Color Image segmentation with EM algorithm

Submitted by: Taneesh, 2018csb1186

Submitted to Dr. Ramanathan

How to run:: Submitted is the python notebook run it after having proper paths setup.

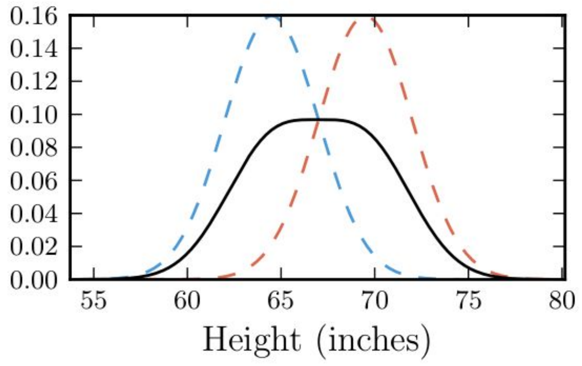
Here is a detailed description of what are the key takeaways from the lab1 assignment which is color image segmentation.

Doing things without knowing the reason behind them is like shooting the arrow in the dark and hoping to get the target shot.

So first I give a brief description of the maths involved which will tell us why are we doing whatever we are doing.
For digging deep into this I searched on internet and found this very helpful note from MIT on EM and the various proofs involved.
https://people.csail.mit.edu/rameshvs/content/gmm-em.pdf

Now our objective is to segment the image into various regions based on the value of the pixels in the image.
For this what we assume is that the pixels in image follow gaussian distribution and the image is the culmination of these different gaussians. Hence we make the groups in our data and make the model being that the data in each group follows normal distribution.
Hence we apply our this model to image where groups will be colors.
For the sake of understanding the example of data of boys and girls is used.

Let's suppose that we have data of boys' and girls' height, and we know that which entry is of which gender.
Then we can plot the data and see what distribution they follow, whether it is gaussian or something else.



But what if we have the values of heights but we want to know which gender this height will belong to.
Though there is no sure short way but we can judge this by probability and guess the answer which turns out to be highly accurate.

Hence if we know the gender we can find the distribution, if we know the distribution we can find the probability of the gender.
Hence now it's a chicken and egg problem.
To resolve this we assume that data does follow gaussian distribution and assigns the first parameter by ourselves and then we see below how does it go well with an insightful algorithm called Expectation Maximisation Algorithm. (EM)

What we want to do here is we want to maximize the log likelihood of our prediction,
We know how can we maximize it's by equating derivative to zero.

Doing it is direct in 1-dimension

$$p_{X_1^n}(x_1^n) = \prod_{i=1}^{n} \mathcal{N}(x_i; \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-(x_1-\mu)^2/2\sigma^2}$$

$$\ln p_{X_1^n}(x_1^n) = \sum_{i=1}^{n} \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2\sigma^2}(x_i-\mu)^2$$

$$\frac{d}{d\mu} \ln p_{X_1^n}(x_1^n) = \sum_{i=1}^{n} \frac{1}{\sigma^2}(x_i-\mu)$$

Hence u is average of all the values and it will give us our gaussian.

Note: It’s assumed in the model that sigma is same for all gaussians.

But for more than one dimension it’s not straightforward.

Below are the mathematical which I derived and understood with the help of the above link.

$$
\begin{aligned}
p_{Y_i}(y_i) &= \sum_{c_i} p_{C_i}(c_i) p_{Y_i|C_i}(y_i|c_i) \\
&= \sum_{c_i} \left(\pi_c \mathcal{N}(y_i; \mu_C, \sigma^2)\right)^{\mathbb{1}(c_i=c)} \\
&= q\mathcal{N}(y_i; \mu_M, \sigma^2) + (1-q)\mathcal{N}(y_i; \mu_F, \sigma^2)
\end{aligned}
$$

Now, the joint density of all the observations is:

$$
p_{Y_1^n}(y_1^n) = \prod_{i=1}^{n} \left(q\mathcal{N}(y_i; \mu_M, \sigma^2) + (1-q)\mathcal{N}(y_i; \mu_F, \sigma^2)\right),
$$

and the log-likelihood of the parameters is then

$$
\ln p_{Y_1^n}(y_1^n) = \sum_{i=1}^{n} \ln \left(\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)\right),
$$

We want to maximize this log likelihood. Hence we differentiate it and get below equation.

$$
\sum_{i=1}^{n} \frac{1}{\pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) + \pi_F \mathcal{N}(y_i; \mu_F, \sigma^2)} \pi_M \mathcal{N}(y_i; \mu_M, \sigma^2) \frac{y_i - \mu_M}{\sigma^2} = 0
$$

Now it’s not straightforward to solve it and we are stuck. But let’s look at an observation.

