TOPIC MODELING
Models for text data

Given text data we want to:

- Organize
- Visualize
- Summarize
- Search
- Predict
- Understand

Topic models allow us to

1. Discover themes in text
2. Annotate documents
3. Organize, summarize, etc.
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 250 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 Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

A probabilistic topic model

- Learns distributions on words called “topics” shared by documents
- Learns a distribution on topics for each document
- Assigns every word in a document to a topic
However, none of these things are known in advance and must be learned

- Each document is treated as a “bag of words”
- Need to define (1) a model, and (2) an algorithm to learn it
- We will review the standard topic model, but won’t cover inference
There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.
There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.
There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.
Latent Dirichlet Allocation

There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.
Latent Dirichlet Allocation

There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.

The generative process for LDA is:

1. Generate each topic, which is a distribution on words
   \[ \beta_k \sim \text{Dirichlet}(\gamma), \quad k = 1, \ldots, K \]
2. For each document, generate a distribution on topics
   \[ \theta_d \sim \text{Dirichlet}(\alpha), \quad d = 1, \ldots, D \]
3. For the \( n \)th word in the \( d \)th document,
   a) Allocate the word to a topic, \( c_{dn} \sim \text{Discrete}(\theta_d) \)
   b) Generate the word from the selected topic, \( x_{dn} \sim \text{Discrete}(\beta_{c_{dn}}) \)
LDA outputs two main things:

1. A set of distributions on words (topics). Shown above are ten topics from NYT data. We list the ten words with the highest probability.

2. A distribution on topics for each document (not shown). This indicates its thematic breakdown and provides a compact representation.
Q: For a particular document, what is $P(x_{dn} = i | \beta, \theta_d)$?

A: Find this by integrating out the cluster assignment,

$$
P(x_{dn} = i | \beta, \theta) = \sum_{k=1}^{K} P(x_{dn} = i, c_{dn} = k | \beta, \theta_d)
$$

$$
= \sum_{k=1}^{K} P(x_{dn} = i, c_{dn} = k | \beta, \theta_d) P(c_{dn} = k | \theta_d)
$$

Let $B = [\beta_1, \ldots, \beta_K]$ and $\Theta = [\theta_1, \ldots, \theta_D]$, then $P(x_{dn} = i | \beta, \theta) = (B\Theta)_{id}$

In other words, we can read the probabilities from a matrix formed by taking the product of two matrices that have nonnegative entries.
NONNEGATIVE MATRIX FACTORIZATION
LDA can be thought of as an instance of nonnegative matrix factorization.

- It is a probabilistic model.
- Standard inference algorithms use techniques not taught in this course.

We will discuss two other related models and their algorithms. These two models are called nonnegative matrix factorization (NMF).

- They can be used for the same tasks LDA can be used for.
- Though nonnegative matrix factorization is a general technique, “NMF” usually just refers to the following two methods.
Nonnegative matrix factorization

- Data $X$ has nonnegative entries. None missing, but likely many zeros.
- The learned factorization $W$ and $H$ also have nonnegative entries.
- The value $X_{ij} \approx \sum_k W_{ik}H_{kj}$, but we don’t write this in terms of vectors.
- But later we will interpret the output in terms of column vectors.
What is some data that can constitute $X$?

- **Text data:**
  - Word term frequencies
  - $X_{ij}$ contains the number of times word $i$ appears in document $j$.

- **Image data:**
  - Face identification data sets
  - Put each vectorized $N \times M$ image of a face on a column of $X$.

- **Other discrete grouped data:**
  - Quantize *continuous* sets of features using K-means
  - $X_{ij}$ counts how many times group $j$ uses cluster $i$.
  - For example: group = song, features = $d \times n$ spectral information matrix
NMF minimizes one of the following two objective functions over $W$ and $H$.

**Choice 1: Squared error objective**

$$\|X - WH\|^2 = \sum_i \sum_j (X_{ij} - (WH)_{ij})^2$$

**Choice 2: Divergence objective**

$$D(X\|WH) = -\sum_i \sum_j [X_{ij} \ln(WH)_{ij} - (WH)_{ij}]$$

- Both add the constraint that $W$ and $H$ have nonnegative values.
- NMF uses a fast, simple algorithm for these two objectives.
Recall what we should look for in minimizing an objective \( \min_h F(h) \):

1. A way to generate a sequence of values \( h^1, h^2, \ldots \), such that

\[
F(h^1) \geq F(h^2) \geq F(h^3) \geq \cdots
\]

2. Convergence of the sequence to a local minimum of \( F \)

The following algorithms fulfill these requirements. In this case:

- Minimization is done via an “auxiliary function.”
- Leads to a “multiplicative algorithm” for \( W \) and \( H \).
- We’ll skip details (see reference).

