MS2505: Bayesian Statistics
Course Project

December 17, 2024

Name	Samuel Jonsson
E-Mail	sajs19@student.bth.se
Person Nr.	19990415-5596
Program	DVAMI19h

1 Setup

- Describe the data and the analysis problem.
- Choose and describe the modeling approach (e.g., non-hierarchical or hierarchical model).
- Justify your prior choice.
- Perform posterior predictive checks.

1.1 Analysis problem

1.2 Data Selection

Describe the data and the analysis problem.

The dataset selected is a datasets containing a list of emails, as well as a label marking each email as "spam" or "ham" (spam or not spam). The first 10 rows of the dataset looks as follows:

Table 1
`mail_data.csv` dataset first 10 rows

Category	Message
ham	"Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat..."
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	"Nah I don't think he goes to usf, he lives around here though"
spam	"FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv"
ham	Even my brother is not like to speak with me. They treat me like aids patent.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
...	...

Then, using a Python script, the labels were converted to 1 if it was "spam" and 0 if it was "ham", for easier analysis.

1.3 Model

- Choose and describe the modeling approach (e.g., non-hierarchical or hierarchical model).
- Justify your prior choice.

The model chosen was a binomial likelihood model with a beta prior. As the goal is to analyse the probability of an email being spam, the fallout will be binary (either it is spam or it is not). Hence, a binomial likelihood, where I want to find the parameter θ in a dataset of fixed size with a set number of "successes" and "fails" (spam and ham), is appropriate.

Additionally, as I do not have any prior knowledge in regards to this distribution, a non-informative prior is the most suited option, and as $Beta(1, 1)$ is a common prior used with binomial likelihood functions, I chose it for this problem.

1.4 Prior checks

,

Perform posterior predictive checks.

2 Results

Include diagnostics to assess model convergence and adequacy.

3 Discussion

Discuss results, problems encountered, and possible improvements.

A R Code

Listing 1
Project R code

```r
# Load required libraries
library(bayesplot)
library(rstanarm)
library(ggplot2)
library(brms)

# Set a global random seed for reproducibility
set.seed(123)

# Create the directory if it doesn't exist
if (!dir.exists("Project/logs")) {
    dir.create("Project/logs", recursive = TRUE)
}

# Create the directory for figures if it doesn't exist
if (!dir.exists("figures")) {
    dir.create("figures", recursive = TRUE)
}

# Specify the log file path
log_file <- "Project/logs/R_output.log"

# Open the sink to redirect output
sink(log_file)

```

```
# Read the data
mail_data <- read.csv("Project/data/mail_data_bin.csv")

# Set metadata
alpha_prior <- 1
beta_prior <- 1
total_emails <- nrow(mail_data)
spam_count <- sum(mail_data$Category == 1)
ham_count <- sum(mail_data$Category == 0)

# Fit the model with a vague prior
fit_prior <- brm(
    Category ~ 1,
    data = mail_data,
    family = bernoulli(),
    prior = prior(beta(1, 1),
        class = "Intercept"
    )
)

# Prior predictive check
pdf("figures/prior_predictive_check.pdf")
pp_check(fit_prior, type = "hist") +
    ggtitle("Prior Predictive Check for Email Spam Model")
dev.off()

# Fit a robust model with a more informative prior
fit_robust <- brm(
    Category ~ 1,
    data = mail_data,
    family = bernoulli(),
    prior = prior(beta(2, 2), class = "Intercept")
)

# Posterior predictive check
pdf("figures/posterior_predictive_check.pdf")
pp_check(fit_robust, type = "dens_overlay") +
    ggtitle("Posterior Predictive Check for Robust Email Spam Model")
dev.off()

# Compute posterior parameters for P(spam)
alpha_post <- alpha_prior + spam_count
beta_post <- beta_prior + ham_count

# Posterior probability of an email being spam
posterior_spam <- alpha_post / (alpha_post + beta_post)

# Print results
cat("Prior: Beta(", alpha_prior, ",", beta_prior, ")\n")
cat("Spam Count:", spam_count, "\n")
cat("Ham Count:", ham_count, "\n")
```

```r
cat("Posterior: Beta(", alpha_post, ",", beta_post, ")\n")
cat("P(spam):", posterior_spam, "\n")

# Posterior Predictive Checking
cat("\n--- Posterior Predictive Checking ---\n")

# Simulate posterior predictive samples
num_samples <- 1000
posterior_samples <- rbeta(num_samples, alpha_post, beta_post)

observed_counts <- spam_count / total_emails

# Generate density overlay
y <- rep(observed_counts, num_samples)
yrep <- matrix(posterior_samples, nrow = num_samples)

# Save density overlay plot
# pdf("../figures/ppc_density_overlay.pdf")
# ppc_dens_overlay(y = y, yrep = yrep) +
#    ggtitle("Posterior Predictive Check: Density Overlay")
# dev.off()

# Generate and save histogram of posterior samples
pdf("figures/ppc_histogram.pdf")
hist(posterior_samples,
    breaks = 30, col = "blue", border = "white",
    main = "Posterior Predictive Distribution", xlab = "P(spam)"
)
dev.off()

# Sensitivity Analysis
cat("\n--- Sensitivity Analysis ---\n")
sensitivity_results <- data.frame()
alpha_values <- seq(0.5, 2, by = 0.5)
beta_values <- seq(0.5, 2, by = 0.5)

for (alpha in alpha_values) {
    for (beta in beta_values) {
        alpha_post_temp <- alpha + spam_count
        beta_post_temp <- beta + ham_count
        posterior_temp <- alpha_post_temp / (alpha_post_temp +
           beta_post_temp)
        sensitivity_results <- rbind(
            sensitivity_results,
            data.frame(alpha, beta, posterior_temp)
        )
    }
}

# Display the sensitivity analysis results
cat("Sensitivity Analysis Results:\n")
```

```
print(sensitivity_results)

# Save sensitivity results to CSV
write.csv(sensitivity_results,
    "Project/logs/sensitivity_analysis.csv",
    row.names = FALSE
)

# Flush the output and close the sink
flush.console()
sink()
```