Clustering using Topic Models

Compiled by Sujatha Das, Cornelia Caragea

Credits for slides: Blei, Allan, Arms, Manning, Rai, Lund, Noble, Page.
Clustering

- Partition unlabeled examples into disjoint subsets of *clusters*, such that:
  - Examples within a cluster are very similar
  - Examples in different clusters are very different
- **Discover new categories** in an *unsupervised* manner
- We don't know the categories apriori, the goal is to identify if there are any underlying “similar” groups
“Hard” Clustering Example
Soft clustering for document collections

- Suppose the clusters correspond to topics in a given document collection
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters rather than assign a specific cluster to it
- Each instance is assigned a probability distribution across a set of discovered categories
  - A document is 80% politics and 20% religion
K-means vs. Topic Models

- **K-means**
  - Works on “points” in space
  - Performance depends on a similarity function between the points
  - Does not automatically give a generative view of documents. Probabilistically explain
    - What an “average” document on a “topic” look like?
    - What are the most likely words for a topic?
  - What if when multiple “modes” are available? For example, how to fold in citation links into clustering?
Topic Modeling

- From David Blei's page (Latent Dirichlet Allocation, a popularly used topic modeling algorithm)
  “Topic models are a suite of algorithms that uncover the hidden thematic structure in document collections. These algorithms help us develop new ways to search, browse and summarize large archives of texts.”

Over the last decade, topic models (pLSA, PCA, LDA) were extensively studied for various applications:

- Discovering topics from a corpus
- Evolution of topics with time
- Model connections between topics
- Find hierarchies of topics
- Model influential articles/authors
- Predict links between articles
- Organize and browse large corpora
Probabilistic Modeling

1. Data are assumed to be observed from a generative probabilistic process that includes hidden variables.
   - *In text, the hidden variables are the thematic structure.*

2. Infer the hidden structure using posterior inference
   - *What are the topics that describe this collection?*

3. Situate new data into the estimated model.
   - *How does a new document fit into the topic structure?*
Latent Dirichlet allocation (LDA)

Seeking Life’s Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here, two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today’s organisms can be sustained with just 235 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn’t be enough.

Although the numbers don’t match precisely, those predictions “are not all that far apart,” especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. “It may be a way of organizing any newly sequenced genome,” explains Aracdy Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an


SCIENCE • VOL. 272 • 24 MAY 1996

Simple intuition: Documents exhibit multiple topics.
Each topic is a distribution over words
Each document is a mixture of corpus-wide topics
Each word is drawn from one of those topics
The posterior distribution

In reality, we only observe the documents

The other structure are hidden variables

- Our goal is to infer the hidden variables
- I.e., compute their distribution conditioned on the documents

$$p(\text{topics, proportions, assignments} \mid \text{documents})$$
Implementations of LDA

There are many available implementations of topic modeling—

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA-C*</td>
<td>A C implementation of LDA</td>
</tr>
<tr>
<td>HDP*</td>
<td>A C implementation of the HDP (“infinite LDA”)</td>
</tr>
<tr>
<td>Online LDA*</td>
<td>A python package for LDA on massive data</td>
</tr>
<tr>
<td>LDA in R*</td>
<td>Package in R for many topic models</td>
</tr>
<tr>
<td>LingPipe</td>
<td>Java toolkit for NLP and computational linguistics</td>
</tr>
<tr>
<td>Mallet</td>
<td>Java toolkit for statistical NLP</td>
</tr>
<tr>
<td>TMVE*</td>
<td>A python package to build browsers from topic models</td>
</tr>
</tbody>
</table>

LDA is extendible

- LDA can be embedded in more complicated models, embodying further intuitions about the structure of the texts.
- LDA models can include syntax, authorship, word sense, dynamics, correlation, hierarchies, ...
- The data generating distribution can be changed.
- LDA models can be built for images, social networks, music, purchase histories, computer code, genetic data, click-through-data, neural spike trains, ...
- The LDA posterior can be used in creative ways
- It can be used for information retrieval, collaborative filtering, document similarity, visualization, ...
Understanding the output from basic LDA

Given a collection of documents, the basic LDA model uses the latent topic and term co-occurrences to estimate two quantities

- Topic-term matrix capturing term distributions for a topic
  \[ w_1 \ w_2 \ w_3 \ldots \]
  \[ T_1: 0.001 \ 0.02 \ 0.4\ldots \]

- Top words for a topic “reveal” the theme captured by that topic for human consumption

- Topic proportions for each document in the corpus
  \[ t_1 \ t_2 \ t_3 \ t_4 \]
  \[ D_1: <0.8, 0.2, 0, 0> \]
Understanding the output from basic LDA

Given a collection of documents, the basic LDA model uses the latent topic and term co-occurrences to estimate two quantities

- Topic-term matrix capturing term distributions for a topic
  \[
  w_1 \ w_2 \ w_3 \ \ldots
  \]
  \[
  T_1: \ 0.001 \ 0.02 \ 0.4 \ \ldots
  \]
- Top words for a topic “reveal” the theme captured by that topic for human consumption (low-dimensional projection)
- Topic proportions for each document in the corpus
  \[
  t_1 \ t_2 \ t_3 \ t_4
  \]
  \[
  D_1: \ <0.8, \ 0.2, \ 0, \ 0 >
  \]
Unsupervised learning in CiteSeer

1. Clustering authors for disambiguation
2. Clustering document collections into subject areas
3. Citation recommendation
4. Predicting Influential Authors
5. Analyzing topic trends over time
6. Improving ranking tasks (Homepage retrieval, Expertise search)
What does a researcher homepage look like?

