W4995 Applied Machine Learning

LSA & Topic Models

04/08/19

Andreas C. Müller

Beyond Bags of Words

Limitations of bag of words:

- Semantics of words not captured
- Synonymous words not represented
- Very distributed representation of documents

Latent Semantic Analysis (LSA)

- Reduce dimensionality of data.
- Can't use PCA: can't subtract the mean (sparse data)
- Instead of PCA: Just do SVD, truncate.
- "Semantic" features, dense representation.
- Easy to compute - convex optimization

LSA with Truncated SVD

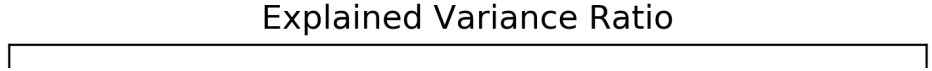
```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer(stop_words="english", min_df=4)
X_train = vect.fit_transform(text_train)
X_train.shape
```

```
(25000, 30462)
```

```
from sklearn.decomposition import TruncatedSVD
lsa = TruncatedSVD(n_components=100)
X_lsa = lsa.fit_transform(X_train)
lsa.components_.shape
```

```
(100, 30462)
```

```
plt.semilogy(lsa.explained_variance_ratio_)
```



First Six eigenvectors

First 6 eigenvectors of LSA

Scale before LSA

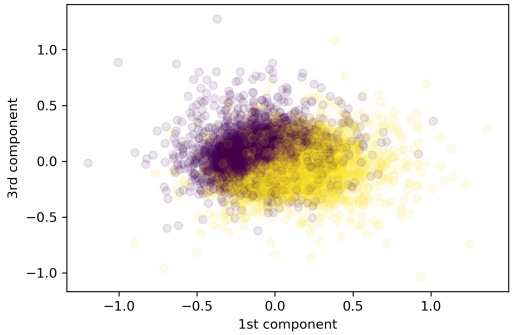
```
from sklearn.preprocessing import MaxAbsScaler
scaler = MaxAbsScaler()
X_scaled = scaler.fit_transform(X_train)

lsa_scaled = TruncatedSVD(n_components=100)
X_lsa_scaled = lsa_scaled.fit_transform(X_scaled)
```

Eigenvectors after scaling

First 6 eigenvectors of LSA after scaling

Some Components Capture Sentiment

```
plt.scatter(X_lsa_scaled[:, 1], X_lsa_scaled[:, 3], alpha=.1, c=y_train)
```




```
lr_lsa = LogisticRegression(C=100).fit(X_lsa_scaled[:, :10], y_train)
lr_lsa.score(X_test_lsa_scaled[:, :10], y_test)
```

```
0.827
```



```
lr_lsa.score(X_lsa_scaled[:, :10], y_train)
```

Topic Models

Motivation

- Each document is created as a mixture of topics
- Topics are distributions over words
- Learn topics and composition of documents simultaneously
- Unsupervised (and possibly ill-defined)

