基于概率潜在语义分析模型的潜在语义分析

2023 秋《统计学方法及其应用》第一次大作业

2021270019 自硕 21 陈昱宏

2023 年 11 月 23 日

摘要

本作业基于概率潜在语义分析模型(Probabilistic Latent Semantic Analysis, PLSA),对给定文本摘要的数据，分析其潜在的主题。第一部分对于 PLSA 模型中的似然函数，利用贝叶斯概率函数对完全数据和非完全数据进行推导，并且也对 EM 算法的 E-step 和 M-step 进行了理论推导。第二部分根据作业给出的文本数据集，利用 Python 语言实现了 EM 算法以及潜在语义的概率求解，并给出了 $P(word|topic)$ 的仿真结果。

目录

1 似然函数与 EM 算法步骤推导 **2**
1.1 符号定义 2
1.2 似然函数推导 3
1.3 EM 算法推导 4

2 参数推断与主题发现 **5**
2.1 EM 算法实现 5
2.2 参数推断与主题发现 6
2.2.1 语料库词频统计 6
2.2.2 主题与学科领域对应性分析 7
2.2.3 潜在语义个数的确定 8

1 似然函数与 EM 算法步骤推导

1.1 符号定义

根据作业说明，我们对文本、词汇和主题变量及概率如图1所示。我们采用概率潜在语义分析模型（Probabilistic Latent Semantic Analysis, PLSA）进行语义分析，其概率图和隐含的概率公式如图2所示。

