Recommender Systems

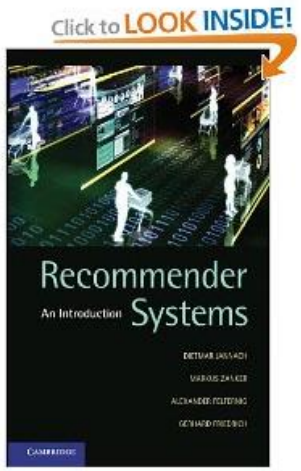
Jee-Hyong Lee

Department of Computer Science & Engineering
Sungkyunkwan University

Outline

1. Introduction
2. Collaborative Filtering Overview
3. Memory based Collaborative Filtering
4. Model based Collaborative Filtering
5. Content based Recommender System
6. Summary

Recommender Systems



Recommender Systems: An Introduction
[Hardcover]
Dietmar Jannach (Author), Markus Zanker (Author), Alexander Felfernig (Author), Gerhard Friedrich (Author)
(2 customer reviews) | Like (8)

List Price: $69.00
Price: $54.75 & this item ships for **FREE with Super Saver Shipping**. Details
You Save: $14.25 (21%)

Usually ships within 2 to 4 weeks.
Ships from and sold by **Amazon.com**. Gift-wrap available.

34 new from $50.81 23 used from $50.76

Buy New $54.75
Quantity: 1
Add to Cart
or
Sign in to turn on 1-Click ordering
Add to Wish List

Sell Us Your Item
For a **$11.17** Gift Card
Trade in
Learn more

Frequently Bought Together

Price For All Three: $216.63
Add all three to Cart | Add all three to Wish List
Some of these items ship sooner than the others. Show details

- **This item:** Recommender Systems: An Introduction by Dietmar Jannach Hardcover $54.75
- Recommender Systems Handbook by Francesco Ricci Hardcover $136.45
- Algorithms of the Intelligent Web by Haralambos Marmanis Paperback $25.43

Recommender Systems

- **Netflix:**
  - 2/3 of the movies watched are recommended
- **Google News:**
  - Recommendations generate 38% more clickthrough
- **Amazon:**
  - 35% sales from recommendations
- **Choicestream:**
  - 28% of the people would buy more music if they found what they liked

Definition of Recommender Systems

- **Given**
  - User profile (usage history, demographics, …)
  - Items (with or without additional information)

- **Goal**
  - Relevance scores of unseen items
  - List of unseen items

- **By using a number of technologies**
  - Information Retrieval: document models, similarity, ranking
  - Machine Learning & Data Mining: classification, clustering, regression, probability, association
  - Others: user modeling, HCI

Approaches

- **Collaborative Filtering**
  - Memory based CF
    - User-based CF, Item-based CF
  - Model based CF
    - Dimension reduction, Clustering, Association rules, restricted Boltzmann machine, Probabilistic approach, Other classifiers
- **Content-based Recommendation**
  - Content/User modeling & similarity
    - TF-IDF, Cosine similarity
- **Context-aware Recommendation**
  - Pre-filtering, Post-filtering
  - Contextual modeling
    - Extension of 2D model, Tensor factorization

Approaches

- **Other Approaches**
  - Combining Multiple Recommendation Approach
  - Combining Multiple Information
    - Hybrid Information Network based CF
    - Collective matrix factorization
  - Diversity in Recommendation
  - Division of Profiles into Sub-Profiles
  - Recommendation for group users

Outline

1. Introduction
2. **Collaborative Filtering Overview**
3. Memory based Collaborative Filtering
4. Model based Collaborative Filtering
5. Content based Recommender System
6. Summary

Overview

- **Basic assumption and idea**
  - Implicit or explicit user ratings to items are available
  - Customers who had similar tastes in the past, will have similar tastes in the future

	I1	I2	I3	I4	I5
Active	5	3	?		4
U1		2	2	3	3
U2	3	2		5	4
U3	4		3	2	
U4			1	4	

Overview

- **Easy to apply any domain**
  - Based on big data: commercial e-commerce sites
  - Easy to explain: wisdom of the crowd
  - Flexible: various algorithms exist
  - Example: book, movies, DVDs, ..

