概率语言模型及其变形系列

PLSA 及 EM 算法

yangliuyx@gmail.com

12/20/2012

本系列博文介绍常见概率语言模型及其变形模型，主要总结 PLSA、LDA 及 LDA 的变形模型及参数 Inference 方法.

概率语言模型及其变形系列-PLSA 及 EM 算法

yangliuyx@gmail.com

December 20th 2012

本系列博文介绍常见概率语言模型及其变形模型，主要总结 PLSA、LDA 及 LDA 的变形模型及参数 Inference 方法。初步计划内容如下

第一篇：PLSA 及 EM 算法

第二篇：LDA 及 Gibbs Samping

第三篇：LDA 变形模型-Twitter LDA，TimeUserLDA，ATM，Labeled-LDA，MaxEnt-LDA 等

第四篇：基于变形 LDA 的 paper 分类总结

第一篇 PLSA 及 EM 算法

前言：本文主要介绍 PLSA 及 EM 算法，首先给出 LSA（隐性语义分析）的早期方法 SVD，然后引入基于概率的 PLSA 模型，其参数学习采用 EM 算法。接着我们分析如何运用 EM 算法估计一个简单的 mixture unigram 语言模型和混合高斯模型 GMM 的参数，最后总结 EM 算法的一般形式及运用关键点。对于改进 PLSA，引入 hyperparameter 的 LDA 模型及其 Gibbs Sampling 参数估计方法放在本系列后面的文章 LDA 及 Gibbs Samping 介绍。

1 LSA and SVD

LSA(隐性语义分析)的目的是要从文本中发现隐含的语义维度-即“Topic”或者“Concept”。我们知道，在文档的空间向量模型（VSM）中，文档被表示成由特征词出现概率组成的多维向量，这种方法的好处是可以将 query 和文档转化成同一空间下的向量计算相似度，可以对不同词项赋予不同的权重，在文本检索、分类、聚类问题中都得到了广泛应用，在 newsgroup18828 文本分类器的 JAVA 实现和 newsgroup18828 文本聚类器的 JAVA 实现系列文章中的分类聚类算法大多都是采用向量空间模型。然而，向量空间模型没有能力处理一词多义和一义多词问题，例如同义词也分别被表示成独立的一维，计算向量的余弦相似度时会低估用户期望的相似度；而某个词项有多个词义时，始终对应同一维度，因此计算的结果会高估用户期望的相似度。

LSA 方法的引入就可以减轻类似的问题。基于 SVD 分解，我们可以构造一个原始向量矩阵的一个低秩逼近矩阵，具体的做法是将词项文档矩阵做 SVD 分解

$$C = U\Sigma V^T$$

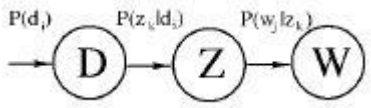
其中C是以词项(terms)为行, 文档(documents)为列做一个大矩阵. 设一共有 t 行 d 列, 矩阵的元素为词项的 tf-idf 值。然后把Σ的 r 个对角元素的前 k 个保留（最大的 k 个保留）, 后面最小的 r-k 个奇异值置 0, 得到Σ_k；最后计算一个近似的分解矩阵

$$C_k = U\Sigma_k V^T$$

则C_k在最小二乘意义下是C的最佳逼近。由于Σ_k最多包含 k 个非零元素，所以C_k的秩不超过 k。通过在 SVD 分解近似，我们将原始的向量转化成一个低维隐含语义空间中，起到了特征降维的作用。每个奇异值对应的是每个“语义”维度的权重，将不太重要的权重置为 0，只保留最重要的维度信息，去掉一些信息“nosie”,因而可以得到文档的一种更优表示形式。将 SVD 分解降维应用到文档聚类的 JAVA 实现可参见此文。

2 PLSA

尽管基于 SVD 的 LSA 取得了一定的成功，但是其缺乏严谨的数理统计基础，而且 SVD 分解非常耗时。Hofmann 在 SIGIR'99 上提出了基于概率统计的 PLSA 模型，并且用 EM 算法学习模型参数。PLSA 的概率图模型如下



其中 D 代表文档，Z 代表隐含类别或者主题，W 为观察到的单词，$P(d_i)$表示单词出现在文档d_i的概率，$P(z_k|d_i)$表示文档d_i中出现主题z_k下的单词的概率，$P(w_j|z_k)$给定主题z_k出现单词w_j的概率。并且每个主题在所有词项上服从 Multinomial 分布，每个文档在所有主题上服从 Multinomial 分布。整个文档的生成过程是这样的：

