컴퓨터응용확률 과제2

-Skin detection using Bayesian Classifier -

12161570 박성연

구현 환경 : visual code 1.29.1

구현 언어 : Python

Part A. Preliminaries

1. (5pts) Cross Validation

   § Summarize the k-fold cross-validation method for evaluating a classifier (less than one page)

   § Important: Use your own explanation. Absolutely DO NOT copy any paragraph from any source (paper, internet, blog, etc.) This will result in 0 point for your entire homework

   Cross Validation은 교차검증으로 설계한 모델에 다른 데이터들이 잘 들어맞는지에 대한 성능을 확인하는 방법이다. K-hold 교차검증은 데이터셋을 k로 나눠서 k개 중 한 개의 데이터셋을 testing 데이터로 사용하고 나머지 (k-1)/k 만큼의 데이터셋을 training 데이터로 사용한다. Training 데이터 셋을 분석하여 모델을 만들고 그 모델에 testing 데이터 셋을 대입하여 그 모델의 정확성에 대해 평가한다. 이런 시행을 k로 나눴을 때 k개의 각각의 데이터를 모두 testing 데이터로 사용하는 것이다.

   예를 들어서, 5-fold cross validation은 데이터가 100개 있을 때, 100개를 5개로 나눠서 첫 번째부터 스무 번째까지는 testing data로 사용하고 나머지 80개를 사용해서 모델을 생성한다. 그 모델에 testing data를 대입하여 정확도를 확인한다. 또 스물 한 번째부터 마흔 번째 까지를 testing data로 하고, 첫 번째부터 스무 번째 데이터와 마흔 한 번째부터 나머지를 사용해서 model을 생성한다. 그 model에 testing data를 대입하여 정확도를 확인한다. 이 과정을 다섯 등분한 데이터들을 하나씩, 모두 다섯번을 시행한다.

Part B. Skin detection using text data

1. Write a code to classify skin pixels or non-skin pixels based on the Bayesian decision rule; by finding a pmf(probability mass function) of likelihood and defining a prior. Evaluate your classifier by computing precision and recall. Use 5-fold cross validation for evaluation. (10pts)

   a. Use only R channel values to classify as skin or non-skin

   (결과)

   Precision
   Recall
   Accuracy
   F1

```
(base) C:₩Users₩hepbu₩rgb>python partB_a.py
0.2977716338302344
0.668312150408819
0.5385199240986718
0.41196617009976616
```

(1) "Skin_NotSkin.txt"를 읽기 형식으로 열고, lines에 저장을 한다. Skin일 때와 NotSkin일 때를 차례대로 저장해놨기 때문에 shuffle함수를 이용해서 무작위로 섞는다. 5-fold cross validation을 하기 위해서 전체 개수를 5로 나누고 그 수를 idx에 저장한다.

```
lines=file. readlines()
npa = np.array(lines)
np.random.shuffle(npa)
lines = list(npa)
idx = int(len(lines)/5)
```

(2) Red 색상이 0부터 255까지 있기 때문에, Skin일 때와 NotSkin일 때를 나눠서 0부터 255의 각각의 개수를 저장하기 위한 배열 red1과 red2를 선언한다.

```
red1 = np.zeros(256)
red2 = np.zeros(256)
```

(3) Testing data set은 idx*k번째부터 idx*(k+1)까지기 때문에 그 외에 처음부터 idx*k까지와 idx*(k+1)부터 끝까지를 training data set으로 사용한다. File을 받은 lines를 한 줄 씩 읽으면서 tab마다 끊는다. 첫 번째는 red이기 때문에, 1일 때 red1 배열의 '그 색상 값 번째' 배열에 값을 하나 더해준다. 똑같이 NotSkin일 때, red2에 똑같이 값을 1씩 더해준다.

```
for line in lines[:idx*k]:
    sline=line.split('\t')
    #print(sline)
    if sline[3][0]=='1':
        red1[int(sline[0])]+=1.
        count1 +=1
    else :
        red2[int(sline[0])]+=1.
        count2 +=1
for line in lines[idx*(k+1):]:
    sline=line.split('\t')
    #print(sline)
    if sline[3][0]=='1':
        red1[int(sline[0])]+=1.
        count1 +=1
    else :
        red2[int(sline[0])]+=1.
        count2 +=1
```

(4) 피부일 때 red가 어떤 값일 확률과 피부가 아닐 때 red가 어떤 값일 확률을 저장할 reds1과 reds2 배열을 똑같이 255개 만든다. 또, red가 어떤 값일 때, 1일 때 red일 확률과 2일 때 red일 확률을 비교해서, 그 값이 피부인지 아닌지를 결정해서 저장할 배열인 redclass를 만든다.

```
reds1 = np.zeros(256)
reds2 = np.zeros(256)
```
```
redclass = np.zeros(256)
```

(5) Red1[i]가 red2[i]보다 클 경우는 red의 색상 값이 i일 경우 피부라는 것이기 때문에 redclass[i]에 1을 저장하고, red2[i]가 더 큰 경우에는 피부가 아니라는 것이기 때문에 redclass[i]에 2를 저장한다.

