^n$. Define $F : \mathbb{R}^n \to \mathbb{R}^n$ by

$$F_i(y) = \begin{cases} g_i(y), & i \le m \\ \langle b_i, y - x \rangle, & i > m \end{cases}$$

The Jacobian of F_i is the $n \times n$ matrix with rows $\nabla g_1(x)$, ..., $\nabla g_m(x)$, b_{m+1}, ..., b_n, which is nonsingular at x by construction, and hence, by continuity, nonsingular in a neighborhood W of x in $\mathbb{R}^n$. Thus by the inverse function theorem there is a neighborhood U of $F(x) = 0$, such that F^{-1} exists and is differentiable on U. Writing $J = \nabla F(x)$ for the Jacobian at x, the inverse Jacobian is $\nabla F^{-1}(0) = J^{-1}$. In terms of the new variables $u = F(y)$ we must solve

$$\begin{aligned} &\text{minimize } f\left(F^{-1}(u)\right) \\ &\text{subject to } u_i = 0, \qquad i = 1, \ldots, m \\ &\qquad\qquad g_i\left(F^{-1}(u)\right) \le 0, \qquad i \in I \end{aligned}$$

Setting $u_1 = \cdots = u_m = 0$, we obtain a problem involving only inequality constraints and the $n - m$ variables $u_{m+1}, \ldots, u_n$. If the assertions of the theorem hold for this problem, then they hold for the original problem, because $T_C(x)$, $K_C(x)$, $K'_C(x)$, and the rays $\{\, \lambda \nabla g_i(x) : \lambda \ge 0 \,\}$ along gradient vectors are geometric objects, not changed by transformation of coordinates. Hence without loss of generality, we may assume that the problem involves only inequality constraints.

If the second set of conditions, now reduced to (b) alone, hold then if $\lambda \ge 0$ and $\lambda \ne 0$ we have $\langle \sum_i \lambda_i \nabla g_i(x), v \rangle < 0$ for some vector v, and this is impossible if the first condition fails. Conversely, if (b) fails to hold, then there is no hyperplane strongly separating the origin and convex hull of the $\nabla g_i(x)$. Such a half-space must exist by the separating hyperplane theorem (Rockafellar, 1970, Corollary 11.4.2) unless the origin is in the convex hull of the $\nabla g_i(x)$, but then the first condition also fails to hold. This shows the equivalence of the two sets of conditions.

By Theorem 2.4.8 $T_C(x) \subset K_C(x)$. We need only prove the reverse inclusion. Suppose $w \in K'_C(x)$. The

$$\frac{g_i(x + \tau w) - g_i(x)}{\tau} \to \langle \nabla g_i(x), w \rangle < 0, \qquad \text{as } \tau \downarrow 0.$$

Hence the left hand side is strictly negative for $0 < \tau < \epsilon$ for some $\epsilon > 0$. Consequently $x + \tau w \in C$ for such τ, and $w \in T_C(x)$. Thus $T_C(x)$, being closed, contains the closure of $K'_C(x)$, which is $K_C(x)$. □

2.4.8 Second Order Conditions

Suppose the Kuhn-Tucker conditions hold. What are a necessary condition or a sufficient condition for the solution to be a local minimum? A sufficient condition is the simpler of the two.

An obvious sufficient condition is that the gradient of the Lagrangian be zero and Hessian of the Lagrangian strictly positive definite. Then it follows from Theorem 2.4.6 that the Lagrangian has a local minimum at the proposed solution, which is thus a local minimum of the constrained optimization problem. But this condition is far too strong.

Lemma 2.4.12. *Suppose the Kuhn-Tucker conditions hold at x with Lagrange multipliers λ, but x is not a local minimum of the problem with objective function f. Let x_n be a sequence in C converging to x such that $f(x_n) < f(x)$ and $(x_n - x)/\|x_n - x\| \to v$. Then $\langle \nabla g_i(x), v \rangle = 0$, for every $i \in A(x)$ such that $\lambda_i > 0$. Also $\langle \nabla g_i(x), v \rangle = 0$, for every $i \in E$.*

Proof. The assertion about equality constraints follows from Lemma 2.4.8. This also implies that $\langle \nabla g_i(x), v \rangle \leq 0$, $i \in A(x)$. Suppose to get a contradiction that $\lambda_k > 0$ and $\langle \nabla g_k(x), v \rangle < 0$ so $\lambda_k \langle \nabla g_k(x), v \rangle < 0$. Then the first Kuhn-Tucker condition and complementary slackness imply

$$-\langle \nabla f(x), v \rangle = \sum_{i \in E \cup A(x)} \lambda_i \langle \nabla g_i(x), v \rangle < 0 \, .$$

But our assumptions imply

$$0 \geq \frac{f(x_n) - f(x)}{\|x_n - x\|} \to \langle f(x), v \rangle \,,$$

so we have the contradiction and $\lambda_i > 0$ must imply $\langle \nabla g_i(x), v \rangle = 0$. □

This motivates the following definition. A constraint is *strongly active* if it is active and its Lagrange multiplier is nonzero. The set of such constraints is

$$A_s(x) = \{\, i \in I : g_i(x) = 0 \text{ and } \lambda_i > 0 \,\} \,.$$

Since the Lagrange multipliers are not necessarily uniquely defined, this depends on the choice of Lagrange multipliers. Also define the set

$$D = \{\, x : g_i(x) = 0,\ i \in E \cup A_s(x) \text{ and } g_i(x) \leq 0,\ i \in I \setminus A_s(x) \,\} \,.$$

Which is a subset of the original constraint set defined by imposing the strongly active constraints with equality.

The lemma implies that any direction v along which there is a sequence $x_n \to x$ such that $f(x_n) < f(x)$ lies in the closed convex cone

$$\begin{aligned} K_D(x) = \{\, v : &\langle \nabla g_i(x), v \rangle = 0,\ i \in E \cup A_s(x) \\ &\text{and } \langle \nabla g_i(x), v \rangle \leq 0,\ i \in A(x) \setminus A_s(x) \,\} \end{aligned}$$

Hence we only need to check the Hessian in directions along vectors $v \in K_D(x)$.

Theorem 2.4.13. *Suppose the Kuhn-Tucker conditions hold, $H = \nabla^2 L(x)$, and*

$$v^T H v > 0, \qquad \forall v \in K_D(x).$$

Then the optimization problem has a strict local minimum at x.

Proof. By complementary slackness and primal feasibility $f(x) = L(x)$, hence for sequences $x_n \to x$ in C and $\tau_n \downarrow 0$ such that $f(x_n) < f(x)$ and $(x_n - x)/\tau_n \to v$

$$\begin{aligned} 0 &\geq f(x_n) - f(x) \\ &= L(x_n) - \textstyle\sum_{i \in I} \lambda_i g_i(x_n) - L(x) \\ &\geq L(x_n) - L(x) \\ &= (x_n - x)^T H (x_n - x) + o(\|x_n - x\|^2) \end{aligned}$$

Dividing by τ_n^2 and letting n go to ∞ gives $0 \geq v^T H v$. By Lemma 2.4.12 $v \in K_D(x)$. But this contradicts the assumptions of the theorem. Hence there exists no such sequence x_n and x is a strict local minimum. □

It would be nice if $v^T H v \geq 0$, $v \in K_D(x)$ were a necessary condition, but, as with the first order conditions, there there is a gap involving constraint qualification. By Lemma 2.4.8 $T_D(x) \subset K_D(x)$, but the inclusion may be strict. If so, our necessary condition is weaker.

Theorem 2.4.14. *Suppose the Kuhn-Tucker conditions hold, $H = \nabla^2 L(x)$, and the optimization problem has a strict local minimum at x. Then*

$$v^T H v \geq 0, \qquad \forall\, v \in T_D(x).$$

Proof. For any $v \in T_D(x)$ there are $x_n \to x$ in D and $\tau_n \downarrow 0$ such that $(x_n - x)/\tau_n \to v$. Also $f(y) = L(y)$ for any $y \in D$. Thus for all large enough n

$$\begin{aligned} 0 &\leq f(x_n) - f(x) \\ &= L(x_n) - L(x) \\ &= (x_n - x)^T H (x_n - x) + o(\|x_n - x\|^2) \end{aligned}$$

Dividing by τ_n^2 and letting n go to ∞ gives $v^T H v \geq 0$. □

Theorem 2.4.11, of course, gives conditions under which $T_D(x) = K_D(x)$.

2.5 Appendix

2.5.1 Linear and Quadratic Functions

If f is a linear transformation represented by a matrix A so that $f(x) = Ax$, then

$$f(x + y) = f(x) + Ay,$$

which satisfies (2.7) (there is no little oh term). Thus $\nabla f(x) = A = f$. A linear function is its own derivative in the abstract view that derivatives are linear transformations. Moreover, the derivative is constant, not depending on x.

If q is a quadratic form, represented by a symmetric matrix A so that $q(x) = \frac{1}{2}\langle x, Ax\rangle$, then

$$\begin{aligned} q(x + y) &= \tfrac{1}{2}\langle x + y, A(x + y)\rangle \\ &= \tfrac{1}{2}\langle x, Ax\rangle + \langle Ax, y\rangle + \tfrac{1}{2}\langle y, Ay\rangle \\ &= q(x) + \langle Ax, y\rangle + \tfrac{1}{2}\langle y, Ay\rangle \end{aligned}$$

The last term on the right hand side is $o(\|y\|)$. Hence the derivative $\nabla q(x)$ is the linear map $y \mapsto \langle Ax, y\rangle$, which is represented by the vector Ax.

Thus q is differentiable everywhere and its derivative $\nabla q(x) = Ax$ is a linear function, considered as a function of x. Hence $\nabla^2 q(x) = A$, for all x.

The Chain Rule

If $f : U \to \mathbb{R}^m$, where U is a neighborhood of x in $\mathbb{R}^n$, is differentiable at x and if $g : V \to \mathbb{R}^k$, where V is a neighborhood of $f(x)$ in $\mathbb{R}^m$ is differentiable at $f(x)$, then the composition $g \circ f$ is defined on some set W which is a neighborhood of x in $\mathbb{R}^n$ and is also differentiable at x, the derivative being given by the *chain rule*

$$\nabla(g \circ f)(x) = \nabla g[f(x)]\nabla f(x).$$

The proof is very much like the proof of the chain rule for univariate functions. A differentiable function is continuous so $f^{-1}(V)$ is a neighborhood of x, hence so is $W = f^{-1}(V) \cap U$. Then

$$\begin{aligned} g[f(x+y)] - g[f(x)] &= \nabla g[f(x)][f(x+y) - f(x)] + o\big(f(x+y) - f(x)\big) \\ &= \nabla g[f(x)][\nabla f(x)y + o(\|y\|)] + o\big(\nabla f(x)y + o(\|y\|)\big) \\ &= \nabla g[f(x)]\nabla f(x)y + o(\|y\|) \end{aligned}$$

Linear and Quadratic Approximation

We say a function $f : U \to \mathbb{R}$, where U is a neighborhood of x in $\mathbb{R}^n$ has a *linear approximation* at x if

$$f(x+y) = a + \langle b, y\rangle + o(\|y\|) \tag{2.35}$$

for some $a \in \mathbb{R}$ and $b \in \mathbb{R}^n$. We say f has a *quadratic approximation* at x if

$$f(x+y) = a + \langle b, y\rangle + \tfrac{1}{2}\langle y, Hy\rangle + o(\|y\|^2) \tag{2.36}$$

for some $a \in \mathbb{R}$ and $b \in \mathbb{R}^n$ and some linear operator H on $\mathbb{R}^n$. Note that we may assume H is symmetric, since the bilinear form $\langle y, Hy\rangle$ only depends on the symmetric part of H.

Note also that we have changed terminology. Following the usage of linear algebra, a linear transformation f satisfies $f(0) = 0$ and a quadratic form q satisfies $q(0) = 0$ and $\nabla q(0) = 0$. Now we are changing to the usage common in statistics, that a linear function can include a constant term and a quadratic function can include constant and linear terms. We hope no confusion will result.

Theorem 2.5.1. *A function $f : U \to \mathbb{R}$, where U is a neighborhood of x in $\mathbb{R}^n$ has a linear approximation at x if and only if it is differentiable at x and in* (2.35) *$a = f(x)$ and $b = \nabla f(x)$.*

If f is twice differentiable at x then it has a unique quadratic approximation (2.36) *with $a = f(x)$, $b = \nabla f(x)$, and $H = \nabla^2 f(x)$.*

Proof. That differentiability implies a linear approximation is true by definition (2.7). Conversely, if f has a linear approximation (2.35), then setting $y = 0$ shows that $a = f(x)$ and then (2.35) satisfies the definition (2.7) with $\nabla f(x) = b$.

If f is twice differentiable, then

$$\nabla f(x + y) = \nabla f(x) + \nabla^2 f(x)y + o(\|y\|)$$

Let $u = y/\|y\|$ be the unit vector along y, and take the inner product with u giving

$$\langle \nabla f(x + y), u \rangle = \langle \nabla f(x), u \rangle + \left\langle u, \nabla^2 f(x)y \right\rangle + o(\|y\|)$$

Now write $s = \|y\|$, so that $y = su$ and

$$\langle \nabla f(x + su), u \rangle = \langle \nabla f(x), u \rangle + s \left\langle u, \nabla^2 f(x)u \right\rangle + o(s)$$

Note that by the chain rule, the left hand side is the ordinary derivative of $s \mapsto f(x+su)$, a real-valued function of one real variable. Now integrate with respect to s from zero to t, giving by the fundamental theorem of calculus

$$f(x + tu) - f(x) = t \langle \nabla f(x), u \rangle + \tfrac{1}{2} t^2 \left\langle u, \nabla^2 f(x)u \right\rangle + \int_0^t s\psi(s)\, ds$$

for some $o(1)$ function ψ. The integral is bounded by

$$\frac{t^2}{2} \sup_{0 \le s \le t} \psi(s)$$

which is $o(t^2)$. Now letting $\|y\| = t$ and plugging back in gives

$$f(x + y) = f(x) + \langle \nabla f(x), y \rangle + \tfrac{1}{2} \left\langle y, \nabla^2 f(x)y \right\rangle + o(\|y\|^2)$$

which is the desired quadratic approximation.

It remains only to be shown that this quadratic approximation is unique. If (2.36) holds, then setting $y = 0$ gives $a = f(x)$, and $\frac{1}{2} \langle y, Hy \rangle = o(\|y\|^2)$ so $b = \nabla f(x)$. This implies

$$\left\langle y, [H - \nabla^2 f(x)], y \right\rangle = o(\|y\|^2) \tag{2.37}$$

which can only happen if $H = \nabla^2 f(x)$, because if the left hand side of (2.37) is nonzero for any vector y, then

$$s \mapsto \left\langle sy, [H - \nabla^2 f(x)], sy \right\rangle$$

is a nonzero quadratic function of the scalar variable s, which is not $o(s^2)$. □

Corollary 2.5.2. *The second derivative of a scalar-valued function is a symmetric linear operator.*

Chapter 3

Optimization Algorithms

3.1 Overview of Algorithms

An optimization algorithm takes a starting point x_0 and generates a sequence of iterates $x_0, x_1, x_2, \ldots$ with the goal of better approximating a solution point at each step. The algorithms are based on a recursion, that is, given a point x_n a recursion will generate x_{n+1} with a lower value of the objective function, f. There are two basic strategies for creating a recipe for the recursion: line search and trust region.

For an algorithm that employs a line search strategy each iteration chooses a search direction s_n. Then it finds the α_n that minimizes the one dimensional function

$$w(\alpha) = f(x_n + \alpha s_n)$$

and sets $x_{n+1} = x_n + \alpha s_n$. An exact minimization of $w(\alpha)$ is expensive and turns out to be unnecessary. Instead the basic strategy is to simply approximate the minimum with a new step length and direction to obtain x_{n+1}.

Another, often more effective, strategy is the trust region method. The

idea here is to construct a model function w_n that mimics the behavior of the objective function in a region centered at the current step x_n. Since the model is not going to be a good approximation to f for all x we restrict the search for a minimum to a region centered at x_n. In this setting we nearly always use a quadratic model for f

$$w_n(x) = f(x_n) + (x - x_n)^T \nabla f(x_n) + \tfrac{1}{2}(x - x_n)^T \nabla^2 f(x_n)(x - x_n) \; .$$

But this model is only good in the neighborhood of the current iterate x_n, say for x satisfying $\|x - x_n\| \le h_n$ for some constant $h_n > 0$, which we call a *trust region* because that is where we "trust" the quadratic model. Thus, given x_n we minimize $w_n(x)$ over $\|x - x_n\| \le h_n$ to obtain x_{n+1}.

An important issue with any iterative procedure for minimizing f is that of convergence. That is, will the algorithm terminate at a sensible point. In some very special cases, such as quadratic programming, this may occur in a finite number of steps but for the general problem convergence is only possible in a limiting sense.

3.1.1 Big Oh Notation

"Big oh" and "little oh" notation are complementary. Between the two, we have a useful description of the convergence behavior of most functions.

"Big oh one" is just another name for locally bounded. A function $\psi : U \to \mathbb{R}^m$, where U is a neighborhood of zero in $\mathbb{R}^n$, is said to be $O(1)$, if

$$\limsup_{x \to 0} \psi(x) < \infty,$$

or in other words if there exists an $\epsilon > 0$ and $M < \infty$ such that

$$\psi(x) \le M, \qquad \|x\| < \epsilon.$$

The same caution we gave for little oh notation also applies to big oh notation: it is a code not decoded according to the usual rules of mathematics.

More generally, given two functions f and g from a neighborhood U of zero in $\mathbb{R}^n$ to $\mathbb{R}^m$, we say that f is $O\big(g(x)\big)$, read "big oh of $g(x)$" if

$$f(x) = |g(x)|\psi(x)$$

for some $O(1)$ function ψ.

3.1.2 Types of Convergence

Suppose an iterative algorithm converges, that is, the iterates x_n converge to a local minimum x. Let $\varepsilon_n = x_n - x$ be the error at iteration n. The algorithm converges *linearly* if

$$\|\varepsilon_{n+1}\| = O(\|\varepsilon_n\|), \tag{3.1}$$

converges *quadratically* if

$$\|\varepsilon_{n+1}\| = O(\|\varepsilon_n\|^2), \tag{3.2}$$

and converges *superlinearly* if

$$\|\varepsilon_{n+1}\| = o(\|\varepsilon_n\|). \tag{3.3}$$

Note that linear convergence doesn't guarantee much since it doesn't even imply $\varepsilon_n \to 0$, though this is implied by the word "convergence" in "linear convergence."

3.2 Newton

Newton's algorithm is more commonly called the *Newton-Raphson* algorithm by statisticians, but it is so important in optimization and has so many variants, quasi-Newton, safeguarded Newton, and so forth, that the longer eponym would be cumbersome. Newton's algorithm is a method of solving

simultaneous nonlinear equations. Suppose $g : \mathbb{R}^n \to \mathbb{R}^n$ is a differentiable map and we are to solve the equation $g(x) = 0$. Write $J(x) = \nabla g(x)$. Now $J(x)$ is an $n \times n$ matrix, generally nonsymmetric, called the *Jacobian* of the map g at the point x. At any point x_n

$$g(x) = g(x_n) + J(x_n)(x - x_n) + o(\|x - x_n\|).$$

Setting this to zero and ignoring higher order terms, yields

$$x = x_n - J(x_n)^{-1} g(x_n)$$

if $J(x_n)$ is nonsingular.

If the one-term Taylor expansion is a perfect approximation, this is the solution. In general it is not, but we take it to be the next point in an iterative scheme. Let x_0 be any point, and generate a sequence $x_1, x_2, x_3, \ldots$ by

$$x_{n+1} = x_n - J(x_n)^{-1} g(x_n).$$

In the context of unconstrained optimization, Newton's method tries to find a zero of the gradient of the objective function f. Write $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$ for the gradient and Hessian of the objective function, then $H(x)$ is the Jacobian of $g(x)$, and the *Newton update* becomes

$$x_{n+1} = x_n - H(x_n)^{-1} g(x_n).$$

Now the Hessian is a symmetric matrix (unlike a general Jacobian).

Another way to look at Newton's algorithm applied to optimization is that it replaces the objective function f with a quadratic model

$$w(x) = f(x_n) + (x - x_n)' g(x_n) + \tfrac{1}{2}(x - x_n)' H(x_n)(x - x_n) \; . \tag{3.4}$$

The model function w has no minimum unless $H(x_n)$ is positive definite. It makes no sense to accept a Newton update unless the Hessian is positive definite.

3.2.1 What's Bad About Newton

Despite the comments at the end of the preceding section, Newton is *not* an optimization algorithm. It pays no attention to the values $f(x_n)$ of the objective function and indeed need not compute them if it can compute the gradient and Hessian without computing f. It just as happily goes uphill as downhill and indeed has no idea which way it is going.

If started close enough to a strict local minimum, Newton's method will converge. But it is usually impossible to tell how close is close enough. Kantorovich and Akilov (1964, Ch. 18) describes what is known about the convergence properties of Newton's method. In general it can be difficult to verify the conditions that imply convergence.

Even in very simple problems, Newton fails if started far from the solution. Consider maximum likelihood for a binomial distribution with one success and one failure. The log likelihood is

$$l(\theta) = \log(p(\theta)) + \log(1 - p(\theta)),$$

where

$$p(\theta) = \frac{e^\theta}{1 + e^\theta}$$

The first and second derivatives with respect to θ of the log likelihood are

$$1 - 2p(\theta) \qquad \text{and} \qquad -2p(\theta)(1 - p(\theta)),$$

respectively. The Newton update is thus

$$\theta_{n+1} = \theta_n + \frac{1 - 2p(\theta_n)}{2p(\theta_n)(1 - p(\theta_n))} \,.$$

Define $s(\theta) = 1 - 2p(\theta)$ and $i(\theta) = 2p(\theta)(1 - p(\theta))$. If one starts close enough to the solution ($\theta = 0$, $p = 1/2$), Newton converges quickly.

θ	p	$l(\theta)$	$s(\theta)$	$i(\theta)$	step
1.00000	0.73106	−1.62652	−0.46212	0.39322	−1.1752
−0.17520	0.45631	−1.39396	0.08738	0.49618	0.1761
0.00090	0.50022	−1.38629	−0.00045	0.50000	−0.0009
-1.2×10^{-10}	0.50000	−1.38629	0.00000	0.50000	0.0000

Great! But if one starts a bit farther away

θ	p	$l(\theta)$	$s(\theta)$	$i(\theta)$	step
3.00000	0.95257	−3.09717	−0.90515	0.09035	−10.01787
−7.01787	0.00089	−7.01967	0.99821	0.00179	558.20537
551.18750	1.00000	$-\infty$	−1.00000	0.00000	$-\infty$

where $-\infty$ indicates overflow of the computer's floating-point arithmetic. The program crashes.

3.2.2 What's Good About Newton

When it converges, Newton converges superlinearly, usually quadratically.

Theorem 3.2.1. *Suppose x_n is a sequence of Newton iterations for minimizing an objective function f converging to a local minimum x^*. Suppose $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$ are continuous in a neighborhood of x^* and $H(x^*)$ is positive definite. Then Newton is superlinearly convergent.*

Proof. By the assumptions about $g(x)$ and $H(x)$,

$$g(y) = g(x) + H(x)(y - x) + o(\|y - x\|) \tag{3.5}$$

holds for all x and y in some neighborhood of x^*, and

$$H(y) = H(x^*) + o(1)$$

A characterization of the Newton update is

$$0 = g(x_n) + H(x_n)(x_{n+1} - x_n). \tag{3.6}$$

Plugging in Taylor expansions around x^* for $g(x_n)$ and $H(x_n)$ in (3.6) gives

$$\begin{aligned} 0 &= g(x^*) + H(x^*)(x_n - x^*) + o(\|x_n - x^*\|) + [H(x^*) + o(1)]\,(x_{n+1} - x_n) \\ &= H(x^*)(x_{n+1} - x^*) + o(\|x_n - x^*\|) + o(\|x_{n+1} - x_n\|) \end{aligned}$$

because $g(x^*) = 0$. Writing $\varepsilon_n = x_n - x^*$ gives

$$\begin{aligned} 0 &= H(x^*)\varepsilon_{n+1} + o(\|\varepsilon_n\|) + o(\|\varepsilon_{n+1} - \varepsilon_n\|) \\ &= H(x^*)\varepsilon_{n+1} + o(\|\varepsilon_n\|) + o(\|\varepsilon_{n+1}\|) \\ &= H(x^*)\varepsilon_{n+1} + o(\|\varepsilon_n\|), \end{aligned}$$

where we have used the triangle inequality and the fact that $o(\|\varepsilon_{n+1}\|)$ is negligible compared to $H(x^*)\varepsilon_{n+1}$. Since $H(x^*)$ is positive definite, it is invertible. This proves (3.3). □

Quadratic convergence requires a bit more than (3.5).

Theorem 3.2.2. *Suppose x_n is a sequence of Newton iterations for a function f converging to a local minimum x^*. Let $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$. Suppose $H(x^*)$ is positive definite, and suppose*

$$g(y) = g(x) + H(x)(y - x) + O(\|y - x\|^2) \tag{3.7}$$

and

$$H(y) = H(x) + O(\|y - x\|) \tag{3.8}$$

for all x and y in some neighborhood of x^*. Then Newton converges quadratically.

Equation (3.8) is referred to as a *Lipschitz* condition. Equation (3.7) is similar, but not usually referred to by that terminology. Both would be implied by Taylor's theorem with remainder if third derivatives of f exist.

Proof. A characterization of the Newton update is

$$0 = g(x_n) + H(x_n)(x_{n+1} - x_n).$$

Using (3.7) and (3.8) to expand around x^* gives

$$\begin{aligned} 0 = g(x^*) + H(x^*)(x_n - x^*) + O(\|x_n - x^*\|^2) \\ + \left[H(x^*) + O(\|x_n - x^*\|)\right](x_{n+1} - x_n) \end{aligned}$$

Since x^* is a local min, $g(x^*) = 0$. Thus, writing $\varepsilon_n = x_n - x^*$,

$$\begin{aligned} 0 &= H(x^*)\varepsilon_{n+1} + O\left(\|\varepsilon_n\|^2 + \|\varepsilon_n\| \, \|\varepsilon_{n+1} - \varepsilon_n\|\right) \\ &= H(x^*)\varepsilon_{n+1} + O\left(\|\varepsilon_n\|^2 + \|\varepsilon_n\| \, \|\varepsilon_{n+1}\|\right) \\ &= H(x^*)\varepsilon_{n+1} + O\left(\|\varepsilon_n\|^2\right) \end{aligned}$$

The last equality using Theorem 3.2.1. Since $H(x^*)$ is a fixed, positive definite, invertible matrix, this proves (3.2). □

Not only does Newton converge quadratically (under fairly weak regularity conditions). Any algorithm that converges superlinearly is asymptotically equivalent to Newton.

Theorem 3.2.3 (Dennis-Moré)**.** *Suppose $x_n \to x^*$ is a sequence of iterations of an optimization algorithm converging to a local minimum of f. Let $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$, and suppose $H(x^*)$ is positive definite. If the algorithm converges superlinearly, then it is asymptotically equivalent to Newton, in the sense that*

$$x_{n+1} - x_n = \Delta_n + o(\|\Delta_n\|), \tag{3.9}$$

where

$$\Delta_n = -H(x_n)^{-1} g(x_n)$$

is the Newton step at x_n.

Proof. Write $\delta_n = x_{n+1} - x_n$ for the steps taken by the algorithm, and write $\varepsilon_n = x_n - x^*$. So $\varepsilon_{n+1} = \delta_n + \varepsilon_n$. The hypothesis of superlinear convergence is that $\varepsilon_{n+1} = o(\|\varepsilon_n\|)$ or that $\delta_n = -\varepsilon_n + o(\|\varepsilon_n\|)$. Similarly the superlinear convergence of Newton asserted by Theorem 3.2.1 implies $\Delta_n = -\varepsilon_n + o(\|\varepsilon_n\|)$, and this implies $\delta_n = \Delta_n + o(\|\varepsilon_n\|)$ and $\delta_n = \Delta_n + o(\|\Delta_n\|)$. The latter is (3.9). □

Corollary 3.2.4. *Every superlinearly convergent algorithm is asymptotically equivalent to any other superlinearly convergent algorithm.*

3.2.3 Fisher Scoring

Fisher scoring is Newton modified by replacing observed with expected Fisher information. To see what this means, we need some definitions. Let $l_n(\theta)$ denote the log likelihood for a statistical model (n indicates sample size, which is fixed throughout most of the discussion). The maximum likelihood problem is to find the point $\hat{\theta}_n$, called the *maximum likelihood estimate* (MLE), which maximizes l_n. The derivative of l_n

$$s_n(\theta) = \nabla l_n(\theta)$$

is the *score*, and minus the second derivative

$$J_n(\theta) = -\nabla^2 l_n(\theta)$$

is the *observed Fisher information*. Both of these quantities are random variables, depending on the data, although this is not indicated by the notation. When we are finding the MLE we consider the data fixed at the observed value, in which case s_n and J_n are just ordinary functions, but when we calculate the expected Fisher information we do consider the data random.

Since $J_n(\theta)$ is minus the Hessian of l_n at θ, the Newton update is

$$\theta_{k+1} = \theta_k + J_n(\theta_k)^{-1} s_n(\theta_k). \tag{3.10}$$

The expectation of $J_n(\theta)$

$$I_n(\theta) = E\{J_n(\theta)\}$$

is called the *expected Fisher information*. The Fisher scoring update replaces (3.10) with

$$\theta_{k+1} = \theta_k + I_n(\theta_k)^{-1} s_n(\theta_k). \tag{3.11}$$

Theorem 3.2.5. *Let $\{\theta_k\}$ denote the sequence of iterates of a Fisher scoring algorithm, and suppose $\theta_k \to \theta^*$. Suppose the observed and expected Fisher information functions J_n and I_n are continuous at θ^* and that $J_n(\theta^*)$, $I_n(\theta^*)$ and $J_n(\theta^*) - I_n(\theta^*)$ have full rank. Then the Fisher scoring algorithm is not superlinearly convergent.*

Proof. As before write $\varepsilon_k = \theta_k - \theta^*$ and

$$\begin{aligned} \delta_k &= \theta_{k+1} - \theta_k \\ &= I_n(\theta_k)^{-1} s_n(\theta_k) \\ \Delta_k &= J_n(\theta_k)^{-1} s_n(\theta_k) \end{aligned}$$

for the Fisher scoring and Newton steps, respectively. Then $\delta_k \to 0$ because θ_k converges. Also s_n is continuous at θ^* because it is differentiable there. Hence $s_n(\theta_k) \to s_n(\theta^*)$ and

$$s_n(\theta_k) = I_n(\theta_k)\delta_k \to I_n(\theta^*) \cdot 0 = 0$$

so $s_n(\theta_*) = 0$. Since $J_n(\theta^*)$ is assumed invertible and J_n continuous at θ^*, it follows that $J_n(\theta)$ is invertible in some neighborhood of θ^* so $J_n(\theta_k)$ is invertible (and the Newton step well defined) for all sufficiently large k, and

$$\Delta_k = J_n(\theta_k)^{-1} s_n(\theta_k) \to J_n(\theta^*)^{-1} \cdot 0 = 0.$$

Now write $u_k = s_n(\theta_k)/\|s_n(\theta_k)\|$ and choose a subsequence so that $u_{k_l} \to u$ (which is always possible because the u_k are unit vectors and the closed unit ball is compact).

Now

$$\frac{\delta_{k_l}}{\|s_n(\theta_{k_l})\|} = J_n(\theta_{k_l})^{-1}u_{k_l} \to J_n(\theta^*)^{-1}u$$

and

$$\frac{\Delta_{k_l}}{\|s_n(\theta_{k_l})\|} = I_n(\theta_{k_l})^{-1}u_{k_l} \to I_n(\theta^*)^{-1}u$$

In order for Fisher scoring and Newton to be asymptotically equivalent, the limits must agree, that is, Fisher scoring cannot be superlinearly convergent unless $J_n(\theta^*)^{-1}u = I_n(\theta^*)^{-1}u$, or equivalently unless $J_n(\theta^*)v = I_n(\theta^*)v$, where $v = J_n(\theta^*)^{-1}u$, but this contradicts the assumption that $J_n(\theta^*) - I_n(\theta^*)$ is full rank. □

Example 3.2.1. *Recall the example in Section 3.2.1 where we considered maximum likelihood for a binomial distribution with one success and one failure. The log likelihood is*

$$l(\theta) = \log(p(\theta)) + \log(1 - p(\theta)),$$

where

$$p(\theta) = \frac{e^\theta}{1 + e^\theta}$$

The first and second derivatives with respect to θ of the log likelihood are

$$1 - 2p(\theta) \qquad \text{and} \qquad -2p(\theta)(1 - p(\theta)),$$

respectively. The expected Fisher information is $p(\theta)(1-p(\theta))(3-p(\theta))$. The Fisher scoring update is thus

$$\theta_{k+1} = \theta_k + \frac{1 - 2p(\theta_k)}{p(\theta_k)(1 - p(\theta_k))(3 - p(\theta_k))} .$$

Define $s(\theta) = 1 - 2p(\theta)$ and $i(\theta) = p(\theta_k)(1 - p(\theta_k))(3 - p(\theta_k))$. The Fisher scoring algorithm converges more slowly than Newton when given a good starting point, $\theta = 1$, and doesn't improve on Newton when started from the bad point, $\theta = 3$.

θ	p	$l(\theta)$	$s(\theta)$	$i(\theta)$
1.00000	0.731059	-1.62652	-0.462117	0.446101
-0.0359026	0.491025	-1.38662	0.0179494	0.627042
-0.00727712	0.498181	1.38631	0.00363854	0.625447
-0.00145961	0.499635	-1.38629	0.000729803	0.625091
-0.000292091	0.499927	-1.38629	0.000146046	0.625018
-5.8425e-05	0.499985	-1.38629	2.92125e-05	0.625004
-1.16853e-05	0.499997	-1.38629	5.84264e-06	0.625001
-2.33707e-06	0.499999	-1.38629	1.16853e-06	0.62500
-4.67414e-07	0.500000	-1.38629	2.33707e-07	0.62500

θ	p	$l(\theta)$	$s(\theta)$	$i(\theta)$
3.0000	0.952574	-3.09717	-0.905148	0.0924959
-6.78582	0.0011284	-6.78808	0.997743	0.00338011
288.395	1	$-\infty$	-1	0
$-\infty$	0	.NaN	1	0

3.3 Descent Methods

Newton is great when close to the solution. Far from the solution, we need something else. A standard scheme is to force reduction in the objective function in each iteration by making a line search. Each iteration chooses a search direction s_n, perhaps the direction of the Newton step, perhaps not. Then it finds the α_n that minimizes the one dimensional function

$$w(\alpha) = f(x_n + \alpha s_n)$$

and sets $x_{n+1} = x_n + \alpha_n s_n$. In order that there exist $\alpha > 0$ such that $w(\alpha) < w(0)$, we require the search direction s_n to satisfy the *descent property* $w'(0) < 0$ or

$$g(x_n)^t s_n < 0 \tag{3.12}$$

where $g(x) = \nabla f(x)$.

Requiring α_n to be the exact minimizer of $w(\alpha)$ is expensive and unnecessary. That is, in practice it makes no sense to waste a lot of time polishing the solution α_n when the solution to the line search subproblem isn't the solution to the main problem. Thus we want criteria for an inexact line search that permit a proof about the properties of descent methods. The following closely follows Fletcher (1987, Section 2.5). Let $0 < \rho < \frac{1}{2}$ and $\rho < \sigma < 1$, and let α_n be any α satisfying

$$w(\alpha) \leq w(0) + \alpha\rho w'(0) \tag{3.13}$$

and

$$w'(\alpha) \geq \sigma w'(0) \tag{3.14}$$

where (3.13) was proposed by Goldstein and (3.14) by Wolfe (see Fletcher (1987) for citations). In terms of f rather than w, these imply

$$f(x_n) - f(x_{n+1}) \geq -\rho g(x_n)^T \delta_n \tag{3.15}$$

and

$$g(x_{n+1})^T \delta_n \geq \sigma g(x_n)^T \delta_n \tag{3.16}$$

where

$$\delta_n = \alpha_n s_n = x_{n+1} - x_n$$

is the step taken in the iteration. The angle θ_n between the gradient and the step is

$$\cos\theta_n = -\frac{g(x_n)^T \delta_n}{\|g(x_n)\| \|\delta_n\|} \,. \tag{3.17}$$

We say a method satisfies the *angle criterion* if (3.17) is bounded away from zero.

Theorem 3.3.1. *For a descent method with inexact line search satisfying* (3.13) *and* (3.14)*, if $g(x) = \nabla f(x)$ is uniformly continuous on a set containing all the iterations, if $f(x_n)$ is bounded below, and if* (3.17) *is bounded away from zero, then $g(x_n) \to 0$.*

Proof. Since $f(x_n)$ is bounded below, $f(x_n) - f(x_{n+1}) \to 0$ and (3.13) and (3.12) imply $g(x_n)^T \delta_n \to 0$. Suppose to get a contradiction that $g(x_n)$ fails to converge to zero, so there is a $c > 0$ and a subsequence x_{n_k} such that $g(x_{n_k}) \geq c$. Then $\delta_{n_k} \to 0$.

Now (3.16) implies

$$[g(x_{n+1}) - g(x_n)]^T \delta_n \geq (\sigma - 1) g(x_n)^T \delta_n$$

or

$$-g(x_n)^T \delta_n \leq \frac{[g(x_{n+1}) - g(x_n)]^T \delta_n}{1 - \sigma} \leq \frac{\|g(x_{n+1}) - g(x_n)\| \, \|\delta_n\|}{1 - \sigma} \tag{3.18}$$

Uniform continuity of $g(x)$ means that for every $\epsilon > 0$ there is a $\eta > 0$ such that

$$\|g(x + \alpha s) - g(x)\| \leq \epsilon$$

whenever $0 \leq \alpha \leq \eta$ for all x and for all unit vectors s. Hence $\alpha_{n_k} = \|\delta_{n_k}\| \to 0$ implies

$$\|g(x_{n_k+1}) - g(x_{n_k})\| = \|g(x_{n_k} + \alpha_{n_k} s_{n_k}) - g(x_{n_k})\| \to 0.$$

Combining (3.18) and (3.17) gives

$$\|g(x_n)\| \, \|\delta_n\| \cos\theta_n \leq \frac{\|g(x_{n+1}) - g(x_n)\| \, \|\delta_n\|}{1 - \sigma}$$

Hence

$$0 \leq \cos\theta_{n_k} \leq \frac{\|g(x_{n_k+1}) - g(x_{n_k})\|}{(1 - \sigma)\|g(x_{n_k})\|} \to 0$$

But this contradicts the angle criterion. Hence the assumption that $g(x_n)$ fails to converge to zero was false. □

Note that the theorem does not assert that x_n converges. Consider the one-dimensional function $f(x) = \exp(-x)$, which is minimized as $x \to \infty$. The conditions of the theorem are trivially satisfied, hence $\nabla f(x_n) = -f(x_n)$ converges to zero, but that requires $x_n \to \infty$.

If, however, the level set $B = \{ x : f(x) \leq f(x_1) \}$ is bounded, since $f(x_n)$ is decreasing, x_n cannot escape B. So B is closed, hence compact, because f is continuous. Thus there is no escape to infinity. Every subsequence has cluster points. For any subsequence x_{n_k} converging to a cluster point x^*, the theorem and the assumed continuity of $\nabla f(x)$ imply $\nabla f(x_{n_k}) \to \nabla f(x^*) = 0$. Thus every cluster point is a stationary point of f. More cannot be said. In practice, the line search will usually force convergence to a local minimum, but does not guarantee this.

A good method of choosing a descent direction is to use the Newton direction

$$s_n = -\frac{\nabla^2 f(x_n)^{-1} \nabla f(x_n)}{\| \nabla^2 f(x_n)^{-1} \nabla f(x_n) \|} \tag{3.19}$$

when this definition satisfies the descent property, which it must if $\nabla^2 f(x_n)$ is positive definite. But if $\nabla f(x)$ is even positive semi-definite for all $x \in B$, then f is convex on B, a strong property that will not always hold in applications. Hence the choice (3.19) will not always work. A descent algorithm must detect when (3.19) fails to satisfy the descent property and make some other choice, such as $s_n = -\nabla f(x_n)$, the steepest descent direction.

