Bayesian Data Analysis - Assignment 9

November 25, 2018

The hierarchical model is the best for this problem as it does treat every machine as a separate entity, but also computes the combination of all the machines as one entity. For that reason it can predict measurements for the machines which have no data. For example, there is no data about the seventh machine, but this model can predict its posterior distribution. The stan model for this is stated in the *Appendix A Source code*.

When sampling from the hierarchical stan model we can an output of all the μ's and *ypred*'s:

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
mu0	93.07	0.22	8.22	77.2	88.51	92.86	97.15	109.6	1448	1.0
sigma0	16.25	0.26	9.7	4.57	10.22	14.1	19.53	39.76	1366	1.0
mu[1]	79.75	0.16	7.04	65.89	75.12	79.78	84.44	93.44	1957	1.0
mu[2]	103.21	0.15	6.64	90.27	98.9	103.34	107.63	115.88	2057	1.0
mu[3]	88.98	0.1	6.18	76.64	84.97	89.11	93.05	101.44	3887	1.0
mu[4]	107.31	0.17	6.94	93.06	102.82	107.46	111.93	120.58	1606	1.0
mu[5]	90.6	0.09	6.06	78.66	86.62	90.71	94.69	102.29	4258	1.0
mu[6]	87.73	0.11	6.28	75.53	83.52	87.72	91.88	100.0	3170	1.0
sigma	15.3	0.05	2.42	11.42	13.55	15.03	16.72	20.75	2484	1.0
ypred[1]	79.51	0.28	16.96	44.57	68.43	79.38	90.73	112.97	3799	1.0
ypred[2]	103.4	0.28	16.77	68.28	92.68	103.7	114.81	135.66	3683	1.0
ypred[3]	89.54	0.27	16.92	56.33	78.63	89.69	100.47	122.68	3892	1.0
ypred[4]	107.04	0.29	16.8	73.94	96.1	107.13	118.31	139.57	3362	1.0
ypred[5]	90.76	0.27	16.85	57.11	80.09	90.75	101.87	123.32	3844	1.0
ypred[6]	87.93	0.27	16.94	54.05	76.76	88.02	98.9	121.21	3897	1.0
ypred[7]	93.28	0.42	25.17	43.03	78.04	93.14	108.5	142.62	3665	1.0
mu7	93.39	0.35	20.08	52.95	83.32	93.52	103.48	134.86	3359	1.0
lp__	-108.9	0.08	2.56	-115.2	-110.3	-108.5	-107.1	-105.1	1155	1.0

Next, we can write a small code that will **compute the expected utilities for the machines**:

```
utility = np.zeros(7)
ypred = fit_hierarchical.extract(permuted=True)['ypred']
for i in range(7):
    for j in range(0, len(ypred)):
        if (ypred[j, i] < 85):
            utility[i] -= 106
        else:
            utility[i] += (200-106)
    print('Machine {0}: {1}'.format(i+1, utility[i]/len(ypred)))
```

This gives us an output with the **expected utilities**:

```
Machine 1: -33.45
Machine 2: 68.15
Machine 3: 16.55
Machine 4: 75.6
Machine 5: 21.5
Machine 6: 6.65

Machine 7: 23.85
```

Looking at the output we can **rank the machines from best to worst**:

```
Machine 4: 75.6
Machine 2: 68.15
Machine 5: 21.5
Machine 3: 16.55
Machine 6: 6.65
Machine 1: -33.45
```

Based on the **utility values of machines 1 - 6** we can conclude that the first machine is expected NOT to be profitable, as its value is negative. On the other hand, all the other machines (2-6) are expected to be profitable.

The seventh machine can be ranked between 5th and 2nd ($Machine\ 7 : 23.85$). Since, its value is positive number it is expected to be profitable. Consequently, the **company should purchase the seventh machine**. *The code to compute the seventh machine is above.*

Appendix A Source code

```python
import numpy as np
import pandas as pd
import pystan

#%% The data
machines = pd.read_fwf('./factory.txt', header=None).values
machines_transposed = machines.T

stan_code_hierarchical = '''
data {
    int<lower=0> N;              // number of data points
    int<lower=0> K;              // number of groups
    int<lower=1,upper=K> x[N];  // group indicator
    vector[N] y;
}
parameters {
    real mu0;                    // prior mean
    real<lower=0> sigma0;        // prior std
    vector[K] mu;                // group means
    real<lower=0> sigma;         // common std
}
model {
```

```
    mu ~ normal(mu0, sigma0);
    y ~ normal(mu[x], sigma);
}
generated quantities {
    vector[K+1] ypred;
    real mu7;
    mu7 = normal_rng(mu0, sigma0);
    for (i in 1:K)
        ypred[i] = normal_rng(mu[i], sigma);
    ypred[K+1] = normal_rng(mu7, sigma);
}
'''

#%% fitting data into the stan model
model_hierarchical = pystan.StanModel(model_code=stan_code_hierarchical)
data_hierarchical = dict(
    N=machines_transposed.size,
    K=6,
    x=[
        1, 1, 1, 1, 1,
        2, 2, 2, 2, 2,
        3, 3, 3, 3, 3,
        4, 4, 4, 4, 4,
        5, 5, 5, 5, 5,
        6, 6, 6, 6, 6,
    ],
    y=machines_transposed.flatten()
)

#%% sampling
fit_hierarchical = model_hierarchical.sampling(data=data_hierarchical, n_jobs=-1)
print(fit_hierarchical)

#%% utility
utility = np.zeros(7)
ypred = fit_hierarchical.extract(permuted=True)['ypred']
for i in range(7):
    for j in range(0, len(ypred)):
        if (ypred[j, i] < 85):
            utility[i] -= 106
        else:
            utility[i] += (200-106)
    print('Machine {0}: {1}'.format(i+1, utility[i]/len(ypred)))
```