Bayesian hierarchical GLMs

Nengjun Yi

This vignette describes Bayesian generalized linear models (GLMs) and the EM-IWLS algorithm, and then the methods for assessing the fitted model and its predictive values. Finally, it explains how to set up and fit Bayesian GLMs using the `bglm` function in the **BhGLM** package.

Generalized Linear Models (GLMs)

Generalized linear models unify the approaches needed to analyze data for which either the assumption of a linear relation or normal variation is not appropriate**.** A generalized linear model consists of three components: *the linear predictor η, the link function h, and the data distribution p*. The linear predictor is a linear function of the predictors:

$$\eta_i = X_i\beta = \beta_0 + x_{i1}\beta_1 + \cdots + x_{iJ}\beta_J \quad (1)$$

The mean of the response variable is related to the linear predictor via a link function *h*:

$$\mu_i = E(y_i \mid \eta_i) = h^{-1}(\eta_i) \quad (2)$$

The data distribution (likelihood function) depends on the linear predictor $X\beta$ and generally also a dispersion (or variance) parameter ϕ, and can be expressed as

$$p(y \mid X\beta, \phi) = \prod_{i=1}^{n} p(y_i \mid X_i\beta, \phi) \quad (3)$$

The data distribution takes the family of exponential distribution, including Gaussian, binomial, Poisson and negative binomial (if the dispersion is fixed) as special cases.

The classical analysis of GLMs is to estimate the parameters by maximizing the likelihood function $p(y \mid X\beta, \phi)$. The commonly used algorithm is the iterative weighted least squares (IWLS), as implemented in standard R function `glm`, which iteratively approximates GLM by a weighted linear model and updates the parameters by fitting the weighted linear model. However, the classical GLMs cannot deal with numerous and correlated predictors.

Prior distributions

Bayesian GLMs include priors on the parameters, which can constraint the coefficients in reasonable ranges and thus allows the model to be reliably fitted and to identify important predictors. The function `bglm` employs uniform priors on the dispersion ϕ and intercept β_0, and four types of informative priors on the coefficients β_i. For specifying appropriate priors for all the coefficients, it is important to transform all variables to have a common scale. The function `covariates` in **BhGLM** can be used to transform both continuous and categorical predictors.

The first two types of priors are double-exponential and Student-*t*, respectively

$$\beta_j \sim DE(\mu_j, \phi s_j) \tag{4}$$

$$\beta_j \sim t_v(\mu_j, \phi s_j^2) \tag{5}$$

where the preset scale parameter s_j controls the shrinkage on the coefficient β_j; smaller scale s_j induces stronger shrinkage on β_j. The Student-*t* distribution includes normal (e.g. $v = +\infty$) and Cauchy (e.g. $v = 1$) distributions as special cases. Different scale values s_j can be specified for different predictors. This is very useful for incorporating prior information about the predictors into the analysis. For example, we can set a large scale for some relevant predictors and a small scale for others. If we have no prior information, we use a common scale *s* for all predictors. We recommend fit models with several scales, for example, $s = 0.03, 0.05, 0.1, 0.5, 1, 2$, and then choose an optimal model from them.

The second types of priors in the `bglm` function are the spike-and-slab mixture double-exponential and Student-*t* distributions:

$$\beta_j \sim DE\left(\mu_j, (1-\gamma_j)s_0 + \gamma_j s_1\right) \tag{6}$$

$$\beta_j \sim t_v\left(\mu_j, (1-\gamma_j)s_0 + \gamma_j s_1\right) \tag{7}$$

where s_0 and s_1 are chosen to be small and large, for modeling irrelevant and relevant coefficients, respectively, and γ_j is the indicator variable. For the spike-and-slab priors, the predictors are divided into groups. A model can include un-grouped predictors that don't belong to any group. The un-grouped predictors are those that are known to be important (e.g. relevant covariates) and thus are assumed to follow weakly informative priors: $\beta_j \sim DE(0, s_1)$ or $t_v(0, s_1)$. The grouped predictors are assumed to follow the above spike-and-slab priors. For predictors in group *g*, the indicator variables are assumed to follow the Berllouli distribution:

$$\gamma_j \mid \theta_g \sim \text{Bin}(\gamma_j \mid 1, \theta_g) = \theta_g^{\gamma_j}(1-\theta_g)^{1-\gamma_j} \tag{8}$$

