CGS698C - Assignment 5

Sahil Tomar (210898)

2024-07-11

Part 1 - Information-theoretic measures and cross-validation

We are given:

- the data : $y = \{10, 15, 15, 14, 14, 14, 13, 11, 12, 16\}$

```
data_obs <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
```

Exercise 1.1 : Graph the posterior distribution for each model

Model 1

- the likelihood assumption : y_i ~ $Binomial(N = 20, \theta)$
- The prior assumption : θ ~ $Beta(6, 6)$

Graphing the posterior distribution of θ :

```
alpha_prime_1 <- 6 + sum(data_obs)
beta_prime_1 <- 6 + sum(20 - data_obs)
print(paste("( ", alpha_prime_1, "," , beta_prime_1, ") "))
```

```
## [1] "(  140 , 72 ) "
```

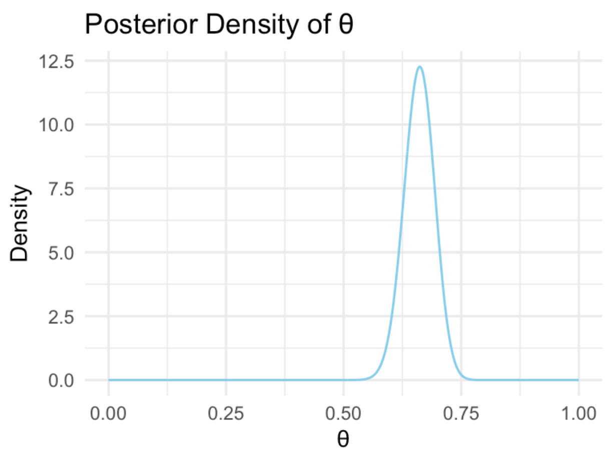
- So, Analytically Derived Posterior Distribution : $\theta|y$ ~ $Beta(140, 72)$

```
theta_values <- seq(0, 1, length.out = 1000)

posterior_density <- dbeta(theta_values, alpha_prime_1, beta_prime_1)

posterior_df_1 <- data.frame(theta = theta_values, density = posterior_density)

ggplot(posterior_df_1, aes(x = theta, y = density)) +
  geom_line(color = "skyblue") +
  labs(title = "Posterior Density of θ",
       x = "θ",
       y = "Density") +
  theme_minimal()
```



Model 2

- the likelihood assumption : y_i ~ $Binomial(N = 20, \theta)$
- The prior assumption : θ ~ $Beta(20, 60)$

Graphing the posterior distribution of θ :

```r
alpha_prime_2 <- 20 + sum(data_obs)
beta_prime_2 <- 60 + sum(20 - data_obs)
print(paste("( ", alpha_prime_2, "," , beta_prime_2, ") "))
```

```
## [1] "(  154 , 126 ) "
```

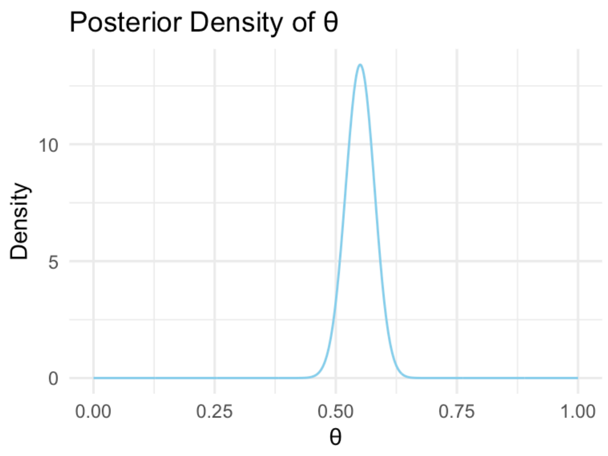
- So, Analytically Derived Posterior Distribution : $\theta|y \sim Beta(154, 126)$

```r
theta_values <- seq(0, 1, length.out = 1000)

posterior_density <- dbeta(theta_values, alpha_prime_2, beta_prime_2)

posterior_df_2 <- data.frame(theta = theta_values, density = posterior_density)

ggplot(posterior_df_2, aes(x = theta, y = density)) +
  geom_line(color = "skyblue") +
  labs(title = "Posterior Density of θ",
       x = "θ",
       y = "Density") +
  theme_minimal()
```



Exercise 1.2 : Compute log pointwise predictive density (lppd) for each model

Model 1

```r
lppd_m1 <- 0
for (i in 1:length(data_obs))
{
  sample_theta <- rbeta(1000, alpha_prime_1, beta_prime_1)
  lpd_i <- log(mean(dbinom(data_obs[i], 20, sample_theta)))
  lppd_m1 <- lppd_m1 + lpd_i
}

print(paste("log pointwise predictive density (lppd) for Model 1: ", lppd_m1))
```

```
## [1] "log pointwise predictive density (lppd) for Model 1:  -20.3610604010218"
```

