COEN 240 Machine Learning

Homework #4

Name: Jinhao Wang   ID: 4302178

Problem1:

$$P(data|\theta) = (\tfrac{2}{3}\theta)^2 \cdot (\tfrac{1}{3}\theta)^3 \cdot (\tfrac{2}{3}(1-\theta))^3 \cdot (\tfrac{1}{3}(1-\theta))^2$$

$$= 4 \cdot (\tfrac{\theta}{3})^5 \cdot 8 \cdot (\tfrac{1-\theta}{3})^5$$

$$\ln P(data|\theta) = \ln 32 + 5 \cdot \ln(\tfrac{\theta}{3}) + 5 \cdot \ln(\tfrac{1-\theta}{3})$$

$$\frac{\partial \ln P(data|\theta)}{\partial \theta} = \frac{15}{\theta} + \frac{15}{1-\theta} = 0$$

$$\hat{\theta}_{ML} = \frac{1}{2}$$

Problem 2:

$$P(data|\theta) = \prod_{i=1}^{n} \theta \cdot x_0^{\theta} \cdot x_i^{-\theta-1}$$

$$\ln P(data|\theta) = \ln\left(\prod_{i=1}^{n} \theta \cdot x_0^{\theta} \cdot x_i^{-\theta-1}\right)$$

$$= \ln(\theta^n) + \ln(x_0^{\theta \cdot n}) + \sum_{i=1}^{n} (\ln x_i^{-\theta-1})$$

$$= n \cdot \ln\theta + n \cdot \theta \cdot \ln x_0 - (\theta+1) \cdot \sum_{i=1}^{n} \ln x_i$$

$$\frac{\partial \ln P(data|\theta)}{\partial \theta} = \frac{n}{\theta} + n \cdot \ln x_0 - \sum_{i=1}^{n} \cdot \ln x_i = 0$$

$$\hat{\theta}_{ML} = \frac{n}{\sum_{i=1}^{n} \ln x_i - n \cdot \ln x_0}$$

Problem 3:

$$P(bus \mid late) = \frac{P(late, bus)}{P(late)} = \frac{P(late \mid bus) \cdot P(bus)}{P(late \mid bus) \cdot P(bus) + P(late \mid bike) \cdot P(bike)}$$

$$= \frac{0.1 \times 0.2}{0.1 \times 0.2 + 0.02 \times 0.8}$$

$$= \frac{5}{9}$$

Problem 4.1:

$$P_e = Pr\{\text{decide } H_0, H_1 \text{ true}\} + Pr[\text{decide } H_1, H_0 \text{ true}]$$

$$= P(H_1)P(H_0 \mid H_1) + P(H_0)P(H_1 \mid H_0)$$

$$= P(H_1)[1 - P(H_1 \mid H_1)] + P(H_0)P(H_1 \mid H_0)$$

$$= P(H_1)[1 - \int_{R_1} p(x \mid H_1)dx] + P(H_0)\int_{R_1} p(x \mid H_0)dx$$

$$= P(H_1) + \int_{R_1} [P(H_0)p(x \mid H_0) - P(H_1)p(x \mid H_1)]dx$$

classifier: $$P(H_1)p(x \mid H_1) \underset{H_0}{\overset{H_1}{\gtrless}} P(H_0)p(x \mid H_0)$$

$$\Downarrow$$

$$\frac{p(x \mid H_1)}{p(x \mid H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{p(H_0)}{P(H_1)}$$, which is equivalent to the MAP decision criterion

Problem 4.2:

$$\max_{0 \le i \le M-1} P(H_i)P(x \mid H_i)$$

$$\max_{0 \le i \le M-1} P(H_i \mid x) \Rightarrow \max \frac{P(H_i)P(x \mid H_i)}{P(x)}$$

$$= \max \frac{P(H_i) \cdot P(x \mid H_i)}{\sum_{i=0}^{M-1} P(H_i)p(x \mid H_i)} \Rightarrow \max_{0 \le i \le M-1} P(H_i)p(x \mid H_i)$$

Problem 4.3:

classifier:

$$\max_{0\le i\le M-1} P(H_i)\prod_{n=1}^{N} P(X_n|H_i)$$

We assume that, all <u>features</u> are conditionally independent from each other under hypothesis H_i.

