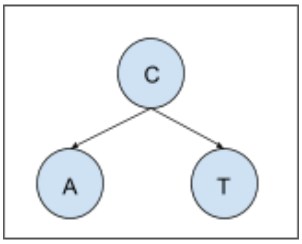
Homework 1: Generative Models & Decision Theory

SHREYA CHETAN PAWASKAR
pawaskas@uci.edu
12041645

Question 1:

a) *Draw a directed graphical model defining the joint distribution of C, A, and T.*

- C is the parent node and A & T are child nodes.
- T and A are conditionally independent given C.



---

b) *P(T = 1):*

- P(C = 1) = 0.2 (prior probability of having a cavity)
- P(T = 1 | C = 1) = 0.9 (probability of dental tool catching tooth if patient has cavity)
- P(T = 1 | C = 0) = 0.2 (probability of dental tool catching tooth if patient does not have cavity)

- Answer = P(T = 1 | C = 1) * P(C = 1) + P(T = 1 | C = 0) * P(C = 0)
  = (0.9 * 0.2) +(0.2 * (1 - 0.2))
  = 0.18 + 0.16

  P(T = 1)= 0.34

---

c) *P(C = 1 | T = 1):*

- P(C = 1) = 0.2 (probability of having a cavity)
- P(T = 1 | C = 1) = 0.9 (probability of the dental tool catching tooth if patient has a cavity)

- P(T = 1) = 0.34 (Answer b)

- Answer = (P(T = 1 | C = 1) * P(C = 1)) / P(T = 1)
  = (0.9 * 0.2) / 0.34
  = 0.18 / 0.34
  = 0.5294117647058824

  P(C = 1 | T = 1) ≈ 0.53

---

d) P(C = 1 | T = 1, A = 0)

- P(C = 1) = 0.2 (probability of having a cavity)
- P(T = 1 | C = 1) = 0.9 (probability of tool catching tooth given patient has cavity)
- P(A = 0 | C = 1) = 1 - P(A = 1 | C = 1) = 1 - 0.6 = 0.4 (probability of not having toothache given patient has cavity)

- P(T = 1, A = 0 | C = 1) = P(T = 1 | C = 1) * P(A = 0 | C = 1) = 0.9 * 0.4 = 0.36
- P(T = 1, A = 0) = P(T = 1, A = 0 | C = 1) * P(C = 1) + P(T = 1, A = 0 | C = 0) * P(C = 0)
  = P(T = 1 | C = 0) * P(A = 0 | C = 0) = 0.2 * 0.9 = 0.18

- P(T = 1, A = 0) = 0.36 * 0.2 + 0.18 * (1 - 0.2) = 0.072 + 0.144 = 0.216

- $$P(C = 1 \mid T = 1, A = 0) = \frac{P(T=1|C=1) * P(A=0|C=1) * P(C=1)}{\sum_{c \in \{0,1\}} P(T = 1|C = c) * P(A = 0|C = c) * P(C = c)}$$

  = (0.36 * 0.2) / 0.216 = 1/3

  P(C = 1 | T = 1, A = 0)≈ *0.33*

---

e) Are the random variables A and T independent? Justify your answer mathematically.

The independence of A and T can be evaluated by checking if the following condition holds for all values of a, t ∈ {0, 1}:

P(T = t, A = a) = P(T = t) * P(A = a)

If this condition is satisfied, then A and T are independent random variables.

- $P(T = 1, A = 1) = \sum_{c \in \{0, 1\}} P(T = 1 \mid C = c) \times P(A = 1 \mid C = c) \times P(C = c)$

  $= (0.9 \times 0.6 \times 0.2) + (0.2 \times 0.1 \times 0.8)$
  $= 0.108 + 0.016$
  $= 0.124$

- $P(T = 1) = P(T = 1 \mid C = 1) \times P(C = 1) + P(T = 1 \mid C = 0) \times P(C = 0)$

  $= 0.9 \times 0.2 + 0.2 \times 0.8$
  $= 0.18 + 0.16$
  $= 0.34$

- $P(A = 1) = P(A = 1 \mid C = 1) \times P(C = 1) + P(A = 1 \mid C = 0) \times P(C = 0)$

