**OCR ของข้อมูลที่เขียนด้วยมือโดยใช้ SVM**

เป้าหมาย

ในบทนี้

* เราจะทบทวนข้อมูลที่เขียนด้วยมือ OCR แต่ด้วย SVM แทน kNN

OCR ของตัวเลขที่เขียนด้วยมือ

ใน kNN เราใช้ความเข้มของพิกเซลเป็นเวกเตอร์จุดสนใจโดยตรง เวลานี้เราจะใช้[ฮิสโตแกรมของ Oriented Gradients](http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients) (HOG) เป็นเวกเตอร์คุณลักษณะ

ที่นี่ก่อนที่จะหา HOG เราจะสร้างภาพโดยใช้ช่วงเวลาสั่งซื้อที่สอง ดังนั้นเราจึงกำหนดฟังก์ชันdeskew ()ซึ่งใช้รูปแบบตัวเลขและเป็นลายลักษณ์อักษร ด้านล่างนี้เป็นฟังก์ชัน deskew ():

**def** deskew(img):

m = cv2.moments(img)

**if** abs(m['mu02']) < 1e-2:

**return** img.copy()

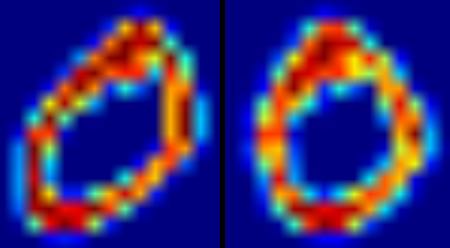
skew = m['mu11']/m['mu02']

M = np.float32([[1, skew, -0.5\*SZ\*skew], [0, 1, 0]])

img = cv2.warpAffine(img,M,(SZ, SZ),flags=affine\_flags)

**return** img

ด้านล่างภาพแสดงด้านบนฟังก์ชัน deskew ใช้กับภาพของศูนย์ รูปภาพด้านซ้ายเป็นภาพต้นฉบับและภาพขวาเป็นภาพแบบสไลด์



ต่อไปเราจะต้องหา HOG Descriptor ของแต่ละเซลล์ จากนั้นเราจะพบอนุพันธ์ของโซเบอร์elในแต่ละเซลล์ในทิศทาง X และ Y จากนั้นหาขนาดและทิศทางการไล่ระดับสีของแต่ละพิกเซล การไล่ระดับสีนี้เป็น quantized ถึง 16 ค่าจำนวนเต็ม แบ่งภาพนี้เป็นสี่สี่เหลี่ยมจัตุรัสย่อย สำหรับแต่ละสี่เหลี่ยมย่อยให้คำนวณฮิสโทแกรมของทิศทาง (16 ถัง) ที่มีน้ำหนักตามขนาด ดังนั้นแต่ละสี่เหลี่ยมย่อยจะมีเวกเตอร์ที่มี 16 ค่า สี่เวกเตอร์ดังกล่าว (สี่สี่เหลี่ยมจัตุรัสย่อย) ร่วมกันทำให้เราเป็นเวกเตอร์คุณลักษณะที่มีค่า 64 ค่า นี่คือเวกเตอร์คุณลักษณะที่เราใช้ในการฝึกอบรมข้อมูลของเรา

**def** hog(img):

gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)

gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)

mag, ang = cv2.cartToPolar(gx, gy)

*# quantizing binvalues in (0...16)*

bins = np.int32(bin\_n\*ang/(2\*np.pi))

*# Divide to 4 sub-squares*

bin\_cells = bins[:10,:10], bins[10:,:10], bins[:10,10:], bins[10:,10:]

mag\_cells = mag[:10,:10], mag[10:,:10], mag[:10,10:], mag[10:,10:]

hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) **for** b, m **in** zip(bin\_cells, mag\_cells)]

hist = np.hstack(hists)

**return** hist

สุดท้ายเช่นเดียวกับในกรณีก่อนหน้านี้เราจะเริ่มต้นด้วยการแยกชุดข้อมูลขนาดใหญ่ของเราลงในแต่ละเซลล์ สำหรับตัวเลขทุกๆ 250 เซลล์จะถูกสงวนไว้สำหรับข้อมูลการฝึกอบรมและเหลือ 250 ข้อมูลสำหรับการทดสอบเท่านั้น รหัสเต็มจะได้รับด้านล่าง:

**import** **cv2**

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

SZ=20

bin\_n = 16 *# Number of bins*

svm\_params = dict( kernel\_type = cv2.SVM\_LINEAR,

svm\_type = cv2.SVM\_C\_SVC,

C=2.67, gamma=5.383 )

affine\_flags = cv2.WARP\_INVERSE\_MAP|cv2.INTER\_LINEAR

**def** deskew(img):

m = cv2.moments(img)

**if** abs(m['mu02']) < 1e-2:

**return** img.copy()

skew = m['mu11']/m['mu02']

M = np.float32([[1, skew, -0.5\*SZ\*skew], [0, 1, 0]])

img = cv2.warpAffine(img,M,(SZ, SZ),flags=affine\_flags)

**return** img

**def** hog(img):

gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)

gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)

mag, ang = cv2.cartToPolar(gx, gy)

bins = np.int32(bin\_n\*ang/(2\*np.pi)) *# quantizing binvalues in (0...16)*

bin\_cells = bins[:10,:10], bins[10:,:10], bins[:10,10:], bins[10:,10:]

mag\_cells = mag[:10,:10], mag[10:,:10], mag[:10,10:], mag[10:,10:]

hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) **for** b, m **in** zip(bin\_cells, mag\_cells)]

hist = np.hstack(hists) *# hist is a 64 bit vector*

**return** hist

img = cv2.imread('digits.png',0)

cells = [np.hsplit(row,100) **for** row **in** np.vsplit(img,50)]

*# First half is trainData, remaining is testData*

train\_cells = [ i[:50] **for** i **in** cells ]

test\_cells = [ i[50:] **for** i **in** cells]

*###### Now training ########################*

deskewed = [map(deskew,row) **for** row **in** train\_cells]

hogdata = [map(hog,row) **for** row **in** deskewed]

trainData = np.float32(hogdata).reshape(-1,64)

responses = np.float32(np.repeat(np.arange(10),250)[:,np.newaxis])

svm = cv2.SVM()

svm.train(trainData,responses, params=svm\_params)

svm.save('svm\_data.dat')

*###### Now testing ########################*

deskewed = [map(deskew,row) **for** row **in** test\_cells]

hogdata = [map(hog,row) **for** row **in** deskewed]

testData = np.float32(hogdata).reshape(-1,bin\_n\*4)

result = svm.predict\_all(testData)

*####### Check Accuracy ########################*

mask = result==responses

correct = np.count\_nonzero(mask)

**print** correct\*100.0/result.size

เทคนิคพิเศษนี้ให้ความถูกต้องเกือบ 94% คุณสามารถลองค่าต่างๆสำหรับพารามิเตอร์ต่างๆของ SVM เพื่อตรวจสอบว่ามีความแม่นยำสูงกว่าหรือไม่ หรือคุณสามารถอ่านเอกสารด้านเทคนิคในพื้นที่นี้และลองใช้งานได้