For the group-specific probability θ_g, we use the uniform prior: $\theta_g \sim U(0, 1)$. We usually set s_1 to be 0.5 or 1, and consider several values for s_0, for example, $s_0 = 0.03, 0.05, 0.08, 0.1, 0.5$. We then choose an optimal model.

With appropriate scales, these four types of priors have long-tails and a peak at zero, thus performing less (strong) shrinkage for relevant (irrelevant) predictors and leading to robust inferences on large-scale data. The spike-and-slab priors not only induce different shrinkages on different coefficients, but also can incorporate group structures of predictors (for example, biological pathways) into the analysis.

EM-IWLS algorithm

The `bglm` function employs the EM-IWLS algorithm to fit the above Bayesian GLMs by finding the posterior modes of the parameters, i.e. estimating the parameters by maximizing the posterior density. The EM-IWLS algorithm incorporates an EM procedure into the usual IWLS algorithm. The EM procedure is based on the hierarchical formulations of the double-exponential and Student-t distributions, i.e. which can be expressed as normal distributions with unknown variances following exponential or inverse-gamma distribution. The algorithm treats the variances and the indicator variables (for the spike-and-slab priors) as missing data.

The algorithm is fast for fitting a GLM with hundreds of predictors, however, can be slow for larger models due to the need to calculate the inverse of a large matrix. The `bglm` function implements two methods to update the coefficients, i.e., jointly updating all coefficients, or updating a group of coefficients at a time and proceeding by cycling through all the groups at each iteration. For models with thousands of predictors, the group update method can be much faster.

Summarizing the fitted model and inferences

The EM-IWLS algorithm returns the estimates of the coefficients and their standard deviations. The *p*-values for testing the hypotheses H_0: $\beta_j = 0$ can be calculated similarly as the classical framework. The `summary.bh` function in **BhGLM** provides the estimates of the coefficients, the standard deviations, and the *p*-values. The `plot.bh` function graphically displays these estimates.

Evaluating the fitted model and the predictive performance

The function `measure.bh` returns several measures for assessing the quality of a fitted GLM, including: (1) Deviance: $-2\sum_{i=1}^{n}\log p(y_i \mid X_i\hat{\beta},\hat{\phi})$; (2) Mean squared error (MSE): MSE = $\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2$, where $\hat{y}_i = h^{-1}(X_i\hat{\beta})$; For logistic models, the function returns two additional measures: (3) area under the ROC curve (AUC), and (4) misclassification: $\frac{1}{n}\sum_{i=1}^{n} I(|\, y_i-\hat{y}_i \,|>0.5)$, where $I(|\, y_i-\hat{y}_i \,|>0.5)=1$ if $|\, y_i-\hat{y}_i \,|>0.5$, and $I(|\, y_i-\hat{y}_i \,|>0.5)=0$ if $|\, y_i-\hat{y}_i \,|\le 0.5$.

The package `BhGLM` provides two ways to evaluate the predictive performance of the model. If we have a training data set and a validation data set, we can use the training data to fit a Bayesian GLM and then evaluate the predictive performance on the validation data (treated as new data) using the function `measure.bh`. If we have only one data set, we can perform K-fold cross-validation using the function `cv.bh`, which returning all the measures described above.

