Assignment 8

Anonymous

2023-01-29

Contents

Exercise 1) Fitting Stan models

Read factory data

```
data("factory")
```

Separate model

The structure for separate model is:

$$\sigma_j \sim Inv - \chi^2(0.1)$$
$$\mu_j \sim N(0, 100)$$
$$y_{ij} \sim N(\mu_j, \sigma_j)$$

- Stan model code:

```
sm <- cmdstan_model(stan_file = "model8_sep.stan")
sm$print()
```

```
##   data {
##   int<lower=0> N;
##   int<lower=0> J;
##   vector[J] y[N];
##   }
##
##   parameters {
##   vector[J] mu;
##   vector<lower=0>[J] sigma;
##   }
##
##   model {
##     // priors
##   for (j in 1:J){
##     mu[j] ~ normal(0,100);
##     sigma[j] ~ inv_chi_square(0.1);
##   }
##
##   // likelihood
##   for (j in 1:J)
##     y[,j] ~ normal(mu[j], sigma[j]);
##   }
##
##   generated quantities {
##   real ypred[J];
##   ypred = normal_rng(mu, sigma); // predictive posterior for all machines
##
##   matrix[N,J] log_lik;
##   for (i in 1:J)
##     for (j in 1:N)
##       log_lik[j,i] = normal_lpdf(y[j,i] | mu[i], sigma[i]);
##   }
```

- Fit model

```
separate_factory <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory)
)
fit_sm <- sm$sample(data = separate_factory, seed= 42, refresh=0)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.1 seconds.
## Chain 2 finished in 0.1 seconds.
## Chain 3 finished in 0.2 seconds.
```

```
## Chain 4 finished in 0.1 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.1 seconds.
## Total execution time: 1.1 seconds.
```

Pooled model

The structure for pooled model is:

$$\mu \sim N(0, 100)$$
$$\sigma \sim Inv - \chi^2(0.1)$$
$$y \sim N(\mu, \sigma)$$

- Stan model code:

```
pm <- cmdstan_model(stan_file = "model8_pool.stan")
pm$print()
```

```
## data {
## int<lower=0> N;
## vector[N] y;
## real pmu;
## real<lower=0> psmu;
## real<lower=1> df;
## }
##
## parameters {
## real mu;
## real<lower=0> sigma;
## }
##
## model {
##   mu ~ normal(pmu, psmu);
##   sigma ~ inv_chi_square(1/df);
##   y ~ normal(mu, sigma);
## }
##
## generated quantities {
## real ypred;
## ypred = normal_rng(mu, sigma);
##
## vector[N] log_lik;
## for (i in 1:N)
##   log_lik[i] = normal_lpdf(y[i] | mu, sigma);
##
## }
```

- Fit model

```
pooled_factory <- list(
  y = unlist(factory, use.names=FALSE),
  N = nrow(factory) * ncol(factory),
  pmu = 0,
  psmu = 100,
  df = 10
)

fit_pm <- pm$sample(data = pooled_factory, seed= 42, refresh=0)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.6 seconds.
```

Hierarchical model:

The structure for hierarchical model is:

$$\begin{aligned} \theta &\sim N(0, 100) \\ \tau &\sim N(0, 100) \\ \sigma &\sim Inv - \chi^2(0.1) \\ \mu_j &\sim N(\theta, \tau) \\ y_{ij} &\sim N(\mu, \sigma) \end{aligned}$$

- Stan model code:

```
hm <- cmdstan_model(stan_file = "model8_hier.stan")
hm$print()
```

```
## // hierarchical prior for the mean
## data {
##   int<lower=0> N; // number of observations
##   int<lower=0> J; // number of groups
##   array[N] int<lower=1, upper=J> x; // discrete group indicators
##   vector[N] y; // real valued observations
## }
## parameters {
##   real theta; // prior  mean
##   real<lower=0> tau; // prior std constrained to be positive
##   vector[J] mu; // group means
##   real<lower=0> sigma; // common std constrained to be positive
## }
## model {
```

```
##   theta ~ normal(0, 100); // weakly informative prior
##   tau ~ normal(0,100); // weakly informative prior
##   mu ~ normal(theta, tau); // population prior with unknown parameters
##   sigma ~ inv_chi_square(0.1); // weakly informative prior
##
##   y ~ normal(mu[x], sigma);
## }
## generated quantities {
## real ypred6;
## ypred6 = normal_rng(mu[6], sigma);
## real mu7;
## mu7 = normal_rng(theta, tau);
##
## vector[N] log_lik;
## for (i in 1:N)
##   log_lik[i] = normal_lpdf(y[i] | mu[x[i]], sigma);
## }
```