NMF for topic models

NMF objectives
NMF Loss and algorithm

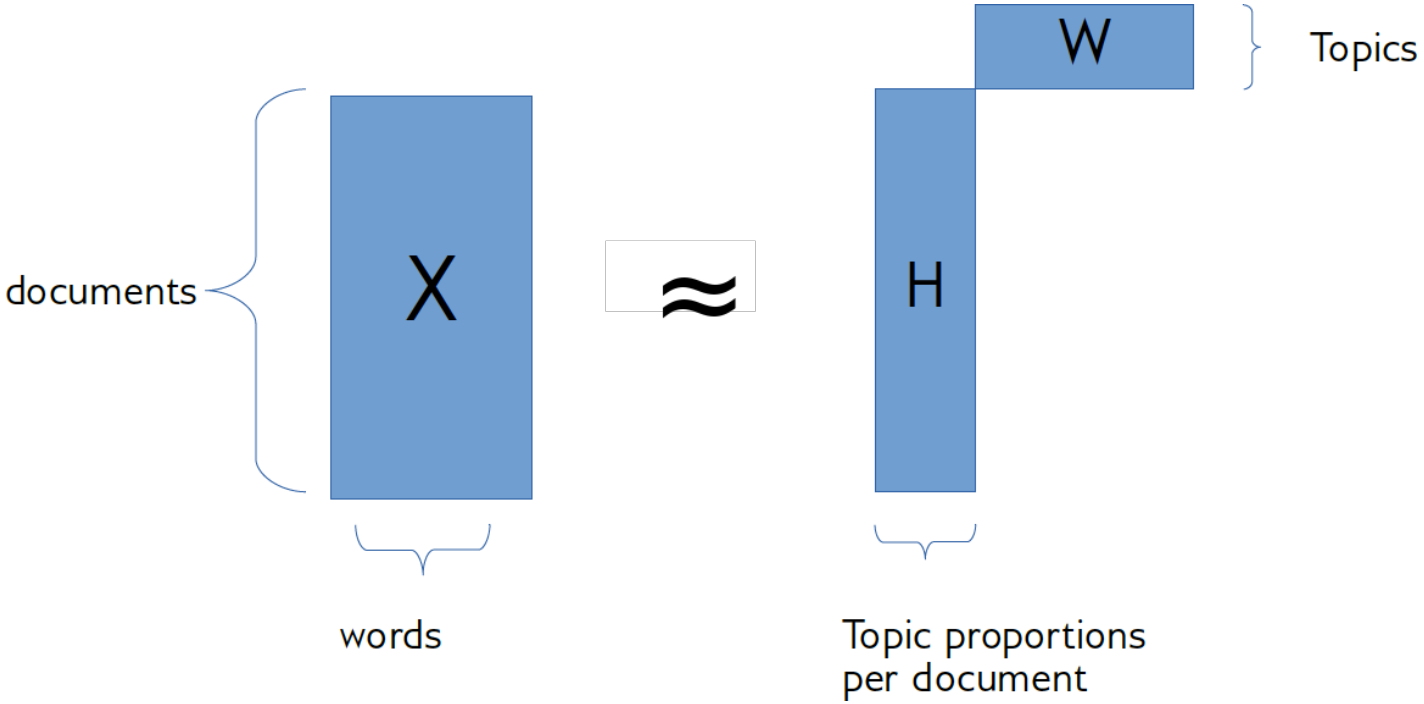
Frobenius loss / squared loss ($Y = HW$):

$$l_{\text{frob}}(X, Y) = ||X - Y||^2_{\text{Fro}} = \sum_{i,j}(X_{ij} - Y_{ij})^2$$

Kulback-Leibler (KL) divergence:

$$l_{\text{KL}}(X, Y) = \sum_{i,j} X_{ij} \log\left(\frac{X_{ij}}{Y_{ij}}\right) - X_{ij} + Y_{ij}$$

NMF for topic models



NMF on Scaled Data

```
nmf_scale = NMF(n_components=100, verbose=10, tol=0.01)
nmf_scale.fit(X_scaled)
```

NMF components without scaling

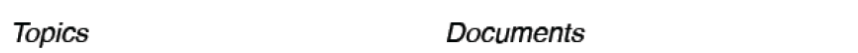
NMF with tfidf

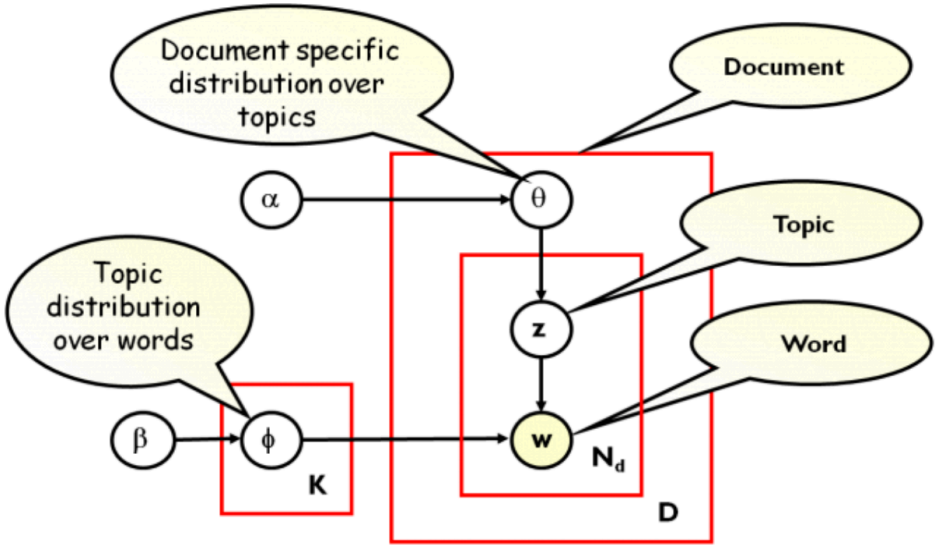
NMF with tfidf and 10 components

Latent Dirichlet Allocation (the other LDA)

