Q6-EM

May 5, 2019

0.0.1 import module

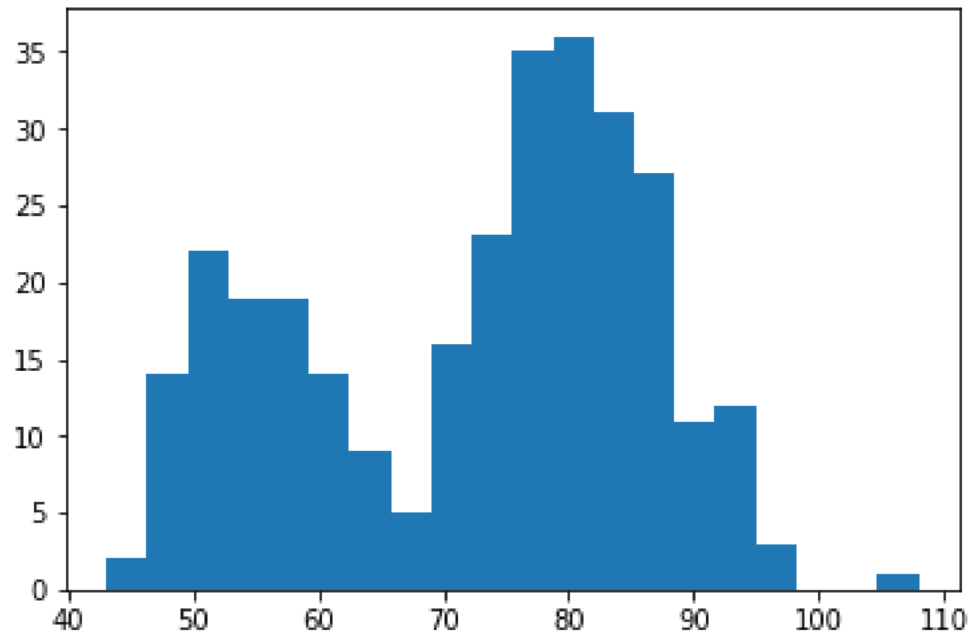
```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

0.0.2 input data

```
In [39]: lst = [80, 71, 57, 80, 75, 77, 60, 86, 77, 56, 81, 50, 89,
                54, 90, 73, 60, 83, 65, 82, 84, 54, 85, 58, 79, 57,
                88, 68, 76, 78, 74, 85, 75, 65, 76, 58, 91, 50, 87,
                48, 93, 54, 86, 53, 78, 52, 83, 60, 87, 49, 80, 60,
                92, 43, 89, 60, 84, 69, 74, 71, 108, 50, 77, 57, 80,
                61, 82, 48, 81, 73, 62, 79, 54, 80, 73, 81, 62, 81,
                71, 79, 81, 74, 59, 81, 66, 87, 53, 80, 50, 87, 51,
                82, 58, 81, 49, 92, 50, 88, 62, 93, 56, 89, 51, 79,
                58, 82, 52, 88, 52, 78, 69, 75, 77, 53, 80, 55, 87,
                53, 85, 61, 93, 54, 76, 80, 81, 59, 86, 78, 71, 77,
                76, 94, 75, 50, 83, 82, 72, 77, 75, 65, 79, 72, 78,
                77, 79, 75, 78, 64, 80, 49, 88, 54, 86, 51, 96, 50,
                80, 78, 81, 72, 75, 78, 87, 69, 55, 83, 49, 82, 57,
                84, 57, 84, 73, 78, 57, 79, 57, 90, 62, 87, 78, 52,
                98, 48, 78, 79, 65, 84, 50, 83, 60, 80, 50, 88, 50,
                84, 74, 76, 65, 89, 49, 88, 51, 78, 85, 65, 75, 77,
                69, 92, 68, 87, 61, 81, 55, 93, 53, 84, 70, 73, 93,
                50, 87, 77, 74, 72, 82, 74, 80, 49, 91, 53, 86, 49,
                79, 89, 87, 76, 59, 80, 89, 45, 93, 72, 71, 54, 79,
                74, 65, 78, 57, 87, 72, 84, 47, 84, 57, 87, 68, 86,
                75, 73, 53, 82, 93, 77, 54, 96, 48, 89, 63, 84, 76,
                62, 83, 50, 85, 78, 78, 81, 78, 76, 74, 81, 66, 84,
                48, 93, 47, 87, 51, 78, 54, 87, 52, 85, 58, 88, 79]
         faithful = np.array(lst)
```

0.1 (a) Draw the histogram

```
In [4]: plt.hist(lst,bins=20)
        plt.show()
```



0.2 (b) Assume a normal distribution, and estimate the mean and standard deviation

as we know MLE of normal distribution is

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{N} x_i \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

```
In [5]: muhat = np.mean(faithful)
        sigmahat = np.sqrt(np.mean((faithful-muhat)**2))
```

```
In [46]: print("When Assume Noraml maximum likelihood estimator muhat = %.3f , sigmahat = %.3f"
         print("log likelihood is : %f" %sum(np.log(NormalLikelihood(faithful,muhat,sigmahat))))
```

```
When Assume Noraml maximum likelihood estimator muhat = 72.318 , sigmahat = 13.870
log likelihood is : -1210.556881
```

so,

$$X \sim N(72.318, 13.870^2)$$

0.3 (c) Use EM algorithm and estimate $\pi_1, \mu_1, \sigma_1, \mu_2, \sigma_2$

First, initaizle the parameters

```
In [45]: initParam1=len(faithful[faithful<70])/len(faithful)\
                        ,faithful[faithful<70].mean() ,faithful[faithful<70].std()
         initParam2=len(faithful[faithful>=70])/len(faithful)\
                        ,faithful[faithful>=70].mean() ,faithful[faithful>=70].std()
```

Define the likelihood of normal distribution

$$L(\mu, \sigma | x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

```
In [8]: def NormalLikelihood(x,mu=0,sigma=1):
            out = np.exp(-((x-mu)**2)/(2*(sigma**2)))/(np.sqrt(2*np.pi)*sigma)
            return out
```

Define the log likelihood of Gaussian Mixture Model

$$l(\pi_1, \mu_1, \sigma_1, \mu_2, \sigma_2 | X) = \sum_{n=1}^{N} \ln \{\pi_1 L(\mu_1, \sigma_1 | x_i) + \pi_2 L(\mu_2, \sigma_2 | x_i)\}$$

where $\pi_2 = 1 - \pi_1$ and $X = \{x_1, \ldots, x_N\}$

```
In [9]: def logLikelihood(x,param1,param2):
            p1,mu1,sigma1 = param1
            p2,mu2,sigma2 = param2
            out = sum(np.log(p1*NormalLikelihood(x,mu1,sigma1)\
                             + p2*NormalLikelihood(x,mu2,sigma2)))
            return out
```

Let $p(z_k = 1) = \pi_k$ then,

$$p(x|z_k = 1) = L(\mu_k, \sigma_k | x) p(x|\mathbf{z}) = \prod_{k=1}^{2} L(\mu_k, \sigma_k | x)^{z_k} p(x) = \sum_{\mathbf{z}} p(\mathbf{z}) p(x|\mathbf{z}) = \sum_{k=1}^{2} \pi_k L(\mu_k, \sigma_k | x)^{z_k}$$

We can caculate $p(z_k = 1|x) = \gamma(z_k)$ by bayes' rule

$$\gamma(z_k) = \frac{\pi_k L(\mu_k, \sigma_k | x)}{\pi_1 L(\mu_1, \sigma_1 | x) + \pi_2 L(\mu_2, \sigma_2 | x)}$$

E-step calculate $p(\mathbf{z}|X, \theta^{(t)})$

```
In [10]: def estep(x,param1,param2,k):
             p1,mu1,sigma1 = param1
             p2,mu2,sigma2 = param2
             if k==1:
                 out = (p1*NormalLikelihood(x,mu1,sigma1))/(p1*NormalLikelihood(x,mu1,sigma1)\
                                                            + p2*NormalLikelihood(x,mu2,sigma2))
             else:
                 out = (p2*NormalLikelihood(x,mu2,sigma2))/(p1*NormalLikelihood(x,mu1,sigma1)\
                                                            + p2*NormalLikelihood(x,mu2,sigma2))
             return out
```

M-step

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

Where $\boldsymbol{\theta} = \{\pi_1, \mu_1, \sigma_1, \mu_2, \sigma_2\}$ and

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \sum_{\mathbf{z}} p(\mathbf{z}|X, \theta^{(t)}) \ln p(X, \mathbf{z}|\boldsymbol{\theta})$$

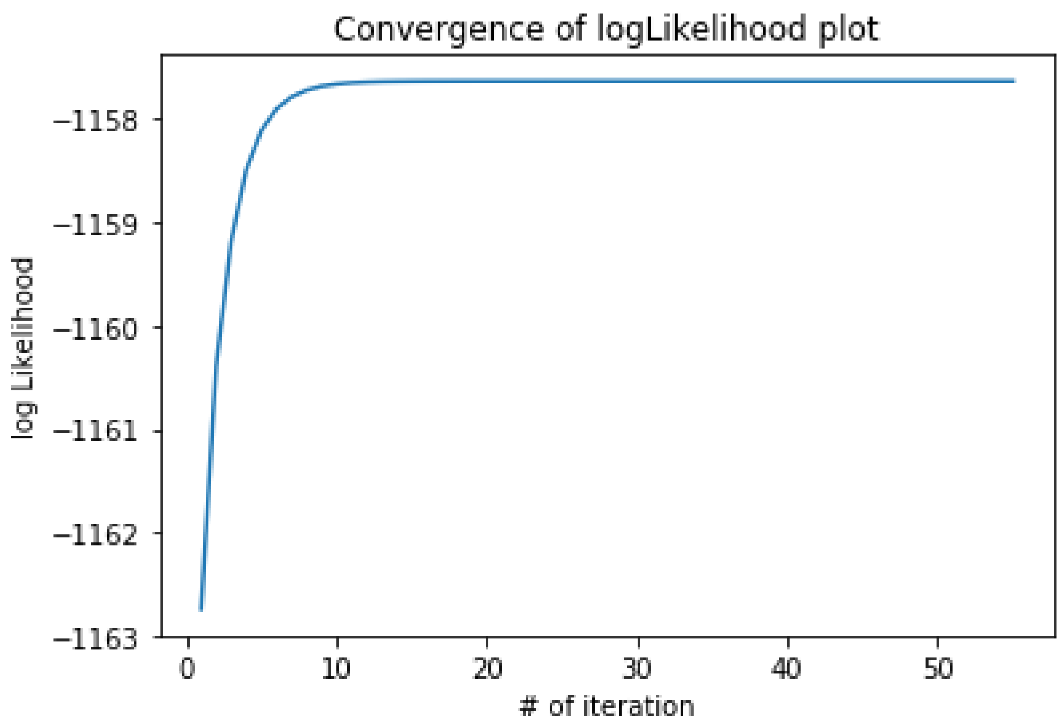
```
In [11]: def mstep(x,param1,param2):
             N = len(x)
             N1 = sum(estep(x,param1,param2,1))
             N2 = sum(estep(x,param1,param2,2))
             mu1New = sum(estep(x,param1,param2,1)*x)/N1
             mu2New = sum(estep(x,param1,param2,2)*x)/N2
             sigma1New = np.sqrt(sum(estep(x,param1,param2,1)*((x-mu1New)**2))/N1)
             sigma2New = np.sqrt(sum(estep(x,param1,param2,2)*((x-mu2New)**2))/N2)
             p1New = N1/N
             p2New = N2/N
             param1New = np.array([p1New,mu1New,sigma1New])
             param2New = np.array([p2New,mu2New,sigma2New])
             return param1New, param2New
```

Iterate E-step and M-step

```
In [34]: def EM(x,initParam1,initParam2,maxiter = 100):
             iteration = 0
             param1,param2 = initParam1,initParam2
             loglikeLst = []
             while iteration < maxiter:
                 iteration = iteration + 1
                 t0 = logLikelihood(x,param1,param2)
                 loglikeLst.append([iteration,t0])
                 param1,param2 = mstep(x,param1,param2)
                 t1 = logLikelihood(x,param1,param2)
                 if iteration%10 == 0:
                     print('%dth iteration\'s loglikelihood : %f' \
                           %(iteration,logLikelihood(x,param1,param2)))
                 if t0==t1:
                     print('%dth iteration\'s loglikelihood : %f' \
                           %(iteration,logLikelihood(x,param1,param2)))
                     break
             plt.plot(pd.DataFrame(loglikeLst)[0],pd.DataFrame(loglikeLst)[1])
             plt.title('Convergence of logLikelihood plot')
             plt.xlabel('# of iteration')
             plt.ylabel('log Likelihood')
             plt.show()
             return(param1,param2,t1)
```

```
In [35]: EMparam1, EMparam2 ,EMloglike= EM(faithful,initParam1=initParam1,initParam2=initParam2)
```

```
10th iteration's loglikelihood : -1157.649911
20th iteration's loglikelihood : -1157.633191
30th iteration's loglikelihood : -1157.633117
40th iteration's loglikelihood : -1157.633117
50th iteration's loglikelihood : -1157.633117
55th iteration's loglikelihood : -1157.633117
```




```
In [36]: print("estimation of pi1,mu1,sigma1 is",EMparam1 )
         print("estimation of pi2,mu2,sigma2 is",EMparam2 )
         print("log Likelihood is : %f" %EMloglike)
```

```
estimation of pi1,mu1,sigma1 is [ 0.30756656 54.20195015  4.95154337]
estimation of pi2,mu2,sigma2 is [ 0.69243344 80.3644296   7.51168764]
log Likelihood is : -1157.633117
```

So, Gaussian Mixture Model is,

$$X \sim 0.3076 \cdot N(54.2020, 4.9515^2) + 0.6924 \cdot N(80.3644, 7.5117^2)$$

0.4 (d) Which one is better

As log likelihood of a Normal distribution is -1210.556881 from (b) and log likelihood of Gaussion Mixture Model is -1157.633117 from (d), Gaussian Mixture Model has larger loglikelihood. So we can Say that Gaussian Mixture Model is better