guassian mixture models

September 24, 2018

1 Importing packages

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

2 Defining functions

```
In [2]: def multi_normal_pdf(x , mean , co_var):
            ex = -0.5 * ((x - mean.T).T * co_var.I * (x - mean.T))
            den = np.sqrt(((2*np.pi)**mean.shape[1]) * np.linalg.det(co_var))
            return np.exp(ex)/den

In [3]: def gaama(x , mean , co_var , weights):
            pdfs = np.asmatrix(np.zeros((len(weights), x.shape[1])))
            for i in range(0,x.shape[1]):
                for j in range(0,len(weights)):
                    pdfs[j , i] = weights[j] * multi_normal_pdf(x[:,i] , mean[j] , np.asmatrix(c
            pdfs_mean = np.asmatrix(np.mean(pdfs , axis = 0))
            pdfs_mean = np.repeat(pdfs_mean , pdfs.shape[0] , axis = 0)
            gama = pdfs / pdfs_mean
            return np.asmatrix(gama)

In [4]: def log_likelihood(x , mean , co_var , weights):
            likelihood = 0
            for i in range(0,x.shape[1]):
                temp_pdf = 0
                for j in range(0,len(weights)):
                    temp_pdf = temp_pdf + weights[j] * multi_normal_pdf(x[:,i] , mean[j] , np.as
                likelihood = likelihood +  np.log(temp_pdf)
            return likelihood
```

3 Parameters

```
In [86]: # no of mixtures
         mix_size = 2
         # dimensions of samples
```

```
dim_sample = 5
# no of samples
samples = 100
```

4 Generating Training samples

```
In [79]: data = np.zeros((dim_sample , samples))
         for i in range(0,mix_size):
         #     mean = np.random.randint(5,size = dim_sample)
             mean = np.linspace(i,i+1,dim_sample)
             temp = np.random.randint(5,size = (dim_sample,dim_sample))
             co_var = np.matmul(temp,np.transpose(temp))
             data = data + np.transpose(np.random.multivariate_normal(mean , co_var , samples))
         data = np.asmatrix(data/mix_size)
         print(data.shape , data[:,0].shape)
```

```
(5, 100) (5, 1)
```

5 Initializing parameters

```
In [83]: mean = np.asmatrix(np.random.rand(mix_size,dim_sample))
         # co_var = np.eye((mix_size,dim_sample , dim_sample))
         co_var = []
         for i in range(0,mix_size):
             co_var.append(np.eye((dim_sample)))
         co_var = np.array(co_var)
         weights = np.ones(mix_size)/mix_size
         diff = 1
         print(co_var.shape , mean.shape, weights)
```

```
(2, 5, 5) (2, 5) [0.5 0.5]
```

6 Finding the optimal parameters

```
In [85]: while(diff > 0):
         #     finding posterior
             gama = gaama(data , mean , co_var , weights)
             init = log_likelihood(data , mean , co_var , weights)
         #     print("before: " , mean[0,0] , "   " , co_var[0,0,0] , '   ' , weights[0])
             n_k = np.sum(gama , axis = 1)
             co_var = []
         #     updating mean , variance
             for i in range(0,mix_size):
                 mean[i] = (np.sum(np.multiply(data, np.asmatrix(np.repeat(gama[i] , dim_sample
                 temp = np.asmatrix(np.zeros((dim_sample,dim_sample)))
```

```
                for j in range(0,samples):
                    temp = temp + (data[:,j]-mean[i].T) * (data[:,j]-mean[i].T).T
                co_var.append(temp)
            co_var = np.asarray(co_var)
        #     updating weights
            weights = n_k/samples
            final = log_likelihood(data , mean , co_var , weights)
            diff = final - init
        #     print(diff)
        print("mean:\n" , mean , "\n weights: \n" , weights , "\n variance: \n" , co_var)
```

```
mean:
 [[0.69261395 1.19404131 1.31907804 1.43610957 1.36098601]
 [0.69261395 1.19404131 1.31907804 1.43610957 1.36098601]]
 weights:
 [[0.77838128]
 [1.22161872]]
 variance:
 [[[2258.62014054 2190.18843748 1608.90451071 1470.86643394 1052.07900726]
  [2190.18843748 2640.42101143 1517.4218878  1580.45206392 1113.99074293]
  [1608.90451071 1517.4218878  1863.23527617 1147.76148554  724.42285258]
  [1470.86643394 1580.45206392 1147.76148554 1347.77987653  627.32694861]
  [1052.07900726 1113.99074293  724.42285258  627.32694861  980.38577457]]

 [[2258.62014054 2190.18843748 1608.90451071 1470.86643394 1052.07900726]
  [2190.18843748 2640.42101143 1517.4218878  1580.45206392 1113.99074293]
  [1608.90451071 1517.4218878  1863.23527617 1147.76148554  724.42285258]
  [1470.86643394 1580.45206392 1147.76148554 1347.77987653  627.32694861]
  [1052.07900726 1113.99074293  724.42285258  627.32694861  980.38577457]]]
```