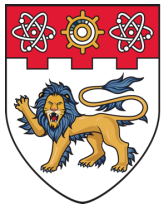
# CE9010: Introduction to Data Science

## Lecture 9: Recommender Systems

Semester 2 2017/18


## Xavier Bresson

School of Computer Science and Engineering
Data Science and AI Research Centre
Nanyang Technological University (NTU), Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

# Outline

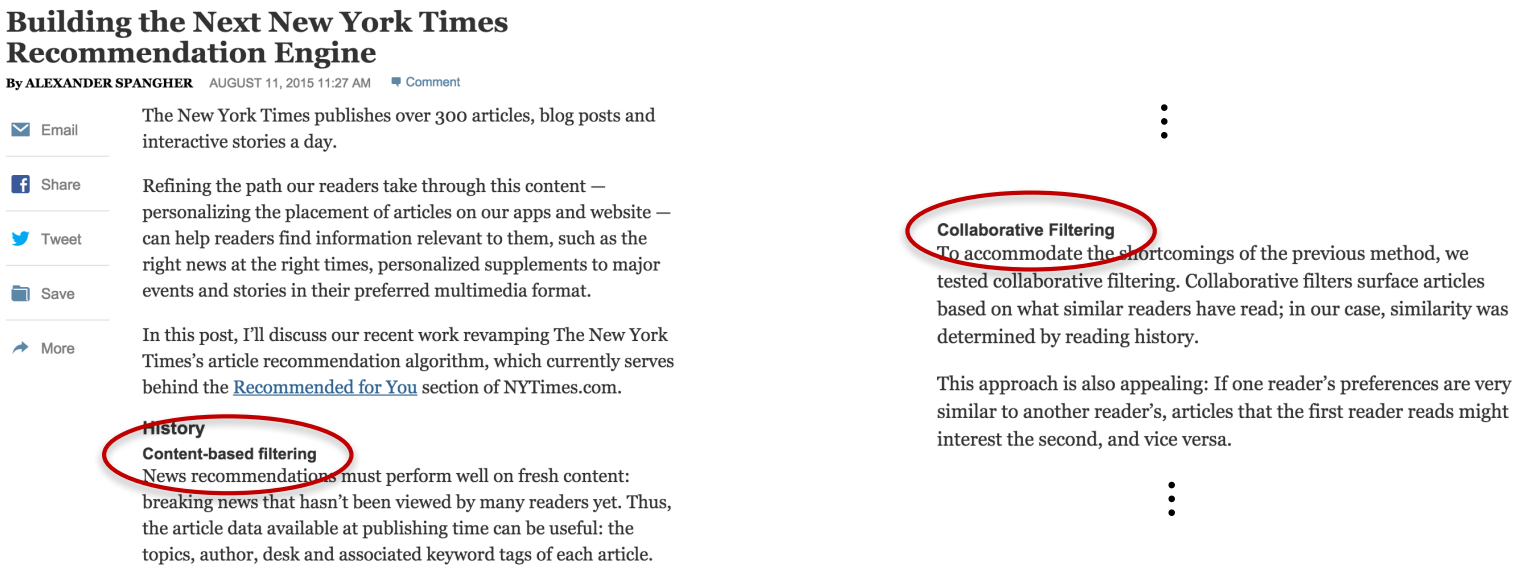
- Introduction
- Content-based recommendation
  - Movie features
  - Rating prediction with linear regression
- Collaborative recommendation
  - Learning movie features – sequential approach
  - Learning movie features – parallel approach
  - Embedding and closest movies
- Conclusion

# Outline

- Introduction
- Content-based recommendation
  - Movie features
  - Rating prediction with linear regression
- Collaborative recommendation
  - Learning movie features – sequential approach
  - Learning movie features – parallel approach
  - Embedding and closest movies
- Conclusion

# Introduction

- Recommender systems have become a central part of intelligent systems.
- Examples: Google search engine recommends webpages on internet, recommending movies on Netflix, friends on Facebook, products on Amazon, jobs on LinkedIn, articles on NY Times website:



**Building the Next New York Times Recommendation Engine**

By ALEXANDER SPANGHER AUGUST 11, 2015 11:27 AM Comment

Email Share Tweet Save More

The New York Times publishes over 300 articles, blog posts and interactive stories a day.