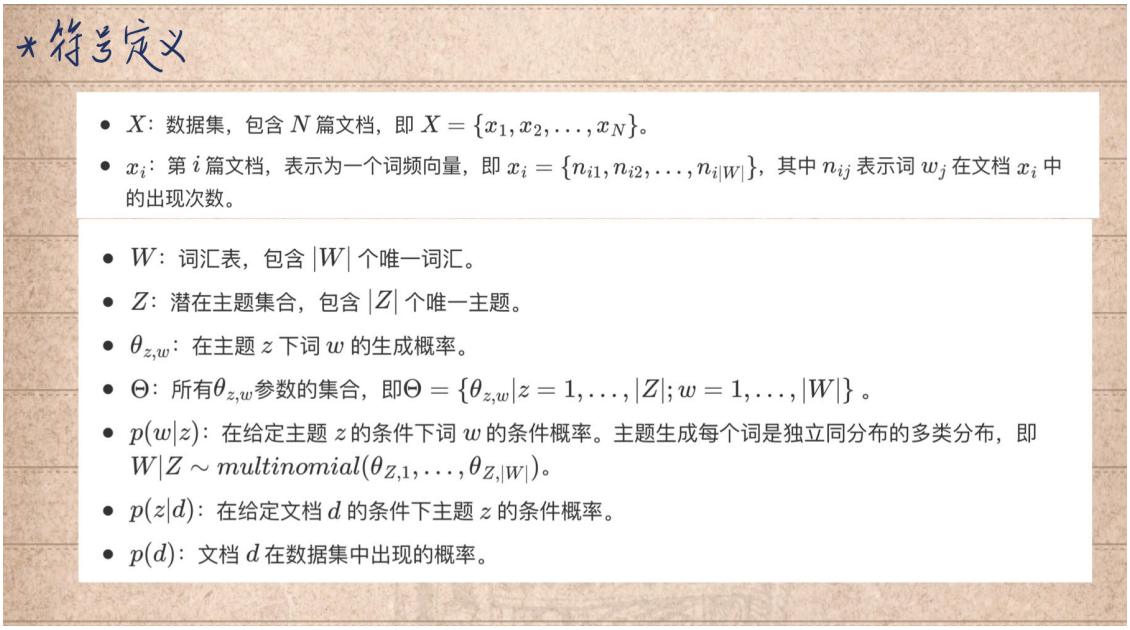

图 1: 符号定义

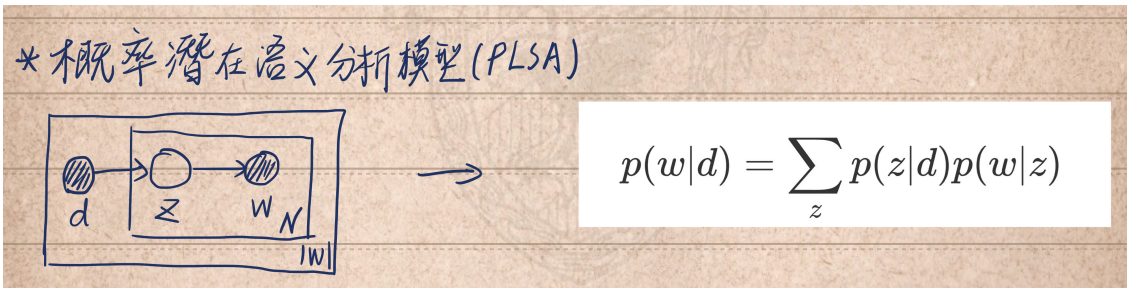

图 2: PLSA 模型

1.2 似然函数推导

推导过程如图3所示，由于似然函数中有 λ_k 参数，因此图3中的 $L(\Theta|\mathbf{X},\mathbf{Z})$ 和 $L(\Theta|\mathbf{X})$ 应该写成 $L(\Theta,\lambda|\mathbf{X},\mathbf{Z})$ 和 $L(\Theta,\lambda|\mathbf{X})$。根绝推导结果我们有

$$L(\Theta,\lambda|\mathbf{X},\mathbf{Z})=\prod_{i=1}^{N}\prod_{k=1}^{|Z|}\left\{\lambda_k\frac{\left(\sum_{j=1}^{|W|}X_{ij}\right)!}{\prod_{j=1}^{|W|}X_{ij}!}\prod_{j=1}^{|W|}\theta_{k,j}^{X_{ij}}\right\}^{I(Z_i=k)} \tag{1}$$

$$L(\Theta,\lambda|\mathbf{X})=\prod_{i=1}^{N}\sum_{k=1}^{|Z|}\left\{\lambda_k\frac{\left(\sum_{j=1}^{|W|}X_{ij}\right)!}{\prod_{j=1}^{|W|}X_{ij}!}\prod_{j=1}^{|W|}\theta_{k,j}^{X_{ij}}\right\} \tag{2}$$

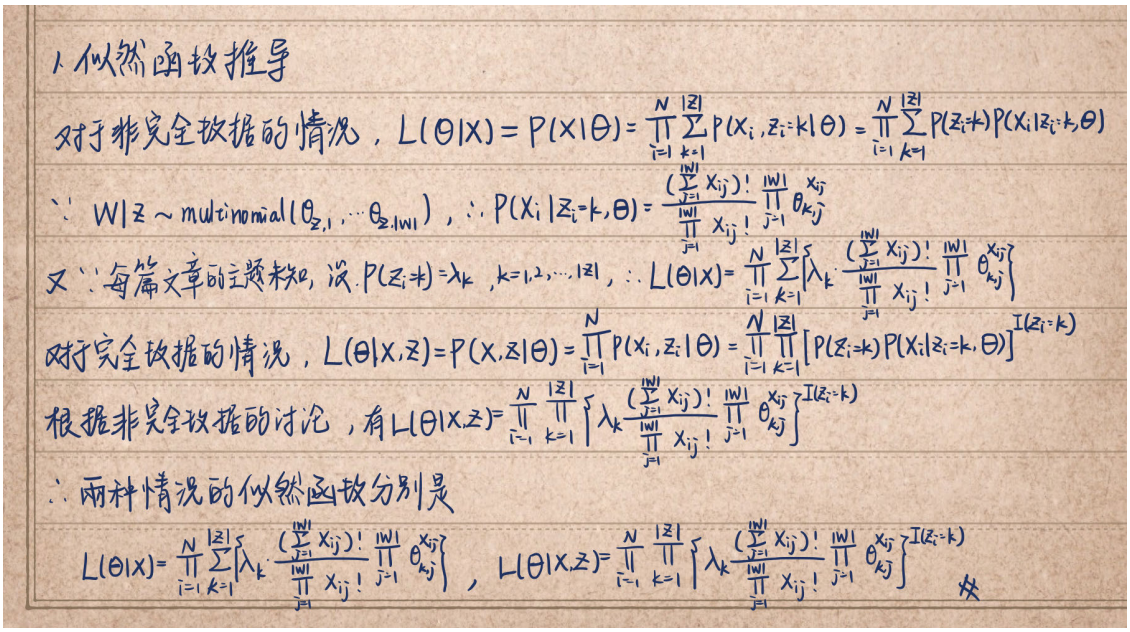

图 3: 似然函数推导过程

1.3 EM 算法推导

在使用 EM 算法求解潜在语义概率时，需要先对 EM 算法的 E 步骤和 M 步骤进行推导，尤其是对于 Q 函数的推导，可以透过对似然函数取期望获得 Q 函数，而 M 步骤则是透过最大似然准则求取参数的估计值，具体推导步骤如图4所示。

