รหัสวิชา: 010113337 ชื่อวิชา: ปฏิบัติการระบบโทรคมนาคม

ชื่อ: _____รหัสนักศึกษา: _____

ภาคการศึกษาที่: ___ ปีการศึกษา: _____

วันที่และเวลาที่ทำการทดลอง: _____

อาจารย์ผู้สอน: ผู้ช่วยศาสตราจารย์ ดร. ไกรสร ไชยซาววงค์

วัตถุประสงค์

1. เพื่อให้นักศึกษารู้จักการประมวลภาพและเทคนิคคอมพิวเตอร์วิทัศน์บนข้อมูลภาพดิจิทัล
2. เพื่อให้นักศึกษารู้จักการใช้คำสั่งที่เกี่ยวข้องใน Python
3. เพื่อให้นักศึกษาเรียนรู้ความสัมพันธ์ระหว่างการแจกแจงแบบ Gaussian กับ histogram ของรูปภาพ
4. เพื่อให้นักศึกษารู้จักการใช้เทคนิค parameter estimation กับ Gaussian mixture model
5. เพื่อให้นักศึกษารู้จักการใช้ Gaussian mixture model มาใช้ในการทำ segmentation

ทฤษฎีเบื้องต้น

Gaussian distribution คือการแจกแจงเชิงสถิติทั่วไปชนิดหนึ่ง ใช้บ่งบอกถึงคุณลักษณะที่เป็นจำนวนจริงที่มีการแจกแจงที่มีรูปแบบกำหนดคือแบบ Gaussian ทั้งนี้ Gaussian distribution เป็นการแจกแจงที่ใช้ค่าเฉลี่ย (mean) และค่าความแปรปรวน (variance) มานิยาม

ในการประมวลภาพ histogram คือกราฟที่บอกถึงการแจกแจงความถี่ของความเข้ม โดยแกนนอนจะเป็นค่าความเข้มในภาพดิจิทัล ซึ่งอาจจะแบ่งเป็น 8-bit (0-255 level) หรือ 12-bit (0-4095 level) เป็นต้น ส่วนแกนตั้งเป็นความถี่ของความเข้มนั้นๆ โดยบอกเป็นจำนวนพิกเซล (pixel) ที่ถูกนับภายในรูปภาพดิจิทัลนั้น

จากลักษณะของ Gaussian distribution ที่ได้กล่าวมาข้างต้น ประกอบกับการใช้ histogram มาอธิบายคุณลักษณะทางการแจกแจงความถี่ของความเข้มในภาพดิจิทัล หากสังเกต histogram ของรูปภาพใดๆ ที่ยกมาเป็นตัวอย่าง โดยทั่วไปจะไม่สามารถเทียบเสมือน (modelling) การแจกแจงที่ปรากฏให้เห็นได้ด้วยการแจกแจงแบบ Gaussian เพียงหนึ่งการแจกแจงเดียว จึงเกิดแนวความคิดเทียบเสมือนการแจกแจงความเข้มใน histogram ให้เกิดจากการรวมกันของหลาย Gaussian distribution โดยที่แต่ละ Gaussian distribution ที่นำมารวมกันมี ค่า mean และ variance ที่แตกต่างกัน การเทียบเสมือน histogram ของภาพนั้น เรียกว่าการเทียบเสมือนโดยการใช้ Gaussian mixture (หรือ Gaussian mixture model)

ให้ Gaussian distribution: $p_k(x_i|\theta_k) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{1}{2\sigma_k^2}(x_i-\mu_k)^2}$

