BDA - Assignment 9

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")
head(factory)
```

```
##   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
```

Exercise 1

Initially, the utility function was coded and tried with the proposed y_pred:

```
# Utiliy function
utility <- function(draws){
  sums <- 0
  subs <- 106*length(draws)
  for (i in draws){
    if (i>85){
      sums <- sums + 200
    }
  }
  return ((sums-subs)/length(draws))
}
```

```
# Utility computation of the provided y_pred test case
y_pred_mark <- c(123.80, 85.23, 70.16, 80.57, 84.91)
utility_mark <- utility(y_pred_mark)
utility_mark
```

```
## [1] -26
```

The stan model is executed and fitted with the following code lines:

```
# Run stan for the hierarchical model
hierarchical_model <- stan_model("hierarchical.stan")
hierarchical_data <- list(y=factory,
                          N=nrow(factory),
                          J=ncol(factory),
```

```
                      p_mu=10,
                      p_alpha=1,
                      p_beta=1)

hierarchical_sampling <- sampling(hierarchical_model, data=hierarchical_data)
```

```
extract_hierarchical <- rstan::extract(hierarchical_sampling, permuted=T)
```

Now, the utility is computed for the simulated samples as following:

```
utility_machines <- c(1:6)
utility_machines[1] <- utility(extract_hierarchical$ypred1)
utility_machines[2] <- utility(extract_hierarchical$ypred2)
utility_machines[3] <- utility(extract_hierarchical$ypred3)
utility_machines[4] <- utility(extract_hierarchical$ypred4)
utility_machines[5] <- utility(extract_hierarchical$ypred5)
utility_machines[6] <- utility(extract_hierarchical$ypred6)
```

The obtained utilities are:

- The utility for the first machine is -55.95
- The utility for the second machine is 63.65
- The utility for the third machine is -4.75
- The utility for the fourth machine is 75.2
- The utility for the fifth machine is 5.8
- The utility for the sixth machine is -12.1

Exercise 2

Given the previous values, the machines can be sorted from the worst one to the best one as: (1, 6, 3, 5, 2, 4).

The utility values give an idea of how profitable the machine is. The computation of the utility is a subtraction of the costs (including salaries, raw materials, maintenance, or initial investment; 106€ in total per product) from the revenues (200€ per sold product), hence, those machines with a positive utility value are profitable, since they produce more revenues than costs. However, many of them do not make any profit, since the costs are higher than the revenues. In the given case, only machines 2, 4 and 5 are profitable, since the utility function yields a positive value for them. The rest of the machines are not profitable.

Exercise 3

The expected utility of the products for a 7th machine is extracted as:

```
utility_machine7 <- utility(extract_hierarchical$ypred7)
```

The computed utility for the 7th machine is -57.15.

Exercise 4

Given the preivously computed utility, the owner should not buy a new machine, since its utility would be negative and, hence it would not be profitable.

Exercise 5

Stan code for the hierarchical model:

```
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 ypred1;
  real ypred2;
  real ypred3;
  real ypred4;
  real ypred5;
  real ypred6;
  real ypred7;
  real mu_7;

  vector [J] log_lik [N];
  // Compute predictive distribution for the sixth machine
  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_7 = normal_rng(tau, theta);
  ypred7 = normal_rng(mu_7, sigma);

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