Mathematics of Big Data, I

Lecture 9: Recommender Systems, Collaborative Filtering, and Topic Modeling based on Non-Negative Matrix Factorization.

Weiqing Gu
Professor of Mathematics
Director of the Mathematics Clinic

Harvey Mudd College
Summer 2017

Topics Today

- **Introduction to Recommender Systems.**
- **Collaborative Filtering.**
- **Non-Negative Matrix Factorization.**
- **Using Non-Negative Matrix Factorization for Topic Modeling.**

Topics Today

- Introduction to Recommender Systems.
- Collaborative Filtering.
- Non-Negative Matrix Factorization.
- Using Non-Negative Matrix Factorization for Topic Modeling.

Why Recommender Systems?

- Recommender Systems have Important Machine Learning Applications.
  - When I on the clinic recruiting trips, I often notice that many companies try to build better recommender systems, such as Apple, Amazon, Netflix, Yelp, and eBay.
  - E.g.s: Amazon/Netflix will recommend books/movies on their webpage for you to buy/watch based on what you had purchased/watched.
  - Good recommendation systems will provide substantial revenues for the companies who use them.
- Big ideas of Machine learning: auto learn important data features instead of hard hand coding. Recommender systems is one of them, it can do auto learning!

Recommendation System Problem Formulation

Example: Predicting movie ratings

User rates movies using one to five stars

3 romantic movies

2 action movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last				
Romance forever				
Cute puppies of love				
Nonstop car chases				
Swords vs. karate				

Recommendation System Problem Formulation

Example: Predicting movie ratings

0

User rates movies using one to five stars

	Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
3 romantic movies	Love at last	5	5	0	0
	Romance forever	5	?	?	0
	Cute puppies of love	?	4	0	?
2 action movies	Nonstop car chases	0	0	5	4
	Swords vs. karate	0	0	5	?

n_u = no. users

n_m = no. movies

$r(i,j)$ = 1 if user j has rated movie i

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j) = 1$)

- ? = did not rate
- In this example $n_u = 4$,
- $n_m = 5$.

$y^{(i,j)}$ is between 0 and 5.

Goal: Given the data r(i,j) and $y^{(i,j)}$, fill out the question marks.

- Do it automatically!

Content Based Recommendations

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	5 ?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

Example: Predicting movie ratings

- User rates movies using ~~one~~ zero to five stars

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	? 4.5	? 0	0
Cute puppies of love	? 5	4	0	? 0
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	? 4

$n_u = 4$ $\quad$ $n_m = 5$

n_u = no. users

n_m = no. movies

$r(i,j)$ = 1 if user j has rated movie i

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j) = 1$)

$0, \ldots, 5$

The goal of the Recommender system is to solve the following problem:

- Given the r(i, j) and $y^{(i,j)}$. How to predict the values for the question marks?

$n_u = 4$, $n_m = 5$

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

Use machine learning to build content based recommender system

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

$n_u = 4$, $n_m = 5$

Features
e.g. x_1 measures the degree of romantic.

Content-based recommender systems

	Movie		Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
X(1).	Love at last	1.	5	5	0	0	0.9	0
X(2).	Romance forever	2.	5	?	?	0	1.0	0.01
X(3).	Cute puppies of love	3.	?	4	0	?	0.99	0
X(4).	Nonstop car chases	4.	0	0	5	4	0.1	1.0
X(5).	Swords vs. karate	5.	0	0	5	?	0	0.9

$x_0 = 1$

No romance, most action.

Now each movie will have a feature vector. $x(1) = [1, 0.9, 0]^T$. Similarly for other x(i).

Let n = number of features, not include x_0.
Here n =2.