Collaborative Filtering

- **Memory based (k-NN approach)**
  - User-based CF
  - Item-based CF

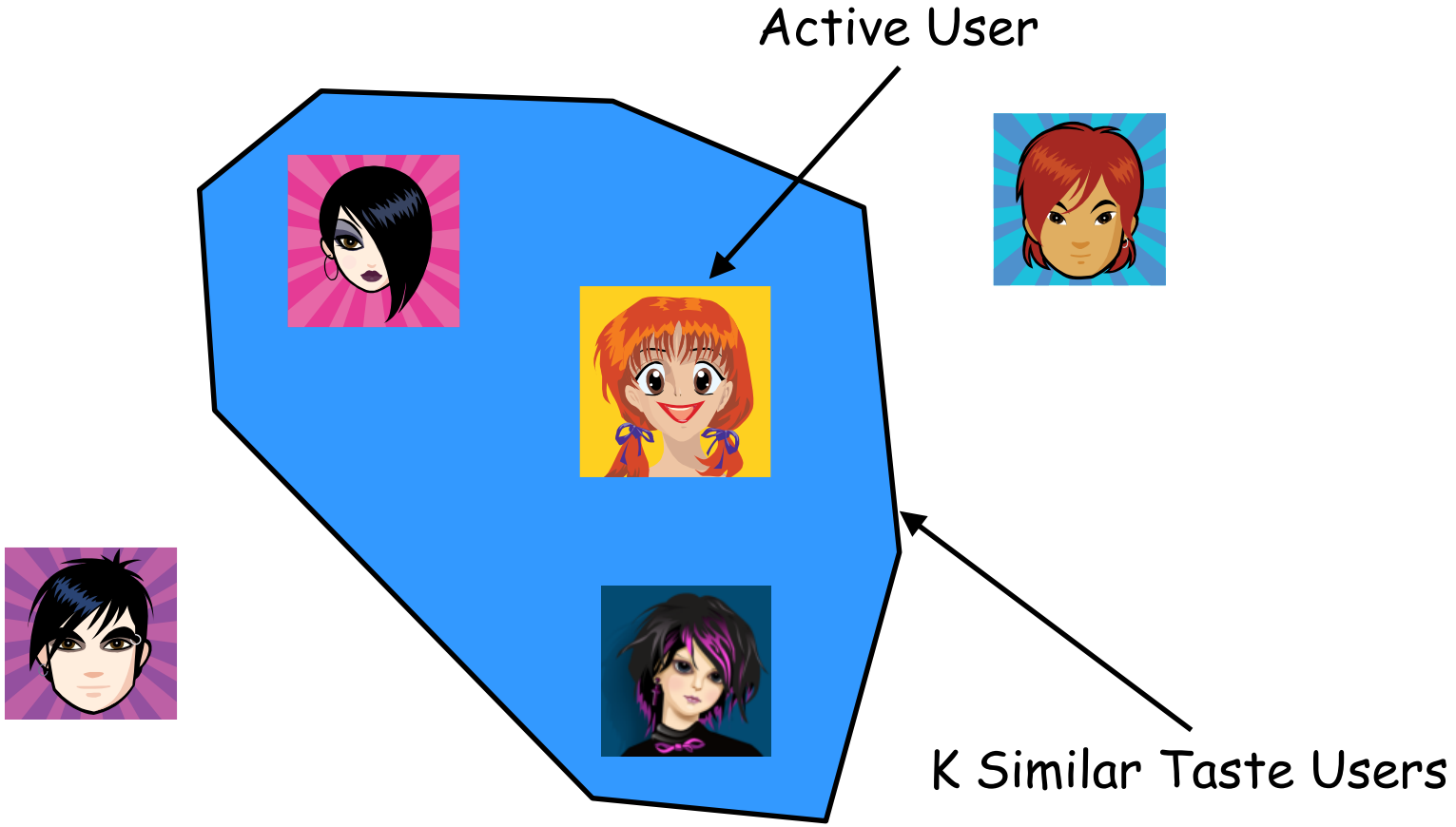
- **Model based (User model construction)**
  - Dimension reduction (Matrix Factorization)
  - Clustering
  - Association rule mining
  - Restricted Boltzmann machine
  - Probabilistic models
  - Various machine learning approaches