The probability that a person / pixel belongs to given gender/segment is :::

$$p_{C_i|Y_i}(c_i|y_i) = \frac{p_{Y_i|C_i}(y_i|c_i)p_{C_i}(c_i)}{p_{Y_i}(y_i)}$$

$$= \frac{\prod_{c\in\{M,F\}}\left(\pi_c \mathcal{N}(y_i;\mu_c,\sigma^2)\right)^{\mathbb{1}(c=c_i)}}{\pi_M \mathcal{N}(y_i;\mu_M,\sigma^2) + \pi_F \mathcal{N}(y_i;\mu_F,\sigma^2)} = q_{C_i}(c_i)$$

Hence it’s easy to observe that some terms are matching here and this is key insight here.

From the above claims it’s easy to observe.

$$\sum_{i=1}^{n} q_{C_i}(M)\frac{y_i-\mu_M}{\sigma^2} = 0$$

$$\mu_M = \frac{\sum_{i=1}^{n} q_{C_i}(M)y_i}{\sum_{i=1}^{n} q_{C_i}(M)}$$

Which is saying that u sum the probabilities that pixel belongs to that color for all pixels weighted by pixels value , and divide by the probabilities sum.
You would get the mu for which expectation is maximum.

Right now it’s chicken egg problem because without the pi and initial u we can’t get the probabilities of each pixel belonging to different segments.

Here come the EM algorithm.

What we do it we assign some values for our initial guess like for pi that is probabilities of each color be equal that is 1/ num_color
And mu be all same. (sigma is always same)

Here is a rigorous proof for EM algorithm that it always converges it’s due to the fact that log has double derivative negative and hence is inverted which gives that log(E(X)) >=E(log(X))

This is Jensen’s inequality Valid for convex functions.

Hence what we do if assign u and pi initially
Now we iterate through
With the help of u and pi we get , what is the probability that pixel belongs to this color. For each pixel -> E step
And with the help of this we find mu for which our the likelihood of our guess is largest. -> M step.

If the difference between new and old values is less than a threshold we say converged.
else we again do E step.
Shown are the mathematical formulations which prove our claim.

We want to maximize the log-likelihood:

$$
\begin{aligned}
& \log p_Y(y;\theta) \\
\text{(Marginalizing over } C \text{ and introducing } q_C(c)/q_C(c)) \quad & = \log\left(\sum_c q_C(c)\frac{p_{Y,C}(y,c;\theta)}{q_C(c)}\right) \\
\text{(Rewriting as an expectation)} \quad & = \log\left(\mathbb{E}_{q_C}\left[\frac{p_{Y,C}(y,C;\theta)}{q_C(C)}\right]\right) \\
\text{(Using Jensen's inequality)} \quad & \geq \mathbb{E}_{q_C}\left[\log\frac{p_{Y,C}(y,C;\theta)}{q_C(C)}\right]
\end{aligned}
$$

Let's rearrange the last version:

$$\mathbb{E}_{q_C}\left[\log\frac{p_{Y,C}(y,C;\theta)}{q_C(C)}\right] = \mathbb{E}_{q_C}\left[\log p_{Y,C}(y,C;\theta)\right] - \mathbb{E}_{q_C}[\log q_C(C)]$$

Maximizing with respect to θ will give us:

$$\boxed{\widehat{\theta} \leftarrow \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{q_C}\left[\log p_{Y,C}(y,C;\theta)\right]}$$