Refining the path our readers take through this content — personalizing the placement of articles on our apps and website — can help readers find information relevant to them, such as the right news at the right times, personalized supplements to major events and stories in their preferred multimedia format.

In this post, I'll discuss our recent work revamping The New York Times's article recommendation algorithm, which currently serves behind the Recommended for You section of NYTimes.com.

**History**

**Content-based filtering**

News recommendations must perform well on fresh content: breaking news that hasn't been viewed by many readers yet. Thus, the article data available at publishing time can be useful: the topics, author, desk and associated keyword tags of each article.

⋮

**Collaborative Filtering**

To accommodate the shortcomings of the previous method, we tested collaborative filtering. Collaborative filters surface articles based on what similar readers have read; in our case, similarity was determined by reading history.

This approach is also appealing: If one reader's preferences are very similar to another reader's, articles that the first reader reads might interest the second, and vice versa.

⋮

# Introduction

- Netflix Prize : 1M$ reward
  - Netflix is the biggest online movie company worldwide:

    #Movies> 100K and #Users> 100M.

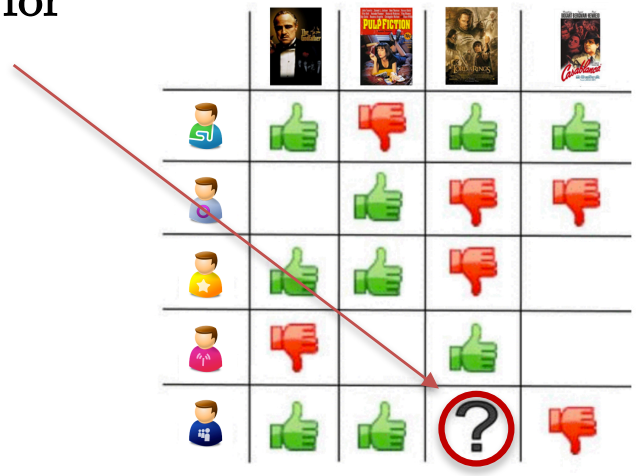

- Netflix prize was a competition in 2009 for the best algorithm that can predict user ratings for movies based on previous ratings.
  - Data:

    480,189 users

    17,770 movies

    100,480,507 ratings

    ⇒ Only 0.011% available ratings



# Introduction

- **Predict the missing ratings** (unwatched movies) of the movie-user matrix given a training set of (user,movie,rating):


| | User 1<br>John | User 2<br>David | User 3<br>Helen | User 4<br>Katy |
|---|---|---|---|---|
| Movie 1<br>Love Story | ? | 0 | 5 | ? |
| Movie 2<br>Casablanca | 1 | 1 | 4 | ? |
| Movie 3<br>Avengers | 5 | ? | 1 | 0 |
| Movie 4<br>Terminator | 4 | 4 | ? | 1 |

# Notations

- $n_m$ : Nb of movies
- $n_u$ : Nb of users
- $R_{im,iu}$ : Indicator matrix of ratings

$$R_{i_m,i_u} = \begin{cases} 1 & \text{if user } i_u \text{ has rated movie } i_m \\ 0 & \text{otherwise} \end{cases}$$

- $Y_{im,iu}$ : Rating matrix

$$\begin{aligned} Y_{i_m,i_u} &= \text{Rating value given by user } i_u \text{ to movie } i_m \\ &= \{0, 1, ..., 5\} \end{aligned}$$

| | User 1 John | User 2 David | User 3 Helen | User 4 Katy |
|---|---|---|---|---|
| Movie 1 Titanic | ? | 0 | 5 | ? |
| Movie 2 Love Story | 0 | 1 | 4 | ? |
| Movie 3 Avengers | 5 | ? | 1 | 0 |
| Movie 4 Terminator | 4 | 4 | ? | 1 |

$$R = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} - & 0 & 5 & - \\ 0 & 1 & 4 & - \\ 5 & - & 1 & 0 \\ 4 & 4 & - & 1 \end{bmatrix}$$