Outline

1. Introduction
2. Collaborative Filtering Overview
3. **Memory based Collaborative Filtering**
   - **3-1. User-based CF**
   - **3-2. Item-based CF**
4. Model based Collaborative Filtering
5. Content based Recommender System
6. Summary

Memory Based CF

- **K-Nearest Neighbors**



User-based Collaborative Filtering

- **Questions**
  - How to determine the similarity of two users?
  - How to predict Active user's rating using neighbors' ratings?

	I1	I2	I3	I4	I5
Active	5	3	?		4
U1		2	2	3	3
U2	3	2		5	4
U3	4		3	2	
U4			1	4	

User-based Collaborative Filtering

- **User Similarity: Pearson's correlation coefficient**

$$sim(u_1, u_2) = \frac{\sum_{i \in I} \left(r_{u_1,i} - \bar{r}_{u_1}\right)\left(r_{u_2,i} - \bar{r}_{u_2}\right)}{\sqrt{\sum_{i \in I} \left(r_{u_1,i} - \bar{r}_{u_1}\right)^2} \sqrt{\sum_{i \in I} \left(r_{u_2,i} - \bar{r}_{u_2}\right)^2}}$$

  - $r_{u,i}$: rating of user **u** for item **i**
  - $\bar{r}_u$: user **u**'s average ratings
  - I: items commonly consumed by u_1 and u_2

User-based Collaborative Filtering

- **Example: Similarity between *Active* and U_1**

	I1	I2	I3	I4	I5
Active	5	3	?		4
U1		2	2	3	3
U2	3	2		5	4
U3	4		3	2	
U4			1	4	

– I = { I2, I5} $sim(Active, u_1) = \frac{0.5}{1 \times \sqrt{0.5}} = 0.71$

$$\sqrt{\sum_{i \in I} \left(r_{Active,i} - \bar{r}_{Active}\right)^2} = \sqrt{(3-4)^2 + (4-4)^2} = 1$$

$$\sqrt{\sum_{i \in I} \left(r_{u_1,i} - \bar{r}_{u_1}\right)^2} = \sqrt{(2-2.5)^2 + (3-2.5)^2} = \sqrt{0.5}$$

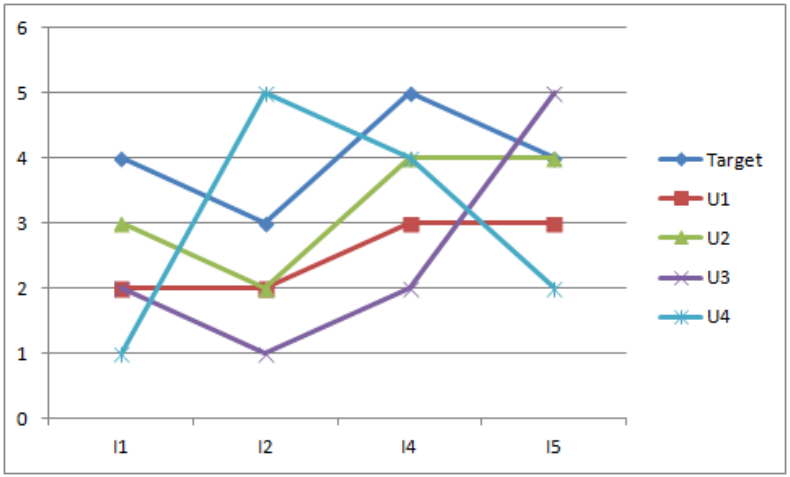
$$\sum_{i \in I} \left(r_{u_1,i} - \bar{r}_{u_1}\right)\left(r_{u_2,i} - \bar{r}_{u_2}\right) = (3-4)(2-2.5) + (4-4)(3-2.5) = 0.5$$

