STA 623 homework 2

Lingyun Shao

Sep. 7, 2018

Problem

The likehood function

$$
\begin{aligned}
& L(X_{n+1}=x|S_{n+1}=j,X^{(n)},S^{(n)}) \\
= \quad & \frac{\int_\Theta L(X_{n+1}=x|S_{n+1}=j,X^{(n)},S^{(n)})L(X^{(n)}|S^{(n)},\theta_j)f(\theta_j)d\theta_j}{\int_\Theta L(X^{(n)}|S^{(n)},\theta_j)f(\theta_j)d\theta_j} \\
= \quad & \frac{\int_\Theta \theta_j^x(1-\theta_j)^{1-x}\binom{n_j}{x_{j+}}\theta_j^{x_{j+}}(1-\theta_j)^{n_j-x_{j+}}f(\theta_j)d\theta_j}{\int_\Theta \binom{n_j}{x_{j+}}\theta_j^{x_{j+}}(1-\theta_j)^{n_j-x_{j+}}f(\theta_j)d\theta_j} \\
= \quad & \frac{\binom{n_j}{x_{j+}}\int_\Theta \theta_j^{x_{j+}+x}(1-\theta_j)^{n_j+1-x_{j+}-x}f(\theta_j)d\theta_j}{\binom{n_j}{x_{j+}}\int_\Theta \theta_j^{x_{j+}}(1-\theta_j)^{n_j-x_{j+}}}
\end{aligned}
$$

- assume $\theta_j \ \sim Beta(\alpha,\beta)$ and solve the above integral.
- generate relevant synthetic in R.
- compare the estimates of two stage Plug-in and Full Bayes.

1. Solution to integral

$$\because \theta_j \sim Beta(\alpha,\beta) \therefore f(\theta_j)=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta_j^{\alpha-1}(1-\theta_j)^{\beta-1}$$

$$
\begin{aligned}
& L(X_{n+1}=x|S_{n+1}=j,X^{(n)},S^{(n)}) \\
= \quad & \frac{\binom{n_j}{x_{j+}}\int_\Theta \theta_j^{x_{j+}+x}(1-\theta_j)^{n_j+1-x_{j+}-x}f(\theta_j)d\theta_j}{\binom{n_j}{x_{j+}}\int_\Theta \theta_j^{x_{j+}}(1-\theta_j)^{n_j-x_{j+}}} \\
= \quad & \frac{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int_\Theta \theta_j^{x_{j+}+x+\alpha-1}(1-\theta_j)^{n_j+\beta-x_{j+}-x}d\theta_j}{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int_\Theta \theta_j^{x_{j+}+\alpha-1}(1-\theta_j)^{n_j+\beta-x_{j+}-1}d\theta_j}
\end{aligned} \tag{1}
$$

The functions of θ_j in both numerator and denominator are apparently kernals of pdf's of some Beta Distribution. By properly rearranging the terms, we can get the function form of a pdf inside the integral, which is equal to 1. $\Gamma(n)=(n-1)!,\ \forall n\in\mathbb{N}^+$. Thus, we have

$$\begin{aligned}
& L(X_{n+1} = x | S_{n+1} = j, X^{(n)}, S^{(n)}) \\
= & \frac{\frac{\Gamma(x_{j+}+x+\alpha)\Gamma(n_j+\beta-x_{j+}-x+1)}{\Gamma(n_j+\alpha+\beta+1)} \int_\Theta \frac{\Gamma(n_j+\alpha+\beta+1)}{\Gamma(x_{j+}+x+\alpha)\Gamma(n_j+\beta-x_{j+}-x+1)} \theta_j^{x_{j+}+x+\alpha-1} (1-\theta_j)^{n_j+\beta-x_{j+}-x} d\theta_j}{\frac{\Gamma(x_{j+}+\alpha)\Gamma(n_j+\beta-x_{j+})}{\Gamma(n_j+\alpha+\beta)} \int_\Theta \frac{\Gamma(n_j+\alpha+\beta)}{\Gamma(x_{j+}+\alpha)\Gamma(n_j+\beta-x_{j+})} \theta_j^{x_{j+}+\alpha-1} (1-\theta_j)^{n_j+\beta-x_{j+}-1} d\theta_j} \\
= & \frac{\frac{\Gamma(x_{j+}+x+\alpha)\Gamma(n_j+\beta-x_{j+}-x+1)}{\Gamma(n_j+\alpha+\beta+1)} \times 1}{\frac{\Gamma(x_{j+}+\alpha)\Gamma(n_j+\beta-x_{j+})}{\Gamma(n_j+\alpha+\beta)} \times 1} \\
= & (x_{j^+} + \alpha + x - 1)_x (n_j + \beta - x_{j^+} - x)_{1-x} \frac{1}{n_j + \alpha + \beta}
\end{aligned}$$