- Fit model:

```
hierachial_factory <- list(
  N = ncol(factory)*nrow(factory),
  J = ncol(factory),
  x = rep(1:ncol(factory), nrow(factory)),
  y = c(t(factory[,1:6]))
)

fit_hm <- hm$sample(data = hierachial_factory, seed= 42, refresh=0)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 0.1 seconds.
## Chain 2 finished in 0.1 seconds.
## Chain 3 finished in 0.1 seconds.
## Chain 4 finished in 0.1 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.1 seconds.
## Total execution time: 0.9 seconds.

## Warning: 28 of 4000 (1.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.
```

Exercise 2) PSIS-LOO elpd values and $\hat{k}$-values

Separate model

- PSIS-LOO elpd:

```
loo_sm <- fit_sm$loo(variables = "log_lik", r_eff = TRUE, save_psis = TRUE, is_method="psis")
```

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

```
loo_sm$estimates
```

```
##            Estimate       SE
## elpd_loo -130.00643 3.904436
## p_loo      10.29567 1.702688
## looic     260.01285 7.808872
```

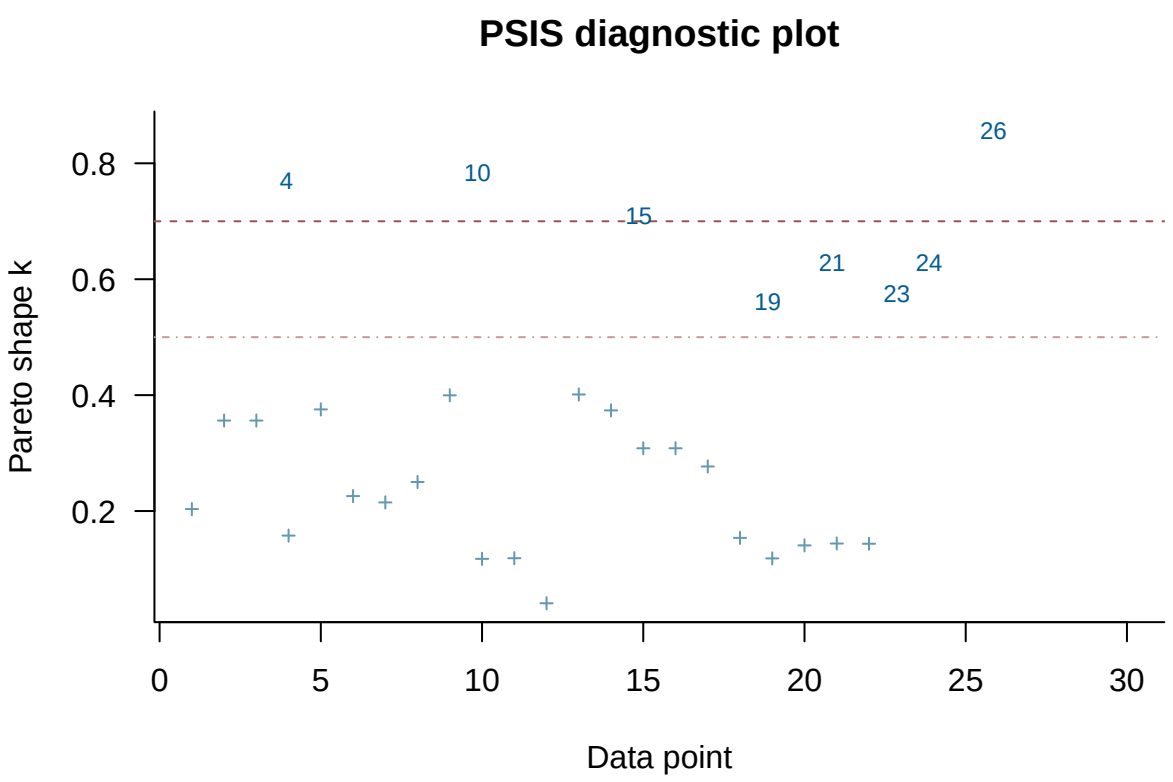
PSIS-LOO elpd of the separate model is -130, with the standard error 4.0.

