**K-Means Clustering ใน OpenCV**

เป้าหมาย

* เรียนรู้การใช้**ฟังก์ชัน cv2.kmeans ()**ใน OpenCV สำหรับการจัดกลุ่มข้อมูล

ทำความเข้าใจพารามิเตอร์

ป้อนพารามิเตอร์

1. **samples** : ควรเป็นประเภทข้อมูลnp.float32และแต่ละคุณลักษณะควรวางไว้ในคอลัมน์เดียว
2. **nclusters(K)**: จำนวนของกลุ่มที่ต้องการในตอนท้าย
3. **criteria**: เป็นเกณฑ์การสิ้นสุดการเลิกจ้าง เมื่อเกณฑ์นี้มีความพอใจการทำซ้ำอัลกอริทึมจะหยุดลง จริงๆแล้วควรเป็น tuple 3 พารามิเตอร์ พวกเขาเป็น( ประเภท, max\_iter, epsilon ) :
   * 3.a - ประเภทของเกณฑ์การยกเลิก: มี 3 ธงดังนี้:

cv2.TERM\_CRITERIA\_EPS - หยุดการทำซ้ำขั้นตอนวิธีการถ้าความถูกต้องระบุepsilonจะมาถึงcv2.TERM\_CRITERIA\_MAX\_ITER - หยุดขั้นตอนวิธีการหลังจากการระบุจำนวนซ้ำที่max\_itercv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER - หยุดการทำซ้ำเมื่อมีการปฏิบัติตามเงื่อนไขข้างต้น

* + 3.b - max\_iter - จำนวนเต็มระบุจำนวนสูงสุดของการทำซ้ำ
  + 3.c - epsilon - ความถูกต้องที่ต้องการ

1. **attempts** : ตั้งค่าสถานะเพื่อระบุจำนวนครั้งที่อัลกอริทึมจะถูกเรียกใช้โดยใช้ห้องแล็บแรก ๆ อัลกอริทึมจะส่งคืนป้ายกำกับที่ให้ผลผลิตได้ดีที่สุด ความเป็นปึกแผ่นนี้จะกลับเป็นเอาท์พุท
2. **flags**  : **flags** นี้ถูกใช้เพื่อระบุว่าศูนย์เริ่มต้นถูกนำมาใช้อย่างไร ปกติสองธงที่ใช้สำหรับการนี้: cv2.KMEANS\_PP\_CENTERSและcv2.KMEANS\_RANDOM\_CENTERS

พารามิเตอร์ขาออก

1. **compactness** : เป็นผลรวมของระยะห่างจากแต่ละจุดไปยังศูนย์ที่ตรงกัน
2. **labels** : นี่เป็นอาร์เรย์ของป้ายกำกับ (เหมือนกับ 'โค้ด' ในบทความก่อนหน้า) โดยที่แต่ละส่วนมีเครื่องหมาย '0', '1' .....
3. **centers** : นี่เป็นศูนย์รวมของกลุ่ม

ตอนนี้เราจะดูวิธีการใช้อัลกอริธึม K-Means ด้วยสามตัวอย่าง

1. ข้อมูลที่มีเพียงหนึ่งคุณลักษณะ

พิจารณาว่าคุณมีชุดข้อมูลที่มีคุณลักษณะเพียงอย่างเดียวนั่นคือแบบมิติเดียว ตัวอย่างเช่นเราสามารถใช้ปัญหาเสื้อยืดของเราซึ่งคุณใช้ความสูงเพียงอย่างเดียวของคนในการตัดสินใจขนาดเสื้อยืด

ดังนั้นเราจึงเริ่มต้นด้วยการสร้างข้อมูลและวางแผนไว้ใน Matplotlib

**import** **numpy** **as** **np**

**import** **cv2**

**from** **matplotlib** **import** pyplot **as** plt

x = np.random.randint(25,100,25)

y = np.random.randint(175,255,25)

z = np.hstack((x,y))

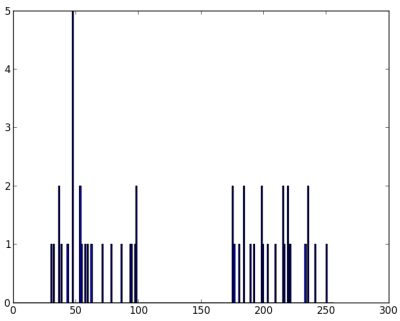
z = z.reshape((50,1))

z = np.float32(z)

plt.hist(z,256,[0,256]),plt.show()

