Topic Modeling

Gregory Lyon

Fall 2021

LDA

- If you read an article, you can typically get a sense of the main topics it covers
- Computers cannot
- Topic modeling: general term for tasks that assign each document to one or more topics, unsupervised usually
- Try to find groups of words (the topics) that appear togethr frequently

LDA

- Topic modeling requires that each document can be understood as a "mixture" of a subset of the topic
- A "topic" may not resemble a topic that we think of in everyday speech but that is resembles more the components extracted by PCA, which may or may not have a semantic meaning
- It may not be something WE would call a "topic"
- In finance article may have words like risk, regulation, assets in a topic
- In technology article, may have words like cloud, data, application, internet
- Words IN THESE GROUPS likely appear together, while it is less likely that cloud and assets appear together
- Aside from the nature of the articles, finance articles may be written by journalist A and tech articles by journalist B and so we may have another "topic" that captures words often used by journalist A and another for words often used by journalist B. These are not "topics" as we think of them but they are

LDA

- Latent (hidden) Dirichlet (distribution) Allocation (to assign words to topics)
- "The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words." (Blei, Ng, and Jordan 2003,)
- LDA assumes the following generative process for each document **w** in corpus D:
  - Choose $N \sim \text{Poisson}(\zeta)$
  - Choose $\theta \sim \text{Dir}(\alpha)$
  - For each of the N words w_n:
    - Choose a topic $z_n \sim \text{Multinomial}(\theta)$
    - Choose a word w_n from $\text{p}(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n

LDA

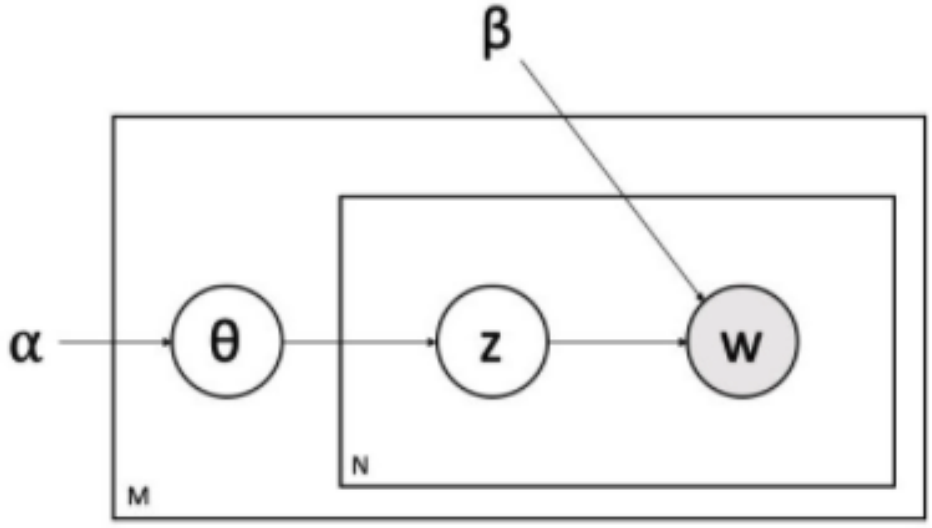
- Assumptions
  - The dimensionality k of the Dirichlet distribution (and so the dimensionality of the topic variable z) is assumed known and fixed
  - The word probabilities are parameterized by a k x V matrix β where $\beta_{ij} = p(w^j = 1 \mid z^i = 1)$, treated as fixed but to be estimated
  - Poisson assumption is not critical to anything that follows so we allows more realistic document length distributions as needed
  - N is independent of all the other data generating variable θ and $\mathbf{z}$. It is ancillary variable, we ignore its randomness

LDA

- Two iterative steps in LDA process: each rectangle (M and N)
- Generative process is carried out for every document in the corpus
- Therefore, "outer" step M represents iterating over each document
- And the "inner" step N is the process of iterating over words
- Circles: parameters, distributions, results
- w (shaded circle) is the word that is selected, the only known or observed data here
- We use w to work backward and execute the generative process
- α - hyperparameter for topic-document distribution (Dirichlet)
- β - distribution of words for each topic (Dirichlet)
- Z - the latent variable for the topic
- θ - the latent variable for distribution of topics for each document (multinomial)

LDA

- Role of:
  - α is to constrain the frequency of topics in documents
    - If α increases, documents become increasingly alike as number of topics increases
    - If α decreases, documents become dissimilar as number of topics in each document decreases
    - Key: the more they can share more topics, the more similar documents will appear
  - β is to constrain the frequency of words in topics



LDA

- Goal is to find arrangement of points in Dirichlet distribution that maximize the probability that the documents generated resemble the actual documents and use LDA to learn the mixture of topics in each document and the word mixture in each topic
- Data format: document-term matrix
- Looking at the text for probability distributions for hidden patterns/structure
- Topic model:
  - Document is a probability distribution (or a “mixture”) of topics
  - Topic is a probability distribution (“or”mixture") of words

LDA

- Topic modeling
  - Topic modeling is LDA applied to text
  - Topic models: methods to analyze the words of original texts to discover the themes that run through them, how those themes are connected to each other, and how they change over time (Blei 2012)
  - Do not require labeled data–the topics emerge from the analysis
  - Documents exhibit multiple topics