That's the M-step. Now we'll rearrange a different way:

$$
\begin{aligned}
\mathbb{E}_{q_C}\left[\log\frac{p_{Y,C}(y,C;\theta)}{q_C(C)}\right] & = \mathbb{E}_{q_C}\left[\log\frac{p_Y(y;\theta)p_{C|Y}(C|y;\theta)}{q_C(C)}\right] \\
& = \log p_Y(y;\theta) - \mathbb{E}_{q_C}\left[\log\frac{q_C(C)}{p_{C|Y}(C|y;\theta)}\right] \\
& = \log p_Y(y;\theta) - D(q_C(\cdot)||p_{C|Y}(\cdot|y;\theta))
\end{aligned}
$$

Maximizing with respect to q_C will give us:

$$\boxed{\widehat{q}_C(\cdot) \leftarrow p_{C|Y}(\cdot|y;\theta)}$$

Here it's clear that we have to fix the number of segments already else algo won't work.

Coding Part:::

Though enough comments are given and it's thoroughly explained let me give a brief outline about the code.

We iterate for each image
And we have kept 2, 3, 4, 5 segments and we divide the image into them.

We have kept u array for each channel corresponding to each color.

```
 mu = (1/num_seg)*( np.ones((num_seg, num_color),
dtype='float'))
```

And we have kept pi array which tells the relative amounts of color segments.

Then we run EM algorithm as described by the maths above.

Ws array is holding the value of probability that the pixel belongs to a given segment for each pixel .

Calculating this is E step.

```
 mu = (1/num_seg)*( np.ones((num_seg, num_color),
dtype='float'))
```

For M step we calculate the mu for which our estimate's probability is maximum and this is derived by the maths as weighted average of all the Ws probability, and pi is probability sum by total number of pixels.

```
for seg_ctr in range(num_seg):
                prob_sum=0
                for pix_ctr in range(num_pixels):
                    mu[seg_ctr] = mu[seg_ctr,:][np.newaxis] +
pixels[pix_ctr].T * (Ws[pix_ctr, seg_ctr])
                    prob_sum+= Ws[pix_ctr, seg_ctr]

                mu[seg_ctr]= mu[seg_ctr]/prob_sum
                pi[seg_ctr] = prob_sum/num_pixels

     if (muDiffSq < .0000001 and piDiffSq < .0000001):
```

```
                print('Convergence Criteria Met at Iteration: 
',iter, ' Exiting code')
                break
```

This is for checking the convergence.

We use the value of u calculated in the final step and to each pixel assign the color segment to which it’s probability of belonging is maximum.

Finally kmeans algorithm is used to make the clusters though without it too the image has already segmented, we assign chosen colors to the images through it.
And we save the images finally.

Hence we successfully segmented the color images.

code

October 27, 2020

[4]:
```
import matplotlib.image as mpimage
import matplotlib.pyplot as plt
import os
from os.path import join
import numpy as np
from PIL import Image
import matplotlib.image as mpimg
from skimage.color import rgb2gray
from skimage.color import label2rgb
from skimage.filters import gaussian
from sklearn.cluster import KMeans
import cv2 as opencv
import time
```

[12]:
```
plt.close('all')
clear = lambda: os.system('clear')
colors = [[1,0,0],[0,1,0],[0,0,1],[0.5,0.5,0.5],[0.5,0,0.2]]
```

[6]:
```
np.random.seed(110)
segmentCounts = [2,3,4,5]
inputpath="/home/taneesh/5thSem/Computer Vision/EM_color_image_segmentation/
↪Lab1_question_&_data/Input"
```

