Assignment 8

Contents

Assignment 8

```
library(loo)
```

```
## This is loo version 2.5.1
## - Online documentation and vignettes at mc-stan.org/loo
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
## Loading required package: ggplot2
## rstan (Version 2.21.5, GitRev: 2e1f913d3ca3)
## 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)
```

```
library(aaltobda)
data("factory")
data = list(y = factory, N = nrow(factory), M = ncol(factory))
data
```

```
## $y
##   V1  V2  V3  V4  V5  V6
## 1 83 117 101 105  79  57
## 2 92 109  93 119  97  92
## 3 92 114  92 116 103 104
## 4 46 104  86 102  79  77
## 5 67  87  67 116  92 100
##
## $N
## [1] 5
##
## $M
## [1] 6
```

Exercise 1

We modify the 3 models from Assignment 7 as follows:

- Separate model:

```
data {
  int<lower=0> N; // number of data points
  int<lower=0> M; // number of machines
  vector[M] y[N];
}
parameters {
  vector[M] mu;
  vector<lower=0>[M] sigma;
}
model {
  for (i in 1:M) {
      mu[i] ~ normal(0, 10);
      sigma[i] ~ gamma(1, 1);
  }
  for (i in 1:M) {
    y[, i] ~ normal(mu[i], sigma[i]);
  }
}
generated quantities {
  real ypred = normal_rng(mu[6], sigma[6]);
  vector[M] log_lik[N];
  for (j in 1:N)
    for (i in 1:M)
      log_lik[j,i] = normal_lpdf(y[j,i] | mu[i], sigma[i]);
}
```

- Pooled model:

```
data {
  int<lower=0> N; // number of data points
  int<lower=0> M; // number of machines
  vector[M] y[N];
}
parameters {
  real mu;
  real sigma;
}
model {
  mu ~ normal(0, 10);
  sigma ~ gamma(1, 1);;
  for (i in 1:M) {
    y[, i] ~ normal(mu, sigma);
  }
}

generated quantities {
  real ypred = normal_rng(mu, sigma);
  vector[N] log_lik;
  for (i in 1:N)
    log_lik[i] = normal_lpdf(y[i] | mu, sigma);
}
```

- Hierarchical model:

```
data {
  int<lower=0> N; // number of data points
  int<lower=0> M; // number of machines
  vector[M] y[N];
}
parameters {
  real mu_0;
  vector[M] mu;
  real sigma_0;
  real sigma;
}
model {
  mu_0 ~ normal(0, 10);
  sigma_0 ~ gamma(1, 1);
  mu ~ normal(mu_0, sigma_0);
  sigma ~ lognormal(0, 0.5);

  for (i in 1:M) {
    y[, i] ~ normal(mu[i], sigma);
  }
}
generated quantities {
  real ypred = normal_rng(mu[6], sigma);
  vector[M] log_lik[N];
  for (j in 1:N)
    for (i in 1:M)
      log_lik[j,i] = normal_lpdf(y[j,i] | mu[i], sigma);
}
```

Separate model:

```
separate = stan(file = "~/notebooks/bda2022/separate_.stan", data = data, refresh = 0)
```

a) Fitting the model:

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## clang -flto=thin -I"/usr/share/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown ty
## namespace Eigen {
## ^
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expected
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not fou
```

