Tutorial_BayModDSGD

Qicheng Zhao

2025-02-08

The following provides a tutorial for the package. After importing the package, we will examine the toy dataset as shown below:

```
library(BayModDSGD)
data(toy)
head(toy)
```

```
##          x.V1       x.V2        x.V3 x.x_coord x.y_coord y
## 1 -0.56047565 -0.7152422  1.07401226 1.5526414  6.298418 1
## 2 -0.23017749 -0.7526890 -0.02734697 8.4585101  3.534138 0
## 3  1.55870831 -0.9385387 -0.03333034 2.1438043  4.247147 0
## 4  0.07050839 -1.0525133 -1.51606762 6.6987324  9.637688 0
## 5  0.12928774 -0.4371595  0.79038534 6.1775645  6.809985 1
## 6  1.71506499  0.3311792 -0.21073418 0.4999978  7.184639 1
```

The dataset will be partitioned into two distinct components: a matrix X comprising covariates (consisting of log-transformed gene expression counts and spatial coordinates for each location) and a response vector Y containing the disease status indicators.

```
x <- toy[, c("x.V1", "x.V2", "x.V3", "x.x_coord", "x.y_coord")]
y <- toy$y
```

To demonstrate the workflow of our method, we present the step-by-step procedure using this dataset. Initially, we perform missing data imputation under the assumption of an ignorable missing data mechanism.

```
missing_indices <- sample(length(y), 10)# select 10 values to let them become missing values
y[missing_indices] <- NA
coord<-as.matrix(toy[, c("x.x_coord", "x.y_coord")])  #To retrieve the spatial information for eac
h spot
imputed_mean<-missing_imputation(y,coord)[missing_indices]
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 0
##    Unobserved stochastic nodes: 10
##    Total graph size: 303
##
## Initializing model
```

```
y[missing_indices]<-rbinom(10,1,imputed_mean) # imputed the missing values by the corresponding es
timated mean
```

The function performs missing data imputation and returns a list of estimated means for each missing value. These estimated means are then used to impute the dataset.

Subsequently, the function is employed to conduct the analysis. The parameter represents a vector containing disease status information for each spatial location or cell, while includes both genetic and spatial data corresponding to each observation.

```
dsgd_single(list_y=y,matrix_x=x) # since the last two columns in x is the spatial information
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.26.22 (Stan version 2.26.1)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
## Chain 1: ------------------------------------------------------------
## Chain 1: EXPERIMENTAL ALGORITHM:
## Chain 1:   This procedure has not been thoroughly tested and may be unstable
## Chain 1:   or buggy. The interface is subject to change.
## Chain 1: ------------------------------------------------------------
## Chain 1:
## Chain 1:
## Chain 1:
## Chain 1: Gradient evaluation took 0.009298 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 92.98 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Begin eta adaptation.
## Chain 1: Iteration:   1 / 250 [  0%]  (Adaptation)
## Chain 1: Iteration:  50 / 250 [ 20%]  (Adaptation)
## Chain 1: Iteration: 100 / 250 [ 40%]  (Adaptation)
## Chain 1: Iteration: 150 / 250 [ 60%]  (Adaptation)
## Chain 1: Iteration: 200 / 250 [ 80%]  (Adaptation)
## Chain 1: Iteration: 250 / 250 [100%]  (Adaptation)
## Chain 1: Success! Found best value [eta = 0.1].
## Chain 1:
## Chain 1: Begin stochastic gradient ascent.
## Chain 1:   iter             ELBO   delta_ELBO_mean   delta_ELBO_med   notes
## Chain 1:    100      -187650.693             1.000            1.000
## Chain 1:    200      -187394.520             0.501            1.000
## Chain 1:    300      -187311.100             0.334            0.001
## Chain 1:    400      -187281.316             0.250            0.001
## Chain 1:    500      -187268.110             0.200            0.000
## Chain 1:    600      -187262.289             0.167            0.000
## Chain 1:    700      -187260.374             0.143            0.000
## Chain 1:    800      -187258.875             0.125            0.000
## Chain 1:    900      -187257.738             0.111            0.000
## Chain 1:   1000      -187257.661             0.100            0.000
## Chain 1:   1100      -187257.884             0.000            0.000
## Chain 1:   1200      -187258.105             0.000            0.000
## Chain 1:   1300      -187257.196             0.000            0.000   MEDIAN ELBO CONVERGED
## Chain 1:
## Chain 1: Drawing a sample of size 1000 from the approximate posterior...
## Chain 1: COMPLETED.
```

```
## Inference for Stan model: anon_model.
## 1 chains, each with iter=1000; warmup=0; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=1000.
##
##                mean se_mean   sd 2.5%  25%  50%   75% 97.5% n_eff khat
## beta[1]        0.93     NaN 0.24 0.46 0.77 0.93  1.10  1.39   NaN 0.47
## beta[2]        1.81     NaN 0.29 1.23 1.63 1.82  2.00  2.36   NaN 0.57
## beta[3]        3.27     NaN 0.40 2.51 3.00 3.27  3.55  4.04   NaN 0.49
## eta            0.03     NaN 0.03 0.00 0.01 0.02  0.04  0.09   NaN 0.54
## beta_gamma[1]  9.57     NaN 4.49 3.75 6.44 8.71 11.84 19.51   NaN 0.61
## beta_gamma[2]  9.68     NaN 4.35 3.95 6.71 8.81 11.63 19.87   NaN 0.59
## beta_gamma[3] 10.53     NaN 4.72 4.01 7.00 9.60 13.24 21.21   NaN 0.56
## w              0.77     NaN 0.19 0.29 0.67 0.84  0.92  0.98   NaN 0.52
## lp__           0.00     NaN 0.00 0.00 0.00 0.00  0.00  0.00   NaN 0.54
##
## Approximate samples were drawn using VB(fullrank) at Wed Feb 19 15:24:30 2025.
```

```
## We recommend genuine 'sampling' from the posterior distribution for final inferences!
```