# Outline

- Introduction
- **Content-based recommendation**
  - **Movie features**
  - Rating prediction with linear regression
- Collaborative recommendation
  - Learning movie features – sequential approach
  - Learning movie features – parallel approach
  - Embedding and closest movies
- Conclusion

# Content-based recommendation

- Content-based recommender system: Predict movie rating given a set of movie features (content):

| | User 1 John | User 2 David | User 3 Helen | User 4 Katy | Feature 1 Romance $x_{(1)}$ | Feature 2 Action $x_{(2)}$ |
|---|---|---|---|---|---|---|
| Movie 1 Titanic | ? | 0 | 5 | ? | 0.7 | 0.2 |
| Movie 2 Love Story | 0 | 1 | 4 | ? | 0.9 | 0.05 |
| Movie 3 Avengers | 5 | ? | 1 | 0 | 0.1 | 0.95 |
| Movie 4 Terminator | 4 | 4 | ? | 1 | 0.15 | 0.97 |

$x_{(1)}$ : Measure the degree of romance in movie

$x_{(2)}$ : Measure the degree of action in movie

# Notations

Movie vector features:

$$x_{i_m} = \begin{bmatrix} 1 \\ x_{i_m}(1) \\ x_{i_m}(2) \end{bmatrix}$$

Add element 1 for offset parameter

Romance feature $x_{(1)}$ for movie $i_m$

Action feature $x_{(2)}$ for movie $i_m$

| | User 1 John | User 2 David | User 3 Helen | User 4 Katy | Feature 1 Romance | Feature 2 Action |
|---|---|---|---|---|---|---|
| Movie 1 Love Story | ? | 0 | 5 | ? | 0.7 | 0.2 |
| Movie 2 Casa blanca | 0 | 1 | 4 | ? | 0.9 | 0.05 |
| Movie 3 Avengers | 5 | ? | 1 | 0 | 0.1 | 0.95 |
| Movie 4 Terminator | 4 | 4 | ? | 1 | 0.15 | 0.97 |

User vector parameter:

$$w_{i_u} = \begin{bmatrix} w_{i_u}(0) \\ w_{i_u}(1) \\ w_{i_u}(2) \end{bmatrix}$$

# Outline

- Introduction
- **Content-based recommendation**
  - Movie features
  - **Rating prediction with linear regression**
- Collaborative recommendation
  - Learning movie features – sequential approach
  - Learning movie features – parallel approach
  - Embedding and closest movies
- Conclusion

# Rating prediction

- We may use a linear regression model $f_w(x)$ to estimate the movie rating:

User vector parameter:

$$w_{i_u} = \begin{bmatrix} w_{i_u}(0) \\ w_{i_u}(1) \\ w_{i_u}(2) \end{bmatrix}$$

Movie vector features:

$$x_{i_m} = \begin{bmatrix} 1 \\ x_{i_m}(1) \\ x_{i_m}(2) \end{bmatrix}$$

| | User 1 John | User 2 David | User 3 Helen | User 4 Katy | Feature 1 Romance | Feature 2 Action |
|---|---|---|---|---|---|---|
| Movie 1 Love Story | ? | 0 | 5 | ? | 0.7 | 0.2 |
| Movie 2 Casablanca | 0 | 1 | 4 | ? | 0.9 | 0.05 |
| Movie 3 Avengers | 5 | ? | 1 | 0 | 0.1 | 0.95 |
| Movie 4 Terminator | 4 | 4 | ? | 1 | 0.15 | 0.97 |

$$f_{w_{i_u}}(x_{i_m}) = w_{i_u}^T x_{i_m} \approx 1.3$$

For each user, we learn a parameter vector $w_{iu}$ that is used to estimate the rating of any movie $x_{im}$.

