

人工智能之机器学习

主题模型

上海育创网络科技有限公司

主讲人：刘老师(GerryLiu)

课程要求

- 课上课下 “九字” 真言
 - 认真听，**善摘录，勤思考**
 - **多温故，乐实践**，再发散
- 四不原则
 - **不懒散惰性，不迟到早退**
 - **不请假旷课，不拖延作业**
- 一点注意事项
 - 违反 “四不原则”，不推荐就业

课程内容

- 主题模型
- LSA
- LDA

主题模型

- 传统判断两个文档相似性的方法是通过查看两个文档共同出现的单词的多少，如TF-IDF等，这种方法没有考虑到文字背后的语义关联，可能在两个文档共同出现的单词很少甚至没有，但两个文档是相似的。
- 举个例子，有两个句子分别如下：
 - “乔布斯离我们而去了，苹果公司可能会发生动荡。”
 - “苹果手机价格会不会降？”

主题模型

- 主题模型(Topic Model)是用来在一系列文档中发现**抽象主题**的一种统计模型。直观来讲，如果一篇文章有一个中心思想，那么一定存在一些特定词语会出现的比较频繁。比方说，如果现在一篇文章是在讲苹果公司的，那么“乔布斯”和“iPhone”等词语出现的频率会更高一些；如果现在一篇文章是在描述微软公司的，那么“Windows”和“Microsoft”等词语出现的频率会更高一些；但真实情况下，一篇文章中通常包含多种主题，而且每个主题所占的比例各不相同，比如一篇文章中10%和苹果公司有关，90%和微软公司有关，那么和微软有关的关键字出现的次数应该是苹果关键字出现次数的9倍。
- 主题模型就是一种自动分析每个文档，统计文档内词语，根据统计的信息判断当前文档包含哪些主题以及各个主题所占比例各为多少。

主题模型

- 主题模型是对文本中隐含主题的一种建模方法，每个主题其实是词表上单词的概率分布；
- 主题模型是一种生成模型，一篇文章中每个词都是通过“以一定概率选择某个主题，并从这个主题中以一定概率选择某个词语”这样一个过程得到的；

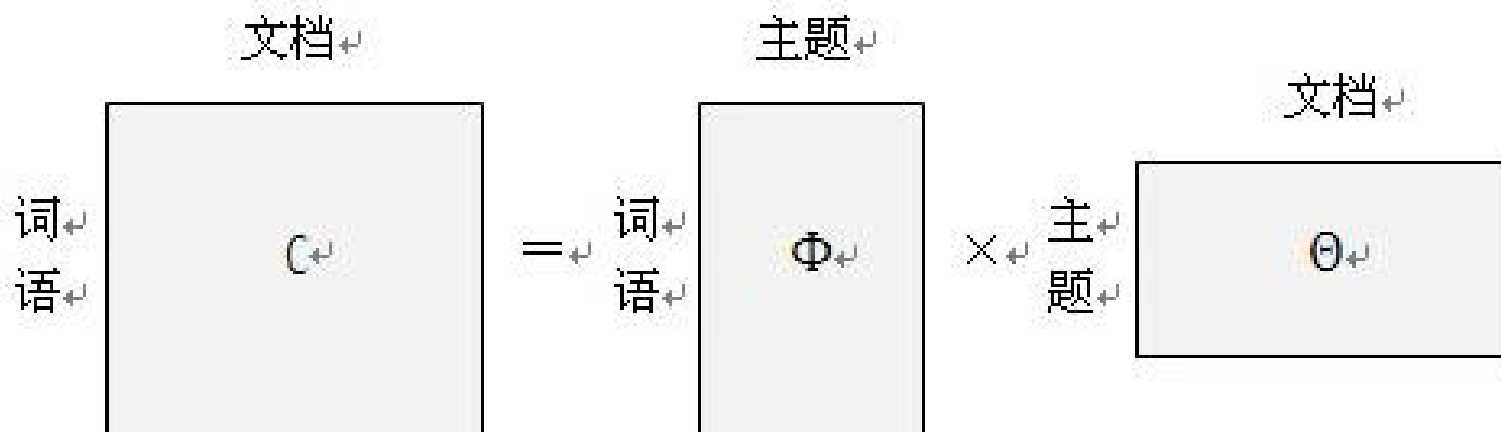
$$p(word|doc) = \sum_{topic} p(topic|doc) * p(word|topic)$$

