ECE 539: Homework 2

Luke Yichao Zhang

September 23, 2024

1 Exercise 1 - Exercise Problem 7, Section 2.6.8 (D2L):

1.1 (a)

From the statement, it can be inferred that:
1. For either test,

$$P(D_1 = 1|H = 0) = P(D_2 = 1|H = 0) = 0.1 \tag{1}$$

$$P(D_1 = 0|H = 1) = P(D_2 = 0|H = 1) = 0.01 \tag{2}$$

2. For healthy people($H = 0$), the probability of having false positive twice is 0.02

$$P(D_1 = 1, D_2 = 1|H = 0) = 0.02 \tag{3}$$

Given that $H = 0$, so

$$P(D_1 = 1, D_2 = 1) = 0.02 \tag{4}$$

$$P(D_1 = 0) = 1 - P(D_1 = 1) = 0.9 \tag{5}$$

$$P(D_1 = 1) = 0.1 = P(D_1 = 1, D_2 = 1) + P(D_1 = 1, D_2 = 0) \tag{6}$$

Therefore, according to (6) and (7), it's easy to get

$$P(D_1 = 1, D_2 = 0) = 0.1 - 0.02 = 0.08 \tag{7}$$

Similarly, we have

$$P(D_1 = 0, D_2 = 1) = P(D_2 = 1) - P(D_1 = 1, D_2 = 1) = 0.08 \tag{8}$$

$$P(D_1 = 1, D_2 = 0) = 0.1 - 0.02 = 0.08 \tag{9}$$

Then we have

$$P(D_1 = 0) = 1 - P(D_1 = 1) = 0.9 \tag{10}$$

$$P(D_1 = 0, D_2 = 0) = P(D_1 = 0) - P(D_1 = 0, D_2 = 1) = 0.82 \tag{11}$$

We have the joint probability table as follows:

	$P(D_2) = 1$	$P(D_2) = 0$
$P(D_1) = 1$	0.02	0.08
$P(D_1) = 0$	0.08	0.82

1.2 (b)

The probability that one is diseased after both tests can be written as follows:

$$P(H = 1|D_1 = 1, D_2 = 1) \tag{12}$$

Apply Bayes' Theorem, we have

$$P(H = 1|D_1 = 1, D_2 = 1) = \frac{P(D_1 = 1, D_2 = 1|H = 1)P(H = 1)}{P(D_1 = 1, D_2 = 1)} \tag{13}$$

For infected patients($H = 1$), the test outcomes are conditionally independent. And for either test,

$$P(D_1, D_2|H = 1) = P(D_1|H = 1)P(D_2|H = 1) \tag{14}$$

$$P(D_1 = 0|H = 1) = P(D_2 = 0|H = 1) = 0.01 \tag{15}$$

Therefore we have

$$P(D_1 = 1, D_2 = 1|H = 1) = (1 - 0.01) \times (1 - 0.01) = 0.9801 \tag{16}$$

According to (a) and D2L-2.6.5.

$$P(D_1 = 1, D_2 = 1|H = 0) = 0.02 \tag{17}$$

$$P(H = 1) = 0.0015 \tag{18}$$

And we have

$$\begin{aligned}
&P(D_1 = 1, D_2 = 1)\\
&= P(D_1 = 1, D_2 = 1, H = 0) + P(D_1 = 1, D_2 = 1, H = 1)\\
&= P(D_1 = 1, D_2 = 1|H = 0)P(H = 0) + P(D_1 = 1, D_2 = 1|H = 1)P(H = 1)\\
&= 0.02 \times (1 - 0.0015) + 0.9801 \times 0.0015\\
&= 0.02144
\end{aligned}$$

Finally, we have

$$P(H = 1|D_1 = 1, D_2 = 1) = \frac{0.9801 \times 0.0015}{0.02144} = 0.06857 \tag{19}$$

2 Exercise 2 - Maximum A Posteriori Classification for Marathon Runners

2.1 (a)

```
def normal_distribution(x, mean, std_dev):
    """
    Compute the probability density function for the normal distribution.
    """
    p = (1/(math.sqrt(2 * math.pi) * std_dev)) * math.exp(-((x-mean) ** 2) /
    (2*(std_dev ** 2)))

    return p

t = np.array([2.5, 3., 3.5])

print('P(t|pro) \t', [normal_distribution(t, mean_pro, std_dev_pro) for t in t])
print('P(t|amateur) \t', [normal_distribution(t, mean_amateur, std_dev_amateur)
for t in t])
```

Participant's time (t)	2.5	3	3.5
$P(t\|pro)$	1.9947114020071635	0.08764150246784272	7.433597573671502e-06
$P(t\|amateur)$	0.008863696823876015	0.10798193302637613	0.48394144903828673

2.2 (b)

```
def posterior_probability(time):
    """
    Compute the posterior probability of a runner being a 'pro'
    or 'amateur' given their marathon time.
    """
    # Calculate likelihoods
    L_pro = normal_distribution(time, mean_pro, std_dev_pro)
    L_amateur = normal_distribution(time, mean_amateur, std_dev_amateur)

    # Calculate posteriors
    P_pro_given_time = (L_pro*P_pro)/((L_pro*P_pro)+(L_amateur*P_amateur))
    P_amateur_given_time = (L_amateur*P_amateur)/((L_pro*P_pro)+(L_amateur*P_amateur))

    return P_pro_given_time, P_amateur_given_time

for i in range(3):
    P_pro_given_time, P_amateur_given_time = posterior_probability(t[i])
    print('P(pro|t) \t', P_pro_given_time)
    print('P(amateur|t) \t', P_amateur_given_time)
```

Participant's time (t)	2.5	3	3.5
$P(pro\|t)$	0.9615454997393204	0.08272132884593233	1.706722740794798e-06
$P(amateur\|t)$	0.03845450026067968	0.9172786711540677	0.9999982932772592

2.3 (c)

```
def classify_runner(time):
    """
    Classify a runner as 'pro' or 'amateur' based on their marathon time using MAP.
    """
    # Calculate posteriors
    P_pro_given_time, P_amateur_given_time = posterior_probability(time)

    # Classify based on the highest posterior
    preds = []

    if P_pro_given_time > P_amateur_given_time:
        decision = 'Pro'  # assign 'pro' or 'amateur' according to MAP rule
    else:
        decision = 'Amateur'

    preds.append(decision)
    return preds

# Test the function
for i in range(3):
    print('If time is \t', t[i])
    print('P(pro|t) \t', classify_runner(t[i]))
```

Participant's time (t)	2.5	3	3.5
$MAPDecision$	**PRO**	**AMATEUR**	**AMATEUR**

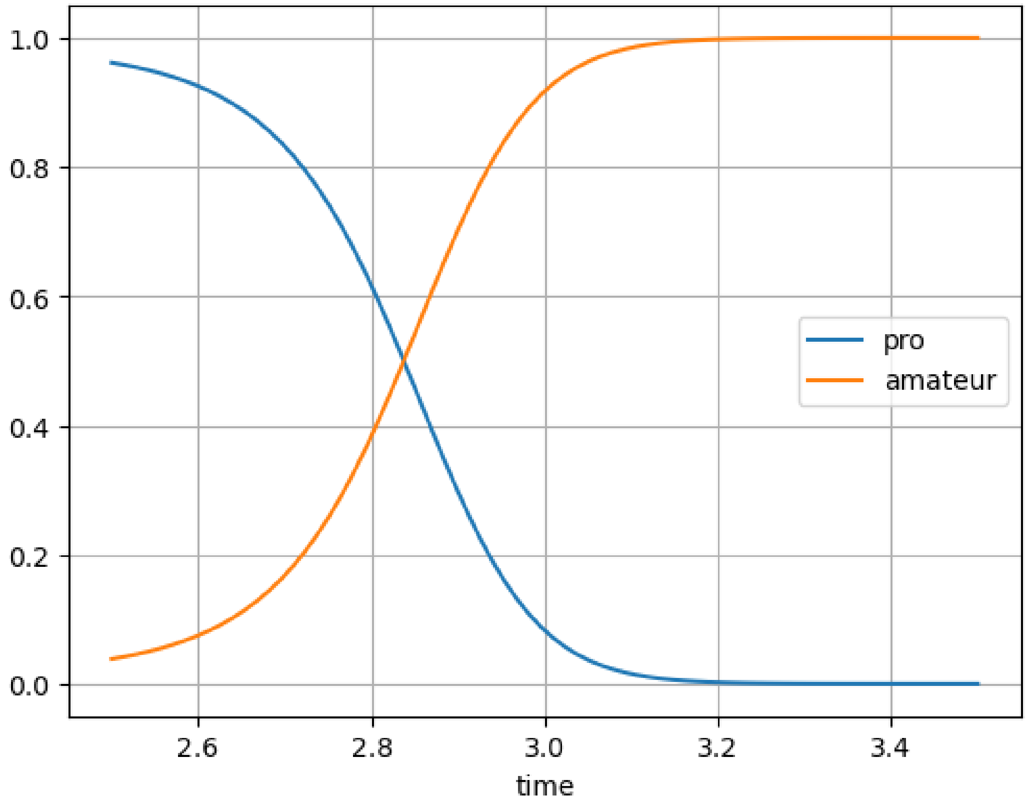
2.4 (d)

```
from matplotlib import pyplot as plt
t = np.linspace(2.5, 3.5, 100)  # 100 times between 2.5 and 3.5
# P_pro_given_time, P_amateur_given_time = ...

# Calculate posterior probabilities for each time point
P_pro_given_time = np.array([posterior_probability(time)[0] for time in t])
P_amateur_given_time = np.array([posterior_probability(time)[1] for time in t])

plt.plot(t, P_pro_given_time, label='pro')
plt.plot(t, P_amateur_given_time, label='amateur')
plt.xlabel('time')
plt.grid('on')
plt.legend()
plt.show()
```

The output is



The intersection is somewhere near time = 2.83

2.5 (e)

```
def sample_amateur_times(n):
    return np.random.randn(n)*std_dev_amateur + mean_amateur

def sample_pro_times(n):
    return np.random.randn(n)*std_dev_pro + mean_pro

n = 1000

amateur_times = sample_amateur_times(n)
pro_times = sample_pro_times(n)

num_mis_amateur = 0
num_mis_pro = 0

for j in range(1000):
    decisions = classify_runner(amateur_times[j])
    if decisions != ['Amateur']:
        num_mis_amateur = num_mis_amateur + 1

P_miss_cls_amateur = num_mis_amateur / 1000

# Print the numbers and probability for misclassification
print('Prob Misclassifying Amateurs', num_mis_amateur, P_miss_cls_amateur)

for j in range(1000):
    decisions = classify_runner(pro_times[j])
    if decisions != ['Pro']:
        num_mis_pro = num_mis_pro + 1

P_miss_cls_pro = num_mis_pro / 1000

# Print the numbers and probability for misclassification
print('Prob Misclassifying Pros', num_mis_pro, P_miss_cls_pro)
```