where $(m)_n = m(m-1)\cdots(m-n+1)$ is the falling factorial and $(m)_0 = 0,\ (m)_1 = m$. We notice that $x \in \{0, 1\}$, so the result can be also written as

$$L(X_{n+1} = x | S_{n+1} = j, X^{(n)}, S^{(n)}) = \begin{cases} \frac{x_{j+}+\alpha}{n_j+\alpha+\beta}, & x = 1 \\ \frac{n_j+\beta-x_{j+}}{n_j+\alpha+\beta}, & x = 0 \end{cases}$$

For the undergraduate version, since uniform distribution is just a special case of Beta distribution when $\alpha = 1,\ \beta = 1$, we can get the result by simply substituting α and β with 1.

2. Simulation study in R

In this problem, each patient has a probability distribution of liver status S, $Pr(S = j) = \pi_j,\ j = 1, 2, 3$. Given $S = j$, the corresponding probability of being positve $\theta_j \sim Beta(\alpha_j, \beta_j)$. We need to generate some "training data" of size N, to help us estimate θ_j.

We assume that the proportion of each status $S = j$ in the training data is identical to that in the population, therefore generating $n_j = N * \pi_j$ samples of status $S = j$. Once given θ_j, the test result of this training data set can be derived from a Bernoulli distribution.

To begin with, I defined a likelihood calculating function, which takes the input of $(\pi,\ \theta,\ N,\ \alpha,\ \beta)$ and returns a table showing the likelihood $L(X_{n+1} = x | S_{n+1} = j, X^{(n)}, S^{(n)})$ given all possible (x, j) by MLE and Full Bayes. The α and β in the input are parameter vectors of Beta distributions where $\theta = (\theta_1, \theta_2, \theta_3)^\top$ is generated.

```r
likelihood = function(pi,theta,N,alpha=c(1,1,1),beta=c(1,1,1) ){
  n = round(N*pi) #size for each status
  S = c(1,2,3) #set of status 1=good, 2=Gil, 3=bad
  s = c(rep(S[1],n[1]),rep(S[2],n[2]),rep(S[3],n[3]))
  #indicator of each sample's status 1=good, 2=Gil, 3=bad
  test = rep(0,N) #test result of each training sample
  set.seed(1)
  for (i in 1:3) {
    test[s == i] = rbinom(n[i],1,theta[i])
  }

  #Plug-in/MLE
  df = data.frame(test,s)
  theta.hat = aggregate(test ~ s, df, FUN = 'mean')$test
  #define the function to calculate likelihood
  L1 = function(x,j){
    return(theta.hat[j]^x*(1-theta.hat[j])^(1-x))
  }

  #Full Bayes
  #define the function to calculate likelihood
  L2=function(x,j){
    a = alpha[j]
    b = beta[j]
    nj = sum(s == j)
    xjp = sum(test[s == j])
    if (x == 1) {
      r = (xjp+a)/(nj+a+b)
    } else {
      r = (nj+b-xjp)/(nj+a+b)
    }
    return(r)
  }

  #generating output table
  result = matrix(rep(0,12), nrow = 6)
  it = 0
  rname = NULL
  for(j in 1:3){
    for(x in 0:1){
      it = it+1
      result[it,] = c(L1(x,j), L2(x,j))
      rname = c(rname, paste0('x=',x,',','j=',j))
    }
  }
  colnames(result) = c('Plug-in','Full Bayes')
  rownames(result) = rname
  return(result)
}
```

