Bayesian Spike-and-Slab Lasso or Ridge GLMs and Cox Models

Nengjun Yi

This vignette describes Bayesian spike-and-slab lasso and ridge GLMs and Cox models, the EM coordinate descent algorithm for fitting these models, and then the methods for assessing the fitted model and its predictive values. Finally, it explains how to set up and fit Bayesian spike-and-slab mixture models using the `bmlasso` function in the **`BhGLM`** package.

Bayesian spike-and-slab GLMs and Cox models

Bayesian spike-and-slab lasso or ridge models defined in the `bmlasso` function are similar to Bayesian GLMs and Cox survival models with the spike-and-slab double-exponential and Student-*t* priors in the functions `bglm` and `bcoxph`. However, the algorithm in `bmlasso` is different from those in `bglm` and `bcoxph`. For GLMs, the current version of the `bmlasso` function only deals with Gaussian, binomial and Poisson models.

The spike-and-slab mixture double-exponential distribution is expressed as:

$$\beta_j \sim DE\left(\mu_j,(1-\gamma_j)s_0+\gamma_j s_1\right)=\frac{1}{(1-\gamma_j)s_0+\gamma_j s_1}\exp\left(-\frac{\left|\beta_j\right|}{(1-\gamma_j)s_0+\gamma_j s_1}\right) \quad (1)$$

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. The spike-and-slab mixture Student-*t* distribution is:

$$\beta_j \sim t_v\left(\mu_j,(1-\gamma_j)s_0+\gamma_j s_1\right) \quad (2)$$

We use the hierarchical expression for the spike-and-slab mixture Student-*t* distribution:

$$\beta_j \sim N(\mu_j,\tau_j^2),\ \tau_j^2 \sim \text{Inv-}\chi^2\left(\nu,(1-\gamma_j)s_0+\gamma_j s_1\right) \quad (3)$$

If the degree of freedom v is set to infinite, the Student-*t* distribution becomes the normal distribution, and thus the spike-and-slab mixture Student-*t* distribution becomes the spike-and-slab mixture distribution: $\beta_j \sim N\left(\mu_j,(1-\gamma_j)s_0+\gamma_j s_1\right)$.

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:

$$\gamma_j \mid \theta_g \sim \text{Bin}(\gamma_j \mid 1, \theta_g) = \theta_g^{\gamma_j}(1-\theta_g)^{1-\gamma_j} \quad (4)$$

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.

The EM coordinate descent algorithm

The `bmlasso` function implements a fast deterministic algorithm, called the EM coordinate descent algorithm to fit the spike-and-slab models by estimating the posterior modes of the parameters. The EM coordinate descent algorithm updates the coefficients β by maximizing the penalized log-likelihood for GLMs or log partial likelihood for Cox models:

$$l(\beta) - \sum_{j=1}^{J} \hat{S}_j^{-1} \left|\beta_j\right| \text{ for the spike-and-slab double-exponential prior} \quad (5)$$

$$l(\beta) - \sum_{j=1}^{J} \hat{\tau}_j^{-1} \beta_j^2 \text{ for the spike-and-slab double-exponential prior} \quad (6)$$

where $\hat{S}_j^{-1}$ is the conditional posterior expectation of $S_j^{-1} = [(1-\gamma_j)s_0 + \gamma_j s_1]^{-1}$, and $\hat{\tau}_j^{-1}$ is the conditional posterior expectation of τ_j^{-1}. At each iteration of the algorithm, these conditional posterior expectations can be calculated.

Thus, the coefficients can be updated using the cyclic coordinate descent algorithm, which has been implemented in the R package **`glmnet`**. Other parameters, e.g. the dispersion in the likelihood and the probability parameters θ can be estimated by maximizing the posterior distribution. The cyclic coordinate descent algorithm in **`glmnet`** is extremely fast, and can estimate some coefficients exactly to zero. The `bmlasso` function incorporates the cyclic coordinate descent algorithm in **`glmnet`** and thus is very fast for analyzing large-scale models.

Summarizing the fitted model and inferences

The EM coordinate descent algorithm returns the estimates of the coefficients, which equals zero for some coefficients. The interval estimates and the p-values cannot be calculated. The `plot.bh` function can graphically displays the estimates of the coefficients.

Evaluating the fitted model and the predictive performance

The methods for evaluating the fitted model and the predictive performance are the same as those described in the vignettes "**Bayesian hierarchical GLMs**" and "**Bayesian hierarchical Cox survival models"**.