$n_u = 4$, $n_m = 5$

$x^{(1)} = \begin{bmatrix} 1 \\ 0. \\ 0 \end{bmatrix}$

Content-based recommender systems

$x_0 = 1$

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
$x^{(1)}$ Love at last 1	5	5	0	0	0.9	0
$x^{(2)}$ Romance forever 2	5	?	?	0	1.0	0.01
$x^{(3)}$ Cute puppies of love 3	? 4.95	4	0	?	0.99	0
Nonstop car chases 4	0	0	5	4	0.1	1.0
$x^{(5)}$ Swords vs. karate 5	0	0	5	?	0	0.9

$n = 2$

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

$\theta^{(j)} \in \mathbb{R}^{n+1}$

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix} \longleftrightarrow \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \qquad (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

Basically it is a linear regression problem for learning $\theta^{(j)}$

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)
$y^{(i,j)} =$ rating by user j on movie i (if defined)

$\theta^{(j)}$ = parameter vector for user j
$x^{(i)}$ = feature vector for movie i
For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ = no. of movies rated by user j
To learn $\theta^{(j)}$:

Optimization objective

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

To learn $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \boxed{\sum_{j=1}^{n_u}} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

n_u = no. users

Algorithm for Recommender System

Optimization algorithm:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} \text{ (for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right) \text{ (for } k \neq 0)$$

Algorithm for Recommender System

Optimization algorithm:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \underbrace{\frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2}_{J(\theta^{(1)},\ldots,\theta^{(n_u)})}$$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} \quad \underline{(\text{for } k = 0)}$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right)}_{\frac{\partial}{\partial \theta^{(j)}} J(\theta^{(1)},\ldots,\theta^{(n_u)})} \quad \underline{(\text{for } k \neq 0)}$$

Topics Today

- Introduction to Recommender Systems.
- **Collaborative Filtering.**
- Non-Negative Matrix Factorization.
- Using Non-Negative Matrix Factorization for Topic Modeling.

Collaborative Filtering

Problem motivation:

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

Problem motivation

$x_0 = 1$

What if the values of x_1 and x_2 are missing?

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

Mathematically, it is the **dual problem** of our previous content based recommendation.

Optimization Algorithm for the dual problem

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(1)}, \ldots, x^{(n_m)}$:

$$\min_{x^{(1)}, \ldots, x^{(n_m)}} \frac{1}{2} \boxed{\sum_{i=1}^{n_m}} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

n_m = no. movies

Collaborative Filtering

Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings),

can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$,

can estimate $x^{(1)}, \ldots, x^{(n_m)}$

$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \ldots$

Guessing

Collaborative Filtering Optimization Objective

Given $x^{(1)}, \ldots, x^{(n_m)}$, estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \boxed{\sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2} + \boxed{\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2}$$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, estimate $x^{(1)}, \ldots, x^{(n_m)}$:

$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2} \boxed{\sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2} + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Minimizing $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)}) = \frac{1}{2} \boxed{\sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2} + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \boxed{\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2}$$

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)})$$

$\cancel{x_0 = 1}$

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

~~$x_0 = 1$~~ $x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^n$

Collaborative filtering algorithm

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \underbrace{\left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)} \right)}_{\frac{\partial}{\partial x_k^{(i)}} J(\ldots)}$$

$$\theta_k^{(j)} := \underline{\theta_k^{(j)}} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)} \right)}$$