Based on simulated training data, we can easily get the likelihood by these two methods using the function I defined, likelihood(). Suppose $\pi = (0.2, 0.7, 0.1),\ \theta_j \sim Beta(1, 1),\ and\ \theta = (0.2, 0.5, 0.9),\ N = 100$, we have results as below.

```
pi=c(0.2,0.7,0.1) #proportion of good, Gil, bad
theta=c(0.2,0.5,0.9) #prob of test being positive
N=100 #the total sample size
(l=likelihood(pi,theta,N))
```

```
##           Plug-in Full Bayes
## x=0,j=1 0.8000000 0.77272727
## x=1,j=1 0.2000000 0.22727273
## x=0,j=2 0.5571429 0.55555556
## x=1,j=2 0.4428571 0.44444444
## x=0,j=3 0.0000000 0.08333333
## x=1,j=3 1.0000000 0.91666667
```

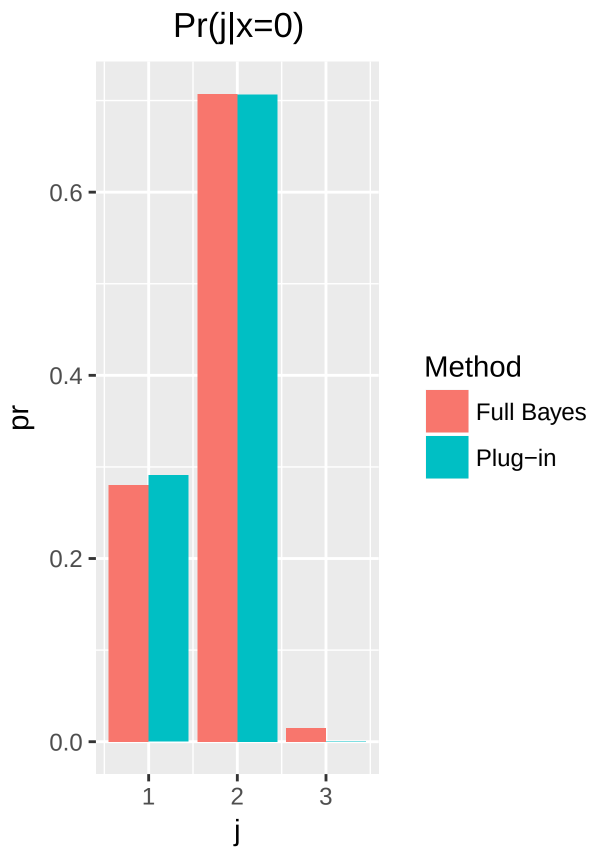
From the probability table, we can see that Plug-in and Full Bayes do generate different likelihood values. More comparison and discussion will be covered in part 3.

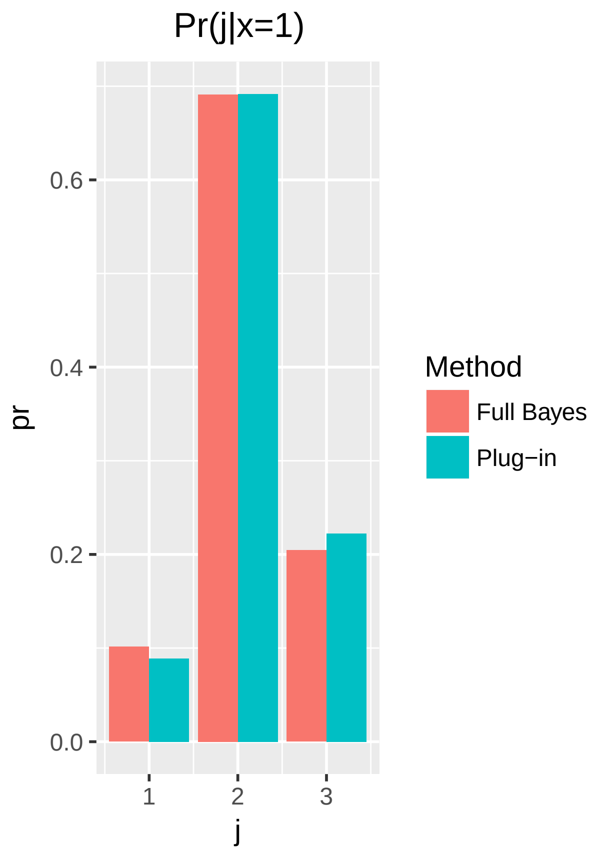
We can calculate the posterior probabilities by two methods using the formula