โดยที่

θ_k คือ parameters vector (mean μ_k, variance σ_k^2) ของ Gaussian distribution ตัวที่ k

x_i คือ random variable ใดๆ

Gaussian mixture model คือการนำ p_k แต่ละตัวมาทำการถ่วงน้ำหนัก ด้วยตัวถ่วงน้ำหนัก (weight proportion) ω_k ใดๆ ของแต่ละ Gaussian distribution มารวมกันทั้งหมด C ตัว

$$\sum_{k=1}^{c} \omega_k p_k(x_i|\theta_k)$$

โดยที่ผลรวมของการถ่วงน้ำหนักที่จะนำมาใช้บน Gaussian distribution คือ

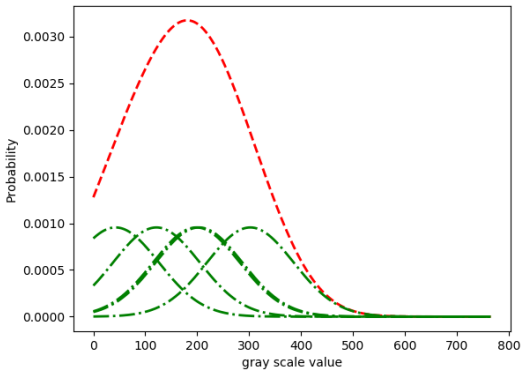
$$\sum_{k=1}^{c} \omega_k = 1, \omega_k > 0$$

parameter estimation คือ การหาค่าของ parameters ที่เราไม่รู้โดยการประมาณ ในกรณีของการเทียบเสมือน histogram คือการประมาณค่า parameters ของ Gaussian distribution แต่ละตัว ให้รวมกันแล้วได้ผลรวมที่คล้ายคลึงกับ histogram ของรูปภาพเป็นที่สุด ทั้งนี้เราสามารถพิจารณา histogram ให้เป็น Probability density function ได้

ตัวอย่าง

$\omega_1 = \omega_2 = \omega_3 = \omega_4 = 0.25$

จากตัวอย่างการ กำหนดให้ตัวถ่วงน้ำหนัก เท่ากันและมี Gaussian distribution 4 ตัว โดยมีความความแปรปรวนเท่ากันและมีค่าเฉลี่ย ที่ต่างกัน จะเห็นได้ว่าผลรวมของ ω_k เท่ากับ 1 ดังนั้น หากมีจำนวน Gaussian distribution มากขึ้น ก็จะมีสัดส่วนที่แตกต่างออกไปขึ้นอยู่กับกำหนดค่าเริ่มต้น แต่จะมีค่าผลรวมของตัวถ่วงน้ำหนักเท่ากับ 1 เท่านั้น



วิธีการทำ parameter estimation ที่จะนำเสนอวิธีหนึ่งคือ likelihood function คือการนำผลลัพธ์ที่ได้จากการประมาณไปหา parameters

สมการ likelihood function:

$$L(\emptyset) = \prod_{i=1}^{n} p(x_i|\emptyset)$$

$$= \prod_{i=1}^{n} \sum_{k=1}^{c} \omega_k p_k(x_i|\theta_k)$$

สมการ log-likelihood function:

$$logL(\emptyset) = log \prod_{i=1}^{n} \sum_{k=1}^{c} \omega_k p_k(x_i|\theta_k)$$

$$= \sum_{i=1}^{n} log \sum_{k=1}^{c} \omega_k p_k(x_i|\theta_k)$$

$$= \sum_{i=1}^{n} log \sum_{k=1}^{c} \omega_k \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{1}{2\sigma_k^2}(x_i-\mu_k)^2}$$

ขั้นตอนการทำ parameter estimation ตามวิธีของ **Dempster, A.P., Laird, N.M., and Rubin, D.B.** (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society*, Series B. 39 (1): 1–38.

1. กำหนด parameters เริ่มต้นสำหรับทุกๆ การแจกแจงแต่ละตัว (mean [μ_k], variance [σ_k^2], weight proportion [ω_k]) โดยที่ k คือจำนวน Gaussian distribution ที่ถูกกำหนดขึ้นเช่นเดียวกัน
2. คำนวณ posterior probabilities จากสมการ ด้านล่างด้วย parameters เริ่มต้นที่กำหนด ซึ่งเรียกขั้นตอนนี้ว่า expectation step

$$\hat{h}_{ki} = \frac{\omega_k p_k(x_i|\theta_k)}{\sum_{s=1}^{c} \omega_s p_s(x_i|\theta_s)}$$