# Problem formulation

- **Regression problem:**

$$\min_{w_{i_u}} \frac{1}{n_{i_u}} \sum_{i_m : R_{i_m, i_u} = 1} \left( f_{w_{i_u}}(x_{i_m}) - Y_{i_m, i_u} \right)^2 + \frac{\lambda}{d} \sum_{j=1}^{d} w_{i_u(j)}^2$$

Regularization hyper-parameter

Nb of movies rated by user $i_u$

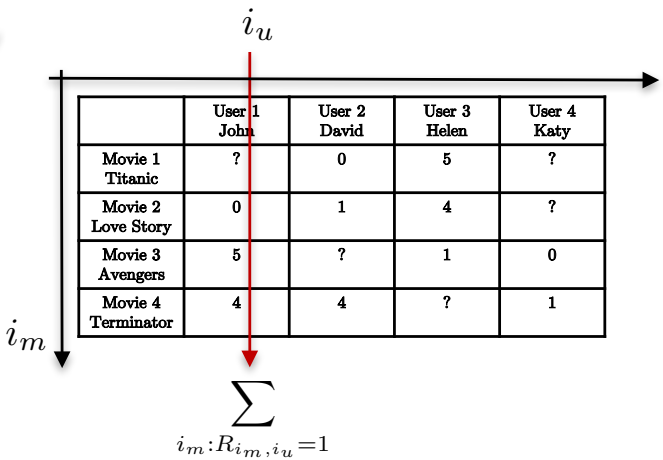
Mean value

Sum over all movies rated by user $i_u$

Square error

$L_2$ regularization

Mean square error




| | User 1 John | User 2 David | User 3 Helen | User 4 Katy |
|---|---|---|---|---|
| Movie 1 Titanic | ? | 0 | 5 | ? |
| Movie 2 Love Story | 0 | 1 | 4 | ? |
| Movie 3 Avengers | 5 | ? | 1 | 0 |
| Movie 4 Terminator | 4 | 4 | ? | 1 |

- **Linear regression problem:**

$$\min_{w_{i_u}} \frac{1}{n_{i_u}} \sum_{i_m : R_{i_m, i_u} = 1} \left( \underbrace{w_{i_u}^T x_{i_m}}_{\text{Linear function}} - Y_{i_m, i_u} \right)^2 + \frac{\lambda}{d} \sum_{j=1}^{d} w_{i_u(j)}^2$$

# Problem formulation

- Learn all user parameters $W=(w_1,\ldots,w_{nu})$ simultaneously:

$$\min_{w_1,\ldots,w_{n_u}} \frac{1}{n_{n_u}} \sum_{i_u=1}^{n_u} \frac{1}{n_{i_u}} \sum_{i_m:R_{i_m,i_u}=1} \left(w_{i_u}^T x_{i_m} - Y_{i_m,i_u}\right)^2 + \frac{\lambda}{d n_u} \sum_{i_u=1}^{n_u} \sum_{j=1}^{d} w_{i_u(j)}^2$$

$$\min_{w_1,\ldots,w_{n_u}} \quad L(w_1,\ldots,w_{n_u})$$

It will not change the solution

| | User 1 John $w_1$ | User 2 David $w_2$ | User 3 Helen $w_3$ | User 4 Katy $w_{n_u}$ | Feature 1 Romance | Feature 2 Action |
|---|---|---|---|---|---|---|
| Movie 1 Love Story | ? | 0 | 5 | ? | 0.7 | 0.2 |
| Movie 2 Casablanca | 0 | 1 | 4 | ? | 0.9 | 0.05 |
| Movie 3 Avengers | 5 | ? | 1 | 0 | 0.1 | 0.95 |
| Movie 4 Terminator | 4 | 4 | ? | 1 | 0.15 | 0.97 |

# Optimization

- Gradient descent (vectorized representation):