User-based Collaborative Filtering

- **User Similarity: Pearson's correlation coefficient**
  - Why?

$$sim(u_1, u_2) = \frac{\sum_{i \in I} \left(r_{u_1,i} - \bar{r}_{u_1}\right)\left(r_{u_2,i} - \bar{r}_{u_2}\right)}{\sqrt{\sum_{i \in I} \left(r_{u_1,i} - \bar{r}_{u_1}\right)^2} \sqrt{\sum_{i \in I} \left(r_{u_2,i} - \bar{r}_{u_2}\right)^2}}$$

	I1	I2	I3	I4	I5
Active	4	3	?	5	4
U1	2	2	2	3	3
U2	3	2	4	5	4
U3	2	3	3	2	5
U4	1	5	1	4	2



User-based Collaborative Filtering

- **How much target user likes I3?**

	I1	I2	I3	I4	I5
Active	5	3	?		4
U1		2	2	3	3
U2	3	2		5	4
U3	4		3	2	
U4			1	4	

$$sim(Active, u_1) = 0.71$$

$$sim(Active, u_2) = \text{Not available; } u_2 \text{ does not watch } I_3$$

$$sim(Active, u_3) = 1.00$$

$$sim(Active, u_3) = \text{Not available; no common items}$$

User-based Collaborative Filtering

- **Prediction: Simple weighted average**

$$pred(u,i) = \frac{\sum_{v \in U} sim(u,v) \cdot r_{v,i}}{\sum_{v \in U} sim(u,v)}$$

	I1	I2	I3	I4	I5	Sim.
Active	5	3	?		4	
U1		2	2	3	3	**0.71**
U2	3	2		5	4	
U3	4		3	2		**1.00**
U4			1	4		

$$pred(\text{Target}, \text{I3}) = \frac{2 \times 0.71 + 3 \times 1.00}{0.71 + 1.00} = 2.58$$

User-based Collaborative Filtering

- **Prediction: Considering user bias**

$$pred(u,i) = \bar{r}_u + \frac{\sum_{v \in U} sim(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in U} sim(u,v)}$$

	I1	I2	I3	I4	I5	Sim.
Active	5	3	?		4	
U1		2	2	3	3	**0.71**
U2	3	2		5	4	
U3	4		3	2		**1.00**
U4			1	4		

$$pred(\text{Target}, \text{I3}) = 4 + \frac{(2-2.5)\times 0.71 + (3-3)\times 1.00}{0.71+1.00} = 3.79$$

User-based Collaborative Filtering

- **How many neighbors?**
  - Only consider positively correlated neighbors (or higher threshold)
  - Can be optimized based on data set
  - Often, between 50 and 200

User-based Collaborative Filtering

- **Which neighbors are more valuable ?**
  - User similarity: Different weights for items
    - Give more weight to items with a higher variance
  - Example: Which user is more similar to Active user ?

	I1	I2	I3	I4	I5
Active	5	3	?	4	5
U1		3	2	4	
U2	5		4		5
U3	5	4		2	5
U4	5	1		5	5

User-based Collaborative Filtering

- **Which neighbors are more valuable ?**
  - Use similarity: Considering the number of co-rated items
    - Adjust weights considering the number of co-rated items
  - Example: Which user is more similar to Active user ?