(1) 以$P(d_i)$的概率选中文档d_i;

(2) 以$P(z_k|d_i)$的概率选中主题z_k;

(3) 以$P(w_j|z_k)$的概率产生一个单词。

我们可以观察到的数据就是(d_i, w_j)对，而z_k是隐含变量。(d_i, w_j)的联合分布为

$$P(d_i, w_j) = P(d_i)P(w_j \mid d_i), \quad P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k)P(z_k \mid d_i).$$

而$P(z_k|d_i)$和$P(w_j|z_k)$分布对应了两组 Multinomial 分布，我们需要估计这两组分布的参数。下面给出用 EM 算法估计 PLSA 参数的详细推导过程。

3 Estimate parameters in PLSA by EM

如文本语言模型的参数估计-最大似然估计、MAP 及贝叶斯估计一文所述，常用的参数估计方法有 MLE、MAP、贝叶斯估计等等。但是在 PLSA 中，如果我们试图直接用 MLE 来估计参数，就会得到似然函数

$$L = \Sigma_{i=1}^{N}\Sigma_{j=1}^{M} n(d_i, w_j)\log p(d_i, w_j) \propto \Sigma_{i=1}^{N}\Sigma_{j=1}^{M} n(d_i, w_j)\log[\Sigma_{k=1}^{K} p(w_j|z_k)p(z_k|d_i)]$$

其中$n(d_i, w_j)$是单词w_j出现在文档d_i中的次数。注意这是一个关于$P(z_k|d_i)$和$P(w_j|z_k)$的函数，一共有 N*K + M*K 个自变量，如果直接对这些自变量求偏导数，我们会发现由于自变量包含在对数和中，这个方程的求解很困难。因此对于这样的包含“隐含变量”或者“缺失数据”的概率模型参数估计问题，我们采用 EM 算法。

EM 算法的步骤是：
(1)E 步骤：求隐含变量 Given 当前估计的参数条件下的后验概率。
(2)M 步骤：最大化 Complete data 对数似然函数的期望，此时我们使用 E 步骤里计算的隐含变量的后验概率，得到新的参数值。
两步迭代进行直到收敛。

先解释一下什么是 Incomplete data 和 complete data。Zhai 老师在一篇经典的 EM 算法 Notes 中讲到，当原始数据的似然函数很复杂时，我们通过增加一些隐含变量来增强我们的数据，得到“complete data”,而“complete data”的似然函数更加简单，方便求极大值。于是，原始的数据就成了“incomplete data”。我们将会看到，我们可以通过最大化“complete data”似然函数的期望来最大化"incomplete data"的似然函数，以便得到求似然函数最大值更为简单的计算途径。

针对我们 PLSA 参数估计问题，在 E 步骤中，直接使用贝叶斯公式计算隐含变量在当前参数取值条件下的后验概率，有

$$P(z_k|d_i, w_j) = \frac{P(w_j|z_k)P(z_k|d_i)}{\sum_{l=1}^{K} P(w_j|z_l)P(z_l|d_i).}$$

在这个步骤中，我们假定所有的$P(z_k|d_i)$和$P(w_j|z_k)$都是已知的，因为初始时随机赋值，后面迭代的过程中取前一轮 M 步骤中得到的参数值。

在 M 步骤中，我们最大化 Complete data 对数似然函数的期望。在 PLSA 中，Incomplete data 是观察到的(d_i, w_j)，隐含变量是主题z_k，那么 complete data 就是三元组(d_i, w_j, z_k)，其期望是

$$\mathbf{E}\left[\mathcal{L}^c\right] = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} P(z_k \mid d_i, w_j) \log\left[P(w_j \mid z_k) P(z_k \mid d_i)\right]$$

注意这里$(z_k|d_i, w_j)$是已知的，取得是前面 E 步骤里面的估计值。下面我们来最大化期望，这又是一个多元函数求极值的问题，可以用拉格朗日乘数法。拉格朗日乘数法可以把条件极值问题转化为无条件极值问题，在 PLSA 中目标函数就是$E(L^c)$,约束条件是

$$\Sigma_{j=1}^{M} p(w_j|z_k) = 1$$
$$\Sigma_{k=1}^{K} p(z_k|d_i) = 1$$

由此我们可以写出拉格朗日函数

$$\mathcal{H} = \mathbf{E}\left[\mathcal{L}^c\right] + \sum_{k=1}^{K} \tau_k \left(1 - \sum_{j=1}^{M} P(w_j \mid z_k)\right) + \sum_{i=1}^{N} \rho_i \left(1 - \sum_{k=1}^{K} P(z_k \mid d_i)\right).$$