- 主题模型克服了传统信息检索中文档相似度计算方法的缺点，并且能够在海量的数据中找出文字间的语义主题。主题模型在自然语言和给予文本的搜索上起到了重要的作用。

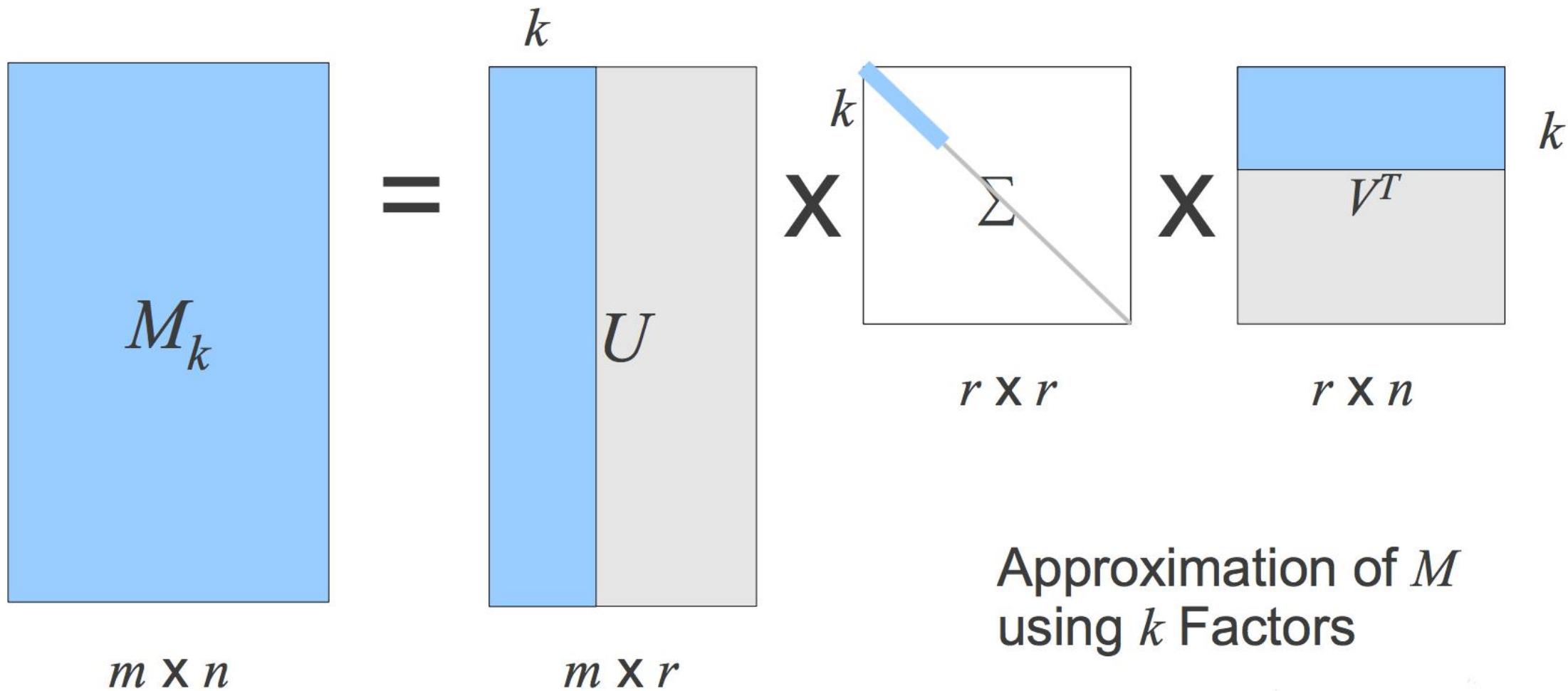
主题模型

- 怎样才能生成主题？对文章的主题应该怎么分析？这是主题模型要解决的问题。

$$p(\text{词语} | \text{文档}) = \sum_{\text{主题}} p(\text{词语} | \text{主题}) \times p(\text{主题} | \text{文档})$$



SVD



SVD的特性

- 对于奇异值,它跟我们特征分解中的特征值类似, 在奇异值矩阵中也是按照从大到小排列, 而且奇异值的减少特别的快, 在很多情况下, 前10%甚至1%的奇异值的和就占了全部的奇异值之和的99%以上的比例。也就是说, 我们也可以用最大的k个的奇异值和对应的左右奇异向量来近似描述矩阵。

$$A_{m*n} = U_{m*m} \Sigma_{m*n} V_{n*n}^T \approx U_{m*k} \Sigma_{k*k} V_{n*k}^T$$

- 由于可以使用小矩阵来近似的描述样本信息的这个重要特性, SVD可以常用于PCA降维、推荐系统以及主题模型等场景中。

- 潜在语义分析(Latent Semantic Analysis, LSA), 也叫做Latent Semantic Indexing, LSI. 是一种常用的简单的主题模型。LSA是基于奇异值分解(SVD)的方法得到文本主题的一种方式。

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T \quad A_{m \times n} \approx U_{m \times k} \Sigma_{k \times k} V_{n \times k}^T$$

- 总结: 我们输入的有m个文本, 每个文本有n个词。而 A_{ij} 则对应第i个文本的第j个词的特征值。k是我们假设的主题数, 一般要比文本数少。SVD分解后, U_{il} 对应第i个文本和第l个主题的相关度。 V_{jm} 对应第j个词和第m个词义的相关度。 Σ_{lm} 对应第l个主题和第m个词义的相关度。

- 假设有11个词、3个文本对应的词频TF矩阵如下：

Terms ↓	d1 ↓	d2 ↓	d3 ↓	q ↓
a	1	1	1	0
arrived	0	1	1	0
damaged	1	0	0	0
delivery	0	1	0	0
fire	1	0	0	0
gold	1	0	1	1
in	1	1	1	0
of	1	1	1	0
shipment	1	0	1	0
silver	0	2	0	1
truck	0	1	1	1

- 假定主题数为2，通过SVD降维后的三个矩阵分布为：

$$\begin{aligned} \mathbf{U} \approx \mathbf{U}_k &= \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} & \mathbf{\Sigma} \approx \mathbf{\Sigma}_k &= \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix} & k = 2 \\ \mathbf{V} \approx \mathbf{V}_k &= \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} & \mathbf{V}^T \approx \mathbf{V}_k^T &= \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix} \end{aligned}$$

- 通过SVD矩阵分解我们可以得到文本、词与主题、语义之间的相关性，但是这个时候计算出来的内容存在负数，我们比较难解释，所以我们可以对LSI得到文本主题矩阵使用余弦相似度计算文本的相似度的计算。最终我们得到第一个和第三个文档比较相似，和第二个文档不太相似。(备注：这个时候直接在文本主题矩阵的基础上直接应用聚类算法即可)

$$\text{sim}(d_1, d_2) = \frac{(-0.4945) * (-0.6458) + 0.6492 * (-0.7194)}{\sqrt{(-0.4945)^2 + 0.6492^2} * \sqrt{(-0.6458)^2 + (-0.7194)^2}} = -0.1872$$

$$\text{sim}(d_1, d_3) = \frac{(-0.4945) * (-0.5817) + 0.6492 * 0.2469}{\sqrt{(-0.4945)^2 + 0.6492^2} * \sqrt{(-0.5817)^2 + 0.2469^2}} = 0.8686$$

