BDA - Assignment 8

Anonymous

Contents

Load packages

```
library(aaltobda)
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.2, 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)
```

```
data("factory")
library(loo)
```

```
## This is loo version 2.3.1

## - 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

##
## Attaching package: 'loo'

## The following object is masked from 'package:rstan':
##
##     loo
```

```
SEED <- 48927 # set random seed for reproducability
```

Exercise 1

Separate

1

```
# Run stan for the separate model
separate_model <- stan_model("separate.stan")
separate_data <- list(y=factory,
                      N=nrow(factory),
                      J=ncol(factory),
                      p_mu=10,
                      p_alpha=1,
                      p_beta=1)

separate_sampling <- sampling(separate_model, data=separate_data)
```

The matrix is extracted as:

```
log_lik_separate <- extract_log_lik(separate_sampling)
```

2

The PSIS-LOO values elpd for the separate model are computed as:

```
loo_separate <- loo(log_lik_separate, r_eff=NULL, save_psis=FALSE, cores=getOption("mc.cores", 1), is_m
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
loo_separate
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -196.4  7.9
## p_loo        19.5  1.4
## looic       392.8 15.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     26    86.7%   880
##  (0.5, 0.7]   (ok)        3    10.0%   787
##    (0.7, 1]   (bad)       1     3.3%   461
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

3

Finally, the lppd is computed with the proposed equation:

$$lppd = \sum_{i=1}^{n} log(\frac{1}{S}\sum_{s=1}^{S} p(y_i|\theta^s))$$

Which is implemented in the following function that receives the matrix containing the log-likelihood values of each observation for every posterior draw:

```
compute_lppd <- function(matrix){
  sum_2 <- 0
  for (i in 1:ncol(matrix)){
    sum_1 <- 0
    for(j in 1:nrow(matrix)){
      sum_1 <- sum_1 + matrix[j,i]
    }
    sum_2 = sum_2 + sum_1/nrow(matrix)
  }
  return(sum_2)
}
```

Finally, the effective number of parameters can be computed as:

```
lppd_separate <- compute_lppd(log_lik_separate)
peff_separate <- lppd_separate - loo_separate$elpd_loo
```

```
## Warning: Accessing elpd_loo using '$' is deprecated and will be removed in
## a future release. Please extract the elpd_loo estimate from the 'estimates'
## component instead.
```

```
peff_separate
```

```
## [1] 10.5928
```

The effective number of parameters is, approximately, 10.5927992.

4

According to the Pareto k estimates, all of them are under 0.7, meaning that the estimates can be considered reliable. The value of k represents how fast the estimate is in terms of convergence. If k were greater than 0.7, there would be impractical convergence rates. In the separate model case, the majority of the values are under 0.5, so the distribution of raw importance ratios has finite variance and the central limit theorem holds.

5

The comparison is done at the end of the assignment.

Hierarchical

1

The requested matrix is:

```
log_lik_hierarchical <- extract_log_lik(hierarchical_sampling)
```

2

The PSIS-LOO values are:

```
loo_hierarchical <- loo(log_lik_hierarchical, r_eff=NULL, save_psis=FALSE, cores=getOption("mc.cores", 
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
loo_hierarchical
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -129.1 4.8
## p_loo         8.4 1.8
## looic       258.2 9.5
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     26    86.7%   1804
##  (0.5, 0.7]   (ok)        4    13.3%   189
##    (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.
```

3

As proceeded before, the effective number of samples is:

```
lppd_hierarchical <- compute_lppd(log_lik_hierarchical)
peff_hierarchical <- lppd_hierarchical - loo_hierarchical$elpd_loo
```

```
## Warning: Accessing elpd_loo using '$' is deprecated and will be removed in
## a future release. Please extract the elpd_loo estimate from the 'estimates'
## component instead.
```

```
peff_hierarchical
```

```
## [1] 5.120249
```

It yields an approximated result of 5.120249.

4

In this case, there are some estimates whose k value is over 0.7, meaning that they are not good, so technically, the previous model yields better results when measuring with the loo technique. However, the majority of the values remains in a good position, under 0.5.

5

The comparison is done at the end of the assignment.

Pooled

1

The requested matrix is:

```
log_lik_pooled <- extract_log_lik(pooled_sampling)
```

2

The PSIS-LOO values are:

```
loo_pooled <- loo(log_lik_pooled, r_eff=NULL, save_psis=FALSE, cores=getOption("mc.cores", 1), is_metho
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```r
loo_pooled
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -134.6 4.9
## p_loo         2.5 0.8
## looic       269.3 9.8
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

3

As in the previous cases, the effective number of samples is computed as:

```r
lppd_pooled <- compute_lppd(log_lik_pooled)
peff_pooled <- lppd_pooled - loo_pooled$elpd_loo
```

```
## Warning: Accessing elpd_loo using '$' is deprecated and will be removed in
## a future release. Please extract the elpd_loo estimate from the 'estimates'
## component instead.
```

```r
peff_pooled
```

```
## [1] 1.372442
```

It yields an approximated result of 1.3724424.

4

When using the proposed metric, the pooled model yields the best results comparing with the previous cases. All its k-values are under 0.5, meaning that the distribution of raw importance ratios has finite variance and the central limit theorem holds. Then, the PSIS-based estimates can be considered reliable.

5

According to the expected log predictive density, it makes sense that that one with a greater value yields a better performance when predicting new samples. It is the case of the hierarchical model. When performing the comparison, the following is obtained.

```r
loo_compare(loo_separate, loo_hierarchical, loo_pooled)
```

```
##        elpd_diff se_diff
## model2   0.0       0.0
## model3  -5.5       4.1
## model1 -67.3      10.0
```

Appendix

Stan codes for the three models:

Separate

```
data {
  int < lower =0 > N ;
  int < lower =0 > J ;
  vector [ J ] y [ N ];
  int p_mu;
  int < lower =0 > p_alpha;
  int < lower =0 > p_beta;
}
parameters {
  vector [ J ] mu ;
  vector < lower =0 >[ J ] sigma ;
}
model {
  // priors
  for ( j in 1: J ){
    mu [ j ] ~ normal (0 , p_mu);
    sigma [ j ] ~ gamma (p_alpha, p_beta);
  }
  // likelihood
  for ( j in 1: J )
    y [ , j ] ~ normal ( mu [ j ] , sigma [ j ]);
}
generated quantities {
  real ypred ;
  vector [J] log_lik [N];
  // Compute predictive distribution for the first machine
  ypred = normal_rng ( mu [6] , sigma [6]);
  for (j in 1:J){
    for (n in 1:N){
      log_lik[n,j] = normal_lpdf(y[n,j] | mu[j], sigma[j]);
    }
  }
}
```

Hierarchical

```
data {
  int < lower =0 > N ;
  int < lower =0 > J ;
  vector [ J ] y [ N ];
  int < lower =0 > p_mu;
  int < lower =0 > p_alpha;
  int < lower =0 > p_beta;
}
parameters {
  vector [ J ] mu ;
  real tau;
  real <lower=0> theta;
  real <lower=0> sigma ;
}
model {
  // Hyperpriors
  tau ~ normal(0, p_mu);
  theta ~ gamma(p_alpha, p_beta);

  // Priors
  for (j in 1: J ){
    mu[ j ] ~ normal (tau, theta);
  }
  sigma ~ gamma(1,1);
  // likelihood
  for ( j in 1: J )
    y [ , j ] ~ normal ( mu [ j ] , sigma);
}
generated quantities {
  real ypred ;
  vector [J] log_lik [N];
  // Compute predictive distribution for the sixth machine
  ypred = normal_rng ( mu [6] , sigma);
  for (j in 1:J){
    for (n in 1:N){
      log_lik[n,j] = normal_lpdf(y[n,j] | mu[j], sigma);
    }
  }
}
```

Pooled

```
data {
  int < lower =0 > N ;
  int < lower =0 > J ;
  vector [ J ] y [ N ];
  int < lower =0 > p_mu;
  int < lower =0 > p_alpha;
  int < lower =0 > p_beta;
}
parameters {
  real mu ;
  real sigma ;
}
model {
  // priors
  mu ~ normal (0 , p_mu);
  sigma  ~ gamma (p_alpha, p_beta);
  // likelihood
  for ( j in 1: J )
    y [ , j ] ~ normal ( mu  , sigma );
}
generated quantities {
  real ypred ;
  vector [J] log_lik [N];
  // Compute predictive distribution for the first machine
  ypred = normal_rng ( mu, sigma);
  for (j in 1:J){
    for (n in 1:N){
      log_lik[n,j] = normal_lpdf(y[n,j] | mu, sigma);
    }
  }
}
```