Assessed Coursework 1

CID 01252821

Question 1

The article by Watson and Holmes (2016) addresses the issue that making decisions based on predictions from probabilistic models may be sensitive to model misspecification, arising from the fact that there is almost no formal guidance on how to assess the impact of model approximation on decision making. The authors argue that "the consequence of statistical model misspecification is contextual, and hence shoul be dealt with under a decision theoretic framework". In Section 4 "formal methods for summarizing decision stability, by exploring the consequence of misspecification within local neighborhoods around the approximate model" are presented. Then three principles defining these methods are introduced in order to define the problem and suggest a robust solution: Context, Consequence, and Coherence. The Context principle suggests that our assessment of a model will depend on the task and the expected loss should be observed under model specification. The Consequence principle only takes into account the states that enter into the loss function, therefore the marginal posterior model would concern only these states. The authors build a neighbourhood of "close" models around the marginal with the idea of studying their expected loss, and the possibility of constructing a nonparametric extension of the model. The Kullaback-Leibler Neighbourhoods are constructed, starting with KL properties, after that showing that the use of KL-divergence leads to a least favourable distribution solution with simple form, stating the distribution of minimum expected loss. Finally, the uniqueness of KL-divergence under the condition of guaranteeing coherent Bayesian updating is shown, meaning KL is the only divergence measure to provide coherent updating of the local least favourable distribution. These results should assist in decision making in the context of complex big data statistical problems, which cover many different fields, for example (bio)medicine where decision are crucial to the health and life of the humanity.

Question 2

Part a

Let's denote the probability density functions of $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$ as $f_1(x) = \frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2}$ and $f_2(x) = \frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2}$ respectively. We have that $f_1(x) > 0 \Rightarrow f_2(x)$ (as well as $f_2(x) > 0 \Rightarrow f_1(x)$ because our distributions are both univariate normal distributions). Therefore, for the KL-divergence of f_2 from f_1 we have:

$$KL(f_1||f_2) = \int_{-\infty}^{\infty} f_1(x)\log\frac{f_1(x)}{f_2(x)}dx = \int_{-\infty}^{\infty} f_1(x)\log(\frac{\sigma_2}{\sigma_1}e^{-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2}e^{\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2})dx =$$

$$= \int_{-\infty}^{\infty} f_1(x)\left(\log\frac{\sigma_2}{\sigma_1} - \frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2 + \frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2\right)dx =$$

$$= \log\frac{\sigma_2}{\sigma_1}\int_{-\infty}^{\infty} f_1(x)dx - \frac{1}{2}\int_{-\infty}^{\infty} f_1(x)(\frac{x-\mu_1}{\sigma_1})^2dx + \frac{1}{2}\int_{-\infty}^{\infty} f_1(x)(\frac{x-\mu_2}{\sigma_2})^2dx =$$

$$= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} + \frac{1}{2\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)(x^2 - 2x\mu_2 + \mu_2^2)dx =$$

$$= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} + \frac{1}{2\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)x^2dx - \frac{\mu_2}{\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)xdx + \frac{\mu_2^2}{2\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)dx =$$

$$= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} + \frac{1}{2\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)(x-\mu_1+\mu_1)^2dx - \frac{\mu_1\mu_2}{\sigma_2^2} + \frac{\mu_2^2}{2\sigma_2^2} =$$

$$= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} + \frac{1}{2\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)(x-\mu_1)^2dx + \frac{\mu_1}{\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)xdx - \frac{\mu_1^2}{\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)dx + \frac{\mu_1^2}{2\sigma_2^2}\int_{-\infty}^{\infty} f_1(x)dx - \frac{\mu_1\mu_2}{\sigma_2^2} + \frac{\mu_2^2}{2\sigma_2^2} =$$

$$= \log\frac{\sigma_2}{\sigma_1} - \frac{1}{2\sigma_1^2} + \frac{\sigma_1^2}{2\sigma_2^2} + \frac{\mu_1^2}{2\sigma_2^2} - \frac{\mu_1\mu_2}{\sigma_2^2} + \frac{\mu_2^2}{2\sigma_2^2} =$$

$$= \frac{(\mu_1-\mu_2)^2 + \sigma_1^2 - \sigma_2^2}{2\sigma_2^2} + \log\frac{\sigma_2}{\sigma_1}$$

Note: If we calculate the KL-divergence of f_1 from f_2 we would achieve the same result but with swapped μ_i and σ_i.