(Not Linear Discriminant Analysis)

The LDA Model





- For each topic k, draw $\phi_k \sim \text{Dirichlet}(\beta), k = 1 \ldots K$
- For each document d draw $\theta_d \sim \text{Dirichlet}(\alpha), d = 1 \ldots D$
- For each word i in document d:
  - Draw a topic index $z_{di} \sim \text{Multinomial}(\theta_d)$
  - Draw the observed word $w_{ij} \sim \text{Multinomial}(\phi_z)$

Multinomial

It models the probability of counts for rolling a k-sided die n times.

wikipedia

$$P(x_1, \ldots, x_k | n, p_1, \ldots, p_k) = \frac{n!}{x_1! \ldots x_k!} p_1^{x_1} \ldots p_k^{x^k}$$

Multinomial

It models the probability of counts for rolling a k-sided die n times.

wikipedia

$$P(x_1, \dots, x_k | n, p_1, \dots, p_k) = \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x^k}$$

Dirichlet

$$P(x_1, \dots, x_k | \alpha_1, \dots \alpha_k) = \frac{1}{B(\alpha)} x_1^{\alpha_1 - 1} \dots x_k^{\alpha_k - 1}$$

Conjugate Prior

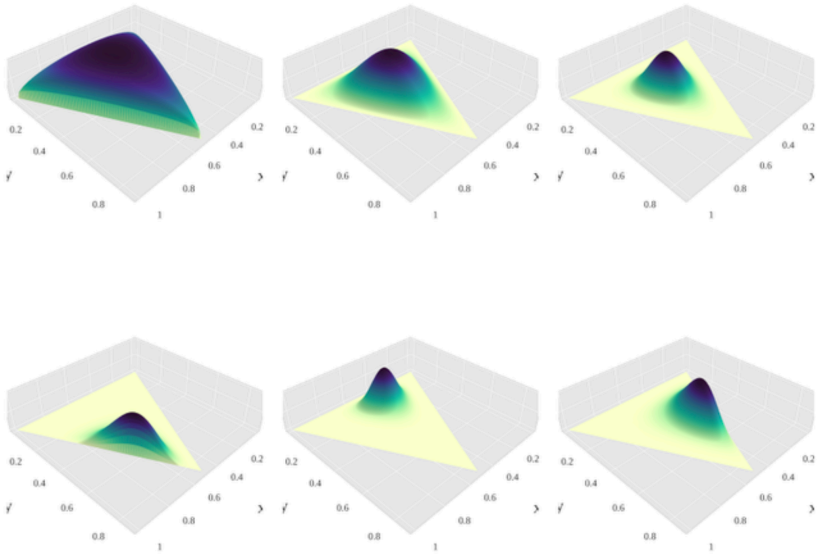
- Prior is called "conjugate" of the posterior has the same form as prior