(6) Idx*k 부터 idx*(k+1)까지는 testing data set으로 사용한다. Red가 어떤 값을 가지면서 실제 피부일 때, redclass의 값이 피부라고 되어 있다면 tp 의 개수를 하나 더해주고 redclass의 값이 피부가 아니라고 되어 있다면 fn 의 개수를 하나 더해준다. 또한 red가 어떤 값을 가지면서 실제로 피부가 아닐 때, redclass의 값이 피부라고 되어 있다면 fp 에 1을 더해주고, 피부가 아니라고 되어 있다면 tn 에 1을 더해준다.

```
for line in lines[idx*k:idx*(k+1)]:
    sline=line.split('\t')
    if int(sline[3])==1:
        if redclass[int(sline[0])]==1 :
            tp+=1
        if redclass[int(sline[0])]==2 :
            fn+=1
    else :
        if redclass[int(sline[0])]==1 :
            fp+=1
        if redclass[int(sline[0])]==2 :
            tn+=1
```

(7) Precision, recall, accuracy 그리고 F1을 5-fold이기 때문에 총 다섯 개를 구하고, 다섯 개의 값을 평균을 내서 출력한다.

```
pre[k] = tp/(tp+fp)
re[k] = tp/(tp+fn)
ac[k] = (tp+tn)/(tp+fp+fn+tn)
f1[k]=2*pre[k]*re[k]/(pre[k]+re[k])
```

b. Use all channel(R/G/B) values to classify as skin or non-skin

(결과)

Precision
Recall
Accuracy
F1

```
(base) C:₩Users₩hepbu₩rgb>python partB_b.py
0.7023209277706307
0.6887562050423246
0.7569198751300729
0.6954687798518767
```

(1) 모든 알고리즘과 코드는 위의 A번과 같다. 변수는 red와 blue 그리고 green으로 개수는 세배가 된다.

```
red1 = np.zeros(256)
red2 = np.zeros(256)
green1 = np.zeros(256)
green2 = np.zeros(256)
blue1 = np.zeros(256)
blue2 = np.zeros(256)
```

(2) A와 같이 lines를 한 줄 씩 읽고 tab 단위로 split했을 때, 첫번째는 red, 두번째는 green, 세번째는 blue이다.

```
sline=line.split('\t')
if sline[3][0]=='1':
    red1[int(sline[0])]+=1.
    green1[int(sline[1])]+=1.
    blue1[int(sline[2])]+=1.
    count1 +=1
```

(3) 베이지안 분류기 공식에 따라 피부일 때 red가 어떤 값을 가질 확률, green이 어떤 값을 가질 확률, blue가 어떤 값을 가질 확률 그리고 전체 중 피부일 확률을 모두 곱해서 result1을 구하고 반대로 피부가 아닐 때로 해서 result2를 구한다.

```
result1= reds1[int(sline[0])]*greens1[int(sline[1])]*blues1[int(sline[2])]*(count1/(count1+count2))
result2= reds2[int(sline[0])]*greens2[int(sline[1])]*blues2[int(sline[2])]*(count2/(count1+count2))
```

(4) Result1이 result2보다 크면 피부라는 의미로 int형인 classify변수에 1을 저장하고 result2가 더 크면 변수 classify에 2를 저장한다.

(5) A와 같이 실제 데이터 값과 training을 통해 얻은 위의 classify값을 비교해서 tp, fn, fp, tn의 개수를 센다.

(6) Precision, recall, accuracy, f1의 값을 fold 개수만큼 모두 구하고 평균을 내어 출력한다.

코드 파일첨부

2. Write a code to classify skin or non-skin based on a Gaussian model and Evaluate your classifier by computing precision and recall. Use 5-fold cross validation for evaluation. (20pts)

a. Use R channel values to classify as skin or non-skin

(결과)

Precision

Recall

Accuracy

F1

```
(base) C:₩Users₩hepbu₩rgb>python partB_2_a.py
0.16738362786614738
0.19562083903463218
0.5210585378792516
0.17893432856107439
```

(1) Skin_NotSkin data가 가우시안 분포를 따른다고 가정을 하고, 가우시안 분포에 필요한 parameter인 평균과 분산을 구한다. 이 때 피부일 때와 아닐 때를 나눠서 red1_av, red2_av, red1_var, red2_var을 구한다.