~~θ_0~~
θ_1
$\vdots$
θ_n

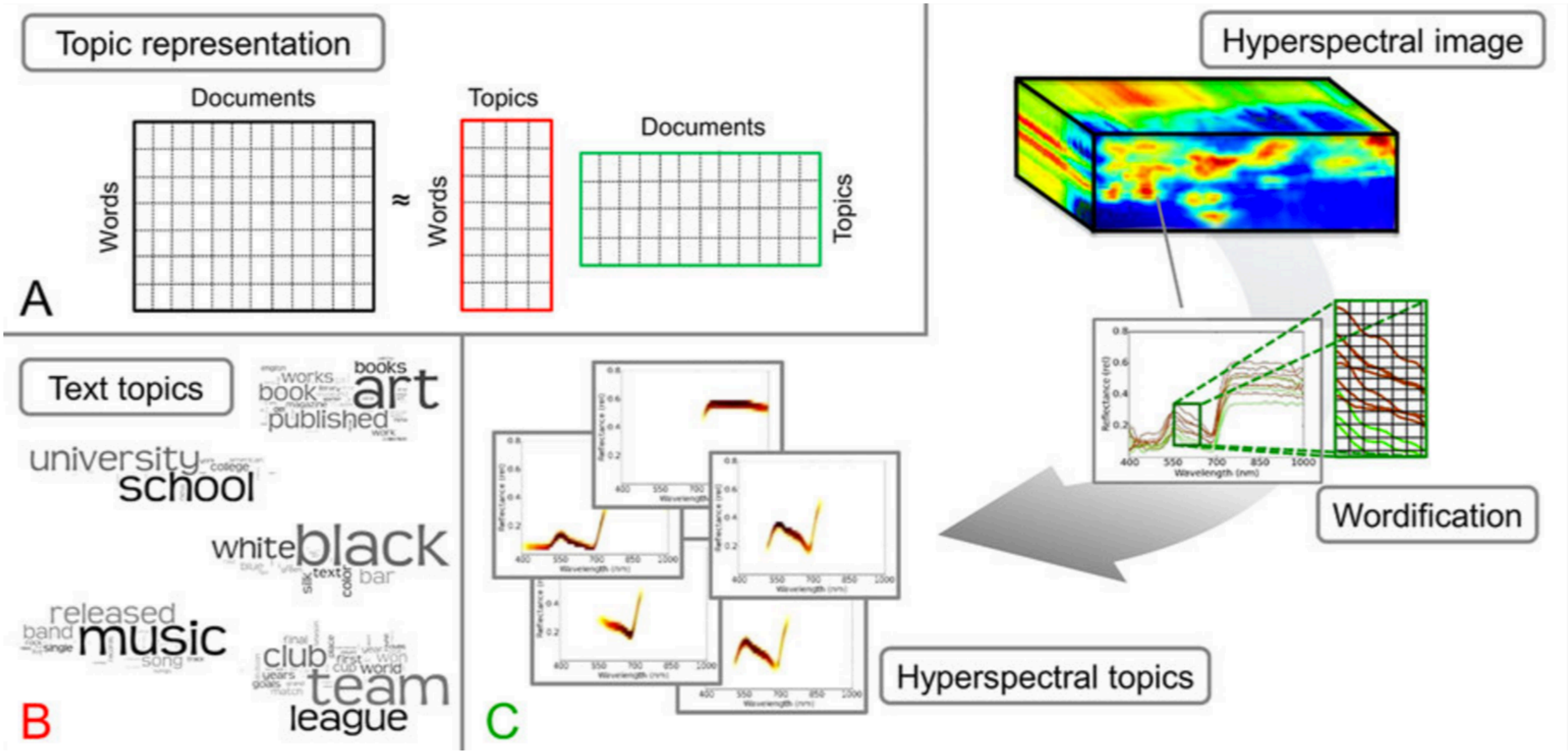
3. For a user with parameters $\underline{\theta}$ and a movie with (learned) features $\underline{x}$, predict a star rating of $\underline{\theta^T x}$.