Model 2

```r
lppd_m2 <- 0
for (i in 1:length(data_obs))
{
  sample_theta <- rbeta(1000, alpha_prime_2, beta_prime_2)
  lpd_i <- log(mean(dbinom(data_obs[i], 20, sample_theta)))
  lppd_m2 <- lppd_m2 + lpd_i
}

print(paste("log pointwise predictive density (lppd) for Model 2: ", lppd_m2))
```

```
## [1] "log pointwise predictive density (lppd) for Model 2:  -25.9328848634411"
```

Exercise 1.3 : Calculate in-sample deviance for each model from the log point-wise predictive density (lppd)

Model 1

```
print(paste("In-sample Deviance for Model 1: ", -2*lppd_m1))
```

```
## [1] "In-sample Deviance for Model 1:  40.7221208020437"
```

Model 2

```
print(paste("In-sample Deviance for Model 2: ", -2*lppd_m2))
```

```
## [1] "In-sample Deviance for Model 2:  51.8657697268822"
```

Why is it called in-sample ?

The in-sample deviance is a measure of how well the model fits the observed data, calculated using the log point-wise predictive density (lppd). It is called "in-sample" because it is based on the same data that was used to fit the model (i.e., the data points y_i).

Exercise 1.4 : Based on in-sample deviance, which model is a better fit to the data?

Since Model 1 has a lower in-sample deviance (40.80) compared to Model 2 (51.82), we can conclude that Model 1 is a better fit to the data. This suggests that the $Beta(6, 6)$ prior for θ in Model 1 is more appropriate for this dataset than the $Beta(20, 60)$ prior in Model 2.

We are given:

- the new data : $y_{new} = \{5, 6, 10, 8, 9\}$

```
data_new <- c(5, 6, 10, 8, 9)
```

Exercise 1.5 : Calculate out-of-sample deviance now to compare your models.

Model 1

```
lppd_new_m1 <- 0
for (i in 1:length(data_new))
{
  sample_theta <- rbeta(1000, alpha_prime_1, beta_prime_1)
  lpd_i <- log(mean(dbinom(data_new[i], 20, sample_theta)))
  lppd_new_m1 <- lppd_new_m1 + lpd_i
}

print(paste("Out-of-sample Deviance for Model 1: ", -2*lppd_new_m1))
```

```
## [1] "Out-of-sample Deviance for Model 1:  50.6677801422507"
```

Model 2

```
lppd_new_m2 <- 0
for (i in 1:length(data_new))
{
  sample_theta <- rbeta(1000, alpha_prime_2, beta_prime_2)
  lpd_i <- log(mean(dbinom(data_new[i], 20, sample_theta)))
  lppd_new_m2 <- lppd_new_m2 + lpd_i
}

print(paste("Out-of-sample Deviance for Model 2: ", -2*lppd_new_m2))
```

```
## [1] "Out-of-sample Deviance for Model 2:  31.5342205167463"
```

Since Model 2 has a lower out-of-sample deviance (31.58) compared to Model 1 (50.27), it indicates that Model 2 is better at predicting new, unseen data despite having a higher in-sample deviance.

This suggests that while Model 1 may overfit the data, Model 2 provides a more robust and generalizable model for predicting new data.

Exercise 1.6 : Perform leave-one-out cross-validation (LOO-CV) to compare model 1 and model 2

Model 1

```r
elppd_model_1 <- 0

for (i in 1:length(data_obs)) {
  y_train <- data_obs[-i]
  y_test <- data_obs[i]

  samples_model_1 <- rbeta(1000, alpha_prime_1, beta_prime_1)

  lpd_1 <- log(mean(dbinom(y_test, size = 20, prob = samples_model_1)))

  elppd_model_1 <- elppd_model_1 + lpd_1
}

print(paste("ELPPD for Model 1:", elppd_model_1))
```

```
## [1] "ELPPD for Model 1: -20.3860084080907"
```

```r
print(paste("In-sample Deviance for Model 1 (LOO-CV):", -2*elppd_model_1))
```

```
## [1] "In-sample Deviance for Model 1 (LOO-CV): 40.7720168161814"
```

Model 2

```r
elppd_model_2 <- 0

for (i in 1:length(data_obs)) {
  y_train <- data_obs[-i]
  y_test <- data_obs[i]

  samples_model_2 <- rbeta(1000, alpha_prime_2, beta_prime_2)

  lpd_2 <- log(mean(dbinom(y_test, size = 20, prob = samples_model_2)))

  elppd_model_2 <- elppd_model_2 + lpd_2
}

print(paste("ELPPD for Model 1:", elppd_model_2))
```

```
## [1] "ELPPD for Model 1: -25.9117394671622"
```

```r
print(paste("In-sample Deviance for Model 1 (LOO-CV):", -2*elppd_model_2))
```

```
## [1] "In-sample Deviance for Model 1 (LOO-CV): 51.8234789343244"
```

Based on the results obtained from leave-one-out cross-validation (LOO-CV), Model 1 shows better predictive performance compared to Model 2.

