Examples with VB, EM and MCMC algorithms

Variational Bayes is a deterministic algorithm that can address the computational limitations of sampling-based algorithms (e.g. Gibbs sampling or Metropolis Hastings) by using approximate inference (ATTIAS 2000; JAAKKOLA 2001). The approach was initially proposed in the machine learning literature (JORDAN et al. 1999); since then, the method has been applied for image data analysis and graphical modeling (HERMOSILLO et al. 2002; BLEI and JORDAN 2006). Recently, VB has been used in many applications in genetics studies, for instance to perform GWAS (LOGSDON et al. 2010; CARBONETTO and STEPHENS 2012; LOH et al. 2015), in the estimation of QTL effects with epistasis (LI and SILLANPÄÄ 2012), for heritability estimation with the G-BLUP (ARAKAWA 2014), for multi-trait analysis and prediction (HAYASHI and IWATA 2013), and for efficient inference of population structure (RAJ et al. 2014). The VB framework has been used to implement methods that perform shrinkage (LI and SILLANPÄÄ 2012; ARAKAWA 2014) and a combination of variable selection and shrinkage (CARBONETTO and STEPHENS 2012; HAYASHI and IWATA 2013) at an individual marker level.

MCMC gives approximate solutions to the exact posterior. If the algorithm converges the Monte Carlo error diminishes as the number of MCMC samples increases; however, in practice the number of MCMC samples used for inferences is finite and MCMC estimates are only approximations to the true posterior estimates. In contrast, VB can arrive at exact solutions to an approximate posterior. Deriving VB estimates usually involves an optimization and update procedure similar to the ones involved in the Expectation-Maximization (EM) algorithm (GELMAN et al. 2004; BISHOP 2006). Iterations in EM ultimately result in convergence to (local) MAP estimates (posterior modes) of the parameters. In contrast, iterations in VB result in an approximation of a closed-form density function that is close to the posterior, rather than of the posterior itself.

In a Bayesian model, the joint posterior density is given as the product of the likelihood function and the prior, i.e. $p(\beta|y) \propto p(y|\beta)p(\beta)$, for a p-dimensional parameter vector β and data-vector y. VB is applied when it is difficult to draw samples from the posterior $p(\beta|y)$ and it is easier to handle some class of approximating distributions $q(\beta|\phi)$ which is used to approximate the true posterior. Here, ϕ represent the parameters of the variational approximation. In principle, VB finds $q(\beta|\phi)$ such that the Kullback-Leibler (KL) distance (COVER and THOMAS 2012) between the approximating density and the true posterior density, $p(\beta|y)$, is minimized. A standard approach in VB, referred as to mean-field approximation, is to structure $q(\beta)$ such that $q(\beta) = \prod_{j=1}^{p}(q_j(\beta_j|\phi_j)]$ (BISHOP 2006). If the parameter vector β comprises p regression coefficients (e.g. marker effects), this condition assumes that the regression coefficients are independent of each other at $q(\beta)$. This factorized form lends VB its computational speed since time complexity now varies linearly with the number of markers (LOGSDON et al. 2010; CARBONETTO and STEPHENS 2012).

Let's consider two simple examples that demonstrates how the VB algorithm is applied. ## Univariate Gaussian This example is taken from BISHOP 2006 pp. 470-476. Let's consider a Gaussian distribution with unknown mean and precision, denoted by μ and τ respectively. We infer the posterior distribution of these parameters given the data $y = y_1, y_2, \ldots y_N$. The likelihood function is then given by:

$$p(y|\mu,\tau) = \big(\frac{\tau}{2\pi}\big)^{N/2} e^{-\frac{\tau}{2}\sum_{i=1}^{N}(y_i-\mu)^2} \qquad [1]$$

Correspondingly, the prior distributions are given by:

$$p(\mu|\tau) = \frac{\lambda_0\tau}{2\pi} * e^{-\frac{\lambda_0\tau}{2}(\mu-\mu_0)^2} = N(\mu|\mu_0, (\lambda_0\tau)^{-1}) \qquad [2]$$

$$p(\tau) = \frac{1}{b_0^{a_0}\Gamma(a_0)} e^{-\frac{\tau}{b_0}} \tau^{a_0-1} = Gamma(\tau|a_0, b_0) \qquad [3]$$