3. พารามิเตอร์ตัวใหม่เกิดจากการใช้ posterior probabilities ที่ได้มาจาก ขั้นตอนที่ 2 มาประมาณค่าเฉลี่ย ค่าความแปรปรวน ค่าตัวถ่วงน้ำหนักตัวใหม่ โดยที่จำนวน Gaussian ที่นำมาเพื่อประมาณการแจกแจง histogram ของภาพยังคงเดิม เรียกขั้นตอนนี้ว่า maximization step

$$\hat{\mu}_k = \frac{\sum_{i=1}^{n} \hat{h}_{ki} x_i}{\sum_{i=1}^{n} \hat{h}_{ki}} \qquad \hat{\sigma}_k^2 = \frac{\sum_{i=1}^{n} \hat{h}_{ki}(x_i-\hat{\mu}_k)^2}{\sum_{i=1}^{n} \hat{h}_{ki}}$$

$$\hat{\omega}_k = \frac{1}{n}\sum_{i=1}^{n} \hat{h}_{ki}$$

4. ใช้สมการ likelihood function, log-likelihood function กับ parameters ที่ได้ออกมาเพื่อเปรียบเทียบความคล้ายกันหรือไม่ โดยจะเปรียบเทียบกับกราฟ histogram ที่ได้จากภาพ
5. หากค่า parameters ที่ได้ทำให้ histogram จากภาพกับการคำนวณใกล้เคียงกันถือว่าใช้ได้ หากไม่ใกล้เคียงกันจะกลับไปขั้นตอนที่ 2 ใหม่ จนกว่าจะได้ค่าที่ใกล้เคียงกัน

<u>หมายเหตุ</u> ถึงแม้ว่าจะได้กราฟที่ใกล้เคียงกับ histogram ของรูปภาพนั้น ก็ไม่ได้หมายความว่า กราฟนั้นจะสามารถแสดงถึงหรือแทนภาพต้นฉบับได้เพราะกราฟไม่ได้บ่งบอกถึงการ mapping pixel ของภาพ มันแค่บอกถึงจำนวนความเข้มที่ค่าต่าง ๆ เท่านั้น

ขั้นตอนการทดลอง

1. อิมพอร์ตไลบรารี่ PIL
   อิมพอร์ตไลบรารี่ skimage
   อิมพอร์ตไลบรารี่ matplotlib.pyplot as plt
   อิมพอร์ตไลบรารี่ cv2
   อิมพอร์ตไลบรารี่ numpy as np
   แล้วใส่ชื่อรูปภาพ 'MyFacePic.jpg' ลงไปในไฟล์ Parameter_Estimation.py
   พิจารณา histogram ของภาพ แล้วประมาณว่าจะต้องใช้จำนวนของ Gaussian distribution เท่าใด ใส่จำนวนที่ประมาณลงไปในไฟล์ Parameter_Estimation.py โดยใส่ค่าที่ตัวแปร C และ ปรับจำนวนของ command lines ให้สอดคล้องกับจำนวน Gaussian distribution ทุกที่ในไฟล์ Parameter_Estimation.py (ดู comment lines ประกอบ)
   หมายเหตุ หากใส่ค่า C ที่มีค่ามาก (ใช้ Gaussian distribution มากเกินความจำเป็น) อาจทำให้เกิดการประมวลผลที่ล่าช้า
   ทำการทดลองซ้ำ โดยปรับค่า epsilon ในไฟล์ Parameter_Estimation.py ให้มีค่าระหว่าง (1.0e-4, 1.0e-3) ที่แตกต่างกันสามค่า แล้วเลือกค่าที่เหมาะสม
   ทำการทดลองซ้ำ โดยปรับเปลี่ยนค่า C ให้เหมาะสมกับค่า epsilon ที่ได้เลือกไว้ แล้วนำเสนอผลการทดลองพร้อมอภิปรายและให้เหตุผลข้อสรุป