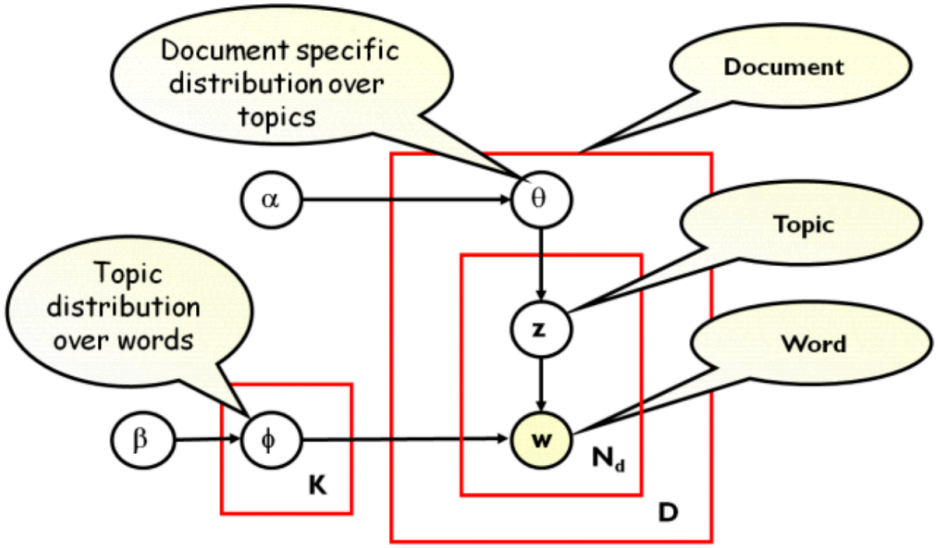
$$P(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta')p(\theta')d\theta'}$$

Dirichlet Distributions

PDF: $\frac{1}{B(\alpha)} \Pi_{i=1}^{K} x_i^{\alpha_i - 1}$

Mean: $E[X_i] = \frac{\alpha_i}{\sum_k \alpha_k}$





- For each topic k, draw $\phi_k \sim \text{Dirichlet}(\beta), k = 1 \ldots K$
- For each document d draw $\theta_d \sim \text{Dirichlet}(\alpha), d = 1 \ldots D$
- For each word i in document d:
  - Draw a topic index $z_{di} \sim \text{Multinomial}(\theta_d)$
  - Draw the observed word $w_{ij} \sim \text{Multinomial}(\beta_z)$

Two Schools (of solvers)

Gibbs Sampling

- Implements MCMC
- Standard procedure for any probabilistic model
- Very accuracte
- Very slow

Variational Inference

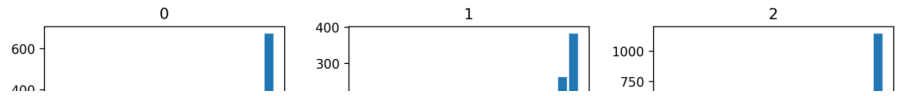
- Extension of expectation-maximization algorithm
- Deterministic
- fast(er)
- Less accurate solutions
- Championed by Dave Blei

Pick a solver

- “Small data” (<= 10k? Documents):
  - Gibbs sampling (lda package, MALLET in Java)
- “Medium data” (<= 1M? Documents):
  - Variational Inference (scikit-learn default)
- “Large Data” (>1M? Documents):
  - Stochastic Variational Inference
  - SVI allows online learning (partial_fit)
- Edward by Dustin Tran: http://edwardlib.org/
  - Tensor-flow based framework for stochastic variational inference.
  - Now Tensorflow Probability: https://www.tensorflow.org/probability

```
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_components=10, learning_method="batch")
X_lda = lda.fit_transform(X_train)
```

```
lda100 = LatentDirichletAllocation(n_components=100, learning_method="batch")
X_lda100 = lda100.fit_transform(X_train)
```



topic 31	topic 3	topic 38	topic 4	topic 29	topic 57	topic 60	topic 10
movie	film	movie	movie	film	series	film	film
just	just	story	funny	films	episode	horror	good

Hyper-Parameters

- α (or θ) = doc_topic_prior
- β (or η) = topic_word_prior
- Both dirichlet distributions
- Large value → more dispersed
- Small value → more concentrated

Rough overview of MCMC and Gibbs sampling

Markov-Chain Monte-Carlo (MCMC)

Goal: Sample from complex distribution $p(X_1, \ldots X_n)$.

Monte Carlo algorithm: "correct in expectation" / "correct on average"

Markov Chain: a sequence of probability distribution, each depending only on the previous state.

Markov Chain Monte Carlo: a sequence of probability distributions so that on average you sample from the target distribution.

(The stationary distribution of the Markov chain is the target distribution p)

Gibbs Sampling

Goal: Sample from complex distribution $p(X_1, \ldots X_n)$.

Assumption: Can sample from conditional $p(X_i | X_{j \neq i})$

Gibbs Sampling

Goal: Sample from complex distribution $p(X_1, \ldots X_n)$.