```
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
```

```
print(separate)
```

```
## Inference for Stan model: separate_.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                 mean se_mean    sd    2.5%     25%     50%     75%   97.5%
## mu[1]          50.69    0.15  8.83   31.64   45.19   51.23   56.76   65.93
## mu[2]          49.00    0.21 12.15   26.11   40.90   48.94   56.84   73.48
## mu[3]          58.37    0.20 11.92   32.80   50.48   59.16   67.21   79.06
## mu[4]          47.38    0.20 12.25   24.07   39.08   47.42   55.55   71.89
## mu[5]          61.10    0.23 12.94   33.75   52.17   61.75   70.96   83.03
## mu[6]          51.51    0.17 10.24   30.95   44.37   51.80   58.82   70.66
## sigma[1]       15.94    0.06  3.36   10.31   13.51   15.59   17.94   23.33
## sigma[2]       24.60    0.08  4.47   15.77   21.65   24.56   27.49   33.59
## sigma[3]       16.08    0.08  4.44    8.40   12.86   15.75   18.90   25.69
## sigma[4]       26.51    0.07  4.49   17.97   23.52   26.44   29.42   35.78
## sigma[5]       15.58    0.09  4.86    7.29   11.99   15.33   18.80   25.70
## sigma[6]       18.52    0.07  3.86   11.69   15.77   18.36   21.02   26.64
## ypred          51.50    0.35 21.37    4.54   37.93   52.77   66.89   88.93
## log_lik[1,1]   -5.78    0.01  0.76   -7.35   -6.28   -5.75   -5.26   -4.40
## log_lik[1,2]   -8.03    0.01  0.87   -9.91   -8.58   -7.99   -7.43   -6.47
## log_lik[1,3]   -7.31    0.01  1.00   -9.39   -7.95   -7.25   -6.59   -5.52
## log_lik[1,4]   -6.59    0.01  0.68   -7.96   -7.02   -6.56   -6.14   -5.29
## log_lik[1,5]   -4.32    0.01  0.81   -5.98   -4.90   -4.30   -3.67   -3.02
## log_lik[1,6]   -3.99    0.01  0.31   -4.73   -4.17   -3.93   -3.77   -3.54
## log_lik[2,1]   -7.16    0.01  0.95   -9.18   -7.78   -7.10   -6.48   -5.52
## log_lik[2,2]   -7.15    0.01  0.77   -8.77   -7.63   -7.12   -6.63   -5.67
## log_lik[2,3]   -6.01    0.01  0.90   -7.83   -6.61   -5.99   -5.40   -4.31
## log_lik[2,4]   -7.92    0.01  0.81   -9.64   -8.44   -7.87   -7.36   -6.45
## log_lik[2,5]   -6.31    0.02  0.97   -8.26   -6.94   -6.29   -5.67   -4.43
## log_lik[2,6]   -6.27    0.01  0.80   -7.91   -6.77   -6.24   -5.72   -4.78
## log_lik[3,1]   -7.16    0.01  0.95   -9.18   -7.78   -7.10   -6.48   -5.52
## log_lik[3,2]   -7.69    0.01  0.82   -9.44   -8.20   -7.65   -7.12   -6.17
## log_lik[3,3]   -5.87    0.01  0.90   -7.67   -6.47   -5.86   -5.27   -4.15
## log_lik[3,4]   -7.61    0.01  0.78   -9.26   -8.11   -7.56   -7.07   -6.20
## log_lik[3,5]   -7.39    0.02  1.04   -9.56   -8.05   -7.33   -6.66   -5.57
## log_lik[3,6]   -8.00    0.01  1.05  -10.23   -8.63   -7.91   -7.28   -6.19
## log_lik[4,1]   -3.94    0.01  0.37   -4.89   -4.03   -3.86   -3.73   -3.55
## log_lik[4,2]   -6.65    0.01  0.72   -8.14   -7.11   -6.64   -6.18   -5.25
## log_lik[4,3]   -5.13    0.01  0.86   -6.86   -5.72   -5.15   -4.53   -3.43
## log_lik[4,4]   -6.34    0.01  0.66   -7.67   -6.77   -6.31   -5.91   -5.06
## log_lik[4,5]   -4.32    0.01  0.81   -5.98   -4.90   -4.30   -3.67   -3.02
## log_lik[4,6]   -4.80    0.01  0.63   -6.07   -5.23   -4.78   -4.35   -3.59
## log_lik[5,1]   -4.23    0.01  0.54   -5.40   -4.57   -4.19   -3.85   -3.34
## log_lik[5,2]   -5.32    0.01  0.60   -6.52   -5.70   -5.33   -4.95   -4.08
## log_lik[5,3]   -3.98    0.01  0.50   -5.18   -4.28   -3.89   -3.59   -3.31
## log_lik[5,4]   -7.61    0.01  0.78   -9.26   -8.11   -7.56   -7.07   -6.20
## log_lik[5,5]   -5.57    0.02  0.97   -7.48   -6.21   -5.59   -4.92   -3.66
## log_lik[5,6]   -7.37    0.01  0.94   -9.31   -7.94   -7.30   -6.71   -5.71
```