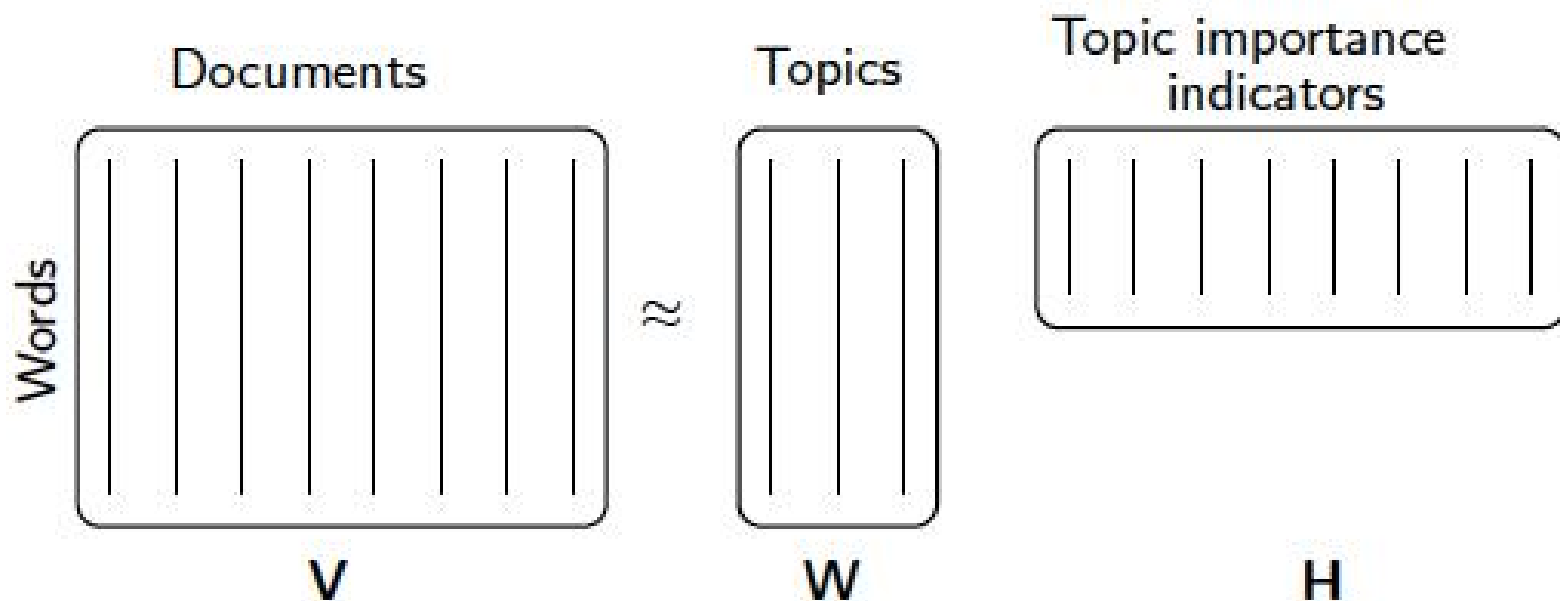
LSA主题模型总结

- 除非数据规模比较小，而且希望快速的粗粒度的找出一些主题分布关系，否则我们一般不会使用LSA主题模型。
- 优点：
 - 原理简单，一次SVD分解即可得到主题模型，可以同时解决词义的问题。
- 缺点：
 - SVD分解的计算非常耗时，对于高维度矩阵做SVD分解非常困难；
 - 主题模型数量的选取对于结果的影响非常大，很难选择合适的k值；
 - LSA模型不是概率模型，缺乏统计基础，结果难以直观的解释。

- 非负矩阵分解(Non-negative Matrix Factorization, NMF)是一种常用的矩阵分解方式，常用于矩阵分解、降维、主题模型等应用场景。
- NMF虽然和SVD一样都是矩阵分解，但是NMF不同的是：它的目标希望是将矩阵分解成为两个子矩阵，两个子矩阵中所有的值均大于等于0。
- 参考： <https://www.csie.ntu.edu.tw/~cjlin/papers/pgradnmf.pdf>

$$V_{m*n} \approx W_{m*k} H_{k*n}$$

- 在NMF中求解出来的W和H，分别体现的是文本和主题的概率相关度，以及词和主题的概率相关度；



- NMF的期望是找到两个W、H矩阵，使得WH的矩阵乘积结果和对应的原矩阵V对应位置的值相比误差尽可能的小。

$$\min_{W,H} f(W,H) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m (V_{ij} - (WH)_{ij})^2$$