Examples

We first use the functions `sim.x` and `sim.y` to simulate data, and then explain how to use `bglm` and other functions for analyzing Bayesian GLMs. We simulate 1000 individuals, 200 predictors, five non-zero coefficients, and several types of responses.

```
library(BhGLM)

set.seed(1234)
N = 1000
K = 200
# simulate K correlated variables
```

```
x = sim.x (n=N, m=K, corr=0.6)
# assign 5 non-zero coefficients, and all others are zero
h = rep(0.1,5)
nz = as.integer(seq(5, K, by=K/length(h)))
# simulate responses
yy = sim.y (x=x[, nz], mu=0, herit=h, p.neg=0.5, sigma=1.6, theta=2)

# show the non-zero coefficients
yy$coefs

#         x5         x45         x85        x125        x165
# 0.7247270 -0.7323255  0.7387685 -0.7297424  0.7287910
```

Analyzing binary response using Bayesian logistic regression with double-exponential prior

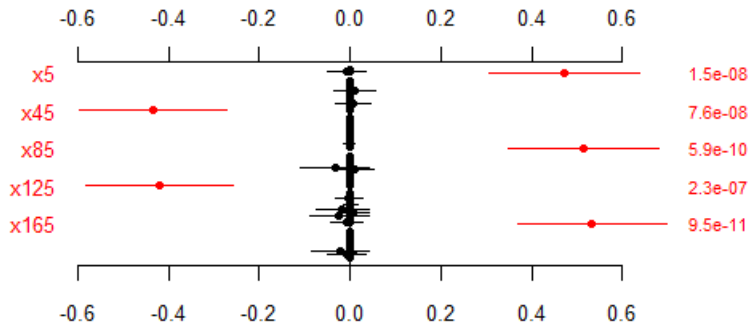
We first analyze the simulated binary response using Bayesian logistic regression with double-exponential prior, and we summarize the fitted model numerically and graphically:

```
y = yy$y.ordinal
f1 = bglm (y ~ ., data=x, family=binomial, prior="de", prior.scale=0.05)
# EM-IWLS iterations: 27
# Computational time: 0.034 minutes

res = summary.bh (f1, digits=4)
head (res)

#                 coef se(coef) exp(coef) lower.95 upper.95    pvalue
# (Intercept) -0.0324   0.0677    0.9681   0.8454   1.1086 6.324e-01
# x1           0.0000   0.0002    1.0000   0.9996   1.0004 9.996e-01
# x2           0.0000   0.0003    1.0000   0.9994   1.0006 9.976e-01
# x3           0.0000   0.0002    1.0000   0.9996   1.0004 9.992e-01
# x4          -0.0093   0.0214    0.9908   0.9493   1.0340 6.640e-01
# x5           0.4721   0.0835    1.6034   1.3569   1.8946 1.541e-08

plot.bh (f1, vars.rm=1, threshold=0.01, gap=10, col.pts=c("red", "black"))
```



In the above Bayesian logistic regression, the prior scale *s* is set to be 0.05 for all the 200 coefficients. If we treat the first three predictors as known relevant covariates with prior DE(0, 2.5) and other predictors with prior DE(0, 0.05), we can run:

```
f2 = bglm (y ~ ., data=x, family=binomial, prior="de", prior.scale=c(2.5, 2.5, 2.5, 0.05))
plot.bh (f2, vars.rm=1, threshold=0.01, gap=10)
```

In the above analyses, we jointly update all the 200 coefficients, which is fast. However, we can update a group of coefficients (say 20) at each time as follows. The group update algorithm can be much faster than the joint update if the model includes thousands of predictors.

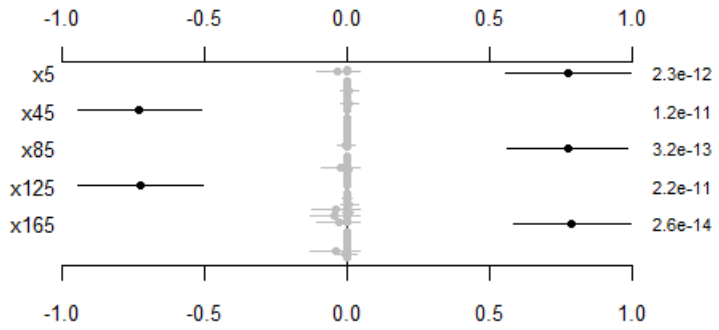
```
f3 = bglm (y ~ ., data=x, family=binomial, prior="de", prior.scale=0.05, method.coef=20)
# EM-IWLS iterations: 25
# Computational time: 0.008 minutes
```

