Bayesian Robust Regression

Jim Albert

May 26, 2016
Regression Example

- Crime dataset from Agresti and Finlay “Statistical Methods for Social Sciences” text
- For each US state, collect a number of variables including:
  - crime \( (y) \) - violent crimes per 100,000 people
  - poverty \( (x_1) \) - percent of population living under poverty line
  - single \( (x_2) \) - percent of population with single parents
- Want to use poverty and single to predict crime
Usual Normal Regression Model

\[ y_i = x_i \beta + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \]

- When you fit this model, you see some problems
- A few large residuals
- One of the large residuals has high leverage
Residual/Leverage Plot

crime

poverty

single
Robust Regression

Try alternative model

\[ y_i = x_i \beta + \epsilon_i, \quad \epsilon_i \sim f(0, \sigma^2) \]

- Choose alternative error distribution \( f \) with flat tails
- Example – choose \( f \) to be a \( t \) distribution with small number of d.f.
- When you fit this robust regression, outliers will be given smaller weight in the fitting
Overview of Bayesian Inference

- Regard parameter $\theta$ as random, and assign it a prior density $g(\theta)$
- How to assign the prior $g$?
- Observe data $y = (y_1, ..., y_n)$, compute likelihood

$$L(\theta) = \prod_{j=1}^{n} f(y_j|\theta)$$

- All inference about $\theta$ is based on the posterior density

$$g(\theta|y) \propto g(\theta) \times L(\theta)$$
Learn about posterior by taking a simulated sample $\theta_1, \ldots, \theta_m$ from the posterior distribution $g(\theta|y)$

Summarize $\{\theta_k\}$ to perform inference

A 90% interval estimate for $\theta$ is given by

$$(\theta_{0.05}, \theta_{0.95})$$

where $\theta_p$ is the $p$th fractile of the simulated draws

Also use simulation to learn about future data $y^*$
Bayesian Robust Regression

- **Sampling:**
  \[ y_i = x_i \beta + \epsilon_i, \quad \epsilon_i \sim t_4(0, \sigma) \]

- **Prior:**
  \[ g(\beta, \sigma) \propto \frac{1}{\sigma} \]

- **Posterior density (product of likelihood and prior)**
  \[ g(\beta, \sigma | y) \propto \left( \prod f(y_i | \beta, \sigma) \right) \times g(\beta, \sigma) \]
Markov Chain Monte Carlo (MCMC)

- Want to construct a Markov Chain that converges to the posterior probability density
- Simulate from Markov Chain – after some initial burn-in period, assume that the simulated sample is approximately a sample from the posterior density
- A popular general method is the class of Metropolis-Hastings algorithms
Metropolis-Hastings Algorithm

- Construct a proposal density $p(y|x)$
- Then the M-H algorithm consists of three steps:

1. Given a current simulated value $x$, PROPOSE a new value $y$ from $p(y|x)$
2. Compute ACCEPTANCE PROBABILITY

$$PROB = \min \left( \frac{f(y)p(x|y)}{f(x)p(y|x)}, 1 \right).$$

3. With probability $PROB$, move to the proposed value $y$; otherwise stay at the current value $x$
Random Walk Metropolis-Hastings

- Let the proposal density have form

\[ p(y|x) = h(y - x) \]

where \( h \) is symmetric about 0

For example, can let \( p(y|x) \sim N(x, C) \)

- So proposal density is chosen in a neighborhood of current value \( x \), where the width and direction of the interval is determined by the var-cov matrix \( C \)
Using M-H Random Walk: A General Strategy

- Helpful to transform parameters so that they are all real-valued
- Do a preliminary fitting to learn about location, spread, and correlation structure of posterior
- Use this information to figure out a suitable starting value, and to decide on the var-cov matrix $C$
Preliminary Fitting: Laplace Approximation

- Use a Newton-Raphson algorithm to find the posterior mode \( \hat{\theta} \)
- Evaluate numerically the second derivative matrix of \( \log g(\theta|y) \) at the mode – call this \( D \)
- This gives the approximation

\[ \theta \sim N(\hat{\theta}, \Sigma) \]

where \( \Sigma = (-D)^{-1} \)
Using LearnBayes Package

- Write a short function `regpost` that defines the log
- Function has two arguments:
  1. `theta` - vector of parameters (here $\beta$ and $\log \sigma$)
  2. `stuff` - data and prior parameters needed to compute the posterior
- Function `laplace` will find posterior mode (and estimate of -
  Function `rwmetrop` implements random walk Metropolis algorithm
Function defining log posterior

```R
regpost <- function(theta, data){
  beta <- matrix(theta[1:3], 3, 1)
  sigma <- exp(theta[4])
  y <- data[, "crime"]
  x <- as.matrix(cbind(1, 
      data[, c("poverty", "single")]))
  sum(dt((y - x %*% beta) / sigma, df=4, log=TRUE) - 
       log(sigma))
}
```
Initial Fit