Part b

The expected squared error loss is given by

$$\int_\Omega l(\hat{\omega}, \omega)f(\omega)d\omega,$$

where $l(\hat{\omega}, \omega)$ is a loss function, and $f(\omega)$ is a density function. In our case we want to work with the squared loss function

$$l(\hat{\omega}, \omega) = (\hat{\omega} - \omega)^2,$$

and let's denote with $f(x)$ the probability density function of $N(\theta|\mu, 1)$, or

$$f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(x-\mu)^2}.$$

i) For our problem for the expected squared error loss of $\hat{\theta}$ as a function of μ we have

$$\int_{-\infty}^{\infty} l(\hat{\theta}, \theta)f(\theta)d\theta = \int_{-\infty}^{\infty}(\hat{\theta}-\theta)^2 f(\theta)d\theta = \int_{-\infty}^{\infty} l(\hat{\theta}, \theta)f(\theta)d\theta = \int_{-\infty}^{\infty}(\hat{\theta}^2 - 2\hat{\theta}\theta + \theta^2)f(\theta)d\theta =$$

$$= \hat{\theta}^2\int_{-\infty}^{\infty} f(\theta)d\theta - 2\hat{\theta}\int_{-\infty}^{\infty}\theta f(\theta)d\theta + \int_{-\infty}^{\infty}\theta^2 f(\theta)d\theta =$$

$$= \hat{\theta}^2 - 2\hat{\theta}\mu + \int_{-\infty}^{\infty}(\theta - \mu + \mu)^2 f(\theta)d\theta =$$

$$= \hat{\theta}^2 - 2\hat{\theta}\mu + \int_{-\infty}^{\infty}(\theta-\mu)^2 f(\theta)d\theta + 2\mu\int_{-\infty}^{\infty}\theta f(\theta)d\theta - 2\mu^2\int_{-\infty}^{\infty} f(\theta)d\theta + \mu^2\int_{-\infty}^{\infty} f(\theta)d\theta =$$

$$= \hat{\theta}^2 - 2\hat{\theta}\mu + 1 + 2\mu^2 - 2\mu^2 + \mu^2 = (\hat{\theta} - \mu)^2 + 1$$

ii) In order to estimate $\hat{\theta}$ we want to minimize the above result. Differentiating with respect to $\hat{\theta}$ and solving the equation when equal to 0:

$$\frac{d}{d\hat{\theta}}\left((\hat{\theta}-\mu)^2+1\right)=0$$

$$2\hat{\theta}-2\mu=0$$

gives us

$$\hat{\theta}=\mu.$$

For an integer-valued estimator $\hat{\theta}\in Z$ and $\hat{\theta}=3$ this gives us the following interval for the real-valued mean μ:

$$\mu\in[2.5,3.5].$$

Here, I will note that for $\mu=2.5$ and $\mu=3.5$ there are two possible values for $\hat{\theta}$ which minimize the expression: 2, 3 and 3, 4 respectively.

