Projection을 이용한 문자 Segmentation과 숫자인식

(시각개론 과제1)

201124506 이한결

# 프로젝트 목적

* 본 프로젝트는 projection을 이용하여 주어진 그림에서 숫자영역을 추출해 내고,

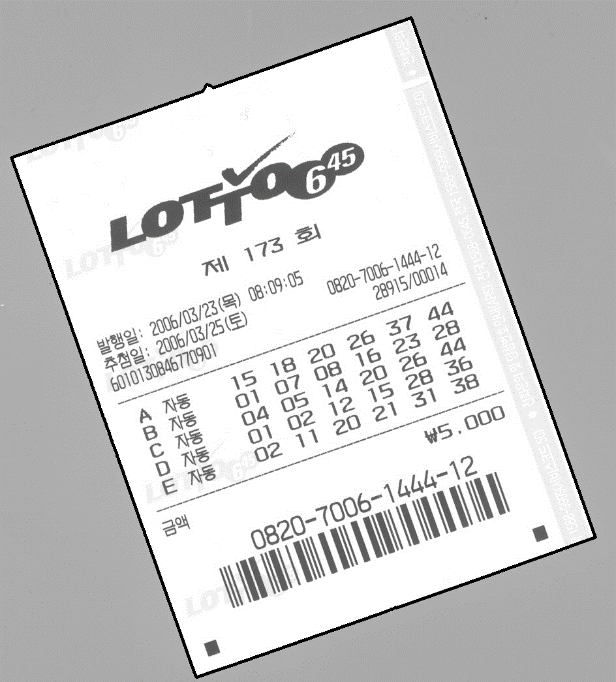
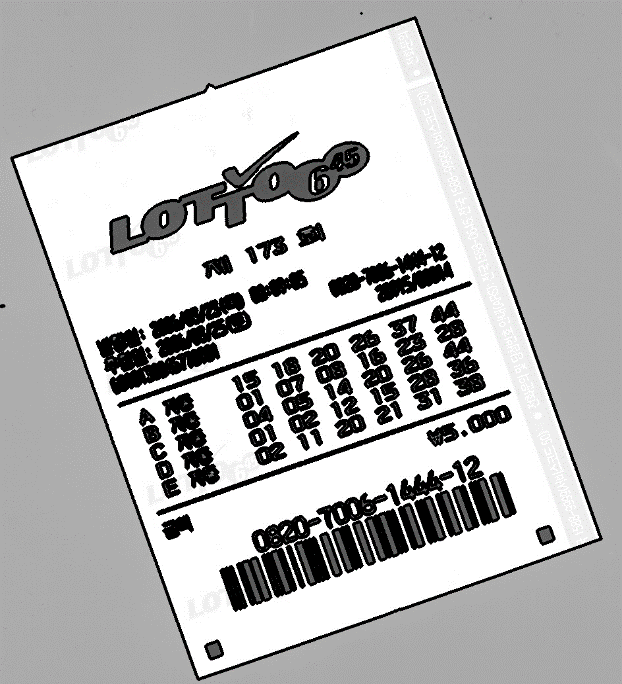
Perceptron 이론을 적용하여 숫자인식을 하는 것이다.

