BSDA: Assignment 8

Anonymous student

Contents

General Information to include

- **Time used for reading and self-study exercises**: ~ 12 hours.
- **Time used for the assignment**: ~10 hours
- **Good with assignment**: This assignment helped me understand the concepts of the leave-one-out cross-validation specially $\text{elpd}_{\text{loo-cv}}$ and p_{eff}, which after reading the book were not that clear.
- **Things to improve in the assignment**:

Model assessment: LOO-CV for factory data with Stan

Use leave-one-out cross-validation (LOO-CV) to assess the predictive performance of the pooled, separate and hierarchical Gaussian models for the factory dataset (see the second exercise in Assignment 7). To read in the data, just use:

```
library(bsda)
data("factory")
# to replicate results
set.seed(123)
```

1. Fit the models with Stan as instructed in Assignment~7. To use the `loo` or `psisloo` functions, you need to compute the log-likelihood values of each observation for every posterior draw (i.e. an S-by-N matrix, where S is the number of posterior draws and $N = 30$ is the total number of observations). This can be done in the {`generated quantities`} block in the Stan code.

Separate model:

$$
\begin{aligned}
y_{ij} &\sim N(\mu_j, \sigma_j) \\
\mu_j &\sim N(95, 20) \\
\sigma_j &\sim \text{Inv-}\chi^2(10)
\end{aligned}
$$

Here is the Stan code from `assignment7_separate.stan`

```
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(95, 20);
    sigma[j] ~ inv_chi_square(10);
  }

  // likelihood
  for (j in 1:J){
    y[,j] ~ normal(mu[j], sigma[j]);
  }
}

generated quantities {
  real ypred1;
  real ypred2;
  real ypred3;
  real ypred4;
  real ypred5;
  real ypred6;
  vector[J] log_lik[N];
  // Compute predictive distribution
  ypred1 = normal_rng(mu[1], sigma[1]);
  ypred2 = normal_rng(mu[2], sigma[2]);
  ypred3 = normal_rng(mu[3], sigma[3]);
  ypred4 = normal_rng(mu[4], sigma[4]);
  ypred5 = normal_rng(mu[5], sigma[5]);
  ypred6 = normal_rng(mu[6], sigma[6]);
  // assignment 8 loo
  for (j in 1:J) {
    for (n in 1:N) {
      log_lik[n, j] = normal_lpdf(y[n,j] | mu[j], sigma[j]);
    }
  }

}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_sep <- rstan::stan_model(file = "../assignment7/assignment7_separate.stan")

stan_data_separate <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory)
```

```
)

model_separate <- rstan::sampling(sm_sep, data = stan_data_separate, seed=123)

set.seed(123)
log_lik_sep <- loo::extract_log_lik(model_separate, merge_chains = FALSE)
r_eff_sep <- loo::relative_eff(exp(log_lik_sep), cores = 8)
loo_sep <- loo::loo(log_lik_sep, r_eff = r_eff_sep, cores = 8)
```

Pooled model:

$$
\begin{aligned}
y_i &\sim N(\mu, \sigma) \\
\mu &\sim N(95, 20) \\
\sigma &\sim \text{Inv-}\chi^2(10)
\end{aligned}
$$

Here is the Stan code from `assignment7_pooled.stan`

```
data {
  int<lower=0> N;
  vector[N] y;
}

parameters {
  real mu;
  real<lower=0> sigma;
}

model {
  // priors
  mu ~ normal(95, 20);
  sigma ~ inv_chi_square(10);

  // likelihood
  y ~ normal(mu, sigma);
}

generated quantities {
  real ypred;
  vector[N] log_lik;
  // Compute predictive distribution
  ypred = normal_rng(mu, sigma);

  for (i in 1:N){
    log_lik[i] = normal_lpdf(y[i] | mu, sigma);
  }

}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_pool <- rstan::stan_model(file = "../assignment7/assignment7_pooled.stan")

stan_data_pooled <- list(
```

```
  y = unname(unlist(factory)),
  N = nrow(factory) * ncol(factory)
)

model_pooled <- rstan::sampling(sm_pool, data = stan_data_pooled, seed= 123)

set.seed(123)
log_lik_pool <- loo::extract_log_lik(model_pooled, merge_chains = FALSE)
r_eff_pool <- loo::relative_eff(exp(log_lik_pool), cores = 8)
loo_pool <- loo::loo(log_lik_pool, r_eff = r_eff_pool, cores = 8)
```