ดังนั้นเราจึงมี 'z' ซึ่งเป็นอาร์เรย์ของขนาด 50 และมีค่าตั้งแต่ 0 ถึง 255 ฉันได้เปลี่ยนรูปแบบ 'z' เป็นเวกเตอร์ของคอลัมน์แล้ว จะมีประโยชน์มากกว่าเมื่อมีคุณสมบัติมากกว่าหนึ่งอย่าง แล้วฉันทำข้อมูลของ np.float32 ชนิด

เราได้ภาพต่อไปนี้:



ตอนนี้เราใช้ฟังก์ชัน KMeans ก่อนที่เราจำเป็นต้องระบุเกณฑ์ เกณฑ์ของฉันเป็นเช่นที่เมื่อใดก็ตามที่ 10 ซ้ำของอัลกอริทึมถูกเรียกใช้หรือมีความถูกต้องของepsilon = 1.0ให้หยุดอัลกอริทึมและตอบกลับ

*# Define criteria = ( type, max\_iter = 10 , epsilon = 1.0 )*

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)

*# Set flags (Just to avoid line break in the code)*

flags = cv2.KMEANS\_RANDOM\_CENTERS

*# Apply KMeans*

compactness,labels,centers = cv2.kmeans(z,2,None,criteria,10,flags)

นี่ทำให้เรามีความหนาแน่นฉลากและศูนย์ ในกรณีนี้ฉันมีศูนย์เป็น 60 และ 207 ป้ายกำกับจะมีขนาดเท่ากับข้อมูลทดสอบที่แต่ละข้อมูลจะมีข้อความว่า '0', '1', '2' ฯลฯ ขึ้นอยู่กับเซนทรอยด์ของพวกเขา ตอนนี้เราแบ่งข้อมูลไปยังกลุ่มต่างๆตามป้ายกำกับ

A = z[labels==0]

B = z[labels==1]

ตอนนี้เราวาด A ในสีแดงและ B ในสีน้ำเงินและ centroids ของพวกเขาในสีเหลือง

*# Now plot 'A' in red, 'B' in blue, 'centers' in yellow*

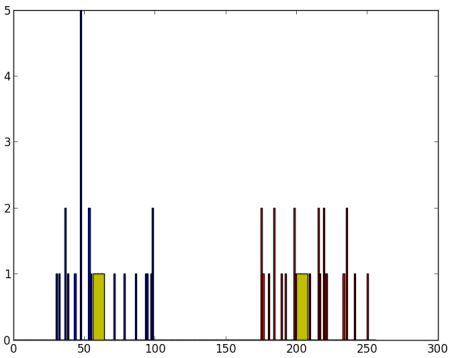
plt.hist(A,256,[0,256],color = 'r')

plt.hist(B,256,[0,256],color = 'b')

plt.hist(centers,32,[0,256],color = 'y')

plt.show()

ด้านล่างคือผลลัพธ์ที่เราได้รับ:

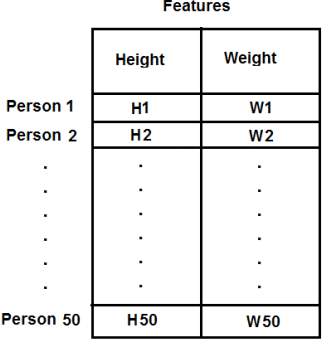


2. ข้อมูลที่มีคุณสมบัติหลายอย่าง

ในตัวอย่างก่อนหน้านี้เราใช้ความสูงเพียงอย่างเดียวสำหรับปัญหาเสื้อยืด ที่นี่เราจะใช้ทั้งความสูงและน้ำหนักเช่นสองคุณสมบัติ

โปรดจำไว้ว่าในกรณีก่อนหน้านี้เราได้ทำข้อมูลของเราไปยังเวคเตอร์คอลัมน์เดียว แต่ละคุณลักษณะจะจัดเรียงไว้ในคอลัมน์แต่ละแถวจะตรงกับตัวอย่างการทดสอบอินพุท