[14]:
```
def pixels_num(imagepath):
    w, h =Image.open(imagepath).size
    return w*h



for imagename in os.listdir(inputpath):
    for segments in segmentCounts:
        #print(imagename)
        image=os.path.join(inputpath, imagename)
        img1=mpimg.imread(image)
        #plt.figure()
        #plt.imshow(img)
       # print('Using Matplotlib Image Library: Image is of datatype ',img.
↪dtype,'and size ',img.shape)
```

```
        #using PIL
        img2= Image.open(image)
        print('Using Pillow (Python Image Library): Image is of datatype ',img1.
↪dtype,'and size ',img1.shape)
        num_seg=segments
        num_pixels=pixels_num(image)
        iter_count=20
        num_color=3
        outputpath =os.path.join(''.join(['Output/',str(num_seg), '_segments/',␣
↪imagename[:-4], '/']))
        if not (os.path.exists(outputpath)):
            os.makedirs(outputpath)

        img2.save(os.path.join(outputpath, "0.png"))
        pixels=np.asarray(img2)
       # print(imgarray)
        #print("hello")
        pixels=pixels.reshape(num_pixels, num_color,1)
        # here we are initializing the vectors for calculations
        # pi is the proportions vector which tells the relative proportions of␣
↪the presence of each segment of the image.

        pi = 1/num_seg*(np.ones((num_seg, 1),dtype='float'))
        increment = np.random.normal(0,.0001,1)
        for seg_ctr in range(len(pi)):
            if(seg_ctr%2==1):
                pi[seg_ctr] = pi[seg_ctr] + increment
            else:
                pi[seg_ctr] = pi[seg_ctr] - increment

        #%% "here we initialize the mu matrix which is num_seg * num_color␣
↪matrix, that is means of the particular segment that is what is the color of␣
↪the each segment. We first initialize it by ones.
        mu = (1/num_seg)*( np.ones((num_seg, num_color), dtype='float'))

        for seg_ctr in range(num_seg):
            if(seg_ctr%2==1):
                increment = np.random.normal(0,.0001,1)
            for col_ctr in range(num_color):
                #print(increment)
                if(seg_ctr%2==1):
                    mu[seg_ctr,col_ctr] = np.mean(pixels[:,col_ctr]) + increment
                else:
                    mu[seg_ctr,col_ctr] = np.mean(pixels[:,col_ctr]) - increment
        #now we have setup the things and now need to iterate which is the soul␣
↪of EM algorithm.
        mu_last_iter=mu;
        pi_last_iter=pi;

        for iter in range(iter_count):
            print(''.join(['Image: ',imagename,' num_seg: ',str(num_seg),␣
↪'iteration:',  str(iter+1), ' E-step']))
            Ws = np.ones((num_pixels,num_seg),dtype='float')
            for pix_ctr in range(num_pixels):
                logAjVec=np.zeros((num_seg, 1), dtype='float')
                for seg_ctr in range(num_seg):
                    x_minus_mu_T = np.transpose(pixels[pix_ctr,:]-(mu[seg_ctr,:
↪])[np.newaxis].T)

                    x_minus_mu= ((pixels[pix_ctr,:]-(mu[seg_ctr,:])[np.newaxis].
↪T))



                    logAjVec[seg_ctr] = np.log(pi[seg_ctr]) - .5*(np.
↪dot(x_minus_mu_T,x_minus_mu))

                logAmax=max(logAjVec.tolist())
                thirdterm=0
                for seg_ctr in range(num_seg):
                    thirdterm=thirdterm+np.exp(logAjVec[seg_ctr] - logAmax)
                for seg_ctr in range(num_seg):
                    logY = logAjVec[seg_ctr] - logAmax - np.log(thirdterm)
                    Ws[pix_ctr][seg_ctr] = np.exp(logY)


        ␣
↪##########################################################################################
        ############### M-step#############

            print(''.join(['Image: ',imagename,' num_seg: ',str(num_seg),␣
↪'iteration:',  str(iter+1), ' M-step:mixture coefficients']))
            mu = np.zeros((num_seg, num_color), dtype='float')
            #mean color for each segment
            pi = np.zeros((num_seg, 1), dtype='float') #proportion in which the␣
↪segments are there.

            for seg_ctr in range(num_seg):
                prob_sum=0
                for pix_ctr in range(num_pixels):
                    mu[seg_ctr] = mu[seg_ctr,:][np.newaxis] + pixels[pix_ctr].T␣
↪* (Ws[pix_ctr, seg_ctr])
```

```
                    prob_sum+= Ws[pix_ctr, seg_ctr]

                mu[seg_ctr]= mu[seg_ctr]/prob_sum
                pi[seg_ctr] = prob_sum/num_pixels

            #print(np.transpose(pi))
           # print("    ")

            muDiffSq = np.sum(np.multiply((mu - mu_last_iter),(mu␣
↪-mu_last_iter)))
            piDiffSq = np.sum(np.multiply((pi - pi_last_iter),(pi -␣
↪pi_last_iter)))

            if (muDiffSq < .0000001 and piDiffSq < .0000001):
                print('Convergence Criteria Met at Iteration: ',iter, ' Exiting␣
↪code')
                break

            mu_last_iter= mu
            pi_last_iter=pi
            segpixels=np.array(pixels)
            cluster=0
            for pix_ctr in range(num_pixels):
                cluster=np.where(Ws[pix_ctr,:]== max(Ws[pix_ctr,:]))
                vec= np.squeeze(np.transpose(mu[cluster,:]))
