<u>**APS/GDS March 2022 Short Course**</u>

https://wiki.socr.umich.edu/index.php/SOCR_News_APS_GDS_ShortCourse_March_2022

Topic 3 Demonstrations

Presenter: Raj Guhaniyogi
Title: *Bayesian High-dimensional Regressions with Tensors and Distributed Computation with Space-time Data*

This document details out the steps to implement various models discussed in the course. The document will point to open-source codes and packages written in R for the methods discussed.

<u>Implementation of Bayesian tensor regression (Guhaniyogi et. al., 2017, Journal of Machine Learning Research)</u>

Available resources:

1. R implementation of the method is available at https://github.com/rajguhaniyogi/APS-demo/tree/main under the file ``R code: Bayesian tensor regression."
2. The file is divided into two sections which are clearly demarcated. Section 1 includes all auxiliary functions which are used while implementing the main function. Section 2 includes the main function.

Step by step implementation

1. Copy Section 1 of the code and paste it on your R terminal.
2. Type and enter the following command to implement the model

```
tensor.reg(z.train, x.train, y.train, nsweep, rank, a.lam, b.lam, phi.alpha, scale = TRUE)
```

Arguments

(a) z.train: n*h matrix containing information of ordinary predictors.
(b) x.train: n*p*...*p dimensional tensor obtained by stacking p*...*p dimensional tensor predictors for n subjects.
(c) y.train: n dimensional scalar response vector.
(d) nsweep: number of MCMC iterations to run.
(e) rank: rank of the tensor coefficient upon PARAFAC decomposition.
(f) a.lam, b.lam, phi.alpha have default specification within the function.
(g) scale: whether each cell of the tensor is standardized. TRUE if standardized.

Values

The output includes all MCMC samples of all parameters. However, we will focus on the MCMC samples of predictor coefficients and error variance.

(a) c0.store: nsweep dimensional vector of intercept.
(b) gam.store: nsweep*h dimensional matrix, each row representing an MCMC sample.

(c) beta.store: nsweep*R*p*d dimensional tensor representing nsweep MCMC iterations of tensor margins.
(d) tau2.store: nsweep dimensional vector of error variance.
Attention: The function uses R package glmnet. Make sure that you use an appropriate version of glmnet.

Implementation of Bayesian tensor response regression (Guhaniyogi and Spencer, 2021, Bayesian Analysis)

Available resources:

1. R implementation of the method is available at https://github.com/danieladamspencer/BTRR/tree/master/R .
2. The site contains three files.
3. File "900_misc.R" contains all auxiliary functions which are used to implement the main function.
4. File "400_Y_x_BTRR.R" contains the function to implement our approach.
5. File "001_Y_x_BTRR_data.R" contains function to simulate data from the proposed model.

Step by step implementation

1. Copy all functions in "900_misc.R" to the R terminal.
2. Type and enter the following command to implement the model

```
BTR_Y_x(input, n.iter, n.burn, ranks,hyperparameters = NULL, save_after = NULL, save_llik = TRUE)
```

Arguments

(a) input: it is a list with input$x storing the T*n dimensional covariate matrix; input$Y is a p_1*...*p_D*T*n dimensional tensor.
(b) n.iter: number of MCMC iterations to run.
(c) n.burn: total number of burn-in MCMC samples.
(d) ranks: rank of the tensor coefficient upon PARAFAC decomposition.
(e) hyperparamters: default values of the hyper-parameters are set inside the function.
(f) save_after: if not NULL, the samples will be saved in a .RData file.
(g) save_llik: if TRUE, log likelihood per iteration is saved in the output.

Values

The output includes all MCMC samples of all parameters. However, we will focus on the post burn-in MCMC samples of the tensor predictor coefficients and error variance. All results are stored under the list "results."

(a) B: (n.iter-n.burn)*R*(p_1+...+p_D) dimensional vector of tensor margins.
(b) sig2y: n.iter dimensional vector having the post burn MCMC samples for the error variance.

Implementation of Bayesian symmetric tensor response regression (Guha and Guhaniyogi, 2021, Technometrics)

Available resources:

1. R implementation of the method is available at https://github.com/rajguhaniyogi/APS-demo/tree/main under the file ``R code: Bayesian symmetric tensor regression."
2. The file is divided into two sections which are clearly demarcated. Section 1 includes the code to simulate data from the model. Section 2 includes the main function to implement the model.