3.4 The EM algorithm

The EM algorithm was named by Dempster et al. (1977) but had been used by many earlier authors. Dempster et al. (1977) proved some of the basic properties of the algorithm but also claimed convergence properties which were not true. The erroneous claims were corrected by Boyles (1983) and Wu (1983), although the conditions implying convergence in the theorems in these papers are difficult to verify.

The EM algorithm is an algorithm for doing maximum likelihood in problems with missing data. There is a family of probability densities $f_\theta(x, y)$ of variables x and y, which may both be multivariate. Only y is observed. So

x is "missing data." The likelihood for the parameter θ is

$$L(\theta) = f_\theta(y) = \int f_\theta(x, y)\, dx. \tag{3.20}$$

The integral here may be intractable, making maximum likelihood difficult, in which case the EM algorithm may be useful. The EM algorithm also applies to problems that are formally similar, problems with latent variables, random effects, or mixtures and empirical Bayes problems.

One of the most popular methods for specifying $f_\theta(x, y)$ is the two-stage hierarchical model (HM). That is, the conditional density of $Y|X$ is specified as $f(y|x; \theta_1)$ while the marginal density of X is $h(x; \theta_2)$ and $\theta = (\theta_1, \theta_2)$.

Example 3.4.1. *Undoubtedly, the most important special case of the HM is the usual normal theory mixed model (McCulloch and Searle, 2001, Chapter 6). Let X and Z be known design matrices of dimension $n \times p$ and $n \times q$, respectively. Suppose $Y|u \sim N(X\beta + Zu, R)$ and $U \sim N(0, D)$. Then $Y \sim N(X\beta, R + ZDZ^T)$ and hence the likelihood* (3.20) *is available in closed form. However, finding maximum likelihood estimates (MLEs) often requires a numerical technique.*

If Θ is the parameter space, and φ is any point in Θ, define a function $Q_\varphi : \Theta \to \mathbb{R}$ by

$$Q_\varphi(\theta) = E_\varphi\{\log f_\theta(X, Y)|Y = y\} = \int \big(\log f_\theta(x, y)\big) f_\varphi(x|y)\, dx \tag{3.21}$$

where

$$f_\theta(x|y) = \frac{f_\theta(x, y)}{f_\theta(y)} = \frac{f_\theta(x, y)}{\int f_\theta(x, y)\, dx}$$

An iteration of the EM algorithm maximizes Q_φ rather than L. Of course, this doesn't solve the problem. The maximizer of Q_φ is not the maximizer of L. So what the EM algorithm does is generate a sequence of iterates θ_1, θ_2, ... having the property that if the current iterate is θ_k then the next iterate

is found by maximizing Q_{θ_k}, that is,

$$Q_{\theta_k}(\theta_{k+1}) = \sup_{\theta \in \Theta} Q_{\theta_k}(\theta)$$

Example 3.4.2. *Suppose f and g are densities with common support. Then if $0 < p < 1$*

$$h(y) = pf(y) + (1-p)g(y)$$

is also a density. Moreover, h is the marginal of the joint density defined by

$$f(y|x) = f(y)I(x=1) + g(y)I(x=0) \qquad X \sim \textit{Binomial}(1,p) \; .$$

Now suppose $Y_1, \ldots, Y_n \overset{iid}{\sim} h$. Then the likelihood is

$$L(p|X,Y) = \prod_{i=1}^{n} [pf(y_i)]^{x_i} [(1-p)g(y_i)]^{1-x_i}$$

and the log-likelihood is

$$l(p|x,y) = \sum_{i=1}^{n} x_i \log(pf(y_i)) + (1-x_i)\log((1-p)g(y_i)) \; .$$

Now $X_i|Y_i, p \sim \textit{Binomial}(1, pf(y_i)/(pf(y_i) + (1-p)g(y_i)))$. Hence

$$\begin{aligned} Q_{p'}(p) &= E[l(p|x,y)|y,p'] \\ &= \sum_{i=1}^{n} [(1-\lambda_i)\log((1-p)g(y_i)) + \lambda_i \log(pf(y_i))] \end{aligned}$$

where

$$\lambda_i = \frac{p' f(y_i)}{p' f(y_i) + (1-p')g(y_i)} \; .$$

Now we need to maximize Q with respect to p

$$Q'_{p'}(p) = \frac{\sum_{i=1}^{n} \lambda_i}{p(1-p)} - \frac{n}{1-p}$$

and setting $Q'_{p'}(p) = 0$ and solving we obtain $p = \bar{\lambda}$, which is easily shown to be a maximum. Here is the EM updating rule

$$p_{k+1} = \frac{1}{n} \sum_{i=1}^{n} \frac{p_k f(y_i)}{p_k f(y_i) + (1-p_k)g(y_i)} \; .$$

Under certain conditions θ_n does converge to the maximizer of the likelihood (3.20). Write $l(\theta) = \log L(\theta)$. Define

$$H_\varphi(\theta) = E_\varphi\{\log f_\theta(X|Y)|Y = y\}. \tag{3.22}$$

Note that Q_φ is the conditional expectation of the log of a joint density while H_φ is the conditional expectation of the log of a conditional density.

It is tacitly assumed here that these definitions make sense for all θ and φ. It can happen that the likelihood $l(\theta)$ is well defined even though the complete data likelihood is not. This happens in variance components problems as shown in the following example.

Example 3.4.3. *Suppose*

$$y = \mu + b_1 + \cdots + b_p$$

and the b_i are independent Normal$(0, \theta_i G_i)$ where the G_i are known matrices. The likelihood $l(\theta)$ is well-defined unless all of the elements of y are the same. The maximum of $l(\theta)$ can occur on the boundary of the parameter space where some of the θ_i are zero. The likelihood is complicated,

$$l(\theta, \mu) = -\tfrac{1}{2}(y-\mu)'V^{-1}(\theta)(y-\mu) - \tfrac{1}{2}\log\det V(\theta),$$

where $V(\theta) = \sum_i \theta_i G_i$ is the variance of y and det *denotes the determinant of a matrix. The complete data log likelihood is simpler. Ignoring terms not containing the parameters, the log likelihood for the b_i is*

$$-\tfrac{1}{2}\sum_i \theta_i^{-1} b_i' G_i^{-1} b_i - \tfrac{n}{2}\sum_i \log\theta_i$$

thus

$$Q_\varphi(\theta, \mu) = -\tfrac{1}{2}\sum_i \theta_i^{-1} E\varphi\{b_i' G_i^{-1} b_i | y - \mu\}) - \tfrac{n}{2}\sum_i \log\theta_i.$$

If all of the terms $E\varphi\{b_i' G_i^{-1} b_i | y-\mu\}$ are strictly positive, Q_φ has a maximum in the interior of the parameter space since it goes to $-\infty$ when any θ_i goes

to zero. But if one of the φ_i is zero, then b_i has variance zero and then the part of Q_φ that depends on θ_i is just $-\frac{n}{2}\log\theta$, which goes to $+\infty$ as $\theta_i \to 0$.

It is still possible to define the EM iteration by continuity. For φ in the interior of the parameter space, Q_φ achieves its maximum when

$$\begin{aligned}\mu &= \bar{y} \\ \theta_i &= \tfrac{1}{n} E\varphi\{b_i' G_i^{-1} b_i' | y - \mu\}, \qquad i = 1, \ldots, p.\end{aligned}$$

These equations also make sense on the boundary, but they have the property that if $\theta_i = 0$, then the EM iteration leaves it zero forever. If the solution is in the interior of the parameter space, then EM must be started in the interior to converge to the solution.

The tacit assumption mentioned above that H_φ and Q_φ are not infinite for φ of interest will be made throughout unless something is explicitly said to the contrary. For the variance components, that means φ is in the interior of the parameter space.

Lemma 3.4.1.

$$l(\theta) = Q_\varphi(\theta) - H_\varphi(\theta), \qquad \textit{for all } \theta \textit{ and } \varphi. \tag{3.23}$$

Proof.

$$\begin{aligned}H_\varphi(\theta) &= E_\varphi\{\log f_\theta(X|Y)|Y = y\} \\ &= E_\varphi\left\{ \log\left(\frac{f_\theta(X,Y)}{f_\theta(Y)}\right) \,\middle|\, Y = y \right\} \\ &= E_\varphi\big\{\log f_\theta(X,Y) \,\big|\, Y = y\big\} - E_\varphi\big\{\log f_\theta(Y) \,\big|\, Y = y\big\} \\ &= Q_\varphi(\theta) - \log f_\theta(y) \\ &= Q_\varphi(\theta) - l(\theta)\end{aligned}$$

□

Lemma 3.4.2. *H_φ achieves its global maximum at φ, i.e., $H_\varphi(\varphi) \geq H_\varphi(\theta)$ for all θ.*

Proof.

$$\begin{aligned} H_\varphi(\theta) - H_\varphi(\varphi) &= E_\varphi\left\{ \log\left(\frac{f_\theta(X|Y)}{f_\varphi(X|Y)}\right) \,\middle|\, Y = y \right\} \\ &\leq \log\left(E_\varphi\left\{ \frac{f_\theta(X|Y)}{f_\varphi(X|Y)} \,\middle|\, Y = y \right\} \right) \\ &= \log\left(\int \frac{f_\theta(x|y)}{f_\varphi(x|y)} f_\varphi(x|y)\, dx \right) \\ &= \log\left(\int f_\theta(x|y)\, dx \right) \\ &= \log(1) = 0 \end{aligned}$$

The inequality is Jensen's inequality, which applies to conditional as well as unconditional expectations (Chung, 1974, p. 302). □

This lemma has nothing special to do with EM. As a statement about unconditional rather than conditional probabilities it is familiar in many contexts, including the consistency of maximum likelihood. It says that the Kullback-Leibler information "distance" between densities f_θ and f_φ is minimized when $\theta = \varphi$. The extension to conditional densities is trivial because Jensen applies to both.

Theorem 3.4.3.

$$l(\theta_{n+1}) - l(\theta_n) \geq Q_{\theta_n}(\theta_{n+1}) - Q_{\theta_n}(\theta_n)$$

Hence every EM iteration increases the log likelihood unless the M-step cannot increase Q_{θ_n}.

Proof. By the lemmas

$$\begin{aligned} l(\theta_{n+1}) - l(\theta_n) &= Q_{\theta_n}(\theta_{n+1}) - Q_{\theta_n}(\theta_n) - [H_{\theta_n}(\theta_{n+1}) - H_{\theta_n}(\theta_n)] \\ &\geq Q_{\theta_n}(\theta_{n+1}) - Q_{\theta_n}(\theta_n) \end{aligned}$$

□

Lemma 3.4.4. *If Q_φ and H_φ are defined on some open set Ω in the parameter space and differentiable at every point of Ω, then*

$$\nabla l(\theta) = \nabla Q_\theta(\theta), \qquad \text{for all } \theta \in \Omega.$$

Proof. Differentiate (3.23). $\nabla H_\varphi(\varphi) = 0$, because H_φ has a global maximum at φ. □

Theorem 3.4.5. *If Q_φ and H_φ are defined on some open set Ω in the parameter space and differentiable at every point of Ω, if the map $(\varphi, \theta) \mapsto \nabla Q_\varphi(\theta)$ is (jointly) continuous, and if the EM sequence θ_n is contained in Ω and converges to a point θ^*, then $\nabla l(\theta^*) = 0$.*

Proof. Since θ_{n+1} maximizes Q_{θ_n}

$$0 = \nabla Q_{\theta_n}(\theta_{n+1}) \to \nabla Q_{\theta^*}(\theta^*).$$

Hence

$$\nabla l(\theta^*) = \nabla Q_{\theta^*}(\theta^*) - \nabla H_{\theta^*}(\theta^*) = 0$$

□

This theorem says that *if EM converges,* it converges to a stationary point of the log likelihood. It does not say that EM does converge. Even if a compactness argument implies convergent subsequences, this argument cannot be adapted to apply to subsequences. It is however a simple way to see most of what is true about EM.

1. When EM converges, the limit is not guaranteed to be a local maximum, rather it can be a saddle point. An example is given by Murray (1977).

2. If EM converges to a local maximum, this need not be the global maximum of the log likelihood, despite each M-step finding the global maximum of Q_{θ_n}. A number of examples of EM converging to local maxima that are not global maxima are given in papers cited in Wu (1983).

It is possible to prove something about subsequences, but the argument is more complicated. Dempster et al. (1977) define a GEM algorithm (for "generalized" EM) to be any algorithm that produces a sequence θ_n such that $Q_{\theta_n}(\theta_{n+1}) \geq Q_{\theta_n}(\theta_n)$, that is, any algorithm that goes uphill on Q_{θ_n}, not necessary finding the global maximum. They are introducing the same principle of inexact search in the M-step that we saw in inexact line search in descent algorithms. Conditions which guarantee the convergence of EM are covered in Section 3.6. As with Newton these can be difficult to verify, however, unlike Newton, EM doesn't typically enjoy superlinear convergence.

Theorem 3.4.6. *Suppose EM converges to a point θ^* in the interior of the parameter space that is a stationary point of the log likelihood, suppose it is possible to differentiate* (3.23) *twice, and $\nabla^2 l(\theta)$, $\nabla^2 Q_\theta(\theta)$, and $\nabla^2 H_\theta(\theta)$ are continuous in θ and have full rank at $\theta = \theta^*$, then the convergence cannot be superlinear.*

Proof. Suppose the EM sequence is $\theta_n \to \theta^*$. Differentiating (3.23) twice gives

$$\nabla^2 l(\theta) = \nabla^2 Q_\theta(\theta) - \nabla^2 H_\theta(\theta) \; .$$

The Newton step at θ for maximizing the log likelihood is

$$\Delta(\theta) = - \left(\nabla^2 l(\theta) \right)^{-1} \nabla l(\theta).$$

The M step at θ is determined by maximizing the function Q_θ. The Newton step at θ for maximizing Q_θ is

$$-\left(\nabla^2 Q_\theta(\theta)\right)^{-1} \nabla Q_\theta(\theta) \tag{3.24}$$

Since the conditions of Theorem 3.2.1 hold, Newton is superlinearly convergent in this subproblem and (3.24) is within little oh of the M step, that is, if we denote the M step at θ by $\delta(\theta)$ then

$$\delta(\theta) + o(\|\delta(\theta)\|) = -\left(\nabla^2 Q_\theta(\theta)\right)^{-1} \nabla Q_\theta(\theta)$$

So by the Dennis-Moré theorem EM can have superlinear convergence $\delta(\theta_n)$ and $\Delta(\theta_n)$ are asymptotically equivalent.

By Lemma 3.4.4 $\nabla Q_\theta(\theta) = \nabla l(\theta)$ so by assumption $\nabla Q_{\theta^*}(\theta^*) = \nabla l(\theta^*) = 0$. Define $u_n = \nabla Q_\theta(\theta_n) = \nabla l(\theta_n)$ and choose a convergent subsequence $u_{n_k} \to u$. Then we have superlinear convergence of EM only if the two limits

$$-\left(\nabla^2 l(\theta_{n_k})\right)^{-1} u_{n_k} \to -\left(\nabla^2 l(\theta^*)\right)^{-1} u$$

and

$$-\left(\nabla^2 Q_{\theta^*}(\theta^*)\right)^{-1} u$$

are the same, call the common limit v, which requires

$$\nabla^2 l(\theta^*) v = \nabla^2 Q_{\theta^*}(\theta^*) v$$

hence

$$\nabla^2 H_{\theta^*}(\theta^*) v = 0,$$

which violates the assumption that $\nabla^2 H_{\theta^*}(\theta^*)$ has full rank. □

3.5 Trust Regions

A method even better than line searches of forcing convergence is the method of trust regions. The idea is that each step should minimize the

quadratic model (3.4). But this model is only good in the neighborhood of the current iterate x_n, say for x in the set

$$\Omega_n = \{\, x : \|x - x_n\| \le h_n \,\}$$

for some constant $h_n > 0$, which we call a *trust region* because that is where we "trust" the quadratic model. Thus in step n we find the next iterate $x_{n+1} = x_n + \delta_n$ by solving the constrained problem

$$\begin{aligned} &\text{minimize } w_n(\delta) = \delta^T g(x_n) + \tfrac{1}{2}\delta^T H(x_n)\delta \\ &\text{subject to } \delta^T\delta \le h_n^2 \end{aligned} \tag{3.25}$$

The Lagrangian is

$$L(\delta) = \delta^T g(x_n) + \tfrac{1}{2}\delta^T H(x_n)\delta + \tfrac{1}{2}\lambda\delta^T\delta.$$

So the Kuhn-Tucker conditions are

- [minimization]

  $$\nabla L(\delta) = g(x_n) + [H(x_n) + \lambda I]\delta = 0$$

  or

  $$\delta = -\left(H(x_n) + \lambda I\right)^{-1} g(x_n) \tag{3.26}$$

- [primal feasibility] $\|\delta\| \le h_n$
- [dual feasibility] $\lambda \ge 0$
- [complementary slackness] either $\lambda = 0$ or $\|\delta\| = h_n$.

Taking the first choice in the complementary slackness condition, let $\lambda = 0$, then (3.26) becomes

$$\delta = -H(x_n)^{-1} g(x_n)$$

the Newton step. But this only satisfies the second order optimality condition in the trust region subproblem if $H(x_n)$ is positive definite and only keeps δ

in the trust region if $\delta^T\delta \leq h_n^2$. If either of these conditions are violated, we make the other choice in the complementary slackness condition, imposing the trust region constraint with equality. One way to solve this subproblem is to apply Newton's method to the system of nonlinear equations in δ and λ

$$\begin{aligned} g(x_n) + [H(x_n) + \lambda I]\delta &= 0 \\ \delta^T\delta &= h_n^2 \end{aligned}$$

A variety of interesting special methods have been proposed for finding λ here, which are discussed in Fletcher (1987, pp. 101–106). Particularly interesting is the Hebden-Moré scheme, for which see Fletcher. However λ is found, δ will then be given by (3.26) and this will minimize the Lagrangian if $H(x_n) + \lambda I$ is positive semi-definite. In fact this always happens. Consider another step u such that $u^Tu = h_n^2$. If we assume δ is the global minimizer of the constrained problem, then

$$\begin{aligned} 0 &\leq w_n(u) - w_n(\delta) \\ &= -g(x_n)^T(\delta - u) + \tfrac{1}{2}u^TH(x_n)u - \tfrac{1}{2}\delta^TH(x_n)\delta \end{aligned}$$

Using (3.26) to eliminate $g(x_n)$ and $u^Tu = \delta^T\delta = h_n^2$, and writing $H(x_n) = H$ gives

$$\begin{aligned} 0 &\leq \delta^T(H + \lambda I)(\delta - u) + \tfrac{1}{2}u^THu - \tfrac{1}{2}\delta^TH\delta \\ &= \tfrac{1}{2}(\delta - u)(H + \lambda I)(\delta - u). \end{aligned}$$

This implies that $H + \lambda I$ is at least positive semi-definite. For a formal statement of these facts see Theorem 5.2.1 in Fletcher (1987).

How do we choose the trust region radius? The idea is to choose it dynamically as the algorithm proceeds. If it seems that we have a satisfactory approximation, we leave the radius alone or increase it. If the approximation seems bad, we decrease it. What criterion do we use? Without doing any

extra work, all we know is the value of the quadratic approximation $w_n(\delta_n)$, the predicted decrease in the objective function, as well as the actual decrease $f(x_{n+1}) - f(x_n)$. We must evaluate f at x_{n+1} in order to do the next iteration, so the comparison costs nothing. Let

$$r_n = \frac{f(x_{n+1}) - f(x_n)}{w(\delta_n)}.$$

Then r_n is about 1 when the predicted and actual decrease are about the same, r_n is near zero when the actual decrease is much smaller than predicted, and r_n is negative if the step actually goes uphill rather than downhill. The actual decision points are rather arbitrary. Fletcher (1987, p. 96) suggests the following simple trust-region update.

1. Solve the constrained problem (3.25) finding x_{n+1}
2. Evaluate $f(x_{n+1})$ and r_n.
3. If $r_n > 0.75$ and the constraint $\delta_n^T \delta_n = h_n^2$ was binding, set $h_{n+1} = 2h_n$.
4. If $r_n < 0.25$, set $h_{n+1} = h_n/4$.
5. If $r_n \leq 0$, don't accept the step: set $x_{n+1} = x_n$.

The last part makes every step go downhill. If no downhill step can be found, the algorithm does not move.

For this algorithm, Fletcher (1987) proves the following theorem.

Theorem 3.5.1. *For the trust region algorithm above, if the sequence of iterates is contained in a compact set K, and if f is continuously twice differentiable on an open set containing K, then there exists a cluster point of the sequence of iterates that satisfies the first and second order necessary conditions for a local minimum.*

Example 3.5.1. *The Rosenbrock function is*

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \, .$$

it is straightforward to show that $x^* = (1, 1)^T$ *is the only local minimizer of this function. This can be verified with the R library* `trust` *(written by Charlie Geyer) that can be found from the link on the class homepage. In the following R code note the use of* `D` *function from* `deriv` *to calculate symbolic derivatives.*

The usage

```
trust(objfun, parinit, rinit, rmax, parscale,
    iterlim = 100, fterm = sqrt(.Machine$double.eps),
    mterm = sqrt(.Machine$double.eps),
    minimize = TRUE, blather = FALSE, ...)
```

The input

```
library(trust)

##### Rosenbrock's function #####
objfun <- function(x) {
    stopifnot(is.numeric(x))
    stopifnot(length(x) == 2)
    f <- expression(100 * (x2 - x1^2)^2 + (1 - x1)^2)
    g1 <- D(f, "x1")
    g2 <- D(f, "x2")
    h11 <- D(g1, "x1")
    h12 <- D(g1, "x2")
    h22 <- D(g2, "x2")
    x1 <- x[1]
    x2 <- x[2]
```

```
    f <- eval(f)
    g <- c(eval(g1), eval(g2))
    B <- rbind(c(eval(h11), eval(h12)), c(eval(h12), eval(h22)))
    list(value = f, gradient = g, hessian = B)
}

trust(objfun, c(3, 1), 1, 5)
```

and the output

```
trust(objfun, c(3, 1), 1, 5)
$value
[1] 5.160801e-15

$gradient
[1]  5.430221e-07 -2.003744e-07

$hessian
          [,1] [,2]
[1,]  802.0001 -400
[2,] -400.0000  200

$argument
[1] 1.000000 1.000000

$converged
[1] TRUE

$iterations
[1] 21
```

3.6 Appendix: Convergence of EM

A GEM algorithm has a set $M(\theta)$ of points among which it chooses θ_{n+1}. Thus M is a set-valued mapping, it maps points θ in the parameter space Θ to subsets $M(\theta) \subset \Theta$. A notation for this is $M : \Theta \rightrightarrows \Theta$. The requirement that satisfied by a GEM algorithm as defined by Dempster et al. (1977) is

$$Q_\varphi(\theta) \geq Q_\varphi(\varphi), \qquad \forall \theta \in M(\varphi) \tag{3.27}$$

But (3.27) is too weak to be used in proving anything. It is, for example, satisfied by any constant sequence. At the very least an algorithm must make progress if it can, that is

$$Q_\varphi(\theta) > Q_\varphi(\varphi), \qquad \forall \theta \in M(\varphi) \tag{3.28}$$

whenever there exists a $\theta' \in \Theta$ such that $Q_\varphi(\theta') > Q_\varphi(\varphi)$. But in order to prove anything we need much stronger regularity conditions. The ones given here follow Wu (1983).

The map M is *outer semicontinuous* (OSC) if its graph

$$\operatorname{grph} M = \{\, (\varphi, \theta) \in \Theta \times \Theta : \theta \in M(\varphi) \,\}$$

is a closed set, that is, if $(\varphi_n, \theta_n) \to (\varphi^*, \theta^*)$ and $\theta_n \in M(\varphi_n)$, then $\theta^* \in M(\varphi^*)$. Suppose we are attempting to prove that every cluster point of the GEM sequence induced by the set-valued mapping M lies in a certain set $\Gamma \subset \Theta$, called the "solution set." Γ might be the set of stationary points of the log likelihood or the set of local maxima of the log likelihood.

Let Ω be a subset of Θ having the property that $M(\varphi) \subset \Omega$ whenever $\varphi \in \Omega$ and that $\Gamma \subset \Omega$. The point of introducing Ω is to deal with solutions on the boundary of the parameter space (if any). It is often true that the EM sequence stays in the interior of Θ if started in interior. If the solution set Γ is contained in the interior, then we may take Ω to be the interior of Θ. On the other hand, if the solution set contains points on the boundary, in

particular if the actual maximum of the likelihood is on the boundary, then we take $\Omega = \Theta$.

Theorem 3.6.1. *Let $M : \Omega \rightrightarrows \Omega$ be the set-valued mapping for an GEM algorithm and $\Gamma \subset \Omega$. Suppose the following conditions.*

(a) *The restriction of M to $\Omega \setminus \Gamma$ is outer semicontinuous.*

(b) *If $\varphi \in \Omega \setminus \Gamma$, then* (3.28) *holds.*

(c) *If $\varphi \in \Gamma$, then* (3.27) *holds.*

(d) *The log likelihood is continuous on Ω and the level set*

$$\{\, \theta \in \Omega : l(\theta) \geq l(\theta_1) \,\}$$

is compact.

Then $l(\theta_n)$ converges to a limit λ, and every cluster point of $\{\theta_n\}$ is contained in Γ.

Proof. The assumptions in (d) imply the log likelihood is bounded above. Hence, since $l(\theta_n)$ is nondecreasing by (3.27), (3.28), and Theorem 3.4.3, it converges to a limit λ. Suppose to get a contradiction that $\theta_{n_k} \to \theta^*$ but $\theta^* \notin \Gamma$. Then $l(\theta^*) = \lambda$, by continuity of l. By the compactness assumption in (d), the sequence θ_{n_k+1} has a convergent subsequence $\theta_{n_{k_l}+1} \to \theta^{**}$. Then $\theta^{**} \in M(\theta^*)$ by outer semicontinuity of M, and $l(\theta^{**}) = \lambda$ by continuity of l. But this contradicts (3.28). □

Corollary 3.6.2. *If the conditions of the theorem hold and the set Γ consists of a single point $\hat{\theta}$, then the EM sequence θ_n converges to $\hat{\theta}$.*

Chapter 4

Integration

4.1 Applied Measure Theory

If X is a random element of some probability space having measure P, then we write

$$E\{g(X)\} = \int g(x)P(dx) \tag{4.1}$$

whenever g is a real-valued function such that the expectation exists. From an applied point of view, we can regard (4.1) as a convenient shorthand. Whatever is meant by the left hand side is whatever is meant by the right hand side.

If X is a discrete random variable taking values in a set S with probability mass function f, then (4.1) means

$$E\{g(X)\} = \sum_{x \in S} g(x)f(x).$$

If X is a continuous random variable with probability density function f, then (4.1) means

$$E\{g(X)\} = \int_{-\infty}^{+\infty} g(x)f(x)\,dx.$$

If X is a continuous random vector taking values in $\mathbb{R}^3$ with probability density function f, then (4.1) means

$$E\{g(X)\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x_1, x_2, x_3) f(x_1, x_2, x_3)\, dx_1\, dx_2\, dx_3.$$

If X is a random variable that is neither discrete nor continuous, for example, X is an exponential random variable with rate parameter λ right censored at T (meaning we observe the exponential random variable or T, whichever is smaller), then (4.1) means

$$E\{g(X)\} = \int_0^T g(x)\lambda e^{-\lambda x}\, dx + g(T)e^{-\lambda T}.$$

So the integral in (4.1) doesn't necessarily mean integration in the sense of calculus. It may mean summation or a combination of integration and summation. And even if it does mean integration in the sense of calculus, it may mean a double, triple, or higher integral.

So measure-theoretic notation is valuable even in applied situations because it allows us to cover all the special cases with one notation. But isn't the left hand side of (4.1) and all those other equations good enough as a common notation? Not really, because it is too vague. The right hand side of (4.1) clearly indicates the measure P and clearly indicates that what expectation means depends (through P) on the probability model in question. The left hand side doesn't. However, if we write $E_P\{g(X)\}$ it should be clear. This clarity will become more apparent as we go along.

We shall not need the abstract measure-theoretic definition of measures like P. It will be enough to know that when outside an integral, a measure is a set function, a map from subsets A of the state space to probabilities $P(A)$. It is a deep theorem that such functions determine abstract integrals like the right hand side of (4.1). But to apply measure theory, one only needs to know that *probability is a special case of expectation*

$$P(A) = E_P\{I_A(X)\} = \int_A P(dx), \tag{4.2}$$

where I_A denotes the *indicator function* of the set A,

$$I_A(x) = \begin{cases} 1, & x \in A \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

4.2 Intractable Integrals

Consider an integral that is an expectation, say

$$E\{g(X)\} = \int g(x)P(dx), \tag{4.4}$$

where X is a random variable with probability measure P. We assume this expectation actually exists. For convenience we denote it by a Greek letter

$$\mu = \int g(x)P(dx). \tag{4.5}$$

Most integrals are impossible to calculate exactly. Hence the best we can do is an approximation.

Example 4.2.1. *Consider a simple version of the so-called logit-normal model. For $i = 1, \ldots, n$ and $j = 1, \ldots q$, assume $y_{ij}|u_i \overset{ind}{\sim}$ Bernoulli(π_{ij}) where π_{ij} satisfies*

$$logit(\pi_{ij}) = \beta x_{ij} + u_i$$

and β is unknown while x_{ij} is a known covariate. Finally, assume that the u_i are iid $N(0, \sigma_u^2)$. The likelihood is clearly analytically intractable:

$$L(\beta, \sigma_u^2; y) \propto \frac{1}{(\sigma_u^2)^{q/2}} \int \prod_{i,j} \frac{\exp\{y_{ij}(\beta + u_i)\}}{1 + \exp\{y_{ij}(\beta + u_i)\}} \times \exp\left\{ -\frac{1}{2\sigma_u^2} \sum_{i=1}^{n} u_i^2 \right\} du \ .$$

A computer algebra system like Mathematica or Maple can probably do more integrals than any one person, but most integrals are provably analytically intractable. When symbolic expression does not exist, no computer program can find it.

A computer algebra system like Mathematica or Maple or the `integrate` function in the base package of R can do numerical integration. This works for many low dimensional integrals. Unfortunately, it does not work well for integrals of even moderate dimension. Five is really pushing it. It is also difficult to assess the error in the approximation of μ when using numerical integration. Lets take a brief look at how basic quadrature methods work.

4.3 Numerical Integration

Suppose $f : \mathbb{R} \to \mathbb{R}$ and we want the value of a linear functional

$$I(f) = \int_a^b f(x)\, dx \qquad -\infty \leq a \leq b \leq \infty\,. \tag{4.6}$$

Our goal is to create an approximation to $I(f)$. Since $I(f)$ is linear it makes sense to think about approximations of the form

$$\sum_{i=0}^{n} a_i f(x_i)$$

where the a_i and x_i are to be determined by some rule. This technique is known as *numerical quadrature*.

Editorial note: Much of the material in this section is based on the presentations in Ralston and Rabinowitz (2001) and Burden and Faires (2005).

4.3.1 Lagrangian Interpolation

The first step in approximating $I(f)$ in (4.6) is to approximate the integrand. Here we consider one method (but not necessarily the best method) for doing this. Suppose $x_0, x_1, \ldots, x_n \in \mathbb{R}$ are $n+1$ distinct points. Let $f : \mathbb{R} \to \mathbb{R}$ be a function whose values are given at these points. Define

$$L_{n,k}(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

and set

$$p(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}(x) \ . \tag{4.7}$$

Then $p(x)$ is a unique polynomial of degree at most n such that for each $k = 0, 1, \ldots, n$

$$f(x_k) = p(x_k) \ .$$

And p is the *nth Lagrange interpolating polynomial.*

Example 4.3.1. *Let $f(x) = 1/x$ for $x > 0$. The nodes are $x_0 = 2, x_1 = 3, x_2 = 5$. We will find the second Lagrange interpolating polynomial. Now*

$$\begin{aligned} L_{2,0}(x) &= \frac{1}{3}(x-3)(x-5) \\ L_{2,1}(x) &= \frac{-1}{2}(x-2)(x-5) \\ L_{2,2}(x) &= \frac{1}{6}(x-2)(x-3) \end{aligned}$$

Then the second Lagrange interpolating polynomial based on these nodes is

$$p(x) = \frac{1}{6}(x-3)(x-5) - \frac{1}{6}(x-2)(x-5) + \frac{1}{30}(x-2)(x-3) \ .$$

This approximation is plotted with the target function in the left hand panel of Figure 4.1. Using different nodes will result in a different approximation, at least over some interval. For example, choosing $x_0 = 2, x_1 = 2.5, x_2 = 4$ yields the following second Lagrange interpolating polynomial

$$p(x) = (.05x - 0.425)x + 1.15 \ .$$

This approximation is plotted with the target function in the right hand panel of Figure 4.1.

An obvious question of interest is quantifying the error in approximating the the target function with an interpolating polynomial.

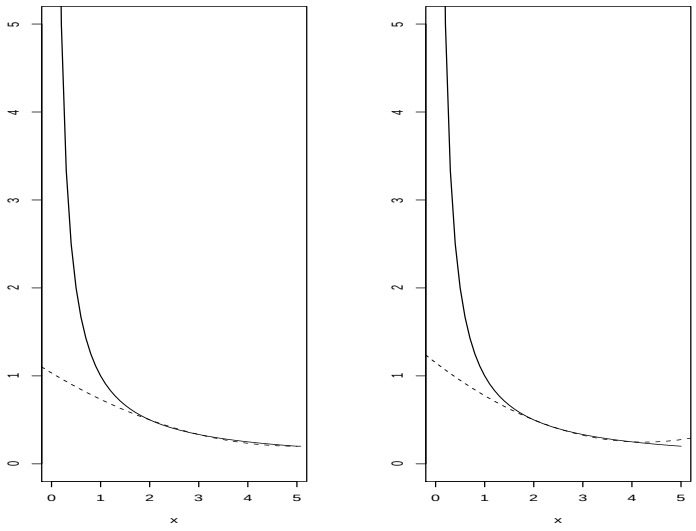

Figure 4.1: Interpolating polynomial approximation of $f(x) = 1/x$. The solid curve in each panel is $f(x) = 1/x$ while the dashed curves are the approximations.

Theorem 4.3.1. *Suppose $x_0, x_1, \ldots, x_n \in [a, b]$ are $n+1$ distinct points and f is $n+1$ times continuously differentiable on $[a, b]$. Then for each $x \in [a, b]$ there exists $\xi(x) \in (a, b)$ such that*

$$f(x) = p(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

where p is defined in (4.7).

4.3.2 Quadrature

Given a Lagrange interpolating polynomial we can approximate the integrand in $I(f)$. To approximate $I(f)$ itself we can integrate the approximation to the integrand and its error term from Theorem 4.3.1

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_a^b p(x)\,dx + \int_a^b \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i)\,dx \\ &= \sum_{i=0}^{n} a_i f(x_i) + \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi(x)) \prod_{i=0}^{n} (x - x_i)\,dx\end{aligned}$$

where, for $i = 0, 1, \ldots, n$

$$a_i = \int_a^b L_{n,i}(x)\,dx\ .$$

Hence the quadrature approximation is

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} a_i f(x_i)$$

with error

$$E(f) = \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi(x)) \prod_{i=0}^{n} (x - x_i)\,dx\ .$$

Note that E is difficult to calculate but, due to the factor $1/(n+1)!$, will decrease rapidly as the number of nodes n increases. What we need are

methods for choosing the number and location of the nodes. The $(n+1)$-*point Closed Newton-Cotes* method chooses the nodes according to

$$x_i = x_0 + i\frac{b-a}{n} \qquad \text{for } i = 1, \ldots, n$$

with $x_0 = a$. Setting $n = 1$ yields the familiar Trapezoidal rule while $n = 2$ correspond to Simpson's rule and $n = 3$ is Simpson's Three-Eighths rule. Newton-Cotes usually works well when $b-a$ is small. Thus composite or extended Newton-Cotes rules divide the interval $[a, b]$ into multiple, nonoverlapping subintervals $[a_i, b_i]$, and then apply a Newton-Cotes rule to each subinterval.

Newton-Cotes forces the nodes to be equally spaced. However, there is no reason that this should be optimal. *Gaussian quadrature* attempts to choose the nodes in a more optimal fashion. We consider only one of the most basic forms which is often called *Gauss-Legendre quadrature*. To begin note that setting $x = c_1 + c_2 t$ where $c_1 = 0.5(b+a)$ and $c_2 = 0.5(b-a)$ gives

$$\int_a^b f(x)\,dx = c_2 \int_{-1}^{1} f(c_1 + c_2 t)\,dt\ .$$

and the quadrature approximation to $I(f)$ is then

$$\int_a^b f(x)\,dx \approx c_2 \sum_{i=1}^{n} w_i f(c_1 + c_2 t_i)\ .$$

We still need to determine the weights and the nodes. The nodes are the roots of a Legendre polynomial. The nth degree Legendre polynomial is given by

$$p_n(t) = \frac{1}{2^n n!}\frac{d^n}{dt^n}(t^2-1)^n\ .$$

Then the weights are given by

$$w_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}} \frac{t - t_j}{t_i - t_j}\,dt\ . \tag{4.8}$$

The following theorem shows that these are sensible choices for a quadrature rule.

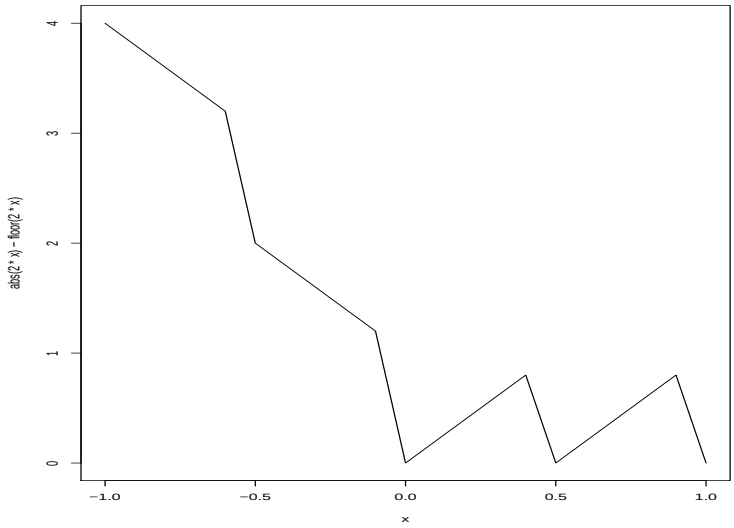

Figure 4.2: A spiky function.

Theorem 4.3.2. *Suppose that $x_1, x_2, \ldots, x_n$ are the roots of the nth degree Legendre polynomial and that the weights w_i are given by* (4.8)*. If $p(x)$ is any polynomial of degree less than $2n$, then*

$$\int_{-1}^{1} p(x)\,dx = \sum_{i=1}^{n} w_i\, p(x_i)\ .$$

Fortunately, there is no reason to actually calculate the weights and nodes "by hand" as both of these have been extensively tabulated; see Abramowitz and Stegun (1972). It is also the case that the `integrate` function in R does something called adaptive quadrature which attempts to distribute the approximation error evenly by using unequally spaced nodes.

Example 4.3.2. *Let $f(x) = |2x| - \lfloor 2x \rfloor$ (see Figure 4.2) and suppose we want to find*

$$\int_{-1}^{1} f(x)\,dx.$$

This is dead easy with R's `integrate` *function.*

```
> integrand<-function(x){abs(2*x) - floor(abs(2*x))}
> integrate(integrand, lower=-1, upper=1)
1 with absolute error < 1.1e-14
```

`integrate` *can also handle integration over an infinite interval. Let* $f(x) = 1/((x+3)^{1.5}\sqrt{x})$ *and we will find its integral over* $(0, \infty)$.

```
> integrand<-function(x){1/(sqrt(x)*(x+3)^1.5)}
> integrate(integrand, lower=0, upper=Inf)
0.6666667 with absolute error < 5.1e-06
```

4.4 Monte Carlo Integration

Consider an integral that is an expectation, say

$$E_P\{g(X)\} = \int g(x)P(dx), \tag{4.9}$$

where X is a random variable with probability measure P. We assume this expectation actually exists. For convenience we denote it by a Greek letter

$$\mu = \int g(x)P(dx). \tag{4.10}$$

Most integrals of interest in probability theory are impossible to calculate exactly. Hence the best we can do is an approximation. That's why normal approximations and large-sample theory are so widely used.