                segpixels[pix_ctr,:]= vec.reshape(vec.shape[0], 1)

            """ Save segmented image at each iteration. For displaying␣
↪consistent image clusters, it would be useful to blur/smoothen the segpixels␣
↪image using a Gaussian filter.
                Prior to smoothing, convert segpixels to a Grayscale image, and␣
↪convert the grayscale image into clusters based on pixel intensities"""

            segpixels = np.reshape(segpixels,(img1.shape[0], img1.shape[1],␣
↪num_color))
            # it's rgb array of image
            segmented_image= Image.fromarray(segpixels)

            #img2.save(os.path.join(outputpath, "0.png"))

            #segmented_image.save(os.path.join(outputpath, pic_name))

            #seg pixels is segmented image's numpy array.
            #rgb_weights for image conversion are used to be [0.2989, 0.5870, 0.
↪1140].
            # here I manually turned image into grayscale
            #rgb_weights= [0.2989, 0.5870, 0.1140]
            #grayscale_seg= np.dot(segpixels[..., :3], rgb_weights)
            #info = np.iinfo(segpixels.dtype)
            #grayscale_seg= grayscale_seg.astype(np.float)/ info.max
            #grayscale_seg = grayscale_seg * 255
            #grayscale_seg=grayscale_seg.astype(np.uint8)


            segpixels = rgb2gray(segpixels.astype('uint8')) # convert to␣
↪grayscale

            # now I have grayscale uint array on which kmeans can be used;

            ############## using Kmeans to cluseter grayscale␣
↪image###########33
            kmeans = KMeans(num_seg).fit(np.reshape(segpixels, (segpixels.
↪shape[0] * segpixels.shape[1], 1)))


            labels=kmeans.labels_
            labels=np.reshape(labels, (img1.shape[0], img1.shape[1]))
            labels = gaussian(np.clip(label2rgb(labels, colors=colors ), a_min␣
↪= 0, a_max = 1), sigma = 2,                    multichannel=False)
            pic_name= str(iter+1)
            pic_name+=".png"

            mpimg.imsave(os.path.join(outputpath, pic_name), labels)
```

```
ype  float32 and size  (480, 319, 3)
Image: jump.png num_seg: 5iteration:1 E-step
Image: jump.png num_seg: 5iteration:1 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:2 E-step
Image: jump.png num_seg: 5iteration:2 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:3 E-step
Image: jump.png num_seg: 5iteration:3 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:4 E-step
Image: jump.png num_seg: 5iteration:4 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:5 E-step
Image: jump.png num_seg: 5iteration:5 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:6 E-step
Image: jump.png num_seg: 5iteration:6 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:7 E-step
Image: jump.png num_seg: 5iteration:7 M-step:mixture coefficients
```

```
Image: jump.png num_seg: 5iteration:8 E-step
Image: jump.png num_seg: 5iteration:8 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:9 E-step
Image: jump.png num_seg: 5iteration:9 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:10 E-step
Image: jump.png num_seg: 5iteration:10 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:11 E-step
Image: jump.png num_seg: 5iteration:11 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:12 E-step
Image: jump.png num_seg: 5iteration:12 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:13 E-step
Image: jump.png num_seg: 5iteration:13 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:14 E-step
Image: jump.png num_seg: 5iteration:14 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:15 E-step
Image: jump.png num_seg: 5iteration:15 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:16 E-step
Image: jump.png num_seg: 5iteration:16 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:17 E-step
Image: jump.png num_seg: 5iteration:17 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:18 E-step
Image: jump.png num_seg: 5iteration:18 M-step:mixture coefficients
Image: jump.png num_seg: 5iteration:19 E-step
Image: jump.png num_seg: 5iteration:19 M-step:mixture coefficients
Convergence Criteria Met at Iteration:  18  Exiting code
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(492, 654, 3)
Image: tiger.png num_seg: 2iteration:1 E-step
Image: tiger.png num_seg: 2iteration:1 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:2 E-step
Image: tiger.png num_seg: 2iteration:2 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:3 E-step
Image: tiger.png num_seg: 2iteration:3 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:4 E-step
Image: tiger.png num_seg: 2iteration:4 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:5 E-step
Image: tiger.png num_seg: 2iteration:5 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:6 E-step
Image: tiger.png num_seg: 2iteration:6 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:7 E-step
Image: tiger.png num_seg: 2iteration:7 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:8 E-step
Image: tiger.png num_seg: 2iteration:8 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:9 E-step