$$\text{subject to : } W_{ia} \geq 0, H_{bj} \geq 0, \forall i, a, b, j$$

$$\sum_{i=1}^n \sum_{j=1}^m (V_{ij} - (WH)_{ij})^2 = \|V - WH\|_2^2$$

- NMF的目标函数中总共包含了 $m*k+k*n$ 个参数，可以直接使用梯度下降法或者拟牛顿法来进行求解。

$$W^{k+1} = \max\left(0, W^k - \alpha \nabla_W f(W^k, H^k)\right)$$

$$H^{k+1} = \max\left(0, H^k - \alpha \nabla_H f(W^k, H^k)\right)$$

NMF

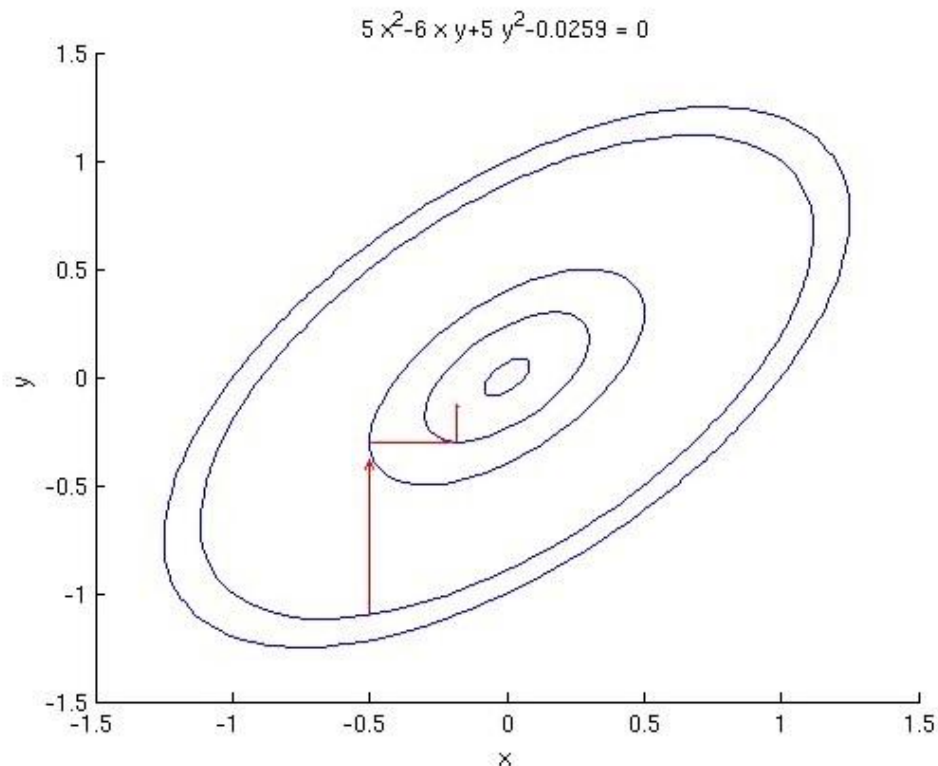
- 为了防止过拟合，也可以在NMF的目标函数的基础上添加一个正则化项

$$\frac{1}{2} \|V - WH\|_2^2 + \alpha \rho \|W\|_1 + \alpha \rho \|H\|_1 + \frac{\alpha(1-\rho)}{2} \|W\|_2^2 + \frac{\alpha(1-\rho)}{2} \|H\|_2^2$$

- 但是当加入L1正则项后，由于没法求解出正常的导函数出来(导函数不是连续的)，也就没法使用梯度下降法和拟牛顿法求解参数，此时一般采用坐标轴下降法来进行参数的求解。

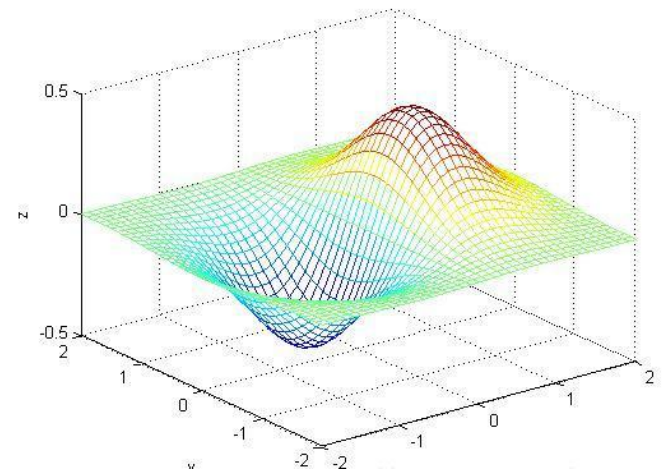
坐标轴下降法

- 坐标轴下降法(Coordinate Descent, CD)是一种迭代法, 通过启发式的方法一步步的迭代求解函数的最小值, 和梯度下降法(GD)不同的时候, 坐标轴下降法是沿着坐标轴的方向去下降, 而不是采用梯度的负方向下降。



坐标轴下降法

- 坐标轴下降法利用EM算法的思想，在参数更新过程中，每次均先固定 $m-1$ 个参数值，求解剩下的一个参数的局部最优解；然后进行迭代式的更新操作。
- 坐标轴下降法的核心思想是多变量函数 $F(X)$ 可以通过每次沿着一个方向优化来获取最小值；其数学依据是：对于一个可微凸函数 $f(\theta)$ ，其中 θ 为 $n \times 1$ 的向量，如果对于一个解 $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ ，使得 $f(\theta)$ 在每一个坐标轴 $\theta_i (i=1, 2, \dots, n)$ 上都能达到最小值，则 $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ 就是的 $f(\theta)$ 全局的最小值点



坐标轴下降法

- 在坐标轴下降法中，优化方向从算法的一开始就固定了，即沿着坐标的方向进行变化。在算法中，循环最小化各个坐标方向的目标函数。即：如果 x^k 给定，那么 x^{k+1} 的第 i 维度为：

$$X_i^{k+1} = \arg \min_{y \in R} f(x_1^{k+1}, \dots, x_{i-1}^{k+1}, y, x_{i+1}^k, \dots, x_n^k)$$

- 因此，从一个初始的 x_0 求得函数 $F(x)$ 的局部最优解，可以迭代获取 x_0 、 x_1 、 $x_2 \dots$ 的序列，从而可以得到：

$$F(X^0) \geq F(X^1) \geq F(X^2) \geq \dots$$

坐标轴下降法算法过程

- 1. 给 θ 向量随机选取一个初值，记做 θ^0 ;
- 2. 对于第 k 轮的迭代，从 θ_1^k 开始计算， θ_n^k 到为止，计算公式如下：

$$\theta_1^k = \arg \min_{\theta_1} J(\theta_1, \theta_2^{k-1}, \theta_3^{k-1}, \dots, \theta_n^{k-1})$$

$$\theta_2^k = \arg \min_{\theta_2} J(\theta_1^k, \theta_2, \theta_3^{k-1}, \dots, \theta_n^{k-1})$$

.....

$$\theta_n^k = \arg \min_{\theta_n} J(\theta_1^k, \theta_2^k, \theta_3^k, \dots, \theta_n)$$