2. ทำการคัดเฉพาะส่วนที่เป็นใบหน้าของภาพ MyFacePic.jpg โดยใช้ คำสั่ง plt.imshow(arr, cmap='gray', vmin = ค่าความเข้มต่ำสุด, vmax = ค่าความเข้มสูงสุด) โดยที่พิจารณาค่า vmin ที่เป็นความเข้มต่ำสุดที่ต้องการให้แสดง และค่า vmax ที่เป็นความเข้มสูงสุดที่ต้องการให้แสดง จากขอบเขตของการแจกแจงแบบ Gaussian ที่เหมาะสมสำหรับการคัดแยก
   a. ใบหน้าและส่วนที่เป็นลำตัวพร้อมกัน
   b. เฉพาะส่วนที่เป็นใบหน้า
   พร้อมทั้งมาร์คตำแหน่งดังกล่าวลงใน histogram ทั้งของรูปจริงและ Gaussian mixture model

3. ทำการทดลองซ้ำ โดยเปลี่ยนไปใช้ไฟล์รูป ClassificationGS.jpg แล้วทำการคัดเลือกส่วนที่เป็น นก จากรูป ClassificationGS.jpg มานำเสนอ พร้อมทั้งมาร์คตำแหน่งดังกล่าวลงใน histogram ทั้งสอง นำเสนอผลการทดลองพร้อมอภิปราย

4. ทำการทดลองซ้ำ โดยเปลี่ยนไปใช้ไฟล์รูป CTScan.jpg แล้วทำการคัดเลือกส่วนที่เป็น
   a. ปอด
   b. กระดูก
   ของรูป CTScan.jpg โดยที่คัดและนำเสนอทีละส่วน พร้อมทั้งมาร์คตำแหน่งดังกล่าวลงใน histogram ทั้งสอง

ภาคผนวก

```
#Gaussian Normal Density

def gaussian_norm_density(x,mu,sig):
    return np.exp(-np.power(x-mu, 2.)/(2*np.power(sig,2.)))/
    (sig*np.sqrt(2*np.pi))

d = max(data)
xl = np.arange(0, d*3,0.1)

p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
# p(C)_est=p_est[c]*gaussian_norm_density(xl,mu_est[c],sigma_est[c]);

plt.figure(3)

plt.plot(xl,p1_est+p2_est+p3_est+p4_est+p5_est+…, 'r--',linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);
# plt.plot(xl,p_est,'g-.',linewidth=2.0);

plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.show()

clas = [ ]
sumclas = [ ]
while np.any(difference >= epsilon) and (counter<25000):
    for j in range(0,c):
        clas.insert(j,p_est[j]*gaussian_norm_density(data,mu_est[j],sigma_est[j]))

        sumclas=clas[0]+clas[1]+clas[2]+clas[3]+clas[4]+clas[c]+…

    for j in range(0, c):
           clas[j] = clas[j]/sumclas
    mu_est_old = mu_est
    sigma_est_old = sigma_est
    p_est_old = p_est
    mu_est = [ ]
    sigma_est = [ ]
    p_est = [ ]

    for j in range(0, c):
          mu_est.insert(j, sum((clas[j]) * data) / sum(clas[j]))
          sigma_est.insert(j, np.sqrt(sum((clas[j])*np.power((data-
        mu_est[j]),2))/sum(clas[j])))
```

```
p_est.insert(j, mean(clas[j]))
    difference
=sum(abs(np.subtract(mu_est_old,mu_est)))+sum(abs(np.subtract(sigma_est_old,sigma_e
st))) \

    +sum(abs(np.subtract(p_est_old,p_est)))

    counter = counter + 800

    xl = np.arange(0, d, 0.1)
```

```
p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
# p(C)_est=p_est[c]*gaussian_norm_density(xl,mu_est[c],sigma_est[c]);
```

```
plt.figure(4)
```

```
plt.plot(xl,p1_est+p2_est+p3_est+p4_est+p5_est+…, 'r--',linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
# plt.plot(xl,p_est,'g-.',linewidth=2.0);
```

```
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.show()
```

```
p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
# p(C)_est=p_est[c]*gaussian_norm_density(xl,mu_est[c],sigma_est[c]);
```

```
plt.figure(5)
```

```
plt.plot(xl,p1_est+p2_est+p3_est+p4_est+p5_est+…, 'r--',linewidth=2.0)
plt.plot(xl, p1_est, 'b-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);
# plt.plot(xl,p_est,'g-.',linewidth=2.0);
```

```
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.show()
```