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, 500 predictors, 6 non-zero coefficients, and several types of responses.

```
library(BhGLM)

set.seed(1234)
N = 1000
K = 500
# simulate K correlated variables
x = sim.x (n=N, m=K, corr=0.6)
x = as.matrix(x)
# assign 6 non-zero coefficients, and all others are zero
h = rep(0.1, 6)
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         x88        x171        x255        x338        x421
#  0.8158238 -0.8004485  0.8187133 -0.8075376  0.7900369 -0.8061452
```

Analyzing binary response

We first analyze the simulated binary response using lasso. The function `glmNet` in **`BhGLM`** fits GLMs or Cox models via the cyclic coordinate descent algorithm using the functions `glmnet` and `cv.glmnet` in the package **`glmnet`**:

```
y = yy$y.ordinal

f = glmNet(x, y, family="binomial", ncv=1)
```

The function `glmNet` uses cross-validation to choose an optimal penalty value in lasso, which is used to calculate the optimal prior scale in double-exponential prior. The optimal prior scale can guide us to choose s0 in the spike-and-slab double-exponential prior:

```
s0 = f$prior.scale
s0
# 0.0665108
```

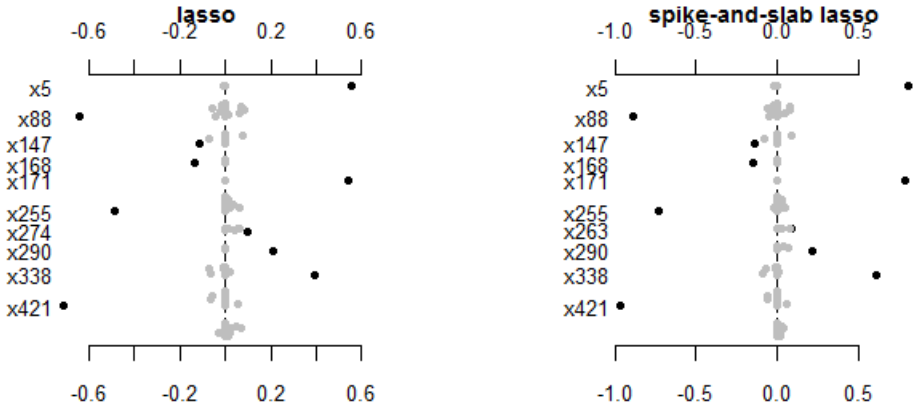
We then analyze the data using a logistic regression with spike-and-slab double-exponential prior with the function `bmlasso`:

```
f1 = bmlasso(x, y, family="binomial", prior="mde", ss=c(s0, 0.5))
# EM Coordinate Decent Iterations: 7
```

```
# Computational time: 0.007 minutes
```

We can display the estimated coefficients for the above two analyses using the function `plot.bh`:

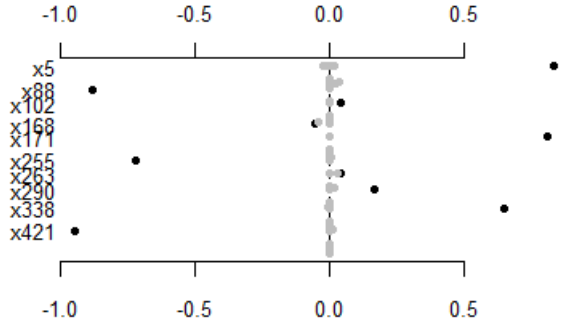
```
par(mfrow = c(1, 2), mar = c(3, 4, 4, 4))
plot.bh (coefs = f$coef, threshold = 10, gap = 100, main = "lasso")
plot.bh (coefs = f1$coef, threshold = 10, gap = 100, main = "spike-and-slab lasso")
```



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-x100, x101-x200, x201-x300, x301-x400, x401-x500. We can incorporate this group structure by running:

```
f2 = bmlasso (x, y, family="binomial", prior="mde", ss=c(0.05, 0.5),
                group=c(3, 100, 200, 300, 400, 500) )

plot.bh (coefs = f2$coef, threshold = 10, gap = 100)
```



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(f2)

```
# deviance               auc               mse misclassification
# 1087.775             0.797             0.184             0.297
```

This is equivalent to
measure.bh (f2, new.x=x, new.y=y)

If we have new data new.x and new.y, we can use measure.bh(f2, new.x, new.y) to evaluate the predictive performance of the model. We also can use cross-validation to assess the predictive pe rformance:

cv.bh(f2, nfolds=10, ncv=5)$measures

```
#      deviance   auc   mse misclassification
# mean 1132.100 0.774 0.193             0.310
# sd      3.981 0.002 0.001             0.005
```

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