# 시스템환경

|  |
| --- |
| Windows 8  Python 2.7  openCV 2.4.13  numpy 1.11.1 |

# 구현 결과

## 3-1. Image Rotation



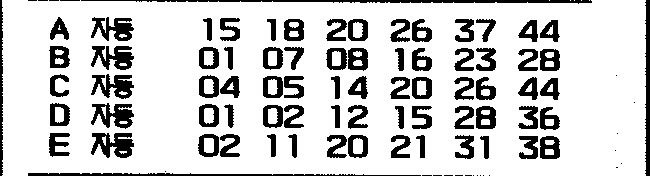
『그림 1』 『그림 2』



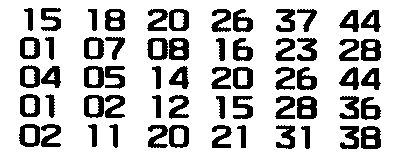
『그림 3』

|  |
| --- |
| img = cv2.imread('lotto.jpg',0)  rows,cols = img.shape  //주어진 로또 사진을 읽어온다.  \_,thresh=cv2.threshold(img,200,255,cv2.THRESH\_BINARY)  contours, hierarchy = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)  //사진을 이진화 시킨 뒤 Contour함수를 적용하여 『그림 1』처럼 사진에 주어진 모든 윤곽선을 잡아낸다.  maxArea = 0  cntIndex = 0  for i in range(len(contours)):  cnt = contours[i]  area = cv2.contourArea(cnt)  if (maxArea < area):  maxArea = area  cntIndex = i  cnt = contours[cntIndex]  rect = cv2.minAreaRect(cnt)  //모든 윤곽선 중 가장 넓은 영역만을 추출해 내면 『그림 2』와 같이 배경화면을 제외한 로또 사진만을 얻을 수 있다.  이를 이용하여 사진영역만을 추출한 뒤 minAreaRect()함수를 이용하여 사진의 기울어진 정도를 찾아낸다. minAreaRect()함수는 반환값으로 (중앙위치값, (넓이,높이),기울기)를 리스트 형식으로 반환한다. 따라서 아래와 같이 rect[2]로 접근하면 기울기값에 접근 가능하다.  M = cv2.getRotationMatrix2D((cols/2,rows/2),rect[2],1)  cx,cy = rect[0]  width,height = rect[1]  cx = int(cx)  cy = int(cy)  width = int(width)  height = int(height)  dst = cv2.warpAffine(img,M,(cols,rows))  \_,dst=cv2.threshold(dst,230,255,cv2.THRESH\_BINARY)  dst = dst[cy-height/2+10:cy+height/2-10,cx-width/2:cx+width/2]  //찾아낸 기울기 값을 이용하여 주어진 기울어진 사진을 『그림 3』과 같이 세운다. |

## 3-2. ROI(Region of Interest) 추출



『그림 4』



『그림 5』



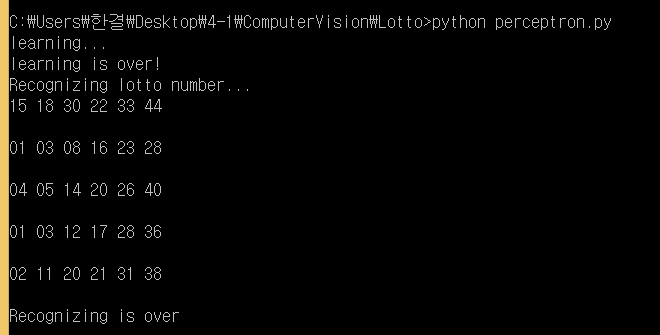
『그림 6』

|  |
| --- |
| def horizontalProjection(image):  maxNum = 0  line = {}  rows,cols = image.shape  for i in range(rows):  tmp = 0  for j in range(cols):  if image[i,j]==0:  tmp = tmp + 1  line[i] = tmp  return line  def verticalProjection(image):  maxNum = 0  line = {}  rows,cols = image.shape  for i in range(cols):  tmp = 0  for j in range(rows):  if image[j,i]==0:  tmp = tmp + 1  line[i] = tmp  return line  line = horizontalProjection(dst)  line = sorted(line.items(),key=operator.itemgetter(1))  low = line[len(line)-1][0]  high = line[len(line)-2][0]  if low < high:  dst = dst[low:high,]  else:  dst = dst[low:high,]  dst = dst[10:168,179:586]  ‘’’  『그림 4』처럼 숫자가 적힌 부분만을 추출하기 위해 HorizontalProjection을 수행하여 가장 값이 큰 두 부분을 찾아내어 『그림 3』 에서 잘라낸다.  추가로 로또 용지의 크기는 동일하므로 『그림 5』와 같이 잘라내어도 모든 로또 용지에 동일하게 적용 시킬 수 있어서 잘라내었다.  ‘’’  ver = verticalProjection(dst)  hor = horizontalProjection(dst)  rows,cols = dst.shape  verLine = []  horLine = []  for i in range(1,len(ver)-2):  if (ver[i-1]<=5 and ver[i] > 5) or (ver[i+1]<=5 and ver[i]> 5):  #cv2.line(dst,(i,0),(i,rows),(0,0,255),1)  verLine.append(i)  #print (i,low),(i,high)  for i in range(1,len(hor)-2):  if (hor[i-1]<=5 and hor[i] > 5) or (hor[i+1]<=5 and hor[i]> 5):  #cv2.line(dst,(0,i),(cols,i),(0,0,255),1)  horLine.append(i)  #print (i,low),(i,high)  cv2.imwrite("test.png",dst)  numOfLine = 0  for i in range(0,len(horLine),2):  numOfLine += 1  y=0  for j in range(0,len(verLine),2):  y+=1  cv2.imwrite(str(numOfLine)+str(y)+'.png',dst[horLine[i]:horLine[i+1],verLine[j]:verLine[j+1]])  ‘’’  Vertical, Horizontal Projection을 이용하여 『그림 6』과 같이 숫자영역을 분리 시킨 후 숫자 하나하나를 저장하였다.  ‘’’ |