The only generally applicable tool for approximating integrals like (4.10) is so-called *Monte Carlo integration*. Suppose we can simulate an iid sequence X_1, X_2, ... of random variables having the probability measure P. Then

$$Y_i = g(X_i), \qquad i = 1, 2, \ldots$$

is an iid sequence of random variables having mean μ, which is the integral (4.10) we want to evaluate.

The strong law of large numbers (SLLN) says that if $E|Y| < \infty$ then with probability 1 as $n \to \infty$

$$\bar{Y}_n := \frac{1}{n}\sum_{i=1}^{n} Y_i \to \mu. \tag{4.11}$$

In words, we can approximate μ with an arbitrary level of precision if we only average over a sufficiently large number of simulations. And using $\bar{Y}_n$ as an approximation for μ is the "Monte Carlo method." The following toy example will illustrate this procedure.

Example 4.4.1 (Gamma)**.** *Suppose* $X \sim$ *Gamma*$(3/2, 1)$ *and we want to calculate*

$$\mu = E\left[\frac{1}{(X+1)\log(X+3)}\right].$$

Then if $X_1, X_2, \ldots, X_n$ *are iid copies of* X *and* $g(x) = [(X+1)\log(X+3)]^{-1}$ *an estimate of* μ *is given by*

$$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{(x_i+1)\log(x_i+3)}.$$

The following R code implements this estimation procedure.

```
> nsim<-1e4
> x<-rgamma(nsim,3/2,1)
> g.hat<-1/((x+1)*log(x+3))
> mu.hat<-mean(g.hat)
> mu.hat
mu.hat
[1] 0.3561899
```

Note that `nsim` *is the Monte Carlo sample size. An important point about Monte Carlo methods is that different runs will give different estimates.*

```
> nsim<-1e4
> x<-rgamma(nsim,3/2,1)
> g.hat<-1/((x+1)*log(x+3))
> mu.hat<-mean(g.hat)
> mu.hat
mu.hat
[1] 0.3573980
```

If the simulation size n is sufficiently large the estimates shouldn't differ by much.

Trivial, is it not? Just a cutesy name for something every statistician already knows and uses every day. But don't let the triviality bother you. It's great! That means you are already completely comfortable with most of the theory of the Monte Carlo method. It's just statistics (actually *large sample, frequentist* statistics).

Despite its simplicity and familiarity to all statisticians, Monte Carlo can be a bit confusing because there are two sorts of samples, sample sizes, sources of stochastic variability. Throughout these notes we will use m to denote the observed data sample size while n will be reserved for the size of the simulation. The following example will make this clear.

Example 4.4.2 (Trimmed Mean)**.** *Suppose X_1, X_2, ..., X_m are a random sample from a standard normal distribution. What is the relative efficiency of the sample mean compared to a 25% trimmed mean $\hat{\theta}$ as an estimator of the true unknown population mean θ?*

The following R code does the simulation

```
mdat <- 30
nsim <- 1e4
theta.hat <- double(nsim)
for (i in 1:nsim) {
```

```
    x <- rnorm(mdat)
    theta.hat[i] <- mean(x, trim = 0.25)
}
```

Note that in the code **`mdat`** *is the data sample size and* **`nsim`** *is the Monte Carlo sample size.*

We want to evaluate $\mathrm{Var}(\hat{\theta})$*. Since a variance is an expectation it can be written in the form* (4.10) *for some function* g*, but the formula would be very messy. A trimmed mean is not a function defined by a nice simple expression. But that doesn't matter in Monte Carlo. If a computer can evaluate the function, no problem.*

The relative efficiency is $\mathrm{Var}(\hat{\theta})$ *divided by the variance of the sample mean, which we know to be* $1/m$ *without doing any Monte Carlo. Thus our Monte Carlo approximation to the relative efficiency is computed by*

```
mdat * mean(theta.hat^2)
```

This formula, as opposed to **`mdat * var(theta.hat)`***, is used because by the symmetry of the normal distribution,* $E(\hat{\theta}) = 0$*.*

A run of this code in R gives 1.178922 for the Monte Carlo approximation to the relative efficiency. Of course, the relative efficiency is a function of m*, so we would have to redo the calculation for any data sample size* m *we were interested in.*

It is very important to keep in mind that a Monte Carlo approximation is not exact. The number 1.178922 calculated in the example is not exact value of the integral we are trying to approximate using the Monte Carlo method. It is off by some amount, which we call *Monte Carlo error*. How large is the Monte Carlo error? Just as everywhere else in statistics, we can never know. The error is $1.178922 - \mu$. Hence we don't know its value unless we know μ, and if we knew that we wouldn't be doing Monte Carlo in the first place.

We know that our Monte Carlo approximation, $\bar{Y}_n$, is the average of some random variables Y_1, Y_2, ... forming an IID sequence. If $E(Y_i^2) < \infty$, then the central limit theorem says

$$\bar{Y}_n \approx \text{Normal}\left(\mu, \frac{\sigma^2}{n}\right) \tag{4.12}$$

where $\text{var}(Y_i) = \sigma^2$. Generally (4.12) tells us all we can know about the Monte Carlo error. Of course, this is no better and no worse than our general knowledge about sampling variability everywhere in statistics. We don't know the error, but do know its sampling distribution and must be satisfied with that.

Also, as elsewhere in statistics, we don't know the variance σ^2 and must estimate it from the samples by

$$S_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \bar{Y}_n)^2.$$

One can produce a confidence interval for the true unknown value of μ, but it often suffices to just report the Monte Carlo standard error (MCSE), $S_n/\sqrt{n}$. We usually only want a rough idea of how accurate our Monte Carlo calculation is. Does it have two significant figures, three significant figures, no significant figures, or what? The only way to know is if the MCSE is calculated and reported.

Example 4.4.3 (Gamma, MCSE)**.** *In this example it is easy to calculate the MCSE for each of the two runs. The first run first ($\hat{\mu} = 0.3561899$)*

```
> sd(g.hat)/sqrt(nsim)
[1] 0.001941493
```

and the second run second ($\hat{\mu} = 0.3573980$)

```
> sd(g.hat)/sqrt(nsim)
[1] 0.001950561
```

As everywhere else in statistics, there is no need to keep a lot of inaccurate significant figures once we figure out what the accuracy actually is. For example, in the first run we would report the estimate as 0.356 *with MCSE* 0.002 *while for the second run it would be* 0.357 *with MCSE* 0.002*.*

Despite its simplicity and familiarity to all statisticians, MCSE can be confusing when there are several variances floating around. The variance involved in the MCSE needn't be, and usually isn't, the variance involved in the expectation μ being calculated. Again, the distinction must be kept crystal clear.

Example 4.4.4 (Trimmed Mean, MCSE)**.** *In the trimmed mean example, the expectation being calculated is a constant times a variance,* $\mu = m \operatorname{Var}(\hat{\theta})$*. We estimated it by*

$$\hat{\mu}_n = \frac{m}{n} \sum_{i=1}^{n} \hat{\theta}_i^2$$

where $\hat{\theta}_1$*,* $\hat{\theta}_2$*, ... are the Monte Carlo samples. The things being averaged to calculate* μ *are the* $m\hat{\theta}_i^2$*, thus* S_n^2 *should be the sample variance of the* $m\hat{\theta}_i^2$*.*

With the preceding discussion, is should now be clear that the MCSE is

```
sqrt(var(mdat * theta.hat^2) / nsim)
```

which turned out to be 0.01671859.

As in the Gamma example, there is no need to keep a lot significant figures. We can now report our result as 1.179 with MCSE 0.017 or if we prefer even fewer figures as 1.18 with MCSE 0.02.

Note that both sample sizes `mdat` and `nsim` appeared in our MCSE calculation, and also that the variance `var(mdat * theta.hat^2)` that appeared in the MCSE is very different from the variance in `mdat * var(theta.hat)` that might have been used as our Monte Carlo estimate.

Monte Carlo is very simple in theory. Its theory is just frequentist large-sample theory (consistency, asymptotic normality, etc.) that we are all familiar with. In practice, one must keep very clear what's what in order not to get confused.

So far we haven't covered anything about how to generate a random sample for estimating $E_P[g(X)]$. This is addressed in the next section.

4.5 Generating a Random Sample

The methods presented in this section assume that we can use the computer to generate $U \sim \text{Uniform}(0, 1)$ which is easy in R; just use `runif`. Technically, what the computer generates is *not* a random number from a uniform distribution but rather a pseudo-random number that is generated by a deterministic algorithm and hence isn't random at all. For this reason the following remark by John von Neumann is often quoted.

> Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

However, an awful lot of research (some of it ongoing) has gone into showing that there are algorithms that produce such pseudo-random numbers that behave as if they were in fact randomly drawn from a uniform distribution. There are some advanced computing situations where it is important to keep in mind the deterministic and periodic nature of pseudo-random number generators but we should not encounter these in our course.

4.5.1 Inversion

Theorem 4.5.1 (Probability Integral Transform)**.** *Suppose X has a continuous, strictly increasing cumulative distribution function $F_X(x)$. If $U \sim$ Uniform$(0, 1)$ then $F_X^{-1}(u) \sim F_X$.*

Proof.

$$\Pr(F_X^{-1}(U) \le t) = \Pr(F_X(F_X^{-1}(U)) \le F_X(t)) = \Pr(U \le F_X(t)) = F_X(t) \ .$$

□

Example 4.5.1. *Our goal is to generate an observation from a Gamma(α, β) distribution where $\alpha \in \mathbb{Z}_+$. Recall that if $X_1, X_2, \ldots, X_\alpha$ are iid Exponential(β) then*

$$\sum_{i=1}^{\alpha} X_i \ \sim \ Gamma(\alpha, \beta) \ .$$

Let $X \sim$ Exponential(β). Then

$$f_X(x) = \frac{1}{\beta} e^{-x/\beta} \qquad \text{and} \qquad F_X(x) = 1 - e^{-x/\beta} \ .$$

Also, $y = 1 - e^{-x/\beta}$ if and only if $x = -\beta \log(1-y)$. Thus $F_X^{-1}(y) = -\beta \log(1-y)$. So if $U \sim$ Uniform$(0,1)$ then $F_X^{-1}(u) = -\beta \log(1-u) \sim$ Exponential(β). It follows now that if $U_1, U_2, \ldots, U_\alpha$ are iid Uniform$(0,1)$ then

$$\sum_{i=1}^{\alpha} -\beta \log(1-u_i) \ \sim \ Gamma(\alpha, \beta) \ .$$

4.5.2 Accept-Reject

This is an indirect method of simulation; we use draws from a density $f_Y(\cdot)$ to get draws from a density $f_X(\cdot)$. That is, we sample from the wrong distribution and correct it.

Theorem 4.5.2. *Let $X \sim f_X$ and $Y \sim f_Y$ where the support of f_Y contains the support of f_X. Define*

$$M := \sup_x \frac{f_X(x)}{f_Y(x)} \ .$$

If $M < \infty$ on the support of f_X then we can generate $X \sim f_X$ as follows:

1. *Generate* $Y \sim f_Y$ *and independently* $U \sim$ *uniform*$(0,1)$.

2. *If*
$$u < \frac{f_X(y)}{M\, f_Y(y)}$$
set $X = Y$*; otherwise discard* Y *and return to the first step.*

Proof.

$$\Pr(X \le x) = \Pr(Y \le x \,|\, X = Y) = \frac{\Pr(Y \le x,\ U \le f_X(y)/Mf_Y(y))}{\Pr(U \le f_X(y)/Mf_Y(y))} \tag{4.13}$$

Now

$$\begin{aligned}
\Pr(Y \le x,\ U \le f_X(y)/Mf_Y(y)) &= E[\Pr(Y \le x,\ U \le f_X(y)/Mf_Y(y) \,|\, Y)] \\
&= E\left[I(y \le x)\frac{f_X(y)}{M\, f_Y(y)}\right] \\
&= \frac{1}{M}\int I(y \le x)\frac{f_X(y)}{f_Y(y)}\, f_Y(y)\, dy \\
&= \frac{1}{M}\int I(y \le x) f_X(y)\, dy
\end{aligned}$$

and

$$\begin{aligned}
\Pr(U \le f_X(y)/Mf_Y(y)) &= E[\Pr(U \le f_X(y)/Mf_Y(y) \,|\, Y)] \\
&= E[f_X(y)/f_Y(y)] \\
&= \frac{1}{M}\int \frac{f_X(y)}{f_Y(y)}\, dy \\
&= \frac{1}{M}\,.
\end{aligned}$$

From (4.13) we have

$$\Pr(X \le x) = \int I(y \le x) f_X(y)\, dy = F_X(x)\,.$$

□

Note that $\Pr(U \leq f_X(y)/Mf_Y(y)) = 1/M$. The number of iterations until the algorithm produces a single draw from F_X is Geometric$(1/M)$ and hence the expected number of iterations until success is M. Lets return to Examples 4.4.1 and 4.5.1.

Example 4.5.2. *Recall from Example 4.4.1 that* $X \sim$ *Gamma*$(3/2, 1)$ *and we want to calculate*

$$\mu = E\left[\frac{1}{(X+1)\log(X+3)}\right] .$$

We've already shown that if $X_1, X_2, \ldots, X_n$ *are iid copies of* X *and* $g(x) = [(X+1)\log(X+3)]^{-1}$ *an estimate of* μ *is given by*

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{(x_i+1)\log(x_i+3)} .$$

In Example 4.5.1 we showed how to generate $X_1, X_2, \ldots, X_n$ *from a Gamma*(α, β) *when* α *is a positive integer. Now we can use the Accept-Reject algorithm to generate from a general Gamma distribution. It seems natural to use a Gamma* $(z, 1)$ *where* $z \in \{1, 2, 3, \ldots\}$*, candidate but if we do this then*

$$M = \sup_{x>0}\frac{f_X(x)}{f_Y(x)} = \frac{2(z-1)!}{\sqrt{\pi}}\sup_{x>0} x^{-z+3/2} = \infty$$

and hence we cannot apply the Accept-Reject algorithm. In a homework exercise you will address a solution this problem.

It is possible to implement Accept-Reject sampling without knowing the value of M explicitly; see Caffo et al. (2002). Also, when the target density is log-concave the Adaptive Rejection sampling method of Gilks (1992) and Gilks and Wild (1992) can work nicely.

4.6 Problems with Ordinary Monte Carlo

The main problem with ordinary Monte Carlo is that it is very hard to do for multivariate stochastic processes. A huge number of methods exist

for simulating univariate random variables. Devroye (1986) is the definitive source. Ripley (1987) is more introductory but is authoritative as far as it goes.

There are a few tricks for reducing multivariate problems to univariate problems. A general multivariate normal random vector $X \sim N(\mu, \Sigma)$ can be simulated using the Cholesky decomposition of the dispersion matrix $\Sigma = LL^T$. Let Z be a $N(0, I)$ random vector (each component is standard normal and the components are independent). Then $X = \mu + LZ$ has the desired $N(\mu, \Sigma)$ distribution (Ripley, 1987, p. 98). Wishart distributions can also be simulated (Ripley, 1987, p. 99–100). There are a few other special cases in which independent simulations of a multivariate process are possible, but not many.

One general method that has occurred to many people is to use the laws of conditional probability. Simulate the first component X_1 from its marginal distribution, simulate the second component X_2 from its conditional distribution given X_1, then simulate X_3 from its conditional distribution given X_1 and X_2, and so forth. Unfortunately, this technique is not that useful in general because the required marginal and conditional distributions are typically unknown and cannot be used for simulation.

Example 4.6.1. *Consider the following conditionally independent hierarchical model. Suppose for $i = 1, \ldots, K$ that*

$$\begin{aligned} Y_i|\theta_i &\sim N(\theta_i, a) & \theta_i|\mu, \lambda &\sim N(\mu, \lambda) \\ \lambda &\sim IG(b, c) & f(\mu) &\propto 1 \; . \end{aligned} \tag{4.14}$$

where a, b, c are all known positive constants. (Note that in this example we say $W \sim Gamma(\alpha, \beta)$ if its density is proportional to $w^{\alpha-1}e^{-\beta w}I(w > 0)$ and and $W^{-1} \sim IG(\alpha, \beta)$.)

Let $\pi(\theta, \mu, \lambda|y)$ be the posterior distribution corresponding to the hierarchy in (4.14)*. Note that θ is a vector containing all of the θ_i and that y is a vector*

containing all of the data. Consider the factorization

$$\pi(\theta, \mu, \lambda|y) = \pi(\theta|\mu, \lambda, y)\pi(\mu|\lambda, y)\pi(\lambda|y). \tag{4.15}$$

If it is possible to sequentially simulate from each of the densities on the right-hand side of (4.15) *we can produce iid draws from the posterior. Now* $\pi(\theta|\mu, \lambda, y)$ *is the product of independent univariate normal densities, i.e.* $\theta_i|\mu, \lambda, y \sim N((\lambda y_i + a\mu)/(\lambda + a), a\lambda/(\lambda + a))$. *Also,* $\pi(\mu|\lambda, y)$ *is a normal distribution, i.e.* $\mu|\lambda, y \sim N(\bar{y}, (\lambda + a)/K)$. *Next*

$$\pi(\lambda|y) \propto \frac{1}{\lambda^{b+1}(\lambda + a)^{(K-1)/2}} e^{-c/\lambda - s^2/2(\lambda+a)}$$

where $\bar{y} = K^{-1}\sum_{i=1}^{K} y_i$ *and* $s^2 = \sum_{i=1}^{K}(y_i - \bar{y})^2$. *An accept-reject algorithm with an* $IG(b, c)$ *candidate can be used to sample from* $\pi(\lambda|y)$ *since if we let* $g(\lambda)$ *be the kernel of an* $IG(b, c)$ *density*

$$\sup_{\lambda \geq 0} \frac{1}{g(\lambda)\lambda^{b+1}(\lambda + a)^{(K-1)/2}} e^{-c/\lambda - s^2/2(\lambda+a)} = \sup_{\lambda \geq 0} (\lambda+a)^{(1-K)/2} e^{-s^2/2(\lambda+a)} = M < \infty$$

It is easy to show that the only critical point is $\hat{\lambda} = s^2/(K-1) - a$ *which is where the maximum occurs if* $\hat{\lambda} > 0$. *But if* $\hat{\lambda} \leq 0$ *then the maximum occurs at 0.*

4.7 Importance Sampling

4.7.1 Densities (More Applied Measure Theory)

We say a probability measure P has a *density* f with respect to another probability measure Q if (4.9) can be rewritten

$$\begin{aligned} E\{g(X)\} &= \int g(x)P(dx) \\ &= \int g(x)f(x)Q(dx) \end{aligned} \tag{4.16}$$

and this holds for all functions g for which the expectation is defined.

For example, consider the case where both P and Q are the probability measures of continuous real-valued random variables. Say X has measure P and Y has measure Q and

$$E\{g(X)\} = \int_{-\infty}^{+\infty} g(x) f_X(x)\, dx = \int g(x) P(dx)$$

for any function g for which the expectation is defined and

$$E\{g(Y)\} = \int_{-\infty}^{+\infty} g(y) f_Y(y)\, dy = \int g(x) Q(dy)$$

for any function g for which the expectation is defined, then in (4.16) we have

$$f(w) = \frac{f_X(w)}{f_Y(w)} \tag{4.17}$$

so long as there is no problem with division by zero (the reader should check that this definition does indeed do the job). When both the numerator and denominator are zero in (4.17), the left hand side may be defined arbitrarily (the reader should check that such a definition still works). When the denominator in (4.17) is zero but the numerator is not, we have a problem. Then P does not have a density with respect to Q. Readers acquainted with measure theory may object that the last statement is imprecise, the density can still be defined so long as $\Pr\{f_Y(X) = 0\} = 0$.

4.7.2 Importance Sampling

One of the most important techniques in Monte Carlo is the importance sampling trick. This uses one distribution to give information about another. Suppose P has a density f with respect to Q and we want to calculate the expectation (4.9) by Monte Carlo. If direct simulation from P is difficult then a glance at (4.16) tells us another way to do it. That is, one Monte

Carlo approximation of $E_P\{g(X)\}$ is

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \tag{4.18a}$$

where X_1, X_2, ... form an identically P-distributed sequence obeying the SLLN. Another is

$$\frac{1}{n}\sum_{i=1}^{n} g(Y_i)f(Y_i) \tag{4.18b}$$

where Y_1, Y_2, ... form an identically Q-distributed sequence obeying the SLLN. This works because

$$E_P\{g(X)\} = E_Q\{g(Y)f(Y)\}$$

when X has measure P and Y has measure Q. This is just (4.17) rewritten in different notation.

As long as we only discuss the SLLN, both estimates are equally good (that is, they both work). If we look a little deeper and consider the variance of the Monte Carlo estimators, one may be better than the other.

In the simplest case using independent sequences, the variance of (4.18a) is

$$\frac{1}{n}\operatorname{Var}_P\{g(X)\} \tag{4.19a}$$

and the variance of (4.18b) is

$$\frac{1}{n}\operatorname{Var}_Q\{g(Y)f(Y)\}\ . \tag{4.19b}$$

The two variances will typically not be the same. They could only be the same by wild coincidence. One may be much smaller than the other. In the extreme, we might have

$$f(y) \propto \frac{1}{g(y)}$$

which would make the random variable $g(Y)f(Y)$ constant and hence (4.19b) zero. Then there would be no Monte Carlo error, but only because we knew

so much about the problem that we didn't need to use Monte Carlo. In practical applications, one cannot arrange for (4.19b) to be zero but can sometimes arrange that (4.19b) is less than (4.19a).

This variance reduction spin on importance sampling is perhaps the least interesting aspect of it. The problem with classical importance sampling is that it's often not worth the trouble. Suppose (4.19a) is 100 times (4.19b) for the same n. This means that one has to have n 100 times larger in (4.18a) than in (4.18b) to get the same accuracy.

So what? If we're talking about 100 seconds versus 1 second, it's not worth the trouble if it takes you hours of extra time figuring out the importance sampling scheme and coding it up.

But strange as it may seem from what we've said so far, importance sampling is often used when efficiency considerations go the other way, when (4.19a) is smaller than (4.19b). The reason is simple. Sometimes the convenience factor goes the other way. Sometimes the importance sampling estimator (4.18b) is easier to do. That is, simulation from Q is so easy that that's the one that is done.

Example

Consider the parametric family

$$\mathcal{P} = \{ P_\theta : \theta \in \Theta \}$$

of probability measures (a statistical model) and we want to calculate

$$\mu(\theta) = E_\theta\{g(X)\} = \int g(x) P_\theta(dx) \tag{4.20}$$

for all $\theta \in \Theta$. The naive way to calculate this by Monte Carlo does a different simulation X_1, X_2, ... identically P_θ-distributed for each θ and uses the estimator (4.18a) to estimate $\mu(\theta)$. The trouble with the naive idea is that it takes an infinite amount of time to do an infinite number of simulations.

Of course, anyone implementing the naive idea will only actually do a finite number of simulations at a grid of points in Θ, but that is still a lot of time, especially if Θ is not one-dimensional.

Suppose all of the measures in the model are dominated by some probability measure Q, say. Then

$$\mu(\theta) = \int g(x) P_\theta(dx) = \int g(y) f_\theta(y) Q(dy) \tag{4.21}$$

for all functions g for which the expectation exists. Then if Y_1, Y_2, ... form an identically Q-distributed sequence obeying the SLLN

$$\bar{\mu}_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} g(Y_i) f_\theta(Y_i) \tag{4.22}$$

is a Monte Carlo estimate of (4.20).

We get an estimate for all $\theta \in \Theta$ with just one Monte Carlo sample! Now that's real efficiency. But not in terms of the the usual "importance sampling" spin. The variance of $\bar{\mu}_n(\theta)$ will vary with θ and may be very bad for some θ. The idea here is not to have the optimal scheme for calculating any one expectation, but a scheme that does an acceptable job of calculating many expectations.

4.7.3 Normalized Importance Sampling

Recall (4.17). In practice we will often know f only up to a ratio of normalizing constants. That is, if h_X and h_Y are nonnegative functions such that

$$f(w) = \frac{h_X(w)/c_X}{h_Y(w)/c_Y} \tag{4.23}$$

where

$$c_X = \int h_X(w)\, dw$$

and c_Y is defined similarly. We know h_X and h_Y but not c_X or c_Y. Unnormalized densities slightly complicate importance sampling. Formula (4.22) doesn't work. However, a slight variant does.

Note that

$$E_P[g(X)] = E_Q\left[\frac{g(Y)f_X(Y)}{f_Y(Y)}\right] = \frac{c_Y}{c_X}E_Q\left[\frac{g(Y)h_X(Y)}{h_Y(Y)}\right] = \frac{c_Y}{c_X}E_Q[g(Y)w(Y)]$$

where the *unnormalized importance weights* are

$$w(y) = \frac{h_X(y)}{h_Y(y)} \, .$$

Hence if if Y_1, Y_2, $\ldots$ Y_n form an identically Q-distributed sequence obeying the SLLN

$$\frac{1}{n}\sum_{i=1}^{n} g(Y_i)w(Y_i) \to E_Q[g(Y)w(Y)]$$

with probability 1 as $n \to \infty$. Now

$$E_Q[w(Y)] = E_Q\left[\frac{h_X(Y)}{h_Y(Y)}\right] = \frac{c_X}{c_Y}E_Q\left[\frac{f_X(Y)}{f_Y(Y)}\right] = \frac{c_X}{c_Y} \, .$$

Thus, if Y_1, Y_2, $\ldots$ Y_n form an identically Q-distributed sequence obeying the SLLN

$$\frac{1}{n}\sum_{i=1}^{n} w(Y_i) = \frac{1}{n}\sum_{i=1}^{n} \frac{h_X(Y_i)}{h_Y(Y_i)} \to \frac{c_X}{c_Y}$$

with probability 1 as $n \to \infty$. Let

$$w^*(y) = \frac{w(y)}{\sum_{i=1}^{n} w(y_i)}$$

which are called the *normalized importance weights*. Then if Y_1, Y_2, $\ldots$ Y_n form an identically Q-distributed sequence obeying the SLLN

$$\sum_{i=1}^{n} g(Y_i)w^*(Y_i) = \frac{\frac{1}{n}\sum_{i=1}^{n} g(Y_i)w(y_i)}{\frac{1}{n}\sum_{i=1}^{n} w(Y_i)} \to \frac{c_Y}{c_X}E_Q[g(Y)w(Y)] = E_P[g(X)]$$

with probability 1 as $n \to \infty$.

Note that w^* is a function having the properties of a probability distribution on the sample $Y_1, \ldots, Y_n$

$$w^*(Y_i) \geq 0, \qquad i = 1, \ldots, n \qquad \text{and} \qquad \sum_{i=1}^{n} w^*(Y_i) = 1 \; .$$

Suppose that the function g for which we are calculating a Monte Carlo expectation is an indicator function (so the expectation is a probability), hence we write

$$P(A) = E_P\{I_A(X)\} = \int_A P(dx)$$

and the Monte Carlo estimate

$$\bar{P}_n(A) = \sum_{i=1}^{n} I_A(Y_i) w^*(Y_i) \; .$$

Now our intuition about probabilities says these should satisfy the complement rule

$$\bar{P}_n(A^c) = 1 - \bar{P}_n(A)$$

and they do, *but only because we are using normalized importance sampling.* The reader should check that ordinary importance sampling doesn't satisfy the complement rule or many other rules of probability simply because unnormalized importance weights don't add to one.

Thus there is some point to using normalized importance sampling even when we have normalized densities so we could use (4.22) instead of (4.29). The little extra work involved in normalizing the importance weights could save you from a major mistake doing something "intuitively obvious" but completely bogus (like using the complement rule).

Example

Suppose, as is often the case, that the densities f_θ defined by (4.21) are known only up to a constant of proportionality, that is, we know functions

h_θ such that

$$f_\theta(x) \propto h_\theta(x)$$

but we do not know the constant of proportionality. Of course, we must "know" it in the theoretical sense. Its value is determined by the requirement that f_θ integrate to one. Hence

$$f_\theta(x) = \frac{h_\theta(x)}{c(\theta)} \tag{4.24a}$$

where

$$c(\theta) = \int h_\theta(x) Q(dx). \tag{4.24b}$$

But we may not know how to do this integral, so we don't know the "normalizing constant" $c(\theta)$ in any practical sense.

Actually, *normalizing function* might be a better name for (4.24b). It is a "constant" in (4.24a) because the densities f_θ are considered functions of x not θ, but $c(\theta)$ does usually depend on θ, as the notation suggests.

It is useful to have terminology to describe this situation. We say that

$$\mathcal{H} = \{\, h_\theta : \theta \in \Theta \,\}$$

is a family of *unnormalized probability densities* with respect to Q. Then (4.24b) defines the *normalizing function* of the family, and (4.24a) defines the corresponding *normalized densities* f_θ.

The SLLN argument for the validity of (4.22) as a Monte Carlo estimate says with probability 1, as $n \to \infty$

$$\frac{1}{n} \sum_{i=1}^{n} g(Y_i) f_\theta(Y_i) \to \mu(\theta) \; .$$

If we plug in (4.24a) we get, as $n \to \infty$

$$\frac{1}{n} \sum_{i=1}^{n} g(Y_i) \frac{h_\theta(Y_i)}{c(\theta)} \to \mu(\theta)$$

with probability 1, or

$$\frac{1}{n} \sum_{i=1}^{n} g(Y_i) h_\theta(Y_i) \to c(\theta) \mu(\theta) \tag{4.25}$$

The special case of (4.25) with $g \equiv 1$ gives, because $E_\theta(1) = 1$

$$\frac{1}{n} \sum_{i=1}^{n} h_\theta(Y_i) \to c(\theta) \tag{4.26}$$

with probability 1 as $n \to \infty$. Dividing (4.25) by (4.26) we get

$$\frac{\frac{1}{n} \sum_{i=1}^{n} g(Y_i) h_\theta(Y_i)}{\frac{1}{n} \sum_{i=1}^{n} h_\theta(Y_i)} \to \mu(\theta) \tag{4.27}$$

The left hand side of (4.27) is the desired Monte Carlo estimator of $\mu(\theta)$ in the case where densities are unnormalized.

The n numbers $h_\theta(Y_i)$ are the *unnormalized importance weights*. When divided by their sum, they are the *normalized importance weights*

$$w_\theta(y) = \frac{h_\theta(y)}{\sum_{i=1}^{n} h_\theta(Y_i)} \, . \tag{4.28}$$

Then the left hand side of (4.27) can be written as a weighted average

$$\bar{\mu}_n(\theta) = \sum_{i=1}^{n} g(Y_i) w_\theta(Y_i) \, . \tag{4.29}$$

Chapter 5

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods are just like the ordinary Monte Carlo methods of Chapter 4 except that instead of simulating an iid sequence we will now be simulating a realization of a Markov chain. This may not seem to be much of an advance since the Monte Carlo data produced by MCMC methods will not generally be independent or even identically distributed. However, we will see that MCMC is the only truly general method of simulating observations that are at least approximately from the target distribution. Moreover, all of the major concepts used in the discussion of GOFMC carry over to the MCMC setting. In particular, our major goal is still to estimate an unknown expectation

$$E_\pi[g(X)] = \int g(x)\, \pi(dx)$$

where now π is the probability distribution we are interested in using for inference. But the complication is that we are now going to assume that direct simulation from π is impossible. This is the realm where MCMC is most useful.

5.1 Transition Kernels

Throughout this chapter X will be a general space[1] For most statisticians Markov chain theory is somewhat eccentric in its notation in that it doesn't use the "bar" notation for conditional probability. Instead of writing $P(A \mid y)$ we write $P(y, A)$, meaning that the measure $P(y, \cdot)$ is a possibly different measure for each different $y \in \mathsf{X}$. We will be interested in *Markov transition kernels*[2] which are conditional probabilities written as $P(y, A)$.

If P is a Markov kernel and $\lambda(\cdot)$ is a probability measure on $\mathcal{B}(\mathsf{X})$ we define

$$\lambda P(A) := \int_{\mathsf{X}} P(x,\, A)\, \lambda(dx) \qquad \text{for each } A \in \mathcal{B}(\mathsf{X}) \tag{5.1}$$

and if f is a measurable function on X we define the conditional expectation of f as

$$Pf(x) := \int_{\mathsf{X}} f(y)\, P(x,\, dy) \qquad \text{for each } x \in \mathsf{X}. \tag{5.2}$$

5.2 Markov Chains

A *Markov chain* is a sequence $X = \{X_1, X_2, \ldots\} \in \mathsf{X}$ of random elements having the property that the future depends on the past only through the present, that is, for any function g for which the expectations are defined

$$E\{g(X_{n+1}, X_{n+2}, \ldots) \mid X_n, X_{n-1}, \ldots\} = E\{g(X_{n+1}, X_{n+2}, \ldots) \mid X_n\}. \tag{5.3}$$

Let

$$h(X_{n+1}) = E[g(X_{n+1}, X_{n+2}, \ldots) | X_{n+1}] \,.$$

[1]Formally, we require that $(\mathsf{X}, \mathcal{B}(\mathsf{X}))$ be a Polish (complete separable metric) space.

[2]The formal definition is that for each $A \in \mathcal{B}(\mathsf{X})$, $P(\cdot, A)$ is a nonnegative measurable function on X and for each $x \in \mathsf{X}$, $P(x, \cdot)$ is a probability measure on $\mathcal{B}(\mathsf{X})$.

Then using the Markov property (5.3) and the iterated expectation theorem we have

$$\begin{aligned} E\{g(X_{n+1}, X_{n+2}, \ldots) | X_1, \ldots, X_n\} &= E\big\{E[g(X_{n+1}, X_{n+2}, \ldots) | X_1, \ldots, X_{n+1}] \;\big|\; X_1, \ldots, X_n\big\} \\ &= E\big\{E[g(X_{n+1}, X_{n+2}, \ldots) | X_{n+1}] \;\big|\; X_1, \ldots, X_n\big\} \\ &= E\{h(X_{n+1}) | X_1, \ldots, X_n\} \\ &= E\{h(X_{n+1}) | X_n\} \end{aligned}$$

Thus in order to verify that a random sequence is a Markov chain it is enough to verify that

$$E\{g(X_{n+1}, X_{n+2}, \ldots) | X_1, \ldots, X_n\} = E\{h(X_{n+1}) | X_n\} \; . \tag{5.4}$$

holds for all functions h for which the expectations are defined and for all integers n.

Suppose h is an indicator function, i.e., $h(x) = I(x \in A)$ where $A \in \mathcal{B}(\mathsf{X})$. Then the conditional expectations

$$E\{h(X_{n+1}) \mid X_n\} = P_n(X_n, A) \tag{5.5}$$

are the *one-step transition probabilities* of the Markov chain and are Markov transition kernels. In general, the transition probabilities (5.5) are allowed to depend on n, but in most applications they do not. In this case, the Markov chain is said to be *time-homogeneous* and we write $P = P_n$.

The time-homogeneous case is by far the most important. In order to avoid saying "time-homogeneous Markov chain" over and over, this is just taken as part of the definition of "Markov chain." However, there has been a lot of recent interest in "adaptive MCMC" where the Markov chains are *not* time-homogeneous; we probably won't say much more about this but the interested reader might look at Atchade and Rosenthal (2005).

The transition probabilities (5.5) determine the conditional probability distribution of X_2, X_3, ... given X_1. In order to specify the joint distribution

of the whole sequence, we need to specify the marginal distribution of X_1, which is called the *initial distribution* of the Markov chain, say λ. Let P_n be the Markov kernel that gives the distribution of X_n given X_{n-1} for $n = 2, 3, \ldots$. Then we can calculate the so-called "finite-dimensional distributions" of the Markov chain as

$$E\{g(X_1, \ldots, X_n)\} = \iint \cdots \int \lambda(dx_1) P_2(x_1, dx_2) \cdots P_n(x_{n-1}, dx_n) g(x_1, \ldots, x_n) \; .$$

It is a deep theorem of measure theory that the finite-dimensional distributions determine a unique infinite-dimensional distribution for the whole sequence.

Recall (5.1) and (5.2). If P_2, P_3, ... are the transition probability kernels of a general Markov chain, then $P_n P_{n+1}$ is the conditional distribution of X_{n+1} given X_{n-1}. If λ is the marginal distribution of X_{n-1}, then $\lambda P_n P_{n+1}$ is the marginal distribution of X_{n+1}, and multiplication is associative

$$(\lambda P_n) P_{n+1} = \lambda (P_n P_{n+1}).$$

Similarly, if f is a measurable function on the state space, then $P_n P_{n+1} f$ is the conditional expectation of $f(X_{n+1})$ given X_{n-1}, and multiplication is associative

$$(P_n P_{n+1}) f = P_n (P_{n+1} f).$$

If P is the transition kernel of a time-homogeneous Markov chain with, then P^n, meaning $PP \cdots P$ with n factors, is the conditional distribution of X_{n+1} given X_1. If λ is the initial measure of the Markov chain, then λP^n is the marginal distribution of X_{n+1}.

Examples

The classical presentation of Markov chains centers on the case where X is at most countable. For example, suppose $\mathsf{X} = \{0, 1, 2, 3, \ldots\}$. In this case,

the transition kernel is a matrix

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ \vdots & & & \end{pmatrix}$$

where each $P_{ij} \geq 0$ and $\sum_{j=0}^{\infty} P_{ij} = 1$ for each $i \geq 0$. The P_{ij} are the one-step transition probabilities.

Example 5.2.1. *Suppose X lives on $\mathsf{X} = \mathbb{Z}$ such that if $x \geq 1$ and $0 < \theta < 1$ then a time-homogeneous Markov chain is defined by the transition kernel*

$$P(x, x+1) = P(-x, -x-1) = \theta\,, \quad P(x, 0) = P(-x, 0) = 1 - \theta\,,$$

$$P(0, 1) = P(0, -1) = \frac{1}{2}\,.$$

Example 5.2.2. *This example concerns a simple hard-shell (also known as hard-core) model. Suppose $\mathcal{X} = \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \subseteq \mathbb{Z}^2$. A proper configuration on $\mathcal{X}$ consists of coloring each point either black or white in such a way that no two adjacent points are white. Let X denote the set of all proper configurations on $\mathcal{X}$ and $N_{\mathsf{X}}(n_1, n_2)$ be the total number of proper configurations.*

Consider the following Markov chain on X. Fix $p \in (0, 1)$ and set $X_0 = x_0$ where $x_0 \in \mathsf{X}$ is an arbitrary proper configuration. Randomly choose a point $(x, y) \in \mathcal{X}$ and independently draw $U \sim$ Uniform$(0, 1)$. If $u \leq p$ and all of the adjacent points are black then color (x, y) white leaving all other points alone. Otherwise, color (x, y) black and leave all other points alone. Call the resulting configuration X_1. Continuing in this fashion yields a Markov chain $\{X_0, X_1, X_2, \ldots\}$ on X.

Most Markov chains encountered in MCMC live on uncountable state spaces. (This doesn't have to mean that they are complicated, however.)

Often the transition kernel can be represented as the integral of a conditional density $k(y \mid x)$, say. Then

$$P(x, A) = \int_A k(y \mid x)\, dy \,.$$

Example 5.2.3. *Consider a Markov chain that evolves on* $\mathsf{X} = (0, 1)$ *as follows. Suppose* $X_n = x$ *and independently draw* $U \sim \mathit{Uniform}\,(0, 1)$*. If* $u \le 1/2$ *then* $X_{n+1} \sim \mathit{Uniform}\,(0, x)$ *but if* $u > 1/2$ *then* $X_{n+1} \sim \mathit{Uniform}\,(x, 1)$*. Then the* X_{n+1} *can be thought of as drawn from the distribution having conditional density*

$$k(y \mid x) = \frac{1}{2}\frac{1}{x} I(0 < y < x) + \frac{1}{2}\frac{1}{1 - x} I(x < y < 1)\,,$$

that is, the Markov kernel determined by

$$P(x, A) = \int_A k(y \mid x)\, dy \,.$$

5.3 Regularity Conditions

A Markov chain is *stationary* if the marginal distribution of X_n does not depend on n. When the Markov chain is stationary, among the variables having the same marginal distribution is X_1, so another way to discuss stationarity is to say that the initial distribution is the same as the marginal distribution of all the variables. Such a distribution is said to be a *stationary distribution* or an *invariant distribution* for the Markov chain. Formally, π is an *invariant distribution* for a Markov kernel P if $\pi P = \pi$.