For $i_u = 0, ..., n_u$

$$w_{i_u} \leftarrow w_{i_u} - \tau \frac{\partial L}{\partial w_{i_u}}$$

$$\leftarrow w_{i_u} - \tau \Big( \frac{2}{n_{i_u}} \sum_{i_m : R_{i_m, i_u} = 1} (w_{i_u}^T x_{i_m} - Y_{i_m, i_u}) x_{i_m} + \frac{2\lambda}{d} w_{i_u} \Big)$$

- Gradient descent (element-wise representation):

For $i_u = 0, ..., n_u$

For $j = 0, ..., d$

$$w_{i_u(j)} \leftarrow w_{i_u(j)} - \tau \frac{\partial L}{\partial w_{i_u(j)}}$$

$$\leftarrow w_{i_u(j)} - \tau \Big( \frac{2}{n_{i_u}} \sum_{i_m : R_{i_m, i_u} = 1} (w_{i_u}^T x_{i_m} - Y_{i_m, i_u}) x_{i_m(j)} + \frac{2\lambda}{d} w_{i_u(j)} \Big)$$

# Limitation

- When we have access to good hand-crafted movie features like genre, release year, actors/actresses, etc, then content-based recommendation has excellent performances.

- However, getting meaningful hand-crafted features may be hard. For example, romantic/action features may require to watch the whole movie to give a good measure. Besides, it is sometimes hard to hand-craft features for item prediction.

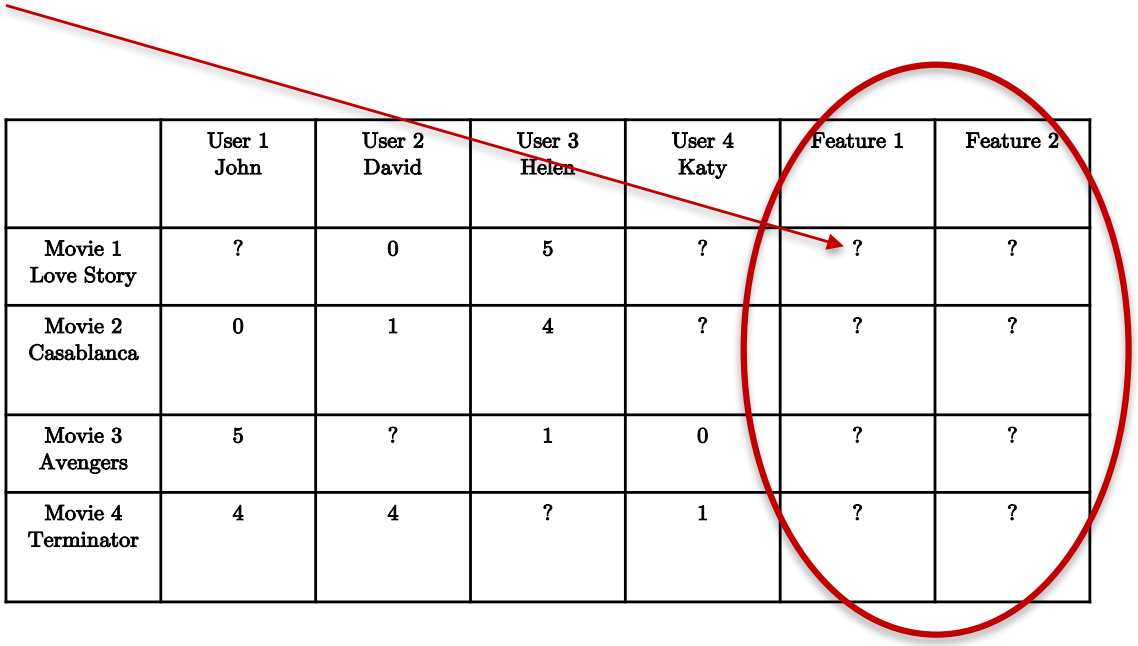
- An alternative approach is collaborative filtering when the features are learned.

