実践　機械学習システム
第4章 トピックモデル

山木翔馬

トピックモデルとは

- データをいくつかの小さなジャンル（トピック）に割り当てる
- クラスタリングであれば、「実践　機械学習システム」は「機械学習」に分類するか「Python」に分類するか決めなければならない
- トピックモデルであれば複数のトピックに割り当てられる
- データの中心的な話題と重要でない話題とを見分けられる

潜在的ディリクレ配分法（LDA)

- 最も単純なトピックモデル

  （scikit-learnのsklearn.ldaは線形判別分析）
- ある文書が複数のトピックから成ることを仮定した言語モデル
  - 「実践 機械学習システム」であれば「機械学習」や「Python」などのトピックから成る

    →「機械学習」トピックの単語と「Python」トピックの単語から文書が製造される

”文章製造機”のリバースエンジニアリングを行いたい

トピックモデルの作成

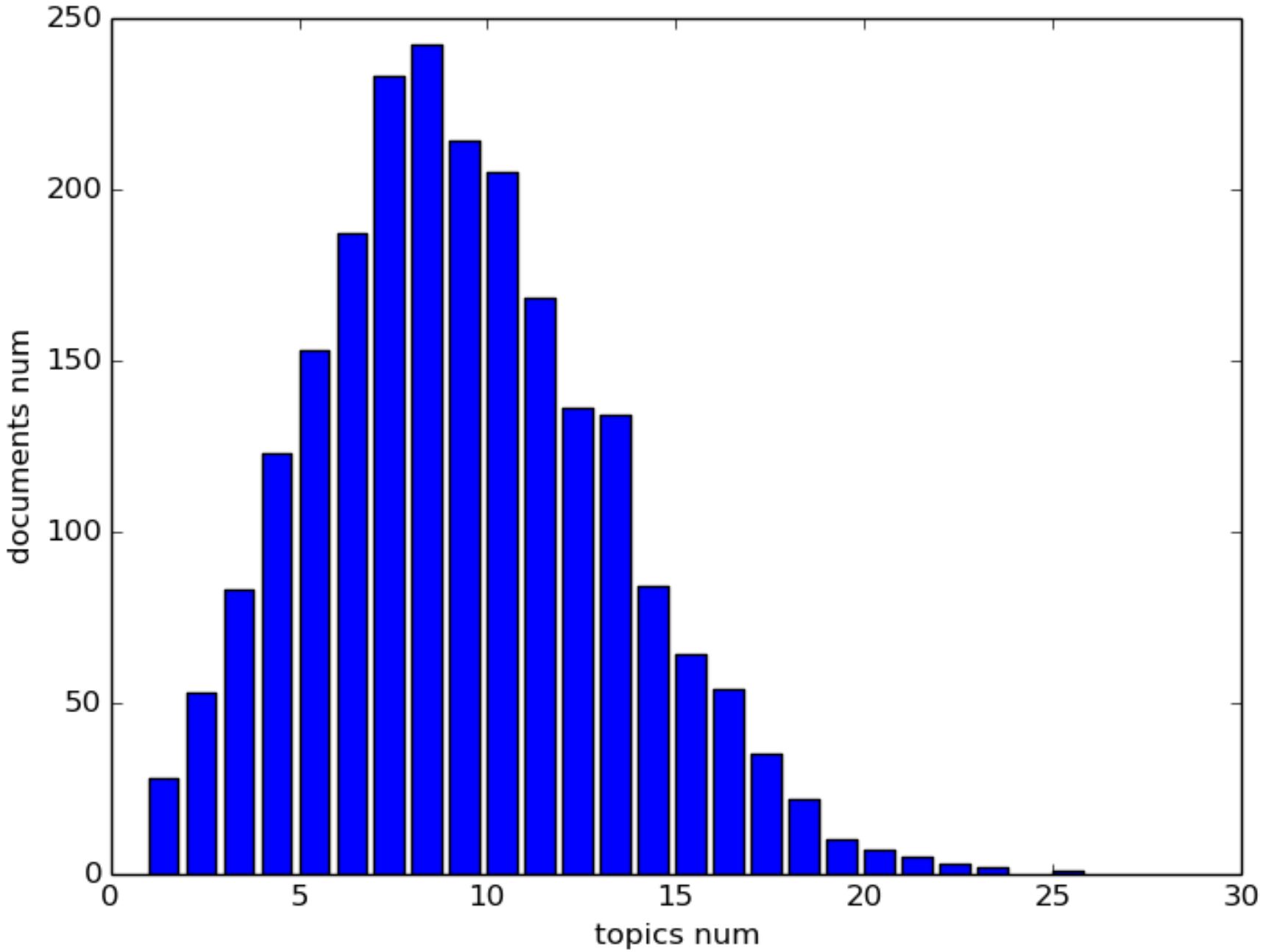
```
#トピックモデル作成
>>> from gensim import corpora, models, similarities
>>> corpus = corpora.BleiCorpus('ap.dat', 'vocab.txt')
>>> model = models.ldamodel.LdaModel(
                corpus, num_topics=100, id2word=corpus.id2word)
#確認
>>> topics = [model[c] for c in corpus]
>>> print topics[0]
```

[(20, 0.040915427848231017)
 (27, 0.029335386071693437)
 ･･･省略･･･
 (97, 0.015340637129850 05)]

