Introduction to Bayesian Computation

• In this lecture we’ll look at some of the basic techniques of Bayesian computation
  • Contour plotting
  • Hidden variable (AKA latent variables, random effects) models
  • Variable transformations
  • Laplace approximation
  • Rejection sampling
  • Importance sampling
  • SIR (Sampling and Importance Resampling)
  • Using samples to make inferences

The Beta-Binomial Overdispersion Model

• In §5.4 Albert introduces the “Beta-binomial model for overdispersion”. Unfortunately, he doesn’t explain very clearly where this model comes from
  • The simplest model for counting cases when \( n \) is the total number of cases and \( y \) the number that fall into category A (the remainder falling into category B) is binomial with a fixed probability \( \theta \) (which Albert calls \( p \)), that is, \( \theta \) is the probability of obtaining \( y \) A’s out of \( n \) trials. The likelihood is

\[
f(y \mid n, \theta) = C^n_y \theta^y (1 - \theta)^{n-y}
\]

The Beta-Binomial Overdispersion Model

• But in many situations, such a simple model is inadequate
  • For example, even though the observed probability of having a boy or a girl is close to 0.5, it is also the case that families having several boys or several girls are more probable than would be expected on a simple binomial model, where \( \theta \approx 0.5 \) is a fixed boy/girl ratio, independent of family:

\[
f(y \mid n, \theta) = C^n_y \theta^y (1 - \theta)^{n-y}
\]
  • The parameter \( \theta \) apparently varies from case to case. We say that the situation is overdispersed

The Beta-Binomial Overdispersion Model

• So, in this example, and in the one that Albert treats in §5.4, this simple model doesn't work. The parameter \( \theta \) isn’t constant from case to case.
  • So we generalize the simple binomial model:

\[
y_i \sim \text{binom}(n_i, \theta_i)
\]
\[
\theta_i \sim \text{beta}(\alpha, \beta)
\]
The Beta-Binomial Overdispersion Model

- So, in this example, and in the one that Albert treats in §5.4, this simple model doesn’t work. The parameter $\theta$ isn’t constant from case to case.

- The idea is to treat $\theta$ as a parameter with a prior. A natural prior for a parameter on $[0,1]$ is a beta prior; So we get

$$f(y \mid n, \theta) = C^n \theta^n (1-\theta)^{n-y}$$

$$g(\theta \mid \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\int_0^1 \theta^{\alpha-1}(1-\theta)^{\beta-1} d\theta} = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)} \text{ Definition!}$$

- This is the same as the equation at the bottom of p. 78, except that for technical reasons Albert prefers to use $K=\alpha+\beta$ and $\eta=\alpha/(\alpha+\beta)$, so $\alpha=K \eta, \beta=K(1-\eta)$.

- Here, $K$ is the precision (reciprocal of the variance) and $\eta$ the mean of the prior we put on $\theta$. We’ll now put a prior on $K, \eta$ and use the likelihood we just derived on the previous chart with that prior

- This is a simple example of a random effects model, (also called hidden Markov model or latent variables model). In this model we take into account the unobservable parameters, but marginalize them out as nuisance parameters.

- Albert chooses his prior independently on $K$ and $\eta$ as follows:

$$g(\eta, K) \propto \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2}$$

- So that the posterior density is

$$g(\eta, K \mid y) \propto \frac{1}{\eta(1-\eta)} \frac{1}{(1+K)^2} \prod_j \frac{B(K \eta + y_j, K(1-\eta) + n_j - y_j)}{B(K \eta, K(1-\eta))}$$
The Beta-Binomial Overdispersion Model

- Analysis 1: Plot the equal-density contours of the posterior distribution
- Albert provides a function in the LearnBayes package to calculate the log of the posterior density in $K$ and $\eta$, called betabinexch0

```r
?betabinexch0
betabinexch0
data(cancermortality)
mycontour(betabinexch0,
c(0.0001,0.003,1,20000),
cancermortality)
```

The Beta-Binomial Overdispersion Model

- The results are not very satisfactory. You see that $K$ pops off the top of the chart, even though we gave it a huge range, while $\eta$ is squeezed into a very narrow range
- To alleviate this problem, Albert uses tricks like the ones we discussed in the previous lecture to transform both $K$ and $\eta$ into a $(-\infty, \infty)$ range for better presentation of the results (Compare his figures 5.1 and 5.2). He uses a logit transformation for $\eta$ and a log transformation for $K$, appropriate for the $0,1$ range for $\eta$ and the $(0, \infty)$ range for $K$

So now we redo it, but this time using Albert’s reparameterized function betabinexch()

```r
?betabinexch
betabinexch
mycontour(betabinexch,
c(-8,-4.5,3,16.5),
cancermortality)
```