Not all Markov chains have stationary distributions. But all of those of use in MCMC do. Moreover, there is never any issue about whether a Markov chain for MCMC has a stationary distribution or what it is, because, as we shall see, all Markov chains for MCMC are constructed to have a specified stationary distribution.

There is, however, a uniqueness question. A Markov chain can have more than one stationary distribution. It is not always obvious whether a Markov chain for MCMC has a unique stationary distribution. If it doesn't have a unique stationary distribution, it is useless for MCMC. Thus the uniqueness question is important, but since it wanders off into fairly obnoxious theory, we will just punt on it. In fact, we will assume more than is required to get a unique invariant distribution. Our standing set of assumptions (henceforth known as the *usual regularity conditions*) are that the Markov chain having invariant distribution π is *aperiodic*, *π-irreducible* and *positive Harris recurrent*

1. Aperiodic means that we cannot partition X in such a way that the Markov chain makes a regular tour through the partition.

2. π-irreducible means that if $\pi(A) > 0$ then there is a positive probability that the chain will eventually visit A.

3. Positive Harris recurrent. "Positive" means that π is a probability distribution; "Harris recurrent" means that no matter the starting distribution of the Markov chain every set of positive π-measure will be visited infinitely often if the chain is run forever.

Note that the assumption of positive Harris recurrence is actually stronger than irreducibility. From a practical point of view, the usual regularity conditions imply that the starting value is irrelevant and that the chain will thoroughly explore the state space as the number of iterations grows large. A Markov chain X satisfying the usual regularity conditions is said to be *Harris ergodic*.

Examples

Example 5.3.1. *Recall the Markov chain defined in Example 5.2.1. This chain is Harris ergodic and its stationary distribution is a vector* $\pi = (\ldots, \pi(-1), \pi(0), \pi(1), \ldots)$

satisfying $\pi P = \pi$. *A straightforward calculation shows that* π *is given by* $\pi(0) = (1-\theta)/(2-\theta)$ *and for* $x \geq 1$

$$\pi(x) = \pi(-x) = \pi(0)\frac{\theta^{x-1}}{2} .$$

Example 5.3.2. *Consider the Markov chain defined in Example 5.2.3. This chain is Harris ergodic and its stationary distribution has density* g *satisfying*

$$g(y) = \int_0^1 k(y \mid x) g(x)\, dx$$

which holds if

$$g(x) = \frac{1}{\pi\sqrt{x(1-x)}} .$$

5.3.1 Reversible Markov Chains

If P is a Markov kernel, then P is *reversible* with respect to a measure π if

$$\iint \pi(dx)P(x, dy)g(x, y) = \iint \pi(dy)P(y, dx)g(y, x) \tag{5.6}$$

whenever g is such that the integrals exist (g is bounded, for example).

Plugging $g(x, y) = I_A(y)$ into (5.6) we get

$$\int \pi(dx)P(x, A) = \int_A \int \pi(dx)P(x, dy) = \int_A \pi(dx) = \pi(A),$$

which is $\pi P = \pi$. Hence P reversible with respect to π implies π is invariant for P.

The reason the notion is called "reversible" (sometimes "time reversible") is that when π is used as the initial distribution (a generally impossible task), so the chain is stationary, (5.6) has the interpretation that the joint distribution of the pair (X_n, X_{n+1}) is the same as the joint distribution of the pair (X_{n+1}, X_n) with the order reversed. From this one easily shows that the k-tuple $(X_{n+1}, \ldots, X_{n+k})$ has the same joint distribution as the k-tuple $(X_{n+k}, \ldots, X_{n+1})$ with the order reversed.

Thus we say the *stationary* chain with reversible kernel P looks the same (in distribution) running forward or backward. Note well that a *non-stationary* chain with kernel P will *not* look the same running forward or backward.

The main use of reversibility in MCMC is constructing kernels that have a specified invariant distribution. Given π, find a P such that P is reversible with respect to π. It turns out that this is a much easier problem than: given π find a P that *preserves* π (meaning π is invariant for P). The reason is that $\pi P = \pi$ is a difficult integral equation to solve for P given π, whereas (5.6) although even more difficult if considered as an integral equation to solve, is quite trivial when considered as a symmetry condition to check (swapping x and y in the argument of g doesn't change anything). The reversibility condition is sometimes written

$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx)$$

meaning that if we hit both sides with $g(x, y)$ and integrate, we get the same thing, regardless of what function g we use (so long as the expectations exist).

5.4 Asymptotics for Markov Chains

5.4.1 Total Variation

A basic issue involved in MCMC is trying to describe how "far apart" the distribution of X_n is from the target. Hopefully, after many iterations these distributions are "close." The most common way of measuring this discrepancy is via the *total variation* norm.

$$\|P^n(x, \cdot) - \pi(\cdot)\| = \sup_{A \in \mathcal{B}(\mathsf{X})} |P^n(x, A) - \pi(A)| \,. \tag{5.7}$$

Generally speaking, it is rare that P^n is available in an analytically tractable form. It is sometimes possible to think of total variation in terms of densities

rather that measures; see the appendix in section 5.6.

Harris ergodic Markov chains enjoy a nice form of convergence. Specifically, (see Meyn and Tweedie (1993a, p. 323)) for all $x \in \mathsf{X}$

$$\|P^n(x,\cdot) - \pi(\cdot)\| \downarrow 0 \quad \text{as} \quad n \to \infty, \tag{5.8}$$

Note that (5.8) says that

$$|P^n(x,A) - \pi(A)| \to 0$$

for every π–continuity set A which is equivalent to convergence in distribution; see Billingsley (1995, Theorem 25.8). Thus, a Harris ergodic Markov chain started from any point in the state space will eventually produce observations that look like they were drawn from the target distribution π.

Later we will be concerned with the rate of total variation convergence. Let $M(x)$ be a nonnegative function and $\gamma(n)$ be a nonnegative decreasing function on $\mathbb{N}$ such that

$$\|P^n(x,\cdot) - \pi(\cdot)\| \leq M(x)\gamma(n)\ . \tag{5.9}$$

When X is *geometrically ergodic* (5.9) holds with $\gamma(n) = t^n$ for some $t < 1$. *Uniform ergodicity* means M is bounded and $\gamma(n) = t^n$ for some $t < 1$. *Polynomial ergodicity of order m* where $m \geq 0$ corresponds to $\gamma(n) = n^{-m}$.

Establishing (5.9) directly may be difficult in general. However, there are constructive methods for establishing the existence of an appropriate M and γ; see Jarner and Roberts (2002), Jones and Hobert (2001) and Meyn and Tweedie (1993a) for a complete introduction to these methods.

5.4.2 The Strong Law of Large Numbers (SLLN)

A Harris ergodic Markov chain $X = (X_1, X_2, \ldots)$ having stationary distribution π satisfies the law of large numbers; that is if $E_\pi|g(X)| < \infty$ then

as $n \to \infty$

$$\bar{g}_n := \frac{1}{n} \sum_{i=1}^{n} g(X_i) \stackrel{a.s.}{\to} E_\pi[g(X)]. \tag{5.10}$$

Dependence on the Initial Distribution

Note that if the chain is not stationary the SLLN still holds, even though none of the X_i have the stationary distribution π. In fact, it is typically the case that

$$E_\pi[g(X)] \neq E[g(X_i)], \qquad \text{for all } i.$$

(And hence $\bar{g}_n$ is a biased estimate of $E_\pi\{g(X)\}$.) The SLLN holds for *any* initial distribution of the Markov chain (Meyn and Tweedie, 1993a, Theorem 17.1.6). This is one aspect of what we mean by saying the initial distribution is irrelevant in MCMC. The other shoe will drop when we discuss the CLT.

5.4.3 MCMC

As we discussed in the introduction, MCMC is just like GOFMC except that X_1, X_2, ... is a Harris ergodic Markov chain with a specified stationary distribution π. Basically, MCMC is the practice of using the left hand side of (5.10) as an estimate of the right hand side, just as GOFMC is the same practice when X_1, X_2, ... are iid with distribution π. In fact, GOFMC is a special case of MCMC because iid sequences are Markov chains too.

Note that all of the arguments in Chapter 4 were based on the SLLN and since (5.10) is exactly the same everything in Chapter 4 applies to MCMC.

5.4.4 The Central Limit Theorem (CLT)

The CLT is the basis of all error estimation in Monte Carlo, MCMC or GOFMC. For large Monte Carlo sample sizes n, and generally we do take

very large sample sizes as a matter of course, the distribution of Monte Carlo estimates is approximately normal so the asymptotic variance tells the whole story about accuracy of estimates.

To simplify notation, let us define notation for the two sides of (5.10)

$$\bar{g}_n = \frac{1}{n} \sum_{i=1}^{n} g(X_i) \tag{5.11a}$$

is the Monte Carlo estimate, and

$$\mu = E_\pi\{g(X)\} \tag{5.11b}$$

is the expectation being estimated. Then the SLLN says

$$\bar{g}_n \overset{a.s.}{\to} \mu$$

and the CLT says that as $n \to \infty$

$$\sqrt{n}(\bar{g}_n - \mu) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \sigma^2) \tag{5.11c}$$

where σ^2 is some nonnegative constant.

In the iid case, the CLT is completely understood: (5.11c) holds if and only if the $\mathrm{Var}[g(X_i)] < \infty$ and, moreover,

$$\sigma^2 = \mathrm{Var}[g(X_i)] \tag{5.11d}$$

and is easily estimated by the sample variance of $g(X_1)$, ..., $g(X_n)$.

In the general Markov chain case, the CLT is incompletely understood. Whether or not $\mathrm{Var}[g(X_i)]$ exists doesn't control whether or not (5.11c) holds. The CLT (5.11c) can fail when the variance exists and hold when the variance doesn't exist. When the CLT does hold, σ^2 is generally not given by (5.11d).

The last point is not surprising. It is just a consequence of the fact that the variance of a sum is the sum of the variances if and only if the terms are uncorrelated. So in the iid case

$$\mathrm{Var}(\bar{g}_n) = \frac{\sigma^2}{n}$$

but in general

$$\operatorname{Var}(\bar{g}_n) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \operatorname{cov}\{g(X_i), g(X_j)\}. \tag{5.11e}$$

Dependence on the Initial Distribution

One thing that is completely understood about the Markov chain CLT is that if (5.11c) holds for any initial distribution, then it holds for every other initial distribution, and the asymptotic variance σ^2 is the same regardless of the distribution (Meyn and Tweedie, 1993a, Theorem 17.1.6).

This is the other shoe dropping. The initial distribution is irrelevant in MCMC in that neither the SLLN nor the CLT depends on it.

Calculation of the Asymptotic Variance

The upshot of the preceding section is that "without loss of generality" we may assume stationarity. Even though we cannot use stationary chains in MCMC (if we could produce even one sample X_1 from the stationary distribution to start MCMC, we could produce many iid samples and do GOFMC), the CLT for the stationary chain is no different from the CLT for the chain we actually use.

The variance formula (5.11e) can be simplified a bit using stationarity, which implies that the joint distribution of X_n and X_{n+k} depends only on k not upon n. Hence all of the terms in (5.11e) having the same difference between i and j are the same, and we can rewrite (5.11e) as

$$n \operatorname{Var}_\pi(\bar{g}_n) = \operatorname{Var}_\pi\{g(X_i)\} + 2 \sum_{k=1}^{n-1} \frac{n-k}{n} \operatorname{cov}_\pi\{g(X_i), g(X_{i+k})\} \tag{5.11f}$$

(the subscripts π are there to remind us that this is valid only for the stationary chain). In the time series literature, the quantity

$$\gamma_k = \operatorname{cov}_\pi\{g(X_i), g(X_{i+k})\} \tag{5.11g}$$

is called the *lag k autocovariance* of the stationary time series $g(X_1)$, $g(X_2)$, ... The function $k \mapsto \gamma_k$ is called the *autocovariance function* of the time series. Using this notation, we can rewrite (5.11f) as

$$n \operatorname{Var}_\pi(\bar{g}_n) = \gamma_0 + 2 \sum_{k=1}^{n-1} \frac{n-k}{n} \gamma_k \tag{5.11h}$$

Since $(n-k)/n \to 1$ as $n \to \infty$ one might suspect that the right hand side of (5.11h) converges to

$$\sigma_g^2 = \gamma_0 + 2 \sum_{k=1}^{\infty} \gamma_k \tag{5.11i}$$

if it converges at all. In fact, this is not quite true. It is mathematically possible for the right hand side of (5.11h) to converge when the infinite sum in (5.11i) does not converge. But in all cases in which the CLT is known to hold (5.11i) gives the asymptotic variance.

Just what are the conditions to guarantee a Markov chain CLT? This is an important question. Not every Markov chain enjoys a CLT and it doesn't have to be a pathological example.

Example 5.4.1. *Consider a Markov chain that evolves as follows. Let the current state be $X_n = x$. Draw $y \sim$ Pareto(α, λ) and independently draw $u \sim$ Uniform$(0, 1)$. Set $X_{n+1} = y$ if*

$$u < x^{\beta-\lambda} y^{\lambda-\beta}$$

otherwise set $X_{n+1} = x$. This defines a Harris ergodic Markov chain on $[\alpha, \infty)$ with stationary distribution is a Pareto(α, β) and is known as a Metropolis–Hastings independence sampler. We will introduce this algorithm more formally in the next chapter. Consider estimating the mean of the stationary distribution, that is,

$$\frac{\alpha\beta}{\beta - 1}.$$

Using results from Mengersen and Tweedie (1996) shows that a CLT will hold if $\lambda \leq \beta$ but an application of results from Roberts (1999) gives that a CLT cannot hold if $\lambda > 2\beta$. There is a grey area for $\beta < \lambda \leq 2\beta$.

This is illustrated empirically in Figure 5.4.4 where three different simulations were performed. Each panel of the figure is the result of performing 1000 independent replications of the above algorithm each for a length of 1000. For each replication $\bar{x}_n$ was computed and saved. The plots are histograms of these empirical means. The top panel's settings are $\alpha = 1$, $\beta = 4$ and $\lambda = 3$, the middle panel's settings are $\alpha = 1$, $\beta = 4$ and $\lambda = 6$, while the bottom panel has $\alpha = 1$, $\beta = 4$ and $\lambda = 9$. A CLT is apparent in the top panel while in the middle panel the chain may not have been run sufficiently long for a CLT to "kick in" and the theory says that the bottom panel will never (no matter how long it is run) enjoy a CLT.

There are many papers on when a Markov chain CLT holds but one the cleanest statements is given by the following result.

Theorem 5.4.1. *Let X be a Harris ergodic Markov chain on X with invariant distribution π and let $g : \mathsf{X} \to \mathbb{R}$. Assume one of the following conditions:*

1. *X is polynomially ergodic of order $m > 1$, $E_\pi M < \infty$ and there exists $B < \infty$ such that $|g(x)| < B$ almost surely;*
2. *X is polynomially ergodic of order m, $E_\pi M < \infty$ and $E_\pi |g(x)|^{2+\delta} < \infty$ where $m\delta > 2 + \delta$;*
3. *X is geometrically ergodic and $E_\pi |g(x)|^{2+\delta} < \infty$ for some $\delta > 0$;*
4. *X is geometrically ergodic, reversible and $E_\pi g^2(x) < \infty$; or*
5. *X is uniformly ergodic and $E_\pi g^2(x) < \infty$.*

Then for any initial distribution, as $n \to \infty$

$$\sqrt{n}(\bar{g}_n - E_\pi g) \xrightarrow{\mathcal{D}} N(0, \sigma_g^2) \; .$$

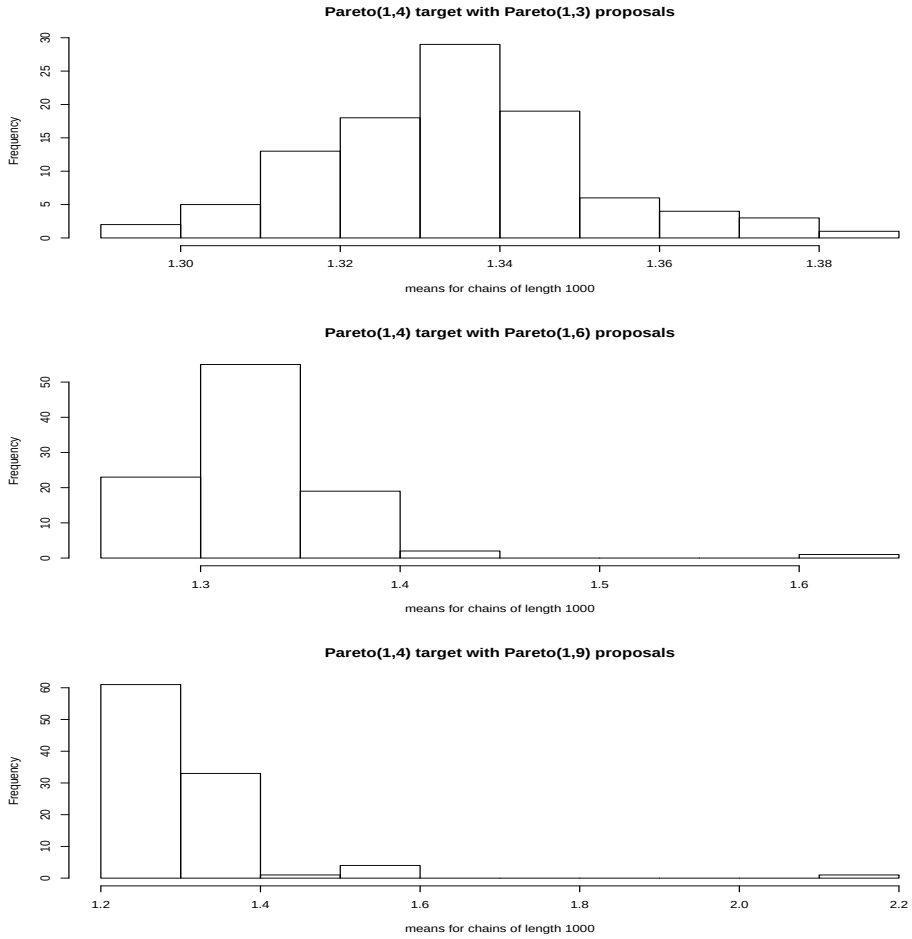

Figure 5.1: An illustration for Example 5.4.1.

The theorem was proved by Ibragimov and Linnik (1971) (condition 5), Roberts and Rosenthal (1997) (condition 4) and Chan and Geyer (1994) (Condition 3). See Jones (2004) for details on conditions 1 and 2 and an overview of the other results. That is the end of the story of the CLT, at least as far as we are concerned. Readers who want to know more must look elsewhere.

5.4.5 Estimating the Variance

In order to get Monte Carlo standard errors of estimates, we need to estimate the variance (5.11i). Generally speaking, this can be difficult. In this section we will consider two of the most basic and effective methods. However, these methods are not without limitations. For other approaches the interested reader should look at Geyer (1992) and Jones et al. (2005).

Batch Means

Suppose $n = ab$ and hence $a = a_n$ and $b = b_n$ are functions of n. This method is based on

$$n \operatorname{Var}(\bar{g}_n) \to \sigma_g^2, \qquad \text{as } n \to \infty.$$

This is not a consequence of the CLT, since convergence in distribution doesn't imply convergence of moments but sometimes may be proved with additional work and sometimes with similar conditions as required for a CLT. Hence

$$m \operatorname{Var}(\bar{g}_m) \approx n \operatorname{Var}(\bar{g}_n)$$

whenever m and n are both large. Thus a (not very good) estimate of $m \operatorname{Var}(\bar{g}_m)$ is

$$m\bigl(\bar{g}_m - \bar{g}_n\bigr)^2$$

where we are thinking here that $1 \ll m \ll n$ meaning m is large compared to 1 but small compared to n (which means n is very large).

As everywhere else in statistics, we can increase precision by averaging. If the Markov chain were stationary, every block of length b would have the same joint distribution. For some reason, early in the history of this subject, the blocks were dubbed "batches" so that is what we will call them. A *batch* of length b of a Markov chain X_1, X_2, ..., is b consecutive elements of the chain. For example, the first batch is

$$X_0, X_1, \ldots, X_{b-1}$$

The *batch mean* is the sample mean of the batch

$$\bar{g}_j = \frac{1}{b} \sum_{i=(j-1)b}^{jb-1} g(X_i) \tag{5.12}$$

The batch means estimator of σ_g^2 is

$$\hat{\sigma}_{BM}^2 = \frac{b}{a-1} \sum_{j=1}^{a} (\bar{Y}_j - \bar{g}_n)^2 \, . \tag{5.13}$$

If the number of batches is fixed (5.13) is not a consistent estimator of σ_g^2 (Glynn and Iglehart, 1990; Glynn and Whitt, 1991). However, if the batch size *and* the number of batches are allowed to increase with n it may be possible to obtain consistency. The following theorem was proved by Jones et al. (2005).

Theorem 5.4.2. *Assume $g : \mathsf{X} \to \mathbb{R}$ such that $E_\pi |g|^{2+\epsilon_1+\epsilon_2} < \infty$ for some $\epsilon_1 > 0$, $\epsilon_2 > 0$ and let X be a Harris ergodic Markov chain with invariant distribution π. Further, suppose X is geometrically ergodic. If*

1. *$a_n \to \infty$ as $n \to \infty$,*

2. *$b_n \to \infty$ and $b_n/n \to 0$ as $n \to \infty$,*

3. $b_n^{-1} n^{2\alpha} [\log n]^3 \to 0$ *as* $n \to \infty$ *where* $\alpha = 1/(2+\epsilon_2)$ *and*

4. *there exists a constant* $c \geq 1$ *such that* $\sum_n (b_n/n)^c < \infty$

then as $n \to \infty$, $\hat{\sigma}^2_{BM} \to \sigma_g^2$ *with probability 1.*

Remark 5.4.1. *It is common to use* $b_n = \lfloor n^\theta \rfloor$ *and* $a_n = \lfloor n/b_n \rfloor$. *If* $1 > \theta > (1+\epsilon_2/2)^{-1}$ *conditions 1–4 of the theorem are met. A rule of thumb is to use* $\theta = 1/2$.

If the batch means procedure is performed according to the conditions of Theorem 5.4.2 we will call it *consistent batch means* (CBM) in order to distinguish it from the batch means (BM) procedure with a fixed number of batches or batch sizes. CBM produces an asymptotically valid confidence interval for $E_\pi g$ via

$$\bar{g}_n \pm t_{a_n - 1} \frac{\hat{\sigma}_{BM}}{\sqrt{n}} \tag{5.14}$$

where t_{a_n-1} is the appropriate quantile from a student's t distribution with $a_n - 1$ degrees of freedom. But this should (as with any other procedure based on estimating σ_g^2) used with caution. If n isn't sufficiently large (whatever that means) the estimate $\hat{\sigma}_{BM}$ isn't going to be any good. On the other hand, it should be obvious that using CBM to produce 5.14 will result in intervals with better coverage than if BM were used; see Jones et al. (2005).

Overlapping Batch Means

A generalization of BM is the method of overlapping batch means (OLBM). Note that there are $n - b + 1$ batches of length b, indexed by k running from zero to $n-b$. The method of overlapping batch means Meketon and Schmeiser (1984) averages all of them. Its estimate of variance is

$$\operatorname{Var} \bar{g}_n \approx \frac{b}{n} \cdot \frac{1}{n-b+1} \sum_{k=0}^{n-b} \bigl(\bar{g}_k - \bar{g}_n\bigr)^2 \tag{5.15}$$

Of course $\bar{g}_k$ is almost the same as $\bar{g}_{k+1}$, because the batches differ by only one element. So there is little point to using all the batches. If we only used half of them

$$\operatorname{Var} \bar{g}_n \approx \frac{b}{n} \cdot \frac{1}{\lfloor (n-b)/2 \rfloor + 1} \sum_{k=0}^{\lfloor (n-b)/2 \rfloor} \bigl(\bar{g}_{2k} - \bar{g}_n \bigr)^2$$

where $\lfloor x \rfloor$ denotes the "floor" of x, the largest integer not exceeding x, our variance estimate would be almost as good. The reason for the name OLBM is that in the early days of the method, intuition told the inventors of the batch means idea that they should use nonoverlapping batches only. Thus the unqualified term "batch means" refers to nonoverlapping batch means (NOLBM). Empirically, OLBM seemed like a big improvement over NOLBM. In hindsight, there was never any good reason for using nonoverlapping batches, so OLBM is the obvious implementation of the batch means idea. However, the asymptotic properties of OLBM are much less well understood than those of nonoverlapping batch means; see e.g. Theorem 5.4.2. That is, the major criticism of overlapping batch means is that, as described here, it is not guaranteed to produce a consistent estimator of σ_g^2.

If no account is taken of the extra work in computing the batch means for more batches, the optimal estimate uses all the batches. You don't get a better answer by using less information.

5.5 Toy Example: Normal AR(1) Markov Chains

Consider the normal AR(1) time series defined by

$$X_{n+1} = \rho X_n + Z_n \tag{5.16}$$

where Z_1, Z_2, ... are normal with mean zero and Z_n is independent of X_1, ..., X_n. In the time series literature, the Z_i are called the *innovations* and their

variance the *innovations variance*. Let τ^2 denote the innovations variance. Then

$$\operatorname{Var}(X_{n+1}) = \rho^2 \operatorname{Var}(X_n) + \operatorname{Var}(Z_n)$$

shows that in order for the sequence to be stationary, which requires that

$$\operatorname{Var}(X_n) = \sigma_X^2$$

not depend on n, we must have

$$\sigma_X^2 = \frac{\tau^2}{1 - \rho^2} \tag{5.17}$$

which requires $\rho^2 \leq 1$ in order for (5.17) to define a variance and $\rho^2 < 1$ in order for the X_i to have a non-degenerate distribution. Now the fact that the sum of normals is normal shows that the normal distribution with mean zero and variance (5.17) is a stationary distribution of this Markov chain.

Asymptotic Variance

Suppose we want to estimate the mean of the stationary distribution (i.e., zero) by Monte Carlo so the Monte Carlo estimator is the sample mean of X_1, X_2, ... or $\bar{X}_n$. Then $\bar{X}_n$ is $AN(0, \sigma_X^2(1+\rho)/(1-\rho))$ since the autocovariance function of this random sequence can be calculated as follows. First

$$\gamma_0 = \operatorname{Var}(X_i) = \sigma_X^2$$

Then

$$\gamma_n = \operatorname{cov}(X_{n+1}, X_1) = \rho \operatorname{cov}(X_n, X_1) + \operatorname{cov}(Z_n, X_1) = \rho\gamma_{n-1}$$

gives a recursion formula, from which we can calculate

$$\gamma_n = \rho^n \sigma_X^2$$

and hence the variance in the Markov chain CLT (5.11i)

$$\begin{aligned} \sigma^2 &= \gamma_0 + 2 \sum_{k=1}^{\infty} \gamma_k \\ &= \sigma_X^2 \left(1 + 2 \sum_{k=1}^{\infty} \rho^k \right) \\ &= \sigma_X^2 \frac{1 + \rho}{1 - \rho} \end{aligned} \tag{5.18}$$

This is just about the only example where we can calculate the variance in the CLT analytically (so this is a really unique and precious toy problem).

Note that the variance (5.18) goes to infinity as $\rho \uparrow 1$ so this gives us examples that are arbitrarily bad for MCMC. Of less interest is the fact that (5.18) goes to zero as $\rho \downarrow -1$, so this gives an example in which MCMC is arbitrarily better than GOFMC. The reason this isn't interesting is that it is just a toy problem. Real examples often show arbitrarily bad behavior, but real examples don't come close to GOFMC in performance (we always prefer GOFMC whenever we can figure out how to do independent sampling).

Bias

If a stationary chain is used for MCMC, there is no bias: $E(\bar{X}_n) = \mu$. But, of course, in real life we cannot use stationary chains, so there will be bias. In particular, if we consider initial distributions concentrated at one point, which is the same as conditioning on X_1, the bias is

$$\begin{aligned} E(\bar{X}_n \mid X_1) &= E\left(\frac{1}{n} \sum_{i=1}^{n} X_i \,\middle|\, X_1 \right) \\ &= \frac{1}{n} \sum_{i=1}^{n} E(X_i \mid X_1) \end{aligned}$$

In our toy problem, we can actually calculate the bias, something we can

never do in a practical problem.

$$\begin{aligned} E(\bar{X}_n \mid X_1) &= \frac{1}{n} \sum_{i=1}^{n} E(X_i \mid X_1) \\ &= \frac{1}{n} \sum_{i=1}^{n} \rho^{i-1} X_1 \\ &= \frac{1}{n} \cdot \frac{1 - \rho^n}{1 - \rho} X_1 \end{aligned}$$

Note that the term ρ^n is negligible compared to one for large n so

$$E(\bar{X}_n \mid X_1) \approx \frac{X_1}{n(1 - \rho)} \tag{5.19}$$

or if one is really fussy, since $|\rho| < 1$ is required for stationarity, we have the bound

$$|E(\bar{X}_n \mid X_1)| \le \frac{2|X_1|}{n(1 - \rho)}$$

Thus the bias is $O(n^{-1})$. This is no surprise, since in general the influence of the stationary distribution is $O_p(n^{-1})$. That general result does not imply the specific result that the bias is $O(n^{-1})$, but it does agree with it.

Numerical Example

Let $\rho = .95$ and $\tau = 1$. The top panel of Figure 5.2 (p. 144) shows one sample path ($n = 10000$) of a normal AR(1) time series. The high autocorrelation is evident. It is even more easily seen in the bottom panel, which shows the initial one-tenth of the same sample path. The sample mean of the run shown in the top panel of Figure 5.2 is $\bar{X}_n = -0.10235$ (the true value is $\mu = 0$).

In this case we will estimate the the asymptotic variance using OLBM. Figure 5.3 (p. 145) plots the batch means for batch length 100 for the run shown in the top panel of Figure 5.2, that is, it plots $\bar{X}_{m,k}$ defined by (5.12) versus k. This is, of course, another time series. The method of OLBM says

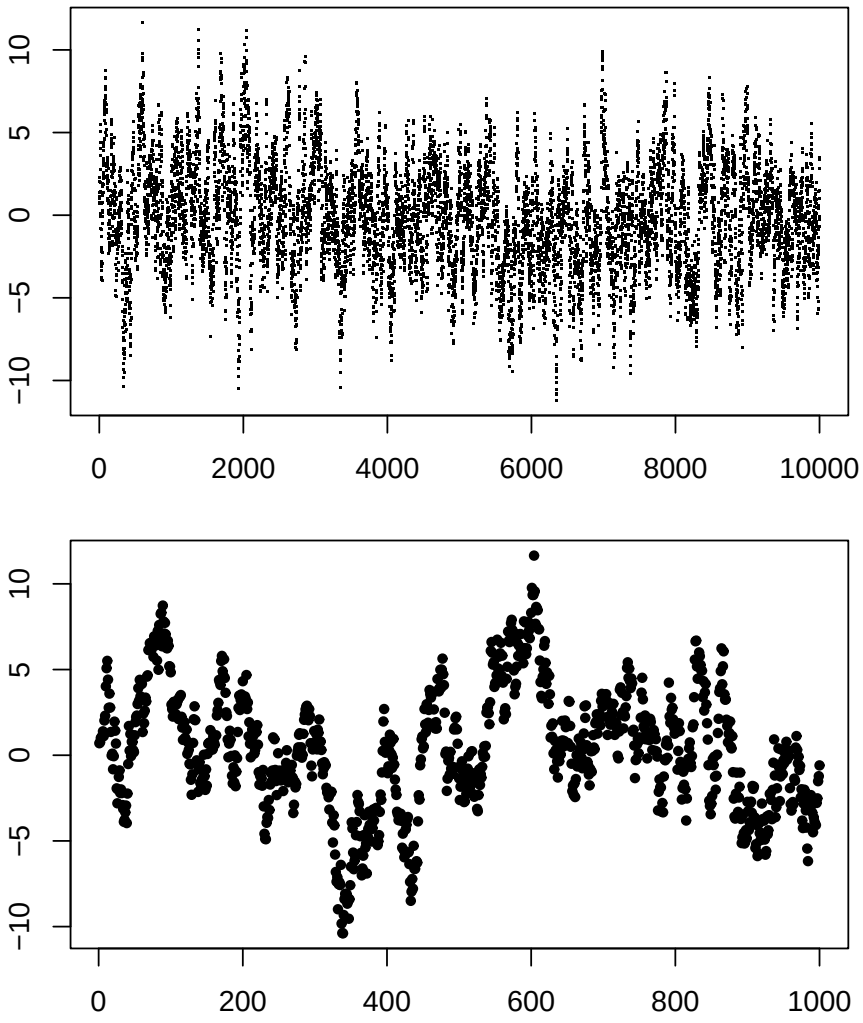

Figure 5.2: (Top) One run of a stationary normal AR(1) time series with $\rho = 0.95$ and $\tau = 1$. (Bottom) The initial tenth (1000 steps) of the same run.

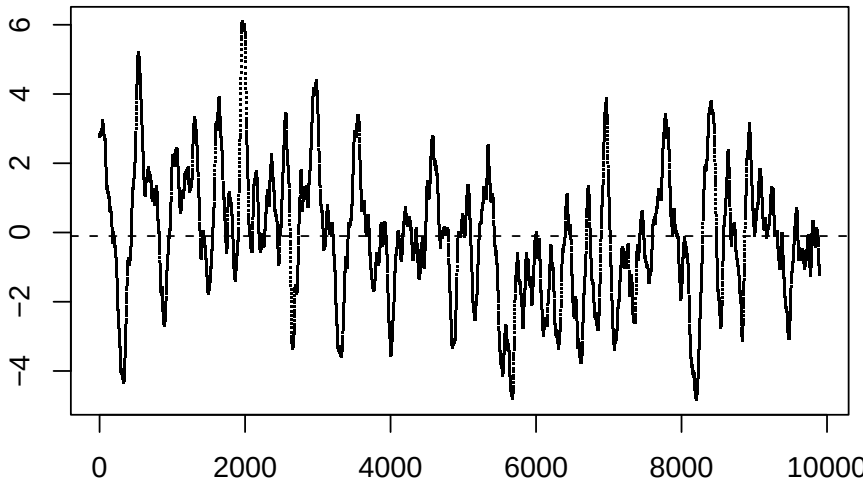

Figure 5.3: Batch means for batch length 100 for the normal AR(1) time series shown in the top panel of Figure 5.2. The ordinate of the dotted line is the sample mean of the whole sequence.

the sample variance of this time series approximates σ^2/m, where σ^2 is the variance in the CLT (5.11i) and m is the batch length. Actually, the proper formula (5.15) is not precisely proportional to the sample variance of the time series of batch means, because $\bar{X}_n$ is not precisely the sample mean of the $\bar{X}_{m,k}$, but for purposes of developing intuition about what's going on the analogy is close enough. The variance estimate (5.15) for the run shown in the top panel of Figure 5.2 is

$$\frac{\hat{\sigma}_n^2}{n} = 0.0363969 \tag{5.20}$$

and, of course, its square root (0.1908) is the MCSE. Thus we can report our results as a little table

estimate	standard error
−0.10	0.19

Presumably, all readers are sophisticated enough to know how to interpret

standard errors, but if we like we can make a confidence interval for the true unknown value (here known to be $\mu = 0$ because of the toyness of the problem). An approximate 95% confidence interval would, of course, be $-0.10 \pm 2 \times 0.19$ or $(-0.48, 0.28)$. We see that (no surprise) statistics works and the confidence interval actually covers the true value. We also know that in actual practice a 95% confidence interval will fail to cover 5% of the time, so failure of the interval to cover wouldn't necessarily have indicated a problem.

We would like to point out that our variance estimate isn't very good. We can calculate σ^2 in this problem. From (5.18) and (5.17) we have

$$\sigma^2 = \sigma_X^2 \frac{1+\rho}{1-\rho} = \frac{\tau^2}{1-\rho^2} \cdot \frac{1+\rho}{1-\rho} = 400$$

for the parameters $\rho = 0.95$ and $\tau = 1$ used in Figure 5.2. In contrast, our estimate, (5.20) times n is

$$\hat{\sigma}_n^2 = 363.97$$

The asymptotic theory behind our confidence interval assumes that n is so large that the difference between $\hat{\sigma}_n^2$ and σ^2 is negligible. The difference here is obviously not negligible. So we can't expect our nominal 95% confidence interval to actually have 95% coverage. The Monte Carlo sample size n must be much larger for all the asymptotics we so casually assume to hold. This is a very common phenomenon; obtaining a good estimate of an asymptotic variance often require larger sample sizes than estimating an asymptotic mean.

5.6 Appendix: Total Variation

Let μ and ν be two probability measures defined on the same measurable space $(\mathsf{X}, \mathcal{B})$. The *total variation distance* of μ and ν is defined as

$$\|\mu - \nu\| = \sup |\mu(\cdot) - \nu(\cdot)| \,. \tag{5.21}$$

Theorem 5.6.1. *If α is any σ–finite measure which dominates μ and ν then*

$$\|\mu - \nu\| = \frac{1}{2}\int |r_1 - r_2|\, d\alpha \tag{5.22}$$

where $r_1 = d\mu/d\alpha$ and $r_2 = d\nu/d\alpha$ are Radon–Nikodym derivatives.

Proof. (Billingsley, 1968, p. 224) Let $\phi = r_1 - r_2$. Then we have

$$0 = \int \phi(x)\alpha(dx) = \int [I_A(x) + I_{A^c}(x)]\phi(x)\alpha(dx)$$

so that

$$\left|\int I_A(x)\phi(x)\alpha(dx)\right| = \left|\int I_{A^c}(x)\phi(x)\alpha(dx)\right|$$

and hence

$$\begin{aligned}
|\mu(A) - \nu(A)| &= \left|\int I_A(x)\phi(x)\alpha(dx)\right| \\
&= \frac{1}{2}\left[\left|\int I_A(x)\phi(x)\alpha(dx)\right| + \left|\int I_{A^c}(x)\phi(x)\alpha(dx)\right|\right] \\
&\leq \frac{1}{2}\left[\int I_A(x)|\phi(x)|\alpha(dx) + \int I_{A^c}(x)|\phi(x)|\alpha(dx)\right] \\
&= \frac{1}{2}\int |\phi(x)|\alpha(dx) \\
&= \frac{1}{2}\int |r_1(x) - r_2(x)|\alpha(dx)\ .
\end{aligned}$$

The supremum is achieved with $A = \{x : r_1(x) - r_2(x) > 0\}$. □

Corollary 5.6.2.

$$\|\mu - \nu\| = \sup_{|f|\leq 1} \frac{1}{2}\left|\int f(x)\mu(dx) - \int f(x)\nu(dx)\right|\ . \tag{5.23}$$

Proof. (Billingsley, 1968, p. 224)

$$\begin{aligned}
\frac{1}{2}\left|\int f(x)\mu(dx) - \int f(x)\nu(dx)\right| &= \frac{1}{2}\left|\int f(x)[r_1(x) - r_2(x)]\alpha(dx)\right| \\
&\leq \frac{1}{2}\int |r_1(x) - r_2(x)|\alpha(dx) \\
&= \|\mu - \nu\|\ .
\end{aligned}$$

The supremum is achieved when $A = \{x : r_1(x) - r_2(x) > 0\}$ and $f = I_A - I_{A^c}$. □

Chapter 6

Practical Markov Chain Monte Carlo

While the title of this chapter has been taken from Geyer's (1992) article, we mean something different. His focus was on estimating the variance of the asymptotic distribution of $\bar{g}_n$. In this chapter we endeavor to cover most of the major topics in how to *do* MCMC in light of the theory covered in the previous chapter.

We are nearly ready to start describing the algorithms for doing real MCMC. But first we need to take care of a few important issues that otherwise will get lost if presented while we are trying to understand the algorithms.

By an *update mechanism* in MCMC we mean a bit of computer code that does something random to the current state of the computer program. We say an update mechanism *preserves* a specified invariant distribution if it changes the marginal probability distribution of the state from the the invariant distribution to the invariant distribution, that is, the specified invariant distribution is unchanged by the update.

Generally, there are many different ways to construct update mechanisms

having a specified invariant distribution. But before we look at any of those ways, we show how to combine different update mechanisms preserving the *same* invariant distribution giving a combined update mechanism that also preserves the same invariant distribution.

The point is that once we learn about procedures for combining updates, we never need to refer to them again. We can concentrate on how "elementary" update mechanisms work.

6.1 Combining Update Mechanisms

6.1.1 Composition

By *composition* of update mechanisms we mean computer code in which one update follows another. If `foo` and `bar` are C functions having one argument, which is a pointer to the state `x`, and both preserve a specified stationary distribution, then so does the combined update

```
foo(x);
bar(x);
```

The reason we call it composition is that it is composition of functions in many ways. This would be obvious if we defined the functions to return a value, which is the pointer to the updated state. Then we could write

```
x = bar(foo(x));
```

which looks exactly like composition of functions.

It is a trivial observation that if each term of a product of kernels preserves the invariant distribution, then so does the product, in mathematical notation

$$\pi P_i = \pi, \qquad \text{for all } i$$

implies

$$\pi P_1 P_2 \ldots P_d = \pi,$$

the proof being that multiplication is associative.

This method of combining elementary update mechanisms is not usually called "composition" in the MCMC literature. To the extent that it has a standard name, it is called *fixed scan*. We do not like that name, because it focuses attention on a type of "scan" rather than on a type of "combining" mechanisms. As we will see, "scan" needlessly restricts the notion. There are many types of combining that aren't one of the traditional "scan" notions.

6.1.2 Simple Mixing

If multiplication of kernels provides one method of combining, perhaps addition provides another? Addition of kernels is well defined and obvious, but does not correspond to any probabilistic operation. The sum of two Markov kernels is not Markov (the sum integrates to two, not one).

By *mixing* of update mechanisms we mean computer code which makes a random choice among update mechanisms. By *simple* mixing we mean a random choice that *does not depend on the state* of the Markov chain.

If `foo` and `bar` are C functions that update the state in place by modifying their argument, which is a pointer to the state `x`, and both preserve a specified stationary distribution, then so does the combined update

```
if (unif_rand() < p)
    foo(x);
else
    bar(x);
```

where `unif_rand()` is a source of uniform random numbers and `p` is a constant between zero and one that does not depend on `x`.

More generally, we can consider a *convex combination* of kernels

$$q_1 P_1 + q_2 P_2 + \cdots + q_d P_d$$

where the P_i are Markov kernels and the q_i are nonnegative real numbers that sum to one (and do not depend on x or A in the kernels), does correspond to a probabilistic operation. This combined update proceeds as follows.

- Choose an index j at random, choosing j with probability q_j.
- Update the state using the mechanism with kernel P_j.

We call this state independent mixing to contrast with state dependent mixing, which we hope to cover later. *State independent* means the q_j do not depend on the state x. That

$$\pi(q_1 P_1 + q_2 P_2 + \cdots + q_d P_d) = \pi$$

is again a trivial matter of algebra (and the fact that the q_i sum to one), and again it is our use of measure-theoretic notation that makes it trivial. It is crucial that the q_i do not depend on x or A. As we shall see, state dependent mixing is rather less trivial.

What we are calling mixing here is more often called "random scan" in the literature, the image being one of making a "scan" over the possible choices in random order. But from our point of view (the "update" point of view), this term is misleading. A state independent mixing update does not do a "scan." Rather it executes the mechanism associated with exactly one P_j (chosen randomly).

6.1.3 Subsampling a Markov Chain

Although rarely thought of as belonging in this section (combining updates), a subsampled Markov chain is just a special case of composition.

Fixed Subsampling

Repeating the same update is a special case of composition. The combined update

```
for (i=0; i<k; i++)
    foo(x);
```

obviously preserves the same stationary distribution that `foo` does. The Markov chain that has this as its update mechanism can also be obtained by taking every k-th element of the chain having `foo` as its update mechanism.

Subsampling a Markov chain gives another Markov chain. If P is a Markov kernel, then P^k is also a Markov kernel, and P^k is clearly the k-fold composition of P with itself. If X_1, X_2, ... is a Markov chain with kernel P, then X_k, X_{2k}, ... is a Markov chain with kernel P^k.

Geyer (1992) also gave a finer analysis. If one considers not only the cost (computing time, memory usage, whatever) of generating the Markov chain, but also the cost of using the samples generated and notices that a subsampled chain will have cost of generation proportional to kn where k is the spacing and n the number of iterates in the subsampled chain, but the cost of using samples will be proportional to n. So subsampling may be effective, but only if the cost of "using" is large compared to the cost of "sampling." The theorems that forbid subsampling mentioned above assume zero cost of "using."

The point is not that one should do such an analysis (the analysis would generally take longer than just going ahead and doing something suboptimal). The point is to be aware of the issue. Before these theorems, subsampling sounded plausible to many MCMC experts. Even if one considers costs, most users' intuitions about subsampling are far too favorable to it. Most users guess that intervals like 100 or 1000 are a good idea when a formal analysis might say something more like 3 or 5 is optimal.

Moreover, subsampling also competes with the method of batch means. If the only point of subsampling is to avoid excessive memory use, then one can batch instead of subsample and get the full accuracy possible without using more memory.

Random Subsampling

Fixed interval subsampling can be "part of the problem not part of the solution" because it can convert an effective sampler into a useless one. This most often happens when the original sampler is periodic, but can also happen when the original sampler is only "nearly" periodic. Subsampling using an interval that is a multiple of the period (or "near" period) can destroy all of the good properties of the sampler (even mere irreducibility).

But subsampling at a *random* interval has no such drawbacks. If we let `k` in the preceding example be a random variable having a distribution that does not depend on the current state (at the time the loop starts), then we are mixing over the various values of `k` and thus using both composition and mixing. For example,

```
k = rgeom(p);
for (i=0; i<k; i++)
    foo(x);
```

where `rgeom(p)` is a source of geometric random numbers and `p` is a constant between zero and one that does not depend on `x`.

Curiously fixed subsampling is widely used in MCMC, probably overused. But random subsampling is almost never used. This is surprising because random subsampling is a major tool of Markov chain theory, used again and again in (Meyn and Tweedie, 1993b, Section 5.5). They call the notion "sampled chains" rather than our "subsampled," but the concept is the same.

Optimal Subsampling

Basically, you don't get a better answer by throwing away data. If the cost of using samples is ignored, optimal subsampling is no subsampling (Geyer, 1992; MacEachern and Berliner, 1994).

That doesn't mean there is no point to subsampling. If one wants a long run but doesn't want that many samples, then subsampling is appropriate. The only point of this section is to warn the reader against the once widespread yet erroneous notion that subsampling can improve accuracy. It can't.

6.2 The Metropolis Update

6.2.1 Algorithm

Given an unnormalized density h with respect to λ, the Metropolis update makes a random change to the state that preserves the distribution having this unnormalized density. Thus, if iterated, it produces a Markov chain with h as the unnormalized density of the equilibrium distribution.

Let q be any function on the product of the state space with itself such that

- $q(x, y) = q(y, x)$ for all x and y,
- $q(x, \cdot\,)$ is a probability density w. r. t. λ for all x, and
- it is possible to simulate random realizations from $q(x, \cdot\,)$ for all x.

The *Metropolis update* of the state moves from a state x to the state x^* according to the following procedure

- [The Proposal] Simulate y from $q(x, \cdot\,)$.

- [The Odds Ratio] Calculate

$$r = \frac{h(y)}{h(x)} \tag{6.1}$$

- [Metropolis Rejection] With probability $\min(r, 1)$ set $x^* = y$, otherwise set $y = x$.

In the last step we say we "accept the proposal" when we set $x^* = y$ and otherwise we say we "reject the proposal."

The update is undefined when $h(x) = 0$, but $h(y) = 0$ is allowed. Such a proposal gives $r = 0$ so we "accept" the proposal with probability zero. If the update is used as the transition probability mechanism of a Markov chain, then so long as $h(x_1) > 0$ each iterate will be well-defined and satisfy $h(x_n) > 0$ as well.

Computer code for the update looks something like this, supposing `rq(x)` simulates a random variate having density $q(x, \cdot)$ and `runif` simulates a uniform $(0, 1)$ random variate

```
y = rq(x);
r = h(y) / h(x);
if (runif() < r)
    x = y;
```

(the value of `x` at the end is what is denoted x^* in the mathematical description).

Note well that when the "Metropolis rejection" step "rejects" the state does not change (we have $x^* = x$). So a Markov chain that is produced by iterating a Metropolis update over and over, has many steps when the state does not change. This is part of preserving π. Any attempt to avoid this "rejection" only ruins the algorithm. The state not changing in "rejection" steps is not a bug, it's a feature. It's what makes the algorithm get the correct stationary distribution with only trivial calculations.

6.2.2 Invariant Distribution for Metropolis

We claim the Metropolis update defines a kernel P that is reversible w. r. t. η which is an unnormalized measure corresponding to the function h. First we define the kernel. Let

$$a(x, y) = \frac{h(y)}{h(x)} \wedge 1$$

be the probability that the "Metropolis rejection" step is "accepted" (executes the assignment $x = y$). Then

$$P(x, A) = \int_A q(x, y)a(x, y)\lambda(dy) + I(x \in A)\left(1 - \int q(x, y)a(x, y)\lambda(dy)\right).$$

This is a little complicated, but sensible if we take it one bit at a time. We can move from x to A by proposing $y \in A$ and accepting (that's the first term) or by having $x \in A$ originally (that's what the $I(x \in A)$ is in there for) and proposing some y that is rejected.

Now

$$\begin{aligned}\iint \eta(dx)P(x, dy)g(x, y) = &\iint h(x)q(x, y)a(x, y)g(x, y)\lambda(dx)\lambda(dy) \\ &+ \int h(x)g(x, x)\left(1 - \int q(x, y)a(x, y)\lambda(dy)\right)\lambda(dx)\end{aligned}$$

and the second term on the right hand side is obviously unchanged if the two arguments of g are swapped (since they are both x). Thus we only need to show that the first term is unchanged if we replace $g(x, y)$ by $g(y, x)$. So

$$\begin{aligned}\iint h(x)q(x, y)a(x, y)g(x, y)\lambda(dx)\lambda(dy) &= \iint h(y)q(y, x)a(y, x)g(y, x)\lambda(dy)\lambda(dx) \\ &= \iint h(y)q(x, y)a(y, x)g(y, x)\lambda(dy)\lambda(dx)\end{aligned}$$

the first inequality being interchange of dummy variables and the other being the symmetry requirement $q(x, y) = q(y, x)$. Thus in order to finish the proof it is enough to show that

$$h(x)a(x, y) = h(y)a(y, x), \qquad \text{for all } x \text{ and all } y. \tag{6.2}$$

To prove that, assume without loss of generality that $h(x) \geq h(y)$. Then

$$a(x, y) = \frac{h(y)}{h(x)} \leq 1 \qquad \text{and} \qquad a(y, x) = 1 \, .$$

Hence

$$h(x)a(x, y) = h(x) \cdot \frac{h(y)}{h(x)} = h(y) = h(y)a(y, x) \, .$$

so (6.2) does indeed hold, and the Metropolis update does indeed preserve η because its kernel is reversible w. r. t. η.

6.2.3 Turning an Update into a Markov Chain

One makes a Markov chain by executing the same random mechanism over and over and letting X_n be the state after the n-th execution. This random mechanism is associated with a transition probability kernel P. The chain starts at X_0 which can have any distribution (the "initial distribution").

> MCMC involves running a Markov chain with transition probability kernel P and invariant distribution π.

That's all there is to it. There's a Markov chain. You run it. And you use averages over the run to estimate properties of π.

Right now, we only know one way to make a kernel P that preserves a specified stationary distribution π, the Metropolis update. So right now our recipe for doing MCMC is to iterate the same Metropolis update over and over.

But we shall soon meet other methods of making what we call *elementary updates*, the "indivisible atoms" of MCMC mechanisms, and we have already met methods of combining elementary updates to make a "combined kernel" P that preserves a specified π. Then everything we said here will apply to these more general update mechanisms. Given π you construct a random mechanism described by a kernel P (elementary or combined) that

preserves π. Then one makes a Markov chain (with invariant distribution π) by executing this random mechanism over and over.

Example 6.2.1. *Suppose the target distribution is Cauchy(θ, σ) so that*

$$h(x) = \left[1 + \left(\frac{x-\theta}{\sigma}\right)^2\right]^{-1} .$$

Suppose we use a $N(x, 1)$ candidate, that is,

$$q(x, y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y-x)^2}$$

so that it is obvious that $q(x, y) = q(y, x)$. The odds ratio is then given by

$$r = \frac{\sigma^2 + (x-\theta)^2}{\sigma^2 + (y-\theta)^2} .$$

Let $\theta = 0$ and $\sigma = 1$. The following R code implements the Metropolis algorithm.

```
set.seed(23)
n<-5e2                    #number of iterations
markov<-c(1,rep(0,n-1))   #initial state of the Markov chain
for(i in 2:n){
  prop<-rnorm(1, mean=markov[i-1])         #get the proposal
  odds.ratio<-(1 +  markov[i-1]^2)/(1 + prop^2)  #calculate the odds ratio
  if (runif(1) < odds.ratio) {markov[i]<-prop}
  else {markov[i]<-markov[i-1]}
}
```

Trace plots of an implementation for this setting are given in Figure 6.2.3 on page 160. Specifically, the top plot shows a run of length 500 while the bottom plot shows a run of length 100. High autocorrelation is apparent in the top plot while in the bottom plot we see that the sampler is frequently stuck at a point for repeated iterations.

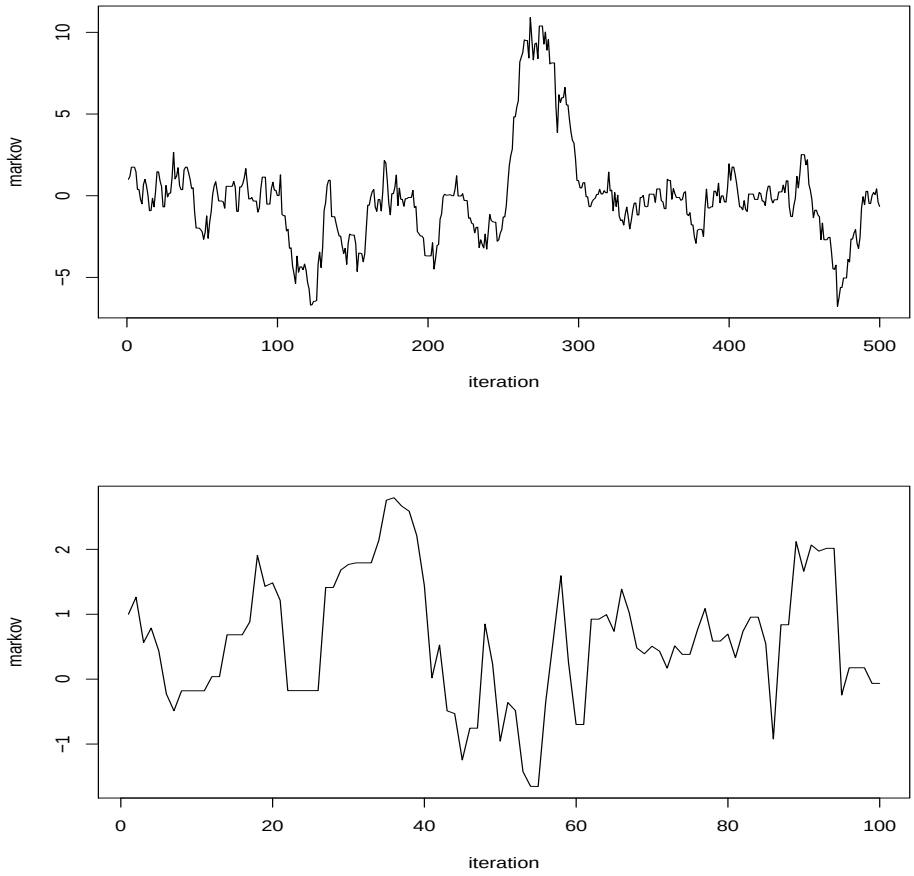

Figure 6.1: One run of the Metropolis algorithm for Example 6.2.1. The top plot shows the initial 500 iterations while the bottom plot shows the initial 100 iterations.

6.2.4 Choosing the Proposal Distribution

The standard example is to make $q(x, \cdot\,)$ be a nondegenerate multivariate normal distribution centered at x. Then

$$q(x, y) = \phi(x - y)$$

where ϕ is multivariate normal centered at zero. Hence q does indeed have the required symmetry property. This is the update used by the `metrop` function in the `mcmc` contributed package for R. The variance-covariance matrix of the normal proposal can be anything so long as it is nonsingular. In order that the Markov chain be time homogeneous, the proposal distribution must not change. We must use the same variance-covariance matrix for all proposals.

Clearly, the particular form of the normal distribution plays no role in the preceding example. We could replace the normal ϕ above by any density having point symmetry about zero

$$\phi(x) = \phi(-x), \text{for all } x.$$

There are not many such symmetric multivariate distributions that can be simulated so as to make suitable proposals. For example, another obviously correct possibility is uniform on any region having a center of symmetry with x being the center. The regions could be balls, ellipsoids, or boxes. It is not clear that any of these are better than normal proposals.

The wonderful feature of the Metropolis algorithm, that *any* proposal works, leaves us with a difficult problem of too many choices. Some proposals will work better than others (will produce more accurate Monte Carlo approximation in the same amount of computer time). Which do we choose?

In general, there is very little one can say. The possible problems that MCMC can be used to solve include every possible probability problem, not to mention every possible integration problem (thus going far beyond probability and statistics). This class of problems is so general, that nothing can be said at this level of generality.

Most discussions in the literature focus on the *acceptance rate* in the Metropolis rejection step (the proportion of Metropolis proposals that are accepted) as a guide. Moreover, they focus on a particular class of proposals (for example, multivariate normal Metropolis proposals). With such simplification of the problem, our choices seem much simpler. How do we adjust the variance matrix of the (multivariate normal) proposal so as to get good performance, and how does the acceptance rate indicate good performance?

It is important to understand that higher acceptance rate is not necessarily good.

- If the unnormalized density h is continuous, then one can always make the acceptance rate as close to one as one pleases by making the proposal variance very very small so $h(y)$ and $h(x)$ are nearly equal and the odds ratio is nearly one.

  But such "baby steps" take a very long time to get anywhere.

- Conversely, consider "giant steps" with very large proposal variance.

  In order for h to be integrable, it must go to zero at infinity, thus if the current position x is from the equilibrium distribution and y is very far from x we will generally have $h(y) \ll h(x)$ and the odds ratio is nearly zero.

The "giant steps are bad" part of the argument is not so clear as the "baby steps are bad" part, but it is clear that we do not want an acceptance rate so low that there are very few acceptances in the entire run of the Markov chain that we are willing to do. It is clear that we don't want an acceptance rate that is either zero or one.

Thus it seems that we have something of a "Goldilocks problem" (we don't want the porridge too hot or too cold as in the children's story of Goldilocks and the three bears). We want an acceptance rate somewhere between zero and one. Surprisingly, it is possible to work out the theoretically optimal

acceptance rate for some very simple problems. Gelman et al. (1996) considered the problem of sampling the multivariate normal distribution (which, of course, does not need MCMC but is simple enough to analyze theoretically) and showed that an acceptance rate of about 20% was right for normal proposal Metropolis (the optimal rate goes to 23.4% and the dimension of the state space goes to infinity).

In a quite different situation Geyer and Thompson (1995) came to a similar conclusion, that a 20% acceptance rate is about right, But they also warned that a 20% acceptance rate could be very wrong and produced an example where a 20% acceptance rate was impossible and attempting to reduce the acceptance rate below 70% would keep the sampler from ever visiting part of the state space.

The 20% magic number must be considered like other rules of thumb we teach in intro courses (like $n > 30$ means means normal approximation is valid). It is not at all clear that the focus on *acceptance rate* as the sole criterion of goodness of proposal makes any sense. Even if one decides to focus on acceptance rate, we have no theory that tells us what acceptance rate to use in general. One should always look at diagnostics such as time series plots as well but there are no guarantees.

Example 6.2.2. *In this example the use of the* `mcmc` *package is illustrated using the setting of Example 6.2.1. The* `metrop` *function requires the log of the unnormalized target density. In this case,*

$$-\log\left(1 + \left(\frac{x-\theta}{\sigma}\right)^2\right) .$$

As in example 6.2.1 let $\theta = 0$ and $\sigma = 1$. Then the following R code implements the Metropolis algorithm and produces the plot in Figure 6.2.4. This code runs the Metropolis algorithm for 3 different settings. In each case it starts from $X_0 = 1$ and uses a Normal proposal distribution but with a different proposal variance; specifically 1/4, 1 and 25. The output shows that

the acceptance rates decrease as the proposal variance increases.

```
h<-function(x){-log(1+x^2)}
library(mcmc)
set.seed(528)
out1<-metrop(h, initial=1, nbatch=500, blen=1, nspac=1, scale=.5)
names(out1)
 [1] "accept"       "batch"        "initial"      "final"        "initial.s
 [6] "final.seed"   "time"         "lud"          "nbatch"       "blen"
[11] "nspac"        "scale"
out1$accept
[1] 0.886
out2<-metrop(h, initial=1, nbatch=500, blen=1, nspac=1, scale=1)
out2$accept
[1] 0.746
out3<-metrop(h, initial=1, nbatch=500, blen=1, nspac=1, scale=5)
out3$accept
[1] 0.382
par(mfrow=c(3,1))
plot(out1$batch[ , 1],type="l")
plot(out2$batch[ , 1],type="l")
plot(out3$batch[ , 1],type="l")
```

6.2.5 Example: Bayesian Logistic Regression

This example is taken from a PhD Qualifying Exam (School of Statistics, University of Minnesota) and is also used in the vignette for the `mcmc` R contributed package. Suppose that for $i = 1, \ldots, 100$ and $j = 0, \ldots, 4$

$$Y_i \mid \beta \sim \text{Bernoulli } (p_i)$$

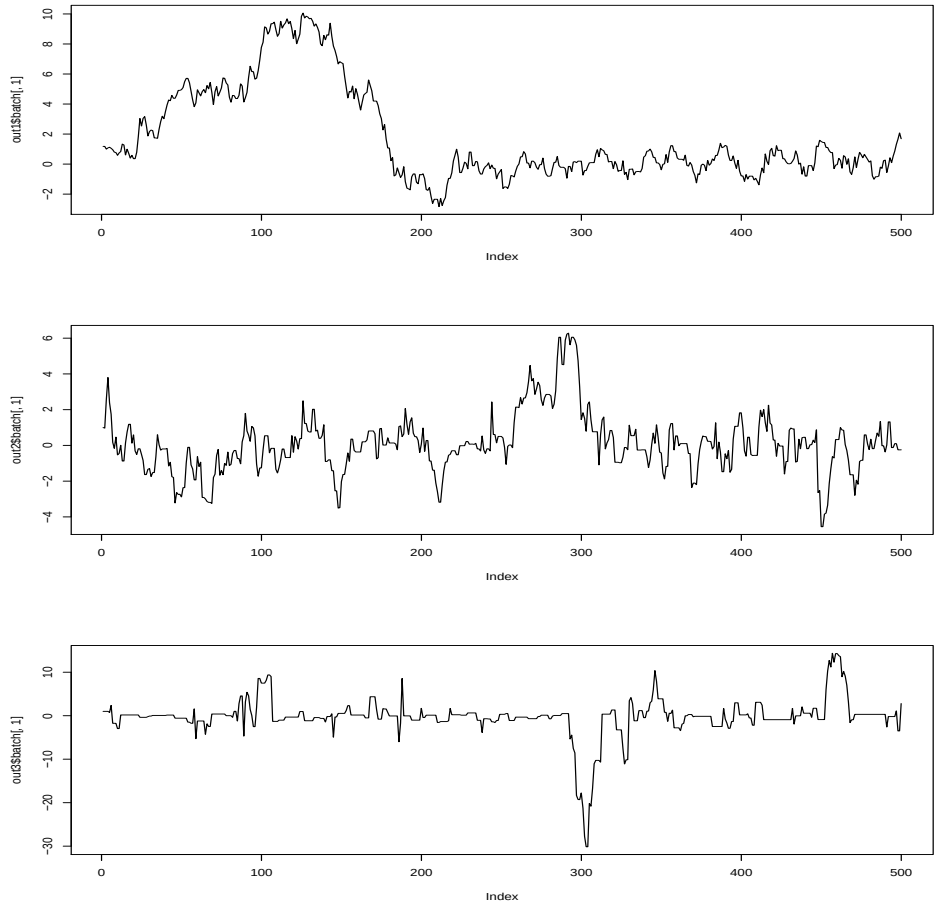

Figure 6.2: Three runs of length 500 of the Metropolis algorithm using the `metrop` R function for Example 6.2.1. The top plot is based on a proposal variance of 1/4, the middle plot on a proposal variance of 1 and the bottom plot on a proposal variance of 25.

where $\beta = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4)^T$,

$$\text{logit}(p_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} = \eta_i$$

and $\beta_j \sim \text{N}(0, 4)$, independently. The posterior is characterized by

$$\pi(\beta \mid y) \propto f(y \mid \beta)\, \pi(\beta)$$

where y is all of the data and hence the log unnormalized posterior is

$$\log[h(\beta \mid y)] = \sum [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] - \frac{1}{8} \sum \beta_j^2$$

where

$$p_i = \frac{e^{\eta_i}}{1 + e^{\eta_i}} \, .$$

Data simulated from this model are given in the file

`http://www.stat.umn.edu/~galin/teaching/8701/logit.txt`

Our goal is to calculate the posterior mean of each of the five regression coefficients.

```
> logit.data<-read.table("logit.txt", header=TRUE)
> out <- glm(y ~ x1 + x2 + x3 + x4, data=logit.data, family=binomial)
> x<-logit.data
> x$y<-NULL
> x<-as.matrix(x)
> x<-cbind(1,x)
> dimnames(x)<-NULL
> y<-logit.data$y
> lupost<-function(beta, x, y){
+ eta<-x %*% beta
+ p<- 1/(1+exp(-eta))
+ logl <- sum(log(p[y==1])) + sum(log(1-p[y==0]))
```

```
+ return(logl + sum(dnorm(beta, 0, 2, log=TRUE)))
+ }
> library(mcmc)
> set.seed(528)
> beta.initial<-as.numeric(coefficients(out))
> out<-metrop(lupost, beta.initial, 1000, x=x, y=y)
> names(out)
 [1] "accept"       "batch"        "initial"      "final"        "initial.seed"
 [6] "final.seed"   "time"         "lud"          "nbatch"       "blen"
[11] "nspac"        "scale"
> out$accept
[1] 0.028
> plot(ts(out$batch),main="scale=1")
```

This acceptance rate is obviously too low. The plot in Figure 6.3 shows this and that little of the space is being explored. Now we try two other values for the scale to achieve a better acceptance rate.

```
> out<-metrop(out, scale = 0.1, x=x, y=y)
> out$accept
[1] 0.729
> plot(ts(out$batch),main="scale=0.1")
> out<-metrop(out, scale = 0.4, x=x, y=y)
> out$accept
[1] 0.238
> plot(ts(out$batch),main="scale=0.4")
```

Using `scale=0.4` results in a reasonable acceptance rate and better exploration (see Figure 6.5) based a pilot run of 1000. Also, the autocorrelation plots in Figure 6.6 show that the all of the autocorrelations are negligible after about lag 20.

```
> acf(out$batch)
```

This means that we should be comfortable using batch sizes of around 25 but lets use batches of length 50 just to be safe.

```
> out<-metrop(out, nbatch=200, blen=50, outfun= function(z,...) z, x=x, y=y
> out$accept
[1] 0.2417
```

Notice that there is an additional argument that gives the functional of the state we want to average. Recall that for this problem we want to estimate the posterior mean. Hence we want to average the state itself.The `outfun` returns `z` for an argument `z`. The ... argument to `outfun` is required since the function is also passed the other arguments (`x` and `y`) to `metrop`.

The batch means are obtained with

```
> post.mean<-apply(out$batch, 2, mean)
> post.mean

[1] 0.6671069 0.7886261 1.1591764 0.4881030 0.7252611
```

These 5 numbers are the Monte Carlo estimates of the posterior means

We still need to calculate Monte Carlo standard errors. We will do this two ways. The first will be with ordinary batch means while in the second case we will use the consistent nonoverlapping batch means (CBM) method of Jones et al. (2005). (We could also use the `olbm` function to do OLBM.) Recall this was described in the previous chapter. Using ordinary batch means is easy for `post.mean`.

```
> post.mean.mcse<-apply(out$batch, 2, sd) / sqrt(out$nbatch)
> post.mean.mcse
```

```
[1] 0.01238445 0.01452696 0.01460955 0.01292095 0.01611571
```

Now lets do the same calculation using consistent version of batch means. An R function, `bm`, to do CBM is included in the Appendix to this chapter.

```
> set.seed(528)
> out<-metrop(out, nbatch=10000, blen=1, outfun= function(z,...) z, x=x, y=y)
> out$accept
[1] 0.2417
> bm(out$batch[,1])

$est
[1] 0.6671069

$se
[1] 0.01256335

$bs
[1] "sqroot"

> bm(out$batch[,2])

$est
[1] 0.7886261

$se
[1] 0.01521510

$bs
[1] "sqroot"
```

```
> bm(out$batch[,3])

$est
[1] 1.159176

$se
[1] 0.01607124

$bs
[1] "sqroot"

> bm(out$batch[,4])
bm(out$batch[,4])
$est
[1] 0.488103

$se
[1] 0.01377448

$bs
[1] "sqroot"

> bm(out$batch[,5])

$est
[1] 0.7252611

$se
[1] 0.01831652
```

```
$bs
[1] "sqroot"
```

The estimates of the posterior means are the same (as they should be) but using the CBM method to calculate the MCSEs results in (slightly) larger MCSEs. This is expected since the theory (see Jones et al., 2005) indicates that this should be the case. Whichever method is used these MCSEs are a little too large (The exam problem asked for MCSEs less than 0.01.) so lets try for some more precision.

```
> out<-metrop(out, nbatch=50000, blen=1, outfun= function(z,...) z, x=x, y=y)
> out$accept
[1] 0.23312
> bm(out$batch[,1])

$est
[1] 0.6647892

$se
[1] 0.005448522

$bs
[1] "sqroot"

> bm(out$batch[,2])

$est
[1] 0.7877401

$se
```

```
[1] 0.007277969

$bs
[1] "sqroot"

> bm(out$batch[,3])

$est
[1] 1.175269

$se
[1] 0.007488702

$bs
[1] "sqroot"

> bm(out$batch[,4])

$est
[1] 0.5208893

$se
[1] 0.00730302

$bs
[1] "sqroot"

> bm(out$batch[,5])

$est
```

```
[1] 0.7195294

$se
[1] 0.008576123

$bs
[1] "sqroot"
```

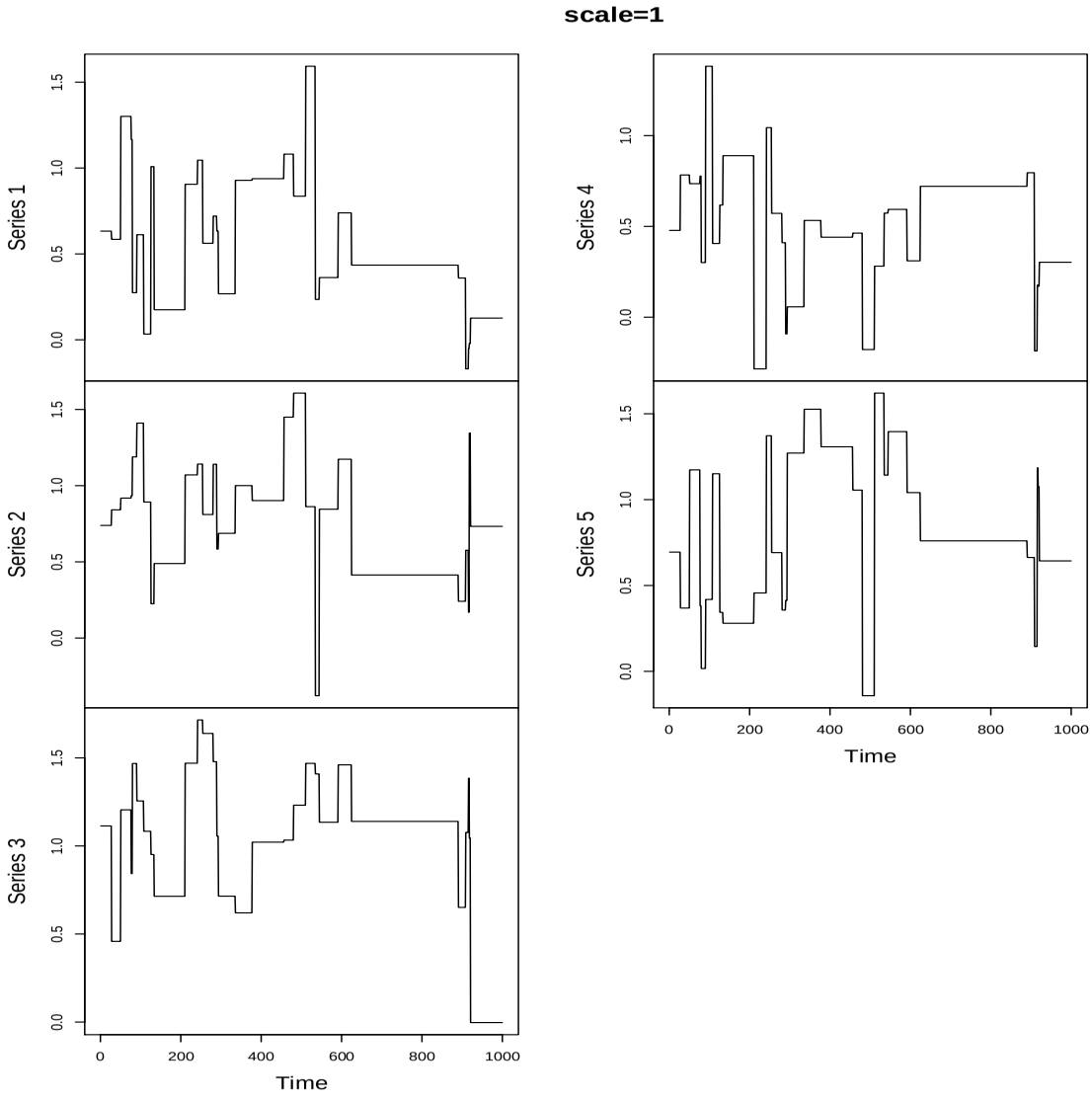

Figure 6.3: Time series plots of MCMC output.

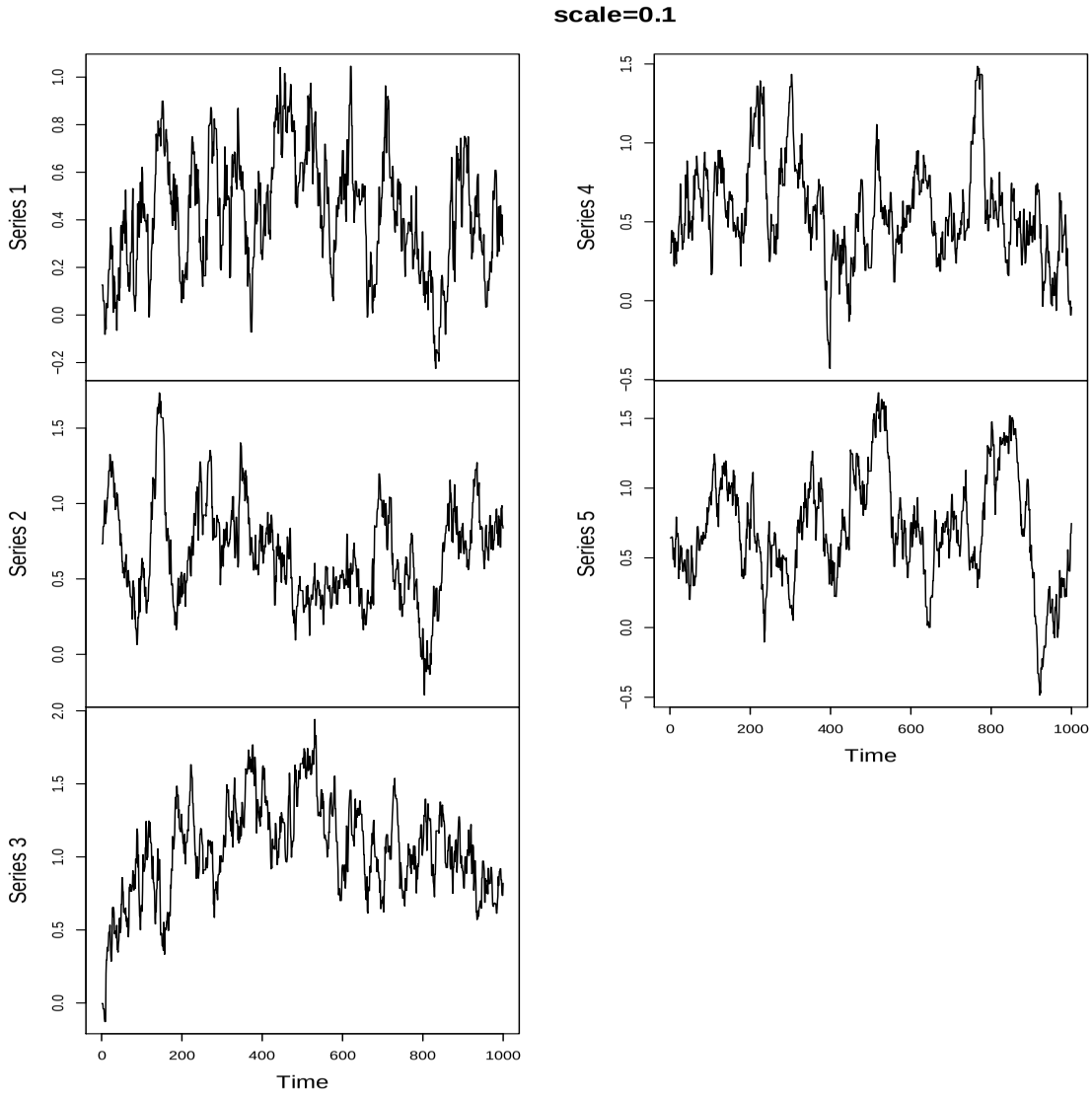

Figure 6.4: Time series plots of MCMC output.

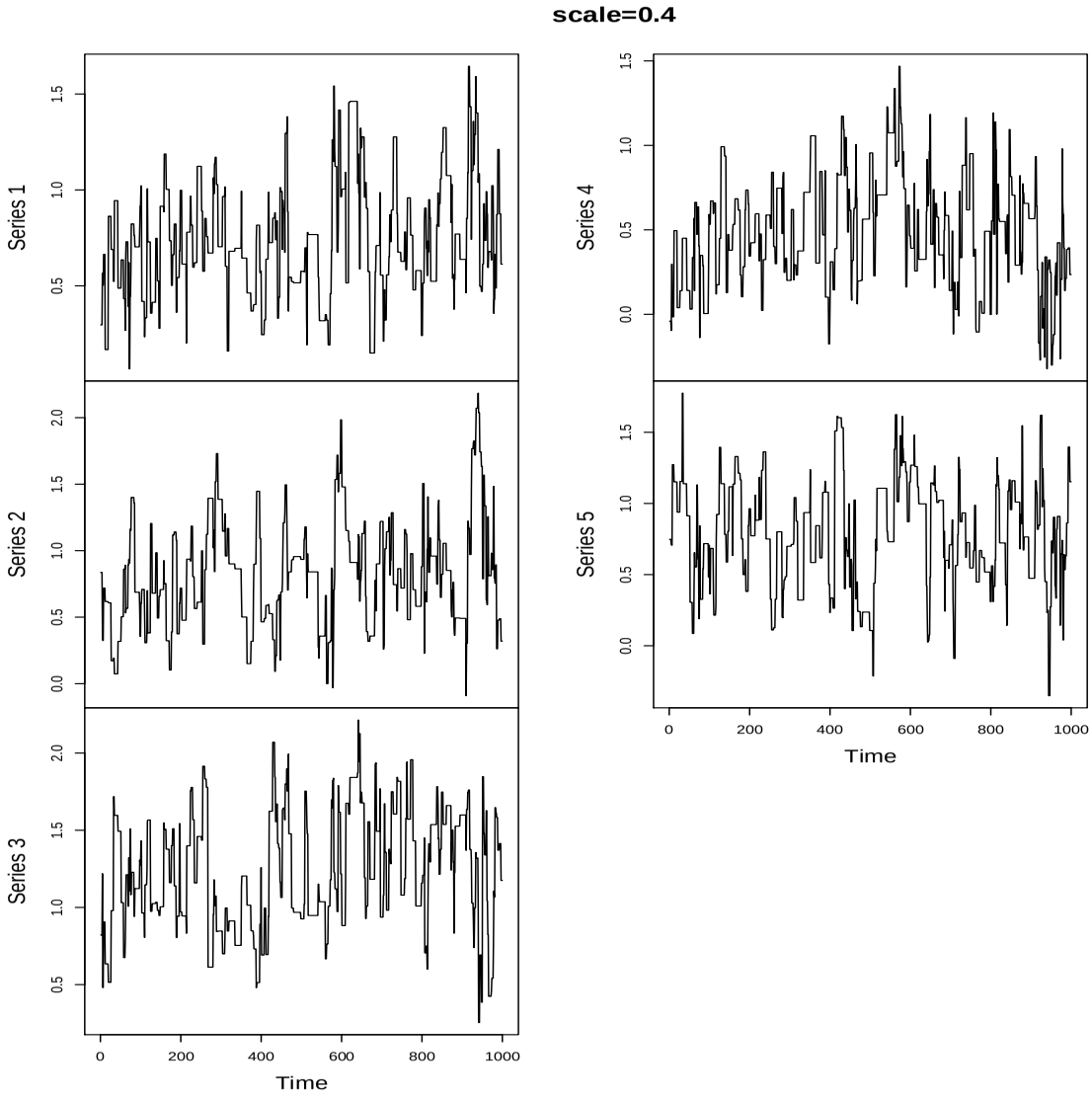

Figure 6.5: Time series plots of MCMC output.