$$
\begin{aligned}
& Pr(S_{n+1} = j | X_{n+1} = x, X^{(n)}, S^{(n)}) \\
= & \frac{Pr(S_{n+1} = j) L(X_{n+1} = x | S_{n+1} = j, X^{(n)}, S^{(n)})}{\sum_{l=1}^{3} Pr(S_{n+1} = l) L(X_{n+1} = x | S_{n+1} = l, X^{(n)}, S^{(n)})}
\end{aligned}
$$

```r
strl=list(x0=l[c(1,3,5),],x1=l[c(2,4,6),]) #structured likelihood
pos.x0=(strl$x0*pi)/colSums(strl$x0*pi)
pos.x1=(strl$x1*pi)/colSums(strl$x1*pi)
(pos=rbind(pos.x0,pos.x1)) #posterior probabilities
```

```
##              Plug-in Full Bayes
## x=0,j=1 0.29090909 0.28009153
## x=0,j=2 0.70681922 0.70707071
## x=0,j=3 0.00000000 0.01510297
## x=1,j=1 0.08888889 0.10140845
## x=1,j=2 0.69160563 0.69135802
## x=1,j=3 0.22222222 0.20450704
```





We can also draw the bar plot for posterior probabilities to compare these two methods. Since there is no loss function given, if we are using Naive Bayes Classifier, we would probably conclude that the patient is in condition of Gilbert's syndrome ($j = 2$ has the greatest probability) no matter $x = 0$ or $x = 1$, which is apparently a bad choice.

3. Plug-in vs Full Bayes

Back to the likelihood, we can compare these two method.

- Plug-in

$$L \approx \hat{\theta}_j^x (1-\hat{\theta}_j)^{1-x} = \begin{cases} \frac{x_{j+}}{n_j} = \bar{x}_j, & x = 1 \\ \frac{n_j - x_{j+}}{n_j} = 1 - \bar{x}_j, & x = 0 \end{cases}$$

- Full Bayes

$$L = \begin{cases} \frac{x_{j+}+\alpha}{n_j+\alpha+\beta} = \frac{n_j}{n_j+\alpha+\beta}\bar{x}_j + \frac{\alpha+\beta}{n_j+\alpha+\beta}\mu_{0j} = w_1(\bar{x}_j) + w_2(\mu_{0j}), & x = 1 \\ \frac{n_j+\beta-x_{j+}}{n_j+\alpha+\beta} = \frac{n_j}{n_j+\alpha+\beta}(1-\bar{x}_j) + \frac{\alpha+\beta}{n_j+\alpha+\beta}(1-\mu_{0j}) = w_1(1-\bar{x}_j) + w_2(1-\mu_{0j}), & x = 0 \end{cases}$$

where $\bar{x}_j = \frac{x_{j+}}{n_j}$ is the sample mean and $\mu_{0j} = \frac{\alpha}{\alpha+\beta}$ is the prior mean, $w_1 = \frac{n_j}{n_j+\alpha+\beta}$, $w_2 = \frac{\alpha+\beta}{n_j+\alpha+\beta}$ are the weights for sample and prior information.

From the equation above, we can clearly see that the likelihood (or probability) using Full Bayes can be transformed as the weighted average of sample mean and prior mean, which shows the common trick of Bayesian inference, using sample information to update our prior belief.