```
## lp__         -346.97    0.06  2.42 -352.54 -348.40 -346.63 -345.19 -343.38
##              n_eff Rhat
## mu[1]         3619    1
## mu[2]         3287    1
## mu[3]         3693    1
## mu[4]         3617    1
## mu[5]         3222    1
## mu[6]         3486    1
## sigma[1]      3490    1
## sigma[2]      3362    1
## sigma[3]      3415    1
## sigma[4]      3805    1
## sigma[5]      3226    1
## sigma[6]      3428    1
## ypred         3654    1
## log_lik[1,1]  4677    1
## log_lik[1,2]  3935    1
## log_lik[1,3]  4778    1
## log_lik[1,4]  4090    1
## log_lik[1,5]  3018    1
## log_lik[1,6]  3229    1
## log_lik[2,1]  5152    1
## log_lik[2,2]  3598    1
## log_lik[2,3]  4256    1
## log_lik[2,4]  4831    1
## log_lik[2,5]  3750    1
## log_lik[2,6]  4701    1
## log_lik[3,1]  5152    1
## log_lik[3,2]  3814    1
## log_lik[3,3]  4188    1
## log_lik[3,4]  4712    1
## log_lik[3,5]  4384    1
## log_lik[3,6]  5621    1
## log_lik[4,1]  2654    1
## log_lik[4,2]  3404    1
## log_lik[4,3]  3872    1
## log_lik[4,4]  3938    1
## log_lik[4,5]  3018    1
## log_lik[4,6]  3464    1
## log_lik[5,1]  3708    1
## log_lik[5,2]  3126    1
## log_lik[5,3]  2572    1
## log_lik[5,4]  4712    1
## log_lik[5,5]  3299    1
## log_lik[5,6]  5403    1
## lp__          1744    1
##
## Samples were drawn using NUTS(diag_e) at Sun Nov 13 18:13:21 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

We can compute the log_likelihood:

```r
log_lik_separate = extract_log_lik(separate)
# print(log_lik_separate)
```

```r
loo_separate = loo(separate)
loo_separate
```

b) Compute the PSIS-LOO elpd values elpd and $\hat{k}$-value of the separate model:

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -196.5  8.0
## p_loo        19.8  1.5
## looic       393.1 16.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     25    83.3%   843
##  (0.5, 0.7]   (ok)        3    10.0%   722
##    (0.7, 1]   (bad)       2     6.7%   232
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```r
elpd_loo_separate = loo_separate$elpd_loo
cat("\nelpd value: ", elpd_loo_separate, "\n")
```

```
##
## elpd value:  -196.5462
```

```r
k_hat_values = loo_separate$diagnostics$pareto_k
cat("\nk_hat_values:\n", k_hat_values)
```

```
##
## k_hat_values:
##  0.2346305 0.4399603 0.4399603 0.7717813 0.3201842 0.2392812 0.2294621 0.227857 0.1603386 0.01050702
```

```r
lppd_loocv = elpd_loo_separate
n_r = nrow(log_lik_separate)
lppd = 0
for (i in (1:ncol(log_lik_separate))){
  col_sum = sum(exp(log_lik_separate[, i]))
  lppd = lppd + log((1/n_r)*col_sum)
}
separate_ploo_cv = lppd-lppd_loocv
cat("p_eff for separate model: ", separate_ploo_cv)
```

c) Compute the p_eff value:

```
## p_eff for separate model:  19.76885
```

d) Most of the $\hat{k}$-values are within the range of (0, 0.7), but some k $>$ 0.7. Thus, the PSIS-LOO estimate may be biased (optimistic).

Pooled model:

```
pooled = stan(file = "~/notebooks/bda2022/pooled_.stan", data = data, refresh = 0)
print(pooled)
```

a) Fitting the model: We can compute log-likelihood:

```
log_lik_pooled = extract_log_lik(pooled, parameter_name = "log_lik")
# print(log_lik_pooled)
```

```
loo_pooled = loo(pooled)
loo_pooled
```

b) Compute the PSIS-LOO elpd values and $\hat{k}$-value of the pooled model:

```
##
## Computed from 4000 by 5 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -134.9 1.6
## p_loo         3.1 1.3
## looic       269.9 3.1
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
elpd_loo_pooled = loo_pooled$elpd_loo
cat("\nelpd value: ", elpd_loo_pooled, "\n")
```

```
##
## elpd value:  -134.9493
```

```
k_hat_values = loo_pooled$diagnostics$pareto_k
cat("\nk_hat_values:\n", k_hat_values)
```

```
##
## k_hat_values:
##  0.2227056 0.3512961 0.4128205 0.4679113 0.2404289
```

```
lppd_loocv = elpd_loo_pooled
n_r = nrow(log_lik_pooled)
lppd = 0
for (i in (1:ncol(log_lik_pooled))){
  col_sum = sum(exp(log_lik_pooled[, i]))
  lppd = lppd + log((1/n_r)*col_sum)
}
pooled_ploo_cv = lppd-lppd_loocv
cat("p_eff for pooled model: ", pooled_ploo_cv)
```

c) Compute p_eff value $\hat{k}$-value of the pooled model:

```
## p_eff for pooled model:  3.115432
```

d) All of the $\hat{k}$-values are < 0.7, therefore, PSIS-LOO estimates are very reliable.

Hierarchical model:

```
hierarchical = stan(file = "~/notebooks/bda2022/hierarchical_.stan", data = data, refresh = 0)
```

a) Fitting the model:

```
## Warning: There were 87 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
print(hierarchical)
```

We can compute the log_likelihood:

```
log_lik_hierarchical = extract_log_lik(hierarchical, parameter_name = "log_lik")
# print(log_lik_hierarchical)
```

```
loo_hierarchical = loo(hierarchical)
loo_hierarchical
```

b) Compute loo elpd and $\hat{k}$-value of the hierarchical model:

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -129.3 4.4
## p_loo         8.2 1.6
## looic       258.6 8.8
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     27    90.0%   168
##  (0.5, 0.7]   (ok)        3    10.0%   206
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```r
elpd_loo_hierarchical = loo_hierarchical$elpd_loo
cat("\nelpd value: ", elpd_loo_hierarchical, "\n")
```

```
##
## elpd value:  -129.3203
```

```r
k_hat_values = loo_hierarchical$diagnostics$pareto_k
cat("\nk_hat_values:\n", k_hat_values)
```

```
##
## k_hat_values:
##  0.3571223 0.42984 0.42984 0.6218028 0.1676221 0.2575033 0.0607283 0.1497198 0.01272849 0.4049724 0.3
```

```r
lppd_loocv = elpd_loo_hierarchical
S = nrow(log_lik_hierarchical)
lppd = 0
for (i in (1:ncol(log_lik_hierarchical))){
  col = sum(exp(log_lik_hierarchical[, i]))
  lppd = lppd + log((1/S)*col)
}
hierarchical_ploo_cv = lppd-lppd_loocv
cat("p_eff for hierarchical model: ", hierarchical_ploo_cv)
```

c) Compute p_eff value:

```
## p_eff for hierarchical model:  8.208414
```

d) Almost all of the $\hat{k}$-values are within the range of (0, 0.7)with only 1 value at approximately 0.7. Thus, the PSIS-LOO estimates can be considered reliable.

e) Assessment of different models with regard to the elpd:

From the computed elpd values, it seems that Hierarchical model has the highest elpd_loo value, its corresponding k_hat value also suggest that the model is reliable. Therefore, it is reasonable to select this model.