STAT 812: Computational Statistics

Laplace Method for Approximating Marginal Likelihood of Gaussian

Longhai Li

March 2020

Contents

1 Laplace Approximation

```
## the generic function for finding laplace approximation of integral of 'f'
## neg_log_f    --- the negative log of the intergrand function
## p0           --- initial value in searching mode
## ...          --- other arguments needed by neg_log_f
bayes_inference_lap <- function(neg_log_f,p0,...)
{   ## looking for the mode and hessian of the log likehood function
    result_min <- nlm(f=neg_log_f,p=p0, hessian=TRUE,...)
    hessian <- result_min$hessian
    neg_log_like_mode <- result_min$minimum


    estimates <- result_min$estimate ## posterior mode
    SIGMA <- solve(result_min$hessian) ## covariance matrix of posterior mode
    sds <- sqrt (diag(SIGMA)) ## standard errors of each estimate
    log_mar_lik <- ## log marginalized likelihood
        - neg_log_like_mode + 0.5 * ( sum(log(2*pi) - log(svd(hessian)$d) ))

    list (estimates = estimates, sds = sds, SIGMA = SIGMA, log_mar_lik = log_mar_lik)
}

## the function for computing log likelihood of normal data
## mu is the unknown mean, and w is the log of standard deviation (sd)
log_lik <- function(x,mu,w)
{   sum(dnorm(x,mu,exp(w),log=TRUE))
}

## the function for computing log prior
```

```r
log_prior <- function(mu,w, mu_0,sigma_mu,w_0,sigma_w)
{   dnorm(mu,mu_0,sigma_mu,log=TRUE) + dnorm(w,w_0,sigma_w,log=TRUE)
}

## the function for computing the negative log of likelihood * prior
neg_log_post <- function(x, theta, mu_0,sigma_mu,w_0,sigma_w)
{   - log_lik(x,theta[1], theta[2]) - log_prior(theta[1],theta[2],mu_0,sigma_mu,w_0,sigma_w)
}



## approximating the log of integral of likelihood * prior
bayes_inference_lap_gaussian <- function(x,mu_0,sigma_mu,w_0,sigma_w)
{   bayes_inference_lap(
                neg_log_post,p0=c(mean(x),log(sqrt(var(x)))),
                x=x,mu_0=mu_0,sigma_mu=sigma_mu,w_0=w_0,sigma_w=sigma_w
    )
}
```

2 Mid-point Rule

```r
## the function for computing log likelihood of normal data
log_lik <- function(x,mu,w)
{   sum(dnorm(x,mu,exp(w),log=TRUE))
}

## the function for computing log prior
log_prior <- function(mu,w, mu_0,sigma_mu,w_0,sigma_w)
{   dnorm(mu,mu_0,sigma_mu,log=TRUE) + dnorm(w,w_0,sigma_w,log=TRUE)
}

## the function for computing the unormalized log posterior
## given transformed mu and w
log_post_tran <- function(x, mu_t, w_t, mu_0,sigma_mu,w_0,sigma_w)
{
    #log likelihood
    log_lik(x,logi(mu_t), logi(w_t)) +
    #log prior
    log_prior(logi(mu_t), logi(w_t), mu_0,sigma_mu,w_0,sigma_w) +
    #log derivative of transformation
    log_der_logi(mu_t) + log_der_logi(w_t)
}

## the logistic function for transforming (0,1) value to (-inf,+inf)
logi <- function(x)
{  log(x) - log(1-x)
}

## the log derivative of logistic function
log_der_logi <- function(x)
{  -log(x) - log(1-x)
```

```
}

## the generic function for approximating 1-D integral with midpoint rule
## the logarithms of the function values are passed in
## the log of the integral result is returned

## log_f  --- a function computing the logarithm of the integrant function
## range  --- the range of integral varaible, a vector of two elements
## n      --- the number of points at which the integrant is evaluated
## ...    --- other parameters needed by log_f
log_int_mid <- function(log_f, range, n,...)
{   if(range[1] >= range[2])
        stop("Wrong ranges")
    h <- (range[2]-range[1]) / n
    v_log_f <- sapply(range[1] + (1:n - 0.5) * h, log_f,...)
    log_sum_exp(v_log_f) + log(h)
}

## a function computing the sum of numbers represented with logarithm
## lx     --- a vector of numbers, which are the log of another vector x.
## the log of sum of x is returned
log_sum_exp <- function(lx)
{   mlx <- max(lx)
    mlx + log(sum(exp(lx-mlx)))
}

## a function computing the normalization constant
log_mar_gaussian_mid <- function(x,mu_0,sigma_mu,w_0,sigma_w,n)
{
    ## function computing the normalization constant of with mu_t fixed
    log_int_gaussian_mu <- function(mu_t)
    {   log_int_mid(log_f=log_post_tran,range=c(0,1),n=n,
                    x=x,mu_t=mu_t,mu_0=mu_0,sigma_mu=sigma_mu,
                    w_0=w_0,sigma_w=sigma_w)
    }

    log_int_mid(log_f=log_int_gaussian_mu,range=c(0,1), n=n)
}
```