$(\theta^{(j)})^T(x^{(i)})$

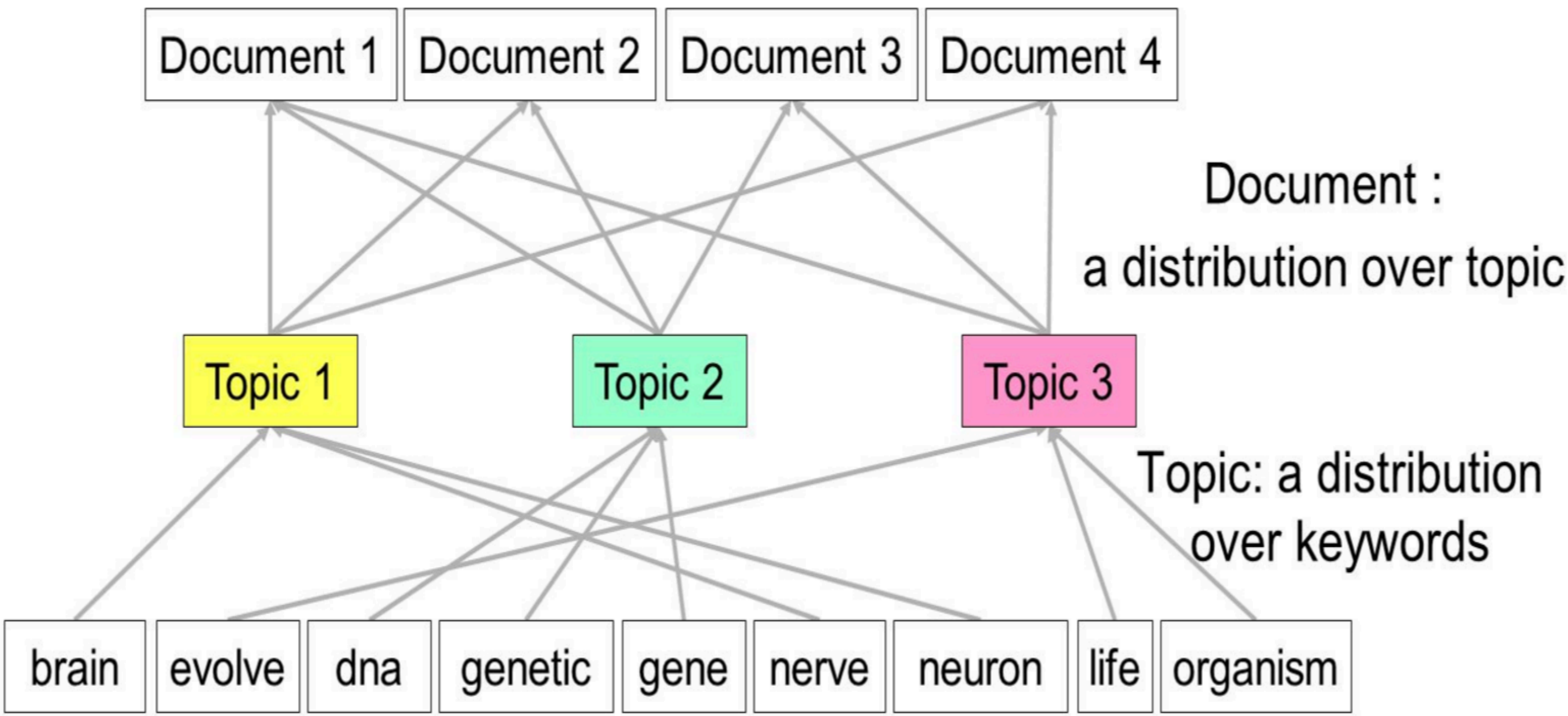
Topics Today

- Introduction to Recommender Systems.
- Collaborative Filtering.
- Non-Negative Matrix Factorization.
- Using Non-Negative Matrix Factorization for Topic Modeling.

Topic Modeling based on Non-negative Matrix Factorization

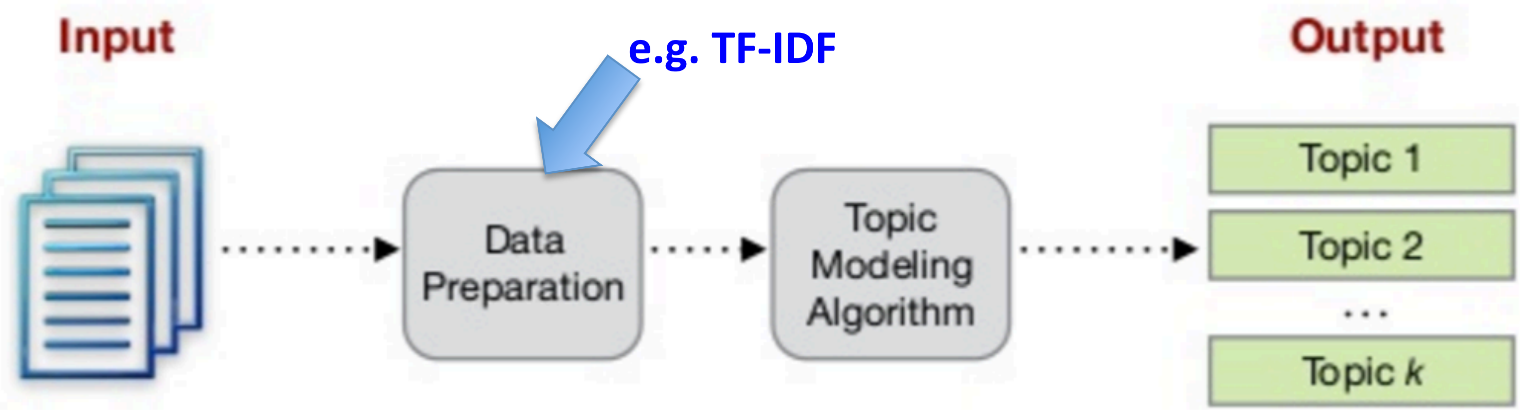


Intro: Topic Modeling



What is the Goal of Topic Modeling?