The Beta-Binomial Overdispersion Model

- The results are now considerably more satisfactory
- Note that the contours of the posterior distribution have quite an odd shape, very unlike the shape that we normally think of when we look at normal distributions
Approximations Based On Posterior Mode

- An old (pre-computer era) computational technique is the *Laplace approximation*.
- It is useful when there is a single mode that is well approximated by a multivariate normal (this is a common but not universal situation)
- The idea is very simple: Approximate the distribution near the mode by the multivariate normal distribution that you get from the first and second derivatives at the mode, and ignore the differences in the tails of the distribution
- Although MCMC methods have largely displaced this approximation, it still has its uses, as we shall see

Approximations Based On Posterior Mode

- So, suppose that \( h(\theta) \) is a log probability density on a vector parameter \( \theta \) (see §5.5 for how this may arise)
- Taylor’s theorem allows us to expand \( h(\theta) \) about any point; with the Laplace approximation, we expand it about the maximum (mode), \( \hat{\theta} \), where the first partial derivatives vanish. We keep second partials and drop the rest. The result is
  \[
  h(\theta) = h(\hat{\theta}) + (\theta - \hat{\theta})^T h''(\hat{\theta})(\theta - \hat{\theta}) / 2
  \]
  where \(^T\) means transpose and " means that we’ve computed the matrix of second partial derivatives (the “Hessian”)
- The first partials vanish because we are evaluating at the maximum

Approximations Based On Posterior Mode

- This is the same as the log of the density of a multivariate normal distribution with mean \( \hat{\theta} \) and variance-covariance matrix
  \[
  V = (h''(\hat{\theta}))^{-1}
  \]
- So, that’s the idea: find the maximum of the log density, compute its second partial derivative matrix at that point, and then replace the actual log density with the above multivariate normal density.
- Since R and the LearnBayes package have capabilities to do these computations, we have a reasonable way to make inferences when the conditions are satisfied

```r
?laplace
laplace
fit=laplace(betabinexch,
           array(c(7,6),c(1,2)),
           cancermortality)
fit
```
Approximations Based On Posterior Mode

- Once we have the Laplace approximation in hand, we can use LearnBayes’ `mycontour()` to plot the contours that arise from this approximation

```r
npar=list(m=fit$mode,v=fit$var)
mycontour(lbinorm,c(-8,-4.5,3,16.5),npar)
title(xlab="logit",ylab="log K")
```

- We notice that the shape of the contours is not at all like those of the actual posterior distribution; but note that the contours are a factor of 10 apart in the actual posterior density, which is quite a few standard deviations. One can believe that very close to the peak, the two agree more closely

Monte Carlo

- We have already seen some simple examples of one of the most commonly used computational techniques in modern Bayesian inference: Simulation by drawing a large random sample from some distribution, known as “Monte Carlo”

- This technique can be used for posterior inference (by drawing the sample from the posterior distribution and using it directly for inference), for prior or posterior prediction (predicting the data that we would expect to observe, which can be used as a “sanity check” on the results we obtain) and for many other purposes, including estimating the divisor in Bayes’ theorem (which is called the marginal probability of the data or the marginal likelihood, and is important in model selection problems)

Monte Carlo

- Thus far, the Monte Carlo methods examples we’ve given all were such that we could sample exactly from the posterior distribution using standard functions available in R

- Unfortunately, in most practical problems, this is not possible

- So we need to develop methods that can be used with a much larger variety of posterior distributions

Expectations from Samples

- Frequently, we will want to compute the expectation of some function $h(\theta)$ of our parameter vector $\theta$

- The definition of the expectation is

$$E(h(\theta)|y) = \frac{\int h(\theta)g(\theta)f(y|\theta)d\theta}{\int g(\theta)f(y|\theta)d\theta}$$

where $g(\theta)$ is the prior and $f(\theta|y)$ is the likelihood function (which, recall, is only defined up to an arbitrary constant factor, so that the product of prior and likelihood is unnormalized. Thus, the denominator is part of the definition

- (Note that Albert’s $g$ on p. 84 is a normalized density)
Expectations from Samples

- But if we have a sample \{\theta_j\} of \(m\) random draws from the posterior distribution on \(\theta\), we can easily approximate \(E(h(\theta | y))\), even though we may not know the normalizing factor. This is one of the reasons that Monte Carlo methods are so useful. Indeed, we have

\[
E(h(\theta | y)) = \frac{1}{m} \sum_{j=1}^{m} h(\theta_j)
\]

with the usual formula for estimating the standard deviation of our estimate (see p. 84)
- There are various ways of obtaining such samples. We'll discuss some now and others later.