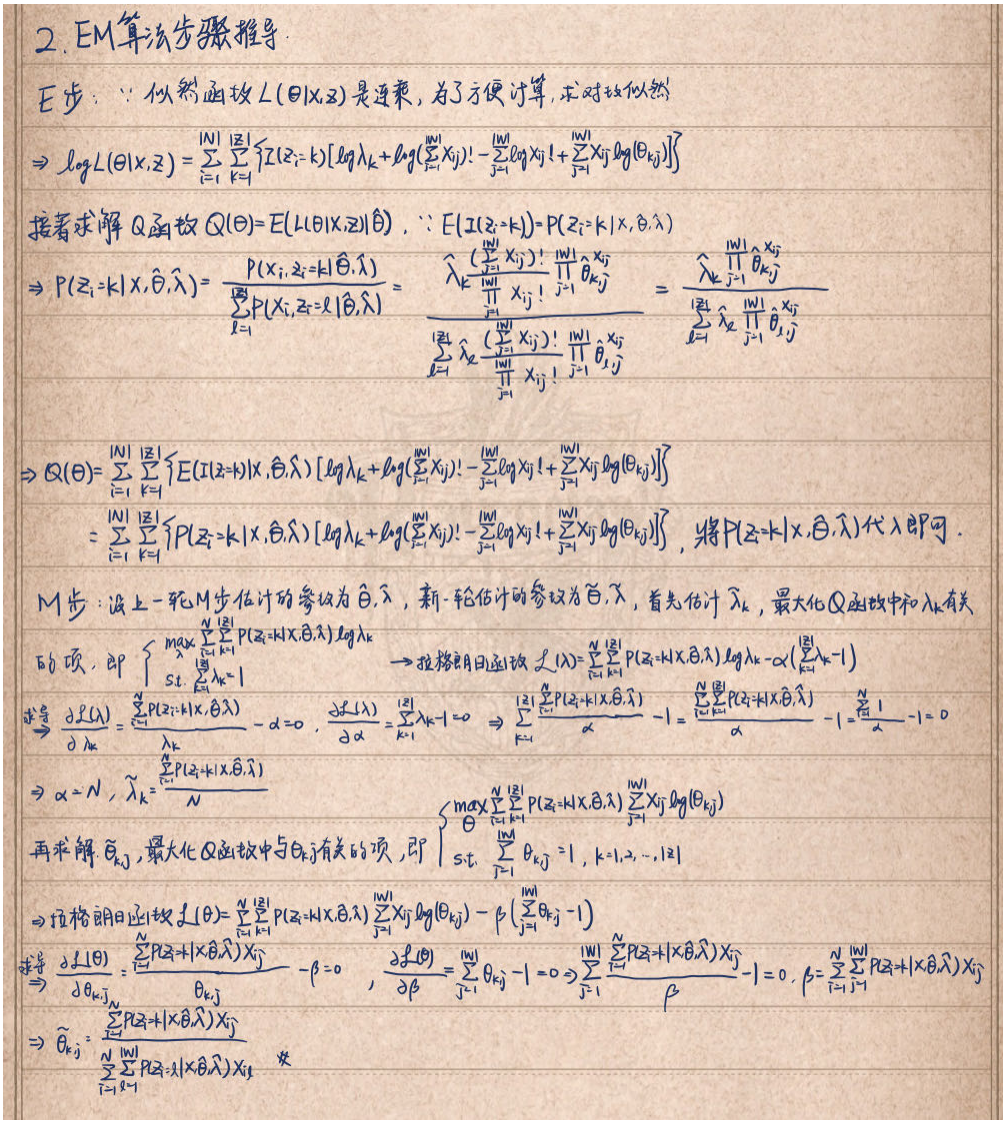

图 4: EM 算法推导过程

2 参数推断与主题发现

2.1 EM 算法实现

本次作业采用 python 代码实现 EM 算法，代码在 EM.py 文件中，由 main.py 文件调用该函数。在 Θ 和 λ 参数的初始化中，透过 np.random.rand 函数随机初始化，再透过归一化确保 $0 \leq \Theta \leq 1$ 和 $0 \leq \lambda \leq 1$。在停止准则上，根据两次 M 步骤计算得到的 Θ 和 λ 之间的变化程度来判断停止与否，即如下的公式：

$$\Delta\Theta = \frac{||\Theta_{m+1} - \Theta_m||_2}{||\Theta_m||_2} \tag{3}$$

$$\Delta\lambda = \frac{||\lambda_{m+1} - \lambda_m||_2}{||\lambda_m||_2} \tag{4}$$