Image: tiger.png num_seg: 2iteration:9 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:10 E-step
Image: tiger.png num_seg: 2iteration:10 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:11 E-step
```

```
Image: tiger.png num_seg: 2iteration:11 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:12 E-step
Image: tiger.png num_seg: 2iteration:12 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:13 E-step
Image: tiger.png num_seg: 2iteration:13 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:14 E-step
Image: tiger.png num_seg: 2iteration:14 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:15 E-step
Image: tiger.png num_seg: 2iteration:15 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:16 E-step
Image: tiger.png num_seg: 2iteration:16 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:17 E-step
Image: tiger.png num_seg: 2iteration:17 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:18 E-step
Image: tiger.png num_seg: 2iteration:18 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:19 E-step
Image: tiger.png num_seg: 2iteration:19 M-step:mixture coefficients
Image: tiger.png num_seg: 2iteration:20 E-step
Image: tiger.png num_seg: 2iteration:20 M-step:mixture coefficients
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(492, 654, 3)
Image: tiger.png num_seg: 3iteration:1 E-step
Image: tiger.png num_seg: 3iteration:1 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:2 E-step
Image: tiger.png num_seg: 3iteration:2 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:3 E-step
Image: tiger.png num_seg: 3iteration:3 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:4 E-step
Image: tiger.png num_seg: 3iteration:4 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:5 E-step
Image: tiger.png num_seg: 3iteration:5 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:6 E-step
Image: tiger.png num_seg: 3iteration:6 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:7 E-step
Image: tiger.png num_seg: 3iteration:7 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:8 E-step
Image: tiger.png num_seg: 3iteration:8 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:9 E-step
Image: tiger.png num_seg: 3iteration:9 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:10 E-step
Image: tiger.png num_seg: 3iteration:10 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:11 E-step
Image: tiger.png num_seg: 3iteration:11 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:12 E-step
Image: tiger.png num_seg: 3iteration:12 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:13 E-step
Image: tiger.png num_seg: 3iteration:13 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:14 E-step
```

```
Image: tiger.png num_seg: 3iteration:14 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:15 E-step
Image: tiger.png num_seg: 3iteration:15 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:16 E-step
Image: tiger.png num_seg: 3iteration:16 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:17 E-step
Image: tiger.png num_seg: 3iteration:17 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:18 E-step
Image: tiger.png num_seg: 3iteration:18 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:19 E-step
Image: tiger.png num_seg: 3iteration:19 M-step:mixture coefficients
Image: tiger.png num_seg: 3iteration:20 E-step
Image: tiger.png num_seg: 3iteration:20 M-step:mixture coefficients
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(492, 654, 3)
Image: tiger.png num_seg: 4iteration:1 E-step
Image: tiger.png num_seg: 4iteration:1 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:2 E-step
Image: tiger.png num_seg: 4iteration:2 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:3 E-step
Image: tiger.png num_seg: 4iteration:3 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:4 E-step
Image: tiger.png num_seg: 4iteration:4 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:5 E-step
Image: tiger.png num_seg: 4iteration:5 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:6 E-step
Image: tiger.png num_seg: 4iteration:6 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:7 E-step
Image: tiger.png num_seg: 4iteration:7 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:8 E-step
Image: tiger.png num_seg: 4iteration:8 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:9 E-step
Image: tiger.png num_seg: 4iteration:9 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:10 E-step
Image: tiger.png num_seg: 4iteration:10 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:11 E-step
Image: tiger.png num_seg: 4iteration:11 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:12 E-step
Image: tiger.png num_seg: 4iteration:12 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:13 E-step
Image: tiger.png num_seg: 4iteration:13 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:14 E-step
Image: tiger.png num_seg: 4iteration:14 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:15 E-step
Image: tiger.png num_seg: 4iteration:15 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:16 E-step