Thus, the posterior distribution can be expressed as:

$$p(\mu,\tau|y) \propto p(y|\mu,\tau)p(\mu|\tau)p(\tau) \qquad [4]$$

We can approximate the true posterior using its variational approximation denoted by $q(\mu,\tau)$, which can be expressed in a factorized form as follows:

$$q(\mu,\tau) = q_\mu(\mu)q_\tau(\tau) \qquad [5]$$

It is to be noted that the true posterior does not approximate this way. Each of these two components is obtained by averaging over the parameter in the other component. We assume here that the parameters μ_0, τ_0, a_0, b_0 are known.

$$log(q_\mu(\mu)) = E_\tau[log(p(y|\mu,\tau))] + E_\tau[log(p(\mu|\tau))] \qquad [6]$$
$$= -\frac{E(\tau)}{2}\lambda_0(\mu-\mu_0)^2 + \sum_{i=1}^{N}(y_i-\mu)^2 + c$$

Completing the square and solving, we get a Gaussian distribution with mean and precision given by μ_N and λ_N respectively, where,

$$\mu_N = \frac{\lambda_0\mu_0 + N\bar{y}}{\lambda_0 + N}$$
$$\lambda_N = (\lambda_0 + N)E(\tau)$$

Likewise,

$$log(q_\tau(\tau)) = E_\mu[log(p(y|\mu,\tau))] + E_\mu[log(p(\mu|\tau))] + E_\mu[log(p(\tau))] \qquad [7]$$

$$= (a_0 - 1)log(\tau) - b_0\tau + \frac{log(\tau)}{2} + \frac{N}{2}log(\tau) - \frac{\tau}{2}E_\mu[\lambda_0(\mu - \mu_0)^2 + \sum_{i=1}^{N}(y_i - \mu)^2] + c$$

Upon solving, we see that $q_\tau(\tau)$ follows a Gamma distribution with parameters a_N and b_N respectively, where

$$a_N = a_0 + \frac{N+1}{2}$$

$$b_N = b_0 + \frac{E_\mu[\lambda_0(\mu - \mu_0)^2]}{2} + \sum_{i=1}^{N}(y_i - \mu)^2 \qquad [8]$$

$$= b_0 + \frac{E_\mu(\mu^2)(N + \lambda_0)}{2} - 2E_\mu(\mu)\big(\sum_{i=1}^{N} y_i + \lambda_0\mu_0\big) + \sum_{i=1}^{N} y_i^2 + \lambda_0\mu_0^2$$

The optimal distributions $q_\mu(\mu)$ and $q_\tau(\tau)$ arose from the structure of the likelihood function and our choice of conjugate priors. We solve for this in an iterative manner by assuming an initial value for $E_\tau(\tau)$ and using this to re-compute the distribution $q_\mu(\mu)$. This will help us re-estimate the moments $E(\mu)$ and $E(\mu^2)$, which can be used to re-compute the distribution $q_\tau(\tau)$. We proceed iteratively in this manner until convergence occurs.

```
rconsole
### R CODE

N = 1000
set.seed(100)
x = rnorm(N)

mu00 = rnorm(1)
b00 = a00 = runif(1)
lambda00  = runif(1)

mu0 = rnorm(1)
b0 = runif(1)
a0 = runif(1)
lambda0  = runif(1)

reps=1000

lambda_N =  b_N = E_tau = E_mu = E_mu2 = q_mu = q_tau = KL = matrix(,reps,1)
E_tau[1,] = a0/b0

mu_N = ((lambda0*mu0)+(N*mean(x)))/(lambda0+N)
a_N = a0+((N+1)/2)
```

```
lambda_N[1,] = (lambda0+N)*E_tau[1,]
q_mu[1,] = rnorm(1,mu_N,lambda_N[1,])
E_mu[1,] = mu_N
E_mu2[1,] = (1/lambda_N[1,])+mu_N^2
b_N[1,] = b0+0.5*((E_mu2[1,])*(N+lambda0)-(2*E_mu[1,]*((N*mean(x))+
(lambda0*mu0)))+((lambda0*(mu0^2))+sum(x^2)))

q_tau[1,] = rgamma(1,a_N,b_N[1,])


for(i in 2:reps){

  lambda_N[i,] = (lambda0+N)*E_tau[i-1,]
  q_mu[i,] = rnorm(1,mu_N,lambda_N[i,])
  E_mu[i,] = mu_N
  E_mu2[i,] = (1/lambda_N[i,])+mu_N^2
  b_N[i,] = b0+0.5*((E_mu2[i,])*(N+lambda0)-(2*E_mu[i,]*((N*mean(x))+
lambda0*mu0)))+((lambda0*(mu0^2))+sum(x^2)))
  q_tau[i,] = rgamma(1,a_N,b_N[i,])
  E_tau[i,] = a_N/b_N[i,]
  }
```