其中，Θ_m 和 λ_m 分别代表第 m 个迭代过程中的参数，当 $max(\Delta\Theta, \Delta\lambda) < \epsilon$ 时，代表参数继续迭代的变化很小，可以停止迭代。

在算法实现上，为了加速计算，我利用 numpy 的向量化计算来代替 for 循环，具体的 E 步骤和 M 步骤代码如下所示。在 E 步骤中，第四行的 Pi_theta$\in \mathbb{R}^{|Z|\times|W|}$ 计算的是 $\theta_{k,j}^{X_{ij}}, k = 1, 2, ..., |Z|$，第六行的 P_z_k$\in \mathbb{R}^{1\times|Z|}$ 计算的是 $\lambda_k \prod_{j=1}^{|W|} \theta_{k,j}^{X_{ij}}$，第八行的 P_z[i]（其中 P_z$\in \mathbb{R}^{N\times|Z|}$）是计算的是 $P(Z_i = k|X, \hat{\Theta}, \hat{\lambda}) = \frac{\lambda_k \prod_{j=1}^{|W|} \theta_{k,j}^{X_{ij}}}{\sum_{l=1}^{|Z|} \lambda_l \prod_{j=1}^{|W|} \theta_{l,j}^{X_{ij}}}$。第九到十一行是避免数值出现 NaN 或是无穷大。在 M 步骤中，第十三到十五行是计算最大似然估计的 $\hat{\theta}_{k,j}^{MLE} = \frac{\sum_{i=1}^{N} P(Z_i=k|X,\hat{\Theta},\hat{\lambda})X_{ij}}{\sum_{l=1}^{|Z|}\sum_{i=1}^{N} P(Z_i=l|X,\hat{\Theta},\hat{\lambda})X_{ij}}$，第十七行是计算最大似然估计的 $\hat{\lambda}_k^{MLE} = \frac{\sum_{i=1}^{N} P(Z_i=k|X,\hat{\Theta},\hat{\lambda})}{N}$。

```
1     # E-step
2     for i in range(N):
3         # calculate \prod_{j=1}^{|W|} theta_{k,j}^{X_ij}
4         Pi_theta = np.power(Theta, X[i])
5         # calculate P(z_i=k)
```

```
            P_z_k = lambda_k * np.prod(Pi_theta, axis=1)
            P_z_k[np.isnan(P_z_k)] = 0
            P_z[i] = P_z_k / np.sum(P_z_k)
        P_z[np.isnan(P_z)] = 0
        P_z[P_z == np.inf] = 0
        P_z[P_z == -np.inf] = 0
        # M-step
        Theta = P_z.T @ document_word_num_list
        sumTheta = np.sum(Theta, axis=1).reshape((Z, 1))
        Theta = Theta / sumTheta
        lambda_k = np.sum(P_z, axis=0) / N
```

为了避免 EM 算法陷入局部最优解，我们可以设计多次的 EM 算法，每次用不同的初始化参数，最后选择 Q 函数最高的结果作为最终参数推断结果。

2.2 参数推断与主题发现

2.2.1 语料库词频统计

词频统计的代码在 WordFrequency.py 文件中。在统计词频前，需要先对语料进行预处理，先对预料中的特殊字符替换成空格，再进行分词。英语的分词比较简单，可以直接透过空格分词，前面对特殊字符处理过后，还会遗留下因为特殊字符分割而生成的短词，我们把长度不足 3 的单词去除。此外，对于一些罕见词和停用词，参考博客给出的停用词进行过滤，停用词保存在 stopwords.txt 中。经过所有条件过滤完的词汇形成词汇表 W，总计有 $|W| = 35814$ 个词汇，所有词汇保存在 dictionary.json 中。

根据前面过滤的词汇表，对每个文档统计词频，会生成 $N \times |W|$ 的矩阵保存语料库的

词频信息，这些数据保存在 document_word_num_list.npy 中。

2.2.2 主题与学科领域对应性分析

假定潜在的语义数量为 4，可以运行 EM 算法推断得到的参数，运行代码在 main.py 中，推断完的 Θ 和 λ 参数结果分别保存在 Theta.npy 和 Lambda.npy 中，为了方便观察每个潜在语义下出现词频最高的词汇，利用 pandas 库将 Theta.npy 的结果排序后保存在 Theta_sorted.csv 中，数据处理的代码在 DataCollation.py 文件中。