这是一个关于$P(z_k|d_i)$和$P(w_j|z_k)$的函数，分别对其求偏导数，我们可以得到

$$\sum_{i=1}^{N} n(d_i, w_j) P(z_k \mid d_i, w_j) - \tau_k P(w_j \mid z_k) = 0, \quad 1 \le j \le M, \quad 1 \le k \le K,$$
$$\sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j) - \rho_i P(z_k \mid d_i) = 0, \quad 1 \le i \le N, \quad 1 \le k \le K.$$

注意这里进行过方程两边同时乘以$P(w_j|z_k)$和$P(z_k|d_i)$的变形，联立上面 4 组方程，我们就可以解出 M 步骤中通过最大化期望估计出的新的参数值

$$P(w_j \mid z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j) P(z_k \mid d_i, w_j)}{\sum_{m=1}^{M} \sum_{i=1}^{N} n(d_i, w_m) P(z_k \mid d_i, w_m)},$$
$$P(z_k \mid d_i) = \frac{\sum_{j=1}^{M} n(d_i, w_j) P(z_k \mid d_i, w_j)}{n(d_i)}.$$

解方程组的关键在于先求出τ_k, ρ_i,其实只需要做一个加和运算就可以把τ_k, ρ_i的系数都化成 1，后面就好计算了。

然后使用更新后的参数值，我们又进入 E 步骤，计算隐含变量z_k Given 当前估计的参数条件下的后验概率。如此不断迭代，直到满足终止条件。

注意到我们在 M 步骤中还是使用对 Complete Data 的 MLE，那么如果我们想加入一些先验知识进入我们的模型，我们可以在 M 步骤中使用 MAP 估计。正如文本语言模型的参数估计-最大似然估计、MAP 及贝叶斯估计中投硬币的二项分布实验中我们加入“硬币一般是两面均匀的”这个先验一样。而由此计算出的参数的估计值会在分子分母中多出关于先验参数的 preduo counts,其他步骤都是一样的。具体可以参考 Mei Qiaozhu 的 Notes。
PLSA 的实现也不难，网上有很多实现 code。

4 Estimate parameters in a simple mixture unigram language model by EM

在 PLSA 的参数估计中，我们使用了 EM 算法。EM 算法经常用来估计包含“缺失数据”或者“隐含变量”模型的参数估计问题。这两个概念是互相联系的，当我们的模型中有“隐含变量”时，我们会认为原始数据是“不完全的数据”，因为隐含变量的值无法观察到；反过来，当我们的数据 incomplete 时，我们可以通过增加隐含变量来对“缺失数据”建模。
为了加深对 EM 算法的理解，下面我们来看如何用 EM 算法来估计一个简单混合 unigram 语言模型的参数。本部分主要参考 Zhai 老师的 EM 算法 Notes。

4.1 最大似然估计与隐含变量引入

所谓 unigram 语言模型，就是构建语言模型是抛弃所有上下文信息，认为一个词出现的概率与其所在位置无关，具体概率图模型可以参见 LDA 及 Gibbs Samping 一文中的介绍。什么是混合模型(mixture model)呢？通俗的说混合概率模型就是由最基本的概率分布比如正态分布、多元分布等经过线性组合形成的新的概率模型，比如混合高斯模型就是由 K 个高斯分布线性组合而得到。混合模型中产生数据的确切“component model”对我们是隐藏的。我们假设混合模型包含两个 multinomial component model,一个是背景词生成模型$p(w|C)$,另一个是主题词生成模型$p(w|\theta_F)$。注意这种模型组成方式在概率语言模型中很常见，比如在 TwitterLDA 中使用的背景词和主题词两个多元分布；TimeUserLDA 中使用的 Global Topic 和 Personal Topic 两个多元分布，都是这类模型。为了表示单词是哪个模型生成的，我们会为每个单词增加一个布尔类型的控制变量。
文档的对数似然函数为

$$\log L(\theta_F) = \log p(\mathcal{F}|\theta_F) = \sum_{i=1}^{k}\sum_{j=1}^{|d_i|}\log((1-\lambda)p(d_{ij}|\theta_F) + \lambda p(d_{ij}|\mathcal{C}))$$

d_{ij}为第 i 个文档中的第 j 个词，λ为表示文档中背景词比例的参数，通常根据经验给定。因此λ是已知的，我们只需要估计$p(w|\theta_F)$即可。
同样的我们首先试图用最大似然估计来估计参数。也就是去找最大化似然函数的参数值，有

$$\begin{aligned}\hat{\theta}_F &= \arg\max_{\theta_F} L(\theta_F) \\ &= \arg\max_{\theta_F} \sum_{i=1}^{k}\sum_{j=1}^{|d_i|} \log((1-\lambda)p(d_{ij}\,|\,\theta_F) + \lambda p(d_{ij}\,|\,\mathcal{C}))\end{aligned}$$

