

利用数据库编程进行LDA模拟和检验

10230327 屈良煜

一、本项目原理：LDA模型简述

LDA将文档生成过程建模成一个概率图模型，即假设文档是由多个单词混合生成的，其中单词的出现是由潜在主题决定的。具体过程如下：

1. 假设每篇文档的主题分布服从一个狄利克雷分布 ($\theta \sim \text{Dirichlet}(\alpha)$)，每个主题的单词分布也服从一个狄利克雷分布 ($\phi \sim \text{Dirichlet}(\alpha)$)
2. 对于每篇文档，我们先从主题分布 (θ) 中抽取一个主题记作 (z)，再从主题 (z) 的单词分布 (ϕ_z) 中抽取一个单词。
3. 则给定 (K) 个主题和文档 (d)，文档中由 n 个词语组成的文本 (ω) 被生成的概率为：

$$P(w) = \prod_{n=1}^N \sum_{z=1}^K P(w_n | z; \phi) P(z | d; \theta)$$

(边缘似然)

二、Dirichlet Distribution概述

密度函数

狄利克雷分布 $\text{Dir}(\boldsymbol{\alpha})$ 是定义在 $K - 1$ 维辛形 (simplex) 上的连续概率分布，是 Beta 分布在多元 (K 元) 情况下的推广。

假设随机向量 $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_K)$ 服从 $\text{Dirichlet}(\boldsymbol{\alpha})$ ，其中 $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_K)$ 是超参数，且 $\alpha_k > 0$ 。

其概率密度函数 (PDF) 为： $f(\boldsymbol{\theta} | \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^K \theta_k^{\alpha_k - 1}$ 约束条件: $\theta_k \in [0, 1]$ 、 $\sum_{k=1}^K \theta_k = 1$

其中 $B(\boldsymbol{\alpha})$ 是多元 Beta 函数 (归一化常数)： $B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^K \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^K \alpha_k)}$

选择狄利克雷分布作为先验的理由

1. 它是多项式分布的“共轭先验” (Conjugate Prior)

当我们使用狄利克雷分布作为多项式分布的先验时，后验分布仍保持狄利克雷分布的形式。后验分布与先验分布的形式相同，使得我们在推断过程中可以进行代数运算，而不是进行复杂的积分运算（尤其是在使用 MCMC 或吉布斯采样时），这在计算上是极其高效的。

2. 它适合建模比例

狄利克雷分布完美地定义在满足和为 1 的向量空间上。LDA 的两个核心参数 $\boldsymbol{\theta}$ 和 $\boldsymbol{\phi}$ 本质上都是概率向量（即所有元素和为 1）

- **文档主题比例 $\boldsymbol{\theta}$** : 一篇文档必须 100% 由所有主题构成。 $\sum_{k=1}^K \theta_k = 1$ 。
- **主题词语比例 $\boldsymbol{\phi}$** : 一个主题必须 100% 由所有词汇构成。 $\sum_{v=1}^V \phi_v = 1$ 。

三、本项目背景与目标

实际上，使用LDA等主题模型分析文本时，模型推断出的结果（文档主题比例）只是模型对真相的猜测，猜测准确度未知。我采用统计模拟的方式对模型的性能进行科学量化，检验LDA模型推断的准确度如何，从而为LDA在推荐系统、文本挖掘等领域的可靠使用提供坚实的统计学基础。

具体来说：

1. 生成 $N = 3000$ 篇文档。每一篇文档都附带一个**已知的、真实的**主题比例向量 (θ_{true})，并存储在数据库中，作为验证的基础。
2. 以这3000篇文档作为输入数据，利用Python训练LDA模型，推断出每篇文档的预测主题分布(θ_{pred})。
3. 使用两个向量的**余弦相似度**作为量化指标，对比 θ_{true} 向量 和 θ_{pred} 向量 在向量空间中的夹角。夹角越小，相似度（分数越接近 1）越高，表明模型推断越准确。
4. 将数据生成、存储、模型推断、结果验证等一系列工程产生的所需的样本和产生的结果完整地储存在数据库中，以便批量进行分析。

注：

1. 本项目主要用到Python、MySQL。本项目所用方法为统计模拟。

四、本项目流程具体介绍

1、初始化与参数

在 config.py 定义了 K 个主题、 V 个词汇， N 篇文档等实验常量和 DB_HOST 、 DB_USER 等数据库连接常量，生成 $V = 1000$ 个词汇，并用“主题+编号”的格式对词汇编码以作区分。

在 `db_manager.py` 中，定义了数据库连接函数 `get_db_connection` 为项目与本地MySQL数据库交互作准备；定义了 `record_simulation_parameters` 函数，通过SQL语句将本次模拟实验的所有配置参数记录在数据库。

在主控程序 `main_run.py` 中，启动本次模拟项目，连接数据库，并获取实验编号（`run_id`）

2、统计模拟（核心步骤）

在 `stat_sim.py` 中，构建 K 行 V 列的 ϕ 矩阵，模拟主题—词汇概率分布。分配策略：将 90% 的概率分配给某个主题的专属词汇，将 10% 的概率分配给其它词汇，保证归一化。

在 `stat_sim.py` 中，定义 `generate_documents` 函数，根据 LDA 原理进行狄利克雷抽样，生成 N 篇文档和对应的真实 θ 向量。最后再进行数据格式化处理，如将词语连接成完整字符串文本、将实验序号和真实 θ 打包成元组。

在主控程序中调用上述函数。

3、数据入库

在 `db_manager.py` 中，定义 `bulk_insert_documents` 函数，通过 SQL 语句，将生成的文档和真实 θ 向量批量导入数据库保存。在主控程序中调用该函数。

4、模型训练与推断

在 `db_manager.py` 中，定义 `fetch_documents_for_analysis` 函数，通过 SQL 语句从数据库中提取第三步的文档和向量作为比对。

调用 `sklearn.LatentDirichletAllocation` 函数对文本进行训练和推断。为使训练更容易，采用 `CountVectorizer` 函数将文本转化为矩阵。从而得到 θ_{pred} 向量。

注：`sklearn.LatentDirichletAllocation` 是 Python 中 `scikit-learn` 库提供的一个类，用于实现潜在狄利克雷分配模型，即本项目中要量化性能的模型。

5、量化检验

在主控函数中主要对比 θ_{pred} 和真实 θ 向量，计算余弦相似度。将结果储存在数据库中。最终结果为 0.7088。

五、本项目实际意义

- 为 LDA 模型的应用提供统计可靠性背书（主要）
- 在推荐系统、文本聚类和情感分析等任务中， θ 向量经常被用作文档的精简特征表示。只有经过高相似度验证（即模型推断准确）的 θ 向量，才能确保后续算法建立在高质量的数据基础之上，避免“垃圾输入，垃圾输出”的问题。

注：本项目在函数的命名和输出的格式化上使用了 AI：Gemini 2.5 Flash