MULTIPLICATIVE UPDATE FOR $\|X - WH\|^2$

**Problem**: Minimize the squared error $\|X - WH\|^2 = \sum_{ij} (X_{ij} - (WH)_{ij})^2$, subject to $W_{ik}, H_{kj} \geq 0$.

**Algorithm**

- Randomly initialize $H$ and $W$ with nonnegative values.
- Iterate the following, first for all values in $H$, then all in $W$:

$$H_{kj} \leftarrow H_{kj} \frac{(W^T X)_{kj}}{(W^T WH)_{kj}},$$

$$W_{ik} \leftarrow W_{ik} \frac{(XH^T)_{ik}}{(WHH^T)_{ik}},$$

until the change in $\|X - WH\|^2$ is “small.”
A visualization that may be helpful. Use the color-coded definition above.

- Use element-wise multiplication/division across the three “columns.”
- Use matrix multiplication within each outlined box.

\[
X_{ij} \sim N\left(\sum_k W_{ik} H_{kj}, \sigma^2\right)
\]

Since \( X_{ij} \geq 0 \) (and often isn’t continuous), we are making an incorrect modeling assumption. Nevertheless, as with PMF it still works well.
MULTIPLICATIVE UPDATE FOR $D(X\|WH)$

**Problem:** Minimize divergence, $D(X\|WH) = \sum_{ij} \left[ X_{ij} \ln \frac{1}{(WH)_{ij}} + (WH)_{ij} \right]$, subject to $W_{ik}, H_{kj} \geq 0$.

**Algorithm**

- Randomly initialize $H$ and $W$ with nonnegative values.

- Iterate the following, first for all values in $H$, then all in $W$:

  $$H_{kj} \leftarrow H_{kj} \frac{\sum_i W_{ik} X_{ij} / (WH)_{ij}}{\sum_i W_{ik}},$$

  $$W_{ik} \leftarrow W_{ik} \frac{\sum_j H_{kj} X_{ij} / (WH)_{ij}}{\sum_j H_{kj}},$$

  until the change in $D(X\|WH)$ is “small.”
Visualizing the update for the divergence penalty is more complicated.

- Use the color-coding definition in the upper right.
- This is the data matrix “dot-divided” by the approximation of it.

1. Normalize the rows of this transposed matrix so they sum to one.
2. Normalize the columns of this matrix so they sum to one.
The maximum likelihood interpretation of the divergence penalty is more interesting than for the squared error penalty.

Model the data as independent Poisson random variables

\[ X_{ij} \sim \text{Pois}((WH)_{ij}), \quad \text{Pois}(x|\lambda) = \frac{\lambda^x}{x!} e^{-\lambda}, \quad x \in \{0, 1, 2, \ldots\} \]

Then the divergence penalty is the maximum likelihood for \( W \) and \( H \).

\[
-D(X||WH) = \sum_{ij} [X_{ij} \ln((WH)_{ij}) - (WH)_{ij}]
\]

\[
= \sum_{ij} \ln P(X_{ij}|W, H) + \text{constant}
\]

We use: 
\[
P(X|W, H) = \prod_{ij} P(X_{ij}|W, H) = \prod_{ij} \text{Pois}(X_{ij}|(WH)_{ij}).
\]
As discussed, NMF can be used for topic modeling. In fact, the divergence penalty is closely related mathematically to LDA (can be shown).

Step 1. Form the term-frequency matrix $X$. ($X_{ij} = \# \text{ times word } i \text{ in doc } j$)

Step 2. Run NMF to learn $W$ and $H$ using $D(X \| WH)$ penalty

Step 3. As an added step, after Step 2 is complete, for $k = 1, \ldots, K$

1. Set $a_k = \sum_i W_{ik}$
2. Divide $W_{ik}$ by $a_k$ for all $i$
3. Multiply $H_{kj}$ by $a_k$ for all $j$

Notice that this is does not change the matrix multiplication $WH$.

Interpretation: The $k$th column of $W$ can be interpreted as the $k$th topic. The $j$th column of $H$ can be interpreted as how much document $j$ uses each topic.
For face modeling, put the face images along the columns of $X$ and factorize. Show columns of $W$ as image. Compare this with K-means and SVD.

K-means (i.e., VQ): Equivalent to each column of $H$ having a single 1. K-means learns averages of full faces.
NMF and Face Modeling

For face modeling, put the face images along the columns of $X$ and factorize. Show columns of $W$ as image. Compare this with K-means and SVD.

PCA

SVD: Finds the singular value decomposition of $X$. Results not interpretable because of ± values and orthogonality constraint.
NMF AND FACE MODELING

For face modeling, put the face images along the columns of $X$ and factorize. Show columns of $W$ as image. Compare this with K-means and SVD.

NMF learns “parts-based” representation. Each column captures something interpretable. This is a result of the nonnegativity constraint.