Assignment 5

Amipriya Anand (220122)

2024-07-09

Part 1: Information-theoretic measures and cross-validation

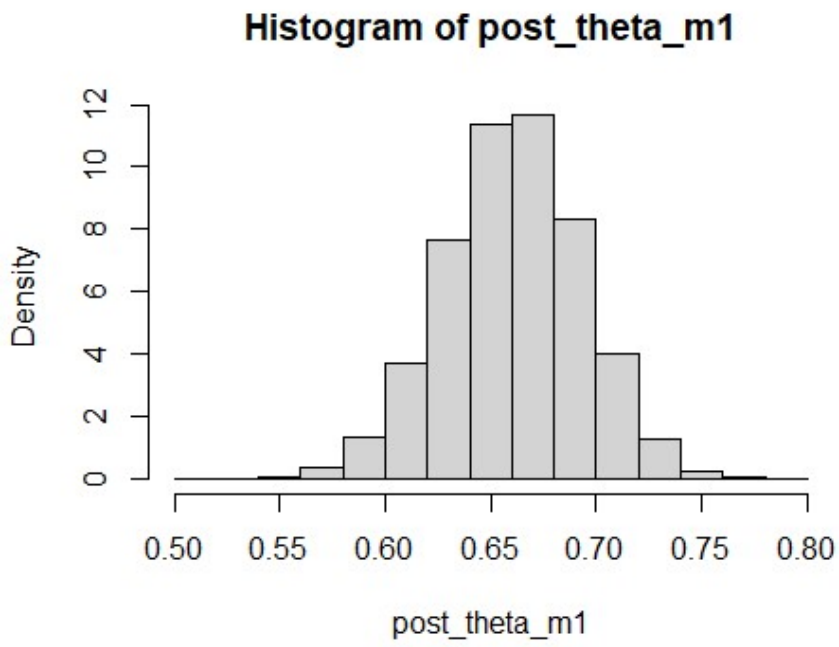
Given 10 independent and identically distributed data points that are assumed to come from a Binomial distribution with sample size 20 and probability of success θ : 10, 15, 15, 14, 14, 14, 13, 11, 12, 16. Two models differing in prior knowledge about the θ parameter. Model 1 has Beta(6,6) prior for θ and, model 2 has Beta(20, 60) prior on θ.

Exercise 1.1: Graph the posterior distribution of ϑ for each model

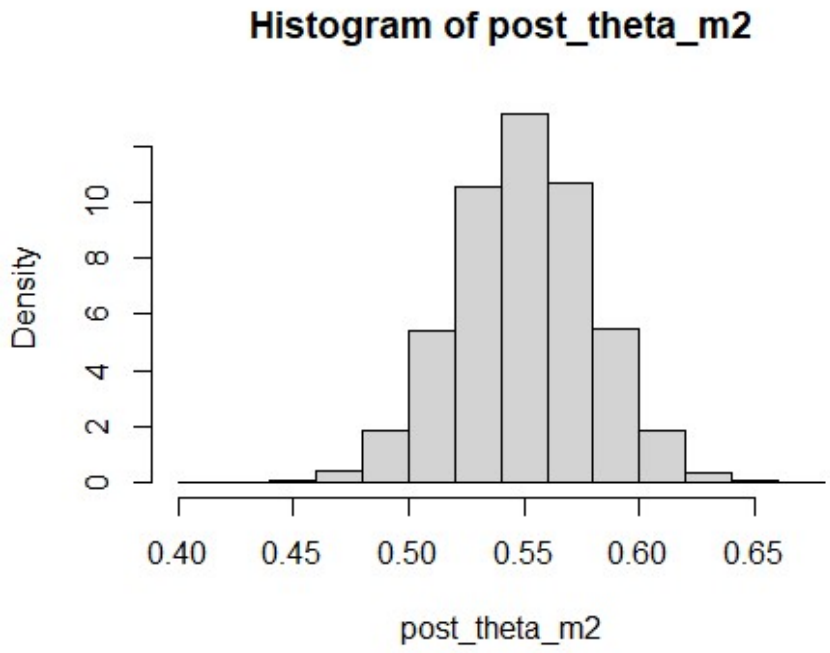
Since here the observerd data is coming from the binomial Distribution and the theta follows the beta Distribution hence, following the binomial-beta prior, thus the posterior follows the beta distribution of Posterior distribution of θ $\theta|y \sim \text{Beta}(a + \sum(i=Nobs, i=1)\ yi, b + Nobs * n - \sum(i=Nobs, i=1)\ yi)$