- **Goal:** Discover hidden thematic structure in a corpus of text (e.g. tweets, Facebook posts, news articles, political speeches).
- Unsupervised approach, no prior annotation required.



- Output of topic modeling is a set of *k* topics. Each topic has:
  1. A descriptor, based on highest-ranked terms for the topic.
  2. Membership weights for all documents relative to the topic.

What is the TF-IDF normalization?

tf–idf = term frequency–inverse document frequency

- TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.
- Mathematically, TF-IDF is the product of two statistics, term frequency and inverse document frequency.

Different ways to define Term Frequency $f_{t,d}$

- Raw frequency of a term in a document: *the number of times that term t occurs in document d, denoted by $f_{t.d}$.*
- Boolean "frequencies" defined as "= 1 if t occurs in d and 0 otherwise".
- logarithmically scaled frequency: 1 + log $f_{t.d}$, or zero if $f_{t.d}$ is zero.

Variants of TF weight

weighting scheme	TF weight
binary	$0, 1$
raw frequency	$f_{t,d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Inverse document frequency

- The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

Note: IDF then is a cross-document normalization, that puts less weight on common terms, and more weight on rare terms.

Different way to define
Inverse document frequency

Variants of IDF weight

weighting scheme	IDF weight ($n_t = \left\|\{d \in D : t \in d\}\right\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t}$
inverse document frequency smooth	$\log(1 + \frac{N}{n_t})$
inverse document frequency max	$\log \left(1 + \frac{\max_{\{t' \in d\}} n_{t'}}{n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

How to calculate tf-idf?

Then tf–idf is calculated as

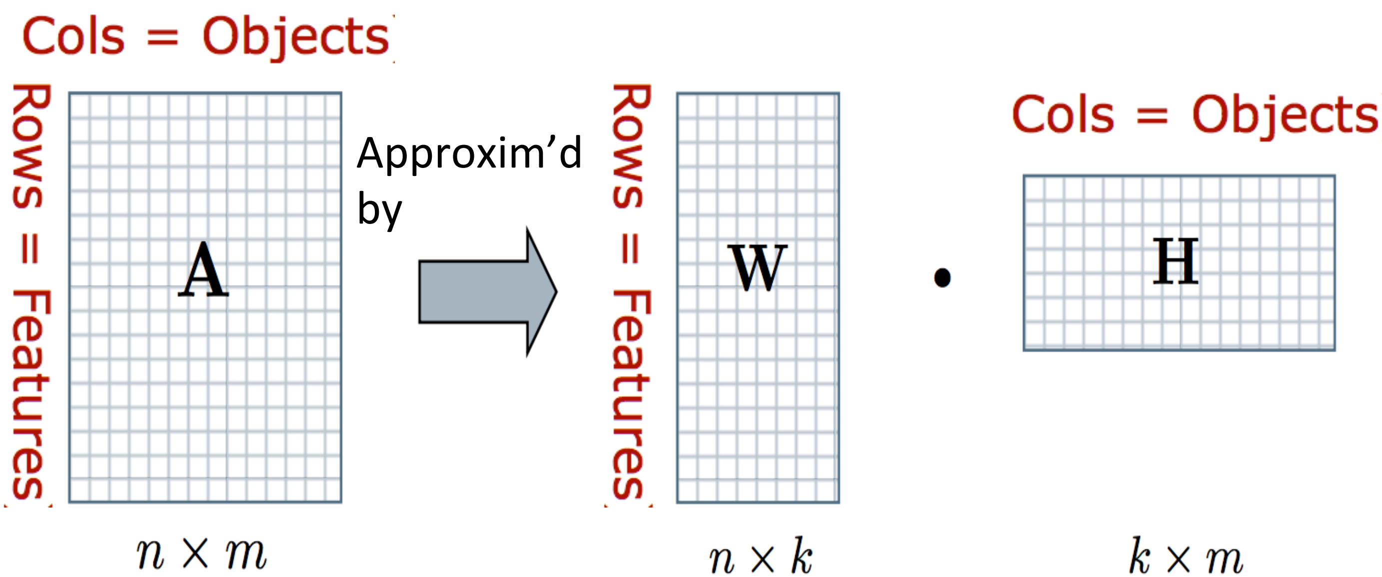
$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Recommended TF-IDF weighting schemes