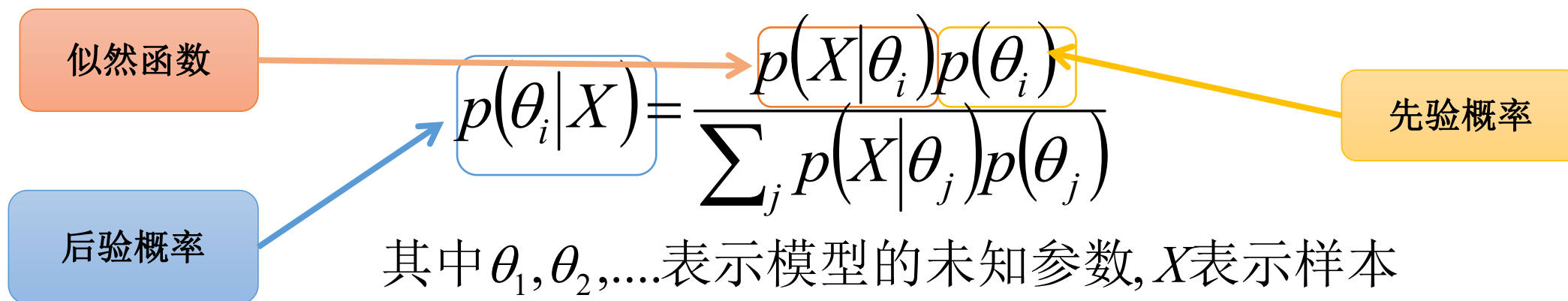
- 检查 θ^k 和 θ^{k-1} 向量在各个维度上的变化情况，如果所有维度的变化情况都比较小的话，那么认为结束迭代，否则继续 $k+1$ 轮的迭代。或者以函数 J 的变化值作为收敛条件。
- 备注：在求解每个参数局部最优解的时候可以求导的方式来求解。

坐标轴下降法和梯度下降法的区别

- 坐标轴下降法在每次迭代中，计算当前点处沿一个坐标方向进行一维搜索，固定其它维度的坐标方向，找到一个函数的局部极小值。而梯度下降总是沿着梯度的负方向求函数的局部最小值；
- 坐标轴下降优化方法是一种非梯度优化算法。在整个过程中依次循环使用不同的坐标方向进行迭代，一个周期的一维搜索迭代过程相当于一个梯度下降的迭代；
- 梯度下降是利用目标函数的导数来确定搜索方向的，该梯度方向可能不与任何坐标轴平行。而坐标轴下降法是利用当前坐标方向进行搜索，不要求目标函数的导数，只按照某一坐标方向进行搜索最小值；
- 两者都是迭代算法，且每一轮迭代都需要 $O(mn)$ 的计算量(m 为样本数， n 为维度数)

概率知识回顾

- **先验概率**：在事情尚未发生前，对该事件发生概率的估计。利用过去历史资料计算出来得到的先验概率叫做客观先验概率；凭主观经验来判断而得到的先验概率叫做主观先验概率。
- **后验概率**：通过调查或其它方式获取新的附加信息，利用贝叶斯公式对先验概率进行修正后，而得到的概率。
- **似然函数**：给定模型参数 θ 的条件下，样本数据服从这一概率模型的相似程度。



图中展示了贝叶斯公式及其各部分的标注：

$$p(\theta_i|X) = \frac{p(X|\theta_i)p(\theta_i)}{\sum_j p(X|\theta_j)p(\theta_j)}$$

标注说明：

- 似然函数：指向分子中的 $p(X|\theta_i)$
- 后验概率：指向整个公式 $p(\theta_i|X)$
- 先验概率：指向分子中的 $p(\theta_i)$

其中 $\theta_1, \theta_2, \dots$ 表示模型的未知参数, X 表示样本

概率知识回顾

- **先验分布**：反映在进行统计试验之前根据其他有关参数知识得到的分布；也就是说在观测获取样本之前，人们对 θ 已经有一些知识，此时这个 θ 的分布函数为 $H(\theta)$ ， θ 的密度函数为 $h(\theta)$ ，分别称为先验分布函数和先验密度函数，统称先验分布。
- **后验分布**：根据样本 X 的分布以及 θ 的先验分布 $\pi(\theta)$ ，使用概率论中求解条件概率的方式可以计算出来已知 X 的条件下， θ 的条件分布 $\pi(\theta|x)$ 。因为该分布是在获取样本 x 之后计算出来的，所以称为后验分布。
- **共轭分布**：如果先验分布和后验分布具有相同的形式，那么先验分布和似然函数被称为共轭分布。

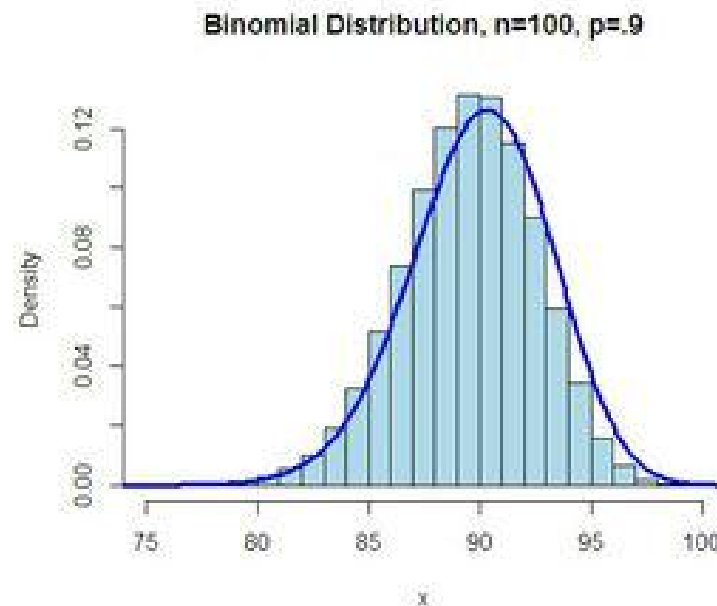
二项分布

- 二项分布是从伯努利分布推导过来的。伯努利分布，又称两点分布或0-1分布，是一个离散型的随机分布，其中的随机变量只有两类取值，非正即负{+, -}。而二项分布即重复n次的伯努利试验，记为 $X \sim b(n, p)$ 。简言之，只做一次实验，是伯努利分布，重复做了n次，是二项分布。二项分布的概率密度函数为：

$$p(K = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$E(x) = np$$

$$D(x) = np(1-p)$$



多项分布

- 多项分布(Multinomial Distribution)是二项分布的推广。
- 多项分布是指单次试验中的随机变量的取值不再是0/1的，而是有多种离散值可能 (1,2,3...,k) 。比如投掷6个面的骰子实验，N次实验结果服从K=6的多项分布。其中K个离散值的概率为：

$$\sum_{i=1}^k p_i = 1$$

- 多项分布的概率密度函数为：

$$X = (x_1, x_2, \dots, x_k) \quad \sum_{i=1}^k m_i = n$$

$$p(x_1 = m_1, x_2 = m_2, \dots, x_k = m_k, n, p_1, p_2, \dots, p_k) = \frac{n!}{m_1! m_2! \dots m_k!} p_1^{m_1} p_2^{m_2} \dots p_k^{m_k}$$

