BDA - Assignment 8

Anonymous

Contents:

Exercise 1

In [18]:

```
import matplotlib.pyplot as plt
import arviz as az
import numpy as np
import pystan
import csv
# read data

file = open(r".\factory.txt", 'r')
temp = file.read().splitlines()

quality = [[] for i in range(5)]
qi = 0
for line in temp:
    values = line.split(" ")
    while True:
        try:
            values.remove("")
        except:
            break
    for i in range(len(values)):
        quality[qi].append(int(values[i]))
    qi += 1
```

Exercise 1.1

Separate Model

In [34]:

```
separate_model = '''
data {
    int<lower=0> N;
    int<lower=0> J;
    matrix[N, J] y;
}

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


model {
    // priors
    for (j in 1:J){
        mu[j] ~ normal(0, 50);
        sigma[j] ~ inv_chi_square(1);
    }

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

generated quantities {
    real ypred[J];
    matrix[J, N] log_lik;

    for (j in 1:J) {
        ypred[j] = normal_rng(mu[j], sigma[j]);
    }

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

In [35]:

```
data = {
    "N": len(quality),
    "J": len(quality[0]),
    "y": quality
}

posterior_separate = pystan.StanModel(model_code=separate_model)
separate_fit = posterior_separate.sampling(data=data)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_9d1766a239a9c884cfd0176e05900f
9f NOW.
```

Pooled Model

In [36]:

```
pooled_model = '''
data {
    int<lower=0> N;
    int<lower=0> J;
    matrix[N, J] y;
}

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


model {
    // priors
    mu ~ normal(0, 10);
    sigma ~ inv_chi_square(10);

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

generated quantities {
    real ypred;
    matrix[J, N] log_lik;

    ypred = normal_rng(mu, sigma);


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

In [37]:

```
data = {
    "N": len(quality),
    "J": len(quality[0]),
    "y": quality
}

posterior_pooled = pystan.StanModel(model_code=pooled_model)
pooled_fit = posterior_pooled.sampling(data=data)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_705df208f704b022b14486965dfd6d
96 NOW.
```

Hierarchical Model

In [38]:

```
hierarchical_model = '''
data {
    int<lower=0> N;
    int<lower=0> J;
    matrix[N, J] y;
}

parameters {
    real mu;
    real<lower=0> sigma;
    real theta[J];
    real<lower=0> sigmaJ;

    real new_machine;
}


model {
    // priors
    mu ~ normal(0, 10);
    sigma ~ inv_chi_square(1);
    for (j in 1:J)
        theta[j] ~ normal(mu, sigma);
    new_machine ~ normal(mu, sigma);
    sigmaJ ~ inv_chi_square(1);

    // likelihood
    for (j in 1:J){
        y[,j] ~ normal(theta[j], sigmaJ);
        y[,j] ~ normal(new_machine, sigmaJ);
    }
}

generated quantities {
    real ypred[J];
    real yprednew;
    matrix[J, N] log_lik;

    for (j in 1:J) {
        ypred[j] = normal_rng(theta[j] , sigmaJ);
    }

    yprednew = normal_rng(new_machine , sigmaJ);

    for (i in 1:J) {
        for (j in 1:N) {
            log_lik[i][j] = normal_lpdf(y[j][i] |  theta[i], sigmaJ);
        }
    }

}
'''
```

In [44]:

```
data = {
    "N": len(quality),
    "J": len(quality[0]),
    "y": quality
}

posterior_hierarchical = pystan.StanModel(model_code=hierarchical_model)
hierarchical_fit = posterior_hierarchical.sampling(data=data)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_ad8316994ac17a162b9f9569bfdae7
e0 NOW.
```

Exercise 1.2

Separate Model

In [50]:

```
separate_data = az.convert_to_inference_data(separate_fit, log_likelihood='log_lik')
separate_loo = az.loo(separate_data, var_name='log_lik', pointwise=True)
separate_loo
```

```
C:\Users\User\anaconda3\lib\site-packages\arviz\stats\stats.py:812: UserWarning: Estim
ated shape parameter of Pareto distribution is greater than 0.7 for one or more sample
s. You should consider using a more robust model, this is because importance sampling
is less likely to work well if the marginal posterior and LOO posterior are very diffe
rent. This is more likely to happen with a non-robust model and highly influential obs
ervations.
  warnings.warn(
```

Out[50]:

```
Computed from 4000 posterior samples and 30 observations log-likelihood matrix.

         Estimate       SE
elpd_loo  -130.61     4.13
p_loo       11.35        -

There has been a warning during the calculation. Please check the results.
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.5]   (good)       23   76.7%
 (0.5, 0.7]   (ok)          3   10.0%
   (0.7, 1]   (bad)         3   10.0%
   (1, Inf)   (very bad)    1    3.3%
```

Pooled Model

In [51]:

```
pooled_data = az.convert_to_inference_data(pooled_fit, log_likelihood='log_lik')
pooled_loo = az.loo(pooled_data, var_name='log_lik', pointwise=True)
pooled_loo
```