# Outline

- Introduction
- Content-based recommendation
  - Movie features
  - Rating prediction with linear regression
- **Collaborative recommendation**
  - **Learning movie features – sequential approach**
  - Learning movie features – parallel approach
  - Embedding and closest movies
- Conclusion

# Collaborative recommendation

- Problem formulation: Suppose we do not know the right movie features to make movie recommendation, how to predict?



| | User 1 John | User 2 David | User 3 Helen | User 4 Katy | Feature 1 | Feature 2 |
|---|---|---|---|---|---|---|
| Movie 1 Love Story | ? | 0 | 5 | ? | ? | ? |
| Movie 2 Casablanca | 0 | 1 | 4 | ? | ? | ? |
| Movie 3 Avengers | 5 | ? | 1 | 0 | ? | ? |
| Movie 4 Terminator | 4 | 4 | ? | 1 | ? | ? |

- The collaborative approach will allow to learn the movie features and the user parameters simultaneously.

# Collaborative recommendation

- Sequential algorithm:
  - Given some initial movie features $X=(x_1,\ldots,x_{nm})$, learn the user parameters $W=(w_1,\ldots,w_{nu})$.
  - Repeat until convergence:
    - Fix the user parameters $W=(w_1,\ldots,w_{nu})$, learn the movie features $X=(x_1,\ldots,x_{nm})$.
    - Fix the movie features $X=(x_1,\ldots,x_{nm})$, learn the user parameters $W=(w_1,\ldots,w_{nu})$.

Fix $W=(w_1,\ldots,w_{nu})$, learn $X=(x_1,\ldots,x_{nm})$

| | User 1 John $w_1$ | User 2 David $w_2$ | User 3 Helen $w_3$ | User 4 Katy $w_{n_u}$ | Feature 1 $x_{(1)}$ | Feature 2 $x_{(2)}$ |
|---|---|---|---|---|---|---|
| Movie 1 Love Story | ? | 0 | 5 | ? | ? $x_{1(1)}$ | ? $x_{1(2)}$ |
| Movie 2 Casablanca | 0 | 1 | 4 | ? | ? | ? |
| Movie 3 Avengers | 5 | ? | 1 | 0 | ? | ? |
| Movie 4 Terminator | 4 | 4 | ? | 1 | ? $x_{n_m(1)}$ | ? $x_{n_m(2)}$ |

| | User 1 John $w_1$ | User 2 David $w_2$ | User 3 Helen $w_3$ | User 4 Katy $w_{n_u}$ | Feature 1 $x_{(1)}$ | Feature 2 $x_{(2)}$ |
|---|---|---|---|---|---|---|
| Movie 1 Love Story | ? | 0 | 5 | ? | ? $x_{1(1)}$ | ? $x_{1(2)}$ |
| Movie 2 Casablanca | 0 | 1 | 4 | ? | ? | ? |
| Movie 3 Avengers | 5 | ? | 1 | 0 | ? | ? |
| Movie 4 Terminator | 4 | 4 | ? | 1 | ? $x_{n_m(1)}$ | ? $x_{n_m(2)}$ |

Fix $X=(x_1,\ldots,x_{nm})$, learn $W=(w_1,\ldots,w_{nu})$

# Movie features

- Fix the user parameters $W=(w_1,\dots,w_{nu})$, learn the movie features $X=(x_1,\dots,x_{nm})$:

$$\min_{x_1,\dots,x_{n_m}} \underbrace{\frac{1}{n_m}\sum_{i_m=1}^{n_m}\frac{1}{n_{i_m}}\sum_{i_u:R_{i_m,i_u}=1}\left(w_{i_u}^T x_{i_m}-Y_{i_m,i_u}\right)^2+\frac{\gamma}{dn_m}\sum_{i_m=1}^{n_m}\sum_{j=1}^{d}x_{i_m(j)}^2}_{L_x(x_1,\dots,x_{n_m})}$$

Nb of users who rated movie $i_m$

- Gradient descent (vectorized representation):

