Assessed Coursework 2

CID 01252821

Question 1

Part a

$$P(X = x|\theta) = \frac{e^{-\theta x}(\theta x)^{(x-1)}}{x!} = \frac{e^{-\theta x}e^{(x-1)\log(\theta x)}}{x!} =$$
$$= \frac{e^{-\theta x + x\log x + x\log\theta - \log\theta - \log x}}{x!} = \frac{x^{x-1}}{x!}e^{x(\log\theta - \theta) - \log\theta}.$$

Therefore, we get:

$$h(x) = \frac{x^{x-1}}{x!},$$
$$g(x) = x,$$
$$B(\theta) = \log\theta - \theta,$$
$$A(\theta) = \log\theta.$$

We have that the Borel distribution belongs to the 1-parameter exponential family.

Part b

We showed that the Borel distribution belongs to the 1-parameter exponential family, and together with `Theorem 3.3` from the lecture notes we have that there exists a sufficient statistic such that:

$$T = \sum_{i=1}^{n} g_i(x_i) = \sum_{i=1}^{n} x_i,$$

which is also minimal sufficient and complete.

Part c

We know that $T = \sum_{i=1}^{n} x_i$ is a complete sufficient statistic for θ, and $T \sim Borel(\theta)$, meaning $E(T) = \frac{1}{1-\theta}$.

$$E(1 - n\{\sum_{i=1}^{n} X_i\}^{-1}) = 1 - nE(\{\sum_{i=1}^{n} X_i\}^{-1}).$$

Part d

We know that $T = \sum_{i=1}^{n} x_i$ is a complete sufficient statistic for θ, and $T \sim Borel(\theta)$, meaning $E(T) = \frac{1}{1-\theta}$. An unbiased estimator ψ for θ would mean $E(\psi) = \theta$. Let

$$\psi = 2 - \frac{2}{n}\sum_{i=1}^{n} \frac{1}{X_i}$$

Therefore,

$$E(\psi) = E(2 - \frac{2}{n}\sum_{i=1}^{n}\frac{1}{X_i}) = 2 - \frac{2}{n}E(\sum_{i=1}^{n}\frac{1}{X_i}) = 2 - \frac{2}{n}\sum_{i=1}^{n}E\frac{1}{X_i} =$$

$$= 2 - 2(1 - \frac{\theta}{2}) = \theta.$$

Therefore $\psi = 2 - \frac{2}{n}\sum_{i=1}^{n}\frac{1}{X_i}$ is an unbiased estimator for θ.

The estimator doesn't fit the SQM because the expectation is not taken over X, the distribution of the observations, but it uses the inverse of the observations.

Part e

mapper.py:

```
#!/usr/bin/env python

import sys

for line in sys.stdin:
    # remove leading or trailing whitespace
    line = line.strip()
    # extract input values
    x, s, w = line.split(",")

    # construct key s_w
    key = s + "_" + w
    # construct composite value <x, count, theta>
    # dummy 1 and 0 respectively
    value = f"{int(x)}_1_0"
    # key-value pair for each district <d, x_r>
    print(key, value, sep="\t")
```

reducer.py:

```
#!/usr/bin/env python

import sys
from collections import Counter


current_key = None
key = None
current_count = Counter()


for line in sys.stdin:
    line = line.strip()
    key, value = line.split("\t")
    x, n, theta = value.split("_")
    x = int(x)
    n = int(n)
    theta = float(theta)
```

```
    if current_key == key:
        current_count[key] += n
        current_count["sum"] += x
    else:
        if current_key is not None:
            current_n = current_count[current_key]
            current_sum = current_count["sum"]
            est = 1 - current_n / current_sum
            print(
                current_key, f"{str(current_sum)}_{str(current_n)}_{str(est)}", sep="\t"
            )

        current_count = Counter()
        current_count[key] += n
        current_count["sum"] += x

        current_key = key


if current_key == key:
    current_n = current_count[current_key]
    current_sum = current_count["sum"]
    est = 1 - current_n / current_sum
    print(current_key, f"{str(current_sum)}_{str(current_n)}_{str(est)}", sep="\t")
```