Step by step implementation

Type and enter the following command to implement the model

```
BSTRR_2d_cont(Y,X1,X2,X3,R,niter=2000)
```

Arguments

(a) Y: N*V*V tensor stacking over N symmetric tensors each of dimension V*V
(b) X1: N dimensional vector of key predictor
(c) X2: N dimensional vector of auxiliary predictor
(d) X3: N dimensional vector of auxiliary predictor
(e) R: fitted rank of the symmetric tensor
(f) niter: number of MCMC iterations

Values

The output includes all MCMC samples of all parameters. However, we will focus on the post burn-in MCMC samples of the tensor predictor coefficients and error variance. All results are stored under the list "results."

(a) W1: niter*[V*(V-1)/2] dimensional matrix with each row representing an MCMC sample of the upper triangular part of the coefficient matrix for X1
(b) beta0: niter dimensional vector containing MCMC samples of the intercept.
(c) beta2: niter dimensional vector containing MCMC samples for the coefficient of X2.
(d) beta3: niter dimensional vector containing MCMC samples for the coefficient of X3.
(e) sigma: niter dimensional vector containing MCMC samples for the error variance.
(f) indic.latvar: niter*V dimensional matrix with each row is a combination of 0's and 1's depending on whether a node is actively related to the key predictor.

illustration

```
rep <- 1

output<-
BSTRR_2d_cont(response[[rep]],c(predictor[[rep]][,1]),c(predictor[[rep]][,2]),c(predictor[[rep]][,3]),R=4,niter=
2000) ## run the function

post.burn <- 1001:2000  ##post burn-in iterations

colMeans(output$indic.latvar[post.burn,]) ## posterior prob. of nodes associated with the pred.
```

```
indic.coef1[[rep]] ## nodes associated with the key predictor in the truth

nodes.output <- list()

nodes.output$truth <- indic.coef1[[rep]]

nodes.output$post.prob <- colMeans(output$indic.latvar[post.burn,])

cbind(nodes.output$truth,nodes.output$post.prob)

colMeans(output$W1[post.burn,]) ## upper triangular part of the matrix coefficient

coef1[[rep]] ## upper triangular part of the true matrix coefficient

mean((colMeans(output$W1[post.burn,])-coef1[[rep]])^2) ## mean squared error of estimating  ##the true
coefficient
```

Implementation of distributed Bayesian inference in spatial data (Guhaniyogi et al., 2022, Statistical Science)

Available resources:

1. R implementation of the method is available at https://github.com/rajguhaniyogi/APS-demo/tree/main under the file ``R code: distributed Bayesian inference."
2. The file is divided into two sections which are clearly demarcated. Section 1 includes the code to simulate data from the model. Section 2 includes implements the model and stores the output.

Step by step implementation

1. Copy-paste Section 1 to simulate the data. The data is divided into training and test data.
2. Both training and test data contains spatial co-ordinates and response at these spatial co-ordinates.
3. In Section 2, set the number of processors you want to use to implement the distributed inference. The default is set at ``n.core=10."
4. Now copy-paste Section 2 of the code.

Values

The output includes point estimation and 95% credible interval for all parameters. It also includes point prediction and 95% predictive interval at all predicted values.

(a) low.quant.combined: 2.5% quantile for the posterior distributions of intercept, spatial variance, error variance and spatial range parameters.
(b) med.quant.combined: 50% quantile for the posterior distributions of intercept, spatial variance, error variance and spatial range parameters.
(c) upp.quant.combined: 97.5% quantile for the posterior distributions of intercept, spatial variance, error variance and spatial range parameters.
(d) Y.lower_qnt: 2.5% quantile for the posterior predictive distribution at all test locations.

(e) Y.med: 50% quantile for the posterior predictive distribution at all test locations.

(f) Y.upper_qnt: 97.5% quantile for the posterior predictive distribution at all test locations.

```
##Requires MBA package to interpolate observed and predicted spatial surface

 library(MBA)
par(mfrow=c(1,2))
pred.surf <- mba.surf(cbind(dat.miss[,1:2], Y.med), no.X=100, no.Y=100, extend=T)$xyz.est
image(pred.surf, xaxs = "r", yaxs = "r", main="Predicted Response")
contour(pred.surf, add=T) ## you may or may not add it

obs.surf <- mba.surf(dat.miss, no.X=100, no.Y=100, extend=T)$xyz.est
image(obs.surf, xaxs = "r", yaxs = "r", main="Observed response")
contour(obs.surf, add=T) ## you may or may not add it

MSPE <- mean((Y.med-c(dat.miss[,3]))^2) ## mean squared prediction error

Pred.coverage <-
length(intersect(which(Y.upper_qnt>dat.miss[,3]),which(Y.lower_qnt<dat.miss[,3])))/length(dat.miss[,3]) ##
predictive coverage

Avg.length <- mean(Y.upper_qnt-Y.lower_qnt) ## average length of 95% predictive interval
```