weighting scheme	document term weight	query term weight
1	$f_{t,d} \cdot \log \frac{N}{n_t}$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_t f_{t,q}}\right) \cdot \log \frac{N}{n_t}$
2	$1 + \log f_{t,d}$	$\log(1 + \frac{N}{n_t})$
3	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$	$(1 + \log f_{t,q}) \cdot \log \frac{N}{n_t}$

What is Non-negative Matrix Factorization?

- Given a non-negative data matrix A.



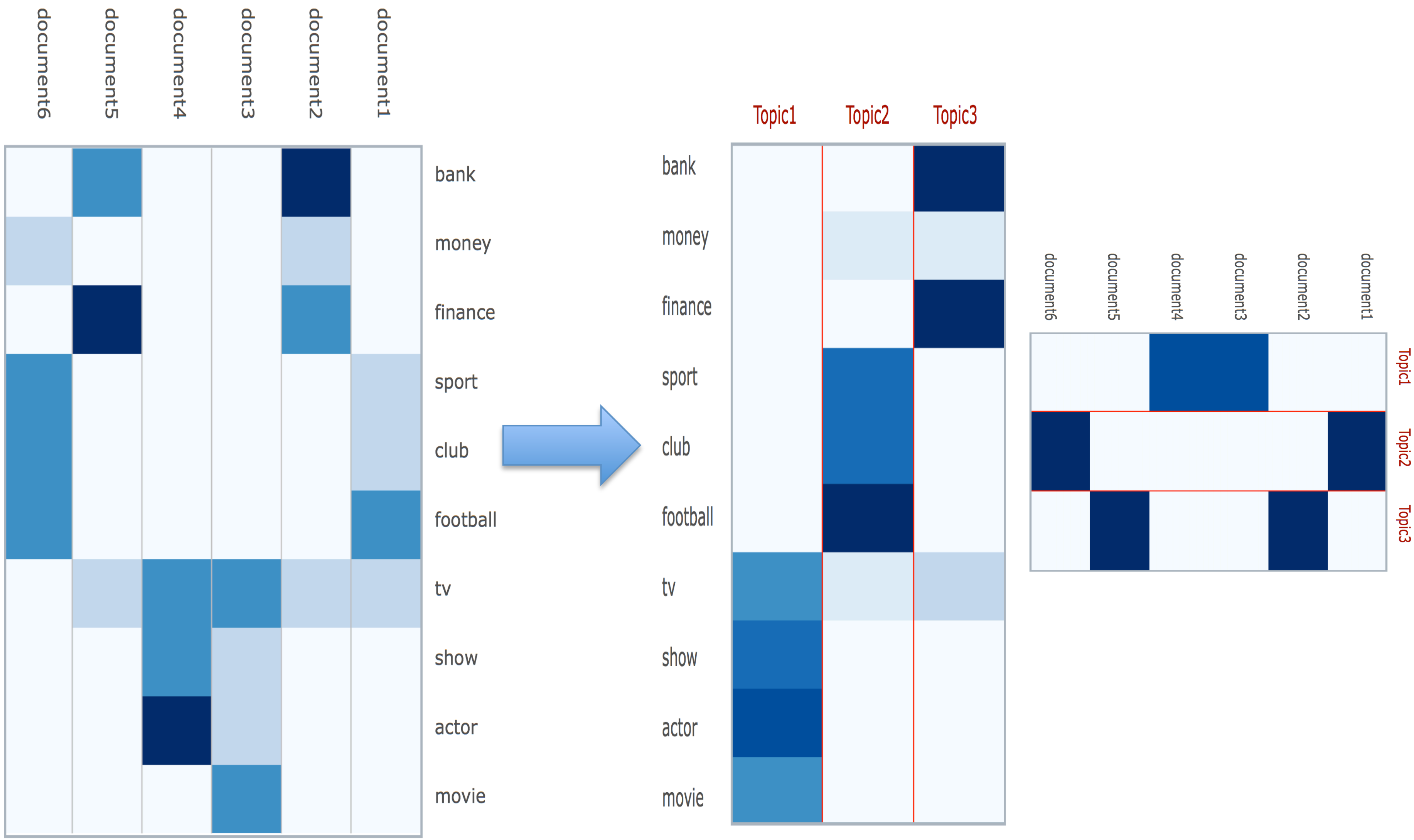
Means each Element of W ≥0

$$\mathbf{W} \geq 0\ ,\ \mathbf{H} \geq 0$$

Data matrix   Base vectors   Coefficient matrix

- W and H are called non-negative factors .

Example: Topic Modeling based on NMF



Goal: Minimizing the error between A and the approximation WH

$$\frac{1}{2}||\mathbf{A}-\mathbf{WH}||_{\mathsf{F}}^2=\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij}-(WH)_{ij})^2$$

- Use EM optimization to refine W and H in order to minimize the objective function.

Non-negative Matrix Factorization Algorithm

- **Input:** Non-negative data matrix (**A**), number of basis vectors (*k*), initial values for factors **W** and **H** (e.g. random matrices).
- **Objective Function:** Some measure of reconstruction error between **A** and the approximation **WH**.

Euclidean Distance (Lee & Seung, 1999)

$$\frac{1}{2}\,||\mathbf{A}-\mathbf{WH}||_{\mathsf{F}}^{2}=\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij}-(WH)_{ij})^{2}$$

- **Optimisation Process:** Local EM-style optimisation to refine **W** and **H** in order to minimise the objective function.
- Common approach is to iterate between two multiplicative update rules until convergence (Lee & Seung, 1999).

1. Update **H**

$$H_{cj}\leftarrow H_{cj}\frac{(W\mathbf{A})_{cj}}{(W\mathbf{WH})_{cj}}$$

2. Update **W**

$$W_{ic}\leftarrow W_{ic}\frac{(\mathbf{A}H)_{ic}}{(\mathbf{WH}H)_{ic}}$$

So What?

- NMF: an unsupervised family of algorithms that simultaneously perform dimension reduction and clustering.
- NMF produces a "parts-based" decomposition of the hidden (or latent) relationships in a data matrix.

Applications of Non-negative Matrix Factorization