  $= 0.6 \times 0.2 + 0.1 \times 0.8$
  $= 0.12 + 0.08$
  $= 0.2$

$P(T = 1, A = 1) = 0.124 \neq P(T = 1) \times P(A = 1) = 0.34 \times 0.2 = 0.068$

<u>A and T are not marginally independent.</u>

Question 2:

a) *Derive equations for ln p(xn | tn = 1) and ln p(xn | tn = 0), the (natural) logarithms of the conditional probability density functions in Equations (2,3). For numerical robustness, simplify your answer so that it does not involve the exponential function.*

- N training instances
- D features
- Two classes

$$p\left(x_n \mid t_n=1\right)=\prod_{d=1}^D \frac{1}{\sqrt{2 \pi \sigma_{1 d}^2}} \exp \left\{-\frac{\left(x_{n d}-\mu_{1 d}\right)^2}{2 \sigma_{1 d}^2}\right\}$$

$$\therefore \ln p\left(x_n \mid t_n=1\right)=\ln \prod_{d=1}^D \frac{1}{\sqrt{2 \pi \sigma_{1 d}^2}} \exp \left\{-\frac{\left(x_{n d}-\mu_{1 d}\right)^2}{2 \sigma_{1 d}^2}\right\}$$

$$\therefore \ln p\left(x_n \mid t_n=1\right)=\sum_{d=1}^D\left(\ln \left(\frac{1}{\sqrt{2 \pi \sigma_{1 d}^2}} \exp \left\{-\frac{\left(x_{n d}-\mu_{1 d}\right)^2}{2 \sigma_{1 d}^2}\right\}\right)\right)$$

$$\therefore \ln p\left(x_n \mid t_n=1\right)=\sum_{d=1}^D\left(\ln \left(\frac{1}{\sqrt{2 \pi \sigma_{1 d}^2}}\right)+\ln \left(\exp \left\{-\frac{\left(x_{n d}-\mu_{1 d}\right)^2}{2 \sigma_{1 d}^2}\right\}\right)\right)$$

$$\therefore \ln p\left(x_n \mid t_n=1\right)=\sum_{d=1}^D\left(\ln \left(2 \pi \sigma_{1 d}^2\right)^{-1 / 2}+\left\{-\frac{\left(x_{n d}-\mu_{1 d}\right)^2}{2 \sigma_{1 d}^2}\right\}\right)$$

$$\therefore \ln p\left(x_n \mid t_n=1\right)=\sum_{d=1}^D\left(-\frac{\ln \left(2 \pi \sigma_{1 d}^2\right)}{2}-\frac{\left(x_{n d}-\mu_{1 d}\right)^2}{2 \sigma_{1 d}^2}\right)$$

Similarly for the second part,

$$\ln p\left(x_n \mid t_n=0\right)=\sum_{d=1}^D\left(-\frac{\ln \left(2 \pi \sigma_{0 d}^2\right)}{2}-\frac{\left(x_{n d}-\mu_{0 d}\right)^2}{2 \sigma_{0 d}^2}\right)$$

---

b)

Python Colab Code:

```
#import libraries needed
from area_roc import area_roc
import numpy as np
from sklearn import metrics
#to plot the curve
import matplotlib.pyplot as plt

# Load the gamma dataset
data = np.load("gamma.npy", allow_pickle=True).item()
```

```
#create training set
train_data = data["train"]
train_labels = data["trainLabels"]

#create testing set
test_data = data["test"]
test_labels = data["testLabels"]

# Estimate P(Y)
py = np.zeros(shape=(2,))
py[0] = np.sum(train_labels == 0) / float(train_labels.shape[0])
py[1] = np.sum(train_labels == 1) / float(train_labels.shape[0])

# Estimate P(X|Y)
num_dims = train_data.shape[1]
sigma_estimate = np.zeros(shape=(2, num_dims))
mu_estimate = np.zeros(shape=(2, num_dims))

# MLE estimates for mu and sigma
mu_estimate[0, :] = np.mean(train_data[train_labels == 0, ], 0)
mu_estimate[1, :] = np.mean(train_data[train_labels == 1, ], 0)
sigma_estimate[0, :] = np.var(train_data[train_labels == 0, ], 0)
sigma_estimate[1, :] = np.var(train_data[train_labels == 1, ], 0)
```

