Lab week 11 MAST90125: Bayesian Statistical learning

Variational Bayes.

In lecture 20, we looked at (mean-field) Variational Bayes as a method to find approximate posterior distributions for sub-blocks of the parameter vector $\boldsymbol{\theta}$, for the model

$$\begin{aligned} p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{u}, \tau_e) &= \mathcal{N}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \frac{1}{\tau_e}\mathbf{I}_n) \\ p(\boldsymbol{\beta}) &\propto 1 \\ p(\mathbf{u}) &= \mathcal{N}(\mathbf{0}_q, \frac{1}{\tau_e}\mathbf{K}) \\ p(\tau_e) &= \text{Ga}(\alpha_e, \gamma_e) \\ p(\tau_u) &= \text{Ga}(\alpha_u, \gamma_u) \end{aligned}$$

Instructions for lab

Download `farmdata.csv` and `Kmat.csv` from Canvas.

Re-format the code given in lecture 20 for performing Variational Bayes so that rather than constructing approximate posteriors,

- $Q(\boldsymbol{\beta})$
- $Q(\mathbf{u})$
- $Q(\tau_e)$
- $Q(\tau_u)$

you determine the approximate posteriors

- $Q(\boldsymbol{\beta}, \mathbf{u})$
- $Q(\tau_e)$
- $Q(\tau_u)$.

Compare the performance of the new blocking structure to that used in class.

Hints

The R code used to implement Variational Bayes in lecture 20 is given below:

```
#Arguments are
#epsilon: accuracy cut-off.
#iter: no of iterations
#Kinv: inverse of K, where p(u) = N(0,\sigma^2_u K)
#Z: Predictor matrix for random effects
#X: Predictor matrix for fixed effects
#y: response vector
#taue_0: initial guess for residual precision.
#tauu_0: initial guess for random effect precision
#a.u, g.u: hyper-parameters of gamma prior for tauu
#a.e, g.e: hyper-parameters of gamma prior for taue

#Output are final estimates, plus iteration number when convergence was reached.
VB.mm<-function(epsilon,iter,Kinv,Z,X,y,taue_0,tauu_0,u0,beta0,a.e,g.e,a.u,g.u){
  n<-dim(X)[1]
  p<-dim(X)[2]
  q<-dim(Z)[2]
  ZTZ<-crossprod(Z)
  ZTY<-crossprod(Z,y)
  XTY<-crossprod(X,y)
  ZTX<-crossprod(Z,X)
  XTX<-crossprod(X)
  XTXinv<-solve(XTX)

  for(i in 1:iter){
  Vu <-solve(taue_0*ZTZ+tauu_0*Kinv) #update Var(u)
  u  <-taue_0*Vu%*%(ZTY-ZTX%*%beta0) #update E(u)
  Vb <-XTXinv/taue_0                 #update Var(beta)
  b  <-XTXinv%*%(XTY-t(ZTX)%*%u)     #update E(beta)
  TrKinvu <- sum(diag(Kinv%*%Vu))
  uKinvu  <- t(u)%*%Kinv%*%u
  tauu    <- (a.u+0.5*q)/(g.u+0.5*as.numeric(uKinvu)+0.5*TrKinvu)
  tauu    <- as.numeric(tauu)
  err     <- y - X%*%b - Z%*%u
  TrXTXb  <- sum(diag(XTX%*%Vb))
  TrZTZu  <- sum(diag(ZTZ%*%Vu))
  taue    <- (a.e+0.5*n)/(g.e+0.5*sum(err^2)+0.5*TrXTXb+0.5*TrZTZu)
  taue    <- as.numeric(taue)

  if(i > 1){
  diffb  <- sqrt((b-beta0)^2)/(abs(b)+0.01)
  diffu  <- sqrt((u-u0)^2)/(abs(u)+0.01)
  diffte <- abs(taue_0-taue)/(taue+0.01)
  difftu <- abs(tauu_0-tauu)/(tauu+0.01)
  diffvb <- sqrt((diag(Vb0) - diag(Vb))^2)/(diag(Vb))
  diffvu <- sqrt((diag(Vu0) - diag(Vu))^2)/(diag(Vu))
  diff.all<-c(diffb,diffu,diffte,difftu,diffvb,diffvu)
  if(max(diff.all) < epsilon) break
  }
  Vu0 <- Vu;u0<-u;Vb0<-Vb;beta0<-b;taue_0<-taue;tauu_0<-tauu
  #Calculate relative change.
  }
```

```
  taue.param<-c((a.e+0.5*n),(g.e+0.5*sum(err^2)+0.5*TrXTXb+0.5*TrZTZu))
  tauu.param<-c((a.u+0.5*q),(g.u+0.5*uKinvu+0.5*TrKinvu))
  param<-list(b,Vb,u,Vu,taue.param,tauu.param,i)
  names(param)<-c('beta_mean','beta_var','u_mean','u_var','tau_e','tau_u','iter')
  return(param)
}
```