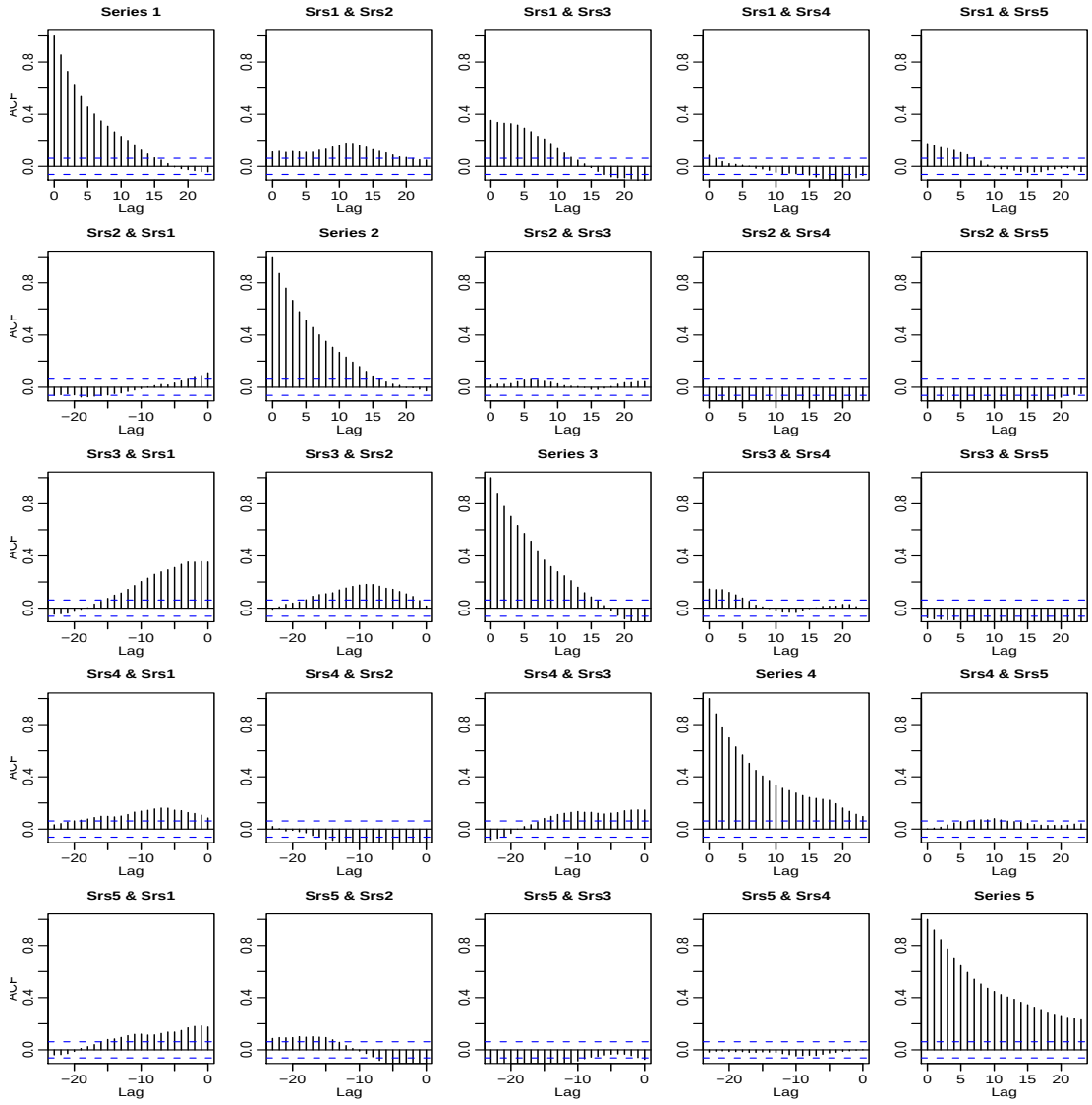

Figure 6.6: Autocorrelation plots of MCMC output.

6.3 The Metropolis-Hastings Update

6.3.1 Algorithm

Hastings (1970) proposed a variant of the Metropolis update which makes the symmetry requirement unnecessary. Everything is the same as described in Section 6.2.1 except that the requirement

- $q(x, y) = q(y, x)$ for all x and y

is *dropped* and replaced by the much weaker

- $q(x, y)$ can be evaluated for all x and y.

Of course, without the symmetry requirement, the algorithm is no longer correct, but Hastings found that the simple change of replacing the Metropolis definition of r in (6.1) by

$$r = \frac{h(y)q(y, x)}{h(x)q(x, y)} \tag{6.3}$$

restores correctness.

Then everything goes through unchanged (use this r in the Metropolis rejection and everything else works the same). The proof in Section 6.2.2 can be altered for this Metropolis-Hastings update, but, since this update is a special case of the more general Metropolis-Hastings-Green update (that we hope to meet later) we shall omit the details.

The requirements on q allow us to be extremely flexible in our choice of proposal distribution. So in some ways we have only complicated matters since some proposals will work better than others. Again we are faced with the question of which one do we choose?

It is basically impossible to give a general recommendation. However, there are a few update recipes that seem to have taken hold in the literature. However, there is no guarantee that any of them will be useful in a particular

problem. If these don't work then try another. The choice of proposal distribution is only limited by our basic requirements and our imagination.

6.3.2 Independence Sampler

The so-called *independence sampler* results when the proposal is chosen independently of the current state. That is, $q(x, y) = q(y)$. Then the odds ratio is

$$r = \frac{h(y)\, q(x)}{h(x)\, q(y)}$$

This update has the property that it either works well or often not at all. This should be surprising since if q doesn't mimic h fairly well then the proposals will be very different from what one would expect from the target. However, this method can occasionally work well in practice and proves to be a continuing source of toy examples used to illustrate complicated theory. In fact, we already met one of these examples when we considered the Markov chain CLT. Here we meet it again.

Example 6.3.1. *Suppose the target distribution is Pareto(α, β) and the proposal distribution is Pareto(α, λ). Then the Hastings ratio is*

$$r = x^{\beta - \lambda} y^{\lambda - \beta} .$$

Thus we can simulate a Markov chain having a Pareto(α, β) invariant distribution as follows. Let the current state be $X_n = x$. Draw $y \sim$ Pareto(α, λ) and independently draw $u \sim$ Uniform$(0, 1)$. Set $X_{n+1} = y$ if

$$u < x^{\beta - \lambda} y^{\lambda - \beta}$$

otherwise set $X_{n+1} = x$.

Mengersen and Tweedie (1996) show that if there exists a $\kappa > 0$ such that

$$\frac{\pi(x)}{q(x)} \le \kappa \quad \forall\, x \in \mathsf{X} \tag{6.4}$$

then the independence sampler having invariant density π and proposal density q is uniformly ergodic and that

$$\|P^n(x,\cdot) - \pi(\cdot)\| \leq \left(1 - \frac{1}{\kappa}\right)^n .$$

Mengersen and Tweedie (1996) also show that if for all $\kappa > 0$ there is a set of positive π measure where (6.4) fails to hold then the chain is not even geometrically ergodic.

Example 6.3.2. *This is a continuation of Example 6.3.1. It is easy to see that if $\lambda \leq \beta$ then this independence sampler is uniformly ergodic since*

$$\frac{\pi(x)}{q(x)} = \frac{\beta}{\lambda}\alpha^{\beta-\lambda}x^{\lambda-\beta} \leq \frac{\beta}{\lambda} .$$

Also, if $\lambda \leq \beta$ and P is the Markov kernel associated with this independence sampler

$$\|P^n(x,\cdot) - \pi(\cdot)\| \leq \left(1 - \frac{\lambda}{\beta}\right)^n .$$

6.3.3 Langevin Update

Grenander and Miller (1994) proposed using a continuous time rather than a discrete time Markov process for simulation. They were not the first to do this, however, one problem with this is that a computer can't do continuous time. One must use a discrete-time approximation. But then one is not actually doing the process one is theorizing about. It turns out that discretizing a continuous time process like this is highly problematic (Roberts and Tweedie, 1996); the convergence properties of the continuous time process need not correspond to those of the discrete time approximation. Fortunately, Besag (1994) in his discussion of Grenander and Miller (1994) pointed out how to fix their algorithm. Simply consider each of their iterates as a mere *proposal* in a Metropolis-Hastings update which must be followed by a Metropolis rejection step.

The Langevin diffusion proposal is multivariate normal but does not have the symmetry property of a Metropolis proposal (which requires the mean be the current state x). It proposes y to be multivariate normal with mean

$$x + \frac{\epsilon}{2} \nabla h(x)$$

and variance-covariance matrix ϵ times the identity. Here ϵ is some "small" number that is the discrete time step length (as $\epsilon \to 0$ we get closer and closer to continuous time) and $\nabla h(x)$ is the gradient (vector of partial derivatives) of h evaluated at the point x.

When we consider this as a Metropolis-Hastings update there is no reason for ϵ to be small; the update is valid for all positive ϵ. As in all Metropolis-Hastings we adjust the "tuning parameter" (here ϵ) so that we get an acceptance rate that is not too large and not too small. There is no reason to make ϵ as small as possible. In fact, this is the worst thing you can do. Making ϵ very small guarantees the algorithm will make only very small steps and have very slow convergence.

Roberts and Rosenthal (1998) show that an acceptance rate of about 50% is optimal for the Langevin diffusion approximation Metropolis-Hastings algorithm, more precisely, they show that for problems in which the equilibrium distribution has IID components that the optimal acceptance rate goes to 57.4% as the dimension of the state space goes to infinity. They also discuss some extensions of their theory to slightly more complicated problems than IID ones, but do not have an extension to completely general equilibrium distributions. Thus, as we saw for simple Metropolis in Section 6.2.4, there is no general theory for setting acceptance rates. Nor is there any general theory that says that *acceptance rates* are the right quantity to look at to adjust the proposal of a Metropolis-Hastings update.

6.4 The Gibbs Update

In this section subscripts denote components of the state vector *not* different variables of a Markov chain.

6.4.1 The Basic Gibbs Update

Given a desired stationary distribution π whose state is a vector $x = (x_1, \ldots, x_d)$, update one variable, say x_j, by giving it a random realization from its conditional distribution given "the rest" $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d)$, this conditional distribution being derived from the joint distribution π.

6.4.2 The Block Gibbs Update

For any subset J of $D = \{1, \ldots, d\}$, let x_J denote the tuple formed from the variables x_j, $j \in J$. A *block Gibbs* update gives x_J a random realization from its conditional distribution given "the rest" $x_{D \setminus J}$, this conditional distribution being derived from the joint distribution π.

6.4.3 The Generalized Gibbs Update

Given any function of the state $g(x)$, a *generalized Gibbs* update gives x a random realization from its conditional distribution given $g(x)$, this conditional distribution being derived from the joint distribution π.

Clearly, a block Gibbs update is the special case obtained when $g(x) = x_J$, and an ordinary Gibbs update is the special case of block Gibbs obtained when $J = \{j\}$. Conversely, generalized Gibbs is the special case of ordinary Gibbs obtained when one does a change of variable so that one of the variables is $g(x)$.

6.4.4 Invariance

Let P be the conditional distribution of X given $g(X)$ for a generalized Gibbs update, and let Q be the marginal distribution of $g(X)$, both marginal and conditional being derived from π. If the current state X has distribution π, then $g(X)$ has distribution Q, and a generalized Gibbs update of the current state has distribution

$$\int Q(dy)P(y, A) = \pi(A)$$

because when we integrate out $y = g(x)$ we get the marginal of the other variable which is the joint distribution π (because the "other" variable is X).

6.4.5 The Gibbs Sampler

The so-called *Gibbs sampler* is an MCMC algorithm using only Gibbs updates. A single Gibbs update does not, by itself, make a good Markov chain. Since (if ordinary) it only changes one variable, it can never sample the equilibrium distribution. One needs to combine the Gibbs updates using any of the combining methods discussed in Section 6.1.

6.4.6 Examples

Toy Example

This example is taken from Jones and Hobert (2001). Let $Y_1, \ldots, Y_m$ be iid $\mathrm{N}(\mu, \theta)$ and let the prior for (μ, θ) be proportional to $1/\sqrt{\theta}$. The posterior density is characterized by

$$\pi(\mu, \theta|y) \propto \theta^{-\frac{m+1}{2}} \exp\left\{ -\frac{1}{2\theta} \sum_{j=1}^{m} (y_j - \mu)^2 \right\} \tag{6.5}$$

where $y = (y_1, \ldots, y_m)^T$. This posterior is proper as long as $m \geq 3$ and we assume this throughout. Using the Gibbs sampler requires the full conditional

densities, $f(\mu|\theta, y)$ and $f(\theta|\mu, y)$, which are as follows:

$$\begin{aligned} \mu|\theta, y &\sim \mathrm{N}(\bar{y}, \theta/m) \,, \\ \theta|\mu, y &\sim \mathrm{IG}\left(\frac{m-1}{2}, \frac{s^2 + m(\bar{y}-\mu)^2}{2}\right) \,, \end{aligned}$$

where $\bar{y}$ is the sample mean and $s^2 = \sum(y_i - \bar{y})^2$. (Note that $W \sim \mathrm{IG}(\alpha, \beta)$ if its density is proportional to $w^{-(\alpha+1)}e^{-\beta/w}I(w > 0)$.)

Consider the Gibbs sampler that updates θ then μ; that is, if we let (θ', μ') denote the current state and (θ, μ) denote the future state, the transition looks like $(\theta', \mu') \to (\theta, \mu') \to (\theta, \mu)$. The state space in this case is $\mathsf{X} = \mathbb{R}^+ \times \mathbb{R}$ and the Markov transition density is

$$k(\theta, \mu|\theta', \mu') = f(\theta|\mu', y)\, f(\mu|\theta, y) \,. \tag{6.6}$$

In other words, the density of the new value (θ, μ) given the current state (θ', μ') is $k(\theta, \mu|\theta', \mu')$. Simulating a random variable from this density can be done sequentially by first taking $\theta \sim f(\theta|\mu', y)$ followed by $\mu \sim f(\mu|\theta, y)$. Jones and Hobert (2001) show that this Gibbs sampler is geometrically ergodic as long as $m \geq 5$.

Benchmark Pump Failure Data

Gaver and O'Muircheartaigh (1987) present a data set concerning the failure rates of 10 pumps at a nuclear power plant, each monitored for different amounts of time. The failure counts for pump i, having been monitored for time t_i, are assumed to follow a Poisson law with a pump-specific mean $t_i\lambda_i$ and observed count y_i. A multilevel model is assumed with $\lambda_i \sim \mathrm{Gamma}(1.802, \beta)$ and $\beta \sim \mathrm{Gamma}(.01, 1)$. (We say $W \sim \mathrm{Gamma}(\alpha, \beta)$ if its density is proportional to $w^{\alpha-1}e^{-\beta w}I(w > 0)$.) Let $\pi(\beta, \lambda|y)$ be the resulting posterior.

A Harris ergodic Gibbs sampler having $\pi(\beta, \lambda|y)$ as its invariant density completes a one-step transition $(\beta', \lambda') \to (\beta, \lambda)$ by simulating $\beta \sim$

Gamma$(18.03, \sum \lambda_i' + 1)$ then each $\lambda_i \sim$ Gamma$(1.802 + y_i, t_i + \beta)$ independently. This Gibbs sampler has been analyzed by many authors including Robert and Casella (1999), Rosenthal (1995), and Tierney (1994).

The following is a (slight) generalization of the above Gibbs sampler. Set $y = (y_1, y_2, \ldots, y_m)^T$ and let $\pi(x, y)$ be a joint density on $\mathbb{R}^{m+1}$ such that the corresponding full conditionals are

$$X|y \sim \text{Gamma}\,(\alpha_1, a + b^T y)$$
$$Y_i|x \sim \text{Gamma}\,(\alpha_{2i}, \beta_i(x))$$

for $i = 1, \ldots, m$, $b = (b_1, \ldots, b_m)^T$ where $a > 0$ and each $b_i > 0$ are known. Since, conditional on x, the order in which the Y_i are updated is irrelevant this is effectively a two variable Gibbs sampler with the transition rule $(x', y') \to (x, y)$. That is we first obtain x conditional on y' then y conditional on x. Jones (2004) shows that this Gibbs sampler is uniformly ergodic if for $i = 1, \ldots, m$ there is a function $g > 0$ such that for all $x > 0$

$$\frac{\beta_i(x)}{b_i x + \beta_i(x)} \geq g(x)\ .$$

Despite this example uniform ergodicity of Gibbs samplers appears to be rare.

Bayesian Inference for the Two-Parameter Normal

Suppose we observe data X_1, ..., X_m iid N(μ, λ^{-1}) and want to make Bayesian inference about the parameters μ and λ. The distribution we want to know about here is the posterior distribution of μ and λ given the data $X = (X_1, \ldots, X_m)$. The posterior depends on the data and on our prior, which we will assume has a probability density function $g(\mu, \lambda)$.

As is well known , there is a closed-form solution to this problem, if we

choose the prior for reasons of mathematical convenience to be of the form[1]

$$\mu \mid \lambda \sim \mathrm{N}\big(\gamma, \delta^{-1}\big), \tag{6.7a}$$

$$\lambda \sim \mathrm{Gamma}(\alpha, \beta) \tag{6.7b}$$

where α, β, γ, and δ are hyperparameters of the prior to be chosen by the user. The likelihood times the prior is proportional to

$$h(\mu, \lambda) = \lambda^{m/2} \exp\left\{ -\frac{m\lambda v_m}{2} - \frac{m\lambda}{2}(\bar{x}_m - \mu)^2 \right\} \lambda^{\alpha-1} e^{-\beta\lambda} \exp\left\{ -\frac{\delta}{2}(\mu - \gamma)^2 \right\}$$

where

$$\bar{x}_m = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{and} \qquad v_m = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x}_m)^2 \,.$$

Using the definition of $h(\mu, \lambda)$ we see that

$$\lambda \mid \mu \sim \mathrm{Gam}\left(\alpha + \frac{m}{2}, \beta + \frac{nv_m}{2} + \frac{m}{2}(\bar{x}_m - \mu)^2\right) \tag{6.8a}$$

$$\mu \mid \lambda \sim \mathrm{N}\left(\frac{m\lambda\bar{x}_m + \delta\gamma}{m\lambda + \delta}, \frac{1}{m\lambda + \delta}\right) \tag{6.8b}$$

So here is the recipe for the Gibbs sampler for this problem. Start anywhere, say at the prior means $\mu_1 = \gamma$ and $\lambda_1 = \alpha/\beta$. Then alternate the update steps. Simulate λ_2 from the distribution (6.8a) with μ_1 plugged in for μ. Then simulate μ_2 from the distribution (6.8b) with λ_2 (the current value) plugged in for λ. Repeat.

- Simulate λ_n from the distribution (6.8a) with μ_{n-1} plugged in for μ.
- Simulate μ_n from the distribution (6.8b) with λ_n plugged in for λ.

[1]The notation $\mathrm{Gamma}(\alpha, \beta)$ here indicates the distribution with density

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \qquad x > 0$$

rather than the other convention which replaces β by $1/\beta$.

This produces a Markov chain (λ_n, μ_n), $n = 1, 2, \ldots$ with state space $\mathbb{R}^2$. Lets look at some R code for implementing this Gibbs sampler.

```
>  alpha <- 1
>  beta <- 20^2
>  gammu <- 50
>  delta <- 1 / 10^2

>  n <- 10
>  xbar <- 41.56876
>  v <- 207.5945

> set.seed(731)
> nsim<-1e3
> mu <- lambda <- rep(NA, nsim)
>  mui <- gammu
> lambdai <- 1 / beta
> for (i in 1:nsim) {
+ lambdai <-  rgamma(1, alpha + n / 2) /
       (beta + n * v / 2 + n * (mui - xbar)^2 / 2)
+ mui <- (n * lambdai * xbar + delta * gammu) / (n * lambdai + delta) +
       rnorm(1) / sqrt(n * lambdai + delta)
+ mu[i] <- mui
+ lambda[i] <- lambdai
+ }
```

There are several ways to look at the simulation output. One is to look at time-series plots of functionals of the chain and the autocorrelation functions.

```
> plot(mu)
> acf(mu)
```

```
> plot(lambda)
> acf(lambda)
```

The time series plots show very little autocorrelation which is also confirmed in Figures 6.9 and 6.10. The reader should be warned that this example is very atypical. Most MCMC time-series plots show much more autocorrelation. This is a very easy Markov chain problem.

Another way to look at the simulation output is a scatter plot of two functionals of the chain. An example is Figure 6.11, which plots μ_n versus $\sigma_n = 1\sqrt{\lambda_n}$.

```
> plot(mu, 1 / sqrt(lambda))
```

In this figure we have lost the time-series aspect. It gives no indication that the sample is from a Markov chain or how much dependence there is in the Markov chain. There is no way to tell, just looking at the figure, whether this is an MCMC sample or an ordinary, independent-sampling sample. This is an important principle of MCMC.

An MCMC scatter plot approximates the distribution of interest, just like a GOFMC scatter plot. This follows from the LLN. Suppose A is any event (some region in the figure). Then the LLN says w.p. 1

$$\frac{1}{n} \sum_{i=1}^{n} I_A(\lambda_n, \mu_n) \to E\{I_A(\lambda, \mu) \mid \text{data}\}$$

Without the symbols, this says the fraction of points in a region A in the figure approximates the posterior probability of that region.

Yet another way to look at the simulation output is a histogram of one functional of the chain.

```
hist(lambda)
```

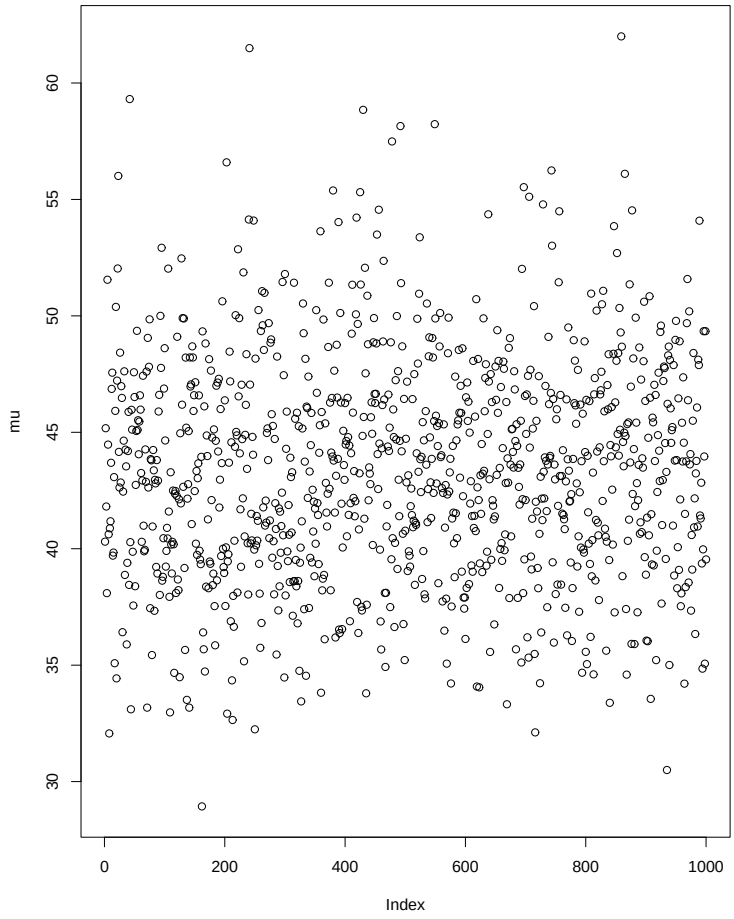

Figure 6.7: Time series plot of Gibbs sampler output for μ in the two-parameter normal model. Sufficient statistics for the data were $\bar{x}_m = 41.56876$, $v_m = 207.5945$, and $n = 10$. Hyperparameters of the prior were $\alpha = 1$, $\beta = 20^2$, $\gamma = 50$, and $\delta = 1/10^2$. The starting point was $\mu = \gamma$ and $\lambda = \alpha/\beta$.

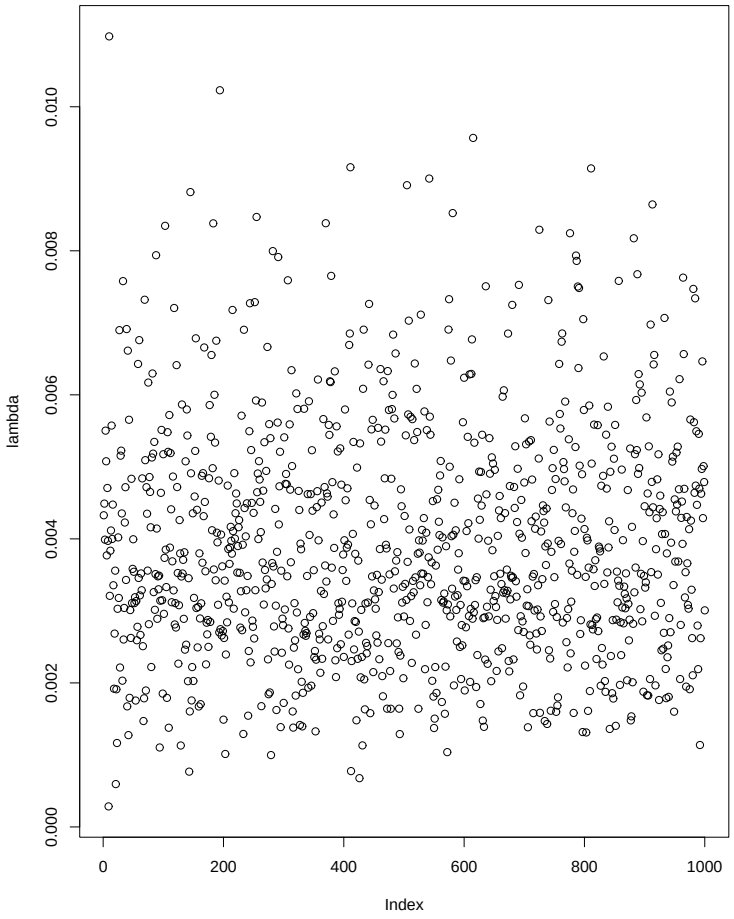

Figure 6.8: Time series plot of Gibbs sampler output for λ in the two-parameter normal model. Sufficient statistics for the data were $\bar{x}_m = 41.56876$, $v_m = 207.5945$, and $n = 10$. Hyperparameters of the prior were $\alpha = 1$, $\beta = 20^2$, $\gamma = 50$, and $\delta = 1/10^2$. The starting point was $\mu = \gamma$ and $\lambda = \alpha/\beta$.

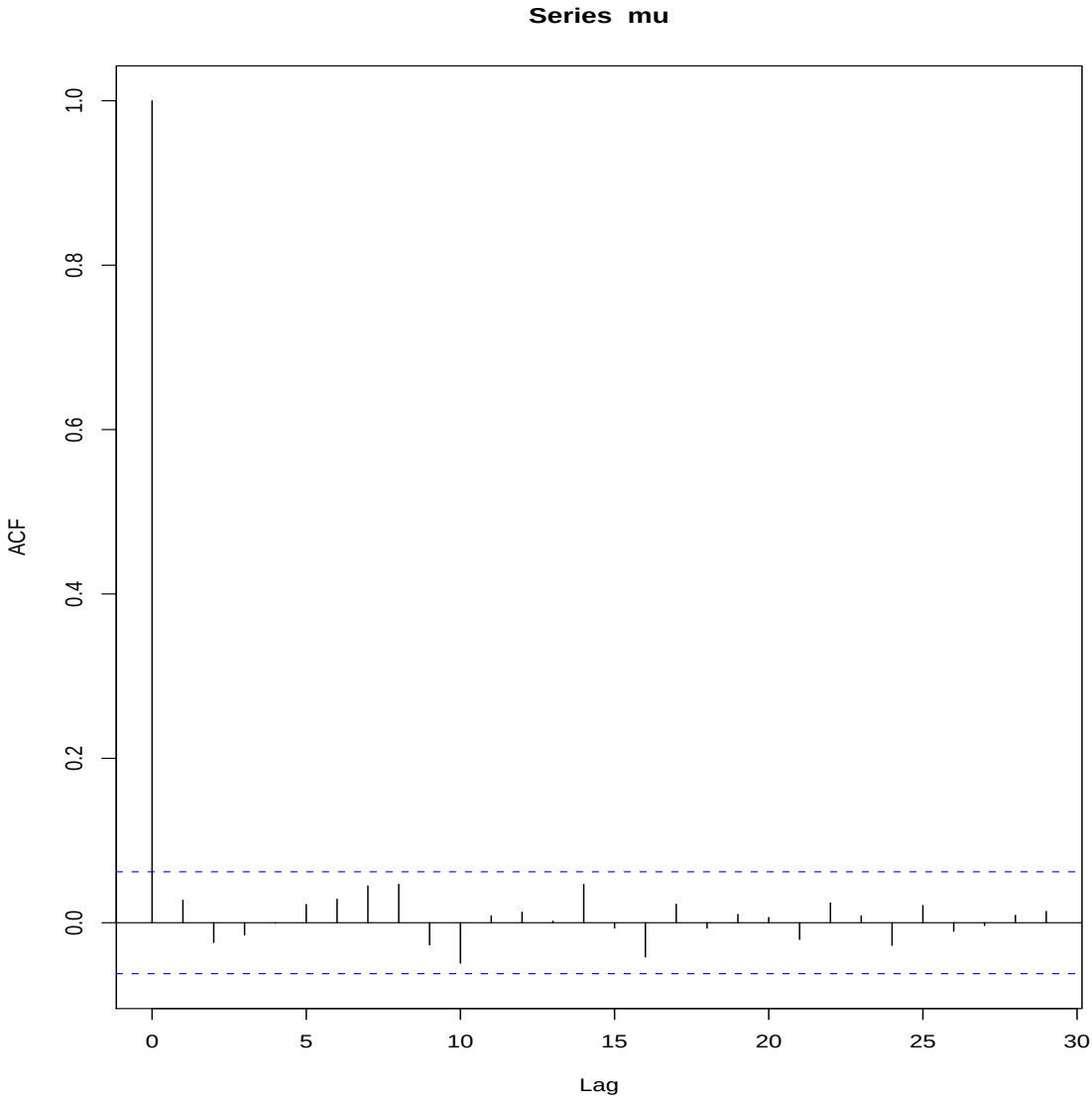

Figure 6.9: Autocorrelation plot of Gibbs sampler output for μ in the two-parameter normal model.

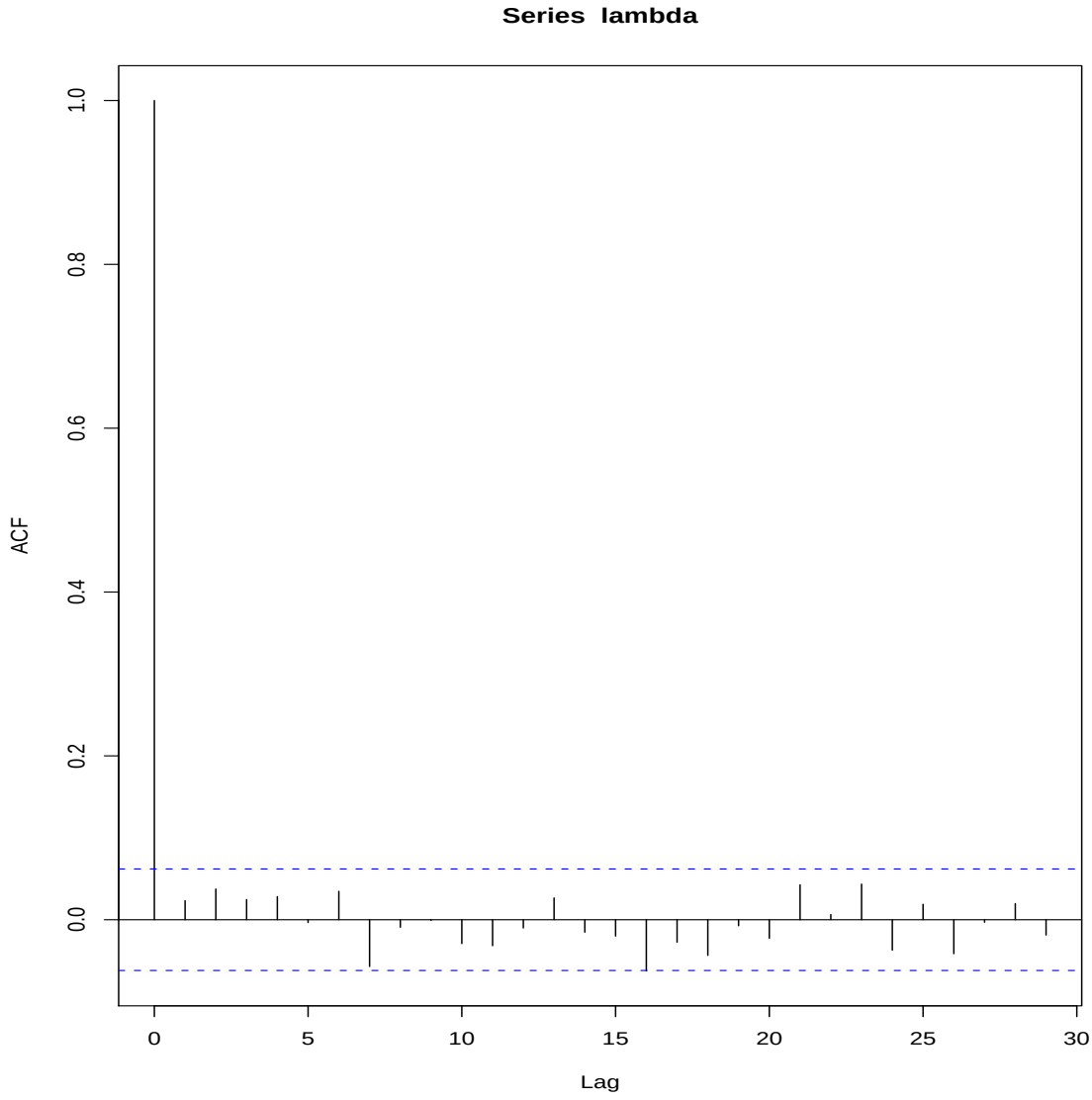

Figure 6.10: Autocorrelation plot of Gibbs sampler output for λ in the two-parameter normal model.

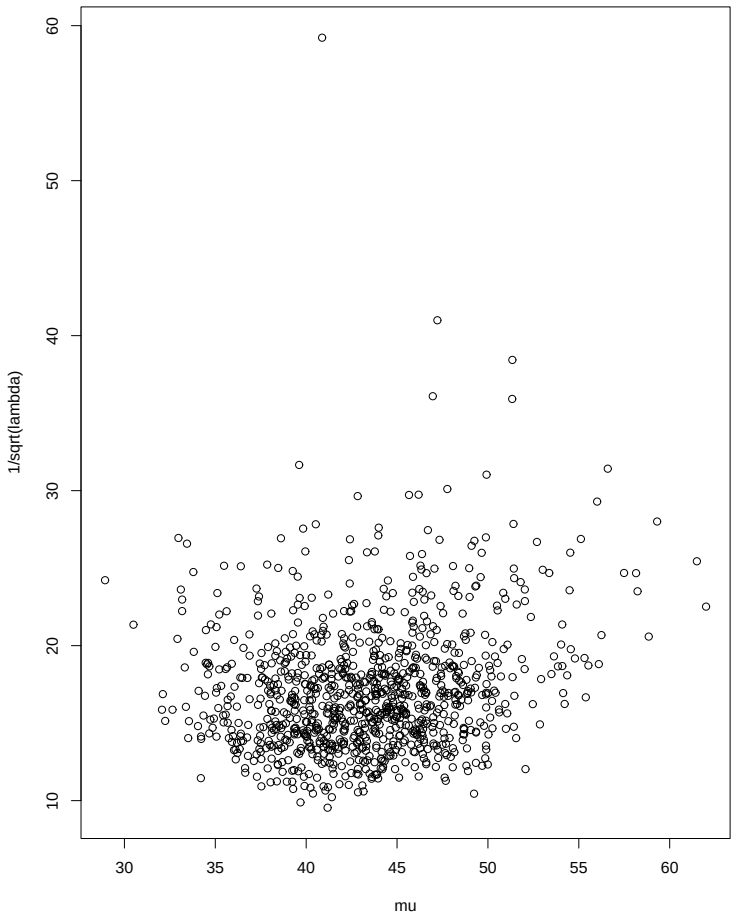

Figure 6.11: Scatter plot of Gibbs sampler output for μ and $\sigma = 1/\sqrt{\lambda}$ in the two-parameter normal model, the same run as shown in Figure 6.7.

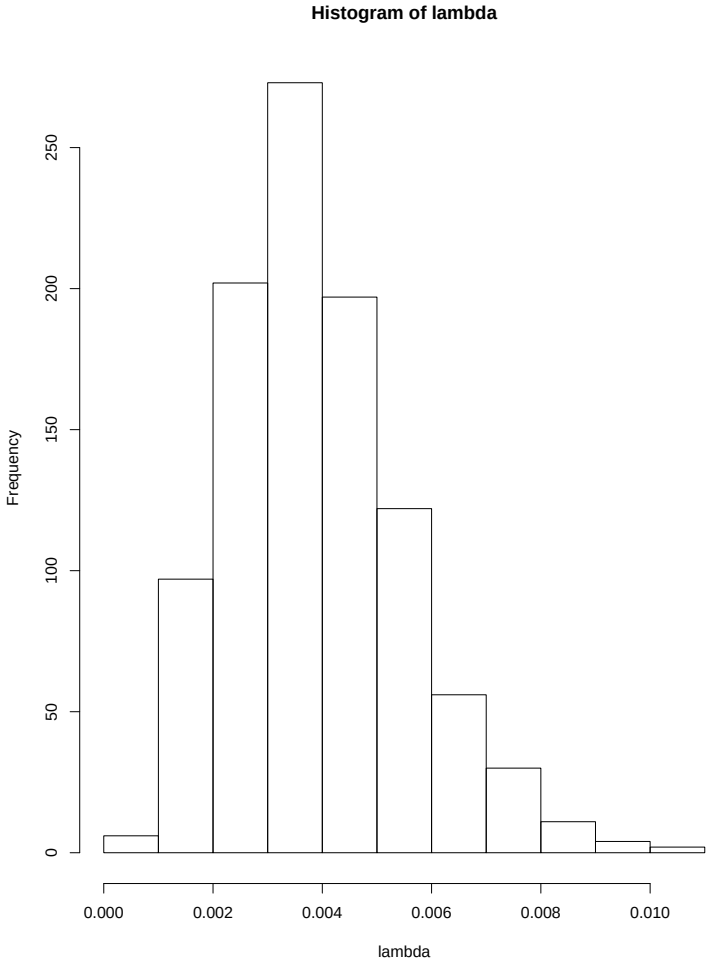

Figure 6.12: Histogram of Gibbs sampler output for λ in the two-parameter normal model, the same run as shown in Figure 6.7. The curve is the estimator of Wei and Tanner (1990) given by (6.9).