thus directly,

```
y = c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
Nobs = 10
# for each model-----
# for model 1
a = 6
b = 6
n = 20
post_theta_m1 = rbeta(100000,a+sum(y),b+(Nobs*n)-sum(y))
hist(post_theta_m1,freq = FALSE)
```




```r
# for model 2
a = 20
b = 60
n = 20
post_theta_m2 = rbeta(100000,a+sum(y),b+(Nobs*n)-sum(y))
hist(post_theta_m2,freq = FALSE)
```



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

```
# Compute log pointwise predictive density (lppd) for the models
# for  model 1
a = 6
b = 6
n = 20
lppd_m1 = 0
for(i in 1:Nobs)
{
  sample_theta =rbeta(1000,a+sum(y),b+(Nobs*n)-sum(y))
  lpd_i = log(mean(dbinom(y[i],size = 20,prob = sample_theta)))
  lppd_m1 = lppd_m1 + lpd_i
}
lppd_m1
```

```
## [1] -20.41417
```

```
# for  model 2
a = 20
b = 60
n = 20
lppd_m2 = 0
for(i in 1:Nobs)
{
  sample_theta =rbeta(1000,a+sum(y),b+(Nobs*n)-sum(y))
  lpd_i = log(mean(dbinom(y[i],size = 20,prob = sample_theta)))
  lppd_m2 = lppd_m2 + lpd_i
}
lppd_m2
```

```
## [1] -25.89447
```

Exercise 1.3: Calculate in-sample deviance for each model from the log pointwise predictive density (lppd) computed in 1.2. Why are we calling this in-sample deviance?

Using the formula for in-sample deviance = -2*lppd.

Lesser deviance implies better predictive accuracy

We are calling it "*in-sample deviance*" as it refers to a measure of how well a statistical model fits the observed data points within the sample it was trained on. Specifically, it quantifies the goodness of fit for the model based on the data it has seen during training.

```
# in-sample deviance for both the models-----
# for model 1 :
deviance_m1 = -2*lppd_m1
deviance_m1
```

```
## [1] 40.82833
```

```r
# for model 2 :
deviance_m2 = -2*lppd_m2
deviance_m2
```

```
## [1] 51.78895
```

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

Based on the in-sample deviance, model 1 has lower in-sample deviance and hence model 1 has better fit of the data compared to model 2.

Exercise 1.5: Suppose that you have 5 new data points: [5, 6, 10, 8, 9]. Which of your models is better at predicting new data? You can calculate out-of-sample deviance now to compare y

```r
y_new = c(5, 6, 10, 8, 9)
Nobs = length(y_new)
# calculating lppd of the both models using the new data and then compairing.
# for  model 1
a = 6
b = 6
n = 20
lppd_m1 = 0
for(i in 1:Nobs)
{
  sample_theta =rbeta(1000,a+sum(y_new),b+(Nobs*n)-sum(y_new))
  lpd_i = log(mean(dbinom(y_new[i],size = 20,prob = sample_theta)))
  lppd_m1 = lppd_m1 + lpd_i
}
lppd_m1
```

```
## [1] -10.4823
```

```r
out_of_sample_deviance_m1 = -2*lppd_m1
out_of_sample_deviance_m1
```

```
## [1] 20.96461
```

```r
# for  model 2
a = 20
b = 60
n = 20
lppd_m2 = 0
for(i in 1:Nobs)
{
  sample_theta =rbeta(1000,a+sum(y_new),b+(Nobs*n)-sum(y_new))
  lpd_i = log(mean(dbinom(y_new[i],size = 20,prob = sample_theta)))
  lppd_m2 = lppd_m2 + lpd_i
}
lppd_m2
```

```
## [1] -11.03218
```

```
out_of_sample_deviance_m2 = -2*lppd_m2
out_of_sample_deviance_m2

## [1] 22.06436
```

Clearly, model 1 again having lower value of out-of sample deviance, hence model 1 performs better than model 2.