Part c

Let's first calculate $KL(P_{|\mu}||P^*)$. From Part a, we know that the KL-divergence between two univariate normal distributions $N(\mu_1,\sigma_1^2)$ and $N(\mu_2,\sigma_2^2)$ is

$$KL(f_1||f_2)=\frac{(\mu_1-\mu_2)^2+\sigma_1^2-\sigma_2^2}{2\sigma_2^2}+\log\frac{\sigma_2}{\sigma_1}.$$

Therefore, for $P_{|\mu}=N(\mu,1)$ and $P^*=N(\mu_2,\sigma_2^2)$

$$KL(P_{|\mu}||P^*)=\frac{(\mu-\mu_2)^2+1-\sigma_2^2}{2\sigma_2^2}+\log\sigma_2.$$

Question 3

Part a

Non-branching graphs use node to node messages, where the marginal of the first node can be computed in linear time, lowering it from exponential time. General singly connected factor graphs use factor-to-variable (variable-to-factor) messages, allowing for them to be reused when evaluating other marginal inferences, saving computational resources. Message schedule is also introduced, it enforces a message can be sent from a node only when that node has received all requisite messages, eliminating the possibility of computing something more than once. In the sum-product algorithm the messages are computed in a schedule, thus enforcing the above definitions and their benefits.

Part b

First we define a directed graph model with `BayesianNetwork`, then we add the CPDs based on the given formula in the problem statement for $P(X_i=x_i|parents_G(X_i))$. The CPD for X_i will consist of pairs θ_i and $1-\theta_i$. Finally we check the CPDs are correctly defined.

```python
import networkx as nx
import pylab as plt
from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD

# Define a directed graph
model = BayesianNetwork([
    ('X_1', 'X_2'),
    ('X_2', 'X_3'),
    ('X_3', 'X_4'), ('X_3', 'X_5'),
    ('X_4', 'X_5'), ('X_4', 'X_7'),
    ('X_5', 'X_6'), ('X_5', 'X_7'),
    ('X_6', 'X_7'),
    ('X_7', 'X_8')
])

# if we want to draw our graph
pos = nx.circular_layout(model)
nx.draw(model, with_labels=True, node_size = 800, node_color = 'lightblue', pos=pos)
plt.show()

# Then, we define CPDs (Conditional Probability Distribution) for the graphical model.
cpd_1 = TabularCPD(
  variable='X_1', variable_card=2, values=[[0.9], [0.1]])
cpd_2 = TabularCPD(
  variable='X_2', variable_card=2, values=[[0.8, 0.2], [0.2, 0.8]],
  evidence=['X_1'], evidence_card=[2])
cpd_3 = TabularCPD(
  variable='X_3', variable_card=2, values=[[0.7, 0.3], [0.3, 0.7]],
  evidence=['X_2'], evidence_card=[2])
cpd_4 = TabularCPD(
  variable='X_4', variable_card=2,values=[[0.6, 0.4], [0.4, 0.6]],
  evidence=['X_3'], evidence_card=[2])
cpd_5 = TabularCPD(
  variable='X_5', variable_card=2,
  values=[[0.5, 0.5, 0.5, 0.5], [0.5, 0.5, 0.5, 0.5]],
  evidence=['X_3', 'X_4'], evidence_card=[2, 2])
cpd_6 = TabularCPD(
  variable='X_6', variable_card=2, values=[[0.4, 0.6], [0.6, 0.4]],
  evidence=['X_5'], evidence_card=[2])
cpd_7 = TabularCPD(
  variable='X_7', variable_card=2, values=[
  [0.3, 0.7, 0.7, 0.3, 0.7, 0.3, 0.3, 0.7],
  [0.7, 0.3, 0.3, 0.7, 0.3, 0.7, 0.7, 0.3]], evidence=['X_4', 'X_5', 'X_6'],
  evidence_card=[2, 2, 2])
cpd_8 = TabularCPD(
  variable='X_8', variable_card=2, values=[[0.2, 0.8], [0.8, 0.2]],
  evidence=['X_7'], evidence_card=[2])

# Associating the CPDs with the graph
model.add_cpds(cpd_1, cpd_2, cpd_3, cpd_4, cpd_5, cpd_6, cpd_7, cpd_8)

# check_model checks the CPDs are correctly defined
```

```
model.check_model()
```

For evaluating probabilities we are going to use `VariableElimination`.

```
from pgmpy.inference import VariableElimination

# Create an inference object by passing the model
inference = VariableElimination(model)
```

For the first probability $P(X_6 = 1|X_5 = 1)$ we have

```
# Part i
x6_1_x5_1 = inference.query(variables=['X_6'], evidence={'X_5': 1})
print(x6_1_x5_1.get_value(X_6 = 1))
```

which gives us $P(X_6 = 1|X_5 = 1) = 0.4000$.