- $\hat{k}$ values:

```
loo_sm$psis_object
```

```
## Computed from 4000 by 30 log-weights matrix
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     22    73.3%   1405
##  (0.5, 0.7]   (ok)        4    13.3%   880
##    (0.7, 1]   (bad)       4    13.3%   73
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_sm, label_points = TRUE)
```



Pooled model

- PSIS-LOO elpd:

```
loo_pm <- fit_pm$loo(variables = "log_lik", r_eff = TRUE, save_psis = TRUE, is_method="psis")
loo_pm$estimates
```

```
##              Estimate        SE
## elpd_loo -130.977307 4.3798573
## p_loo       2.093329 0.8496023
## looic     261.954614 8.7597146
```

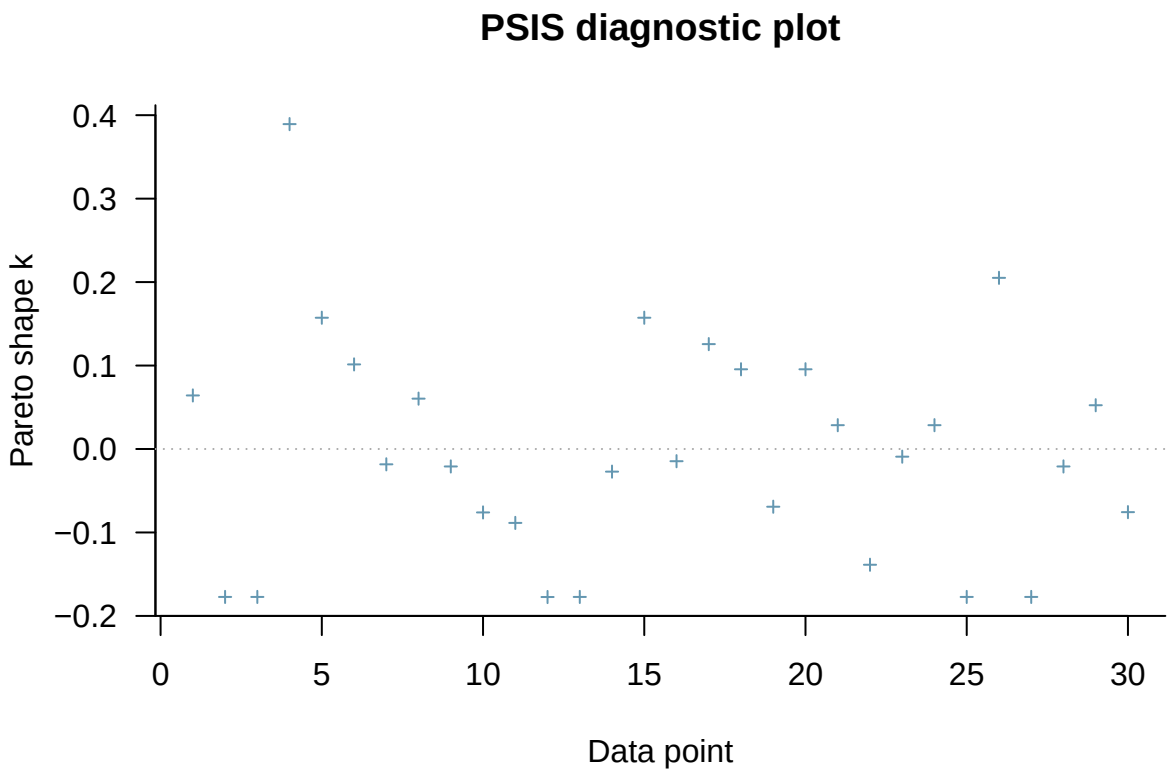
PSIS-LOO elpd of the pooled model is -131, with the standard error 4.4.