Beta分布

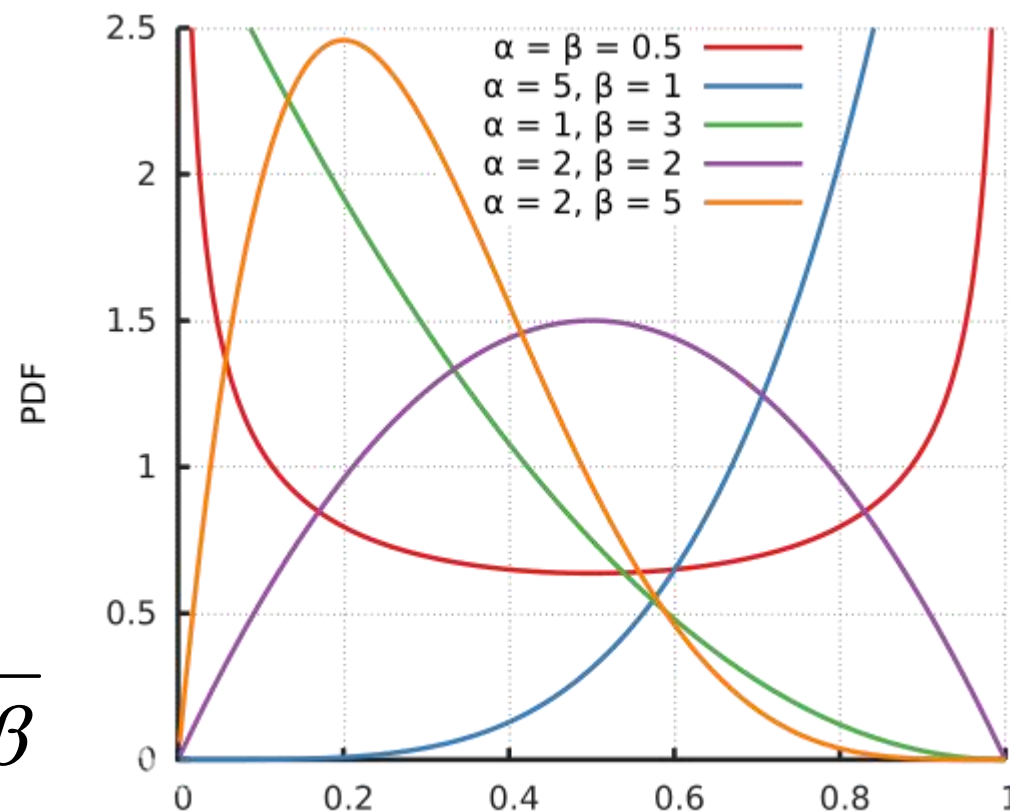
- Beta分布是二项分布的共轭分布，是指一组定义在(0,1)区间的连续概率分布，具有两个参数： $\alpha, \beta > 0$;

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}, x \in [0,1]$$

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)}$$

$$\Gamma(n) = (n-1)!$$

$$E(x) = \frac{\alpha}{\alpha + \beta}$$



Beta分布和二项分布

- 除去系数不看，Beta分布和二项分布具有相同的形式。将Beta分布当做先验分布，将二项分布当做似然函数。

$$p(\theta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}$$

$$p(x = k | \theta) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

Beta分布和二项分布

$$p(\theta|x) = \frac{p(\theta)p(x|\theta)}{p(x)} \propto p(\theta)p(x|\theta)$$
$$= \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1} \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

$$= \frac{\binom{n}{k}}{B(\alpha, \beta)} \theta^{(\alpha+k)-1} (1-\theta)^{(n-k+\beta)-1}$$

$$\propto \frac{1}{B(\alpha+k, n-k+\beta)} \theta^{(\alpha+k)-1} (1-\theta)^{(n-k+\beta)-1}$$

- 除去系数不看，可以发现先验概率和后验概率都属于Beta分布，所以认为Beta分布和二项分布属于共轭分布

Dirichlet分布

- Dirichlet分布是由Beta分布推广而来的，是多项式分布的共轭分布。

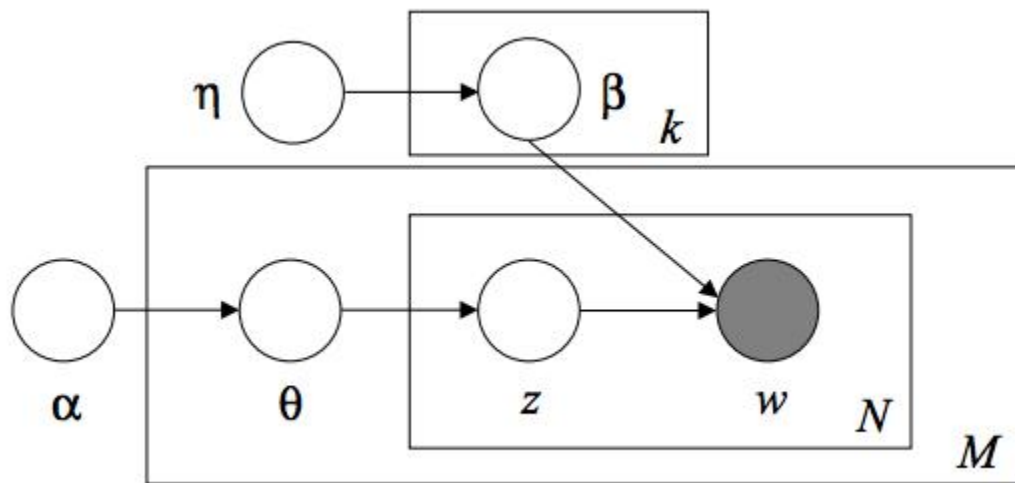
$$f(x_1, x_2, \dots, x_k, \alpha_1, \alpha_2, \dots, \alpha_k) = \frac{1}{B(\alpha)} \prod_{i=1}^k x_i^{\alpha_i - 1}$$

