Exercises in Bayesian Modeling and Machine Learning

Ryan T. Moore

2023-01-10

Set Up

```
library(here)
library(tidyverse)

# Bayes:
library(coda)
library(MCMCpack)

# LASSO, ridge, elastic net:
library(glmnet)

# tidymodels:
library(tidymodels)
```

Scan the available Bayesian models in `MCMCpack` with `help(package = "MCMCpack")`.

Next, prepare the data. We'll do some usually-unnecessary mutations just to facilitate the way `glmnet` expects the data (it doesn't adopt the typical `y ~ x` formula specification). We could do this through a `recipe()`, as well.

```
social <- read_csv("http://j.mp/2Et71U0")

social <- social %>% mutate(age = 2006 - yearofbirth,
                            isFemale = (sex == "female"),
                            isFemale = as.numeric(isFemale),
                            sentNeighbors = (messages == "Neighbors"),
                            sentNeighbors = as.numeric(sentNeighbors))

social <- social %>% sample_n(50000)
```

Estimate Bayesian logistic regression

For practice, estimate the simple Bayesian logistic regression (with Gaussian priors on the β's):

```
mc_post <- MCMClogit(primary2006 ~ messages, data = social)
```

Look at the summary of `mc_post`.

Now, estimate a fuller Bayesian logistic regression, modeling `primary2006` as function of `primary2004`, `age`, `age`^{2}, and `messages`. The algorithm is a Metropolis one, similar to the "full conditional probability distribution" Gibbs sampling. Set the MCMC burnin to 500, the number of MCMC draws to 2000, with a thinning value of 2. Store output as `mc_post_full`.

Look at summary of `mc_post_full`.

Examine the "highest posterior density" intervals for the coefficients. These are the 95% of values most common in the posterior. They may differ from the central credible interval.

```
HPDinterval(mc_post_full)
```

Create graphics to diagnose the MCMC. Careful – check where the plot will be stored. What working directory will your plot be produced from?

```
pdf("mcmc_diagnose.pdf")
plot(mc_post_full)
dev.off()
```

Do a posterior probability calculation from the posterior draws. Specifically, what's the posterior probability the `Neighbors` postcard has a larger coefficient than the `Hawthorne` card?

```
mean(mc_post_full[, "messagesHawthorne"] < mc_post_full[, "messagesNeighbors"])
```

```
## [1] 1
```

Now, what's the posterior probability that the `Neighbors` coefficient is greater than 0.7?

Estimate Machine Learning Models

Set up the data objects for `glmnet`. `X` will be a raw numeric matrix; `y` will be a raw numeric vector. (We can avoid much of this with the `tidyverse`.)

```
predictors <- c("isFemale", "primary2004", "sentNeighbors", "hhsize", "age")

X <- social[, predictors]
```

Next we'll do some naive "feature engineering". This might include polynomial functions of predictors, logarithmic and power transformations, deep interactions, or other transformations of predictors.

```
X <- X %>% mutate(age2 = age^2,
                  age3 = age^3,
                  age4 = age^4,
                  age5 = age^5) %>%
  as.matrix()

# Extract outcome as raw numeric vector, for glmnet:
y <- social[, "primary2006"] %>% unlist %>% as.numeric()
```

Estimate the least squares (LS) model using all these predictors.

```
lm_out <- lm(primary2006 ~ isFemale + primary2004 + sentNeighbors + hhsize +
               age + I(age ^ 2) + I(age ^ 3) + I(age ^ 4) + I(age ^ 5),
             data = social)

coefs_lm <- coef(lm_out)
```

Examine your results. Next, estimate the LASSO:

```
lasso_out <- glmnet(X, y, alpha = 1)
```

Look at the `lasso_out` and `coef(lasso_out)` objects.

How should we choose among these sets of coefficients? We'll use cross-validation to see which predicts best in *out-of-sample* tests. `glmnet` automates creating many training data sets, estimating the LASSO, then seeing how well the trained coefficients do in held-out test data.

```
set.seed(281) # (So that our train-test sets are consistent across implementations)
cv_lasso_out <- cv.glmnet(X, y, alpha = 1)
```

Now, show the coefficients of the "best" fit model, the one that gives the minimum deviance:

```
coef(cv_lasso_out, s = "lambda.min")
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)    6.984516e-02
## isFemale      -2.054705e-03
## primary2004    1.421135e-01
## sentNeighbors  7.152518e-02
## hhsize        -1.425083e-03
## age            1.037849e-03
## age2           5.890873e-05
## age3           1.422583e-07
## age4           .
## age5          -8.586565e-11
```

Consider the coefficients of a more parsimonious model, one that has deviance within 1 SE of the minimum:

```
coef(cv_lasso_out, s = "lambda.1se")
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                           s1
## (Intercept)    6.248193e-02
## isFemale       .
## primary2004    1.316125e-01
## sentNeighbors  5.324689e-02
## hhsize         .
## age            3.893162e-03
## age2           .
## age3           .
## age4           .
## age5          -6.121534e-12
```

```
coefs_lasso <- coef(cv_lasso_out, s = "lambda.1se")
```

Compare the model coefficients:

```
cbind(coefs_lasso, coefs_lm)
```

```
## 10 x 2 sparse Matrix of class "dgCMatrix"
##                           s1      coefs_lm
## (Intercept)    6.248193e-02 -1.837626e+00
## isFemale       .            -2.087444e-03
## primary2004    1.316125e-01  1.402359e-01
## sentNeighbors  5.324689e-02  7.117105e-02
## hhsize         .             2.064420e-03
## age            3.893162e-03  2.006594e-01
## age2           .            -7.669371e-03
## age3           .             1.398580e-04
```

```
## age4              .              -1.190440e-06
## age5         -6.121534e-12  3.766219e-09
```