## 3-3. Perceptron Learning

|  |
| --- |
| import numpy as np  import cv2  from lotto import numOfLine  #내가 설정한 Desired Output  number = np.array([[-1,-1,-1,-1],  [-1,-1,-1,1],  [-1,-1,1,-1],  [-1,-1,1,1],  [-1,1,-1,-1],  [-1,1,-1,1],  [-1,1,1,-1],  [-1,1,1,1],  [1,-1,-1,-1],  [1,-1,-1,1]],dtype=int)  eta = np.random.random()  #주어진 로또 사진에서 0-8까지 한 개의 사진으로만 학습시키지 않고 인식률을 더 높이기 위해 각각 여러 개를 써서 학습시켰다.  numData = {0:['./number/0-1.png','./number/0-2.png','./number/0-3.png','./number/0-4.png',  './number/0-5.png'],  1:['./number/1-1.png','./number/1-2.png','./number/1-3.png','./number/1-4.png'  ,'./number/1-5.png'],  2:['./number/2-1.png','./number/2-2.png','./number/2-3.png','./number/2-4.png'  ,'./number/2-5.png'],  3:['./number/3-1.png','./number/3-2.png','./number/3-3.png','./number/3-4.png'  ,'./number/3-5.png'],  4:['./number/4-1.png','./number/4-2.png','./number/4-3.png','./number/4-4.png'  ,'./number/4-5.png'],  5:['./number/5-1.png','./number/5-2.png','./number/5-3.png'],  6:['./number/6-1.png','./number/6-2.png','./number/6-3.png','./number/6-4.png'],  7:['./number/7-1.png','./number/7-2.png'],  8:['./number/8-1.png','./number/8-2.png','./number/8-3.png','./number/8-4.png',  './number/8-5.png']}  W = np.random.rand(4,20\*25)\*10-3  class NN:  def \_\_init\_\_(self,num):  self.eta = eta  self.D = number[num]  self.isCorrect = [False,False,False,False]  self.num = num  img = [cv2.imread(numData[num][i],0) for i in range(len(numData[num]))]  img = [cv2.resize(img[i],(20,25)) for i in range(len(img))]  self.X = []  for index in range(len(img)):  rows, cols = img[index].shape  self.X.append(np.empty([rows,cols]))  for i in range(rows):  for j in range(cols):  if img[index][i,j]==255:  self.X[index][i,j]=0  else:  self.X[index][i,j]=1  self.X[index] = self.X[index].reshape((1,rows\*cols))  def changeWeight(self,index,itx):  W[index] = W[index]+self.eta\*(self.D[index]-self.Y[0,index])\*self.X[itx]  def numXLearning(self):  for itx in range(len(self.X)):  self.Y = np.dot(self.X[itx],W.T)  for i in range(4):  if self.Y[0,i] >= 0:  self.Y[0,i] = 1  else:  self.Y[0,i] = -1  for i in range(4):  if self.Y[0,i]!=self.D[i]:  self.changeWeight(i,itx)  else:  self.isCorrect[i] = True  numOf = [NN(i) for i in range(9) ]  print "learning..."  for iteration in range(1000000): #학습 횟수  for index in range(9):  numOf[index].numXLearning()  '''learning is over'''  '''test start'''  print "learning is over!"  print "Recognizing lotto number..."  for i in range(1,numOfLine+1):  testImg = [[cv2.imread(str(i)+str(j)+'.png',0) for j in range(1,13,2)],  [cv2.imread(str(i)+str(j)+'.png',0) for j in range(2,13,2)]]  for it in range(6):  string = ""  for lottoNumIndex in range(2):  maxNum = 0  testImg[lottoNumIndex][it] = cv2.resize(testImg[lottoNumIndex][it],(20,25))  rows, cols = testImg[lottoNumIndex][it].shape  X = np.empty([rows,cols])  for i in range(rows):  for j in range(cols):  if testImg[lottoNumIndex][it][i,j]==255:  X[i,j]=0  else:  X[i,j]=1  X = X.reshape((1,rows\*cols))  result = -1  Y = np.dot(X,W.T)  #print Y  for i in range(4):  if Y[0,i]>=0:  Y[0,i] = 1  else:  Y[0,i] = -1  isBreak = False  for num in range(9):  count = 0  for i in range(4):  if number[num][i] == Y[0,i]:  count = count+1  if maxNum <= count:  result = num  maxNum = count  if isBreak==True:  break    string = string + str(result)  print string,  print '\n'  print "Recognizing is over" |