Hierarchical model:

$$
\begin{aligned}
y_{ij} &\sim N(\mu_j, \sigma) \\
\mu_j &\sim N(\theta, \alpha) \\
\theta &\sim N(95, 20) \\
\alpha &\sim Half - t(4, 0, 20) \\
\sigma &\sim \text{Inv-}\chi^2(10)
\end{aligned}
$$

Here is the Stan code from `assignment7_h.stan`

```
data {
  int<lower=0> N;
  int<lower=0> J;
  vector[J] y[N];
}

parameters {
  vector[J] mu;
  real<lower=0> sigma;
  real theta;
  real<lower=0> alpha;
}

model {
  theta ~ normal(95, 20);
  alpha ~ student_t(4, 0, 20);
  // priors
  for (j in 1:J){
    mu[j] ~ normal(theta, alpha);
    sigma ~ inv_chi_square(10);
  }

  // likelihood
  for (j in 1:J){
    y[,j] ~ normal(mu[j], sigma);
  }
}

generated quantities {
  real ypred1;
  real ypred2;
  real ypred3;
```

```
  real ypred4;
  real ypred5;
  real ypred6;
  real ypred7;
  real mu_pred7;
  vector[J] log_lik[N];
  // Compute predictive distribution
  ypred1 = normal_rng(mu[1], sigma);
  ypred2 = normal_rng(mu[2], sigma);
  ypred3 = normal_rng(mu[3], sigma);
  ypred4 = normal_rng(mu[4], sigma);
  ypred5 = normal_rng(mu[5], sigma);
  ypred6 = normal_rng(mu[6], sigma);
  mu_pred7 = normal_rng(theta, alpha);
  ypred7 = normal_rng(mu_pred7, sigma);
  // assignment 8 loo
  for (j in 1:J) {
    for (n in 1:N) {
      log_lik[n, j] = normal_lpdf(y[n,j] | mu[j], sigma);
    }
  }
}
```

Here is the R implementation

```
options(mc.cores = parallel::detectCores())
sm_h <- rstan::stan_model(file = "../assignment7/assignment7_h.stan")

stan_data_h <- list(
  y = factory,
  N = nrow(factory),
  J = ncol(factory)
)

model_h <- rstan::sampling(sm_h, data = stan_data_h, seed= 123)

set.seed(123)
log_lik_h <- loo::extract_log_lik(model_h, merge_chains = FALSE)
r_eff_h <- loo::relative_eff(exp(log_lik_h), cores = 8)
loo_h <- loo::loo(log_lik_h, r_eff = r_eff_h, cores = 8)
```