LDA

- Example document
  - In 2019, Thomas Panek, an avid runner and President & CEO of Guiding Eyes for the Blind, was at a Google hackathon and he brought up the question/challenge: "can we help a blind runner navigate (without a guide dog or running guide)?". He wasn't expecting more than a discussion about the possibilities. But the team at Google built a rough demo on the same day using the smartphone (camera) to track a line on the ground and providing audio feedback for the user to walk along the line. From there, they moved on to test the concept on an indoor track. Thomas wore a running belt with a phone strapped on it so that the rear camera was pointing forward. An app on the phone would run computer vision algorithms to track a line on the track and provide audio cues via bone-conducting headphones to Thomas. It worked really well indoors in a controlled environment.

LDA

- Example document
  - In 2019, Thomas Panek, an avid runner and President & CEO of Guiding Eyes for the Blind, was at a Google hackathon and he brought up the question/challenge: "can we help a blind runner navigate (without a guide dog or running guide)?". He wasn't expecting more than a discussion about the possibilities. But the team at Google built a rough demo on the same day using the smartphone (camera) to track a line on the ground and providing audio feedback for the user to walk along the line. From there, they moved on to test the concept on an indoor track. Thomas wore a running belt with a phone strapped on it so that the rear camera was pointing forward. An app on the phone would run computer vision algorithms to track a line on the track and provide audio cues via bone-conducting headphones to Thomas. It worked really well indoors in a controlled environment.
  - Topic 1: Technology - smartphone, camera, audio, feedback
  - Topic 2: Sports - avid runner, belt, indoor track

LDA

- Topic:
  - A (multinomial) distribution over a fixed vocabulary
    - The "Technology" topic has words like smartphone and camera that have high probabilities
    - The "Sports" topic has words like runner and track that havehigh probabilities
  - We assume the topics are specified before any data has been generated (model assumes topics are generated that, in turn, produce the documents)
  - For each document in our collection, we generate the words in two steps
    - Randomly choose a distribution over topics (e.g. 70% technology, 30% sports)
    - For each word in the document:
    - Randomly choose a topic from the distribution over topics in first step
    - Randomly choose a word from corresponding distribution over the vocabulary (e.g. each word in each theoretically comes from one of the topics)
  - Documents reflect **multiple** topics

LDA

- Key: all documents in the collection share the same set of topics, but each document contains those topics in different proportion

LDA

- Again, our goal is to discover topics from a collection of documents
- Documents: observed
- Topic structure–topics, per-document topic distributions, and per-document perword toic assignment–is latent, or the "hidden structure"
- When we implement it, we use the observed documents to infer the hidden topic structure
- Thus we engage in reverse engineering as we identify the hidden structure that would most likely to generate the observed collection of documents
- This reverse engineering (or generative process) defines a joint probability distribution over bot hthe observed and hidden random variables
- We use that joint distribution to compute conditional distributions of the hidden variables, given the observed variables (conditional distirbtion also known as posterior distribution)

LDA

- Utility of topic models
  - Inferred hidden structure resembles themes and structure of collection
  - Extremely difficult to do by hand
  - But topics generated are not always sensical
  - Need to apply domain knowledge to topics and evaluate

LDA

- Implement topic modeling in Pyton with `scikit-learn`
  - Clean and process text
  - Instantiate vectorizer, specify parameters
  - Apply vectorization to corpus column with text to create document-term matrix
  - Instantiate LDA model, specify parameters
  - Fit LDA model to vectorized text

LDA

- Implement topic modeling in Pyton with `scikit-learn`
  - Clean and process text
  - Instantiate vectorizer, specify parameters
  - Apply vectorization to corpus column with text to create document-term matrix
  - Instantiate LDA model, specify parameters
  - Fit LDA model to vectorized text

```
df
```

```
##   customer                        review
## 0    cust1   Great noisy bar fun time
## 1    cust2  noisy bar could not sleep
```

LDA

```
from sklearn.feature_extraction.text import CountVectorizer

# 1 Instantiate vectorizer
vec = CountVectorizer(stop_words="english")
```

LDA

```
from sklearn.feature_extraction.text import CountVectorizer

# 1 Instantiate vectorizer
vec = CountVectorizer(stop_words="english")

# 2 Apply vectorizer to create document-term matrix
X = vec.fit_transform(df["review"])
```

LDA

```
from sklearn.feature_extraction.text import CountVectorizer

# 1 Instantiate vectorizer
vec = CountVectorizer(stop_words="english")

# 2 Apply vectorizer to create document-term matrix
X = vec.fit_transform(df["review"])

# 3 Instantiate LDA
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_components=3) # n_compon
```

LDA

```
from sklearn.feature_extraction.text import CountVectorizer

# 1 Instantiate vectorizer
vec = CountVectorizer(stop_words="english")

# 2 Apply vectorizer to create document-term matrix
X = vec.fit_transform(df["review"])

# 3 Instantiate LDA
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_components=3) # n_compone

# 4 Fit LDA model to vectorized text
doc_topics = lda.fit_transform(X)
```