3 Naive Monte Carlo For Computinglog Marginalized Likelihood

```
## we use Monte Carlo method to debug the above function
log_mar_gaussian_mc <- function(x,mu_0,sigma_mu,w_0,sigma_w,iters_mc)
{
    ## draw samples from the priors
    mus <- rnorm(iters_mc,mu_0,sigma_mu)
    ws <- rnorm(iters_mc,w_0,sigma_w)
    one_log_lik <- function(i)
    {  log_lik(x,mus[i],ws[i])
    }
    v_log_lik <- sapply(1:iters_mc,one_log_lik)
```

```r
    log_sum_exp(v_log_lik) - log(iters_mc)
}
```

4 Testing and comparing Laplace and Mid-point Approximation

```r
## test with a data set with mean 5
x <- rnorm(50, mean = 5)
## True values for mu and log (sigma)
5; log(1)
```

```
## [1] 5
```

```
## [1] 0
```

```r
bayes_inference_lap_gaussian(x,0,100,0,5)
```

```
## $estimates
## [1] 4.85232496 0.05647914
##
## $sds
## [1] 0.14963842 0.09998775
##
## $SIGMA
##              [,1]        [,2]
## [1,] 2.239166e-02 4.63331e-06
## [2,] 4.633310e-06 9.99755e-03
##
## $log_mar_lik
## [1] -84.1901
```

```r
## compare with naive Monte carlo and midpoint rule for computing log mar lik.
log_mar_gaussian_mc(x,0,100,0,5,1000000)
```

```
## [1] -84.06363
```

```r
log_mar_gaussian_mid(x,0,100,0,5,100)
```

```
## [1] -86.50798
```

```r
x <- rnorm(50, mean = -5)
## True values for mu and log (sigma)
-5; log(1)
```

```
## [1] -5
```

```
## [1] 0
```

```r
bayes_inference_lap_gaussian(x,0,100,0,5)
```

```
## $estimates
## [1] -4.77589516 -0.06634202
##
## $sds
## [1] 0.1323435 0.0999926
##
## $SIGMA
```

```
##               [,1]         [,2]
## [1,] 1.751480e-02 1.115655e-06
## [2,] 1.115655e-06 9.998521e-03
##
## $log_mar_lik
## [1] -78.16937
```

```r
log_mar_gaussian_mc(x,0,100,0,5,1000000)
```

```
## [1] -78.17463
```

```r
log_mar_gaussian_mid(x,0,100,0,5,100)
```

```
## [1] -82.97552
```

```r
x <- rnorm(50, mean = -50, sd = 4)
## True values for mu and log (sigma)
5; log(4)
```

```
## [1] 5
```

```
## [1] 1.386294
```

```r
bayes_inference_lap_gaussian(x,0,100,0,5)
```

```
## $estimates
## [1] -48.741167   1.294047
##
## $sds
## [1] 0.51583029 0.09994116
##
## $SIGMA
##               [,1]         [,2]
## [1,] 2.660809e-01 2.641903e-05
## [2,] 2.641903e-05 9.988236e-03
##
## $log_mar_lik
## [1] -145.0073
```

```r
log_mar_gaussian_mc(x,0,100,0,5,10000000)
```

```
## [1] -145.0028
```

```r
log_mar_gaussian_mid(x,0,100,0,5,100)
```

```
## [1] -268.7777
```

```r
x <- rnorm(50, mean = -50, sd = 10)
## True values for mu and log (sigma)
5; log(10)
```

```
## [1] 5
```

```
## [1] 2.302585
```

```r
bayes_inference_lap_gaussian(x,0,100,0,5)
```

```
## $estimates
## [1] -47.812074   2.203725
##
## $sds
```

```
## [1] 1.28099265 0.09991411
##
## $SIGMA
##             [,1]        [,2]
## [1,] 1.640942162 0.000157629
## [2,] 0.000157629 0.009982829
##
## $log_mar_lik
## [1] -189.6591
```

```r
log_mar_gaussian_mc(x,0,100,0,5,1000000)
```

```
## [1] -189.6675
```

```r
log_mar_gaussian_mid(x,0,100,0,5,100)
```

```
## [1] -268.6024
```

We see that mid point rule may not work well. The reason is that in applying mid-point numerical quadrature, we use "logistic" transformation which assigns more points around "zero" but the integrant function has high density in the region around -50 for μ, and $\log(10) = 2.3$ for w.