	I1	I2	I3	I4	I5	Sim.
Active	5	3	?		4	
U1		2	2	3	3	**0.71**
U2	3	2		5	4	
U3	4		3	2		**1.00**
U4			1	4		

User-based Collaborative Filtering

- **Possible issues**
  - Scalability
    - Nearest neighbor computation cost is very high
    - Pre-computation of similarities possible but potentially unstable
      Preference of users are changing

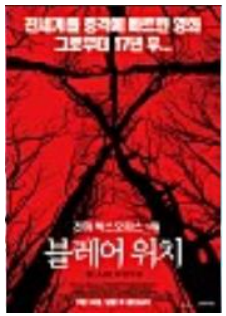
    => Item-based CF

  - Sparsity
    - Users purchases less than 1% of items
    - Problem of finding enough neighbors
    - Users with preferences for niche products

  => Model based approach

Item-based Collaborative Filtering

- **K Nearest Neighbors**



Item-based Collaborative Filtering

- **Questions**
  - How to determine the similarity of two items?
  - How to predict Active user's rating using neighbors' ratings?

	I1	I2	I3	I4	I5
Active	5	3	?		4
U1		2	4	3	
U2	2	2		5	3
U3	3		5	4	
U4	1		3	2	

Item-based Collaborative Filtering

- **Similarity between items: Pearson correlation**

$$sim(i_1, i_2) = \frac{\sum_{u \in U} \left(r_{u,i_1} - \bar{r}_{i_1}\right) \cdot \left(r_{u,i_2} - \bar{r}_{i_2}\right)}{\sqrt{\sum_{u \in U} \left(r_{u,i_1} - \bar{r}_{i_1}\right)^2} \sqrt{\sum_{u \in U} \left(r_{u,i_2} - \bar{r}_{i_2}\right)^2}}$$

  - $r_{u,i}$: rating of user **u** for item **i**
  - $\bar{r}_i$: item ***i***s average ratings
  - U: users commonly consumed i_1 and i_2

Item-based Collaborative Filtering

- **Similarity between items**
  - Cosine similarity

$$sim(i_1, i_2) = \frac{\sum_{u \in U} r_{u,i_1} \cdot r_{u,i_2}}{\sqrt{\sum_{u \in U} r_{u,i_1}^2} \sqrt{\sum_{u \in U} r_{u,i_2}^2}}$$

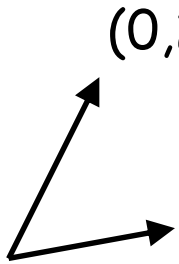
  - Adjusted cosine similarity
    - Considering user bias

$$sim(i_1, i_2) = \frac{\sum_{u \in U} \left(r_{u,i_1} - \bar{r}_u\right) \cdot \left(r_{u,i_2} - \bar{r}_u\right)}{\sqrt{\sum_{u \in U} \left(r_{u,i_1} - \bar{r}_u\right)^2} \sqrt{\sum_{u \in U} \left(r_{u,i_2} - \bar{r}_u\right)^2}}$$

Item-based Collaborative Filtering

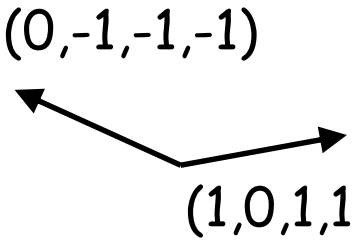
- **Similarity between items**

	I1	I2	I3	I4	I5
Target	5	3	?		4
U1		2	4	3	
U2	2	2		5	3
U3	3		5	4	
U4	1		3	2	



$$sim(i_1, i_3) = 0.658$$

	I1	I2	I3	I4	I5
Target	5	3	?		4
U1		-1	1	0	
U2	-1	-1		2	0
U3	-1		1	0	
U4	-1		1	0	



$$sim(i_1, i_3) = -0.667$$

Item-based Collaborative Filtering

- **Prediction**
  - weighted sum of the ratings of the target user

$$r_{u,i} = \frac{\sum_{j \in I} sim(i, j) \cdot r_{u,j}}{\sum_{i \in I} sim(i, j)}$$