Part 2 - Marginal likelihood and prior sensitivity

We are given:

- The model's likelihood assumption: y_i ~ $Binomial(N, \theta)$
- Prior assumptions: θ ~ $Beta(a, b)$
- Marginal Likelihood - $ML(n, k, a, b) =$ $ ^{n}C_k $

Exercise 2.1 : Calculate Marginal Likelihood of the models having following

priors

- $k = 2, N = 10$
- List of Prior Assumptions -$Beta(0.1, 0.4), Beta(1, 1), Beta(2, 6), Beta(6, 2), Beta(20, 60), Beta(60, 20)$

```
ML_binomial <- function(k,n,a,b){
ML <- (factorial(n)/(factorial(k)*factorial(n-k)))*(
             factorial(k+a-1)*factorial(n-k+b-1)/factorial(n+a+b-1))
           ML
}

k <- 2
n <- 10

alphas <- c(0.1, 1, 2, 6, 20, 60)
betas <- c(0.4, 1, 6, 2, 60, 20)
```

Marginal Likelihoods for all given priors -

```
for (i in 1:length(alphas))
{
  ml <- ML_binomial(k, n, alphas[i], betas[i])
  print(paste("Marginal Likelihood for Beta (", alphas[i], ",", betas[i], ") : ", ml))
}
```

```
## [1] "Marginal Likelihood for Beta ( 0.1 , 0.4 ) :  0.473956366844962"
## [1] "Marginal Likelihood for Beta ( 1 , 1 ) :  0.0909090909090909"
## [1] "Marginal Likelihood for Beta ( 2 , 6 ) :  0.00472689075630252"
## [1] "Marginal Likelihood for Beta ( 6 , 2 ) :  0.000231386260798025"
## [1] "Marginal Likelihood for Beta ( 20 , 60 ) :  5.07939675334424e-21"
## [1] "Marginal Likelihood for Beta ( 60 , 20 ) :  1.50663033900523e-23"
```

Exercise 2.2 : Estimate the marginal likelihood using Monte Carlo Integration

method.

```r
ML_MarkovChain <- function(a, b, k = 2, n = 10, sample_size = 50000, step_size = 0.08) {
  theta_p <- c(0.4)
  ML <- 0

  i <- 2

  while (i < sample_size) {
    sample_theta <- rnorm(1, theta_p[i-1], step_size)

    if (sample_theta > 0 & sample_theta < 1) {
      post_new <- dbinom(k, n, sample_theta) * dbeta(sample_theta, a, b)
      post_prev <- dbinom(k, n, theta_p[i-1]) * dbeta(theta_p[i-1], a, b)

      proposal_density <- (post_new * dnorm(theta_p[i-1], sample_theta, step_size)) /
                          (post_prev * dnorm(sample_theta, theta_p[i-1], step_size))

      p_str <- min(1, proposal_density)

      if (p_str > runif(1, 0, 1)) {
        theta_p[i] <- sample_theta
        lkl <- dbinom(k, n, sample_theta) * dbeta(sample_theta, a, b) / dnorm(sample_theta, theta_p[i-1], st
ep_size)

        ML <- ML + lkl

        i <- i + 1
      }
    }
  }

  estimated_ml <- ML / sample_size
  return(estimated_ml)
}
```

Performing the Markov Chain for all the given priors -

```r
for (i in 1:length(alphas))
{
  ml <- ML_MarkovChain(alphas[i], betas[i])
  print(paste("Marginal Likelihood for Beta (", alphas[i], ",", betas[i], ") : ", ml))
}
```

```
## [1] "Marginal Likelihood for Beta ( 0.1 , 0.4 ) :  0.0401464975809962"
## [1] "Marginal Likelihood for Beta ( 1 , 1 ) :  0.0903324212441418"
## [1] "Marginal Likelihood for Beta ( 2 , 6 ) :  0.211728869622122"
## [1] "Marginal Likelihood for Beta ( 6 , 2 ) :  0.0103094773551404"
## [1] "Marginal Likelihood for Beta ( 20 , 60 ) :  0.413054477790503"
## [1] "Marginal Likelihood for Beta ( 60 , 20 ) :  0.00116662856649865"
```

One Can see that for large values of a and b, Markov Chain method generates very significant errors.