As the sample size increases, the sample weight approaches 1 and we are more likely to believe the sample information rather than any prior information. Therefore, Plug-in and Full Bayes methods are equivalent given sample size is sufficiently large.

Apart from that the parameters of prior distributions can also influence the results. When the parameters are very large, prior belief comes into dominance. If μ_{0j} is quite different from $\bar{x}_j$, then the results of using these two methods can be largely divergent.

I did some simulation to support my comparison.

3.1 Large sample size

```r
pi=c(0.2,0.7,0.1) #proportion of good, Gil, bad
theta=c(0.2,0.5,0.9) #prob of test being positive
N=10000 #the total sample size
likelihood(pi,theta,N)
```

```
##           Plug-in Full Bayes
## x=0,j=1 0.7960000  0.7957043
## x=1,j=1 0.2040000  0.2042957
## x=0,j=2 0.5001429  0.5001428
## x=1,j=2 0.4998571  0.4998572
## x=0,j=3 0.1040000  0.1047904
## x=1,j=3 0.8960000  0.8952096
```

If we give a very large sample size $N = 10000$, the result reveals that there is little difference between these two methods.

3.2 Large prior weight

```r
pi=c(0.2,0.7,0.1) #proportion of good, Gil, bad
alpha=c(1,10,100)
beta=c(1,5,10)
theta=c(rbeta(1,alpha[1],beta[1]),
        rbeta(1,alpha[2],beta[2]),
        rbeta(1,alpha[3],beta[3]))
N=100 #the total sample size
likelihood(pi,theta,N,alpha,beta)
```

```
##         Plug-in Full Bayes
## x=0,j=1     0.1 0.13636364
## x=1,j=1     0.9 0.86363636
## x=0,j=2     0.4 0.38823529
## x=1,j=2     0.6 0.61176471
## x=0,j=3     0.0 0.08333333
## x=1,j=3     1.0 0.91666667
```

$\theta_1 \sim Beta(1,1)$, $\theta_2 \sim Beta(10,5)$, $\theta_3 \sim Beta(100,10)$. From the result of likelihood, we can see that as parameters of prior distributions grows, the impact of prior information also inflates, leading to a greater divergence between these two methods.

3.3 Population proportion

```r
theta=c(0.2,0.5,0.9) #prob of test being positive
N=100 #the total sample size

#Extreme population
pi=c(0.05,0.9,0.05) #proportion of good, Gil, bad
likelihood(pi,theta,N)
```

```
##           Plug-in Full Bayes
## x=0,j=1 0.8000000  0.7142857
## x=1,j=1 0.2000000  0.2857143
## x=0,j=2 0.5222222  0.5217391
## x=1,j=2 0.4777778  0.4782609
## x=0,j=3 0.0000000  0.1428571
## x=1,j=3 1.0000000  0.8571429
```

```r
#Extreme population
pi=c(0.05,0.5,0.45) #proportion of good, Gil, bad
likelihood(pi,theta,N)
```

```
##            Plug-in Full Bayes
## x=0,j=1 0.80000000 0.71428571
## x=1,j=1 0.20000000 0.28571429
## x=0,j=2 0.48000000 0.48076923
## x=1,j=2 0.52000000 0.51923077
## x=0,j=3 0.04444444 0.06382979
## x=1,j=3 0.95555556 0.93617021
```

```r
#balanced population
pi=c(0.33,0.34,0.33) #proportion of good, Gil, bad
```

```
likelihood(pi,theta,N)
```

```
##             Plug-in Full Bayes
## x=0,j=1 0.81818182 0.80000000
## x=1,j=1 0.18181818 0.20000000
## x=0,j=2 0.50000000 0.50000000
## x=1,j=2 0.50000000 0.50000000
## x=0,j=3 0.03030303 0.05714286
## x=1,j=3 0.96969697 0.94285714
```

By setting different population proportions, the simulation study indicates that extreme population tend to cause divergence. Since extreme population will result in insufficient sample size of some status in our training data, the Plug-in method might lose a lot of accuracy and subsequently leading to the difference between two methods.