Out[51]:

```
Computed from 4000 posterior samples and 30 observations log-likelihood matrix.

         Estimate       SE
elpd_loo  -136.29     3.35
p_loo        2.54        -
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.5]   (good)       30  100.0%
 (0.5, 0.7]   (ok)          0    0.0%
   (0.7, 1]   (bad)         0    0.0%
   (1, Inf)   (very bad)    0    0.0%
```

Hierarchical Model

In [52]:

```
hierarchical_data = az.convert_to_inference_data(hierarchical_fit, log_likelihood='log
hierarchical_loo = az.loo(hierarchical_data, var_name='log_lik', pointwise=True)
hierarchical_loo
```

Out[52]:

```
Computed from 4000 posterior samples and 30 observations log-likelihood matrix.

         Estimate       SE
elpd_loo  -126.54     2.70
p_loo        4.56        -
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.5]   (good)       27   90.0%
 (0.5, 0.7]   (ok)          3   10.0%
   (0.7, 1]   (bad)         0    0.0%
   (1, Inf)   (very bad)    0    0.0%
```

Exercise 1.3

From the book we see that $p_{loo-cv} = lppd - lppd_{loo-cv}$ where lppd is the log pointwise prediction density.

We also know that the lppd is equal to $\sum_{i=1}^{n} log(\frac{1}{S}\sum_{s=1}^{S} p(y_i|\theta^s))$

In [87]:

```
def compute_peff(log_densities, elpd):
    final_result = 0
    for s in range(log_densities.shape[2]):
        intermediary = np.zeros(log_densities.shape[0])
        for n in range(log_densities.shape[1]):
            intermediary += log_densities[:,n,s]
        final_result += np.mean(intermediary)
    final_result -= elpd
    return final_result

separate_log_densities = separate_fit.extract("log_lik")["log_lik"]
pooled_log_densities = pooled_fit.extract("log_lik")["log_lik"]
hierarchical_log_densities = hierarchical_fit.extract("log_lik")["log_lik"]

separate_peff = compute_peff(separate_log_densities, separate_loo[0])
print(f"The p_eff for the separate model is {separate_peff}")
pooled_peff = compute_peff(pooled_log_densities, pooled_loo[0])
print(f"The p_eff for the separate model is {pooled_peff}")
hierarchical_peff = compute_peff(hierarchical_log_densities, hierarchical_loo[0])
print(f"The p_eff for the separate model is {hierarchical_peff}")
```

```
The p_eff for the separate model is 7.62875426023075
The p_eff for the separate model is 1.410234999951797
The p_eff for the separate model is 2.7018203273021584
```

Exercise 1.4

In the case of the separate model, we see that the estimated shape parameter of the Pareto distribution is higher than 0.7 for 4 samples. Just as the warning states, these results alert us that the model is not robust and therefore unreliable, because it might be too biased.

The pooled model has 30 "good" values for the estimated shape parameter, so we can say that the PSIS-LOO values are reliable.

The hierarchical model has 27 "good" and 3 "ok" values for the estimated shape parameter, so we can say that the PSIS-LOO values are reliable.

Exercise 1.5

In [88]:

```
az.compare({
    'separate': separate_data,
    'pooled': pooled_data,
    'hierarchical': hierarchical_data
})
```

```
C:\Users\User\anaconda3\lib\site-packages\arviz\stats\stats.py:812: UserWarning: Estim
ated shape parameter of Pareto distribution is greater than 0.7 for one or more sample
s. You should consider using a more robust model, this is because importance sampling
is less likely to work well if the marginal posterior and LOO posterior are very diffe
rent. This is more likely to happen with a non-robust model and highly influential obs
ervations.
  warnings.warn(
```

Out[88]:

	rank	loo	p_loo	d_loo	weight	se	dse	warning	loo_scale
hierarchical	0	-126.537363	4.559376	0.000000	1.000000e+00	2.700342	0.000000	False	log
separate	1	-130.614959	11.354380	4.077595	3.624878e-14	4.131561	2.209957	True	log
pooled	2	-136.285117	2.535460	9.747754	0.000000e+00	3.347950	3.587888	False	log

Using arviz's comparison function, we can get a ranking of the models on the terms of PSIS-LOO. Once again, the function warns us that the separate model is not robust enough and it might be biased.

We notice that they're different, given that there is an obvious hierarchy in terms of the elpd values (loo in the table). According to theory, the model with higher elpd (deviance) is the better model to use. In my case, this is the hierarchical model. This is also supported by the "weight" column which tells us the probability of each model being chosen, and in this case we can easily see that the hierarchical model is the clear winner.

Exercise 1.6

In [89]:

```
Done.
```

```
  File "<ipython-input-89-27d1a7a9b78b>", line 1
    Done.
        ^
SyntaxError: invalid syntax
```

In [ ]: