Chao Xu (chaox2)
Zhenbang “Zachary” Wu (zw12)
Edward Villasenor (villasr2)

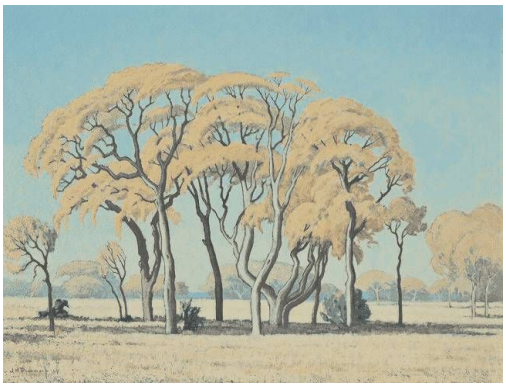
CS 498 AML Homework 8

1. Image 1 ie trees ie 2cf92c40c3f3306321d789f7e9c12893.jpg - segmented into 10, 20 and 50 segments

Number of Segments = 10

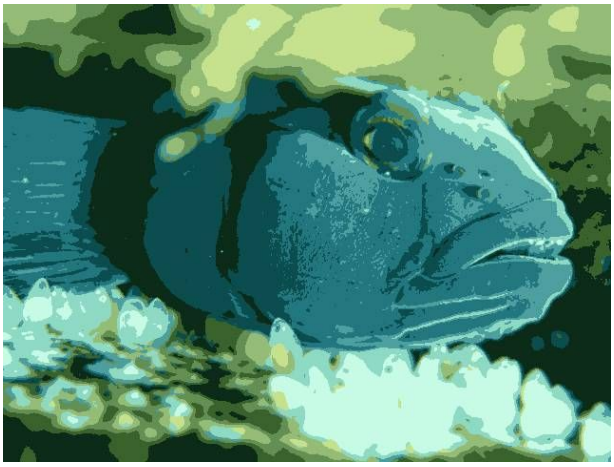
Number of Segments = 20

Number of Segments = 50



2. Image 2 ie RobertMixed03.jpg segmented into 10,20,50 segments

Number of Segments = 10



Number of Segments = 20

Number of Segments = 50

3. Image 3 ie smallstrelitzia.jpg segmented into 10, 20 and 50 segments

Number of Segments = 10

Number of Segments = 20

Number of Segments = 50

4. Image 4 ie smallsunset.jpg segmented into 10,20 and 50 segments

Number of Segments = 10

Number of Segments = 20

Number of Segments = 50

5. Display of tree image with 20 segments with 5 different initial points





6. Code snippets for EM initialisation, updates

EM Initialization

```
# there're NPixel (480*640) pixels
# choose t (NCluster: 10/20/50) centers randomly
# theta(n) = (Mu[1..t],Pi[1..t]), n: #loops
NPixel = Dim1[1]*Dim1[2]
t = NCluster[NCluster_choose]
x = FlatImage*10
# use KMeans to get initial points
Kmeans <- kmeans(x, t)
Mu <- Kmeans$centers
# randomly get initial points
# sampleRows = sample(1:NPixel, t)
# Mu = x[sampleRows,]
Pi = rep(1/t, t)
w = matrix(0, NPixel, t)
```

E-Step

Compute weights w_{ij} linking the i'th data item to the j'th cluster center, using

$$w_{ij}^{(n)} = \frac{[exp(-0.5(x_i-\mu_j^{(n)})^T(x_i-\mu_j^{(n)}))]\pi_j^{(n)}}{\sum_k[exp(-0.5(x_i-\mu_k^{(n)})^T(x_i-\mu_k^{(n)}))]\pi_k^{(n)}}$$

M-Step

Estimate new μ and π

$$\mu_j^{(n+1)} = \frac{\sum_i x_i w_{ij}^{(n)}}{\sum_i w_{ij}^{(n)}}$$

$$\pi_j^{(n+1)} = \frac{\sum_i w_{ij}^{(n)}}{N}$$

```
# for testing
NLoops = 0
repeat{
    NLoops = NLoops + 1
    old_Pi = Pi
    # E-Step
    for (i in 1:NPixel){
        sum = 0
        for (k in 1:t)
            sum = sum + (exp(-0.5*t(x[i,]-Mu[k,])%*%(x[i,]-Mu[k,])))*Pi[k]
        for (j in 1:t)
            w[i,j] = ((exp(-0.5*t(x[i,]-Mu[j,])%*%(x[i,]-Mu[j,]))*Pi[j])/sum)
    }
    # M-Step
    for (j in 1:t){
        sum2 = 0
        for (i in 1:NPixel){
            sum2 = sum2 + x[i,]*w[i,j]
        }
        Mu[j,] = sum2/sum(w[,j])
        Pi[j] = sum(w[,j])/NPixel
    }
    print(NLoops)
    print(mean(Pi-old_Pi))
    # check if terminate
    if (abs(mean(Pi-old_Pi)) < 0.00000001) break
}
```

CS498 AML HW8 - EM Algorithm

In [ ]:
```
# import necessary libraries
# readJPEG(), writeJPEG()
library(jpeg)
# Reshape()
library(pracma)
# registerDoParallel()
library(doParallel)
# accelerate using parallel computing
registerDoParallel(makeCluster(detectCores()))
```

Setting Parameters

In [ ]:
```
# variable used to choose image and number of clusters
Image_choose = 4
NCluster_choose = 2
```

Read data

In [ ]:
```
# INPUT: read blog data (train & test)
ImageStrings = c("RobertMixed03.jpg","smallstrelitzia.jpg","smallsunset.jpg", "tree.jpg")
NCluster = c(10, 20, 50)
# Note: double-type (original RGB value divided by 255)
Image = readJPEG(ImageStrings[Image_choose])
# A 3d 480*640*3 matrix (Row*Col*R/G/B)
# out: 480, 640, 3
Dim1 = dim(Image)
# A 2d (480*640)*3 matrix ((Row*Col)*R/G/B)
FlatImage = Reshape(Image,Dim1[1]*Dim1[2],Dim1[3])
```

EM Initialization

In [ ]:
```
# there're NPixel (480*640) pixels
# choose t (NCluster: 10/20/50) centers randomly
# theta(n) = (Mu[1..t],Pi[1..t]), n: #loops
NPixel = Dim1[1]*Dim1[2]
t = NCluster[NCluster_choose]
x = FlatImage*10
# use KMeans to get initial points
Kmeans <- kmeans(x, t)
Mu <- Kmeans$centers
# randomly get initial points
# sampleRows = sample(1:NPixel, t)
# Mu = x[sampleRows,]
Pi = rep(1/t, t)
w = matrix(0, NPixel, t)
```

E-Step

Compute weights w_{ij} linking the i'th data item to the j'th cluster center, using

$$w_{ij}^{(n)} = \frac{[exp(-0.5(x_i-\mu_j^{(n)})^T(x_i-\mu_j^{(n)}))]\pi_j^{(n)}}{\sum_k [exp(-0.5(x_i-\mu_k^{(n)})^T(x_i-\mu_k^{(n)}))]\pi_k^{(n)}}$$

M-Step

Estimate new μ and π

$$\mu_j^{(n+1)} = \frac{\sum_i x_i w_{ij}^{(n)}}{\sum_i w_{ij}^{(n)}}$$

$$\pi_j^{(n+1)} = \frac{\sum_i w_{ij}^{(n)}}{N}$$

In [ ]:
```
# for testing
NLoops = 0
repeat{
    NLoops = NLoops + 1
    old_Pi = Pi
    # E-Step
    for (i in 1:NPixel){
        sum = 0
        for (k in 1:t)
            sum = sum + (exp(-0.5*t(x[i,]-Mu[k,])%*%(x[i,]-Mu[k,])))*Pi[k]
        for (j in 1:t)
            w[i,j] = ((exp(-0.5*t(x[i,]-Mu[j,])%*%(x[i,]-Mu[j,]))*Pi[j])/sum)
    }
    # M-Step
    for (j in 1:t){
        sum2 = 0
        for (i in 1:NPixel){
            sum2 = sum2 + x[i,]*w[i,j]
        }
        Mu[j,] = sum2/sum(w[,j])
        Pi[j] = sum(w[,j])/NPixel
    }
    print(NLoops)
    print(mean(Pi-old_Pi))
    # check if terminate
    if (abs(mean(Pi-old_Pi)) < 0.00000001) break
}
```

In [ ]:
```
# assign each segment to the Mu with biggest possibility
x_segmented = x
for (i in 1:NPixel){
    x_segmented[i,] = Mu[which.max(w[i,]),]
}
```

In [ ]:
```
# Reshape the image back to 3d
dim(x_segmented) = c(Dim1[1],Dim1[2],Dim1[3])
writeJPEG(x_segmented/10, target = "output.jpg")
```