Problem 5.1:

Test Image　　Ground Truth Mask　　Classification Result

Problem 5.2:

E:\Anaconda\python.exe

```
True_Positive Rate =  84.43 %
True_Negative Rate =  93.8 %
False_Positive Rate =  6.2 %
False_Negative Rate =  15.57 %
```

Attachment:

Problem 5 Code:

```
from PIL import Image
import numpy as np
import math

train_image = Image.open('family.jpg').convert('RGB')
train_mask = Image.open('family.png').convert('RGB')

width = train_image.size[0]
height = train_image.size[1]
#pixel numbers of skin and bg of train image
skin_count = 0
nonskin_count = 0
#save rg chromaticity space of train image
skin_rspace = []
nonskin_rspace = []
skin_gspace = []
nonskin_gspace = []

for x in range(width):
    for y in range(height):
        imageR, imageG, imageB = train_image.getpixel((x,y))
        maskR, maskG, maskB = train_mask.getpixel((x,y))
        #ground truth mask shows it is a skin pixel
        if (maskR > 250 and maskG > 250 and maskB > 250):
            skin_count += 1
            if (imageR + imageG + imageB == 0):
                skin_rspace.append(round(1/3, 2))
                skin_gspace.append(round(1/3, 2))
            else:
                skin_rspace.append(round(imageR/(imageR + imageG + imageB), 2))
                skin_gspace.append(round(imageG/(imageR + imageG + imageB), 2))
        #ground truth mask shows it is a bg pixel
        else:
            nonskin_count += 1
            if (imageR + imageG + imageB == 0):
                nonskin_rspace.append(round(1/3, 2))
                nonskin_gspace.append(round(1/3, 2))
            else:
                nonskin_rspace.append(round(imageR/(imageR + imageG + imageB), 2))
                nonskin_gspace.append(round(imageG/(imageR + imageG + imageB), 2))
```

```python
#calculate H0 = bg, H1 = skin
skin_probability = round(skin_count/(width*height), 2)
nonskin_probability = round(nonskin_count/(width*height), 2)
probability = round(nonskin_probability / skin_probability, 2)

#calculate properties of rg chromaticity space used for classification
skin_rmean = np.mean(skin_rspace)
skin_gmean = np.mean(skin_gspace)
skin_rvar = np.var(skin_rspace)
skin_gvar = np.var(skin_gspace)
nonskin_rmean = np.mean(nonskin_rspace)
nonskin_gmean = np.mean(nonskin_gspace)
nonskin_rvar = np.var(nonskin_rspace)
nonskin_gvar = np.var(nonskin_gspace)

#calculate kth pixel in the test image
test_image = Image.open('portrait.jpg').convert('RGB')
test_mask = Image.open('portrait.png').convert('RGB')
test_width = test_image.size[0]
test_height = test_image.size[1]
#used to print the detected binary mask
result = test_image.copy()
pixel = result.load()
#used to calculate TPR, TNR, FPR, FNR
correct_skin = 0
correct_bg = 0
wrong_skin = 0
wrong_bg = 0
true_skin = 0
true_bg = 0
#save rg chromaticity space of each pixel of test image
rspace = 0
gspace = 0
for x in range(test_width):
    for y in range(test_height):
        testR, testG, testB = test_image.getpixel((x,y))
        tmaskR, tmaskG, tmaskB = test_mask.getpixel((x,y))
        if (testR + testG + testB == 0):
            rspace = round(1/3, 2)
            gspace = round(1/3, 2)
        else:
            rspace = round(testR/(testR + testG + testB), 2)
            gspace = round(testG/(testR + testG + testB), 2)
```

```
        #calculate p(x|H1) and p(x|H0)
        skin_prediction = math.exp(-(rspace - skin_rmean)**2 / (2 * skin_rvar) - (gspace - skin_gmean)**2 / (2 * skin_gvar)) / (2 * np.pi * math.sqrt(skin_rvar) * math.sqrt(skin_gvar))
        nonskin_prediction = math.exp(-(rspace - nonskin_rmean)**2 / (2 * nonskin_rvar) - (gspace - nonskin_gmean)**2 / (2 * nonskin_gvar)) / (2 * np.pi * math.sqrt(nonskin_rvar) * math.sqrt(nonskin_gvar))
        prediction = round(skin_prediction / nonskin_prediction, 2)

        if (prediction >= probability):
            pixel[x, y] = (255, 255, 255)
            #not skin, but calculated as skin
            if (tmaskR <= 250 or tmaskG <= 250 or tmaskB <= 250):
                wrong_bg += 1
                true_bg += 1
            else:
                correct_skin += 1
                true_skin += 1
        else:
            pixel[x, y] = (0, 0, 0)
            if (tmaskR <= 250 or tmaskG <= 250 or tmaskB <= 250):
                correct_bg += 1
                true_bg += 1
            #not bg, but calculated as bg
            else:
                wrong_skin += 1
                true_skin += 1
#output the result
result.save('result.jpg')
true_positive = round(correct_skin/true_skin * 100, 2)
true_negative = round(correct_bg/true_bg * 100, 2)
false_positive = round(wrong_bg/true_bg * 100, 2)
false_negative = round(wrong_skin/true_skin * 100, 2)
print("True_Positive Rate = ", true_positive, "%\nTrue_Negative Rate = ", true_negative, "%\nFalse_Positive Rate = ", false_positive, "%\nFalse_Negative Rate = ", false_negative, "%")
```