- To get some sense of the location of the posterior, use the `laplace` function.
- Inputs are `regpost`, guess at mode (use ordinary least squares values), and data.
- Outputs are:
  1. `mode` – posterior mode
  2. `var` - estimate at posterior var-cov matrix
library(LearnBayes)

## Loading required package: coda

laplace.fit <- laplace(regpost, 
  c(ols$coef, log(245)), cdata)

summary(laplace.fit)

## Var 1 : Mean = -1445.718 SD = 254.25
## Var 2 : Mean = 9.645 SD = 8.299
## Var 3 : Mean = 170.078 SD = 24.933
## Var 4 : Mean = 5.161 SD = 0.13
Running the M-H Random Walk Algorithm

- Use function `rwmetrop` in `LearnBayes` package
- Arguments are
  1. function defining log posterior
  2. estimate at variance covariance matrix $\Sigma$, scale parameter $c$
  3. starting value for simulation
  4. number of simulations and
  5. data used in posterior

- Proposal value will be

$$\theta^P = \theta^C + cZ,$$

where $Z$ is normal$(0, \Sigma)$

- Output will be (1) matrix of simulated draws and (2) acceptance rate in algorithm
For Our Example

- Have an estimate at the var-cov matrix $\Sigma$ (from laplace fitting)
- I’ll use the mode of the posterior as a starting value (but . . .)
- I’ve stored the mode and var-cov matrix in list fit
- I’ll start with using the scale parameter $c = 1.5$
- Run the chain for 10,000 iterations
Running M-H Random Walk

```r
rw.fit <- rwmetrop(regpost,
                   list(var=laplace.fit$var, scale=1.5),
                   start=laplace.fit$mode,
                   10000,
                   cdata)
```
MCMC Diagnostics

- Question: Is the MCMC simulations a reasonable approximation to the posterior density?
- Burn-in? (Has the algorithm approximately converged?)
- Mixing? (What does the autocorrelation of draws look like?)
- Sufficient iterations? (Have we taken enough iterations to learn about the quantities of interest?)
coda package in R

- Contains a collection of useful functions for MCMC diagnostics
- Convert the matrix of simulated draws to a mcmc object
- The summary and plot methods are helpful in summarizing and plotting output
- Get essential summaries; get trace plot and density estimate for each parameter
coda Package: Trace Plots

Trace plots for the parameters beta1, beta2, beta3, and log sigma, showing their evolution over iterations from -2000 to 10000.
coda Package: Autocorrelation Plots
coda Package: Density Plots

- Density plots for $eta_1$, $eta_2$, $eta_3$, and $\log \sigma$.
## coda Package: Summary Method

```
##
## 1. Empirical mean and standard deviation for each variable, plus standard error of the mean:
##
##|   |       Mean      |       SD       | Naive SE | Time-series SE |
##|---|----------------|----------------|----------|----------------|
##| beta1 | -1388.675     | 230.7975     | 2.307975 | 8.66645        |
##| beta2 |     11.170     |    8.8652     | 0.088652 |    0.34521     |
##| beta3 |    163.180     |   23.0795     | 0.230795 |    0.86463     |
##| log sigma |    5.191     |    0.1353     | 0.001353 |    0.00553     |

## 2. Quantiles for each variable:

```
Inference

- Suppose one is interested in learning about a function $h(\beta)$.
- Learn about the posterior of $h$:
  - for each simulated draw $\beta^*$ from posterior, compute $h(\beta^*)$
  - summarize and graph sample $\{h(\beta^*)\}$
Suppose we wish to predict crime $y$ at particular set of values of covariate $x$

Simulate value $y^*$ from posterior predictive distribution:

1. simulate $\beta^*$ from posterior $g(\beta|y)$
2. simulate $y^*$ from sampling density $f(y_{new}|\beta^*)$
Write short function `inference.prediction.R`

- Inputs are `bayes.fit` (output from `rwmetrop`) and covariate vector `x`
- Outputs are
  - simulated sample from posterior of `xβ`
  - simulated sample from predictive distribution of `y`
Comparison of Probability Intervals for $x\beta$ and $y$*

## Warning: Removed 80 rows containing non-finite values (stat_density).

![Graph showing comparison of probability intervals for $x\beta$ and $y$ with a warning about non-finite values removed.](image)
MCMC algorithms are potentially very powerful for summarizing high-dimensional posterior distributions.

Illustrated (1) using a laplace method to find posterior mode and (2) using the output to construct a reasonable M-H random walk chain.

Important to monitor MCMC output using tools like those supplied in the coda package.

Simulation attractive for inference (about functions of parameters), and for prediction.