Bash command:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
  -libjars $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
  -input /shared_data/supermarkets.csv \
  -output /users/jtl22/ass2/output \
  -file mapper.py \
  -mapper 'python3 mapper.py' \
  -file reducer.py \
  -combiner 'python3 reducer.py' \
  -file reducer.py \
  -reducer 'python3 reducer.py'
```

Theta estimations for each supermarket and every day of week:

Supermarket	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Asdi	0.1973	0.2525	0.2275	0.2745	0.1749	0.6019	0.6995
Morrisons	0.1504	0.2010	0.1783	0.2235	0.1225	0.5430	0.6475
Sainsburys	0.2495	0.3027	0.2709	0.3242	0.2252	0.6516	0.7487
Tesco	0.3009	0.3535	0.3236	0.3743	0.2770	0.6964	0.8003
Waitrose	0.0996	0.1531	0.1249	0.1733	0.0765	0.5028	0.5956

Question 2

Hadoop Job 1

`mapper1.py`:

```python
#!/usr/bin/env python

import sys

input_file = sys.stdin
next(sys.stdin)
for line in input_file:
    # remove leading or trailing whitespace
    line = line.strip()
    # extract input values
    features = line.split(",")
    # diagnosis as key
    key = features[0]
    r, t, s, c = features[1:5]
    value = f"{r}_{t}_{s}_{c}"
    # key-value pair <diagnosis, r_t_s_c>
    print(key, value, sep="\t")
```

reducer1.py:

```python
#!/usr/bin/env python

import sys
from collections import Counter

import numpy as np

overall_n = 413
current_key = None
key = None
current_count = Counter()


for line in sys.stdin:
    line = line.strip()
    key, value = line.split("\t")
    r, t, s, c = map(float, value.split("_"))

    if current_key == key:
        current_count[key] += 1
        current_x = np.array([[r], [t], [s], [c]])
        current_count["X"] += current_x
        current_count["X_Xt"] += current_x @ np.transpose(current_x)

    else:
        if current_key is not None:
            current_n = current_count[current_key]
            current_mu = current_count["X"] / current_n
            cov_matrix = (current_count["X_Xt"] / current_n) - np.matmul(
                current_mu, np.transpose(current_mu)
            )

            print(
                current_key,
```

```
                np.transpose(current_mu),
                cov_matrix.tolist(),
                current_n / overall_n,
                sep="\t",
            )

        current_count = Counter()
        current_count[key] += 1
        current_x = np.array([[r], [t], [s], [c]])
        current_count["X"] += current_x
        current_count["X_Xt"] += current_x @ np.transpose(current_x)

        current_key = key

if current_key == key:
    current_n = current_count[current_key]
    current_mu = current_count["X"] / current_n
    cov_matrix = (current_count["X_Xt"] / current_n) - np.matmul(
        current_mu, np.transpose(current_mu)
    )

    print(
        current_key,
        np.transpose(current_mu),
        cov_matrix.tolist(),
        current_n / overall_n,
        sep="\t",
    )
```

Bash command:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
  -libjars $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
  -input /shared_data/breast_cancer_train.csv \
  -output /users/jtl22/ass2/q2/output \
  -file mapper1.py \
  -mapper 'python3 mapper1.py' \
  -file reducer1.py \
  -reducer 'python3 reducer1.py'
hadoop fs -get /users/jtl22/ass2/q2/output/part-00000 output_file.txt
```

Results:

(i) Estimated mean parameters $\mu_M = \frac{1}{n_M}\sum_1^n X_i$ and $\mu_B = \frac{1}{n_B}\sum_1^n X_i$:

$$\mu_M = (17.46, 21.63, 0.10, 0.14)$$

$$\mu_B = (12.22, 17.93, 0.09, 0.08)$$

(ii) Estimated covariances Σ_M and Σ_B:

$$\Sigma_M = \begin{bmatrix} 9.836 & 0.455 & -0.005 & 0.018 \\ 0.455 & 14.91 & -0.007 & -0.002 \\ -0.005 & -0.007 & 0.0002 & 0.0005 \\ 0.018 & -0.002 & 0.0005 & 0.003 \end{bmatrix}$$

$$\Sigma_B = \begin{bmatrix} 2.774 & -0.426 & -0.004 & 0.004 \\ -0.426 & 16.54 & -0.014 & -0.011 \\ -0.004 & -0.014 & 0.0002 & 0.0003 \\ 0.004 & -0.011 & 0.0003 & 0.001 \end{bmatrix}$$

(iii)