Exercise 1.6: Now suppose you do not have new data. Perform leave-one-out cross-validation (LOO-CV) to compare model 1 and model 2

```
y = c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
Nobs = 10

# using LOO-CV for both the models
#for model 1 :
a = 6
b = 6
elpd_m1 = 0
for(i in 1:Nobs)
{
  y_train = y[-i]
  y_test = y[i]
  sample_theta =rbeta(1000,a+sum(y_train),b+(Nobs*n)-sum(y_train))
  lpd_i = log(mean(dbinom(y_test,size = 20,prob = sample_theta)))
  elpd_m1 = elpd_m1 + lpd_i
}
elpd_m1

## [1] -22.91949

#for model 1 :
a = 20
b = 60
elpd_m2 = 0
for(i in 1:Nobs)
{
  y_train = y[-i]
  y_test = y[i]
  sample_theta =rbeta(1000,a+sum(y_train),b+(Nobs*n)-sum(y_train))
  lpd_i = log(mean(dbinom(y_test,size = 20,prob = sample_theta)))
  elpd_m2 = elpd_m2 + lpd_i
}
elpd_m2

## [1] -31.52051
```

Using LOO-CV we can say that model 1 having higher elpd(expected log predictive density) out-performs model 2

Part 2: Marginal likelihood and prior sensitivity

Consider a Binomial model with sample size n and probability of success θ and prior on θ is Beta(a,b). The marginal likelihood of the model for k successes will be:
nCk[(k+a−1)!(n−k+b−1)!/(n+a+b−1)!]

```
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
}
```

Exercise 2.1: For k=2 and n=10, calculate marginal likelihood of the models having following priors on ϑ:

– *Beta(0.1,0.4)*

```
a = 0.1
b = 0.4
ML_binomial(k=2,n = 10,a,b)

## [1] 0.4739564
```

– *Beta(1,1)*

```
a = 1
b = 1
ML_binomial(k=2,n = 10,a,b)

## [1] 0.09090909
```

– *Beta(2,6)*

```
a = 2
b = 6
ML_binomial(k=2,n = 10,a,b)

## [1] 0.004726891
```

– *Beta(6,2)*

```
a = 6
b = 2
ML_binomial(k=2,n = 10,a,b)

## [1] 0.0002313863
```

– *Beta(20,60)*

```
a = 20
b = 60
ML_binomial(k=2,n = 10,a,b)
```

```
## [1] 5.079397e-21
```

– *Beta(60,20)*

```
a = 60
b = 20
ML_binomial(k=2,n = 10,a,b)
```

```
## [1] 1.50663e-23
```

Exercise 2.2: Estimate the marginal likelihood of the model given the above prior assumptions using Monte Carlo Integration method.

Lets create a function that will estimate the Marginal Likelihood using the Monte-Carlo Integration Method.

```
ML_MC = function(k,n,a,b)
{
  lkl = c()
  for(i in 1:10000)
  {
    theta_i = rbeta(1,a,b)
    likelihood = dbinom(k,size = n,prob = theta_i)
    lkl = c(lkl,likelihood)
  }
  mean(lkl)
}
```

Now calculating ML using Monte Carlo integration method for each of them-

– *Beta(0.1,0.4)*

```
a = 0.1
b = 0.4
ML_MC(k=2,n = 10,a,b)
```

```
## [1] 0.03908727
```

– *Beta(1,1)*

```
a = 1
b = 1
ML_MC(k=2,n = 10,a,b)
```

```
## [1] 0.09003135
```

– *Beta(2,6)*

```
a = 2
b = 6
ML_MC(k=2,n = 10,a,b)
```

```
## [1] 0.1991219
```

– *Beta(6,2)*

```
a = 6
b = 2
ML_MC(k=2,n = 10,a,b)
```

```
## [1] 0.009762455
```

– *Beta(20,60)*

```
a = 20
b = 60
ML_MC(k=2,n = 10,a,b)
```

```
## [1] 0.2695549
```

– *Beta(60,20)*

```
a = 60
b = 20
ML_MC(k=2,n = 10,a,b)
```

```
## [1] 0.0007859686
```