For part ii, we can evaluate $P(X_4 = 1)$ via

```
# part ii
x4_prob = inference.query(variables=['X_4'])
print(x4_prob.get_value(X_4 = 1))
```

getting $P(X_4 = 1) = 0.4808$.

Next, we want to calculate $P(X_3 = 0|X_1 + X_8 = 1)$, given that these are binary variables we have two possibilities for $X_1 + X_8 = 1$: $X_1 = 0, X_8 = 1$ and $X_1 = 1, X_8 = 0$. Moreover, these cases do not have anything in common, so we have $P(X_1 \cap X_8) = 0$. Applying Bayes theorem:

$$P(X_3 = 0|X_1 + X_8 = 1) = \frac{P(X_1 + X_8 = 1|X_3 = 0)P(X_3 = 0)}{P(X_1 + X_8 = 1)} =$$

$$= \frac{\left(P(X_1 = 0, X_8 = 1|X_3 = 0) + P(X_1 = 1, X_8 = 0|X_3 = 0)\right).P(X_3 = 0)}{P(X_1 = 0, X_8 = 1) + P(X_1 = 1, X_8 = 0)}$$

These are values we can easily extract with Python. For $P(X_1 = 0, X_8 = 1|X_3 = 0)$ and $P(X_1 = 1, X_8 = 0|X_3 = 0)$ we use:

```
x1x8_x3_0 = inference.query(variables=["X_1", "X_8"], evidence={'X_3': 0})
print(x1x8_x3_0.get_value(X_1=0, X_8=1))
print(x1x8_x3_0.get_value(X_1=1, X_8=0))
```

getting $P(X_1 = 0, X_8 = 1|X_3 = 0) = 0.4726$ and $P(X_1 = 1, X_8 = 0|X_3 = 0) = 0.0316$.

For $P(X_3 = 0)$ we get $P(X_3 = 0) = 0.5960$ via:

```
x3_0 = inference.query(['X_3'],show_progress=False)
print(x3_0.get_value(X_3 = 0))
```

Finally, for $P(X_1 = 0, X_8 = 1)$ and $P(X_1 = 1, X_8 = 0)$ from

```
# Perform variable elimination to calculate the probability
x1x8 = inference.query(variables=["X_1", "X_8"])
print(x1x8.get_value(X_1=0, X_8=1))
print(x1x8.get_value(X_1=1, X_8=0))
```

we get $P(X_1 = 0, X_8 = 1) = 0.4510$ and $P(X_1 = 1, X_8 = 0) = 0.0501$.

Then, for $P(X_3 = 0|X_1 + X_8 = 1)$ we have

$$P(X_3 = 0|X_1 + X_8 = 1) = \frac{(0.4726 + 0.0316) * 0.5960}{0.4510 + 0.0501} = 0.5997$$

For Part iii, we have

$$E(S) = E(\sum_{i=1}^{8} X_i) = \sum_{i=1}^{8} EX_i = \sum_{i=1}^{8} p_i x_i$$

where p_i is the probability for x_i. In our case x_i is binary, therefore

$$E(S) = \sum_{i=1}^{8} P(X_i = 1).$$

We can do this in Python:

```
sum_x = 0
for node in list(model.nodes()):
    temp = inference.query([node])
    sum_x += temp.values[1]
sum_x
```

getting a result $E(S) = 3.2441856$.