	I1	I2	I3	I4	I5
Active	5	3	?		4
U1		2	4	3	
U2	2	2		5	3
U3	3		5	4	
U4	1		3	2	
Sim.	**0.68**	**0.52**		**0.73**	**NA**

$$\frac{5 \times 0.68 + 3 \times 0.52}{0.68 + 0.52} = 4.13$$

Item-based Collaborative Filtering

- **Comparison to User-based CF**
  - Similarity between items is static.
    - User-based CF: Similarity between users is dynamic
      - » Precomputing user similarity lead to poor predictions
  - Enables precomputing of item-item similarity
    - Table lookup for the similarity values
  - Space for precomputation
    - At maximum n^2 similarities
    - Lower than this because of many item pairs with no co-ratings
    - Further reduction: minimum threshold for co-ratings

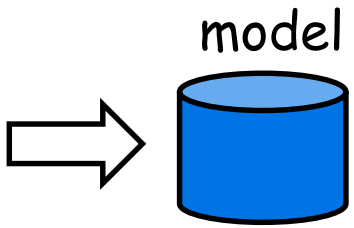
Outline

1. Introduction
2. Collaborative Filtering Overview
3. Memory based Collaborative Filtering
4. **Model based Collaborative Filtering**
   - **4-1. Overview**
   - **4-2. Matrix Factorization**
   - **4-3. Singular Value Decomposition (SVD)**
   - **4-4. Prediction with SVD**
   - **4-5. Matrix Factorization through Optimization**
5. Content based Recommender System
6. Summary

Overview

- **Memory-based: kNN based methods**
  - Rating matrix is directly used for recommendation
  - Not scalable for most real-world scenarios
  - Tens of millions of customers and millions of items
- **Model-based approaches**
  - Based on models learned from rating matrix
  - Learned model is used to make predictions
  - Models are updated / re-trained periodically
  - Model-building and updating can be computationally expensive

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
abnormalities	0	0	0	0	0	0	0	1	0	1	0	0	0	0
age	1	0	0	0	0	0	0	0	0	0	0	1	0	0
behavior	0	0	0	0	1	1	0	0	0	0	0	0	0	0
blood	0	0	0	0	0	0	0	1	0	0	1	0	0	0
close	0	0	0	0	0	0	1	0	0	0	1	0	0	0
culture	1	1	0	0	0	0	0	1	1	0	0	0	0	0
depressed	1	0	1	1	1	0	0	0	0	0	0	0	0	0
discharge	1	1	0	0	0	1	0	0	0	0	0	0	0	0
disease	0	0	0	0	0	0	0	0	1	0	1	0	0	0
fast	0	0	0	0	0	0	0	0	0	1	0	1	1	1
generation	0	0	0	0	0	0	0	0	1	0	0	0	1	0
oestrogen	0	0	1	1	0	0	0	0	0	0	0	0	0	0
patients	1	1	0	1	0	0	0	1	0	0	0	0	0	0
pressure	0	0	0	0	0	0	0	0	0	0	1	0	0	1
rats	0	0	0	0	0	0	0	0	0	0	0	0	1	1
respect	0	0	0	0	0	0	0	1	0	0	0	1	0	0
rise	0	0	0	1	0	0	0	0	0	0	0	0	0	1
study	1	0	1	0	0	0	0	0	1	0	0	0	0	0