Next, we present an example involving multiple samples from individuals, utilizing an alternative toy dataset referred to as 'toy2'.

```
data(toy2)
head(toy2)
```

```
##         x.V1        x.V2        x.V3 x.x_coord x.y_coord y label
## 1 -0.7400457  0.30878658  0.51980559  5.447189  1.724372 1     1
## 2  0.5030884  0.38848124  0.44233855  1.959015  2.130551 1     1
## 3 -0.8790444 -0.02834531 -0.31232314  9.302594  0.248819 0     1
## 4 -0.8813801 -0.59399920  0.09072223  1.638433  9.264955 1     1
## 5  0.1820235  0.48096339  1.01203654  3.286397  2.392358 1     1
## 6 -1.1870649 -0.05291483 -0.50489724  4.259975  2.055192 1     1
```

```
x <- toy2[, c("x.V1", "x.V2", "x.V3", "x.x_coord", "x.y_coord")]
y <- toy2$y
label<- toy2$label
```

The fundamental structure remains unchanged, with the only distinction being the addition of a new column, 'label,' which differentiates the data slices. The following function can then be executed to fit the model.

```
dsgd_multiple(y,x,label)
```

```
## Chain 1: ------------------------------------------------------------
## Chain 1: EXPERIMENTAL ALGORITHM:
## Chain 1:   This procedure has not been thoroughly tested and may be unstable
## Chain 1:   or buggy. The interface is subject to change.
## Chain 1: ------------------------------------------------------------
## Chain 1:
## Chain 1:
## Chain 1:
## Chain 1: Gradient evaluation took 0.004392 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 43.92 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Begin eta adaptation.
## Chain 1: Iteration:   1 / 250 [  0%]  (Adaptation)
## Chain 1: Iteration:  50 / 250 [ 20%]  (Adaptation)
## Chain 1: Iteration: 100 / 250 [ 40%]  (Adaptation)
## Chain 1: Iteration: 150 / 250 [ 60%]  (Adaptation)
## Chain 1: Iteration: 200 / 250 [ 80%]  (Adaptation)
## Chain 1: Iteration: 250 / 250 [100%]  (Adaptation)
## Chain 1: Success! Found best value [eta = 0.1].
## Chain 1:
## Chain 1: Begin stochastic gradient ascent.
## Chain 1:   iter             ELBO   delta_ELBO_mean   delta_ELBO_med   notes
## Chain 1:    100       -83579.648             1.000            1.000
## Chain 1:    200       -83448.534             0.501            1.000
## Chain 1:    300       -83404.031             0.334            0.002
## Chain 1:    400       -83386.394             0.251            0.002
## Chain 1:    500       -83377.813             0.200            0.001
## Chain 1:    600       -83374.287             0.167            0.001
## Chain 1:    700       -83372.288             0.143            0.000
## Chain 1:    800       -83371.136             0.125            0.000
## Chain 1:    900       -83370.623             0.111            0.000
## Chain 1:   1000       -83370.001             0.100            0.000
## Chain 1:   1100       -83369.309             0.000            0.000
## Chain 1:   1200       -83369.612             0.000            0.000
## Chain 1:   1300       -83369.861             0.000            0.000
## Chain 1:   1400       -83369.382             0.000            0.000   MEDIAN ELBO CONVERGED
## Chain 1:
## Chain 1: Drawing a sample of size 1000 from the approximate posterior...
## Chain 1: COMPLETED.
```

```
## Inference for Stan model: anon_model.
## 1 chains, each with iter=1000; warmup=0; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=1000.
##
##                mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff khat
## beta[1]        0.19     NaN 0.18 -0.17  0.06  0.19  0.32  0.52   NaN 0.69
## beta[2]        0.23     NaN 0.15 -0.07  0.12  0.22  0.33  0.53   NaN 0.65
## beta[3]       -0.20     NaN 0.18 -0.54 -0.31 -0.19 -0.08  0.15   NaN 0.65
## eta            0.17     NaN 0.09  0.06  0.11  0.15  0.21  0.40   NaN 0.66
## beta_gamma[1]  9.88     NaN 4.67  3.65  6.73  8.76 12.07 21.27   NaN 0.62
## beta_gamma[2]  9.93     NaN 4.74  3.65  6.80  8.84 12.23 21.60   NaN 0.70
## beta_gamma[3]  9.88     NaN 4.47  3.95  6.62  8.98 12.09 21.09   NaN 0.66
## w              0.78     NaN 0.18  0.31  0.68  0.83  0.92  0.98   NaN 0.72
## U[1]          -0.08     NaN 0.25 -0.57 -0.25 -0.09  0.09  0.42   NaN 0.67
## U[2]          -0.23     NaN 0.23 -0.64 -0.40 -0.24 -0.08  0.24   NaN 0.64
## sigma_square   0.96     NaN 0.03  0.90  0.96  0.97  0.98  0.99   NaN 0.69
## rho            0.54     NaN 0.28  0.04  0.30  0.56  0.78  0.97   NaN 0.54
## lp__           0.00     NaN 0.00  0.00  0.00  0.00  0.00  0.00   NaN 0.69
##
## Approximate samples were drawn using VB(fullrank) at Wed Feb 19 15:25:23 2025.
```

```
## We recommend genuine 'sampling' from the posterior distribution for final inferences!
```

Notice: This package is currently under development, and further refinements will be made in subsequent updates.