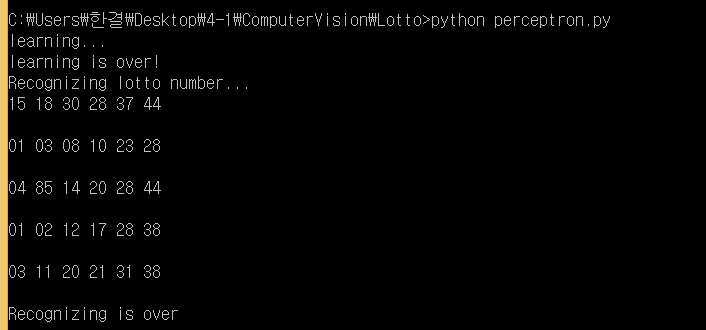
500번 학습 결과



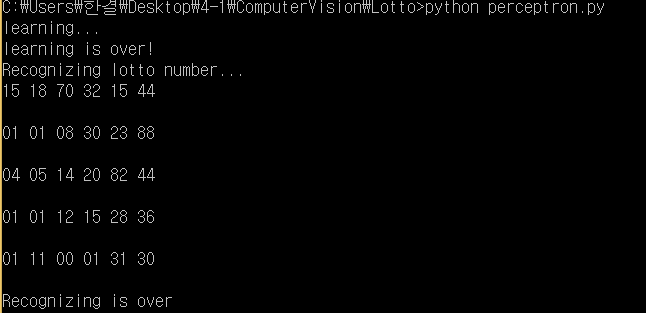
50000번 학습 결과



100000번 학습 결과



300000번 학습 결과



# 결론

본 프로젝트덕에 openCV를 사용하여 이미지 프로세싱에 관해 많은 것을 배울 수 있었다. 또한, 가장 중요한 Projection의 원리를 정확하게 이해하였고 이를 잘 적용시켜 숫자를 추출해 낼 수 있었다. Perceptron에 대해 이해도는 높아졌지만 학습량의 차이인지, 데이터의 부족인지 숫자인식률은 생각보다 좋지 못하였다. 그래서 하나의 숫자 사진으로만 학습을 시켰던 것을 동일한 숫자에 대해 여러 사진으로 학습을 시켰더니 인식률이 훨씬 좋아졌다. 하지만 학습횟수를 늘린다고 해서 무조건 인식률이 반드시 좋아지지는 않았다.

로또 사진에서 숫자를 인식 할 때, 위 사진에서는 총 5라인의 로또 번호가 적혀있는데 numOfLine 변수에 총 라인 수가 저장되어 있어서 다른 로또 사진을 이용하여 5개의 라인보다 적게 있어도 인식이 가능하도록 하였다. 또한 사진이 주어지면 자동으로 기울기, 숫자영역을 추출할 수 있다.

추가로, 주어진 로또 사진에서 9가 없어서 0-8에 대한 숫자만 학습을 시켜 진행하였다.