第20章 潜在狄利克雷分配

1.狄利克雷分布的概率密度函数为$p(\theta|\alpha)=\frac{\Gamma(\sum_{i=1}^{k}\alpha_i)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\prod_{i=1}^{k}\theta_i^{\alpha_i-1}$ 其中$\sum_{i=1}^{k}\theta_i=1,\theta_i\geq 0,\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_k),\alpha_i>0,i=1,2,\cdots$,狄利克雷分布是多项分布的共轭先验。

2.潜在狄利克雷分配2.潜在狄利克雷分配（LDA）是文本集合的生成概率模型。模型假设话题由单词的多项分布表示，文本由话题的多项分布表示，单词分布和话题分布的先验分布都是狄利克雷分布。LDA模型属于概率图模型可以由板块表示法表示LDA模型中，每个话题的单词分布、每个文本的话题分布、文本的每个位置的话题是隐变量，文本的每个位置的单词是观测变量。

3.LDA生成文本集合的生成过程如下：

（1）话题的单词分布：随机生成所有话题的单词分布，话题的单词分布是多项分布，其先验分布是狄利克雷分布。

（2）文本的话题分布：随机生成所有文本的话题分布，文本的话题分布是多项分布，其先验分布是狄利克雷分布。

（3）文本的内容：随机生成所有文本的内容。在每个文本的每个位置，按照文本的话题分布随机生成一个话题，再按照该话题的单词分布随机生成一个单词。

4.LDA模型的学习与推理不能直接求解。通常采用的方法是吉布斯抽样算法和变分EM算法，前者是蒙特卡罗法而后者是近似算法。

5.LDA的收缩的吉布斯抽样算法的基本想法如下。目标是对联合概率分布$p(w,z,\theta,\varphi|\alpha,\beta)$进行估计。通过积分求和将隐变量$\theta$和$\varphi$消掉，得到边缘概率分布$p(w,z|\alpha,\beta)$；对概率分布$p(w|z,\alpha,\beta)$进行吉布斯抽样，得到分布$p(w|z,\alpha,\beta)$的随机样本；再利用样本对变量$z$，$\theta$和$\varphi$的概率进行估计，最终得到LDA模型$p(w,z,\theta,\varphi|\alpha,\beta)$的参数估计。具体算法如下对给定的文本单词序列，每个位置上随机指派一个话题，整体构成话题系列。然后循环执行以下操作。对整个文本序列进行扫描，在每一个位置上计算在该位置上的话题的满条件概率分布，然后进行随机抽样，得到该位置的新的话题，指派给这个位置。

6.变分推理的基本想法如下。假设模型是联合概率分布$p(x,z)$，其中x是观测变量（数据），z是隐变量。目标是学习模型的后验概率分布$p(z|x)$。考虑用变分分布$q(z)$近似条件概率分布$p(z|x)$，用KL散度计算两者的相似性找到与$p(z|x)$在KL散度意义下最近的$q^*(z)$，用这个分布近似$p(z|x)$。假设$q(z)$中的z的所有分量都是互相独立的。利用Jensen不等式，得到KL散度的最小化可以通过证据下界的最大化实现。因此，变分推理变成求解以下证据下界最大化问题：$L(q,\theta)=E_q[\log p(x,z|\theta)]-E_q[\log q(z)]$

7.LDA的变分EM算法如下。针对LDA模型定义变分分布，应用变分EM算法。目标是对证据下界$L(\gamma,\eta,\alpha,\varphi)$进行最大化，其中$\alpha$和$\varphi$是模型参数，$\gamma$和$\eta$是变分参数。交替迭代E步和M步，直到收敛。

- （1）E步：固定模型参数α，φ，通过关于变分参数γ，η的证据下界的最大化，估计变分参数γ，η。
- （2）M步：固定变分参数γ，η，通过关于模型参数α，φ的证据下界的最大化，估计模型参数α，φ。

潜在狄利克雷分配（latent Dirichlet allocation,LDA），作为基于贝叶斯学习的话题模型，是潜在语义分析、概率潜在语义分析的扩展，于2002年由Blei等提出dA在文本数据挖掘、图像处理、生物信息处理等领域被广泛使用。

LDA模型是文本集合的生成概率模型假设每个文本由话题的一个多项分布表示，每个话题由单词的一个多项分布表示，特别假设文本的话题分布的先验分布是狄利克雷分布，话题的单词分布的先验分布也是狄利克雷分布。先验分布的导入使LDA能够更好地应对话题模型学习中的过拟合现象。