这是一个关于$p(w|\theta_F)$的函数，同样的，$p(w|\theta_F)$包含在了对数和中。因此很难求解极大值，用拉格朗日乘数法，你会发现偏导数等于 0 得到的方程很难求解。所以我们需要依赖数值算法，而 EM 算法就是其中常用的一种。

我们为每个单词引入一个布尔类型的变量 z 表示该单词是 background word 还是 topic word.即

$$z_{ij} = \begin{cases} 1 & \text{if word } d_{ij} \text{ is from background} \\ 0 & \text{otherwise} \end{cases}$$

这里我们假设"complete data"不仅包含可以观察到 F 中的所有单词，而且还包括隐含的变量 z。那么根据 EM 算法，在 E 步骤我们计算“complete data”的对数似然函数有

$$\begin{aligned}L_c(\theta_F) &= \log p(\mathcal{F}, \mathbf{z}\,|\,\theta_F) \\ &= \sum_{i=1}^{k}\sum_{j=1}^{|d_i|}[(1-z_{ij})\log((1-\lambda)p(d_{ij}\,|\,\theta_F)) + z_{ij}\log(\lambda p(d_{ij}\,|\,\mathcal{C}))]\end{aligned}$$

比较一下$L_c(\theta_F)$和$L(\theta_F)$，求和运算在对数之外进行，因为此时通过控制变量 z 的设置，我们明确知道了单词是由背景词分布还是 topic 词分布产生的。$L_c(\theta_F)$和$L(\theta_F)$的关系是怎样的呢？如果带估计参数是θ,原始数据是 X，对于每一个原始数据分配了一个隐含变量 H，则有

$$p(X,H|\theta) = p(H|X,\theta)p(X|\theta).$$

$$L_c(\theta) = \log p(X,H|\theta) = \log p(X|\theta) + \log p(H|X,\theta) = L(\theta) + \log p(H|X,\theta)$$

4.2 似然函数的下界分析

EM 算法的基本思想就是初始随机给定待估计参数的值，然后通过 E 步骤和 M 步骤两步迭代去不断搜索更好的参数值。更好的参数值应该要满足使得似然函数更大。我们假设一个潜在的更好参数值是θ，第 n 次迭代 M 步骤得到的参数估计值是$\theta^{(n)}$,那么两个参数值对应的似然函数和"complete data"的似然函数的差满足

$$L(\theta) - L(\theta^{(n)}) = L_c(\theta) - L_c(\theta^{(n)}) + \log \frac{p(H|X,\theta^{(n)})}{p(H|X,\theta)}$$

我们寻找更好参数值的目标就是要最大化$L(\theta) - L(\theta^{(n)})$,也等价于最大化$L(\theta)$。我们来计算隐含变量在给定当前数据 X 和当前估计的参数值$\theta^{(n)}$条件下的条件概率分布即$p(H|X,\theta^{(n)})$，有

$$L(\theta) - L(\theta^{(n)}) = \sum_H L_c(\theta)p(H|X,\theta^{(n)}) - \sum_H L_c(\theta^{(n)})p(H|X,\theta^{(n)}) + \sum_H p(H|X,\theta^{(n)}) \log \frac{p(H|X,\theta^{(n)})}{p(H|X,\theta)}$$

其中右边第三项是$p(H|X,\theta^{(n)})$和$p(H|X,\theta)$的相对熵，总为非负值。因此我们有

$$L(\theta) - L(\theta^{(n)}) \geq \sum_H L_c(\theta)p(H|X,\theta^{(n)}) - \sum_H L_c(\theta^{(n)})p(H|X,\theta^{(n)})$$

$$L(\theta) \geq \sum_H L_c(\theta)p(H|X,\theta^{(n)}) + L(\theta^{(n)}) - \sum_H L_c(\theta^{(n)})p(H|X,\theta^{(n)})$$