For $i_m = 0,\dots,n_m$

$$x_{i_m} \leftarrow x_{i_m} - \tau\frac{\partial L_x}{\partial x_{i_m}}$$

$$\leftarrow x_{i_m} - \tau\left(\frac{2}{n_{i_m}}\sum_{i_m:R_{i_m,i_u}=1}\left(w_{i_u}^T x_{i_m}-Y_{i_m,i_u}\right)w_{i_u}+\frac{2\gamma}{d}x_{i_m}\right)$$

# User parameters

- **Fix the movie features $X=(x_1,\dots,x_{nm})$, learn the user parameters $W=(w_1,\dots,w_{nu})$:**

$$\min_{w_1,\dots,w_{n_u}} \underbrace{\frac{1}{n_m} \sum_{i_u=1}^{n_u} \frac{1}{n_{i_u}} \sum_{i_m:R_{i_m,i_u}=1} \left(w_{i_u}^T x_{i_m} - Y_{i_m,i_u}\right)^2 + \frac{\lambda}{d n_u} \sum_{i_u=1}^{n_u} \sum_{j=1}^{d} w_{i_u(j)}^2}_{L_w(w_1,\dots,w_{n_u})}$$

- **Gradient descent (vectorized representation):**

For $i_u = 0,\dots,n_u$

$$w_{i_u} \leftarrow w_{i_u} - \tau \frac{\partial L_w}{\partial w_{i_u}}$$

$$\leftarrow w_{i_u} - \tau \left( \frac{2}{n_{i_u}} \sum_{i_m:R_{i_m,i_u}=1} \left(w_{i_u}^T x_{i_m} - Y_{i_m,i_u}\right) x_{i_m} + \frac{2\lambda}{d} w_{i_u} \right)$$

# Alternate optimization

- The sequential algorithm solves the optimization problem in a sequential way:
  - Repeat until convergence:
    - Fix the user parameters $W=(w_1,\ldots,w_{nu})$, learn the movie features $X=(x_1,\ldots,x_{nm})$.
    - Fix the movie features $X=(x_1,\ldots,x_{nm})$, learn the user parameters $W=(w_1,\ldots,w_{nu})$.

- Sequential optimization may be very long:
  - $W \rightarrow X \rightarrow W \rightarrow X \rightarrow W \rightarrow X \rightarrow \ldots$

- We can speed up the optimization process.

# Outline

- Introduction
- Content-based recommendation
  - Movie features
  - Rating prediction with linear regression
- **Collaborative recommendation**
  - Learning movie features – sequential approach
  - **Learning movie features – parallel approach**
  - Embedding and closest movies
- Conclusion

# Collaborative loss

- Combining movie feature loss and user parameter loss:

$$\min_{\substack{x_1,\ldots,x_{n_m}\\ w_1,\ldots,w_{n_u}}} \underbrace{\frac{1}{n_{mu}} \sum_{i_m,i_u:R_{i_m,i_u}=1} \left(w_{i_u}^T x_{i_m} - Y_{i_m,i_u}\right)^2 + \frac{\gamma}{dn_m} \sum_{i_m=1}^{n_m} \sum_{j=1}^{d} x_{i_m(j)}^2 + \frac{\lambda}{dn_u} \sum_{i_u=1}^{n_u} \sum_{j=1}^{d} w_{i_u(j)}^2}$$

Nb of ratings

$$\min_{\substack{x_1,\ldots,x_{n_m}\\ w_1,\ldots,w_{n_u}}} L(x_1,\ldots,x_{n_m},w_1,\ldots,w_{n_u})$$

# Collaborative algorithm

- Alternate learning algorithm:
  - Given some initial random user parameters $W=(w_1,\ldots,w_{nu})$.
  - Repeat until convergence:

For $i_u = 0, \ldots, n_u$ and for $i_m = 0, \ldots, n_m$

$$x_{i_m} \leftarrow x_{i_m} - \tau\left(\frac{2}{n_{mu}} \sum_{i_u : R_{i_m, i_u}=1} \left(w_{i_u}^T x_{i_m} - Y_{i_m, i_u}\right) w_{i_u} + \frac{2\gamma}{dn_m} x_{i_m}\right)$$

$$w_{i_u} \leftarrow w_{i_u} - \tau\left(\frac{2}{n_{mu}} \sum_{i_m : R_{i_m, i_u}=1} \left(w_{i_u}^T x_{i_m} - Y_{i_m, i_u}\right) x_{i_m} + \frac{2\lambda}{dn_u} w_{i_u}\right)$$

- Prediction:
  - Select a movie $i_m$ and a user $i_u$, predict a rating with the formula:

$$f_{w_{i_u}}(x_{i_m}) = w_{i_u}^T x_{i_m} \approx 1.3$$

# Low-rank prediction

- The predicted rating matrix $\boldsymbol{F}$ has a low-rank structure:

$$\begin{bmatrix} | & & | \\ x_1 & \dots & x_{n_u} \\ | & & | \end{bmatrix} = X$$

$\boldsymbol{d}$ x $\boldsymbol{n_u}$

$$W^T = \begin{bmatrix} —w_1^T— \\ \vdots \\ —w_{n_m}^T— \end{bmatrix}$$

$\boldsymbol{n_m}$ x $\boldsymbol{d}$

$$F = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$i_u$, $i_m$

$\boldsymbol{n_m}$ x $\boldsymbol{n_u}$

$$F_{i_m,i_u} = w_{i_u}^T x_{i_m}$$

Low-rank matrix factorization: $F = W^T X$

$\boldsymbol{n_m}$ x $\boldsymbol{n_u}$ $\quad$ $\boldsymbol{d}$ x $\boldsymbol{n_u}$

$\boldsymbol{n_m}$ x $\boldsymbol{d}$

small value $d$=low-rank

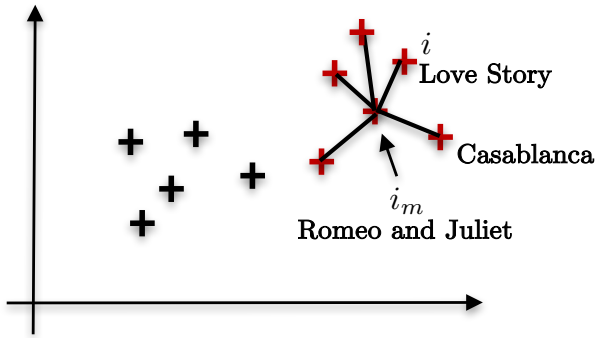
# Outline

- Introduction
- Content-based recommendation
  - Movie features
  - Rating prediction with linear regression
- **Collaborative recommendation**
  - Learning movie features – sequential approach
  - Learning movie features – parallel approach
  - **Embedding and closest movies**
- Conclusion

# Finding related movies

- For each movie (product), we have learned a feature vector $x_{im}$ in $\mathrm{R}^{\mathrm{d}}$.
- The learned features represent an embedding of all movies.
- An embedding is a compact and linear representation of the data. It is used in many data science tasks like natural language processing (NLP) to represent the words for translation, questions and answers, etc.
- The embedding can be used to find similar data. Here, we can find the movies that are the most related to a given movie $i_m$.
  - Select the 5 most similar movies to $i_m$ by finding the 5 smallest distances $i$ :

$$\{\ i\ :\ \|x_{i_m} - x_i\|_2\ \}$$



# Outline

- Introduction
- Content-based recommendation
  - Movie features
  - Rating prediction with linear regression
- Collaborative recommendation
  - Learning movie features – sequential approach
  - Learning movie features – parallel approach
  - Embedding and closest movies
- **Conclusion**

# Conclusion

- Recommender systems are the backbone of companies like Amazon, Netflix, Google, etc
- There exist two classes of recommender systems: Content-based and collaborative techniques.
- Linear prediction can be improved with polynomial capacity and neural networks (state-of-the-art performances).
- Performances depend on the number of available data (user data, product data and ratings): the more the better.

Questions?