- Also known as positive matrix factorization (PMF) and nonnegative matrix approximation (NNMA).
- No strong statistical justification or grounding.
- But has been successfully applied in a range of areas:

  - *Bioinformatics (e.g. clustering gene expression networks).*
  - *Image processing (e.g. face detection).*
  - *Audio processing (e.g. source separation).*
  - *Text analysis (e.g. document clustering).*

How to select k?

- As with LDA, the selection of number of topics k is often performed manually. No definitive model selection strategy.

- Various alternatives comparing different models:
  - Compare reconstruction errors for different parameters.

- Natural bias towards larger value of k.

  – Build a “consensus matrix” from multiple runs for each k, assess presence of block structure (Brunet et al, 2004).
  – Examine the stability (i.e. agreement between results) from multiple randomly initialized runs for each value of k.

Variants of Non-negative Matrix Factorization

Different objective functions:

- KL divergence (Sra & Dhillon, 2005).

More efficient optimization:

- Alternating least squares with projected gradient method for sub-problems (Lin, 2007).

Constraints:

- Enforcing sparseness in outputs (e.g. Liu et al, 2003).
- Incorporation of background information (Semi-NMF)

Different inputs:

- Symmetric matrices - e.g. document-document cosine similarity matrix (Ding & He, 2005).

Discussion

- Discuss with the students about what are key elements they need to first understand when they are trying to read a research paper.

Work on the board for understanding the power of manifolds with the examples (*Only time permits*)

- A sphere can be viewed as a collection of all 2 planes passing through origin.
- One can use a 3-O.N. basis vectors (called a moving frame) to characterize the motions of a 3D-robot (e.g. UAV) and their matrix representation using SO(3), and how one can still take derivatives to find tangent vectors.
- Talk about: the collection of all the distributions form so called "Statistical manifold". So one can define the probability distribution of distributions.