LDA的文本集合的生成过程如下：首先随机生成一个文本的话题分布，之后在该文本的每个位置，依据该文本的话题分布随机生成一个话题，然后在该位置依据该话题的单词分布随机生成一个单词，直至文本的最后一个位置，生成整个文本。重复以上过程生成所有文本。

LDA模型是含有隐变量的概率图模型。模型中，每个话题的单词分布，每个文本的话题分布，文本的每个位置的话题是隐变量；文本的每个位置的单词是观测变量。LDA模型的学习与推理无法直接求解通常使用吉布斯抽样（ Gibbs sampling）和变分EM算法（variational EM algorithm），前者是蒙特卡罗法，而后者是近似算法。

```
from gensim import corpora, models, similarities
from pprint import pprint
import warnings
```

```
f = open('data/LDA_test.txt')
stop_list = set('for a of the and to in'.split())
# texts = [line.strip().split() for line in f]
# print 'Before'
# pprint(texts)
print('After')
```

```
After
```

```
texts = [[
    word for word in line.strip().lower().split() if word not in stop_list
] for line in f]
print('Text = ')
pprint(texts)
```

```
Text =
[['human', 'machine', 'interface', 'lab', 'abc', 'computer', 'applications'],
 ['survey', 'user', 'opinion', 'computer', 'system', 'response', 'time'],
 ['eps', 'user', 'interface', 'management', 'system'],
 ['system', 'human', 'system', 'engineering', 'testing', 'eps'],
 ['relation', 'user', 'perceived', 'response', 'time', 'error',
'measurement'],
 ['generation', 'random', 'binary', 'unordered', 'trees'],
 ['intersection', 'graph', 'paths', 'trees'],
 ['graph', 'minors', 'iv', 'widths', 'trees', 'well', 'quasi', 'ordering'],
 ['graph', 'minors', 'survey']]
```

```
dictionary = corpora.Dictionary(texts)
print(dictionary)
```

```
Dictionary(35 unique tokens: ['abc', 'applications', 'computer', 'human',
'interface']...)
```

```
V = len(dictionary)
corpus = [dictionary.doc2bow(text) for text in texts]
corpus_tfidf = models.TfidfModel(corpus)[corpus]
corpus_tfidf = corpus

print('TF-IDF:')
for c in corpus_tfidf:
    print(c)
```

```
TF-IDF:
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1)]
[(2, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1)]
[(4, 1), (10, 1), (12, 1), (13, 1), (14, 1)]
[(3, 1), (10, 2), (13, 1), (15, 1), (16, 1)]
[(8, 1), (11, 1), (12, 1), (17, 1), (18, 1), (19, 1), (20, 1)]
[(21, 1), (22, 1), (23, 1), (24, 1), (25, 1)]
[(24, 1), (26, 1), (27, 1), (28, 1)]
[(24, 1), (26, 1), (29, 1), (30, 1), (31, 1), (32, 1), (33, 1), (34, 1)]
[(9, 1), (26, 1), (30, 1)]
```

```
print('\nLSI Model:')
lsi = models.LsiModel(corpus_tfidf, num_topics=2, id2word=dictionary)
topic_result = [a for a in lsi[corpus_tfidf]]
pprint(topic_result)
```

```
LSI Model:
[[(0, 0.9334981916792652), (1, 0.10508952614086528)],
 [(0, 2.031992374687025), (1, -0.047145314121742235)],
 [(0, 1.5351342836582078), (1, 0.13488784052204628)],
 [(0, 1.9540077194594532), (1, 0.21780498576075008)],
 [(0, 1.2902472956004092), (1, -0.0022521437499372337)],
 [(0, 0.022783081905505403), (1, -0.7778052604326754)],
 [(0, 0.05671567576920905), (1, -1.1827703446704851)],
 [(0, 0.12360003320647955), (1, -2.6343068608236835)],
 [(0, 0.23560627195889133), (1, -0.9407936203668315)]]
```

```
print('LSI Topics:')
pprint(lsi.print_topics(num_topics=2, num_words=5))
```

```
LSI Topics:
[(0,
  '0.579*"system" + 0.376*"user" + 0.270*"eps" + 0.257*"time" + '
  '0.257*"response"'),
 (1,
  '-0.480*"graph" + -0.464*"trees" + -0.361*"minors" + -0.266*"widths" + '
  '-0.266*"ordering"')]
```

```
similarity = similarities.MatrixSimilarity(lsi[corpus_tfidf])   #
similarities.Similarity()