(2) 피부일 때 또는 아닐 때 평균과 표준편차 그리고 testing data set에 있는 각 줄의 red값을 parameter로 하는 normpdf 함수를 만든다. 이 함수는 평균이 mean이고 표준편차가 st인

```
def normpdf(size, mean, st):
    var = float(st)**2
    pi = 3.1415926
    denom = (2*pi*var)**.5
    num = math.exp(-(float(size)-float(mean))**2/(2*var))
    return num/denom
```

가우시안 분포를 따르는 data set에서 size의 포인트에서의 확률을 구하는 함수이다. 이는 아래의 식을 그대로 이용하여서 풀었다.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

또 다른 방법으로는 색상의 값이 1 단위로 증가하기 때문에, size값에서 0.5를 더하고 뺀 범위에서 가우시안 분포의 모형을 적분하는 방법이 있었다.

(3) 피부일 때 또는 아닐 때의 평균과 분산 그리고 testing data set의 데이터를 이용해서, 피부일 때의 가우시안 분포에서의 확률과 피부가 아닐 때의 가우시안 분포에서의 확률을 각각 구한다.

```
if sline[3][0]=='1':
    red1 = normpdf(int(sline[0][0]),red1_av,red1_var**0.5)
elif sline[3][0] =='2' :
    red2 = normpdf(int(sline[0][0]),red2_av,red2_var**0.5)
```

(4) 두 확률을 비교하고 red1이 red2보다 크면 피부라고 결정하고, red2가 더 크면 피부가 아니라고 결정한다. 그 후 원래 testing data set에 있는 피부인지 아닌 지에 대한 데이터와 비교하고 tp, fn, fp, tn의 개수를 센다.

(5) 마찬가지로 precision, recall, accuracy, f1을 각 fold마다 구한 뒤 평균을 내서 출력한다.

b. Use all channel(R/G/B) values to classify as skin or non-skin

(결과)

Precision

Recall

Accuracy

F1

```
(base) C:₩Users₩hepbu₩rgb>python partB_2_b.py
0.1647400956064579
0.2591706847596234 6
0.4807818652955459
0.2014134464626379 8
```

(1) A번과 마찬가지로 red value에 대해 피부일 때 아닐 때, green value에 대해 피부일 때 아닐 때 그리고 blue value에 대해 피부일 때 아닐 때, 총 여섯 종류의 가우시안 분포를 생각하고 training data set을 이용하여 각 여섯 가지 가우시안 분포의 평균과 분산을 구한다.

(2) 피부일 때 또는 아닐 때 RGB 각각의 평균과 분산 그리고 testing data set의 red , green,

```
sline=line.split('\t')
if sline[3][0]=='1':
    red1 = normpdf(int(sline[0][0]),red1_av,red1_var**0.5)
    green1 = normpdf(int(sline[1][0]),green1_av,green1_var**0.5)
    blue1 = normpdf(int(sline[2][0]),blue1_av,blue1_var**0.5)
elif sline[3][0] =='2' :
    red2 = normpdf(int(sline[0][0]),red2_av,red2_var**0.5)
    green2 = normpdf(int(sline[1][0]),green2_av,green2_var**0.5)
    blue2 = normpdf(int(sline[2][0]),blue2_av,blue2_var**0.5)
```

blue 값을 parameter로 넣어서 각각의 가우시안 함수에서의 함수를 구한다.

(3) Red, blue, green의 값은 모두 독립으로 각 가우시안 분포에서 나온 확률들을 곱하고 총 피부일 또는 피부가 아닐 확률을 곱해준 뒤 그 확률을 비교한다. 마찬가지로 1일 때의 확률일 2일 때의 확률보다 크면 피부라고 결정하고, 아니면 피부가 아니라고 결정한다.

```
if (red1*green1*blue1*(count1/(count1+count2))) >= (red2*green2*blue2*(count2/(count1+count2))) :
    classify = 1
else :
    classify = 2
```

(4) 이 때 결정 값과 실제 데이터에서의 값을 비교해서 tp, fn, fp, tn의 개수를 세고 precision, recall, accuracy, f1을 모든 fold마다 구하고, 그 값들의 평균을 내서 출력한다.