The R code used to implement Gibbs sampling for this mixed model in lecture 20 is given below:

```
#Arguments are
#iter: no of iterations
#Z: Predictor matrix for random effects
#X: Predictor matrix for fixed effects
#y: response vector
#burnin: number of initial iterations to discard.
#taue_0: initial guess for residual precision.
#tauu_0: initial guess for random effect precision
#Kinv: inverse of K, where p(u) = N(0,\sigma^2_u K)
#a.u, b.u: hyper-parameters of gamma prior for tauu
#a.e, b.e: hyper-parameters of gamma prior for taue

normalmm.Gibbs<-function(iter,Z,X,y,burnin,taue_0,tauu_0,Kinv,a.u,b.u,a.e,b.e){
n   <-length(y) #no. observations
p   <-dim(X)[2] #no of fixed effect predictors.
q   <-dim(Z)[2] #no of random effect levels
tauu<-tauu_0
taue<-taue_0
beta0<-rnorm(p)
u0   <-rnorm(q,0,sd=1/sqrt(tauu))

#Building combined predictor matrix.
W<-cbind(X,Z)
WTW <-crossprod(W)
WTy <-crossprod(W,y)
library(mvtnorm)

#storing results.
par <-matrix(0,iter,p+q+2)
#Calculating log predictive densities
lppd<-matrix(0,iter,n)

#Create modified identity matrix for joint posterior.
I0  <-diag(p+q)
diag(I0)[1:p]<-0
I0[-c(1:p),-c(1:p)]  <-Kinv

for(i in 1:iter){
#Conditional posteriors.
  uKinvu <- t(u0)%*%Kinv%*%u0
  uKinvu <-as.numeric(uKinvu)
tauu <-rgamma(1,a.u+0.5*q,b.u+0.5*uKinvu)
#Updating component of normal posterior for beta,u
Prec <-WTW + tauu*I0/taue
```

```
P.mean<- solve(Prec)%*%WTy
P.var <-solve(Prec)/taue
betau <-rmvnorm(1,mean=P.mean,sigma=P.var)
betau <-as.numeric(betau)
err   <- y-W%*%betau
taue  <-rgamma(1,a.e+0.5*n,b.e+0.5*sum(err^2))
#storing iterations.
par[i,]<-c(betau,1/sqrt(tauu),1/sqrt(taue))
beta0  <-betau[1:p]
u0     <-betau[p+1:q]
lppd[i,]= dnorm(y,mean=as.numeric(W%*%betau),sd=1/sqrt(taue))
}

lppd      = lppd[-c(1:burnin),]
lppdest   = sum(log(colMeans(lppd)))        #Estimating lppd for whole dataset.
pwaic2    = sum(apply(log(lppd),2,FUN=var)) #Estimating effective number of parameters.
par <-par[-c(1:burnin),]
colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),'sigma_b','sigma_e')
mresult<-list(par,lppdest,pwaic2)
names(mresult)<-c('par','lppd','pwaic')
return(mresult)
}
```

Any other code you may use can be modified from the code given in Lecture 20. Possible things you may want to check include:

- convergence of Gibbs sampler
- comparing the empirical and approximate distributions using density plots.