于是我们得到了潜在更好参数值θ的 incomplete data 似然函数的下界。这里我们尤其要注意右边后两项为常数，因为不包含θ。所以 incomplete data 似然函数的下界就是 complete data 似然函数的期望，也就是诸多 EM 算法讲义中出现的 Q 函数，表达式为

$$Q(\theta;\theta^{(n)}) = E_{p(H|X,\theta^{(n)})}[L_c(\theta)] = \sum_H L_c(\theta)p(H|X,\theta^{(n)})$$

可以看出这个期望等于 complete data 似然函数乘以对应隐含变量条件概率再求和。对于我们要求解的问题，Q 函数就是

$$\begin{aligned} Q(\theta_F;\theta_F^{(n)}) &= \sum_{\mathbf{z}} L_c(\theta_F)p(\mathbf{z}|\mathcal{F},\theta_F^{(n)}) \\ &= \sum_{i=1}^{k}\sum_{j=1}^{|d_i|}[p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})\log((1-\lambda)p(d_{ij}|\theta_F)) + p(z_{ij}=1|\mathcal{F},\theta_F^{(n)})\log(\lambda p(d_{ij}|\mathcal{C}))] \end{aligned}$$

这里多解释几句 Q 函数。单词相应的变量 z 为 0 时，单词为 topic word,从多元分布θ_F中产生；当 z 为 1 时，单词为 background word,从多元分布θ产生。同时我们也可以看到如何求 Q 函数即

complete data 似然函数的期望，也就是我们要最大化的那个期望(EM 算法最大化期望指的就是这个期望)，我们要特别关注隐含变量在观察到数据 X 和前一轮估计出的参数值$\theta^{(n)}$条件下取不同值的概率，而隐含变量不同的值对应 complete data 的不同的似然函数，我们要计算的所谓的期望就是指 complete data 的似然函数值在不同隐含变量取值情况下的期望值。

4.3 EM 算法的一般步骤

通过 4.2 部分的分析，我们知道，如果我们在下一轮迭代中可以找到一个更好的参数值$\theta^{(n+1)}$使得

$$Q(\theta^{(n+1)};\theta^{(n)}) > Q(\theta^{(n)};\theta^{(n)}).$$

那么相应的也会有$L(\theta^{(n+1)}) > L(\theta^{(n)})$，因此 EM 算法的一般步骤如下

(1) 随机初始化参数值$\theta^{(0)}$，也可以根据任何关于最佳参数取值范围的先验知识来初始化$\theta^{(0)}$。

(2) 不断两步迭代寻找更优的参数值$\theta^{(n+1)}$：

(a) E 步骤（求期望） 计算 Q 函数

$$Q(\theta;\theta^{(n)}) = E_{p(H|X,\theta^{(n)})}[L_c(\theta)] = \sum_H L_c(\theta)p(H|X,\theta^{(n)})$$

(b)M 步骤（最大化）通过最大化 Q 函数来寻找更优的参数值$\theta^{(n+1)}$

$$\theta^{(n+1)} = argmax_\theta Q(\theta;\theta^{(n)})$$

(3) 当似然函数$L(\theta)$收敛时算法停止。

这里需要注意如何尽量保证 EM 算法可以找到全局最优解而不是局部最优解呢？第一种方法是尝试许多不同的参数初始值，然后从得到的很多估计出的参数值中选取最优的；第二种方法是通过一个更简单的模型比如只有唯一全局最大值的模型来决定复杂模型的初始值。

通过前面的分析可以知道，EM 算法的优势在于 complete data 的似然函数$L_c(\theta)$更容易最大化，因为已经假定了隐含变量的取值，当然要乘以隐含变量取该值的条件概率，所以最终变成了最大化期望值。由于隐含变量变成了已知量，Q 函数比原始 incomplete data 的似然函数更容易求最大值。因此对于“缺失数据”的情况，我们通过引入隐含变量使得 complete data 的似然函数容易最大化。

在 E 步骤中，主要的计算难点在于计算隐含变量的条件概率$p(H|X,\theta^{(n)})$，在 PLSA 中就是

$$P(z_k|d_i,w_j)=\frac{P(w_j|z_k)P(z_k|d_i)}{\sum_{l=1}^{K}P(w_j|z_l)P(z_l|d_i).}$$

在我们这个简单混合语言模型的例子中就是

$$p(z_{ij}=1|\mathcal{F},\theta_F^{(n)}) = \frac{\lambda p(d_{ij}|C)}{\lambda p(d_{ij}|C)+(1-\lambda)p(d_{ij}|\theta_F^{(n)})}$$

$$p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})=1-p(z_{ij}=1|\mathcal{F},\theta_F^{(n)})$$