Rejection Sampling

- The first example of a more powerful technique for obtaining a sample from the posterior distribution is rejection sampling. To use it, we must be able to find a density \(p(\theta)\) that we can sample from easily, with the property that for all \(\theta\), there is a constant \(c\) such that

\[
g(\theta | y) \leq cp(\theta)
\]

where \(g(\theta | y)\) is the desired target density
- Important: We only need to know \(g\) up to a constant, since any unknown constant can be absorbed into \(c\)
- The more closely the shape of \(p\) resembles that of \(g\), the more efficient the sampling will be

Rejection Sampling

- Strategy: Once we have a suitable \(p\), we can simulate any number of draws from \(g\) as follows:
  1. make a proposal draw \(\theta\) from \(p(\theta)\)
  2. generate a uniform random number \(x\) on [0,1]
  3. if \(x \leq cp(\theta)/g(\theta | y)\), accept the proposal and add it to our list of draws; otherwise reject it
  4. repeat (1-3) until enough samples have been accepted
- The set of accepted draws from step (1) will be a sample from the desired target distribution

Rejection Sampling

- Albert implements this scheme with this chapter’s beta-binomial example as the target distribution
- He chooses as his proposal distribution \(p\) a multivariate \(t\) distribution with 4 degrees of freedom based upon the mode and variance-covariance matrix (the scale matrix for the multivariate \(t\) distribution) that comes out of laplace()
- The idea is that a \(t\) distribution has fatter tails than would a normal distribution. We saw that the normal approximation didn’t do well in the tails; the idea is that a \(t\) distribution with moderately heavy tails, when inflated by the constant \(c\), will everywhere be able to dominate the target posterior distribution, and thus satisfy \(g(\theta | y) \leq cp(\theta)\) everywhere
Rejection Sampling

- The $d$-variate $t$ distribution with mode $\hat{\theta}$, scale matrix $V$, and degrees of freedom (dof) $\nu$ has the density

$$p(\theta | \hat{\theta}, V, \nu) \propto \frac{1}{[1 + (\theta - \hat{\theta})'(\nu V)^{-1}(\theta - \hat{\theta})]^{(\nu + d)/2}}$$

- It is relatively simple to sample from this density, and Albert provides functions rmt and dmt to support these calculations.

Rejection Sampling

- The first thing we have to do is to find $\hat{\theta}$ and $V$; we already did this earlier when discussing the Laplace approximation.

- The next thing is to compute a constant $c$ so that the inequality $g(\theta) \leq c p(\theta)$ will be satisfied everywhere.

- Albert uses a clever idea: One of the things the function laplace() does is to calculate the point where a function is maximized; so have it compute the point where the function $g(\theta) / p(\theta)$ is maximized, and set the value of this ratio at that point equal to $c$. Then this value of $c$ (or any bigger value) will automatically cause the inequality to be satisfied.

Rejection Sampling

- Since laplace() needs logs, the function to use is the difference of the logs of the two functions. Here is the code that computes the log of $c$.

```r
betabinT=function(theta,datapar){
data=datapar$data; tpar=datapar$par
d=betabinexch(theta,data)-dmt(theta,mean=c(tpar$m),
S=tpar$var,df=tpar$df,log=TRUE)
return(d)
}

tpar=list(m=fit$mode,var=2*fit$var,df=4)
datapar=list(data=cancermortality,par=tpar)
start=array(c(-6.9,12.4),c(1,2))
fit1=laplace(betabinT,start,10,datapar)
fit1$mode
logc=betabinT(fit1$mode,datapar); logc
```

Rejection Sampling

- Use the function rejectsampling() that comes with the LearnBayes package to calculate the sample.

```r
theta=rejectsampling(betabinexch,tpar,logc,10000,cancermortality)
dim(theta)
mycontour(betabinexch,c(-8,-4.5,3,16.5),cancermortality)
points(theta[,1],theta[,2])
```
Importance Sampling

- Importance sampling is a technique for evaluating integrals (expectations) by Monte Carlo simulation.
- Unlike rejection sampling, we do not discard samples; every sample counts in the result.
- However, the sample is not a sample from the posterior distribution, but instead is a sample with weights that tell us how much each sample should count in the calculation of the integral.

Here’s the idea: find a distribution $p(\theta)$ from which we can easily sample, which approximates the target distribution (this is similar to rejection sampling).
- Albert uses a multivariate t distribution for $p(\theta)$.
- Then consider the integral
  \[
  E(h(\theta)|y) = \frac{\int h(\theta)g(\theta|y)d\theta}{\int g(\theta|y)d\theta}
  \]

This can be evaluated

\[
E(h(\theta)|y) = \frac{\int h(\theta)g(\theta|y)d\theta}{\int g(\theta|y)d\theta} = \frac{\int h(\theta)\frac{g(\theta|y)}{p(\theta)}p(\theta)d\theta}{\int \frac{g(\theta|y)}{p(\theta)}p(\theta)d\theta} = \frac{\int h(\theta)w(\theta)p(\theta)d\theta}{\int w(\theta)p(\theta)d\theta}
\]

where $w(\theta)=f(y|\theta)g(\theta)/p(\theta)$ (we don’t need the normalization constant as it will cancel).