- $\hat{k}$ values:

```
loo_pm$psis_object
```

```
## Computed from 4000 by 30 log-weights matrix
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
plot(loo_pm, label_points = TRUE)
```



Hierarchical model

- PSIS-LOO elpd:

```
loo_hm <- fit_hm$loo(variables = "log_lik", r_eff = TRUE, save_psis = TRUE, is_method="psis")
```

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

```
loo_hm$estimates
```

```
##             Estimate       SE
## elpd_loo -126.674156 4.272669
## p_loo       5.739879 1.494193
## looic     253.348313 8.545337
```

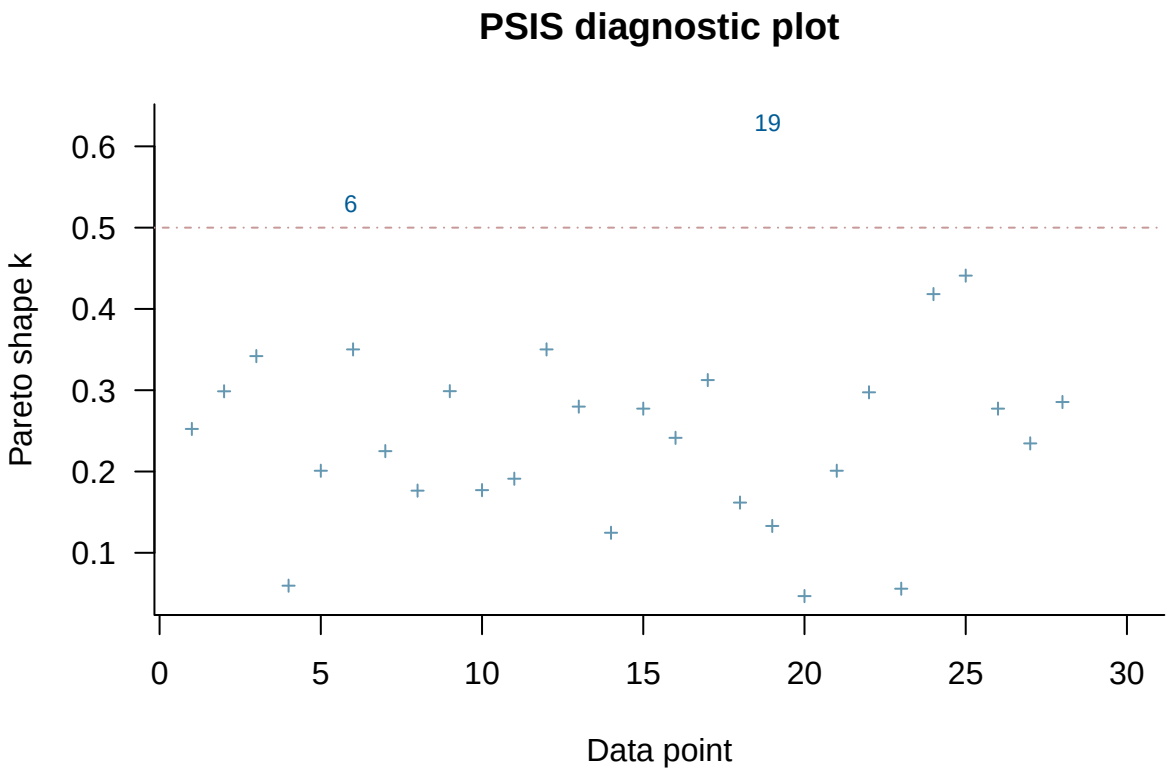
PSIS-LOO elpd of the pooled model is -127, with the standard error 4.3.

- $\hat{k}$ values:

```
loo_hm$psis_object
```

```
## Computed from 4000 by 30 log-weights matrix
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     28    93.3%   1260
##  (0.5, 0.7]   (ok)        2     6.7%   362
##    (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.
```

```
plot(loo_hm, label_points = TRUE)
```



Exercise 3) Effective number of parameters p_{eff}

There are 2 ways to get p_eff:

1. By equations (7.15) and (7.5) in BDA book:

$$p_{\text{loo-cv}} = lppd - lppd_{\text{loo-cv}}$$

where $lppd_{\text{loo-cv}}$ is the obtained ELPD values from Ex.2 and lppd is the log pointwise predictive density, defined as:

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

Code implementation of the above equations:

```
p_eff <- function(fit, N, S, elpd){
    res = 0
    for (i in 1:N)
      res = res + log(sum(exp(fit$draws("log_lik")[,,i]))) - log(S)
    return (res - elpd)
}
```

2. The effective number of paramters p_{eff} can also be calculated by `loo` package, which is the value `p_loo` when calling `loo$estimates`.