我们假设 z 的取值只于当前那一个单词有关，计算很容易，但是在 LDA 中用这种方法计算隐含变量的条件概率和最大化 Q 函数就比较复杂，可以参见原始 LDA 论文的参数推导部分。我们也可以用更简单的 Gibbs Sampling 来估计参数，具体可以参见 LDA 及 Gibbs Samping。

继续我们的问题，下面便是 M 步骤。使用拉格朗日乘数法来求 Q 函数的最大值，约束条件是

$$\sum_{w\in V}p(w|\theta_F)=1$$

构造拉格朗日辅助函数

$$g(\theta_F)=Q(\theta_F;\theta_F^{(n)})+\mu(1-\sum_{w\in V}p(w|\theta_F))$$

对自变量$p(w|\theta_F)$求偏导数

$$\frac{\partial g(\theta_F)}{\partial p(w|\theta_F)} = [\sum_{i=1}^{k}\sum_{j=1,d_{ij}=w}^{|d_i|}\frac{p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})}{p(w|\theta_F)}]-\mu$$

令偏导数为 0 解出来唯一的极值点

$$p(w|\theta_F) = \frac{\sum_{i=1}^{k}\sum_{j=1,d_{ij}=w}^{|d_i|} p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})}{\sum_{i=1}^{k}\sum_{j=1}^{|d_i|} p(z_{ij}=0|\mathcal{F},\theta_F^{(n)})} = \frac{\sum_{i=1}^{k} p(z_w=0|\mathcal{F},\theta_F^{(n)})c(w,d_i)}{\sum_{i=1}^{k}\sum_{w\in V} p(z_w=0|\mathcal{F},\theta_F^{(n)})c(w,d_i)}$$

容易知道这里唯一的极值点就是最值点了。注意这里 Zhai 老师变换了一下变量表示，把对文档里面词的遍历转化成了对词典里面的 term 的遍历，因为 z 的取值至于对应的那一个单词有关，与上下文无关。因此 E 步骤求隐含变量的条件概率公式也相应变成了

$$p(z_w=1|\mathcal{F},\theta_F^{(n)}) = \frac{\lambda p(w|C)}{\lambda p(w|C)+(1-\lambda)p(w|\theta_F^{(n)})}$$

最后我们就得到了简单混合 Unigram 语言模型的 EM 算法更新公式
即 E 步骤 求隐含变量条件概率和 M 步骤 最大化期望估计参数的公式

$$p(z_w=1|\mathcal{F},\theta_F^{(n)}) = \frac{\lambda p(w|C)}{\lambda p(w|C)+(1-\lambda)p(w|\theta_F^{(n)})} \quad \text{E-step}$$

$$p(w|\theta_F^{(n+1)}) = \frac{\sum_{i=1}^{k}(1-p(z_w=1|\mathcal{F},\theta_F^{(n)}))c(w,d_i)}{\sum_{i=1}^{k}\sum_{w\in V}(1-p(z_w=1|\mathcal{F},\theta_F^{(n)})c(w,d_i))} \quad \text{M-step}$$

整个计算过程我们可以看到，我们不需要明确求出 Q 函数的表达式。取而代之的是我们计算隐含变量的条件概率，然后通过最大化 Q 函数来得到新的参数估计值。
因此 EM 算法两步迭代的过程实质是在寻找更好的待估计参数的值使得原始数据即 incomplete data 似然函数的下界不断提升，而这个“下界“就是引入隐含变量之后的 complete data 似然函数的期望，也就是诸多 EM 算法讲义中出现的 Q 函数，通过最大化 Q 函数来寻找更优的参数值。同时，上一轮估计出的参数值会在下一轮 E 步骤中当成已知条件计算隐含变量的条件概率，而这个条件概率又是最大化 Q 函数求新的参数值是所必需的。

5 Estimate parameters in GMM by EM

经过第 3 部分和第 4 部分用 EM 算法求解 PLSA 和简单 unigram 混合模型参数估计问题的详细分析，相信大部分读者已经对 EM 算法有了一定理解。关于 EM 算法的材料包括 PRML 会首先介绍用 EM 算法去求解混合高斯模型 GMM 的参数估计问题。下面就让我们来看看如何用 EM 算法来求解混合高斯模型 GMM。

混合高斯模型 GMM 由 K 个高斯模型的线性组合组成，高斯模型就是正态分布模型，其中每个高斯模型我们成为一个”Component“，GMM 的概率密度函数就是这 K 个高斯模型概率密度函数的线性组合即

$$p(\mathrm{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x}|\mu_k, \Sigma_k).$$