print('Similarity:')
pprint(list(similarity))
```

```
Similarity:
[array([ 1.        ,  0.9908607 ,  0.9997008 ,  0.9999994 ,  0.9935261 ,
        -0.08272626, -0.06414512, -0.06517283,  0.13288835], dtype=float32),
 array([0.9908607 , 0.99999994, 0.9938636 , 0.99100804, 0.99976987,
        0.0524564 , 0.07105229, 0.070025  , 0.2653665 ], dtype=float32),
 array([ 0.9997008 ,  0.9938636 ,  0.99999994,  0.999727  ,  0.99600756,
        -0.05832579, -0.03971674, -0.04074576,  0.15709123], dtype=float32),
 array([ 0.9999994 ,  0.99100804,  0.999727  ,  1.        ,  0.9936501 ,
        -0.08163348, -0.06305084, -0.06407862,  0.13397504], dtype=float32),
 array([0.9935261 , 0.99976987, 0.99600756, 0.9936501 , 0.99999994,
```

```
       0.03102366, 0.04963995, 0.04861134, 0.24462426], dtype=float32),
 array([-0.08272626,  0.0524564 , -0.05832579, -0.08163348,  0.03102366,
        0.99999994,  0.99982643,  0.9998451 ,  0.97674036], dtype=float32),
 array([-0.06414512,  0.07105229, -0.03971674, -0.06305084,  0.04963995,
        0.99982643,  1.        ,  0.9999995 ,  0.9805657 ], dtype=float32),
 array([-0.06517283,  0.070025  , -0.04074576, -0.06407862,  0.04861134,
        0.9998451 ,  0.9999995 ,  1.        ,  0.9803632 ], dtype=float32),
 array([0.13288835, 0.2653665 , 0.15709123, 0.13397504, 0.24462426,
       0.97674036, 0.9805657 , 0.9803632 , 1.        ], dtype=float32)]
```

```
print('\nLDA Model:')
num_topics = 2
lda = models.LdaModel(
    corpus_tfidf,
    num_topics=num_topics,
    id2word=dictionary,
    alpha='auto',
    eta='auto',
    minimum_probability=0.001,
    passes=10)
doc_topic = [doc_t for doc_t in lda[corpus_tfidf]]
print('Document-Topic:\n')
pprint(doc_topic)
```

```
LDA Model:
Document-Topic:

[[(0, 0.02668742), (1, 0.97331256)],
 [(0, 0.9784582), (1, 0.021541778)],
 [(0, 0.9704323), (1, 0.02956772)],
 [(0, 0.97509205), (1, 0.024907947)],
 [(0, 0.9785106), (1, 0.021489413)],
 [(0, 0.9703556), (1, 0.029644381)],
 [(0, 0.04481229), (1, 0.9551877)],
 [(0, 0.023327617), (1, 0.97667235)],
 [(0, 0.058409944), (1, 0.9415901)]]
```

```
for doc_topic in lda.get_document_topics(corpus_tfidf):
    print(doc_topic)
```

```
[(0, 0.026687337), (1, 0.9733126)]
[(0, 0.9784589), (1, 0.021541081)]
[(0, 0.97043234), (1, 0.029567692)]
[(0, 0.9750935), (1, 0.024906479)]
[(0, 0.9785101), (1, 0.021489937)]
[(0, 0.9703557), (1, 0.029644353)]
[(0, 0.044812497), (1, 0.9551875)]
[(0, 0.02332762), (1, 0.97667235)]
[(0, 0.058404233), (1, 0.9415958)]
```

```
for topic_id in range(num_topics):
    print('Topic', topic_id)
    # pprint(lda.get_topic_terms(topicid=topic_id))
    pprint(lda.show_topic(topic_id))
similarity = similarities.MatrixSimilarity(lda[corpus_tfidf])
print('Similarity:')
pprint(list(similarity))

hda = models.HdpModel(corpus_tfidf, id2word=dictionary)
topic_result = [a for a in hda[corpus_tfidf]]
print('\n\nUSE WITH CARE--\nHDA Model:')
pprint(topic_result)
print('HDA Topics:')
print(hda.print_topics(num_topics=2, num_words=5))
```

```
Topic 0
[('system', 0.094599016),
 ('user', 0.073440075),
 ('eps', 0.052545987),
 ('response', 0.052496374),
 ('time', 0.052453455),
 ('survey', 0.031701956),
 ('trees', 0.03162545),
 ('human', 0.03161709),
 ('computer', 0.031570844),