Separate model

```
# S = 4000: 4 chains, 1000 iterations
p_eff_sm = p_eff(fit = fit_sm,
                 N=ncol(factory) * nrow(factory),
                 S = 4000 ,
                 elpd = loo_sm$estimates[1,1])
print(paste0("p_loo computed by equation 7.15: ", p_eff_sm))
```

```
## [1] "p_loo computed by equation 7.15: 10.2956660523166"
```

```
print(paste0("p_loo computed by LOO library:"))
```

```
## [1] "p_loo computed by LOO library:"
```

```
print(loo_sm$estimates[2,])
```

```
##  Estimate        SE
## 10.295666  1.702688
```

For separate model, $p_{\text{eff}} = 10.3$ with standard error $= 1.7$.

Pooled model

```
p_eff_pm = p_eff(fit = fit_pm,
                 N=ncol(factory) * nrow(factory),
                 S = 4000 ,
                 elpd = loo_pm$estimates[1,1])
print(paste0("p_loo computed by equation 7.15: ", p_eff_pm))
```

```
## [1] "p_loo computed by equation 7.15: 2.09332940509819"
```

```
print(paste0("p_loo computed by LOO library:"))
```

```
## [1] "p_loo computed by LOO library:"
```

```
print(loo_pm$estimates[2,])
```

```
##  Estimate        SE
## 2.0933294 0.8496023
```

For pooled model, $p_{\text{eff}} = 2.1$ with standard error $= 0.8$.

Hierarchical model

```
p_eff_hm = p_eff(fit = fit_hm,
                 N=ncol(factory) * nrow(factory),
                 S = 4000 ,
                 elpd = loo_hm$estimates[1,1])
print(paste0("p_loo computed by equation 7.15: ", p_eff_hm))
```

```
## [1] "p_loo computed by equation 7.15: 5.73987875365587"
```

```
print(paste0("p_loo computed by LOO library:"))
```

```
## [1] "p_loo computed by LOO library:"
```

```
print(loo_hm$estimates[2,])
```

```
## Estimate       SE
## 5.739879 1.494193
```

For hierarchical model, $p_{\text{eff}} = 5.8$ with standard error $= 1.5$.

Exercise 4)

Patero $\hat{k}$ can be used to assess the reliability of the estimate:

- If k < 0.5: the variance of the raw importance ratios is finite, the central limit theorem holds, and the estimate converges quickly.
- $0.5 \leq$ k ≤ 1: the variance of the raw importance ratios is infinite but the mean exists, the generalized central limit theorem for stable distributions holds, and the convergence of the estimate is slower. The variance of the PSIS estimate is finite but may be large.
- k > 1, the variance and the mean of the raw ratios distribution do not exist. The variance of the PSIS estimate is finite but may be large.

However, in practice, good performance for values of $\hat{k}$ can range up to 0.7 instead of 0.5. Hence, 0.7 is usually used as threshold of $\hat{k}$.

In pooled model, all Patero $\hat{k}$ estimates are less than 0.5, indicating that the PSIS-LOO estimate converges quickly and thus very reliable. In separate model, there's 4 cases where $0.7 < \hat{k} \leq 1$ and in hierarchical model, there's 1 case. This means that the PSIS estimate for separate model converges slower than the other models and less reliable. On the other hand, the estimate for hierarchical model is decently reliable.

Exercise 5)

```
comp <- loo_compare(loo_sm, loo_pm, loo_hm)
comp
```

```
##        elpd_diff se_diff
## model3  0.0       0.0
## model1 -3.3       5.9
## model2 -4.3       6.2
```

From the above table, model 1 is separate model, model 2 is the pooled model and model 3 is the hierarchical model.

The difference in ELPD (elpd_diff) between hierarchical and separate model is smaller than 4 (3.2), this means that the 2 models have quite similar predictive performance. On the other hand, elpd_diff between hierarchical and pooled model is slightly larger than 4, meaning that hierarchical model has better predictive performance than pooled model.

In general, based on ELPD values, we have $elpd_{\text{hierarchical}} > elpd_{\text{separate}} > elpd_{\text{pool}}$, meaning that the hierarchical model should be selected according to PSIS-LOO.