Image: tiger.png num_seg: 4iteration:16 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:17 E-step
```

```
Image: tiger.png num_seg: 4iteration:17 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:18 E-step
Image: tiger.png num_seg: 4iteration:18 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:19 E-step
Image: tiger.png num_seg: 4iteration:19 M-step:mixture coefficients
Image: tiger.png num_seg: 4iteration:20 E-step
Image: tiger.png num_seg: 4iteration:20 M-step:mixture coefficients
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(492, 654, 3)
Image: tiger.png num_seg: 5iteration:1 E-step
Image: tiger.png num_seg: 5iteration:1 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:2 E-step
Image: tiger.png num_seg: 5iteration:2 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:3 E-step
Image: tiger.png num_seg: 5iteration:3 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:4 E-step
Image: tiger.png num_seg: 5iteration:4 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:5 E-step
Image: tiger.png num_seg: 5iteration:5 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:6 E-step
Image: tiger.png num_seg: 5iteration:6 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:7 E-step
Image: tiger.png num_seg: 5iteration:7 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:8 E-step
Image: tiger.png num_seg: 5iteration:8 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:9 E-step
Image: tiger.png num_seg: 5iteration:9 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:10 E-step
Image: tiger.png num_seg: 5iteration:10 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:11 E-step
Image: tiger.png num_seg: 5iteration:11 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:12 E-step
Image: tiger.png num_seg: 5iteration:12 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:13 E-step
Image: tiger.png num_seg: 5iteration:13 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:14 E-step
Image: tiger.png num_seg: 5iteration:14 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:15 E-step
Image: tiger.png num_seg: 5iteration:15 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:16 E-step
Image: tiger.png num_seg: 5iteration:16 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:17 E-step
Image: tiger.png num_seg: 5iteration:17 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:18 E-step
Image: tiger.png num_seg: 5iteration:18 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:19 E-step
Image: tiger.png num_seg: 5iteration:19 M-step:mixture coefficients
Image: tiger.png num_seg: 5iteration:20 E-step
```

```
Image: tiger.png num_seg: 5iteration:20 M-step:mixture coefficients
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(312, 252, 3)
Image: water_coins.png num_seg: 2iteration:1 E-step
Image: water_coins.png num_seg: 2iteration:1 M-step:mixture coefficients
Image: water_coins.png num_seg: 2iteration:2 E-step
Image: water_coins.png num_seg: 2iteration:2 M-step:mixture coefficients
Image: water_coins.png num_seg: 2iteration:3 E-step
Image: water_coins.png num_seg: 2iteration:3 M-step:mixture coefficients
Image: water_coins.png num_seg: 2iteration:4 E-step
Image: water_coins.png num_seg: 2iteration:4 M-step:mixture coefficients
Image: water_coins.png num_seg: 2iteration:5 E-step
Image: water_coins.png num_seg: 2iteration:5 M-step:mixture coefficients
Image: water_coins.png num_seg: 2iteration:6 E-step
Image: water_coins.png num_seg: 2iteration:6 M-step:mixture coefficients
Convergence Criteria Met at Iteration:  5  Exiting code
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(312, 252, 3)
Image: water_coins.png num_seg: 3iteration:1 E-step
Image: water_coins.png num_seg: 3iteration:1 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:2 E-step
Image: water_coins.png num_seg: 3iteration:2 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:3 E-step
Image: water_coins.png num_seg: 3iteration:3 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:4 E-step
Image: water_coins.png num_seg: 3iteration:4 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:5 E-step
Image: water_coins.png num_seg: 3iteration:5 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:6 E-step
Image: water_coins.png num_seg: 3iteration:6 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:7 E-step
Image: water_coins.png num_seg: 3iteration:7 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:8 E-step
Image: water_coins.png num_seg: 3iteration:8 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:9 E-step
Image: water_coins.png num_seg: 3iteration:9 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:10 E-step
Image: water_coins.png num_seg: 3iteration:10 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:11 E-step
Image: water_coins.png num_seg: 3iteration:11 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:12 E-step
Image: water_coins.png num_seg: 3iteration:12 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:13 E-step
Image: water_coins.png num_seg: 3iteration:13 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:14 E-step
