CS229 Lecture notes

原作者：[Andrew Ng](http://cs229.stanford.edu/)（[吴恩达](http://open.163.com/movie/2008/1/M/C/M6SGF6VB4_M6SGHFBMC.html)）

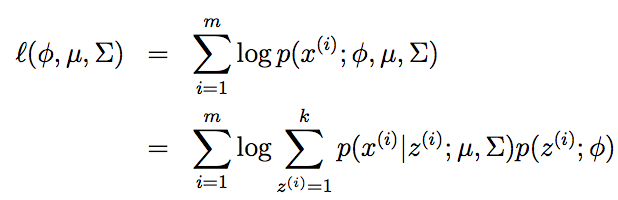
翻译：[CycleUser](https://zhuanlan.zhihu.com/python-kivy)

# 混用高斯 (Gaussians) 和期望最大化算法(the EM algorithm)

在本章讲义中，我们要讲的是使用期望最大化算法（EM，Expectation-Maximization）来进行密度估计（density estimation）。

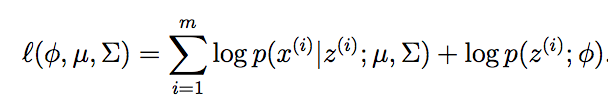
一如既往，还是假设我们得到了某一个训练样本集 {x(1), ... , x(m)}。由于这次是非监督学习（unsupervised learning）环境，所以这些样本就没有什么分类标签了。

我们希望能够获得一个联合分布 p(x(i), z(i)) = p(x(i)|z(i))p(z(i)) 来对数据进行建模。其中的 z(i) ∼ Multinomial(φ) (即 z(i) 是一个以 φ 为参数的多项式分布，其中 φj ≥ 0, Σ kj=1 φj = 1，而参数 φj 给出了 p(z(i) = j))，另外 x(i)|z(i) = j ∼ N(μj,Σj) （一个以 μj 和 Σj 为参数的正态分布）。我们设 k 来表示 z(i) 能取的值的个数。因此，我们这个模型就是在假设每个 x(i) 都是从 z(i) {1, ..., k}中随机选取来生成的，然后 x(i) 就是从一个在 z(i) 上的高斯分布中的 k 个值当中的一个。这就叫做一个混合高斯模型（mixture of Gaussians model）。此外还要注意的就是这里的 z(i) 是潜在的随机变量（latent random variables），这就意味着其取值可能还是隐藏的或者未被观测到的。这就会增加这个估计问题（estimation problem）的难度。

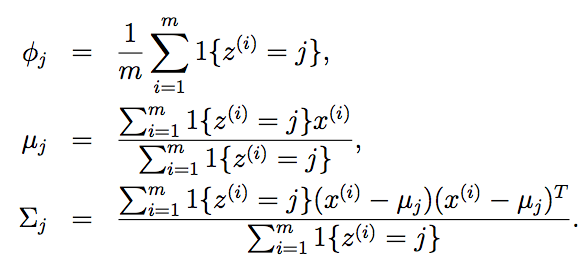
我们这个模型的参数也就是 φ, μ 和 Σ。要对这些值进行估计，我们可以写出数据的似然函数（likelihood）：

然而，如果我们用设上面方程的导数为零来尝试解各个参数，就会发现根本不可能一闭合形式（closed form）来找到这些参数的最大似然估计（maximum likelihood estimates）。（不信的话你自己试试咯。）

随机变量 z(i) 表示着 x(i) 所属于的 k 个高斯分布值。这里要注意，如果我们已知 z(i)，这个最大似然估计问题就简单很多了。那么就可以把似然函数写成下面这种形式：



对上面的函数进行最大化，就能得到对应的参数φ，μ 和 Σ：



事实上，我们已经看到了，如果 z(i) 是已知的，那么这个最大似然估计就几乎等同于之前用高斯判别分析模型（Gaussian discriminant analysis model）中对参数进行的估计，唯一不同在于这里的 z(i) 扮演了高斯判别分析当中的分类标签的角色。