 ('testing', 0.031543963)]
Topic 1
[('graph', 0.0883405),
 ('trees', 0.06323685),
 ('minors', 0.06296622),
 ('interface', 0.03810195),
 ('computer', 0.03798469),
 ('human', 0.03792907),
 ('applications', 0.03792245),
 ('abc', 0.037920628),
 ('machine', 0.037917122),
```

```
 ('lab', 0.037909806)]
Similarity:
[array([1.        , 0.04940351, 0.05783966, 0.05292428, 0.04934979,
       0.05791992, 0.99981046, 0.99999374, 0.99940336], dtype=float32),
 array([0.04940351, 1.        , 0.99996436, 0.9999938 , 1.        ,
       0.99996364, 0.06883725, 0.04587576, 0.08387101], dtype=float32),
 array([0.05783966, 0.99996436, 1.0000001 , 0.99998796, 0.99996394,
       1.        , 0.07726298, 0.05431345, 0.09228647], dtype=float32),
 array([0.05292428, 0.9999938 , 0.99998796, 1.        , 0.9999936 ,
       0.9999875 , 0.07235384, 0.04939714, 0.08738345], dtype=float32),
 array([0.04934979, 1.        , 0.99996394, 0.9999936 , 1.        ,
       0.99996316, 0.06878359, 0.04582203, 0.08381741], dtype=float32),
 array([0.05791992, 0.99996364, 1.        , 0.9999875 , 0.99996316,
       0.99999994, 0.07734313, 0.05439373, 0.09236652], dtype=float32),
 array([0.99981046, 0.06883725, 0.07726298, 0.07235384, 0.06878359,
       0.07734313, 0.99999994, 0.9997355 , 0.9998863 ], dtype=float32),
 array([0.99999374, 0.04587576, 0.05431345, 0.04939714, 0.04582203,
       0.05439373, 0.9997355 , 0.99999994, 0.9992751 ], dtype=float32),
 array([0.99940336, 0.08387101, 0.09228647, 0.08738345, 0.08381741,
       0.09236652, 0.9998863 , 0.9992751 , 1.        ], dtype=float32)]
```

```
USE WITH CARE--
HDA Model:
[[(0, 0.18174982193320122),
  (1, 0.02455260642448283),
  (2, 0.741340573910992),
  (3, 0.013544078061059922),
  (4, 0.010094377639823477)],
 [(0, 0.39419292675663636),
  (1, 0.2921969355337328),
  (2, 0.26125786014858376),
  (3, 0.013539627392486701),
  (4, 0.01009410883245766)],
 [(0, 0.5182077872999125),
  (1, 0.3880947736463974),
  (2, 0.023895609845034207),
  (3, 0.01805202212531745),
  (4, 0.013458421673222807)],
 [(0, 0.03621384798236036),
  (1, 0.5504573172680752),
  (2, 0.020442846194997377),
  (3, 0.348529241707211),
  (4, 0.011535562414627153)],
 [(0, 0.9049762450848856),
  (1, 0.024748801100993395),
  (2, 0.017919024335434904),
  (3, 0.013543460312481508),
  (4, 0.010093932388992328)],
 [(0, 0.04681359723231631),
```

```
  (1, 0.03233799461088905),
  (2, 0.8510430252219996),
  (3, 0.01805587061936895),
  (4, 0.013458128836093802)],
 [(0, 0.42478083784052273),
  (1, 0.03858547281122597),
  (2, 0.4528531768644199),
  (3, 0.021680841796584305),
  (4, 0.016150009359845837),
  (5, 0.011953757612369628)],
 [(0, 0.2466808290730598),
  (1, 0.6908552821243853),
  (2, 0.015924569811569197),
  (3, 0.012039668311419834)],
 [(0, 0.500366457263008),
  (1, 0.048221177670061226),
  (2, 0.34671234963274666),
  (3, 0.02707530995137571),
  (4, 0.02018763747377598),
  (5, 0.014942188361070167),
  (6, 0.010992923111633942)]]
HDA Topics:
[(0, '0.122*graph + 0.115*minors + 0.098*management + 0.075*random +
0.063*error'), (1, '0.114*human + 0.106*system + 0.086*user + 0.064*iv +
0.063*measurement')]
```

代码参考：邹博

本文代码更新地址：https://github.com/fengdu78/lihang-code

中文注释制作：机器学习初学者公众号：ID:ai-start-com

配置环境：python 3.5+

代码全部测试通过。