---

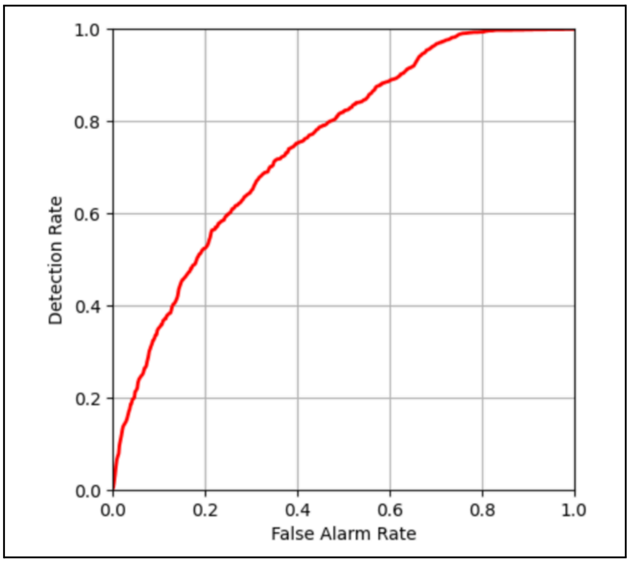
c)

Python Colab Code:

```
log_py = np.zeros(shape=(test_data.shape[0], 2))
for i in range(test_data.shape[0]):
   for y in range(2):
       log_py[i, y] = np.log(py[y]) + np.sum(-0.5 * ((test_data[i, :] -
mu_estimate[y, :])**2) / sigma_estimate[y, :]
                                              - 0.5 * np.log(2.0 * np.pi *
sigma_estimate[y, :]))

# Take max along class dimension
max_log_py = np.max(log_py, axis=1)
argmax_log_py = np.argmax(log_py, axis=1)

# Compute posterior probabilities by normalizing exp() of log probabilities
```

```
post_pr_y = np.exp(log_py - np.expand_dims(max_log_py, 1))
post_pr_y = post_pr_y / np.expand_dims(np.sum(post_pr_y, axis=1), 1)

assert np.allclose(np.sum(post_pr_y, axis=1), 1)

# Generate ROC Curve
aroc, _,_ = area_roc(post_pr_y[:, 1], test_labels, do_plot=True)
```

1. **Accuracy: 0.72897**
2. **So the accuracy is 72.89% on the test data.**
3. **ROC Curve for Naive Bayes Classification:**



---

d) Python Colab Code:

```
y_hat = argmax_log_py
num_pos = np.sum(test_labels == 1)
num_neg = np.sum(test_labels == 0)

true_pos = np.sum((y_hat == 1) * (test_labels == 1)) / num_pos
false_pos = np.sum((y_hat == 1) * (test_labels == 0)) / num_neg

accuracy = np.sum(y_hat == test_labels) / float(test_labels.shape[0])
print("Accuracy =", round(accuracy, 5))
print("True Positive Rate =", round(true_pos, 5), ", False Positive Rate =",
round(false_pos, 5))
```

1. MAP calculates the probability of a hypothesis given the data by combining the prior probability of the hypothesis with the likelihood of the data given the hypothesis.
2. The optimal Bayesian classification rule, minimizing errors.
3. It selects the class with the highest posterior probability, following the Maximum A Posteriori (MAP) decision rule.
4. **True Positive Rate = 0.911**
5. **False Positive Rate = 0.635**

---

e)

1. Let L_{FN} represent the loss when making a false negative decision.
2. Let L_{FP} the loss associated with a false positive).
3. We need to calculate $p(t = 1 \mid x)$ and $p(t = 0 \mid x)$.
4. Choose $\hat{t} = 1 \iff \frac{p(t = 1 \mid x)}{p(t = 0 \mid x)} > \frac{L_{FP}}{L_{FN}}$.
5. We will choose $\hat{t} = 1 \text{ when } \frac{p(t = 1 \mid x)}{p(t = 0 \mid x)} > \frac{1}{50}$