然而，在密度估计问题里面，z(i) 是不知道的。这要怎么办呢？

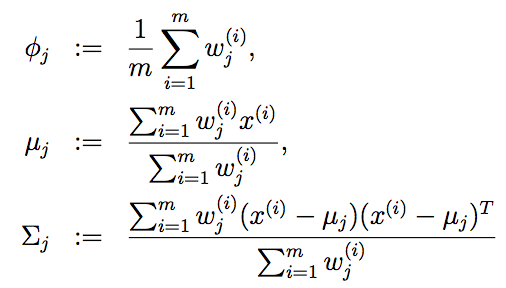
期望最大化算法（EM，Expectation-Maximization）是一个迭代算法，有两个主要的步骤。针对我们这个问题，在 E 这一步中，程序是试图去“猜测（guess）” z(i) 的值。然后在 M 这一步，就根据上一步的猜测来对模型参数进行更新。由于在 M 这一步当中我们假设（pretend）了上一步是对的，那么最大化的过程就简单了。下面是这个算法：

重复下列过程直到收敛（convergence）: {

(E-步骤) 对每个 i, j, 设



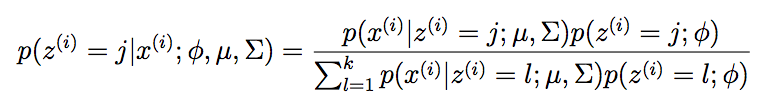
(M-步骤) 更新参数：



}

1这里的式子和之前在 PS1 中高斯判别分析的方程还有一些小的区别，这首先是因为在此处我们把 z(i) 泛化为多项式分布（multinomial），而不是伯努利分布（Bernoulli），其次是由于这里针对高斯分布中的每一项使用了一个不同的 Σj。

在 E 步骤中，在给定 x(i) 以及使用当前参数设置（current setting of our parameters）情况下，我们计算出了参数 z(i) 的后验概率（posterior probability）。使用贝叶斯规则（Bayes rule），就得到下面的式子：



上面的式子中，p(x(i)|z(i) = j;μ,Σ) 是通过评估一个高斯分布的密度得到的，这个高斯分布的均值为 μj，对 x(i) 的协方差为Σj ；p(z(i) = j;φ) 是通过 φj 得到，以此类推。在 E 步骤中计算出来的 wj(i) 代表了我们对 z(i) 这个值的“弱估计（soft guesses）”。另外在 M 步骤中进行的更新还要与 z(i) 已知之后的方程式进行对比。它们是相同的，不同之处只在于之前使用的指示函数（indicator functions），指示每个数据点所属的高斯分布，而这里换成了 wj (i) 。EM 算法也让人想起 K 均值聚类算法，而在 K 均值聚类算法中对聚类重心 c(i) 进行了“强（hard）”赋值，而在 EM 算法中，对 wj(i) 进行的是“弱（soft）”赋值。与 K 均值算法类似，EM 算法也容易导致局部最优，所以使用不同的初始参数（initial parameters）进行重新初始化（reinitializing），可能是个好办法。

很明显，EM 算法对 z(i)’ 进行重复的猜测，这种思路很自然；但这个算法是怎么产生的，以及我们能否确保这个算法的某些特性，例如收敛性之类的？在下一章的讲义中，我们会讲解一种对 EM 算法更泛化的解读，这样我们就可以在其他的估计问题中轻松地使用 EM 算法了，只要这些问题也具有潜在变量（latent variables），并且还能够保证收敛。

2 这里用的词汇“弱（soft）”是指我们对概率进行猜测，从 [0, 1] 这样一个闭区间进行取值；而与之对应的“强（hard）”值得是单次最佳猜测，例如从集合 {0,1} 或者 {1, ..., k} 中取一个值。