其中

$$\mathcal{N}(\mathrm{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathrm{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathrm{x}-\mu)\right\}$$

就是高斯分布即正态分布的概率密度函数。这是 x 为向量的情况，对于 x 为标量的情况就是

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

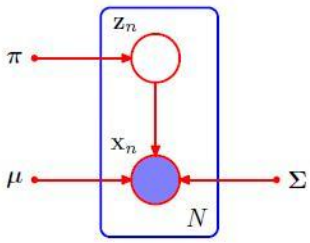
大部分读者应该对标量情形的概率分布更熟悉。这里啰嗦几句，最近看机器学习的论文和书籍，里面的随机变量基本都是多维向量，向量的计算比如加减乘除和求导运算都和标量运算有一些区别，尤其是求导运算，向量和矩阵的求导运算会麻烦很多，看 pluskid 推荐的一本册子 Matrix Cookbook,里面有很多矩阵求导公式，直接查阅应该会更方便。

下面继续说 GMM。根据上面给出的概率密度函数，如果我们要从 GMM 的分布中 Sample 一个样本，实际上可以分为两步：首先随机地在这 K 个 Component 之中选一个，每个 Component 被选中的概率实际上就是它的系数 π_k，选中了 Component 之后，再单独地考虑从这个 Component 的分布中选取一个样本点就可以了。在 PRML 上，引入了一个 K 维二值随机变量 z,只有 1 维是 1，其他维都是 0。唯一那个非零的维对应的就是 GMM 参数样本时被选中的那个高斯分布，而某一维非零的概率就是π_k，即

$$p(z_k = 1) = \pi_k$$

下面我们开始估计 GMM 的参数，包括这 K 个高斯分布的所有均值和方差以及线性组合的系数。我们给每个样本数据增加一个隐含变量z_n, 就是上面所说的 K 维向量，表明了x_n是从哪个高斯分布中 sample 出来的。对应的概率图模型就是

Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{\mathbf{x}_n\}$, with corresponding latent points $\{\mathbf{z}_n\}$, where $n = 1, \ldots, N$.



观察变量的对数似然函数为

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

令对μ_k的偏导数等于 0 我们有

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

注意这里我们定义了$\gamma(z_{nk})$表示后验概率$p(z_k = 1|x)$，也就是第 n 个样本是有第 k 个高斯分布产生的概率。可以解出

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

N_k就是由第 K 个高斯分布产生的样本点的总数；用聚类的观点看，就是聚到 cluster k 的样本点总数。然后我们将对数似然函数对Σ_k求偏导数，令偏导数为 0，得到协方差矩阵

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

最后我们求系数π_k。注意到系数的和为 1，即

$$\sum_{k=1}^{K} \pi_k = 1$$

这就是约束条件，最大化对数似然函数又成为了条件极值问题。我们仍然用拉格朗日乘数法，构造辅助函数如下

$$\ln p(\mathbf{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) + \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)$$

对π_k求导数，令导数为 0 有

$$0 = \sum_{n=1}^{N}\frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)} + \lambda \qquad \pi_k = \frac{N_k}{N}$$

这样我们就估计出来系数项。

因此用 EM 算法估计 GMM 参数的步骤如下

(1) E 步骤：估计数据由每个 Component 生成的概率:对于每个数据x_n 来说，它由第 k 个 Component 生成的概率为

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}.$$

注意里面 μ_k 和 Σ_k 也是需要我们估计的值，在 E 步骤我们假定 μ_k 和 Σ_k 均已知，我们使用上一次迭代所得的值（或者初始值）。

(2)M 步骤：由最大估计求出高斯分布的所有均值、方差和线性组合的系数，更新待估计的参数值，根据上面的推导，计算公式是

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

其中

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

(3)重复迭代 E 步骤和 M 步骤，直到似然函数

$$\ln p(\mathbf{X}|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right\}$$