观察排序后的 Θ 参数，如图5所示，Topic2 能够对应上计算机学科，高频单词有 learning、network、neural、training 等；Topic4 能够对应上物理学科，高频单词有 magnetic、energy、temperature、density 等；而 Topic1 和 Topic3 由于数学和统计学用词很多重叠部分，因此较难区分，不过 Topic3 的出现的高频单词有 manifolds、vector、complex 比较接近数学学科，剩余的 Topic1 单从词频很难精确确定是统计学，因为许多领域都会有统计学的用词，不过通过排除法可以任务 Topic1 对应的是统计学。

	Topic1_Word	Topic1_Probability	Topic2_Word	Topic2_Probability	Topic3_Word	Topic3_Probability	Topic4_Word	Topic4_Probability
2	inside	0.042308304	inside	0.039949532	inside	0.042778253	inside	0.041280261
3	data	0.008343546	learning	0.012056247	paper	0.00782727	model	0.009204878
4	method	0.008304362	data	0.010799046	prove	0.005817544	method	0.007052514
5	paper	0.006989115	model	0.009354233	study	0.005500447	based	0.006556801
6	model	0.006546288	method	0.006944081	method	0.005449872	paper	0.004521595
7	based	0.005779306	network	0.006126908	space	0.005418905	data	0.004310854
8	algorithm	0.005020877	models	0.006000907	function	0.003633298	learning	0.004292597
9	time	0.004418975	networks	0.005833207	solutions	0.003617776	models	0.003895125
10	proposed	0.004294774	time	0.005204802	model	0.003539788	system	0.003622029
11	methods	0.003585562	paper	0.005186581	finite	0.003450693	phase	0.003512349
12	learning	0.003533497	neural	0.004835044	equation	0.003412318	networks	0.003477412
13	system	0.003501569	based	0.004807289	field	0.003404511	idea	0.003450822
14	idea	0.003456289	deep	0.00459515	functions	0.00334984	study	0.003338214
15	linear	0.0031378	algorithm	0.004592726	algebra	0.003341969	neural	0.003331567
16	dimensional	0.003124402	algorithms	0.004357475	based	0.00319305	field	0.003256854
17	systems	0.003078296	propose	0.004292672	result	0.003155383	spin	0.003245513
18	function	0.003073807	methods	0.004062514	dimensional	0.003146392	network	0.003106436
19	analysis	0.003048888	performance	0.004025286	time	0.003102406	propose	0.002874226
20	magnetic	0.002991071	proposed	0.003637573	type	0.002970054	space	0.002781699
21	models	0.002985875	real	0.003525001	curvature	0.002854301	magnetic	0.002717081
22	study	0.002968606	study	0.00291121	set	0.002668506	theory	0.002494501
23	control	0.002927114	training	0.002883361	theory	0.002659833	algorithm	0.002480305
24	propose	0.002925081	function	0.002813847	models	0.002590868	time	0.002479387
25	algorithms	0.002775693	tasks	0.002771853	linear	0.002474851	energy	0.002428196
26	approximation	0.002762868	idea	0.002726501	methods	0.002439098	temperature	0.002420415
27	field	0.002670838	image	0.002653887	boundary	0.002438319	function	0.002378086
28	structure	0.002647587	set	0.002652868	class	0.002404027	performance	0.002245818
29	optimal	0.002611879	demonstrate	0.002626663	vector	0.002385058	density	0.002229553
30	set	0.002438171	graph	0.002606211	manifolds	0.002384617	analysis	0.002147237
31	solution	0.0024127	classification	0.002580873	complex	0.002364544	deep	0.002130021
32	theory	0.002339778	framework	0.002557605	representations	0.002351879	training	0.002031969
33	network	0.002337392	features	0.002455674	lie	0.002348501	structure	0.001932023
34	spin	0.00232733	task	0.002454933	properties	0.002331516	framework	0.001881216

图 5: 参数推断结果

2.2.3 潜在语义个数的确定

在潜在语义的数量未知下，我们可以把每个文本的词频向量聚类，根据聚类的聚类簇数量来决定潜在语义的数量。但由于每个词频向量维度非常高，且很稀疏，因此可以考虑采用主成分分析等方法进行降维后再聚类。