Hybrid Approximation Demo

Eric Chuu

4/26/2022

Bayesian Linear Regression Demo

Consider the Bayesian Linear Regression setup, where the parameters are $\left(\beta, \sigma^2\right) \in \mathbb{R}^{p+1}$. We place a multivariate normal inverse gamma (MVN-IG) prior on $\left(\beta, \sigma^2\right)$, and we want to compute the marginal likelihood. This has the following form:

$$\int \mathcal{N}\left(y \mid X\beta, \sigma^2 I_p\right)\ \mathcal{N}\left(\beta \mid \mu_\beta, \sigma^2 V\right)\ \mathcal{IG}\left(\sigma^2 \mid a_0, b_0\right)\ d\beta\ d\sigma^2$$

Define the following hyperparameter settings.

```r
library(dplyr)
set.seed(123)
d = 20                         # dimension of (beta, sigmasq)
n = 200                        # sample size
J = 100                        # number of posterior samples
p       = d - 1                # dimension of beta
mu_beta = rep(0, p)            # prior mean for beta
V_beta  = diag(1, p)           # prior scaled precision matrix for beta
a_0     = 1                    # prior shape for sigmasq
b_0     = 0.5                  # prior scale for sigmasq
```

Next, initialize the true parameters and generate the data.

```r
## initalize true parameters
sigmasq = 4
beta    = sample(-10:10, p, replace = T)

## generate data using true parameters
X = matrix(rnorm(n * p), n, p)                # (n x p) design matrix
eps = rnorm(n, mean = 0, sd = sqrt(sigmasq))  # (n x 1) vector of residuals
y = X %*% beta + eps                          # (n x 1) response vector
```

The posterior distribution is also MVN-IG, and so we can compute the posterior parameters.

```r
### compute posterior parameters
V_beta_inv = solve(V_beta)
V_n_inv = t(X) %*% X + V_beta_inv

V_n  = solve(V_n_inv)                                # (p x p)
mu_n = V_n %*% (t(X) %*% y + V_beta_inv %*% mu_beta) # (p x 1)
a_n =  a_0 + n / 2
b_n =  c(b_0 + 0.5 * (t(y) %*% y +
                        t(mu_beta) %*% V_beta_inv %*% mu_beta -
                        t(mu_n) %*% V_n_inv %*% mu_n))

# store prior, posterior parameters
params = list(V_beta = V_beta, V_beta_inv = V_beta_inv, mu_beta = mu_beta,
              a_0 = a_0, b_0 = b_0,
              V_n = V_n, V_n_inv = V_n_inv, mu_n = mu_n,
              a_n = a_n, b_n = b_n,
              y = y, X = X, n = n, d = d, p = p)
```

True value of marginal likelihood

The true marginal likelihood is available in closed form and can be computed as follows.

```
# calculate true marginal likelihood
logML = function(params) {
  with(params,
       a_0 * log(b_0) + lgamma(a_n) + 0.5 * hybrid::log_det(V_n) -
         0.5 * hybrid::log_det(V_beta) - n / 2 * log(2 * pi) - lgamma(a_0) -
         a_n * log(b_n))
} # end of logML() function
logML(params)
```

```
## [1] -520.8286
```

Hybrid Estimator

In order to use the Hybrid estimator, we first need to be able to sample from the posterior distribution. We define the following sampling function.

```
# sample_beta(): sample from [ beta | sigmasq, y ]
sample_beta = function(sigmasq, params) {
  mvtnorm::rmvnorm(1, mean = params$mu_n, sigma = sigmasq * params$V_n)
}

# sample_post(): sample from [ beta, sigmasq | y ]
# draw J samples from the posterior distribution
sample_post = function(J, params) {
  sigmasq_post = with(params, MCMCpack::rinvgamma(J, shape = a_n, scale = b_n))
  beta_post = t(sapply(sigmasq_post, sample_beta, params = params))
  data.frame(beta_post, sigmasq_post)
}
```

Then, we can draw samples from the posterior distributiont, where each posterior sample is stored row-wise, where the first p elements correspond to β, and the $(p+1)$-th element corresponds to σ^2.

```
sample_post(5, params)
```

```
##         X1       X2       X3        X4         X5       X6         X7        X8
## 1 3.908025 8.094329 2.671556 -7.833475 -1.0352228 7.069316 -0.3736885 -6.163005
## 2 4.052609 8.122981 3.253917 -7.818292 -0.5479845 6.889079  0.1047297 -6.187769
## 3 4.237387 8.194975 2.887780 -7.519853 -0.4379353 6.877620 -0.1101951 -6.124606
## 4 4.109094 8.326545 3.001456 -8.052817 -1.0247498 7.089352 -0.1002584 -6.199744
## 5 3.996480 8.422861 3.092522 -7.872486 -0.6732528 6.645359 -0.3262611 -6.218917
##         X9      X10       X11      X12       X13       X14       X15       X16
## 1 9.329833 2.728233 -6.501840 8.017752 -2.116835 -8.255740 -3.061497 -3.775831
## 2 9.189641 3.212478 -5.865150 7.820110 -2.087525 -7.978597 -2.643413 -3.640037
## 3 9.250524 2.859082 -6.327013 7.841689 -1.729985 -7.648799 -3.030143 -3.723449
## 4 9.621268 3.065881 -6.215700 8.150258 -1.854625 -8.005705 -2.849811 -3.874725
## 5 9.248972 3.137977 -6.196518 7.653511 -1.772689 -8.216646 -2.846375 -4.133297
##          X17       X18      X19 sigmasq_post
## 1 -1.2598972 -1.993132 7.775803     6.143389
## 2 -1.3333428 -2.157438 7.991492     5.782808
## 3 -0.8319245 -1.751335 8.006560     6.162643
## 4 -1.0133865 -1.806241 7.960834     5.484881
## 5 -1.3280252 -1.846314 7.894712     5.008939
```

We also need to evaluate the negative log posterior. For the hybrid-ep estimator, we also need the gradient and hessian of the negative log posterior as well. We supply all of these below.

```r
# psi(): negative log posterior
psi = function(u, params) {
  p = params$p
  # extract beta, sigmasq from posterior sample
  beta    = unname(unlist(u[1:p])) # (p x 1)
  sigmasq = unname(unlist(u[p+1])) # (1 x 1)

  # compute log posterior
  logpost = with(params,
                 sum(dnorm(y, X %*% beta, sqrt(sigmasq), log = T)) +
                   mvtnorm::dmvnorm(beta, mu_beta, sigmasq * V_beta, log = T) +
                   log(MCMCpack::dinvgamma(sigmasq, shape = a_0, scale = b_0)))
  return(-logpost)
} # end of psi() function

# For hybrid-ep estimator -- define gradient, hessian functions
grad = function(u, params) { pracma::grad(psi, x0 = u, params = params) }
hess = function(u, params) { pracma::hessian(psi, x0 = u, params = params) }
```

Now, we can proceed to compute both the vanilla hybrid estimator and the hybrid-ep estimator.

```r
# sample from posterior
samps = sample_post(J, params)
# evaluate the negative log posterior
u_df = hybrid::preprocess(samps, d, params)

# compute vanilla hybrid estimator
hybrid::hybml_const(u_df)$logz
```

```
## [1] -516.8085
```

```r
# compute hybrid-ep estimator
hybrid::hybml(u_df, params, psi, grad, hess)$logz
```

```
## [1] -520.6006
```