코드 파일첨부

Part C. Skin detection using real data

1. Parse the annotation file to recover the face areas. Construct a training dataset in the same format of the Skin_NonSkin.txt in Part B. Note: You must carefully read the data format in README.txt (10pts)

(1) Anaconda prompt 에서 명령어 "conda install -c conda-forge opencv"를 사용해서 opencv를 설치한다.

(2) Cv2를 import 시켜준다.

```
import cv2
```

(3) 데이터를 적을 output.txt를 만들고 쓰기 모드로 파일을 연다.

```
f = open("output.txt", "w")
```

(4) FDDB ellipseList 를 열고 그 파일을 imgfile 변수로 받는다.

```
for k in range(1,11) :

    file = open("FDDB-fold-"+'%02d'%k +"-ellipseList.txt","r"
```

(5) File을 읽어왔을 때 '₩n'이 포함되어 있기 때문에 한 글자를 줄이고, '/'로 된 것을 '₩'로 바꿔서 경로를 표시할 수 있게 해준 뒤, '.jpg'를 붙여서 파일을 열수 있게 한다. 그 후 cv2.imread함수를 사용해서 txt에 있는 파일을 연다.

```
imgfile01 = imgfile01[:-1]
imgfile01 = imgfile01.replace("/","\\")+".jpg"
img = cv2.imread(imgfile01)
```

(6) 파일 이름 다음으로 나오는 타원형의 개수를 num으로 받고 num 만큼 타원을 그리는 cv2.ellipse함수를 이용한다. 이때 ellipse 함수는 파일이름, 중심좌표, 축의 길이, 각도, 색상을 parameter로 받는다. 이는 FDDB ellipseList 파일에 축의 길이, 각도, 중심좌표 순서로 되어 있으므로, 그 값을 split 함수로 잘라서 순서대로 ellipse함수에 넣어준다. 이때, ellipse 함수는 parameter로 모두 int형을 받기 때문에 int형으로 변환을 해준다.

```
num = int(file.readline())
```

```
draw = cv2.ellipse(img, (int(float(datas[3])), int(float(datas[4]))),
        (int(float(datas[1])),int(float(datas[0]))), float(datas[2]),0,360, (255,0,0), 1)
```

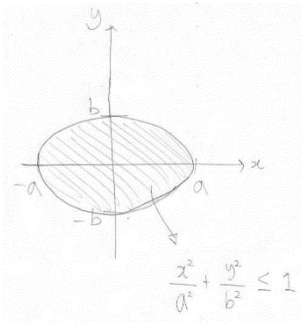
(7) 이미지에 그렸던 타원에 대한 축의 길이와 중심좌표에 대한 정보를 타원의 개수만큼 배열에 저장을 하고 이를 이용해서 그 타원에 있는지 없는지에 대해 판별을 한다.

```
    width = int(img.shape[1])
    height = int(img.shape[0])
    center_x.append(datas[3])
    center_y.append(datas[4])
    major.append(datas[0])
    minor.append(datas[1])
```

(8) 이 함수의 parameter는 해당 이미지의 가로와 세로 픽셀 수, 타원의 축의 길이 그리고 중심 좌표이다. 타원의 모든 픽셀에 대해 그 점이 해당 이미지 내의 모든 타원 내에 들어 있는지 확인하는 함수이며, 타원 내에 있으면 1을 반환하고 아니면 2를 반환한다.

```
def func(i,j,num,minor,major,center_x,center_y):
    for l in range(0,num):
        if ((i-float(center_x[l]))**2)/(float(minor[l])**2)+((j-float(center_y[l]))**2)/(float(major[l])**2) <=1 :
            return 1
    return 2
```

이는 고등학교 때 배우는 타원에 대한 공식을 이용해서 만든 함수이다.

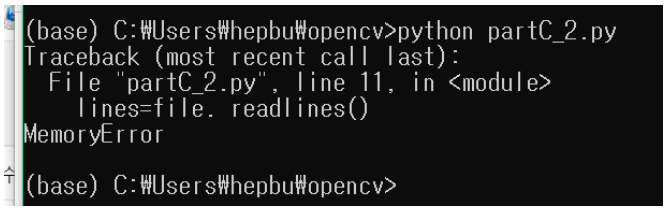


(9) 반환한 값을 detection이라는 변소에 저장을 하고, output.txt에 그 픽셀의 RGB와 detection을 저장한다. 이때 각 픽셀에 대한 RGB 값은 item이라는 함수를 사용하고 parameter로는 픽셀의 위치 I,j와 2는 red, 1은 green, 0은 blue를 의미한다.

```
for i in range(0,height) :
    for j in range(0,width) :
        detection = func(i,j,num,minor,major,center_x,center_y)
        f.write(str(img.item(i,j,2))+'\t'+str(img.item(i,j,1))+
            '\t'+str(img.item(i,j,0))+'\t' + str(detection) +'\n')
```

2. Repeat 2-b in the part B. Note: Choose the proper prior if needed. Select and visualize five good and five bad classification results. (20pts)

1번에서 추출한 output.txt을 Part B의 2-b의 data set으로 사용하여 프로그램을 돌렸다.