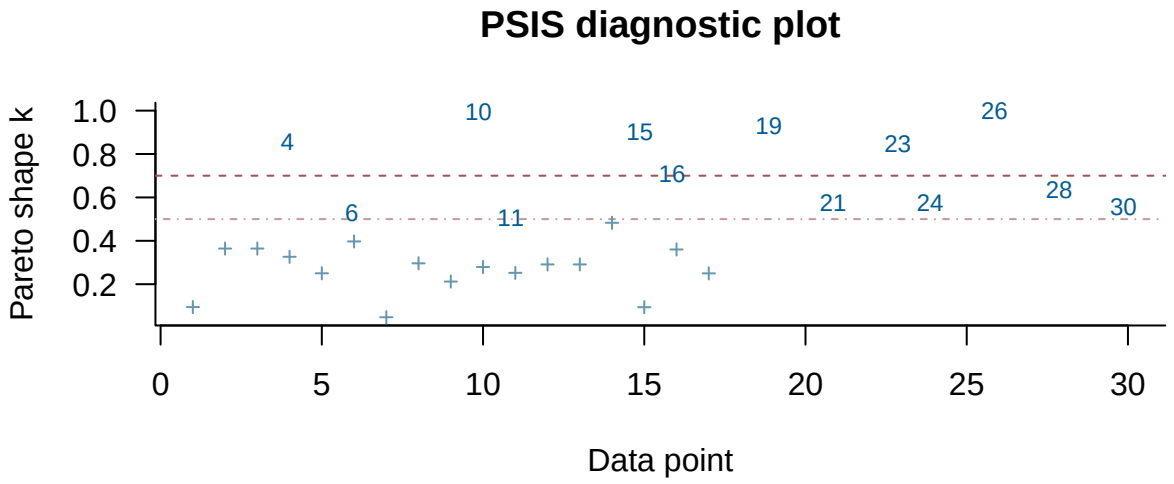
2. Compute the PSIS-LOO elpd values and the $\hat{k}$-values for each of the three models.

```
# Separate Model
loo_sep
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -137.7  8.9
## p_loo        20.1  4.9
## looic       275.4 17.7
```

```
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     17    56.7%   2132
##  (0.5, 0.7]   (ok)        6    20.0%   1146
##    (0.7, 1]   (bad)       7    23.3%   10
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

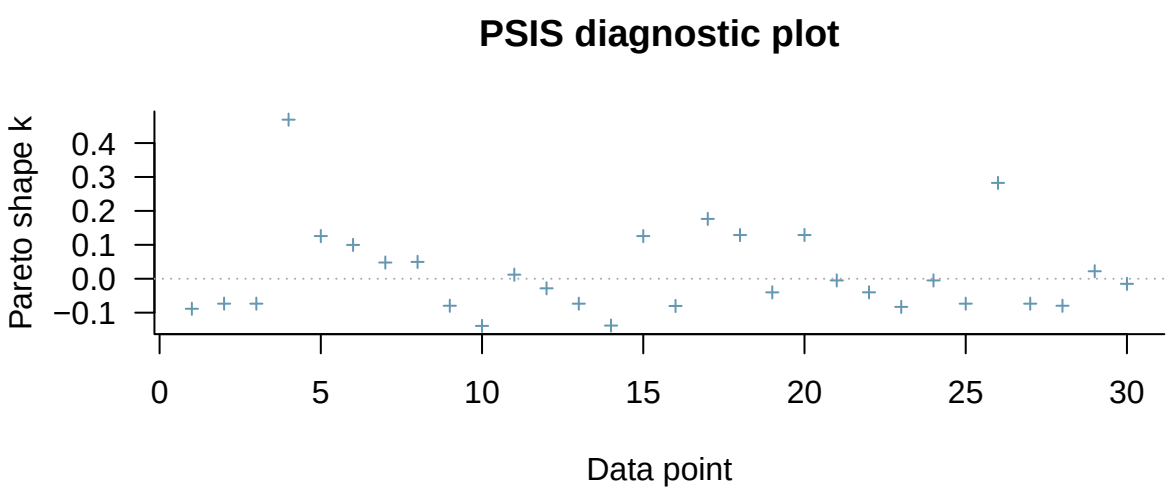
```
plot(loo_sep, label_points = TRUE)
```