Assumption: Can sample from conditional $p(X_i | X_{j \neq i})$

Initialize

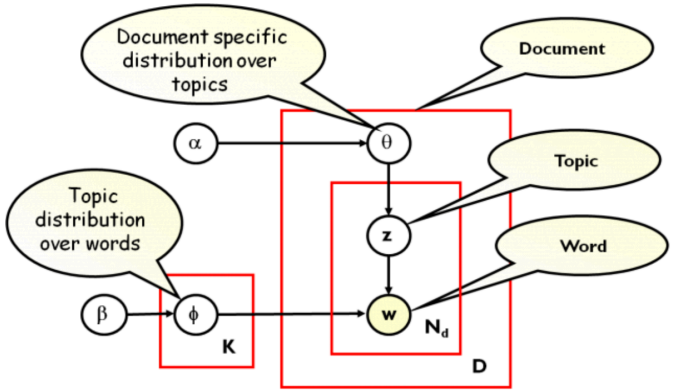
Start from random initialization $(x_1^0, \ldots, x_n^0)$

Construct x^{k+1} from x^k:

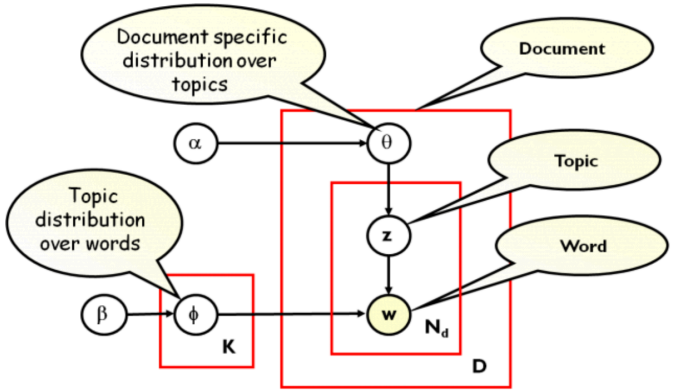
$$x_0^{k+1} \propto p(X_i | x_1^k, \ldots, x_n^k)$$

$$x_i^{k+1} \propto p(X_i | x_0^{k+1}, \ldots, x_{i-1}^{k+1}, x_{i+1}^k, \ldots, x_n^k)$$

(Block) Gibbs Sampling for LDA



(Block) Gibbs Sampling for LDA



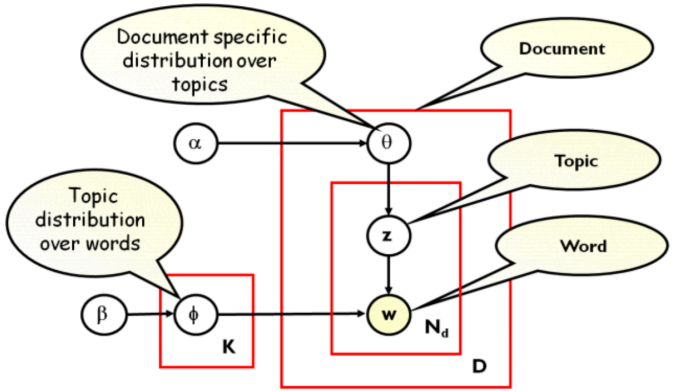
Updates sampled from:

$$p(z_{iv} = k \mid \theta_i, \phi_k) \propto \exp(\log \theta_{ik} + \log \phi_{k,y_{iv}})$$

$$p(\theta_i \mid z_{iv} = k, \phi_k) = \text{Dir}(\alpha + \textstyle\sum_l \mathbb{I}(z_{il} = k))$$

$$p(\phi_k \mid z_{iv} = k, \theta_i) = \text{Dir}(\beta + \textstyle\sum_i \sum_l \mathbb{I}(y_{il} = v, z_{il} = k))$$

Collapsed Gibbs Sampling for LDA



Sample only $p(z_i = k|\mathbf{z}_{-i})$

Compute θ, ϕ from $\mathbf{z}$

Further Reading

- Rethinking LDA: Why Priors Matter - Hanna Wallach
- LDA Revisited: Entropy, Prior and Convergence – Zhang et. al.

Questions ?