Part D. Discuss about your approach and analyses. (20pts)

중간고사가 모두 끝나고 11월 7일 수요일부터 과제를 시작했다.

Part A는 cross validation에 대해서 tistory 블로그들과 다른 백과사전 사이트를 보면서 PartB에 필요한 cross validation이 무엇인지 알아내고 공부했다. Tp, fn, fp, tn은 수업시간에 배웠던 걸 다시 상기시켰다. 인터넷을 찾아보니 precision과 recall 말고도 accuracy와 f1을 사용하는데, accuracy는 말 그대로 정확도이고 f1은 precision과 recall에 대한 조화평균으로 이 알고리즘이 좋은 알고리즘인가에 대한 평가 척도로 많이 사용되고 있다고 한다. 따라서 과제 지침에 있는 precision과 recall만 출력하지 않고 accuracy와 f1도 같이 출력했다.

Part B는 우선 part C의 open CV를 python으로 더 쉽게 사용할 수 있을 것 같아서 part B도 python으로 구현하기 시작했다. 베이지안 분류기를 만들 때에는 4장을 강의노트를 참고했다. P(Skin|Red, Blue, Green)과 P(NotSkin|Red, Blue, Green)는 P(red, green, blue | Skin(or NotSkin))/P(d)를 통해 구할 수 있었다. 수업 중에 P(d)는 보통 생략을 한다고 해서 과감하게 생략을 했다. 이는 P(red|Skin(or NotSkin))*P(green|Skin(or NotSkin))*P(blue|Skin(or NotSkin))*P(Skin(or NotSkin))을 이용해서 구했다. 이 때 Skin일 때 확률과 NotSkin일 때의 확률을 비교하고 더 큰 쪽으로 결정을 하도록 코드를 구현했다. Dataset은 피부일 때와 피부가 아닐 때로 나뉘어져서 있었기 때문에 무작위로 섞어야 할 필요가 있었고 따라서 shuffle 함수를 사용했다. 5-fold cross validation을 하기 위해 전체 data set을 5로 나누어서 training set과 testing set의 범위를 k를 이용해서 정했다. 중간고사 1번문제를 봤을 때는 대체 무엇을 하는 문제인지 조차 몰랐는데, 이 과제를 중간고사 전에 시작을 했더라면 중간고사 1번문제를 정말 풀 수 있었을 것 같았다.

PartB의 2번인 가우시안에 대해 구현을 할 때에는, 주어진 Skin_NotSkin.txt 데이터가 가우시안 분포를 따른다고 가정을 하고, training set으로 가우시안 분포에 필요한 parameter인 mean값과 variance값을 구했다. 그러고 나서 testing set이 그 가우시안 분포를 따른다고 가정을 하고 testing set의 어떤 값이 그 가우시안 분포에서 확률을 몇이라고 나타내는지 구했다. 처음에는 색상이 1단위로 올라가기 때문에 0.5씩 더하고 빼서 그 구간에 대해 적분을 하는 코드를 짜려고 했는데, 적분은 어떻게 구현을 하는지 모르겠어서 가우시안 함수를 따로 함수로 구현했다.

PartC는 open cv를 여름방학 때 과 내의 프로그래밍 공모전에서 잠시 사용해봤기 때문에 조금 수월했다. Imread, ellipse, item 함수에 대해서 알았고 고등학교 때의 타원공식을 오랜만에 접했다. 타원 공식을 이용해서 풀었지만, 같이 과제를 했던 다른 친구는 이미지를 새로 만들어서 거기에 타원을 채워서 그리고 그 타원을 그린 이미지와 실제 이미지를 비교하면서 피부인 지 피부가 아닌 지를 판별하게 했다. 속도는 내가 더 빠를 거라고 자부했다.

하지만 output.txt를 모두 뽑고, 가우시안 함수로 그 output.txt를 cross validaiton을 하려고 했는데, memory error가 났다. 이 건 내 힘으로 해결할 수 없을 것 같았다..

이번 과제는 한달을 잡고 한 것 때문인지, 문제해결기법 과제를 매주 밤새면서 풀면서 느꼈던 성취감보다

더 큰 성취감과 보람을 느꼈다.