Topic Models

Zach Alexander - UW Machine Learning

May 4, 2017

About Me

Principal Data Scientist – Salesforce.com

My team supports the entire Search organization.

- Search relevance/ranking
- Query Understanding
    - Entity prediction
    - Query expansion (synonyms, did you mean?, etc..)
    - Document understanding

Past

- Ph.D. in Applied Math from University of Colorado
  - Computational Topology (not machine learning)
- Data Scientist at Seagate Technology
  - Hard Disk Quality Modeling
- Certificate in Data Science @ UW (2013)
- Data Scientist at Microsoft
  - Real-time media @ Skype
  - Call quality / root-cause analysis
- Instructor at Galvanize
  - 12-week Data Science ‘boot camp’

Goals for Today

- Solid, intuitive understanding of SVD
  - Which leads to some intuition about all topic models
- Practical experience implementing topic models in R
  - Singular Value Decomposition
  - Non-negative Matrix Factorization
  - Latent Dirichlet Allocation
- Practical experience *interpreting* topic models in R
- NOT theoretical foundations of the three algorithms.

Singular Value Decomposition

- The most fundamental matrix decomposition

$$\underset{m\,x\,n}{M} = \underset{m\,x\,r}{U} * \underset{r\,x\,r}{\Sigma} * \underset{r\,x\,n}{V^T}$$

  - r is the *rank* of M (number of linearly independent columns/rows)
  - U is column-orthonormal and maps rows of M to *latent features*
  - Σ is diagonal, and measures the strength of the latent features.
  - V^T is row-orthonormal and maps latent features to columns of M

Nonnegative Matrix Factorization

- A more heuristic matrix factorization approach

$$\underset{m \times n}{M} = \underset{m \times k}{W} * \underset{k \times n}{H}$$

- k is the desired number of latent features
- W maps rows of M to latent features
- H maps latent features to columns of M
- Add the constraint that W, H >= 0.

Algorithm:

1. Hold W constant, solve for H
2. Clip H below at 0.
3. Hold H constant, Solve for W.
4. Clip W from below at 0.
5. Repeat…

Latent Dirichlet Allocation

- LDA is a probabilistic approach to topic modeling.
- *Assume the following:*
  - Each document is a specific mixture of K topics. For example, some document might be 30% politics, and 70% economics
  - Each topic assigns a probability to each word. For example:
    - P(democrat | topic='politics') = .07
    - P(playoffs | topic='politics') = .00003

- $P\left(w_j \middle| doc_i\right) = \Sigma_{k \in topics} P\left(w_j \middle| top_k\right) \; * P(top_k | doc_i)$
- This looks just like a matrix factorization. You need to know the mapping of words to topics and the mapping of topics to documents.
- 'Learn' the mappings via maximum likelihood estimation