Overview

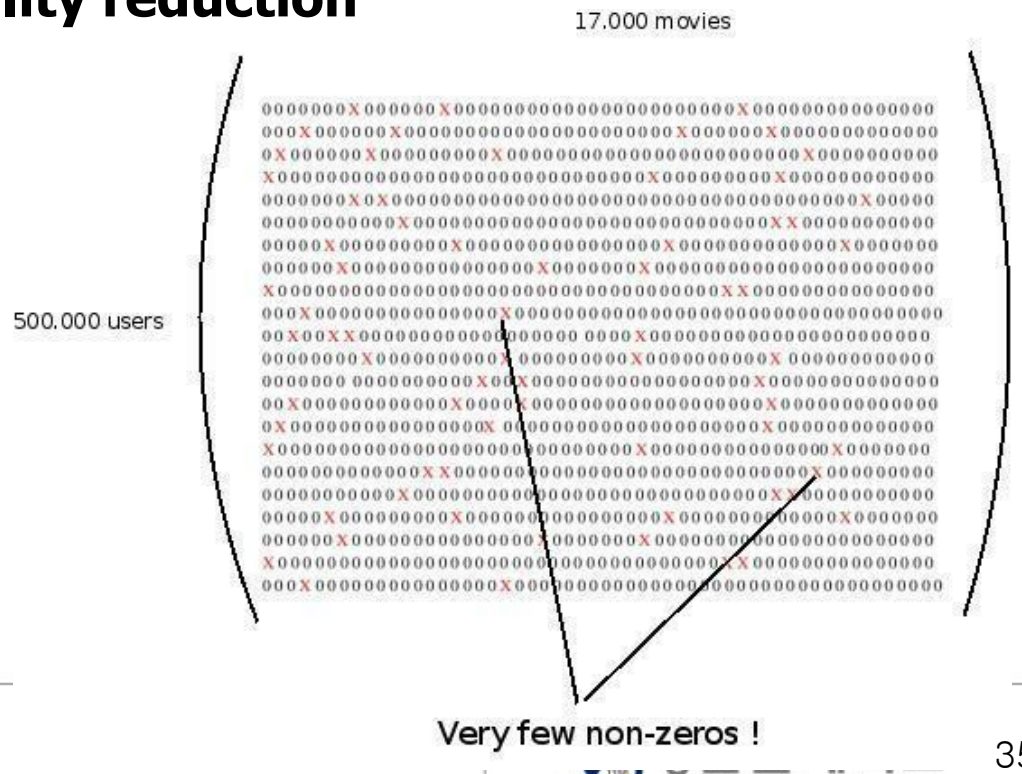
- **Basic Techniques**
  - Dimension reduction (Matrix Factorization)
  - Clustering
  - Association rule mining
  - Restricted Boltzmann machine
  - Probabilistic models
  - Various machine learning approaches

Matrix Factorization

- **Netflix 100M data**
  - Possibly 8,500M ratings (500,000 x 17,000)
  - But, there are only 100 M non-zero ratings

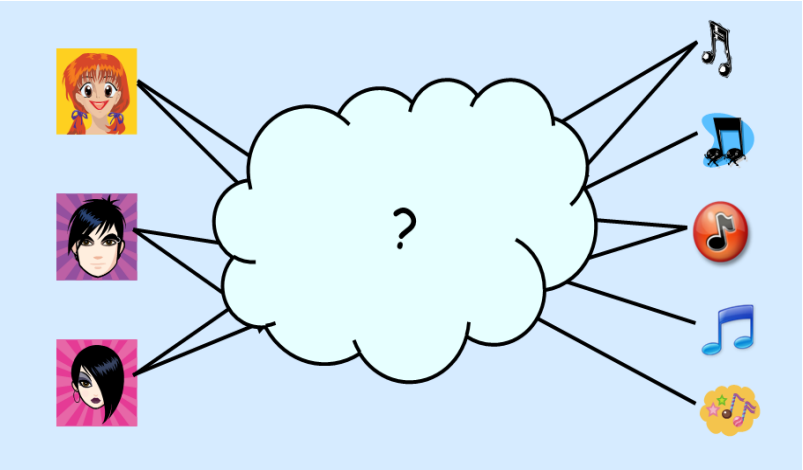
- **Methods of dimensionality reduction**
  - Matrix Factorization
  - Clustering
  - Projection (PCA…)