Code:

```
cost_rt = 1 / 50.0
y_hat = cost_rt < (post_pr_y[:,1] / post_pr_y[:, 0])

true_pos = np.sum((y_hat == 1) * (test_labels == 1)) / num_pos
false_pos = np.sum((y_hat == 1) * (test_labels == 0)) / num_neg

print("True Positive Rate =", round(true_pos, 5), ", False Positive Rate =",
round(false_pos, 5))
```

- **True Positive Rate = 0.967**
- **False Positive Rate = 0.70087**

Question 3:

- Let $p(x \mid T=0) = \theta_0 e^{-\theta_0 x}$ and $p(x \mid T=1) = \theta_1 e^{-\theta_1 x}$,

  where $\theta_0 = 1$ and $\theta_1 = 1/50 = 0.02$

- The optimal Bayesian classification rule can be written in terms of the log-likelihood ratio:

$$L(x) = \ln \frac{p(x \mid T=1)}{p(x \mid T=0)} = \ln \theta_1 - \theta_1 x - \ln \theta_0 + \theta_0 x = (\theta_0 - \theta_1)\, x + \ln \frac{\theta_1}{\theta_0}.$$

a)
With P(T = 0) = P(T = 1) = 0.5, we want to find the threshold c that maximizes the probability of correct prediction.

$$\begin{aligned} L(x) &\leq 0 \\ (\theta_0 - \theta_1)x &\leq \ln \frac{\theta_0}{\theta_1} \\ x &\leq \frac{1}{\theta_0 - \theta_1} \ln \frac{\theta_0}{\theta_1} \\ x &\leq \frac{1}{0.98} \ln(50) = 3.99186 \\ &\approx 3.99. \end{aligned}$$

- So the optimal threshold is **c = 3.99.**
- **Predict T = 0 if x <= 3.99**
- **Else predict T = 1.**

---

b)

Class T=0 is the most probable if $p(x \mid T=1)(1-q) \leq p(x \mid T=0)q$,

where $q = P(T=0) = 0.99$.

In terms of the log-likelihood ratio, it will be:

$$
\begin{aligned}
L(x) &\leq \ln \frac{q}{1-q} \\
(\theta_0 - \theta_1)x &\leq \ln \frac{q}{1-q} + \ln \frac{\theta_0}{\theta_1} \\
x &\leq \frac{1}{\theta_0 - \theta_1}\left(\ln \frac{q}{1-q} + \ln \frac{\theta_0}{\theta_1}\right) \\
x &\leq \frac{1}{0.98}(\ln(99) + \ln(50)) \\
x &\leq \frac{1}{0.98}(8.50714) \approx 8.68.
\end{aligned}
$$

- **Threshold c = 8.68**
- The processors are correctly functioning if :
  - **Predict T = 0 if x <= 8.68, else predict T = 1.**

---

c)

λ_{01} is the cost of prediction of correctly functioning processors is defective.

The cost of prediction of defective processor is correct equals $\lambda_{10} = 500\lambda_{01}$

.

T=0 has the smallest loss when:

$$
\begin{aligned}
L(x) &\leq \ln \frac{q}{1-q} + \ln \frac{\lambda_{01}}{\lambda_{10}}, \\
(\theta_0 - \theta_1)x &\leq \ln \frac{q}{1-q} + \ln \frac{\lambda_{01}}{\lambda_{10}} + \ln \frac{\theta_0}{\theta_1} \\
x &\leq \frac{1}{\theta_0 - \theta_1}\left(\ln \frac{q}{1-q} + \ln \frac{\lambda_{01}}{\lambda_{10}} + \ln \frac{\theta_0}{\theta_1}\right) \\
x &\leq \frac{1}{0.98}(\ln(99) - \ln(500) + \ln(50)) \\
x &\leq \frac{1}{0.98}(4.595 - 6.214 + 3.912) \\
x &\leq \frac{1}{0.98}(2.293) \\
x &\leq 2.33979591837 \\
&\approx 2.34.
\end{aligned}
$$

- **Threshold c =2.34**
- Processors are correctly functioning if **(Predict T = 0) if the observed time is below c = 2.34.**
- **Else predict T=1.**