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
```

Subsequently, the dataset will be divided into two components: X for covaraites and Y for outcome.

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

Next, we will apply the function of dsgd_single to run the anslysis:

```
dsgd_single(y,x,nrow(x),ncol(x)-2) # 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
```

```
## 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 0.012147 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 121.47 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      -187648.407             1.000            1.000
## Chain 1:    200      -187390.397             0.501            1.000
## Chain 1:    300      -187305.355             0.334            0.001
## Chain 1:    400      -187274.348             0.250            0.001
## Chain 1:    500      -187260.340             0.200            0.000
## Chain 1:    600      -187254.045             0.167            0.000
## Chain 1:    700      -187251.827             0.143            0.000
## Chain 1:    800      -187250.218             0.125            0.000
## Chain 1:    900      -187249.011             0.111            0.000
## Chain 1:   1000      -187248.869             0.100            0.000
## Chain 1:   1100      -187249.032             0.000            0.000
## Chain 1:   1200      -187249.197             0.000            0.000
## Chain 1:   1300      -187248.310             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]        1.15     NaN 0.26 0.64 0.97 1.15  1.33  1.65   NaN 0.39
## beta[2]        2.04     NaN 0.31 1.40 1.84 2.05  2.25  2.63   NaN 0.41
## beta[3]        3.61     NaN 0.44 2.76 3.31 3.60  3.90  4.42   NaN 0.37
## eta            0.03     NaN 0.03 0.00 0.01 0.02  0.04  0.10   NaN 0.44
## beta_gamma[1]  9.58     NaN 4.49 3.75 6.45 8.72 11.85 19.52   NaN 0.63
## beta_gamma[2]  9.70     NaN 4.35 3.96 6.72 8.83 11.65 19.90   NaN 0.58
## beta_gamma[3] 10.58     NaN 4.73 4.03 7.04 9.64 13.29 21.27   NaN 0.51
## w              0.77     NaN 0.19 0.29 0.67 0.84  0.92  0.98   NaN 0.40
## lp__           0.00     NaN 0.00 0.00 0.00 0.00  0.00  0.00   NaN 0.44
##
## Approximate samples were drawn using VB(fullrank) at Fri Feb  7 20:51:18 2025.
```

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