

# STAT 812: Computational Statistics

## Midpoint Rule Approximating Marginal Likelihood of Gaussian

Longhai Li

March 2020

### Contents

<b>1</b>	<b>Using Midpoint Rule to Compute Marginal Likelihood of t distribution</b>	<b>1</b>
<b>2</b>	<b>Test with simulated datasets</b>	<b>3</b>
2.1	Checking the accuracy of numerical quadrature . . . . .	3
2.2	Comparing log marginal likelihoods of different models . . . . .	4

```
library ("metRology")
```

```
##
```

```
## Attaching package: 'metRology'
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##      cbind, rbind
```

## 1 Using Midpoint Rule to Compute Marginal Likelihood of t distribution

```
## the function for computing log likelihood of normal data
```

```
log_lik <- function(x,mu,w,df=Inf)
{
  sum(dt.scaled(x,df,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 unnormalized log posterior
```

```
## given transformed mu and w
```

```
log_post_tran <- function(x, mu_t, w_t, mu_0,sigma_mu,w_0,sigma_w, df=Inf)
{
  #log likelihood
  log_lik(x,logi(mu_t), logi(w_t),df) +
  #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 variable, 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_marlik_mid <- function(x,mu_0,sigma_mu,w_0,sigma_w, n, df=Inf)
{
  ## 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, df=df)
  }

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

## we use Monte Carlo method to debug the above function
log_marlik_mc <- function(x,mu_0,sigma_mu,w_0,sigma_w,itters_mc, df=Inf)
{

```

```

## 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], df)
}
v_log_lik <- sapply(1:iters_mc,one_log_lik)
log_sum_exp(v_log_lik) - log(iters_mc)
}

```

## 2 Test with simulated datasets

### 2.1 Checking the accuracy of numerical quadrature

```

x <- rt.scaled(50, mean=2, sd = 2, df=2)
log_marlik_mid(x,0,10,0,10,100)

```

```
## [1] -152.7519
```

```
log_marlik_mc(x,0,10,0,10,100000)
```

```
## [1] -152.5442
```

Another test

```

x <- rt.scaled(100, mean=2, sd = 2, df=Inf)
log_marlik_mid(x,0,10,0,10,100)

```

```
## [1] -217.3958
```

```
log_marlik_mc(x,0,10,0,10,100000)
```

```
## [1] -217.4403
```

```
## looking at the convergence
```

```

for(i in seq(10,90,by=10))
{ cat("n = ",i,",")
  cat(" Estimated Log Marginal Likelihood =",
      log_marlik_mid(x,0,10,0,10,i),"\\n")
}

```

```

## n = 10 , Estimated Log Marginal Likelihood = -216.3278
## n = 20 , Estimated Log Marginal Likelihood = -217.4292
## n = 30 , Estimated Log Marginal Likelihood = -217.4065
## n = 40 , Estimated Log Marginal Likelihood = -217.3953
## n = 50 , Estimated Log Marginal Likelihood = -217.3958
## n = 60 , Estimated Log Marginal Likelihood = -217.3958
## n = 70 , Estimated Log Marginal Likelihood = -217.3958
## n = 80 , Estimated Log Marginal Likelihood = -217.3958
## n = 90 , Estimated Log Marginal Likelihood = -217.3958

```

## 2.2 Comparing log marginal likelihoods of different models

### 2.2.1 Comparing Priors

```
x <- rnorm(100)
log_marlik_mid(x,mu_0=0,sigma_mu=1,w_0=0,sigma_w=1,100)

## [1] -134.758
log_marlik_mid(x,mu_0=0,sigma_mu=0.1,w_0=0,sigma_w=1,100)

## [1] -132.8738
log_marlik_mid(x,mu_0=0,sigma_mu=0.01,w_0=0,sigma_w=1,100)

## [1] -133.482
log_marlik_mid(x,mu_0=-2,sigma_mu=0.1,w_0=0,sigma_w=1,100)

## [1] -215.9492
log_marlik_mid(x,mu_0=0,sigma_mu=10,w_0=0,sigma_w=1,100)

## [1] -137.0543
log_marlik_mid(x,mu_0=0,sigma_mu=100,w_0=0,sigma_w=1,100)

## [1] -139.3568
log_marlik_mid(x,mu_0=0,sigma_mu=1000000,w_0=0,sigma_w=1,100)

## [1] -148.5671
log_marlik_mid(x,mu_0=-5,sigma_mu=10,w_0=0,sigma_w=1,100)

## [1] -137.1827
log_marlik_mid(x,mu_0=-5,sigma_mu=1,w_0=0,sigma_w=1,100)

## [1] -147.4969
```

### 2.2.2 Comparing Models for Data

Data from Normal\*

```
x <- rt.scaled(100, mean=2, sd = 2, df=Inf)
log_marlik_mid(x,0,10,0,10,100, df = Inf)

## [1] -235.0123
log_marlik_mid(x,0,10,0,10,100, df = 2)

## [1] -236.9174
log_marlik_mid(x,0,0.1,0,10,100, df = Inf) # if prior is too narrow

## [1] -264.6732
log_marlik_mid(x,0,100,0,100,100, df = Inf) # if prior is too diffuse

## [1] -239.5856
```

```
log_marlik_mid(x,0,1000,0,1000,100, df = Inf) # if prior is too diffuse
```

```
## [1] -244.1904
```

Data from t

```
x <- rt.scaled(100, mean=2, sd = 2, df=2)
```

```
log_marlik_mid(x,0,10,0,10,100, df = Inf)
```

```
## [1] -325.0373
```

```
log_marlik_mid(x,0,10,0,10,100, df = 2)
```

```
## [1] -263.7216
```

```
log_marlik_mid(x,0,0.1,0,10,100, df =2) # if prior is too narrow
```

```
## [1] -282.0241
```

```
log_marlik_mid(x,0,100,0,100,100, df = 2) # if prior is too diffuse
```

```
## [1] -268.307
```

```
log_marlik_mid(x,0,1000,0,1000,100, df = 2) # if prior is too diffuse
```

```
## [1] -272.912
```

We see that although the prior impact marginal likelihood, the error in mis-specification in data model can be still detected.

#### A Case when numerical quadrature fails

```
x <- rt.scaled(100, mean=50, sd = 2, df=Inf)
```

```
log_marlik_mid(x,0,100,0,100,100, df = Inf)
```

```
## [1] -535.0219
```

```
log_marlik_mc(x,0,100,0,100,10000, df = Inf)
```

```
## [1] -285.2609
```

What has gone wrong? The inverse-logistic transformation maps most points between (0,1) to the region around 0.5. But the likelihood function has its mode around 50!