An example is Figure 6.12, which plots a histogram of the λ_n. By the LLN again, this is the MCMC approximation of the marginal posterior distribution of λ (same argument as for scatter plots).

A histogram is a limited way to look at a distribution. A clever method due to Wei and Tanner (1990) gives a much better estimate. Consider estimating the distribution of μ. Wei and Tanner's method curiously ignores the simulated values of μ and uses only the simulated values of λ. The distribution of μ given λ is a known normal distribution (6.8b). Denote its density by $f(\mu \mid \lambda, \text{data})$. Let $f_\lambda(\lambda \mid \text{data})$ denote the marginal posterior density of λ (which is not known). The marginal posterior for μ is then given by

$$f_\mu(\mu \mid \text{data}) = \int f(\mu \mid \lambda, \text{data}) f_\lambda(\lambda \mid \text{data}) \, d\lambda.$$

The integrand is the joint posterior of (μ, λ) given the data, so integrating out λ gives the marginal for μ. We cannot easily do the integral analytically, but we can do it by Monte Carlo

$$\hat{f}_{\mu,n}(\mu \mid \text{data}) = \frac{1}{n} \sum_{i=1}^{n} f(\mu \mid \lambda_i, \text{data}) \tag{6.9}$$

where the λ_i are the simulated values from the MCMC run. Note well that (6.9) is to be considered a function of μ. For fixed data and MCMC output $\lambda_1, \ldots, \lambda_n$, we vary μ obtaining the smooth curve in Figure 6.13. Clearly the smooth curve is a much better estimate of the marginal posterior than the histogram. It is also much better than the histogram smoothed using standard methods of density estimation, such as kernel smoothing.

```
> mumu <- pretty(mu)
> mumu <- seq(min(mumu), max(mumu), 0.2)
> dmumu <- rep(0, length(mumu))
> for (i in 1:nsim) {
+ dmumu <- dmumu + dnorm(mumu, (n * lambda[i] * xbar + delta * gammu) /
```

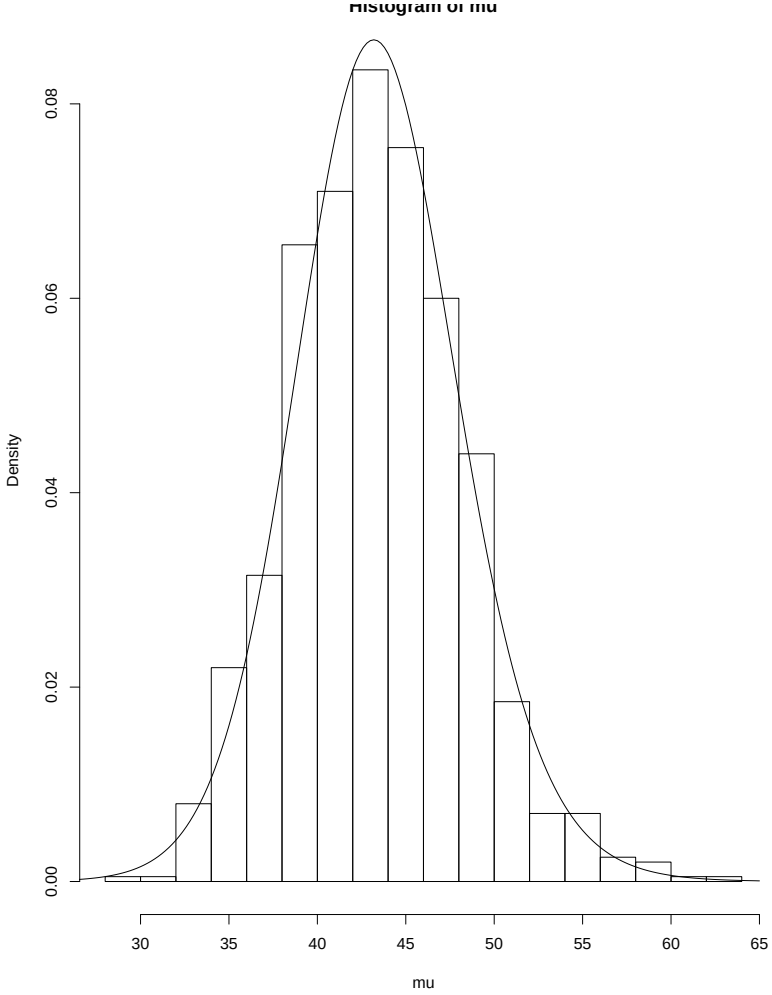

Figure 6.13: Histogram of Gibbs sampler output for μ in the two-parameter normal model, the same run as shown in Figure 6.7. The curve is the estimator of Wei and Tanner (1990) given by (6.9).

```
       (n * lambda[i] + delta), 1 / sqrt(n * lambda[i] + delta))
+  }

> dmumu <- dmumu / nsim
> hist(mu, probability=TRUE, nclass=15, ylim=range(dmumu))
> lines(mumu, dmumu)
```

We can also get a highest posterior density (HPD) region for μ. An HPD region is a level set of the posterior density, in this case a set of the form

$$A_c = \{\, \mu : f_\mu(\mu \mid \text{data}) \geq c \,\}$$

for some constant c, which is chosen to give a desired posterior coverage, e. g., a 95% HPD region chooses c so that $P(\mu \in A_c \mid \text{data}) = 0.95$. For any event A, the LLN says that this probability is approximated by

$$P(\mu \in A \mid \text{data}) \approx \frac{1}{n} \sum_{i=1}^{n} I_A(\mu_i)$$

So a region A will have 95% coverage, as estimated by MCMC, if it contains 95% of the points μ_1, ..., μ_n. It will be a HPD region if it has the property that $f_\mu(\mu \mid \text{data})$ is larger for any $\mu \in A$ than for any $\mu \notin A$. Thus we estimate c by the fifth percentile of the n numbers $f_{\mu,n}(\mu_i \mid \text{data})$, $i = 1, \ldots,$ n, and estimate A_c by

$$A_{c,n} = \{\, \mu : f_{\mu,n}(\mu \mid \text{data}) \geq c \,\}$$

Then the MCMC estimate of $P(\mu \in A_{n,c} \mid \text{data})$ is 0.95 by construction, and $A_{c,n}$ approximates the HPD region A_c.

```
> dmu <- rep(0, length(mu))
> for (i in 1:nsim) {
+ dmu <- dmu + dnorm(mu, (n * lambda[i] * xbar + delta * gammu) /
       (n * lambda[i] + delta), 1 / sqrt(n * lambda[i] + delta))
```

```
+  }
> dmu <- dmu / nsim
> quantile(dmu, .95)
       95%
0.08633233
> foo <- spline(mumu, dmumu, n=1001)
> max(foo$x[foo$x < median(mu) & foo$y < quantile(dmu, 0.05)])

[1] 34.2
> min(foo$x[foo$x > median(mu) & foo$y < quantile(dmu, 0.05)])

[1] 52.96
```

Thus, for the run shown in Figure 6.13, the fifth percentile is 0.0863, giving a 95% HPD region $(34.2, 52.96)$.

6.4.7 Variable-at-a-Time Metropolis-Hastings

When the state X is a vector $X = (X_1, \ldots, X_d)$, the Metropolis-Hastings update can be done one variable at a time, just like the Gibbs update. The algorithm is essentially the same as before, although some changes in notation are required because the proposal only changes a single variable and hence the proposal density $q(x, y)$ is not a density with respect to the measure μ on the whole space. (Warning: for the rest of the section, subscripts indicate components of the state vector, not the time index of a Markov chain.)

Suppose μ is a product measure $\mu_1 \times \cdots \times \mu_d$. For a Metropolis-Hastings update of the i-th variable, we need a proposal density $q_i(x, \cdot\,)$ with respect to μ_i. The update then works as follows. The current position is x, and the update changes x to its value at the next iteration.

1. Simulate a random variate y having the density $q_i(x, \cdot\,)$. Note that y

has the dimension of x_i not x. Let x_y denote the state with x_i replaced by y

$$x_y = (x_1, \ldots, x_{i-1}, y, x_{i+1} \ldots x_d).$$

2. Evaluate the Hastings ratio

$$r = \frac{h(x_y) q_i(x_y, x_i)}{h(x) q_i(x, y)}.$$

3. Do Metropolis rejection: with probability $\min(1, r)$ set $x = x_y$.

Note that, as with the original Metropolis-Hastings update, this update also stays in feasible states if started in a feasible state.

It is easy enough to go through the statements and proofs of Section 6.2.2 making the necessary notational changes to obtain the analogous results for one-variable-at-a-time Metropolis-Hastings. But we won't bother at this point.

6.4.8 Why Gibbs is a Special Case of Metropolis-Hastings

Gibbs updates a variable x_i from its conditional distribution given the rest. The unnormalized joint density of all the variables is $h(x) = h(x_1, \ldots, x_d)$. As usual, this is also an unnormalized conditional density of x_i given x_{-i}.

A Gibbs update is a Metropolis-Hastings update in which the proposal density is proportional to $x_i \mapsto h(x_1, \ldots, x_d)$, that is,

$$q_i(x, y) = h(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_d)/c$$

where c is the unknown normalizing constant that makes h a proper conditional probability density. Then using the notation of the preceding section, the Hastings ratio is

$$\frac{h(x_y) q_i(x_y, x_i)}{h(x) q_i(x, y)} = \frac{h(x_y) h(x)}{h(x) h(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_d)} = 1.$$

Thus this Metropolis-Hastings simulates a new value of x_i from its conditional given the rest and always accepts the proposal. It does exactly the same thing as a Gibbs update.

6.5 Doing MCMC

From a "big picture" point of view, MCMC is simple.

1. Construct a Markov chain having a specified stationary distribution.
2. Simulate the Markov chain.
3. Average over the run to get estimates, including Monte Carlo standard errors.

6.5.1 The Fundamental Problem of MCMC

There are a lot of open research questions involving Markov chains and MCMC. But there is only one practical problem. *You can never be sure that MCMC works.* Except, of course, in easy problems for which the correct answer is discoverable by means other than MCMC and you know MCMC works if it agrees with the correct answer.

Because of the dependence in the observed values of the Markov chain, even a large sample can be very unrepresentative of the stationary distribution. All of the sample points may lie in a small subset of the state space. If the dependence is very strong, then samples with very large (Monte Carlo) sample sizes can be very unrepresentative.

The fundamental problem is often referred to as the problem of "nonconvergence" but, strictly speaking, this is a gross misnomer. Finite sequences don't converge, only infinite sequences. Better terminology would be the problem of "unrepresentativeness."

There is a huge literature on the diagnosis of "nonconvergence" (or unrepresentativeness). Most so-called convergence diagnostics use one of the following ideas.

- The Markov chain may not look stationary.
- Starting from a different place may give different results.
- Running longer may give different results.

A different kind of diagnostic uses so-called *perfect sampling*, which uses a Markov chain simulation to produce a single draw from *exactly* the target distribution. Perfect sampling is the subject of much ongoing research but is outside the scope of these notes. As far as I know perfect sampling is the only reliable "convergence diagnostic" in that failure of a perfect sampler "diagnoses nonconvergence" of MCMC using the same Markov chain.

Unfortunately, the current state of the art in perfect sampling is pretty much restricted to toy problems. For most moderately complicated problems no perfect sampler is known. Perfect sampling aside, there are no reliable "convergence diagnostics."

> *Convergence diagnostics may alert you to a problem.*
>
> *Or they may fail to find problems that actually exist.*
>
> *They can* ***never*** *show absence of problems.*

If you are going to do MCMC, you just have to accept the possibility of being fooled. There is no escape.

Fortunately, the situation is not as bad as the preceding discussion makes it sound. Many MCMC problems are easy. It is obvious from the nature of the problem that the extremely strong dependence of "hard" problems is absent. Also, there are many possible MCMC schemes for any given problem. If you are worried about one scheme, you can try a better one. This last

point is usually not thought of as a "convergence diagnostic" but is much better (barring perfect sampling) than all the "convergence diagnostics" in the literature put together. Basically, the recommendation is (1) *if worried, try a better sampler*, and (2) *if still worried, try one better still.* The trouble with this recommendation is that eventually you run out of ideas, time, patience, or all three, and you may still be worried. In that case, you just have to accept the worry since *if the best sampler anyone can devise doesn't work, then nothing will.*

Additional suggested reading at `http://www.stat.umn.edu/~charlie/mcmc/diag.htm`

6.5.2 The "Burn-In" Non-Problem

Folklore makes MCMC seem much harder than the simple view described at the beginning of this section. Folklore and naive intuition say that one needs to run a stationary Markov chain. Otherwise your estimates are biased, and everyone knows bias is a bad thing.

Actually, theory tells us that bias is negligible for large Monte Carlo sample sizes, more precisely, that the influence of the initial distribution is $O_p(n^{-1})$ whereas the Monte Carlo error is $O_p(n^{-1/2})$. Moreover, in practice we *can't* start the Markov chain in the stationary distribution. If we can produce even one sample from the stationary distribution, we can produce many and do GOFMC so there's no need for MCMC. Thus what folklore and naive intuition require is unnecessary and impossible.

So one might think that folklore and intuition about stationarity and unbiasedness would have been dumped so people could get on with business. But folklore and intuition are strange things, often not affected by theory. Many people choose instead to attempt the impossible. Of course, they don't think of it that way. What they think they are doing is the next best thing. They choose to start in something approximating the stationary distribution so they are approximately unbiased.

Their method for starting in approximate stationarity has many names: burn-in; warm-up; throwing away an initial transient. We will call it "burn-in." The idea is this.

- Start somewhere (anywhere?)
- Run m steps, but "throw away" the results

Now the current state of the chain (after m "burn-in" steps) is supposed to be a good starting point, one in which the distribution of the current state X_m is approximately stationary. It is at this point that many people also confound this idea with those of the previous section. After all, how can the sample be representative without being stationary? But this is just confused.

There are at least three things very wrong with the burn-idea. The first is that it is theoretically unnecessary. By this we mean that if the SLLN, CLT and even the Law of the Iterated Logarithm hold for one distribution then they hold for *any* starting distribution. Recall that it is also possible to obtain consistent estimates of the variance of the asymptotic normal distribution from any starting distribution. Second, one generally has no idea how well it works. That is, if we don't know how to start from the invariant distribution then how do we know if we run for 100 or 1000 or 10000 iterations if we are really close to stationarity? The answer is we don't. There are exceptions to this; see Jones and Hobert (2001) for discussion and references. But the theory required to do this is substantial and hence has (so far) only been done for some fairly simple examples. The third objection is that it is a very limited way of selecting an initial distribution. This is ultimately the fatal objection. Burn-in is just limiting. Why only that way of selecting an initial distribution? In most areas of research, we are always striving to find good new ideas. Why is it that on this particular issue, people cling to the old idea? The following slogan is intended to wake those people up.

Burn-in is only one method, and not a particularly good method,

of finding a good starting point.

People woofing about burn-in are worried about a legitimate issue (they just don't have a sensible solution). Our analysis suggesting that the initial distribution is irrelevant, because its influence is $O_p(n^{-1})$ while the Monte Carlo error is $O_p(n^{-1/2})$, is the full story only in asymptopia, where n has gone to infinity. Reality is not asymptopia, so we can't completely ignore the initial distribution. So long as n stays finite (as of course it always must), the initial distribution cannot be completely ignored.

Looking at our AR(1) toy problem, where we have an explicit formula (5.19) for the bias, we see that the bias is indeed $O(n^{-1})$, but we also see that when n is considered fixed there are initial values X_1 so large that the bias is huge. In real life, n, if not exactly fixed—we are always free to run a little longer—does not go to infinity either. There is an upper limit to the amount of time we are willing to wait for answers.

For purposes of discussion consider the Monte Carlo sample size n fixed. Then it is clear that there is always an X_1 large enough so that the bias completely swamps the variance. Thus it is necessary to avoid such bad starting points.

Starting far out in the tail of the stationary distribution is bad.

On the other hand, any point that *isn't* "far out in the tail" is as good a starting point as any other. One way to say this is

Any point you don't mind having in a sample is a good starting point.

And what if you have not a clue as to what your sample should look like? The slogan isn't much good then, but neither is anything else.

If completely clueless, you don't know how long to burn-in either.

6.5.3 Other Methods of Starting

Having beaten up on burn-in enough, it is time to suggest some alternate methods of starting a Markov chain. Before we do, we stress that there is nothing special about these methods. We only claim that they are not bad and can be used instead of burn-in.

Start Where the Last Run Stopped

R saves random number generator seeds in a dataset `.Random.seed` so that the random number generators are used in one continuous stream. The analogous practice in MCMC is to at the end of each run write the *complete state*, all the variables of the program and all the random number generator seeds into a dataset and to use that as the starting point of the next run. The idea is that multiple runs of the Markov chain behave exactly as if they were bits of one run.

In order that this strategy work it is necessary to write the code so that it allows an arbitrary starting point, which is input at the beginning of the run.

There are a great many benefits to this strategy besides having a starting method. One can do the runs in short pieces, so that little is lost when a machine crashes. With only a little bit more work, this strategy can be converted to a "checkpointing" method. If the entire state (including random number generator seeds) is written to disk every five minutes, then no more than five minutes of computer time is ever lost in a crash. All that requires is making the code to write out the state a subroutine that can be called at any time, rather than just at the end of a run.

This method is not guaranteed to find a good starting point. But we can say that if it doesn't, then the Markov chain in question is completely useless. In all the running it ever did, it never got any samples that were

representative of the stationary distribution. Burn-in wouldn't have helped (all the running you ever did wasn't enough burn-in).

On the other hand, if the chain is of any use at all, then this method provides good starting points. If the previous run produced any output representative of the stationary distribution (even a little bit at the end after a long burn-in), then where it stopped is as good a starting point as any. No further burn-in is necessary.

Additional suggested reading at `http://www.stat.umn.edu/~charlie/mcmc/one.html`

Start at a Known Good Point

There are many situations where some feature of the stationary distribution is available.

When the stationary distribution is specified by an unnormalized density, one can if one wants calculate the mode. This may or may not be near the mean or other notion of center of the distribution. So it may not be a reasonable starting point. But if one thinks it is, use it. The main message here is that a little bit of analysis and careful thinking about the problem at hand can be much better than starting at an arbitrary point.

6.5.4 The Multistart Non-Solution

Many people, not trusting Markov chains (and rightly so in any complicated situation) hope that injecting some independence back into the situation may help.

Perhaps several different runs of a Markov chain with independently chosen starting points will give better answers than just one run. They will, if one does the wrong comparison. But here as everywhere else we should always *compare methods that use* ***equal computer time.*** Thus what should be compared is one long run versus several (or many) short runs. The short

runs no longer look good by this standard. If one run of length mn is too short to be trusted, then m runs of length n are way too short to be trusted. The slogan for this is

> *Many short runs isn't MCMC. It's i. i. d. sampling from a slightly fuzzed version of the starting distribution.*

Additional suggested reading at `http://www.stat.umn.edu/~charlie/mcmc/one.html`

6.6 Appendix: R function for CBM

```
## Function to implement consistent Batch Means procedure
## (Jones, Haran, Caffo and Neath, 2006, JASA)
## Galin L. Jones, Murali Haran, Brian S. Caffo, and Ronald Neath,
## "Fixed-Width Output Analysis for Markov Chain Monte Carlo"

## Author: Murali Haran

## A function for computing batch means in R
## input: vals, a vector of N values (from a Markov chain),
## bs=batch size and g, a function
## output: estimate of E(g(x)) and an estimate of the Monte Carlo
## standard error of estimate of E(g(x))
id <- function(x) return(x)  # default: identity function

bm <- function(vals,bs="sqroot",g=id,warn=FALSE)
  {
    N <- length(vals)
    if (N<1000)
      {
        if (warn) # if warning
```

```
      cat("WARNING: too few samples (less than 1000)\n")
    if (N<10)
      return(NA)
  }

if (bs=="sqroot")
  {
    b <- floor(sqrt(N)) # batch size
    a <- floor(N/b) # number of batches
  }
else
  if (bs=="cuberoot")
    {
      b <- floor(N^(1/3)) # batch size
      a <- floor(N/b) # number of batches
    }
else # batch size provided
  {
    stopifnot(is.numeric(bs))
    b <- floor(bs) # batch size
    if (b > 1) # batch size valid
      a <- floor(N/b) # number of batches
    else
      stop("batch size invalid (bs=",bs,")")
  }

Ys <- sapply(1:a,function(k) return(mean(g(vals[((k-1)*b+1):(k*b)]))) )

muhat <- mean(g(Ys))
sigmahatsq <- b*sum((Ys-muhat)^2)/(a-1)
```

```
  bmse <- sqrt(sigmahatsq/N)

  return(list(est=muhat,se=bmse,bs=bs))
}
```

Chapter 7

Advanced Sampling Techniques

7.1 State Independent Mixing

Recall from Section 6.1.2 that any random mixture of update mechanisms preserving the same distribution also preserves the same distribution so long as the mixing probabilities do not depend on the current state. This section gives more details and an example.

Suppose that for each possible value z of a random variable Z, there is an update mechanism corresponding to the Markov kernel P_z, all of which preserve the same stationary distribution. Then the Markov chain that uses the update mechanism corresponding to the kernel

$$P_{\text{mix}}(x, A) = E\{P_Z(x, A)\} = \int P_z(x, A)Q(dz), \tag{7.1}$$

where Q is the probability distribution governing Z, also preserves the same stationary distribution.

More precisely, what is to be shown is that if η is an unnormalized measure proportional to the desired stationary distribution, so that for any event A

and any z

$$\int \eta(dx) P_z(x, A) = \eta(A)$$

(this is the property of P_z preserving η), this implies the same equation with P_z replaced by P_{mix}, that is,

$$\int \eta(dx) \int Q(dz) P_z(x, A) = \eta(A).$$

The proof of this is trivial, a simple consequence of reversing the order of integration (i.e., Fubini's theorem).

$$\begin{aligned}
\int \eta(dx) \int Q(dz) P_z(x, A) &= \iint \eta(dx) Q(dz) P_z(x, A) \\
&= \int Q(dz) \int \eta(dx) P_z(x, A) \\
&= \int Q(dz) \eta(A) \\
&= \eta(A) \int Q(dz) \\
&= \eta(A)
\end{aligned}$$

A real measure theory fan also wants a proof that P_{mix} is actually a kernel, that is,

- $x \mapsto P_{\text{mix}}(x, A)$ is measurable for each A.
- $A \mapsto P_{\text{mix}}(x, A)$ is a measure for each x.

The former is one part of the Fubini theorem and the only nontrivial part of the latter is countable additivity, which follows by the monotone convergence theorem.

7.1.1 The Hit-and-Run Algorithm

An example of general state-independent mixing is the so-called "hit-and-run" algorithm which has the following basic updates. The state space is a

sunset of $\mathbb{R}^d$. The random variables Z involved in the mixing are random directions in $\mathbb{R}^d$. We can think of Z as being a unit vector along a random direction. The hit-and-run algorithm basic update with kernel P_z moves the state in the direction z, that is we move from the current position x to a point $y = x + \lambda z$ for some real λ. In words, a hit-and-run update makes a one-dimensional move (restricted to points along a line) but in a random direction.

There are hit-and-run samplers that make a random choice among Gibbs updates, those that make a random choice among Metropolis updates, and those that make a random choice among Metropolis-Hastings updates. They are all just special cases of state independent mixing of elementary updates.

The original motivation for hit-and-run algorithms seems to have been the poor performance of one-variable-at-a-time algorithms on certain problems. To see what this is all about we compare traditional Gibbs samplers for the uniform distribution on a rectangle with sides parallel to the coordinate axes and a rectangle with sides at 45° angles to the coordinate axes.

Example 7.1.1 (Gibbs Sampling a Uniform Distribution)**.** *Consider a bounded set A in $\mathbb{R}^d$. A conventional Gibbs sampler uses d updates, one for each coordinate. The i-th update updates the i-th coordinate, giving it a new value simulated from its conditional distribution given the rest of the coordinates, which is uniform on some line segment.*

If the region A is a rectangle parallel to the coordinate axes, the sampler produces i. i. d. samples. Starting at the point (x_1, y_1) in Figure 7.1, it simulates a new x value uniformly distributed over its possible range thereby moving to a position uniformly distributed along the horizontal dashed line, say to (x_2, y_1). Then it simulates a new y value uniformly distributed over its possible range thereby moving to a position uniformly distributed along the vertical dashed line, say to (x_2, y_2). This clearly produces a point uniformly distributed in the rectangle and uncorrelated with the previous point.

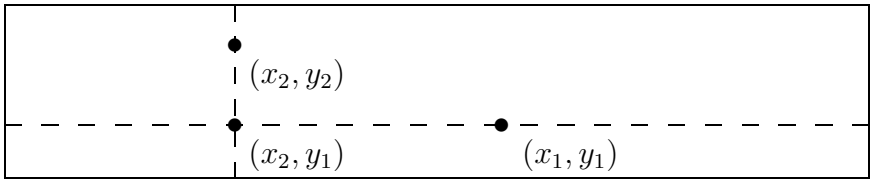

Figure 7.1: Moves of a Gibbs sampler for the uniform distribution on a rectangle with sides parallel to coordinate axes.

If the region A is not a rectangle parallel to the coordinate axes, then the Gibbs sampler has autocorrelation. The update moves are still parallel to the coordinate axes. The possible range of values for each update is the intersection of a horizontal or vertical line, as the case may be, with A. Clearly, starting from the point (x_1, y_1) shown in Figure 7.2, it would take several moves to get into the upper half of the rectangle. Conclusion: the Gibbs sampler for the second rectangle is less efficient.

This example is an important toy problem. What it lacks in realism, it makes up for in simplicity. It is very easy to visualize this Gibbs sampler. Moreover, it does share some of the characteristics of realistic problems.

Example 7.1.2 (Hit-and-Run Sampler for a Uniform Distribution)**.** *The hit-and-run sampler is almost the same as the Gibbs sampler, except that it moves in an arbitrary direction. A hit-and-run step simulates a random angle θ uniformly distributed between 0 and 2π. Then it simulates a new point uniformly distributed along the intersection of A and the line through the current point making angle θ. It is obvious from Figure 7.3 that some hit-and-run update steps move farther than Gibbs update steps. Some hit-and-run steps, not many, only those in a fairly small range of angles, can go from one end of the rectangle to the other. No Gibbs update step can do that.*

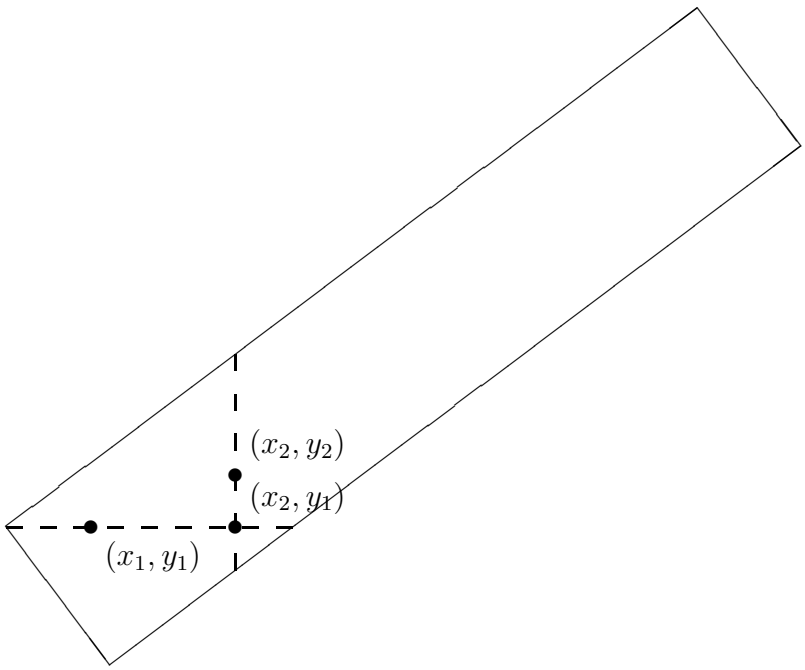

Figure 7.2: Moves of a Gibbs sampler for the uniform distribution on a rectangle with sides not parallel to coordinate axes.

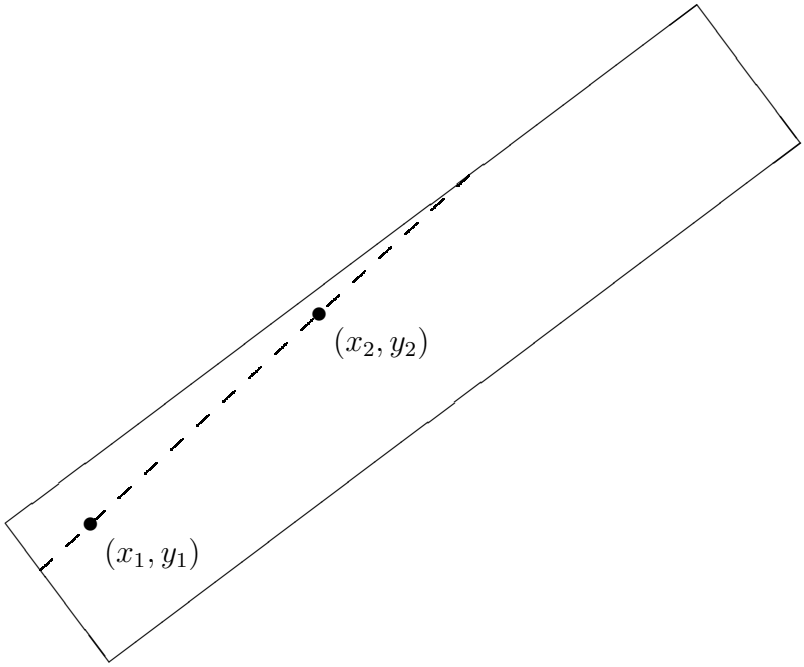

Figure 7.3: Moves of a Hit-and-Run sampler for the uniform distribution on a rectangle.

7.2 The Metropolis-Hastings-Green Algorithm

7.2.1 Radon-Nikodym Derivatives

Suppose μ and ν are two positive measures. We say μ is *dominated by* ν, written $\mu \ll \nu$, if

$$\nu(A) = 0 \quad \text{implies} \quad \mu(A) = 0, \qquad \text{for all measurable sets } A. \tag{7.2a}$$

The Radon-Nikodym theorem says that if μ and ν are both are sigma-finite and $\mu \ll \nu$, then μ has a density with respect to ν, that is, a function f such that

$$\mu(A) = \int_A f(x)\nu(dx), \qquad \text{for all measurable sets } A. \tag{7.2b}$$

By the basic property of integration that integrating over a set of measure zero gives zero the theorem gives a necessary and sufficient condition, that is, not only does (7.2a) imply (7.2b) but also (7.2b) implies (7.2a).

The density f in (7.2b) is also called the *Radon-Nikodym derivative* of μ with respect to ν and is often written

$$f = \frac{d\mu}{d\nu} \tag{7.3a}$$

or

$$f(x) = \frac{d\mu}{d\nu}(x) \tag{7.3b}$$

to indicate explicitly that it is a function of x. Thus in this case (μ dominated by ν) "Radon-Nikodym derivative" is a fancy name for an ordinary concept. Radon-Nikodym derivatives are just densities of one probability distribution with respect to another, the kind of thing explained in Section 4.7.1.

The non-dominated case is a bit trickier. The Lebesgue decomposition theorem, that says for any positive measures μ and ν defined on the same measurable space it is possible to decompose μ into a part singular with respect to ν and and a part dominated by ν. This means there exists a set A

that is a *support* of ν, that is, $\nu(A^c) = 0$, such that the restriction of μ to A has a density with respect to ν, that is, there exists a function f such that

$$\mu(A \cap B) = \int_{A \cap B} f(x)\nu(dx), \qquad \text{for all measurable sets } B.$$

This function f is also called the *Radon-Nikodym derivative* of μ with respect to ν and written (7.3a) or (7.3b), although one might also say in a more long winded way that it is the Radon-Nikodym derivative of the part of μ that is dominated by ν.

All of this seems very technical, but it is easy to calculate any Radon-Nikodym derivatives that arise in practice. Here are some examples.

Example 7.2.1 (Normal Distributions)**.** *Let μ and ν be normal probability measures with means θ_1 and θ_2 and variances σ_1^2 and σ_2^2, respectively. The measures dominate each other because the only sets of probability zero are sets of Lebesgue measure zero, which are the same for both. Thus the Radon-Nikodym derivative is just the ratio of the densities with respect to Lebesgue measure*

$$\frac{d\mu}{d\nu}(x) = \frac{\sigma_1}{\sigma_2} \exp\left(-\frac{(x-\theta_1)^2}{2\sigma_1^2} + \frac{(x-\theta_2)^2}{2\sigma_2^2} \right)$$

Example 7.2.2 (Uniform Distributions)**.** *Let μ and ν be uniform probability measures with supports (a, b) and (c, d). These distributions do not necessarily dominate one another.*

Case I *Clearly $\mu \ll \nu$ if and only if*

$$c \le a < b \le d$$

in which case the Radon-Nikodym derivative is again the ratio of densities

$$\frac{d\mu}{d\nu}(x) = \frac{I_{(a,b)}(x)}{b-a} \cdot \frac{d-c}{I_{(c,d)}(x)} = \begin{cases} \frac{d-c}{b-a}, & a < x < b \\ 0 & c < x < a \text{ or } b < x < d \\ \textit{arbitrary}, & \textit{otherwise} \end{cases}$$

as this indicates, the derivative can be defined arbitrarily off the support of ν, where the ratio of the indicator functions is the indeterminate zero over zero. Usually, one picks the simplest definition

$$\frac{d\mu}{d\nu}(x) = \frac{d-c}{b-a} I_{(a,b)}(x), \qquad x \in \mathbb{R}.$$

Case II *Clearly μ and ν are singular with respect to each other if and only if their supports do not overlap, that is,*

$$b \le c \quad or \quad d \le a$$

in which case the part of μ dominated by ν is the zero measure and the Radon-Nikodym derivative is zero

$$\frac{d\mu}{d\nu}(x) = 0, \qquad for\ all\ x.$$

Case III *What remains to be worked out are three cases of partial overlap. For simplicity, we only look at one*

$$a < c < b < d$$

The support of ν is

$$A = (c, d),$$

and the part of μ that is dominated by ν is the restriction the part that sits on A, which is actually concentrated on (c, b). We denote that measure by $\mu|A$ (the restriction of μ to A). It has density with respect to Lebesgue measure

$$g(x) = \frac{1}{b-a} I_{(c,b)}(x), \qquad x \in \mathbb{R}.$$

Note that this is not a probability density because it integrates to

$$\mu(A) = \frac{b-c}{b-a}$$

which is less than one. The Radon-Nikodym derivative $d\mu/d\nu$ is the density of $\mu|A$ with respect to ν, which is the ratio of these densities

$$\frac{d\mu}{d\nu}(x) = \frac{I_{(c,b)}(x)}{b-a} \cdot \frac{d-c}{I_{(c,d)}(x)} = \begin{cases} \frac{d-c}{b-a}, & c < x < b \\ 0 & b < x < d \\ \textit{arbitrary}, & \textit{otherwise} \end{cases}$$

Again, one usually chooses the simplest formula where the ratio of densities is indeterminate

$$\frac{d\mu}{d\nu}(x) = \frac{d-c}{b-a} I_{(c,b)}(x), \qquad x \in \mathbb{R}.$$

Note that unlike case II this Radon-Nikodym derivative does not integrate to one because it is really a density of $\mu|A$ rather than μ and hence has the mass of $\mu|A$ which is $\mu(A)$.

The other two cases of partial overlap

$$c < a < d < b$$

and

$$a < c < d < b$$

are left as exercises for the reader.

Note that, as a matter of cleaning up notation, all cases can be written

$$\frac{d\mu}{d\nu}(x) = \frac{d-c}{b-a} I_{(a,b)\cap(c,d)}(x), \qquad x \in \mathbb{R}.$$

In fact this formula gives us the idea of a simpler derivation without case splitting. The Radon-Nikodym derivative must be concentrated on the region of overlap of the supports of the two measures $(a, b)\cap(c, d)$, hence the indicator function in the formula. On this (possibly empty, as in case II) interval both densities are finite and nonzero and the Radon-Nikodym derivative is their ratio.

7.2.2 The Elementary Update

The Metropolis-Hastings-Green (MHG) algorithm replaces the densities in the Hastings ratio with one Radon-Nikodym derivative. Here's what we mean. Replace the unnormalized density h with an "unnormalized probability measure" η, that is, η is just a general finite positive measure. It is a constant times the invariant probability measure π we want the update to preserve. Replace the proposal density q with a general transition kernel Q. Recall that for each x in the state space $Q(x, \cdot\,)$ is a probability measure, which is the distribution of the proposal given the current state is x.

With this setup we need to figure out what the Hastings ratio is supposed to be. Let S be the state space. The measure η and kernel Q define a joint measure m on S^2 by

$$m(A) = \iint I_A(x, y)\eta(dx)Q(x, dy) \tag{7.4a}$$

Note that (7.4a) characterizes the joint distribution of the current state X and the *proposal* Y, which may or may not be the next state, depending on whether it is accepted in the "Metropolis rejection" step. To define the analog of the Hastings ratio in the MHG algorithm (the Green ratio) we also need to define the "transpose" or "reverse" of m

$$\begin{aligned} m_R(A) &= \iint I_A(y, x)\eta(dx)Q(x, dy) \\ &= \iint I_A(x, y)\eta(dy)Q(y, dx) \end{aligned} \tag{7.4b}$$

which is the same as the definition of m except for x and y being swapped either in the indicator function or in the measures (it makes no difference, of course, because x and y are dummy variables), that is, m is the joint distribution of the pair (X, Y) and m_R is the joint distribution of the pair (Y, X). The Green ratio is then the Radon-Nikodym derivative

$$R = \frac{dm_R}{dm} \tag{7.4c}$$

It is sometimes written

$$R(x, y) = \frac{\eta(dy)Q(y, dx)}{\eta(dx)Q(x, dy)} \tag{7.4d}$$

which makes it look a lot like (6.3). Despite the familiarity of (7.4d), it means precisely the same thing as (7.4c). Both formulas indicate the same Radon-Nikodym derivative. If that isn't obvious, then the notation in (7.4d) is problematical rather than helpful. The reason (7.4d) is supposed to make sense is that it is supposed to remind you of the "measure" parts of (7.4a) and (7.4b).

Now we can explain the MHG elementary update. Denote the current position by x.

1. Simulate a random variate y having probability measure $Q(x, \cdot)$.
2. Calculate the Green ratio given by (7.4a), (7.4b), and (7.4c)[1]
3. With probability $\min(1, R)$ set $x = y$.

Everything is just the same as with MH except for measures replacing densities. In particular, once the Green ratio is calculated, the "Metropolis rejection" step (3) is exactly the same.

The virtue of the MHG algorithm is that the proposal distributions $Q(x, \cdot)$ and the invariant distribution π are not required to have densities with respect to the same measure. We have already mentioned one instance, so-called one-variable-at-a-time Metropolis-Hastings (Section 6.4.7), where one might want to do this. The following section gives another example.

7.3 State Dependent Mixing

When the distribution of the mixing variable Z in the Section 7.1 depends on the current state, the proof of that section doesn't work. It is not at all

[1] or by (7.4d) if you prefer that notation

obvious that one can do state-dependent mixing.

However there is a way to do state-dependent mixing discovered by Green (1995). Suppose we have a family of update mechanisms corresponding to kernels P_z. We make no assumption about what distribution, if any, these updates preserve. As we shall see, that's not what's needed here. Suppose for each point x in the state space, there is a density q_x with respect to some measure μ that tells us the probabilities of using each P_z. Thus the state-dependent mixture kernel is

$$P_{\text{mix}}(x, A) = E_x\{P_Z(x, A)\} = \int \mu(dz) q_x(z) P_z(x, A). \tag{7.5}$$

Note the difference between (7.1) and (7.5). In (7.1) we just wrote $Q(dz)$ allowing an arbitrary mixing distribution. In (7.5) we write $\mu(dz)q_x(z)$ instead of $Q_x(dz)$. The point is that we need to use a dominated family of mixing distributions for reasons that will become apparent presently.