收敛时算法停止。

更多关于 EM 算法的深入分析，可以参考 PRML 第 9 章内容。

最后我们给出用 EM 算法估计 GMM 参数的 Matlab 实现，作者 pluskid 的博客

```
function varargout = gmm(X, K_or_centroids)
% ================================================================
% Expectation-Maximization iteration implementation of
% Gaussian Mixture Model.
%
% PX = GMM(X, K_OR_CENTROIDS)
% [PX MODEL] = GMM(X, K_OR_CENTROIDS)
%
%  - X: N-by-D data matrix.
%  - K_OR_CENTROIDS: either K indicating the number of
%       components or a K-by-D matrix indicating the
%       choosing of the initial K centroids.
%
%  - PX: N-by-K matrix indicating the probability of each
%       component generating each point.
%  - MODEL: a structure containing the parameters for a GMM:
%       MODEL.Miu: a K-by-D matrix.
%       MODEL.Sigma: a D-by-D-by-K matrix.
%       MODEL.Pi: a 1-by-K vector.
% ================================================================

    threshold = 1e-15;
    [N, D] = size(X);

    if isscalar(K_or_centroids)
        K = K_or_centroids;
        % randomly pick centroids
        rndp = randperm(N);
        centroids = X(rndp(1:K), :);
    else
        K = size(K_or_centroids, 1);
        centroids = K_or_centroids;
    end

    % initial values
    [pMiu pPi pSigma] = init_params();

```

```
    Lprev = -inf;
    while true
        Px = calc_prob();

        % new value for pGamma
        pGamma = Px .* repmat(pPi, N, 1);
        pGamma = pGamma ./ repmat(sum(pGamma, 2), 1, K);

        % new value for parameters of each Component
        Nk = sum(pGamma, 1);
        pMiu = diag(1./Nk) * pGamma' * X;
        pPi = Nk/N;
        for kk = 1:K
            Xshift = X-repmat(pMiu(kk, :), N, 1);
            pSigma(:, :, kk) = (Xshift' * ...
                (diag(pGamma(:, kk)) * Xshift)) / Nk(kk);
        end

        % check for convergence
        L = sum(log(Px*pPi'));
        if L-Lprev < threshold
            break;
        end
        Lprev = L;
    end

    if nargout == 1
        varargout = {Px};
    else
        model = [];
        model.Miu = pMiu;
        model.Sigma = pSigma;
        model.Pi = pPi;
        varargout = {Px, model};
    end

    function [pMiu pPi pSigma] = init_params()
        pMiu = centroids;
        pPi = zeros(1, K);
        pSigma = zeros(D, D, K);

        % hard assign x to each centroids
        distmat = repmat(sum(X.*X, 2), 1, K) + ...
            repmat(sum(pMiu.*pMiu, 2)', N, 1) - ...
            2*X*pMiu';
        [dummy labels] = min(distmat, [], 2);

        for k=1:K
            Xk = X(labels == k, :);
            pPi(k) = size(Xk, 1)/N;
            pSigma(:, :, k) = cov(Xk);
```

```
        end
    end

    function Px = calc_prob()
        Px = zeros(N, K);
        for k = 1:K
            Xshift = X-repmat(pMiu(k, :), N, 1);
            inv_pSigma = inv(pSigma(:, :, k));
            tmp = sum((Xshift*inv_pSigma) .* Xshift, 2);
            coef = (2*pi)^(-D/2) * sqrt(det(inv_pSigma));
            Px(:, k) = coef * exp(-0.5*tmp);
               end
           end
       end
```

6 全文总结

本文主要介绍 PLSA 及 EM 算法，首先给出 LSA（隐性语义分析）的早期方法 SVD，然后引入基于概率的 PLSA 模型，接着我们详细分析了如何用 EM 算法估计 PLSA、混合 unigram 语言模型及混合高斯模型的参数过程，并总结了 EM 算法的一般形式和运用关键点。关于 EM 算法收敛性的证明可以参考斯坦福机器学习课程 CS229 Andrew Ng 老师的课程 notes 和 JerryLead 的笔记。EM 算法在”缺失数据“和包含”隐含变量“的概率模型参数估计问题中非常常用，是机器学习、数据挖掘及 NLP 研究必须掌握的算法。

参考文献及推荐 Notes

本文主要参考了 Hoffman 的 PLSA 论文、Zhai 老师的 EM Notes 以及 PRML 第 9 章内容。

[1] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] Gregor Heinrich. Parameter estimation for text analysis. Technical report, 2004.

[3] Wayne Xin Zhao, Note for pLSA and LDA, Technical report, 2011.

[4] Freddy Chong Tat Chua. Dimensionality reduction and clustering of text documents.Technical report, 2009.

[5] CX Zhai, A note on the expectation-maximization (em) algorithm 2007

[6] Qiaozhu Mei, A Note on EM Algorithm for Probabilistic Latent Semantic Analysis 2008

[7] pluskid, 漫谈 Clustering, Gaussina Mixture Model

[8] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.