- 出力フォーマットは(topic_index, topic_weight)
- ほとんどの文書は一部のトピックが割り当てられる
- トピックモデルは疎なモデル

＊データ＊

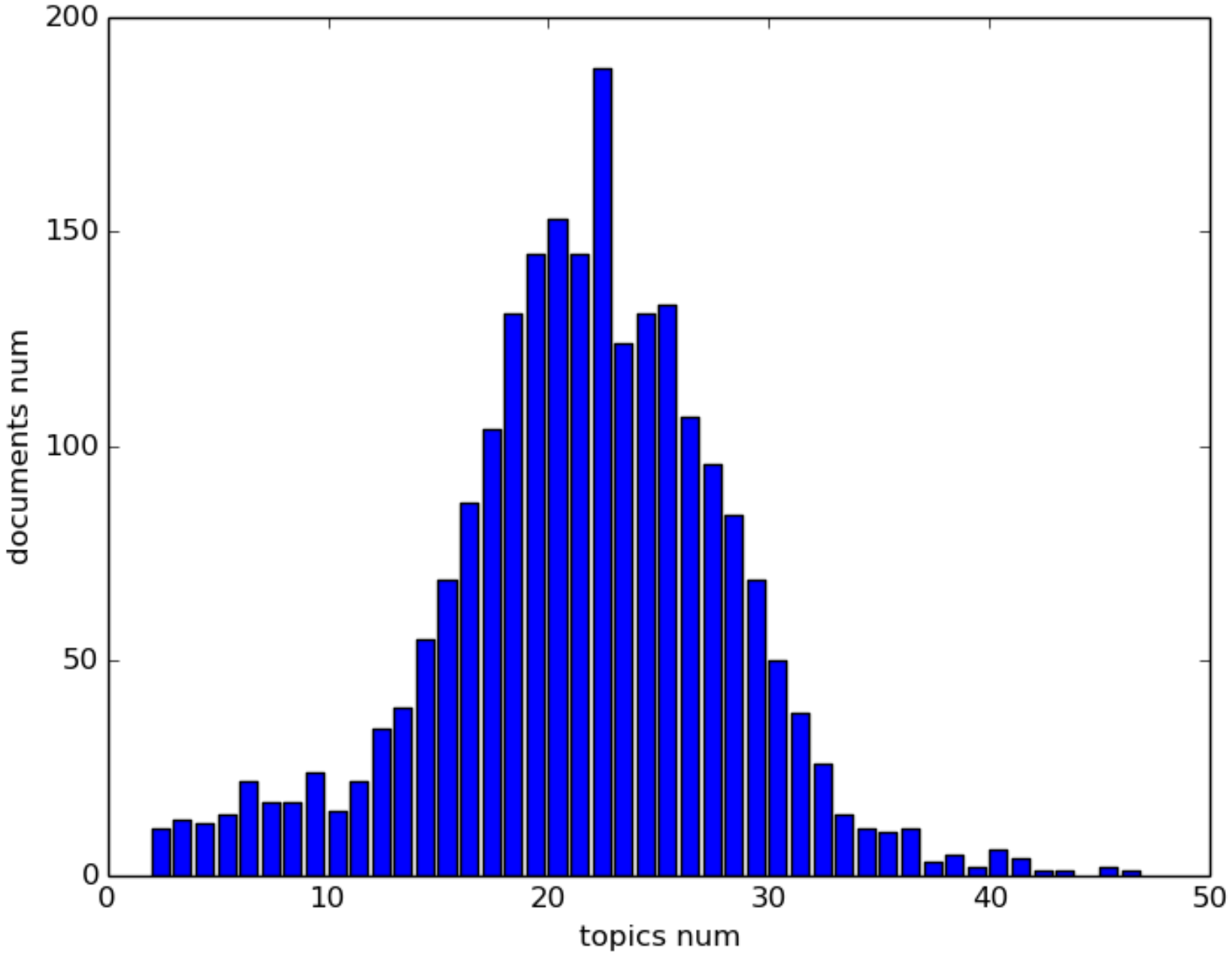
http://www.cs.princeton.edu/%7Eblei/lda-c/ap.tgz


250
200
150
100
50
0
documents num
0
5
10
15
20
25
30
topics num

alphaパラメータ

- トピックモデル生成関数のalphaパラメータを大きくすると, 文書が持つトピック数を増やすことができる

```
>>> model = models.ldamodel.LdaModel(
                corpus,
                num_topics=100,
                id2word=corpus.id2word,
                alpha=1)
```


200
150
100
50
0
documents num
0
10
20
30
40
50
topics num

トピックは何を表しているか

- 単語についての多項分布を示し, 各単語に対して確率を与える
- 確率の高い単語はそのトピックと関連性が高い

```
>>> print model.print_topic(0)

0.005*new + 0.005*percent + 0.004*states + 0.004*government +
0.004*i + 0.004*people + 0.004*theodore + 0.003*two +
0.003*united + 0.003*president

>>> print model.print_topic(1)

0.008*percent + 0.005*year + 0.004*i + 0.004*new +
0.004*president + 0.004*government + 0.003*states + 0.003*billion
+ 0.003*first + 0.003*million
```

トピック空間での類似度の比較

- 2つの文書が同じトピックについて論じられていれば、それは似ている文書である
- 2つの文書で共通する単語があまりない場合でも、トピックでの比較が可能

  →異なる表現で同じ意味の場合などに対応できる
- トピックベクトルは低次元（ここでは100次元）であり計算が高速

トピックベクトルの作成

```
#トピック行列
>>> dense = np.zeros((len(topics), 100), float)
>>> for ti, t in enumerate(topics):
...     for tj, v in t:
...             dense[ti, tj] = v

#距離行列
>>> from scipy.spatial import distance
>>> pairwise = distance.squareform(distance.pdist(dense))
```

トピックベクトルの作成

- 距離行列の対角要素に大きな値を設定する（同じ文書の距離を大きくする）

```
>>> largest = pairwise.max()
>>> for ti in range(len(topics)):
...     pairwise[ti, ti] = largest+1

#各文書に最も近い文書を見つける
>>> def closest_to(doc_id):
…         return pairwise[doc_id].argmin()
```