Bayesian logistic regression with spike-and-slab double-exponential prior

We can analyze the data using the spike-and-slab double-exponential prior with s0=0.05 and s1=0.5 by running:

```
f4 = bglm (y ~ ., data=x, family=binomial, prior="mde", ss=c(0.05, 0.5))

plot.bh (f4, vars.rm=1, threshold=0.01, gap=10)
```



With the spike-and-slab prior, we can incorporate any known group structure into the model. Although this simulated data have no meaningful group structure, we explain how to incorporate any group structure into the analysis. Assume that the first three predictors are un-grouped covariates, and other predictors form groups: x4-x30, x31-x60, x61-x100, x101-x150, x151-x200. We can incorporate this group structure by running:

```
f5 = bglm (y ~ ., data=x, family=binomial, prior="mde", ss=c(0.05, 0.5),
           group=c(3, 30, 60, 100, 150, 200) )
```

To check whether the group structure has been correctly set, we can look at:

```
f5$group.vars
f5$ungroup.vars
```

Assessing the fitted models and the predictive values

We can use the function `measure.bh` to calculate measures for assessing the fitted model:

```
measure.bh(f1)

# deviance               auc               mse misclassification
# 1154.393             0.779             0.196             0.295
```

This is equivalent to
measure.bh (f1, new.x=x, new.y=y)

If we have new data new.x and new.y, we can use: measure.bh(f1, new.x, new.y),
to evaluate the predictive performance of the model f1.

We also can use cross-validation to assess the predictive performance:

```
cv.bh(f1, nfolds=10, ncv=5)$measures

#      deviance   auc   mse misclassification
# mean 1181.480 0.761 0.202             0.307
# sd      2.272 0.002 0.000             0.004
```

The cross-validation can be used to choose optimal prior scale and to compare different models.

Analyzing count response using Bayesian negative binomial model

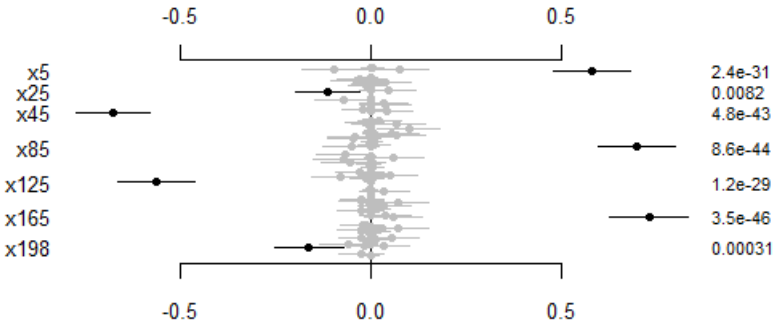
The negative binomial model is the standard method to analyze over-dispersed count data. To jointly fit numerous predictors, we use Bayesian negative binomial model:

```
y = yy$y.nb
f1 = bglm(y ~ ., data=x, family=NegBin, prior="de", prior.scale=0.05)
# EM-IWLS iterations: 18
# Computational time: 0.061 minutes

res = summary.bh(f1, digits=3)
head(res)

#                coef se(coef)   pvalue
# (Intercept)  0.031    0.041 4.52e-01
# x1           0.004    0.014 7.74e-01
# x2           0.000    0.005 9.34e-01
# x3          -0.096    0.042 2.35e-02
# x4           0.074    0.039 5.99e-02
# x5           0.579    0.050 2.43e-31

plot.bh(f1, vars.rm=1, threshold=0.01, gap=10)
```



measure.bh(f1)

```
# deviance      mse      mae
# 2914.848    6.279    1.342
```

cv.bh(f1, nfolds=10, ncv=5)$measures

```
#      deviance   mse   mae
# mean 3114.213 9.111 1.527
# sd     12.631 0.438 0.013
```