```
# Pooled Model
loo_pool
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -131.1  5.2
## p_loo         2.3  1.0
## looic       262.1 10.3
## ------
## 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.
```

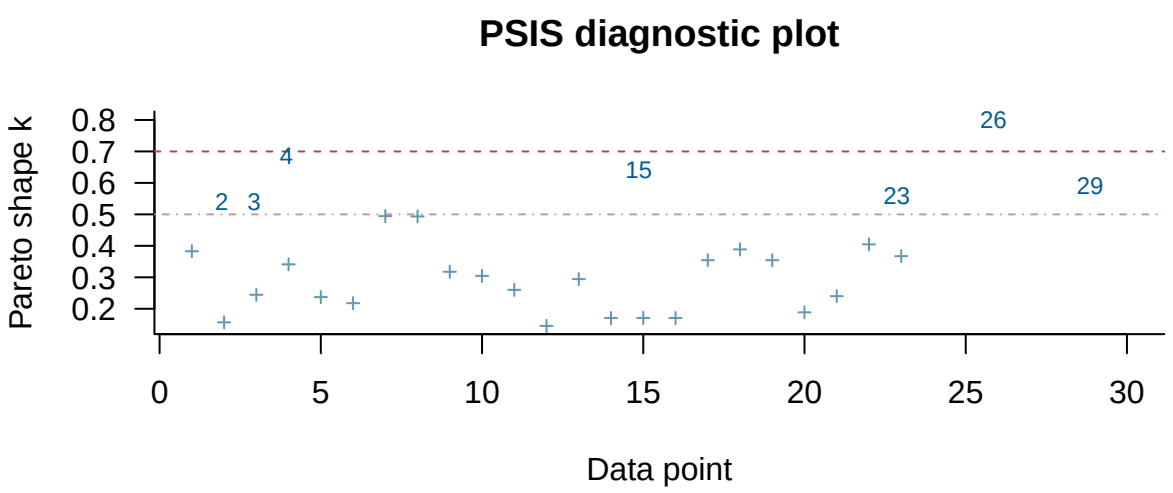
```
plot(loo_pool, label_points = TRUE)
```




```
# Hierarchical Model
loo_h
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -135.7 10.3
## p_loo        14.6  4.1
## looic       271.5 20.6
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     23    76.7%   722
##  (0.5, 0.7]   (ok)        6    20.0%   78
##    (0.7, 1]   (bad)       1     3.3%   54
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

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



3. Compute the effective number of parameters p_{eff} for each of the three models.

```
# Separate Model
loo_sep$estimates[2]
```

```
## [1] 20.08094
```

```
# Pooled Model
loo_pool$estimates[2]
```

```
## [1] 2.287403
```

```
# Hierarchical Model
loo_h$estimates[2]
```

```
## [1] 14.58554
```

4. Assess how reliable the PSIS-LOO estimates are for the three models based on the $\hat{k}$-values.

Separate Model:

- Pareto k diagnostic: 7 "bad" values with $\hat{k}$ greater than 0.7
- This suggests potential problems with model fit in some observations. This model may not be as reliable in certain parts of the data.

Pooled Model:

- Pareto k diagnostic: 0 "bad" values with $\hat{k}$ greater than 0.7
- This suggests that LOO estimates for this model are reliable.
- This model appears to be the most reliable.

Hierarchical Model:

- Pareto k diagnostic: 1 "bad" value with $\hat{k}$ greater than 0.7
- This model may not be as reliable in certain parts of the data.

5. An assessment of whether there are differences between the models with regard to the $\text{elpd}_{\text{loo-cv}}$, and if so, which model should be selected according to PSIS-LOO.

```
loo::loo_compare(loo_sep, loo_pool, loo_h)
```

```
##        elpd_diff se_diff
## model2  0.0       0.0
## model3 -4.7       6.8
## model1 -6.6       7.3
```

The reference model in `loo_compare` is automatically chosen and set to the model with the highest $\text{elpd}_{\text{loo-cv}}$ value. In this case, model2 which is the pooled model is the reference model, and the differences are calculated relative to it. Therefore, the pooled model is expected to have better predictive performance than the hierarchical and separate model.

6. Both the Stan and R code should be included in your report.