We need to show that the overall composite kernel (7.5) preserves a particular measure η (the unnormalized distribution of interest). As always we want to use the divide-and-conquer strategy of making simpler checks on simpler objects. That is, we want to make a check for each value of z and have that imply what we want about the composite update.

Now what we need to check is properties of the kernels

$$K_z(x, A) = q_x(z) P_z(x, A) \tag{7.6}$$

that are the integrands in (7.5). Now the interesting issue here is that the K_z are not Markov kernels. They do not correspond to updates of any Markov chain. What we mean by that is they do not give probability one to the whole state space S because

$$K_z(x, S) = q_x(z) P_z(x, S) = q_x(z)$$

which is not one in general. Thus we cannot verify that the K_z preserve η because only Markov kernels preserve distributions. Instead we verify detailed

balance. Unlike the property of preserving a distribution (which only Markov kernels can have), detailed balance with respect to a measure η applies to general kernels. Recall that we say K satisfies detailed balance with respect to η if

$$\iint f(x)g(y)\eta(dx)K(x,dy) = \iint g(x)f(y)\eta(dx)K(x,dy)$$

And this works.

Theorem 7.3.1. *If each K_z defined by* (7.6) *satisfies detailed balance with respect to η, then so does P_{mix} defined by* (7.5)*. Moreover if each P_z in* (7.5) *and* (7.6) *is Markov, then so is P_{mix}.*

Proof. The assertion about detailed balance is just changing the order of integration (the Fubini theorem)

$$\begin{aligned}
\iint f(x)g(y)\eta(dx)P_{\text{mix}}(x,dy) &= \iint f(x)g(y)\eta(dx) \int \mu(dz)q_x(z)P_z(x,dy) \\
&= \iiint f(x)g(y)\eta(dx)\mu(dz)K_z(x,dy) \\
&= \int \mu(dz) \iint f(x)g(y)\eta(dx)K_z(x,dy)
\end{aligned}$$

and by assumption the inner integral on the bottom line is unchanged in value by interchanging f and g, hence so is the integral on the left hand side of the top line. And that proves the detailed balance assertion.

The Markov assertion is even more trivial. If S denotes the whole state space

$$P_{\text{mix}}(x,S) = \int \mu(dz)q_x(z)P_z(x,S) = \int \mu(dz)q_x(z) = 1$$

because each q_x is assumed to be a (normalized) probability density. □

This state dependent mixing is quite simple to set up and fairly obvious in hindsight, although no one before Green (1995) saw it, so it couldn't have been all that obvious.

The only thing that remains to be done is to show how we arrange that a K_z rather than a P_z be reversible with respect to η.

7.4 The Metropolis-Hastings-Green Update Revised

As in Section 7.2.2 the Metropolis-Hastings-Green (MHG) elementary update Green (1995) replaces the densities with Radon-Nikodym derivatives. The only novelty here is that there is state-dependent mixing so we are working with kernels K_z as in the preceding section.

As in Section 7.2.2 the unnormalized probability measure that is proportional to the desired stationary distribution of the Markov chain is denoted by η. Now for each z in some set we have a proposal distribution $Q_z(x, \cdot\,)$, which is the distribution of the proposal given the current state is x. We also have, the mixing densities q_x with respect to some measure μ. Don't get confused between "big Q_z" and "little q_x".

Now the MHG elementary update corresponding to the kernel P_z will be the result of the usual Metropolis rejection applied to some Green ratio R_z (which will be defined presently, for now its exact form is unspecified). As in all our previous descriptions of Metropolis-like updates we denote the current position by x, and the update changes x to its value at the next iteration.

1. Simulate a random variate y having probability measure $Q_z(x, \cdot\,)$.

2. Calculate the Green ratio $R_z(x, y)$.

3. With probability $\min(1, R_z(x, y))$ set $x = y$.

Everything is just the same as in Section 7.2.2 except for the subscripts z.

This update corresponds to the kernel P_z defined by

$$P_z(x, A) = r_z(x)I(x, A) + \int_A a_z(x, y)Q_z(x, dy) \tag{7.7a}$$

where

$$a_z(x, y) = \min(1, R_z(x, y)) \tag{7.7b}$$

and

$$r_z(x) = 1 - \int a_z(x, y)Q_z(x, dy). \tag{7.7c}$$

So the question is now with P_z defined by (7.7a), how do we define the Green ratio R_z so that the kernel K_z defined by (7.6) is reversible with respect to η?

In order to do that we follow Section 7.2.2 in defining joint measures m_z and $m_{R,z}$ on S^2 where S is the state space by

$$m_z(A) = \iint I_A(x, y)\eta(dx)q_x(z)Q_z(x, dy) \tag{7.8a}$$

$$m_{R,z}(A) = \iint I_A(y, x)\eta(dx)q_x(z)Q_z(x, dy) \tag{7.8b}$$

and finally

$$R_z = \frac{dm_{R,z}}{dm_z} \tag{7.9}$$

or if you prefer

$$R_z(x, y) = \frac{\eta(dy)q_y(z)Q_z(y, dx)}{\eta(dx)q_x(z)Q_z(x, dy)} \tag{7.10}$$

7.4.1 Why It Works

What must be shown is that the kernel K_z defined by (7.6) and (7.7a) satisfies detailed balance with respect to η. That is, we must show that

$$\begin{aligned}\iint f(x)g(y)\eta(dx)K_z(x,dy) &= \iint f(x)g(y)\eta(dx)q_x(z)P_z(x,dy) \\ &= \int f(x)g(x)r_z(x)\eta(dx)q_x(z) \\ &\quad + \iint f(x)g(y)a_z(x,y)\eta(dx)q_x(z)Q_z(x,dy)\end{aligned}$$

is unchanged in value if we interchange f and g. This is clearly true of the first term in the final expression above. Thus we only need to work on the second term

$$\iint f(x)g(y)a_z(x,y)\eta(dx)q_x(z)Q_z(x,dy)$$

First note that by definition of the Green ratio (7.9)

$$\begin{aligned}&\iint f(x)g(y)a_z(x,y)\eta(dx)q_x(z)Q_z(x,dy) \\ &\qquad = \iint f(y)g(x)a_z(y,x)\eta(dy)q_y(z)Q_z(y,dx) \\ &\qquad = \iint f(y)g(x)a_z(y,x)R_z(x,y)\eta(dx)q_x(z)Q_z(x,dy)\end{aligned}$$

Thus it is enough to show that

$$a_z(x,y) = a_z(y,x)R_z(x,y), \qquad \text{for almost all } (x,y)\ [m_z], \tag{7.11}$$

because the measure $\eta(dx)q_x(z)Q_z(x,dy)$ we are integrating with respect to is m_z and what happens on a set of measure zero does not change an integral. Equation (7.11) follows from the way Metropolis rejection works and the property of Radon-Nikodym derivatives

$$R_z(y,x) = \frac{1}{R_z(x,y)}, \qquad \text{for almost all } (x,y)\ [m_z], \tag{7.12}$$

which is proved below. There are two cases.

- For almost all x and y such that $R_z(x, y) \geq 1$ we have

$$\begin{aligned} a_z(x, y) &= 1 \\ a_z(y, x) &= R_z(y, x) = \frac{1}{R_z(x, y)} \end{aligned}$$

  and (7.11) holds.

- And for almost all x and y such that $R_z(x, y) \leq 1$ we have

$$\begin{aligned} a_z(x, y) &= R_z(x, y) \\ a_z(y, x) &= 1 \end{aligned}$$

  and again (7.11) holds.

So far the proof was just like the proof of ordinary Metropolis-Hastings.

The only thing that remains is the bit of measure-theoretic business (7.12). To prove that we will use Green's recipe for constructing Green ratios. Let ξ be any symmetric measure on $S \times S$ that dominates both m_z and $m_{R,z}$. (A measure ξ is *symmetric* if $\xi = \xi_R$.) Define

$$f_z(x, y) = \frac{dm_z}{d\xi}.$$

Then

$$f_z(y, x) = \frac{dm_{R,z}}{d\xi}.$$

and

$$R_z(x, y) = \frac{f_z(x, y)}{f_z(y, x)} \tag{7.13}$$

where we take (7.13) to be $+\infty$ if the numerator is nonzero and the denominator is zero and to be 1 if the numerator and denominator are both zero. Allowing the Green ratio to be $+\infty$ causes no problems because we are only interested in values less than one and we define $\min(1, +\infty) = 1$. Note that (7.12) follows immediately from (7.13); it even holds for all x and y if we

consider $1/\infty = 0$ and $1/0 = \infty$. Hence, if we took (7.13) as a definition of the Green ratio (as Green does), there would be nothing further to prove. Since we take the fundamental notion of Radon-Nikodym derivative as the definition of the Green ratio, we have something left to prove: that Green's recipe does actually calculate the Radon-Nikodym derivative.

Define

$$A = \{\, (x, y) : f_z(x, y) = 0 \,\}$$

Then, of course,

$$A_R = \{\, (x, y) : f_z(y, z) = 0 \,\}$$

is the "reverse" set. Note that A^c is a support of m_z. Hence what must be checked is that

$$m_{R,z}(B \cap A^c) = \int_B R_z(x, y) m_z(dx, dy) \tag{7.14}$$

for all measurable sets B (this is the defining property of a Radon-Nikodym derivative in this situation, compare with the displayed equation in the middle of page 218). But (7.14) is an obvious consequence of Green's recipe

$$\begin{aligned} \int_B R_z(x, y) m_z(dx, dy) &= \int_B R_z(x, y) f(x, y) \xi(dx, dy) \\ &= \int_{B \cap A^c} f(y, x) \xi(dx, dy) \\ &= m_{R,z}(B \cap A^c) \end{aligned}$$

And that proves the unstated theorem that the Metropolis-Hastings-Green algorithm actually works.

7.5 Bayesian Model Comparison

7.5.1 The Theory of Bayesian Model Comparison

The Bayesian competitor to frequentist hypothesis testing and model selection involves computing Bayes factors for the various models under consid-

eration. To a Bayesian anything uncertain is random. All lack of knowledge is properly described by probability theory. When you don't know which model is correct, just as everywhere else in Bayesian inference, you put a prior distribution on what is unknown (here on models) and use Bayes rule to calculate posteriors. Philosophically, that's all there is to it. Everything that follows is just turning the mathematical crank.

Suppose we have a family $\mathcal{M}$ of models. In a hypothesis testing situation, $\mathcal{M}$ will have just two models (which a frequentist would call the null and alternative hypotheses, though a Bayesian treats models evenhandedly and needs no such distinguishing terminology). In a model selection situation there may be many models (for example, choosing the right subset of predictors in regression).

The Bayesian starts with a prior distribution h on models, that is a probability distribution

$$h(m), \qquad m \in \mathcal{M}.$$

With each model $m \in \mathcal{M}$ is associated a parameter set Θ_m. In a hypothesis testing situation, these will be the parameter sets Θ_0 and Θ_1 specified by the null and alternative hypotheses. There is also a prior distribution g on model parameters

$$g(\theta \mid m), \qquad \theta \in \Theta_m, \ m \in \mathcal{M}.$$

Note that this distribution is rather weird in that the dimension of the variable θ depends on the value of the conditioning variable m.

Finally, we have the part of the Bayesian model specification that is what a frequentist would call the model specification (a probability distribution for data x given the parameters)

$$f(x \mid \theta, m), \qquad \theta \in \Theta_m, \ m \in \mathcal{M}.$$

The joint distribution of (X, θ, m) is, of course,

$$f(x \mid \theta, m) g(\theta \mid m) h(m), \qquad \theta \in \Theta_m, \ m \in \mathcal{M}.$$

We emphasize again that the dimension of θ changes as m changes, but other than that everything is just the standard Bayesian setup. There are a lot of Bayesian questions that can be asked and answered (any question about the distribution of any or all of the parameters given the data), but here we are only interested in model comparison, and for that we want the posterior probabilities of models given the data

$$p(m \mid x), \qquad m \in \mathcal{M},$$

which are given by Bayes rule as

$$p(m \mid x) = \frac{\displaystyle\int_{\Theta_m} f(x \mid \theta, m) g(\theta \mid m) h(m) \, d\theta}{\displaystyle\sum_{m \in \mathcal{M}} \int_{\Theta_m} f(x \mid \theta, m) g(\theta \mid m) h(m) \, d\theta} \tag{7.15}$$

If you have to chose a model, the one with the highest posterior probability is best. So that's the story on Bayesian model selection except for a few caveats and cautions.

Improper Priors The prior is the product $g(\theta \mid m)h(m)$ and is allowed to be improper, but parts of the prior $g(\theta \mid m)$ for each model aren't allowed to be improper by themselves. If $g(\theta \mid m)$ were improper, it would be unnormalizable, and hence would have no natural "level" meaning that it could be multiplied by an arbitrary constant $c(m)$ without changing its interpretation. But if such arbitrary constants are inserted in (7.15) giving

$$p(m \mid x) = \frac{c(m) \displaystyle\int_{\Theta_m} f(x \mid \theta, m) g(\theta \mid m) h(m) \, d\theta}{\displaystyle\sum_{m \in \mathcal{M}} c(m) \int_{\Theta_m} f(x \mid \theta, m) g(\theta \mid m) h(m) \, d\theta}$$

we get nonsense. The arbitrary constants $c(m)$ do not cancel out, hence the result is arbitrary. Thus the only impropriety that is allowed is in $h(m)$, but

that only makes sense when $\mathcal{M}$ is an infinite set, which is not usually the case.

If the argument given above is unsatisfying, here is another that shows more directly how the arbitrary constants arise. Suppose we wish to use proper but very "diffuse" priors (meaning the priors are almost flat over the region where the likelihood is appreciable). For example we could use priors $g(\theta \mid m)$ that are multivariate normal with mean zero and variance a (large) constant times the identity. Of course the dimension varies with m, say model m has dimension d_m, and with that notation let us write the prior variance for model m as σ_m^2 times the identity. Then

$$g(\theta \mid m) = \left(\frac{1}{\sqrt{2\pi}\sigma_m} \right)^{d_m} \exp(-\|\theta\|^2 / 2\sigma_m^2)$$

Now assume that for each m, the likelihood is actually integrable, that is, we could use a flat prior if we were not doing model comparison,

$$\int_{\Theta_m} f(x \mid \theta, m) \, d\theta < \infty.$$

Then

$$\int_{\Theta_m} f(x \mid \theta, m) \exp(-\|\theta\|^2 / 2\sigma_m^2) \, d\theta \to \int_{\Theta_m} f(x \mid \theta, m) \, d\theta, \qquad \text{as } \sigma_m \to \infty$$

by dominated convergence. For very large σ_m the formula (7.15) becomes

$$p(m \mid x) \approx \frac{(2\pi\sigma_m^2)^{-d_m/2} h(m) \displaystyle\int_{\Theta_m} f(x \mid \theta, m) \, d\theta}{\displaystyle\sum_{m \in \mathcal{M}} (2\pi\sigma_m^2)^{-d_m/2} h(m) \int_{\Theta_m} f(x \mid \theta, m) \, d\theta} .$$

So now we see the source of the arbitrary constants. We can allow the "diffuse" priors to approach flatness in many different ways. By picking different sequences σ_{mn} we can arrange that

$$(2\pi\sigma_{mn}^2)^{-d_m/2} \to c(m), \qquad \text{as } n \to \infty$$

for any constants $c(m)$. Thus, unlike the situation when we are not doing model comparison, the interpretation of an improper prior depends critically on which "diffuse" proper prior it is thought to approximate.

The message to be taken away from this analysis is that the submodel priors $g(\theta \mid m)$ are not allowed to be improper or even "sort of" improper (proper but "diffuse"). The submodel priors cannot be "noninformative" in any sense. They must be highly informative, quite proper, priors encapsulating someone's subjective opinion about possible parameter values in each submodel.

Bayes Factors When choosing a model it is customary to report Bayes factors rather than posterior probabilities. The *Bayes factor* for comparing models m and m' is the ratio of posterior to prior odds

$$\frac{p(m \mid x)}{p(m' \mid x)} \cdot \frac{h(m')}{h(m)} = \frac{\displaystyle\int_{\Theta_m} f(x \mid \theta, m) g(\theta \mid m)\, d\theta}{\displaystyle\int_{\Theta_{m'}} f(x \mid \theta, m') g(\theta \mid m')\, d\theta} \tag{7.16}$$

We sometimes call the numerator on the right hand side the *unnormalized Bayes factor* for model m. Strictly speaking, it is the probability of the data x given the model m (with the parameter θ integrated out). The point is that the Bayes factors themselves are just ratios of "unnormalized Bayes factors."

Why are Bayes factors interesting? Actually it's not clear they are, and many Bayesians consider them bogus, but those that do like them give the following argument. The posterior probability is strongly dependent on the prior probability. If the Bayes factor for comparing models H_0 and H_1 is 100 but the prior odds are 10^{10} in favor of H_1, then the posterior odds are still 10^8 in favor of H_1. This sounds like (and is) very strong odds, but is entirely due to the prior. What the data have to say about the situation actually goes the other way. H_1 is 100 times less likely after observing the data than

before. The Bayes factor focuses on this influence of the data, "factoring out," as it were, the influence of the posterior.

Or, to be precise, we should say "factoring out" a *part* of the influence of the prior because the Bayes factor is influenced by $g(\theta \mid m)$ and $g(\theta \mid m')$ and these are in no sense "factored out." They're still there in (7.16). This is part of what makes Bayes factors controversial. The other part is how one chooses the priors $g(\theta \mid m)$ is any way that looks natural enough to be explained without arousing strong objections in the audience. That we leave aside as a philosophical issue of no interest in a computing course.

Bayes Factors and Improper Priors If $g(\theta|m)$ is itself improper then the prior marginal probability of model m is

$$\int_{\Theta_m} g(\theta \mid m) h(m) \, d\theta = \infty$$

so there is no "prior odds" to use in the definition of Bayes factors. The constants $h(m)$ are not prior probabilities on models unless the $g(\theta \mid m)$ are proper probability densities.

7.5.2 Bayesian Logistic Regression

For a concrete example of Bayesian model comparison, let us consider again Bayesian logistic regression (Section 6.2.5). In that model there were three predictors. Hence there are $2^3 = 8$ different models that can be formed by including or excluding any of these predictors. One, the *full model*, which has all three predictors and four regression coefficients including the intercept, is the one we already analyzed in Example 6.2.5. Another, the *null model* has no predictors and just one regression coefficient, the intercept, and just fits a Bernoulli model to the data (that is, the data Y_i are i. i. d. Ber(p) with p the single unknown parameter). Between these are three models with

one predictor and another three with two predictors. The *model selection problem* is to select the single model that that best fits the observed data.

The parameter spaces for different submodels typically have different dimensions. For our logistic regression example, the parameter spaces have dimensions between one (for the null model) and four (for the full model). In order to distinguish different parameter spaces with the same dimension, we denote them $\mathbb{R}^I$, where I is a subset of $\{0, 1, 2, 3\}$ that contains 0, and are shown in the Figure 7.4.[2] The parameter spaces of the logistic regression model selection problem are partially ordered by embedding, the arrows in the diagram denoting the natural embeddings, which set certain coordinates to zero, for example, the arrow going from $\mathbb{R}^{\{0,2\}}$ to $\mathbb{R}^{\{0,1,2\}}$ represents the embedding $(\beta_0, \beta_2) \mapsto (\beta_0, 0, \beta_2)$.

7.5.3 Priors

As we concluded in our discussion of improper priors, there are no "diffuse" or "noninformative" priors that make sense. It is clear that we want priors centered at zero for the regression coefficients, because anything else biases the choice in favor of particular models, and we don't want to do that. It is also clear that the meaning of a regression coefficient depends on the values of the corresponding predictor that occur in the data. Thus we will standardize all the predictors to have mean zero and variance one (this does not change the family of logistic regression distributions in the model, it

[2]This is set-theoretic notation. For any sets A and B, the symbol A^B denotes the set of all functions from B to A, hence $\mathbb{R}^S$ means the set of all functions from S to $\mathbb{R}$, and an element $\beta \in \mathbb{R}^{\{0,1,3\}}$ is a function from $\{0, 1, 3\}$ to $\mathbb{R}$, which can be specified by giving its values $\beta(0)$, $\beta(1)$ and $\beta(3)$ at the points of the domain. If we write β_i instead of $\beta(i)$ we get the more familiar notation for vectors. An element $\beta \in \mathbb{R}^{\{0,1,3\}}$ represents a 3-vector $(\beta_0, \beta_1, \beta_3)$. Notice the value of the notation. The parameter spaces $\mathbb{R}^{\{0,1,3\}}$ and $\mathbb{R}^{\{0,2,3\}}$ are different. They index different models. If we denoted both of them by $\mathbb{R}^3$, we would not be able to distinguish them.

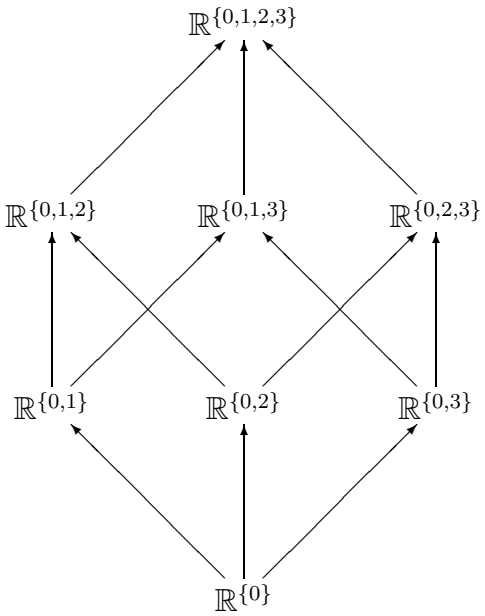

Figure 7.4: Lattice of models for a regression model with three predictors plus intercept.

only reparameterizes each submodel). next we put a standard normal prior on each regression coefficient not constrained to be zero in each submodel. Why variance one? Because, in this artificial situation (i.e., toy problem), we have no idea what would be a sensible variance. But we do know that making the variance really large makes the results meaningless. So we want some "reasonable sized" prior variance. In fact, the standardization of the predictors does not make them the same. If you know anything at all about the data, you may know more about some regression coefficients than others (standardized predictors or no). Hence you should not be using the same prior variance for all predictors.

7.5.4 An MHG Sampler, Try One

The simplest MHG sampler for this problem works as follows.

1. **Staying in one model.** Maybe we could use the "default update" we used in Example 6.2.5). Recall that there we used independent normal proposals for each variable centered at the current value of the variable and all the normal moves having the same standard deviation σ. However, since the dimensions of the models are different these moves should have different step sizes, say σ_m for model m.

2. **Going down one step.** These are moves that go in the reverse direction of the arrows in the figure, dropping one of the variables. The simplest move for these is deterministic. Just delete the variable, leaving the rest alone.

3. **Going up one step.** These are moves that go along one of the arrows in the figure, adding one variable. The simplest move is to propose a step centered at the "current value" in the smaller model, which is zero with the standard deviation say σ_{m,m^*} for the move from model m to model m^*.

We call these moves *lateral*, *down*, and *up*, respectively.

So what are the Green ratios for these steps? The lateral moves we already know how to do. An unnormalized posterior is the likelihood times the prior $L(\beta)g(\beta \mid m)h(m)$. The lateral moves are Metropolis, so the Green ratio is just

$$R = \frac{L(\beta^*)g(\beta^* \mid m)}{L(\beta)g(\beta \mid m)}$$

where β is the current position and β^* is the proposed position.

The up and down moves for the same arrow in the figure must be considered together, since one is the reverse move of the other. For a down move, say from model m to model m^*, the proposal β^* in the parameter space of model m^* is just the current position β in model m with one coordinate set to zero. The move is deterministic, so the $Q(x, dy)$ part of (7.4d) is equal to one (there is no randomness, hence probability one, in the move). The reverse move of this down move is an up move, which proposes a new value for one coordinate, say β_i that was zero in β^*. Since β and β^* agree in all coordinates except the i-th and $\beta_i^* = 0$, we can write $(\beta_i)^2$ as $\|\beta - \beta^*\|^2$, obtaining a notation that does not explictly mention i. The proposal distribution is normal, having density

$$\frac{1}{\sigma_{m^*,m}} \phi\left(\frac{\|\beta^* - \beta\|}{\sigma_{m^*,m}}\right)$$

where ϕ is the standard normal density. This is the $Q(y, dx)$ part of (7.4d). Hence the Green ratio for a down move is

$$R = \frac{L(\beta^*)g(\beta^* \mid m^*)h(m^*)\frac{1}{\sigma_{m^*,m}}\phi\left(\frac{\|\beta^* - \beta\|}{\sigma_{m^*,m}}\right)}{L(\beta)g(\beta \mid m)h(m)}. \tag{7.17}$$

It should be clear that the Green ratio for an up move just changes the roles of up and down, hence the numerator and denominator of (7.17) switch. The Green ratio for an up move from β to β^* is the reciprocal of that for a

down move from β^* to β.

$$\frac{L(\beta^*)g(\beta^* \mid m^*)h(m^*)}{L(\beta)g(\beta \mid m)h(m)\frac{1}{\sigma_{m,m^*}}\phi\left(\frac{\|\beta-\beta^*\|}{\sigma_{m,m^*}}\right)} \tag{7.18}$$

Then each combined update might consist of

- one lateral move, followed by
- one up or down move, every possible move being chosen with equal probability (1/3 in our example, one over the number of predictors in general)

It is important that the up and down moves are balanced so that the probability an arrow is chosen for a move is the same each way (up or down), otherwise the algorithm is incorrect (you are not yet expected to understand why as this will be explained in the next section on "state-dependent mixing").

7.5.5 A Note About Importance Sampling

It is an important fact that

- although the Bayes factors themselves do not depend on the prior probabilities for models $h(m)$, and we are only interested in computing the Bayes factors,
- the behavior of the sampler is critically dependent on the $h(m)$.

  The ratio of time (in the long run) the sampler spends in a model estimates the posterior odds of that model and is the prior odds times the Bayes factor, hence is *proportional to* $h(m)$.
- Bayes factors can be huge numbers, like 10^5 or 10^{-10}.

- Thus, if one uses the flat prior on models, $h(m) \equiv 1$, it can take extremely long runs to estimate posterior probabilities.
- Since the Bayes factors do not depend on the $h(m)$ one is free to choose them for reasons of computational convenience. They do not need to have anything to do with anyone's subjective probabilities.

Our analysis seems to have arrived at a useless conclusion: if you already know the answer, then you also know how to calculate the answer (but if you don't know the answer, then you don't know how to calculate). But things are not as bad as they seem. They just indicate a need for some iteration (trial and error). If the sampler doesn't visit one of the models, increase its prior probability. If the prior probability is increased enough, then it will be visited.

At first sight, the trial and error seems onerous, but there is a trick that helps a lot. Since the only theory we have about MCMC is Markov chain theory, it requires that we run a Markov chain with a specified stationary distribution, but only as the last run we use for final calculations. What we do for trial and error can be anything useful for trial and error. It doesn't even have to be a Markov chain (although presumably it will use most of the code of the MCMC sampler).

In this context, a simple trick for this trial and error is the following.

- After each iteration, reduce the prior on the model that is the current state of the sampler.
- This gives a Markov chain with nonstationary transition probabilities (because the priors keep changing), hence none of the Markov chain theory we know applies.
- But if the changes are small, the sampler won't be too different in behavior from the chain with stationary transition probabilities we will use for final calculations.

- The simplest change is to multiply the prior probability of the current model by $e^{-\lambda}$ for some small positive number λ, in each iteration, or, what is the same, is to subtract λ from the log prior (it makes sense to store the priors as logs to prevent overflow).

7.5.6 Tuning the Sampler

We have to find the tuning parameters for the proposals, σ_m for the lateral proposals and σ_{m,m^*} for the up proposals, and the priors (or log priors) that give even posteriors.

Here is a record of tuning the sampler for the problem described in Example 6.2.5.[3] The initial values of all the tuning parameters (the σ_m and σ_{m,m^*}) equal to 1.0. This is known to be reasonable because we adjusted the predictors so the significant regression coefficients would have order of magnitude 1.0. I also started with a flat prior on models. The table below shows the tuning of the prior.

n	λ	$\log h(m) - \min \log h(m)$							
10^4	10^{-4}	1.00	0.99	0.84	0.53	1.00	0.98	0.69	0.00
10^4	10^{-3}	4.01	3.65	2.30	1.16	4.01	3.50	1.52	0.00
10^5	10^{-3}	8.77	4.80	2.40	1.01	8.69	4.40	1.57	0.00
10^5	10^{-4}	8.79	4.72	2.39	0.98	8.62	4.31	1.58	0.00

Since the log prior did not change much in the last run and the occupation numbers of the models (not shown) were fairly even, we stop here.

The next step is to adjust the other tuning parameters to get acceptance rates of about 20%. The sampler keeps track of acceptance rates for each type of move. The eight different acceptance rates for lateral moves (in each of the eight different models) ranged from 29% in the model with only the constant predictor to 3% in the model with all four predictors. Since the

[3]This was originally performed by Charlie Geyer and later updated by me.

acceptance rate seemed to depend strongly on the dimension of the model we changed the proposal standard deviation for lateral moves from 1.0 to $1/\sqrt{d}$, where d is the model dimension (as we shall see, this didn't work perfectly, but at least goes in the right direction). The 24 different up and down acceptance rates (12 up along the arrows in Figure 7.4 and 12 down in the opposite direction of the arrows) ranged between 44% and 29%. We changed the proposal standard deviation for up moves from 1.0 to 2.0 (the down moves are deterministic with nothing to adjust).

Then we did another run of length 10^5. The acceptance rates for lateral moves now ranged from 31% to 17%, and the acceptance rates for up and down moves ranged from 27% to 19%. We called this good enough. After all, we have no theory which tells us what acceptance rates are optimal for this model.

The occupation numbers for this run were

$\{0\}$	$\{0,1\}$	$\{0,2\}$	$\{0,3\}$	$\{0,1,2\}$	$\{0,1,3\}$	$\{0,2,3\}$	$\{0,1,2,3\}$
11700	11805	12287	12496	12412	13110	12856	13334

And the log priors were

$\{0\}$	$\{0,1\}$	$\{0,2\}$	$\{0,3\}$	$\{0,1,2\}$	$\{0,1,3\}$	$\{0,2,3\}$	$\{0,1,2,3\}$
8.7858	8.6212	2.3912	4.7222	1.5793	4.3068	0.9807	0.0000

(these are unchanged from the end of the adjustment, but look a bit different because the models have been reordered).

The occupation numbers divided by the run length (here 10^5) estimate the posterior probabilities. The occupation numbers divided by the prior give unnormalized Bayes factors. Since unnormalized Bayes factors may be multiplied by an arbitrary constant, we scale them so the model with the largest Bayes factor is 1.00.

model	Bayes factor	$\log_{10}$ Bayes factor
$\{0\}$	0.00013	−3.87240
$\{0, 1\}$	0.00016	−3.79703
$\{0, 2\}$	0.08433	−1.07400
$\{0, 3\}$	0.00834	−2.07901
$\{0, 1, 2\}$	0.19187	−0.71700
$\{0, 1, 3\}$	0.01325	−1.87778
$\{0, 2, 3\}$	0.36160	−0.44177
$\{0, 1, 2, 3\}$	1.00000	0.00000

And we are almost done. The Bayes factors, if we can trust the Monte Carlo calculation, show that the full model has the most support from the data, but two other models, $\{0, 1, 2\}$ and $\{0, 2, 3\}$ also look good. The 3 to 1 odds against $\{0, 2, 3\}$ and the 5 to 1 odds against $\{0, 1, 2\}$ (both compared to the full model) are not very strong evidence against them. Even the 12 to 1 odds against $\{0, 2\}$ are not very strong. So the only firm conclusion from our Bayesian model selection is that predictor 2 must be in the model because without it the odds against are 75 to 1 or worse.

But our Monte Carlo analysis is not done until we produce Monte Carlo standard errors. Here we used the delta method to calculate standard errors for log unnormalized Bayes factors.

First we estimate the mean occupation numbers and their variances and covariances by the method of overlapping batch means with batch size 10^3. Since our estimator for log unnormalized Bayes factor for model i is

$$\log_{10} \hat{\mu}_i - \log_{10} \hat{\mu}_j - \log_{10} h(i) + \log_{10} h(j) \tag{7.19}$$

where j is the model we are taking as a reference (here the full model), the delta method gives

$$\frac{1}{\log(10)^2} \left(\frac{\operatorname{Var}(\hat{\mu}_i)}{\mu_i^2} - \frac{2 \operatorname{cov}(\hat{\mu}_i, \hat{\mu}_j)}{\mu_i \mu_j} + \frac{\operatorname{Var}(\hat{\mu}_j)}{\mu_j^2} \right) \tag{7.20}$$

for the asymptotic variance of (7.19). Of course we estimate (7.20) by plugging $\hat{\mu}_i$ for μ_i everywhere. This gives

model	$\log_{10}$ Bayes factor	MCSE
$\{0\}$	-3.872	0.018
$\{0, 1\}$	-3.797	0.019
$\{0, 2\}$	-1.074	0.016
$\{0, 3\}$	-2.079	0.018
$\{0, 1, 2\}$	-0.717	0.016
$\{0, 1, 3\}$	-1.878	0.016
$\{0, 2, 3\}$	-0.442	0.014
$\{0, 1, 2, 3\}$	0.000	0.000

Thus we have two decimal places in our estimate of the log Bayes factor.

Please note the near miraculous result of our calculations. By our trick of choosing priors for computational reasons (to get a uniform posterior for models) rather than reflecting anyone's prior opinion we have accurately calculated some extremely small probabilities. If we convert these log Bayes factors back to posterior probabilities corresponding, for example, to the uniform prior on models we get

model	posterior probability
$\{0\}$	0.0000808
$\{0, 1\}$	0.0000961
$\{0, 2\}$	0.0508
$\{0, 3\}$	0.00502
$\{0, 1, 2\}$	0.116
$\{0, 1, 3\}$	0.00798
$\{0, 2, 3\}$	0.218
$\{0, 1, 2, 3\}$	0.603

We won't bother with explicit standard error calculation, but it is clear from the standard error calculation for the log unnormalized Bayes factors that

each of these has two or three correct significant figures, *including the very small probabilities* for models $\{0\}$ and $\{0, 1\}$, which would be exceedingly difficult to estimate without our trick.

Appendix A

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position

regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent

appearance of the work's title, preceding the beginning of the body of the text.

A.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if

there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

- State on the Title page the name of the publisher of the Modified Version, as the publisher.

- Preserve all the copyright notices of the Document.

- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- Include an unaltered copy of this License.

- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled “History”

in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire

aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will

be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliography

Abramowitz, M. and Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York.

Atchade, Y. F. and Rosenthal, J. S. (2005). On adaptive Markov chain Monte Carlo algorithms. *Bernoulli*, 11:815–828.

Besag, J. (1994). Discussion of Grenander and Miller (1994). *Journal of the Royal Statistical Society, Series B*, 56:591–592.

Billingsley, P. (1968). *Convergence of Probability Measures*. Wiley, New York.

Billingsley, P. (1995). *Probability and Measure*. Wiley, New York, third edition.

Boyles, R. A. (1983). On the convergence of the EM algorithm. *Journal of the Royal Statistical Society Series B*, 45:47–50.

Burden, R. L. and Faires, J. D. (2005). *Numerical Analysis*. Thomson, 8th edition.

Caffo, B. S., Booth, J. G., and Davison, A. C. (2002). Empirical supremum rejection sampling. *Biometrika*, 89(4):745–754.

Chan, K. S. and Geyer, C. J. (1994). Comment on "Markov chains for exploring posterior distributions". *The Annals of Statistics*, 22:1747–1758.

Chung, K. L. (1974). *A Course in Probability Theory*. Academic Press, New York, 2nd edition.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39:1–38.

Devroye, L. (1986). *Non-uniform Random Variate Generation*. Springer-Verlag Inc.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley, Chichester; New York, 2nd edition.

Gaver, D. P. and O'Muircheartaigh, I. G. (1987). Robust empirical Bayes analyses of event rates. *Technometrics*, 29:1–15.

Gelman, A., Roberts, G. O., and Gilks, W. R. (1996). Efficient Metropolis jumping rules. In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., editors, *Bayesian Statistics 5 – Proceedings of the Fifth Valencia International Meeting*, pages 599–607. Clarendon Press [Oxford University Press].

Geyer, C. J. (1992). Practical Markov chain Monte Carlo (with discussion). *Statistical Science*, 7:473–511.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920.

Gilks, W. R. (1992). Derivative-free adaptive rejection sampling for Gibbs sampling. In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., editors, *Bayesian Statistics 4. Proceedings of the Fourth Valencia International Meeting*, pages 641–649. Clarendon Press.

Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41:337–348.

Glynn, P. W. and Iglehart, D. L. (1990). Simulation output analysis using standardized time series. *Mathematics of Operations Research*, 15:1–16.

Glynn, P. W. and Whitt, W. (1991). Estimating the asymptotic variance with batch means. *Operations Research Letters*, 10:431–435.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732.

Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society, Series B*, 56:549–603. With discussion.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.

Ibragimov, I. A. and Linnik, Y. V. (1971). *Independent and Stationary Sequences of Random Variables*. Walters-Noordhoff, The Netherlands.

Jarner, S. F. and Roberts, G. O. (2002). Polynomial convergence rates of Markov chains. *Annals of Applied Probability*, 12:224–247.

Jones, G. L. (2004). On the Markov chain central limit theorem. *Probability Surveys*, 1:299–320.

Jones, G. L., Haran, M., Caffo, B. S., and Neath, R. (2005). Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association,* to appear.

Jones, G. L. and Hobert, J. P. (2001). Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science*, 16:312–334.

Kantorovich, L. V. and Akilov, G. P. (1964). *Functional Analysis in Normed Linear Spaces*. Pergamon Press, Oxford. Translated from the original Russian (Fizmatgiz, Moskow, 1959) by D. E. Brown and A. P. Robertson.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In Neyman, J., editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press.

MacEachern, S. N. and Berliner, L. M. (1994). Subsampling the gibbs sampler. *American Statistician*, 48:188–190.

McCulloch, C. E. and Searle, S. R. (2001). *Generalized, Linear, and Mixed Models*. John Wiley & Sons, New York.

Meketon, M. S. and Schmeiser, B. W. (1984). Overlapping batch means: Something for nothing? In Sheppard, S., Pooch, U. W., and Pegden, C. D., editors, *1984 Winter Simulation Conference Proceedings*, pages 227–230. Elsevier/North-Holland, New York/ Amsterdam.

Mengersen, K. and Tweedie, R. L. (1996). Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, 24:101–121.

Meyn, S. P. and Tweedie, R. L. (1993a). *Markov Chains and Stochastic Stability*. Springer-Verlag, London.

Meyn, S. P. and Tweedie, R. L. (1993b). *Markov Chains and Stochastic Stability*. Springer-Verlag, London.

Murray, G. D. (1977). Discussion of the paper by Professor Dempster et al. *Journal of the Royal Statistical Society Series B*, 39:27–28.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer, New York, 1st edition.

Ralston, A. and Rabinowitz, P. (2001). *A First Course in Numerical Analysis*. Dover, New York, 2nd edition.

Ripley, B. D. (1987). *Stochastic Simulation*. John Wiley & Sons.

Robert, C. P. and Casella, G. (1999). *Monte Carlo Statistical Methods*. Springer, New York.

Roberts, G. O. (1999). A note on acceptance rate criteria for CLTs for Metropolis-Hastings algorithms. *Journal of Applied Probability*, 36:1210–1217.

Roberts, G. O. and Rosenthal, J. S. (1997). Geometric ergodicity and hybrid Markov chains. *Electronic Communications in Probability*, 2:13–25.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society, Series B*, 60:255–268.

Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2:341–363.

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press.

Rockafellar, R. T. and Wets, R. J.-B. (1998). *Variational Analysis*. Springer-Verlag.

Rosenthal, J. S. (1995). Minorization conditions and convergence rates for Markov chain Monte Carlo. *JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION*, 90:558–566.

Tierney, L. (1994). Markov chains for exploring posterior distributions (with discussion). *The Annals Of Statistics*, 22:1701–1762.

Wu, C.-F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11:95–103.