Gaussian Mixture Model

Code

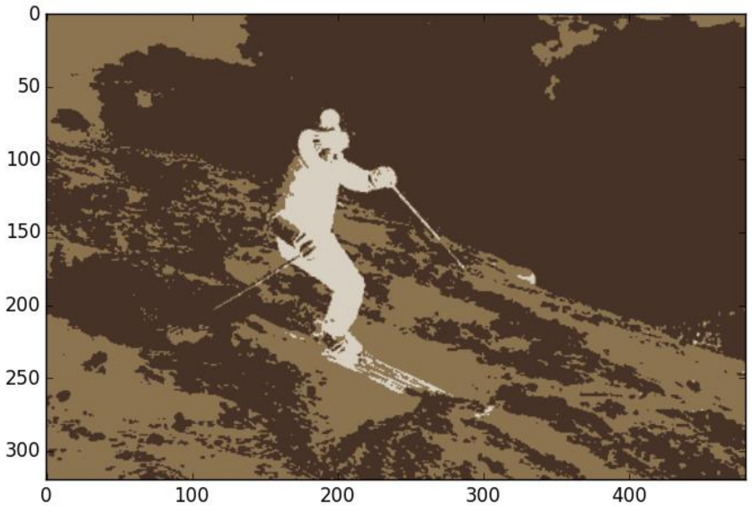
```
import glob, os
from PIL import Image
import numpy as np
import PIL
from matplotlib import pyplot

# reading image into matrix
img = Image.open('ski_image.jpg').resize((480,320), Image.ANTIALIAS)
pixels = np.asarray(((img.getdata())))

#total no of pixels
N = 153600

# initializing means, variances and weights
feat = pixels
v = [0,1,2]
val = 250
mean = [np.array([120, 120, 120]), np.array([12, 12, 12]), np.array([180, 180, 180])]
var = [val*np.identity(3), val*np.identity(3), val*np.identity(3)]

weights = [float(1/3), float(1/3), float(1/3)]

# gaussian function
def gau(mean, var, varInv, feature):
    a = np.sqrt(2*np.pi**3)
    b = np.exp(-0.5*np.dot((feature-mean), np.dot(varInv, (feature-mean).transpose())))
    return b/a

# calculating responsibilities
def res(likelihoods):
    tempList = []
    for comp in likelihoods:
        tempList.append(comp/sum(likelihoods))
    return tempList

# calculating likelihoods
def likeli(mean, var, varInv, weights, feature):
    temp = []
    for x in v:
        temp.append(weights[x]*gau(mean[x], var[x], varInv[x], feature))
    return temp
```

```
varInv = [np.linalg.inv(var[0]), np.linalg.inv(var[1]), np.linalg.inv(var[2])]
meanPrev = [np.array([0, 0, 0]), np.array([0, 0, 0]), np.array([0, 0, 0])]
iteration = []
logLikelihoods = []
counterr = 0

# iterating until convergence is reached
while sum(sum(np.absolute(np.asarray(mean) - np.asarray(meanPrev)))) >= 3:
        resp = []
        likelihoods = []
        for feature in feat:
                classLikelihoods = likeli(mean, var, varInv, weights, feature)
                rspblts = res(classLikelihoods)
                likelihoods.append(sum(classLikelihoods))
                resp.append(rspblts)
        logLikelihoods.append(sum(np.log(likelihoods)))
        nK = [sum(np.asarray(resp)[:,0:1]), sum(np.asarray(resp)[:,1:2]), sum(np.asarray(resp)[:,2:3])]
        weights = [float(nK[0]/N), float(nK[1]/N), float(nK[2]/N)]
        meanIterator =  np.dot(np.asarray(resp).T, feat)
        meanPrev = mean
        mean = [meanIterator[0]/nK[0], meanIterator[1]/nK[1], meanIterator[2]/nK[2]]
        counterr += 1
        iteration.append(counterr)

resp = []
for feature in feat:
        classLikelihoods = likeli(mean, var, varInv, weights, feature)
        rspblts = res(classLikelihoods)
        resp.append(rspblts)

result = []
counter = 0
segmentedImage = np.zeros((N, np.shape(img)[2]), np.uint8)

# assigning values to pixels of different segments
for response in resp:
  maxResp = max(response)
  respmax = response.index(maxResp)
  result.append(respmax)
  segmentedImage[counter] = mean[respmax]
  counter = counter + 1

blue0 = segmentedImage[:,0]
green0 = segmentedImage[:,1]
red0 = segmentedImage[:,2]
```

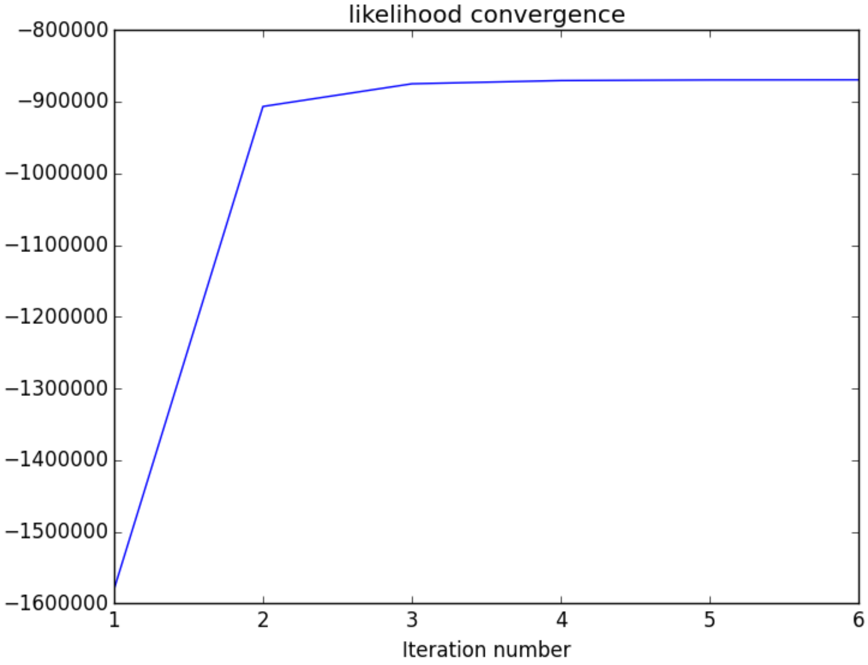
```
# rgb values of all the pixels segmented according to gaussian models
blue = np.reshape(blue0.flatten(), (np.shape(img)[0],np.shape(img)[1]))
green = np.reshape(green0.flatten(), (np.shape(img)[0],np.shape(img)[1]))
red = np.reshape(red0.flatten(), (np.shape(img)[0],np.shape(img)[1]))

recns = np.zeros((320, 480, 3))

for i in range(320):
        for j in range(480):
                recns[i][j] = np.array([blue[i][j], green[i][j], red[i][j]])

# plotting segmented image
pyplot.imshow(recns)
pyplot.show()

# plotting the graph of likelihood versus number of iterations
pyplot.plot(iteration, logLikelihoods)
pyplot.title('likelihood convergence')
pyplot.ylabel('Likelihood')
pyplot.xlabel('Iteration number')

# Show the figure.
pyplot.show()
```

Results

Segmented output



Log likelihood graph


likelihood convergence
Iteration number