Image: water_coins.png num_seg: 3iteration:14 M-step:mixture coefficients
Image: water_coins.png num_seg: 3iteration:15 E-step
Image: water_coins.png num_seg: 3iteration:15 M-step:mixture coefficients
```

```
Image: water_coins.png num_seg: 3iteration:16 E-step
Image: water_coins.png num_seg: 3iteration:16 M-step:mixture coefficients
Convergence Criteria Met at Iteration:  15  Exiting code
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(312, 252, 3)
Image: water_coins.png num_seg: 4iteration:1 E-step
Image: water_coins.png num_seg: 4iteration:1 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:2 E-step
Image: water_coins.png num_seg: 4iteration:2 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:3 E-step
Image: water_coins.png num_seg: 4iteration:3 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:4 E-step
Image: water_coins.png num_seg: 4iteration:4 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:5 E-step
Image: water_coins.png num_seg: 4iteration:5 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:6 E-step
Image: water_coins.png num_seg: 4iteration:6 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:7 E-step
Image: water_coins.png num_seg: 4iteration:7 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:8 E-step
Image: water_coins.png num_seg: 4iteration:8 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:9 E-step
Image: water_coins.png num_seg: 4iteration:9 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:10 E-step
Image: water_coins.png num_seg: 4iteration:10 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:11 E-step
Image: water_coins.png num_seg: 4iteration:11 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:12 E-step
Image: water_coins.png num_seg: 4iteration:12 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:13 E-step
Image: water_coins.png num_seg: 4iteration:13 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:14 E-step
Image: water_coins.png num_seg: 4iteration:14 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:15 E-step
Image: water_coins.png num_seg: 4iteration:15 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:16 E-step
Image: water_coins.png num_seg: 4iteration:16 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:17 E-step
Image: water_coins.png num_seg: 4iteration:17 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:18 E-step
Image: water_coins.png num_seg: 4iteration:18 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:19 E-step
Image: water_coins.png num_seg: 4iteration:19 M-step:mixture coefficients
Image: water_coins.png num_seg: 4iteration:20 E-step
Image: water_coins.png num_seg: 4iteration:20 M-step:mixture coefficients
Using Pillow (Python Image Library): Image is of datatype  float32 and size
(312, 252, 3)
Image: water_coins.png num_seg: 5iteration:1 E-step
```

```
Image: water_coins.png num_seg: 5iteration:1 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:2 E-step
Image: water_coins.png num_seg: 5iteration:2 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:3 E-step
Image: water_coins.png num_seg: 5iteration:3 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:4 E-step
Image: water_coins.png num_seg: 5iteration:4 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:5 E-step
Image: water_coins.png num_seg: 5iteration:5 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:6 E-step
Image: water_coins.png num_seg: 5iteration:6 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:7 E-step
Image: water_coins.png num_seg: 5iteration:7 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:8 E-step
Image: water_coins.png num_seg: 5iteration:8 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:9 E-step
Image: water_coins.png num_seg: 5iteration:9 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:10 E-step
Image: water_coins.png num_seg: 5iteration:10 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:11 E-step
Image: water_coins.png num_seg: 5iteration:11 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:12 E-step
Image: water_coins.png num_seg: 5iteration:12 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:13 E-step
Image: water_coins.png num_seg: 5iteration:13 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:14 E-step
Image: water_coins.png num_seg: 5iteration:14 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:15 E-step
Image: water_coins.png num_seg: 5iteration:15 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:16 E-step
Image: water_coins.png num_seg: 5iteration:16 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:17 E-step
Image: water_coins.png num_seg: 5iteration:17 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:18 E-step
Image: water_coins.png num_seg: 5iteration:18 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:19 E-step
Image: water_coins.png num_seg: 5iteration:19 M-step:mixture coefficients
Image: water_coins.png num_seg: 5iteration:20 E-step
Image: water_coins.png num_seg: 5iteration:20 M-step:mixture coefficients
```

```
[ ]:
```