$$B(\alpha) = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}$$

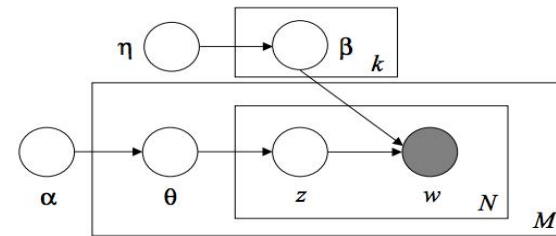
$$E(x_i) = \frac{\alpha_i}{\sum_{j=1}^k \alpha_j}$$

$$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k)$$

- 隐含狄利克雷分布(Latent Dirichlet Allocation, LDA)是一种基于贝叶斯算法模型，利用先验分布对数据进行似然估计并最终得到后验分布的一种方式。LDA是一种比较常用的主题模型。
- LDA假设文档主题是多项分布，多项分布的参数(先验分布)是服从Dirichlet分布，其实LDA是一种三层的贝叶斯模型。

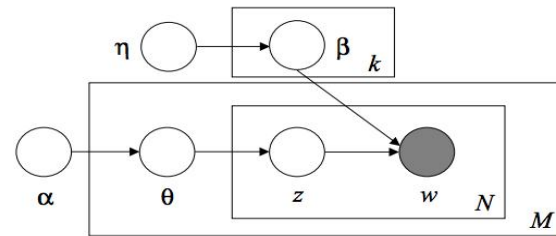


LDA



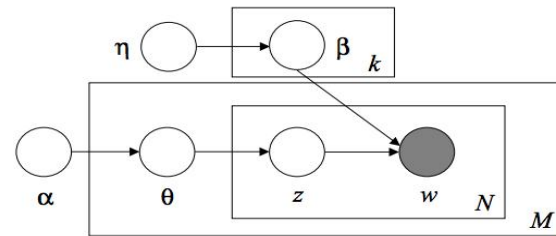
- 共有 M 篇文档，每个文档有 N_m 个单词，一共涉及到 K 个主题；
- 每篇文档都有各自的主题，主题分布是多项式分布，该多项式分布的参数服从Dirichlet分布，该Dirichlet分布的参数为 α ；
- 每个主题都有各自的词分布，词分布为为多项式分布，该多项式分布的参数服从Dirichlet分布，该Dirichlet分布的参数为 η ；
- 对于某篇文档 d 中的第 n 个词，首先从该文档的主题分布中采用一个主题，然后再这个主题对应的词分布中采用一个词，不断重复该操作，直到 m 篇文档全部完成上述过程。

LDA详细解释



- 词汇表中共有 V 个term(不可重复);
- 语料库中共有 m 篇文档 d_1, d_2, \dots, d_m ; 对于文档 d_i , 是由 N_i 个word组成的(word可重复); 语料库共有 K 个主题 T_1, T_2, \dots, T_K ;
- α 和 η 是先验分布(Dirichlet分布)的参数;
- θ 是每篇文档的**主题分布**, 是一个 K 维的向量;
- 对于第 i 篇文档 d_i , 在主题分布 θ_i 下, 可以确定一个具体的主题 $z_{ij}=k$
- β 是每个主题的**词分布**, 是一个 V 维的向量;
- 由 z_{ij} 选择 $\beta_{z_{ij}}$ (单词概率), 表示由词分布 $\beta_{z_{ij}}$ 确定term, 即可得到最终的观测值 w_{ij} 。