Hadoop Job 2

mapper2.py:

```
#!/usr/bin/env python

import sys


input_file = sys.stdin
next(sys.stdin)
for line in input_file:
    # remove leading or trailing whitespace
    line = line.strip()
    # extract input values
    features = line.split(",")
    # diagnosis as key
    key = features[0]
    r, t, s, c = features[1:5]
    value = f"{r}_{t}_{s}_{c}"
    # key-value pair <diagnosis, r_t_s_c>
    print(key, value, sep="\t")
```

reducer2.py:

```
#!/usr/bin/env python

import sys
from collections import Counter

import numpy as np

current_key = None
key = None
current_count = Counter()

print("D", "all", "#M", "#B", sep="\t")

est = {}
with open("output_file.txt", "r") as f:
    for line in f:
        line = line.strip()
        key, mu, sigma, p = line.split("\t")
```

```
        mu_values = mu.strip("[]").split()
        float_mu_values = [float(val) for val in mu_values]
        mu = np.transpose(np.array([float_mu_values]))

        sigma = np.array(eval(sigma))
        p = float(p)

        est[key] = {"mu": mu, "sigma": sigma, "p": p}

K = 2 * (np.log(est["B"]["p"]) - np.log(est["M"]["p"]))

results = {}
for line in sys.stdin:
    line = line.strip()
    key, value = line.split("\t")
    r, t, s, c = map(float, value.split("_"))

    if current_key == key:
        current_count[key] += 1
        current_x = np.array([[r], [t], [s], [c]])
        part1 = (
            np.transpose(current_x - est["B"]["mu"])
            @ np.linalg.inv(est["B"]["sigma"])
            @ (current_x - est["B"]["mu"])
        )
        part2 = np.log(np.linalg.det(est["B"]["sigma"]))
        part3 = (
            np.transpose(current_x - est["M"]["mu"])
            @ np.linalg.inv(est["M"]["sigma"])
            @ (current_x - est["M"]["mu"])
        )
        part4 = np.log(np.linalg.det(est["M"]["sigma"]))
        current_res = part1 + part2 - part3 - part4 > K

        current_count["y=M"] += current_res  # y=M

    else:
        if current_key is not None:
            current_n = current_count[current_key]

            results[current_key] = {
                "count": current_n,
                "y=M": current_count["y=M"][0][0],
                "y=B": current_n - current_count["y=M"][0][0],
            }

        current_count = Counter()
        current_count[key] += 1
        current_x = np.array([[r], [t], [s], [c]])
        part1 = (
            np.transpose(current_x - est["B"]["mu"])
            @ np.linalg.inv(est["B"]["sigma"])
            @ (current_x - est["B"]["mu"])
```

```
        )
        part2 = np.log(np.linalg.det(est["B"]["sigma"]))
        part3 = (
            np.transpose(current_x - est["M"]["mu"])
            @ np.linalg.inv(est["M"]["sigma"])
            @ (current_x - est["M"]["mu"])
        )
        part4 = np.log(np.linalg.det(est["M"]["sigma"]))
        current_res = part1 + part2 - part3 - part4 > K

        current_count["y=M"] += current_res  # y=M

        current_key = key

if current_key == key:
    current_n = current_count[current_key]

    results[current_key] = {
        "count": current_n,
        "y=M": current_count["y=M"][0][0],
        "y=B": current_n - current_count["y=M"][0][0],
    }


sensitivity = (results["M"]["y=M"]) / (results["M"]["y=M"] + results["M"]["y=B"])
specifcity = (results["B"]["y=B"]) / (results["B"]["y=B"] + results["B"]["y=M"])


accuracy = (results["M"]["y=M"] + results["B"]["y=B"]) / (
    results["M"]["y=M"]
    + results["M"]["y=B"]
    + results["B"]["y=B"]
    + results["B"]["y=M"]
)
precision = results["M"]["y=M"] / (results["M"]["y=M"] + results["B"]["y=M"])
print(sensitivity, specifcity, accuracy, precision)
```

Bash command:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
  -libjars $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
  -input /shared_data/breast_cancer_test.csv \
  -output /users/jtl22/ass2/q2/output_standard \
  -file output_file.txt \
  -file mapper2.py \
  -mapper 'python3 mapper2.py' \
  -file reducer2.py \
  -reducer 'python3 reducer2.py'
```

Results:

Sensitivity = True Positives / (True Positives + False Negatives) = 0.90

Specificity = True Negatives / (True Negatives + False Positives) = 0.94

Accuracy = (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False

Negatives) = 0.93

Precision = True Positives / (True Positives + False Positives) = 0.84