# **COEN 240 Machine Learning**

# Homework #4

Name: Jinhao Wang ID: 4302178

## Problem1:

P(dotalo)
$$= (\frac{1}{3}\theta)^{2} \cdot (\frac{1}{3}\theta)^{3} \cdot (\frac{1}{3}(1-\theta))^{3} \cdot (\frac{1}{3}(1-\theta))^{2}$$

$$= 4 \cdot (\frac{1}{3}\theta)^{2} \cdot 8 \cdot (\frac{1-\theta}{3}\theta)^{2}$$

$$= \ln P(\text{dotal}\theta) = \ln 32 + 5 \cdot \ln (\frac{1}{3}\theta) + 5 \cdot \ln (\frac{1-\theta}{3}\theta)$$

$$= \frac{15}{10} + \frac{15}{1-\theta} = 0$$

$$= \frac{1}{10} + \frac{15}{1-\theta} = 0$$

## **Problem 2:**

$$P(\operatorname{doto}|\Theta) = \frac{n}{1!} \theta x_0^{\theta} x_1^{-\theta-1}$$

$$\ln P(\operatorname{doto}|\Theta) = \ln \left( \frac{n}{1!} \Theta \cdot x_0^{\theta} \cdot x_1^{-\theta-1} \right)$$

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

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

$$\frac{\partial \ln P(\operatorname{doto}|\Theta)}{\partial \theta} = \frac{n}{\Theta} + n \cdot \ln x_0 - \frac{n}{1!} \cdot \ln x_i = 0$$

$$\widehat{\Theta}_{mL} = \frac{n}{\frac{n}{1!} \ln x_i - n \cdot \ln x_0}$$

# **Problem 3:**

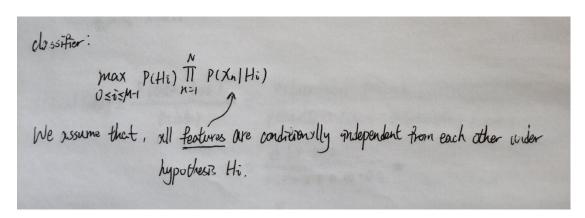
P(bus| late) = 
$$\frac{P(lxt_2 \cdot bus)}{P(lxt_2 \mid bus)} = \frac{P(lxt_2 \mid bus) \cdot P(bus)}{P(lxt_2 \mid bus) \cdot P(bus) + P(lote| bike) \cdot P(bike)}$$
$$= \frac{0.1 \times 0.2}{0.1 \times 0.2 + 0.02 \times 0.8}$$
$$= \frac{5}{4}$$

# Problem 4.1:

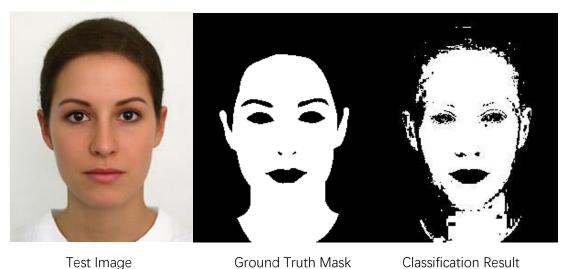
#### Problem 4.2:

max 
$$P(H_i)P(x|H_i)$$
 $0 \le i \le M-1$ 
 $0 \le i \le M-1$ 
 $P(H_i)P(M|H_i)$ 
 $P(X)$ 
 $P(X)$ 

# Problem 4.3:

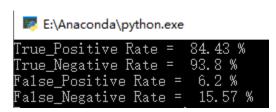


#### Problem 5.1:



**Ground Truth Mask** Test Image

# Problem 5.2:



#### 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 \text{ and } maskG > 250 \text{ 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))
```

```
\#calculate HO = 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 t_{ask} <= 250 or t_{ask} <= 250 or t_{ask} <= 250:
                wrong bg += 1
                true\_bg += 1
            else:
                correct_skin += 1
                true skin += 1
        else:
            pixel[x, y] = (0, 0, 0)
            if t = 250 or t = 250 or t = 250 or t = 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,
"%")
```