LDA详细解释

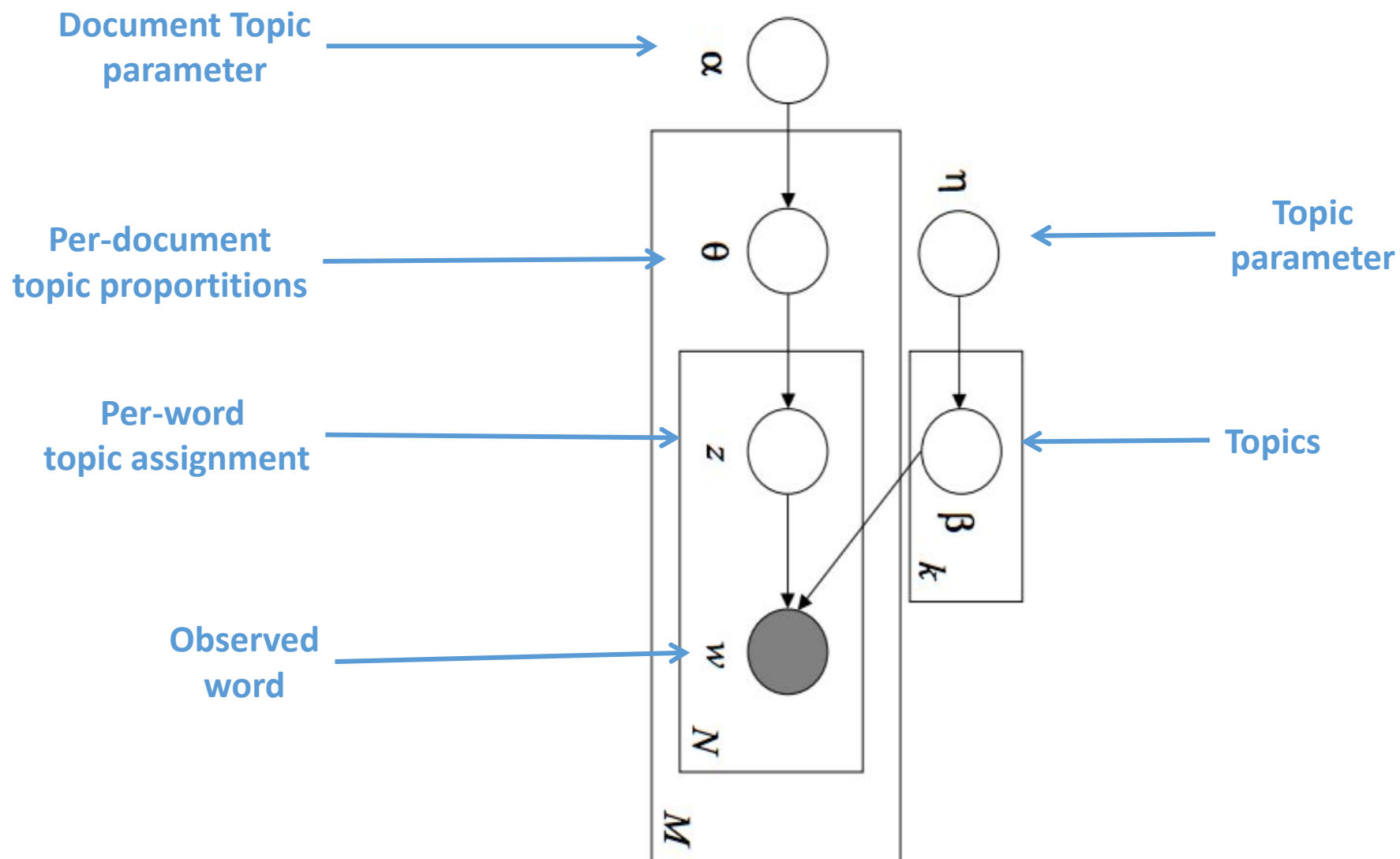


- LDA在进行文本的主题模型构建的时候，对于D个文档，K个主题的模式构建流程如下所示：

- 对于每一个Topic k ，计算出 β 的值： $\beta_k \sim \text{Dirichlet}(\eta), k = 1 \dots K$
- 对于每一个Document d ，计算出 θ 的值： $\theta_d \sim \text{Dirichlet}(\alpha), d = 1 \dots D$
- 对于Document d 中的word i :
 - 计算所属的topic z 值； $z_{di} \sim \text{Multinomial}(\theta_d)$
 - 计算出观测到的单词 w ； $w_{ij} \sim \text{Multinomial}(\beta_{z_{di}})$
- 对于 θ 、 β 、 z 的参数估计，基于贝叶斯算法可以得到如下公式：

$$p(z, \theta, \beta | w, \alpha, \eta) = \frac{p(z, \theta, \beta, w, \alpha, \eta)}{p(w, \alpha, \eta)} = \frac{p(z, \theta, \beta, w | \alpha, \eta) p(\alpha, \eta)}{p(w | \alpha, \eta) p(\alpha, \eta)} = \frac{p(z, \theta, \beta, w | \alpha, \eta)}{p(w | \alpha, \eta)}$$

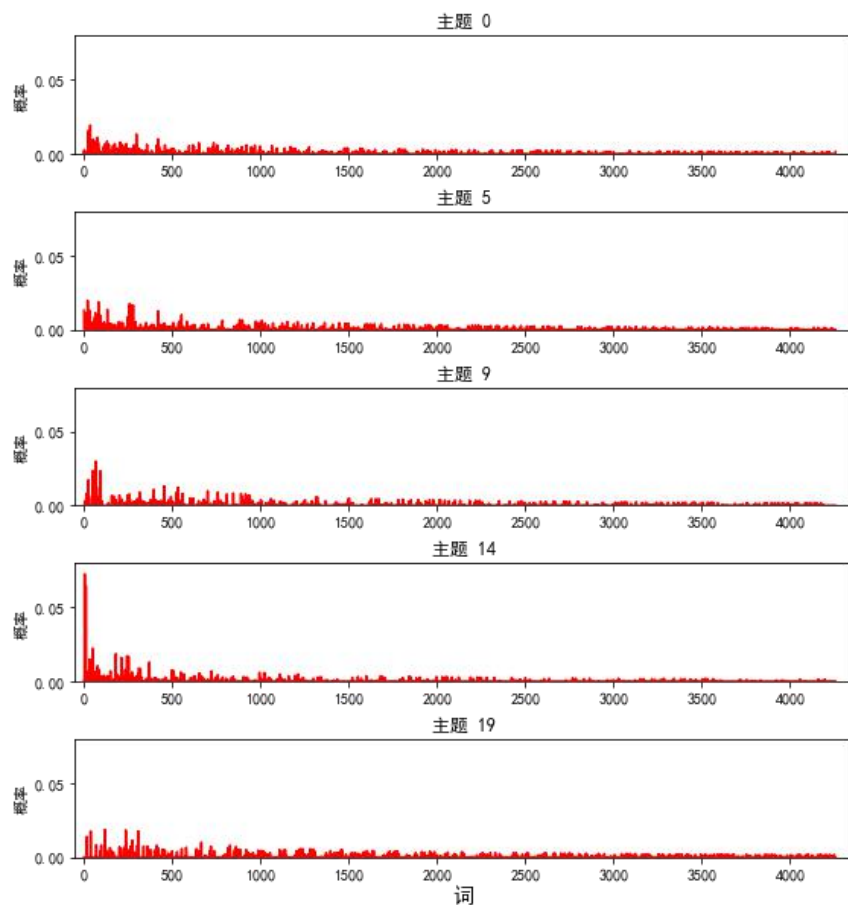
LDA理解



LDA主题模型案例

- 安装lda, eg: `pip install lda`

主题的词分布



文档的主题分布