ตัวอย่างเช่นในกรณีนี้เราจะตั้งค่าข้อมูลทดสอบขนาด 50x2 ซึ่งมีความสูงและน้ำหนักของ 50 คน คอลัมน์แรกสอดคล้องกับความสูงของทั้ง 50 คนและคอลัมน์ที่สองตรงกับน้ำหนักของพวกเขา แถวแรกประกอบด้วยสององค์ประกอบแรกคือความสูงของคนแรกและตัวที่สองน้ำหนักของตัวเอง ในทำนองเดียวกันแถวที่เหลือจะตรงกับความสูงและน้ำหนักของคนอื่น ๆ ตรวจสอบภาพด้านล่าง:



ตอนนี้ฉันตรงไปยังรหัส:

**import** **numpy** **as** **np**

**import** **cv2**

**from** **matplotlib** **import** pyplot **as** plt

X = np.random.randint(25,50,(25,2))

Y = np.random.randint(60,85,(25,2))

Z = np.vstack((X,Y))

*# convert to np.float32*

Z = np.float32(Z)

*# define criteria and apply kmeans()*

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)

ret,label,center=cv2.kmeans(Z,2,None,criteria,10,cv2.KMEANS\_RANDOM\_CENTERS)

*# Now separate the data, Note the flatten()*

A = Z[label.ravel()==0]

B = Z[label.ravel()==1]

*# Plot the data*

plt.scatter(A[:,0],A[:,1])

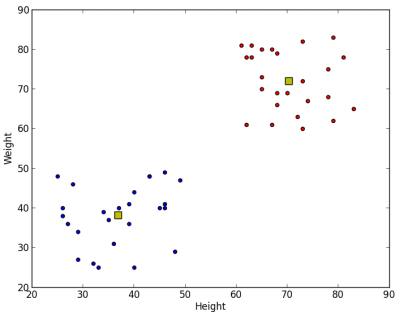
plt.scatter(B[:,0],B[:,1],c = 'r')

plt.scatter(center[:,0],center[:,1],s = 80,c = 'y', marker = 's')

plt.xlabel('Height'),plt.ylabel('Weight')

plt.show()

ด้านล่างเป็นผลลัพธ์ที่เราได้รับ:



3. สี Quantization

Color Quantization คือกระบวนการลดจำนวนสีในภาพ เหตุผลหนึ่งที่ต้องทำก็คือการลดความจำ บางครั้งอุปกรณ์บางอย่างอาจมีข้อ จำกัด เพื่อให้สามารถผลิตได้เฉพาะสีที่ จำกัด ในกรณีเหล่านี้ quantization สีจะดำเนินการ ที่นี่เราใช้การจัดกลุ่มแบบ k-means สำหรับการหาค่าสี

ไม่มีอะไรใหม่ที่จะอธิบายที่นี่ มีคุณสมบัติ 3 อย่างเช่น R, G, B ดังนั้นเราจำเป็นต้องปรับรูปใหม่ให้เป็นขนาด Mx3 (M คือจำนวนพิกเซลในภาพ) และหลังจากการจัดกลุ่มเราใช้ค่าเซนทรอยด์ (เป็น R, G, B) กับพิกเซลทั้งหมดเช่นภาพที่ได้จะมีจำนวนสีที่ระบุ และอีกครั้งเราจำเป็นต้องปรับรูปร่างใหม่กลับไปเป็นรูปร่างของภาพต้นฉบับ ด้านล่างนี้เป็นรหัส:

**import** **numpy** **as** **np**

**import** **cv2**

img = cv2.imread('home.jpg')

Z = img.reshape((-1,3))

*# convert to np.float32*

Z = np.float32(Z)

*# define criteria, number of clusters(K) and apply kmeans()*

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)

K = 8

ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS\_RANDOM\_CENTERS)

*# Now convert back into uint8, and make original image*

center = np.uint8(center)

res = center[label.flatten()]

res2 = res.reshape((img.shape))

cv2.imshow('res2',res2)

cv2.waitKey(0)

cv2.destroyAllWindows()

CV2 imshow ( 'res2' , res2 )

CV2 waitKey ( 0 ) CV2 destroyAllWindows ()

ดูผลลัพธ์ด้านล่างสำหรับ K = 8:

