EM 算法：

EM 算法拟解决的问题（参数估计问题）：假设随机变量 X 是由 K 个"高斯分布"（理论上也可以是其他分布）混合而成, X 来自各个高斯分布的概率为 $\pi 1, \pi 2 \ldots \pi k$,第 i 个高斯分布的均值为 μi ,方差为 σi 。若观测到随机变量 X 的一系列样本 $x1$,$x2$,...,xn ,试估计参数 π,μ,σ 的值。

举例：随机挑选 10000 位志愿者,测量他们的身高：若样本中存在男性和女性，且男/女身高分别服从 $\mathcal{N}$（$\mu 1,\sigma 1$）和 $\mathcal{N}(\mu 2,\sigma 2)$的高斯分布，试估计 $\mu 1,\sigma 1$ 和 $\mu 2,\sigma 2$

解读：随机变量 X（身高）由 2 个（男/女）“高斯分布”混合而成,样本 X 以 $\pi 1$ 的概率来自男性高斯分布,以 $\pi 2$（即 $1-\pi 1$）的概率来自女性高斯分布，（换言之，如有一样本 x=180cm,它属于男性的身高概率假定为 0.7，则它属于女性身高的概率为 0.3）。我们需要估计的参数是：

1. 样本来自男/女两个高斯分布的“概率（$\pi 1, \pi 2$=？）”分别是多少。
2. 两高斯分布的“均值和方差（$\mu 1,\sigma 1$ 和 $\mu 2,\sigma 2$=？）”分别是多少。

在实际问题当中，上面的身高例子，只给出样本的具体身高值，却没有给出其所属性别，所以性别是属于隐变量，而我们是根据经验推测是性别是影响分布的隐变量。从而假定原样本整体是由男/女两个高斯分布混合而成，接着我们要把这两个高斯分布都分别求出来（求出其两个重要参数：均值和方差）。EM 算法可以看做是寻找隐变量问题。

EM 算法是一种迭代型的算法，在每一次的迭代过程中，主要分为两步：即求期望(Expectation)步骤和最大化(Maximization)步骤。以下假定：样本总数为 N，样本隐含 K 个高斯分布。

1.首先初始化（人为给定）需要估计的参数值 π_k , $\boldsymbol{\mu_k}$, $\boldsymbol{\sigma_k}$ （***K=1,2,3...***）

2.E 步骤：根据高斯分布的“概率公式”计算出每一个样本数据 x_i 是由第 k 个高斯分布产生的概率：

$$\gamma(i,k)=\frac{\pi_k \mathcal{N}(x_i \mid \mu_k,\sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i \mid \mu_j,\sigma_j)}$$

高斯分布概率公式： $\mathcal{N}(x|\ \mu,\sigma)=f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

3.M 步骤：计算第 k 个高斯分布的各个参数

$$N_k=\sum_{i=1}^{N}\gamma(i,k)$$
$$\mu_k=\frac{1}{N_k}\sum_{i=1}^{N}\gamma(i,k)x_i$$
$$\sigma_k=\frac{1}{N_k}\sum_{i=1}^{N}\gamma(i,k)(x_i-\mu_k)(x_i-\mu_k)^T$$
$$\pi_k=\frac{N_k}{N}=\frac{1}{N}\sum_{i=1}^{N}\gamma(i,k)$$

N_k： 每一个样本来自第 k 个高斯分布的概率之和 = 所有样本中来自第 k 个高斯分布的样本数量
μ_k：第 k 个高斯分布的均值
σ_k： 第 k 个高斯分布的方差

π_k：来自第 k 个高斯分布的样本数量占“样本总量”的百分比

4. 将 1 步骤设定的初始参数 π_k，μ_k，σ_k 代入 2 步骤（E 步）计算出 $\gamma(i,k)$，再将其代入 3 步骤（M 步）计算得到新的参数 π_k，μ_k，σ_k 并重新回代入 2 步骤，如此循环迭代 n 次，直到参数 π_k，μ_k，σ_k 保持稳定不变，最终稳定的参数值即为所求值。

Python 代码：

```
from __future__ import division
from numpy import *
import math as mt
#首先生成一些用于测试的样本
#指定两个高斯分布的参数，这两个高斯分布的方差相同
sigma = 6
miu_1 = 40
miu_2 = 20

#随机均匀选择两个高斯分布，用于生成样本值
N = 1000
X = zeros((1, N))
for i in xrange(N):
    if random.random() > 0.5:#使用的是numpy模块中的random
        X[0, i] = random.randn() * sigma + miu_1
    else:
        X[0, i] = random.randn() * sigma + miu_2

#上述步骤已经生成样本
#对生成的样本，使用EM算法计算其均值miu

#取miu的初始值
k = 2
miu = random.random((1, k))
#miu = mat([40.0, 20.0])
Expectations = zeros((N, k))
for step in xrange(1000):#设置迭代次数
    #步骤1，计算期望
    for i in xrange(N):
        #计算分母
        denominator = 0
        for j in xrange(k):
            denominator = denominator + mt.exp(-1 / (2 * sigma ** 2) * (X[0, i] - miu[0, j]) ** 2)

        #计算分子
        for j in xrange(k):
            numerator = mt.exp(-1 / (2 * sigma ** 2) * (X[0, i] - miu[0, j]) ** 2)
            Expectations[i, j] = numerator / denominator

    #步骤2，求期望的最大
    #oldMiu = miu
    oldMiu = zeros((1, k))
    for j in xrange(k):
        oldMiu[0, j] = miu[0, j]
        numerator = 0
        denominator = 0
        for i in xrange(N):
            numerator = numerator + Expectations[i, j] * X[0, i]
            denominator = denominator + Expectations[i, j]
        miu[0, j] = numerator / denominator

    #判断是否满足要求
    epsilon = 0.0001
    if sum(abs(miu - oldMiu)) < epsilon:
        break

    print step
    print miu

print miu
```