If $\{\theta\}$ is a sample drawn from the density $p(\theta)$, then we can approximate

\[
E(h(\theta)|y) = \bar{h} = \frac{\sum h(\theta')w(\theta')}{\sum w(\theta')}
\]

with simulation error (standard deviation)

\[
se = \sqrt{\frac{\sum ((h(\theta') - \bar{h})w(\theta'))^2}{\sum w(\theta')}}
\]

Note that the book’s formula for $se$ is not correct (the LearnBayes code is OK). See book’s errata page at [http://bayes.bgsu.edu/bcwr/errata.txt](http://bayes.bgsu.edu/bcwr/errata.txt).
Importance Sampling

- The function `impsampling()`, part of the LearnBayes package, implements the calculation of the sample and the weights.
- The following code calculates the mean of log($K$), which, recall, is the second component of the object returned by `betabinexch()`.

```r
myfunc=function(theta)
    return(theta[,2])
s=impsampling(betabinexch,tpar,
    myfunc,10000,cancermortality)
cbind(s$est,s$se)
```

Sampling Importance Resampling (SIR)

- This is a variation on importance sampling.
- Again, we draw a large sample from a proposal distribution that ideally approximates the target distribution.
- Then, we resample with replacement from this sample, with probabilities that are proportional to the ratio between the target density and the proposal density at each sample point.
- Some sample points will appear in the resampled sample more than once, and others will not appear at all; but the resampled sample has the statistical properties of a sample from the target distribution and can be used as such to compute means, quantiles, etc.

Sampling Importance Resampling (SIR)

- The SIR idea can also be used to investigate how inferences may depend on the prior.
  - Suppose a sample from the posterior distribution is available.
  - We wish to see how inferences would be affected by the use of another prior.
  - Then we resample, using as weights the ratios between the new prior and the old prior at each sampled point.
  - J. O. Berger advocates publishing the entire sample on the web, so that other researchers can start with the sample and use techniques like SIR to investigate the problem in ways that the original researcher had not anticipated.

Sampling Importance Resampling (SIR)

- We can actually use the output of the function `impsampling()` to give us the raw materials for SIR more simply than in the text, because this function also returns the sample and the weights:

```r
s$theta[1:20,]
indices = sample(1:10000,size=10000,
    prob=s$wt,replace=T)
theta.s=s$theta[indices,]
dim(theta.s)
theta.s[1:20,]
```

- Note that you don’t have to normalize the weights; the new versions of R do this for you.
Sampling Importance Resampling (SIR)

• An illustration of another use of the SIR idea is a sensitivity analysis, to see how sensitive the results of an analysis are to the deletion of individual data points from the data set
• Call the data set with the $i$th observation removed $y_{(-i)}$. Then we want to resample the sample from the full data with weights

\[
  w(\theta) = \frac{g(\theta | y_{(-i)})}{g(\theta | y)} = \frac{1}{f(y_i | \theta)} = \frac{B(K\eta, K(1-\eta))}{B(K\eta + y_i, K(1-\eta) + n_i - y_i)}
\]

Sampling Importance Resampling (SIR)

• We see from the graph that a few data points (#10, 15 and 19) have somewhat more influence than the others, but all subsamples seem to give consistent results

Sampling Importance Resampling (SIR)

• LearnBayes has a function `bayes.influence()` that does the resampling and provides information that can be used to inspect the sensitivity graphically

```r
?bayes.influence
bayes.influence
S=bayes.influence(theta.s, cancermortality)
plot(c(0,0,0), S$summary, type="b", lwd=3,
     xlim=c(-1,21), ylim=c(5,13),
     xlab="Observation removed",
     ylab="log K")
for(i in 1:20)
  lines(c(i,i,i), S$summary.obs[i,],
        type="b")
```

Summary

• We’ve seen some very important ideas in this set of charts
  • A simple random effects model
  • Methods for plotting posterior densities for bivariate problems
  • The importance of choosing the right set of variables, and of transforming (e.g., logit and log transformations)
  • Several uses for the Laplace approximation
  • Several different schemes for drawing a sample from a posterior distribution
  • Using the samples for various purposes, e.g., computing posterior means and quantiles, investigating sensitivity of results to deletion of data