A CS researcher homepage is a mixture of types of information (topics)
Output from LDA on homepages from DBLP

Table 4. Top words of topics related to homepages

talk slides invited part talks tutorial seminar summer book introduction
page home publications links contact personal list updated fax email
students graduate faculty research cse student undergraduate college current ph
member program committee chair teaching board editor courses state activities
Researcher homepages often contain their research information.

<table>
<thead>
<tr>
<th>data</th>
<th>multimedia</th>
<th>systems</th>
<th>design</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>content</td>
<td>distributed</td>
<td>circuits</td>
</tr>
<tr>
<td>databases</td>
<td>presentation</td>
<td>computing</td>
<td>systems</td>
</tr>
<tr>
<td>information</td>
<td>document</td>
<td>peer</td>
<td>digital</td>
</tr>
<tr>
<td>management</td>
<td>media</td>
<td>operating</td>
<td>signal</td>
</tr>
<tr>
<td>query</td>
<td>data</td>
<td>grid</td>
<td>vlsi</td>
</tr>
<tr>
<td>systems</td>
<td>documents</td>
<td>storage</td>
<td>ieee</td>
</tr>
<tr>
<td>xml</td>
<td>based</td>
<td>middleware</td>
<td>hardware</td>
</tr>
<tr>
<td>acm</td>
<td>hypermedia</td>
<td>system</td>
<td>fpga</td>
</tr>
<tr>
<td>vldb</td>
<td>video</td>
<td>scale</td>
<td>implementation</td>
</tr>
</tbody>
</table>
How can we use the output from LDA?

\[
\text{score}(s, t) = \sum_{w \in s} \phi_{w,t}
\]

The research description segment is extracted using

\[
p = \operatorname{argmax}_{t \in ST, s \in S} \text{score}(s, t)
\]

Fig. 3. Sample research description segments extracted from homepages

http://yann.lecun.com/

Note: the best way to reach me is by email or through Hong (I don’t check my voicemail very often).

My main research interests are Machine Learning, Computer Vision, Mobile Robotics, and Computational Neuroscience. I am also interested in Data Compression, Digital Libraries, the Physics of Computation, and all the applications of machine learning (Vision, Speech, Language, Document understanding, Data Mining, Bioinformatics).


PhD in Computer Science from the University of São Paulo (USP). Disciplinas 2010-1 Compiladores | Programação Research interests Machine learning (especially unsupervised learning, online learning), one-class classification, novelty detection, concept drift, natural computing and bio-inspired computing (especially evolutionary computation, genetic programming, genetic algorithms and artificial neural networks),
How can we use the output from LDA?

Improving homepage retrieval

- Train a re-ranking function on top of results from a search engine
References


Thank you!
Download Mallet from http://mallet.cs.umass.edu/download.php

sdas@ubuntu:~/setups/mallet-2.0.7$ ls
bin  class  lib  Makefile  sample-data  stopwords  test.seq  train.seq
build.xml  dist  LICENSE  pom.xml  src  test  train.model
sdas@ubuntu:~/setups/mallet-2.0.7$ echo $PATH
/home/sdas/setups/jdk1.7.0_51/bin:/home/sdas/setups/eclipse:/home/sdas/setups/mallet-2.0.7/bin:/usr/lib/lightdm/bin:
lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sdas@ubuntu:~/setups/mallet-2.0.7$ echo $JAVA_HOME
/home/sdas/setups/jdk1.7.0_51
sdas@ubuntu:~/setups/mallet-2.0.7$
After downloading and building Mallet, change to the Mallet directory and run the following command:

```
bins/mallet import-file --input /data/web/data.txt --output web.mallet
```

In this case, the first token of each line (whitespace delimited, with optional comma) becomes the instance name, the second token becomes the label, and all additional text on the line is interpreted as a sequence of word tokens. Note that the data in this case will be a vector of feature/value pairs, such that a feature consists of a distinct word type and the value is the number of times that word occurs in the text.

There are many additional options to the import-dir and import-file commands. Add the --help option to either of these commands to get a full list. Some commonly used options to either command are:

- **--keep-sequence.** This option preserves the document as a sequence of word features, rather than a vector of word feature counts. Use this option for sequence labeling tasks. The Mallet topic modeling toolkit also requires feature sequences rather than feature vectors.

- **--preserve-case.** Mallet by default converts all word features to lowercase.

- **--remove-stopwords.** This option tells Mallet to ignore a standard list of very common English adverbs, conjunctions, pronouns and prepositions. There are several other options related to stopword specification.

```
mallet import-file --input sample.txt --output sample.mallet --keep-sequence
```
null