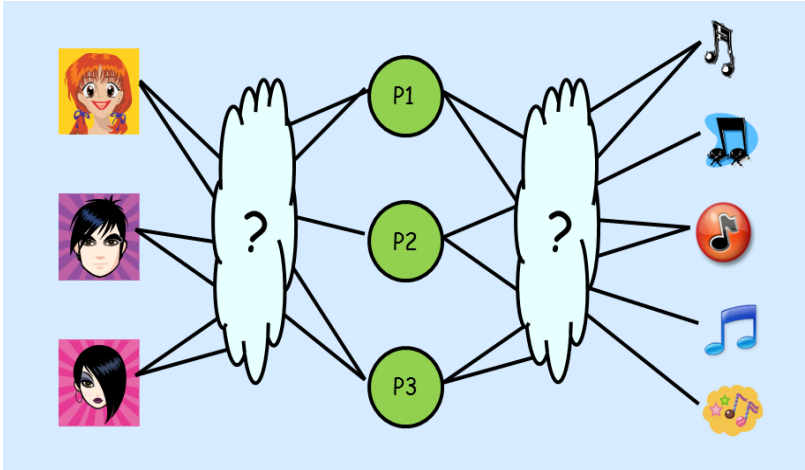
- **Space complexity**
  - Worst case: O(mn)
  - In practice: O(m + n)



Matrix Factorization

- **Assume some latent factors in user preference**





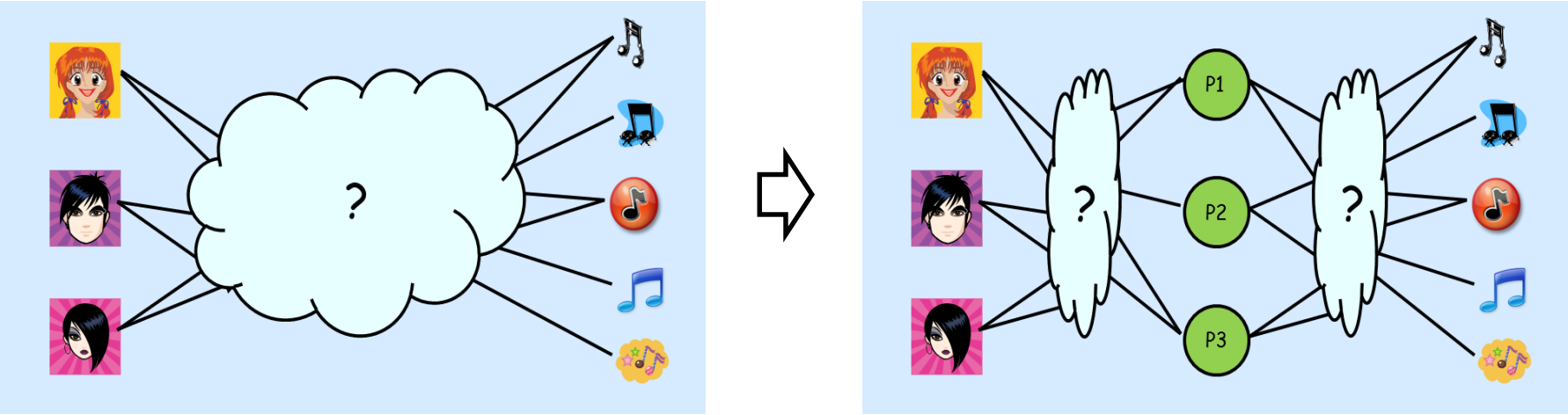
User 1 watched Item 1

User 1 likes Romance
Item 1 is Romance
So, User 1 watched Item1

Unfortunately, we do not know about such latent factors !!

Matrix Factorization

- **Assume some latent factors in user preference**



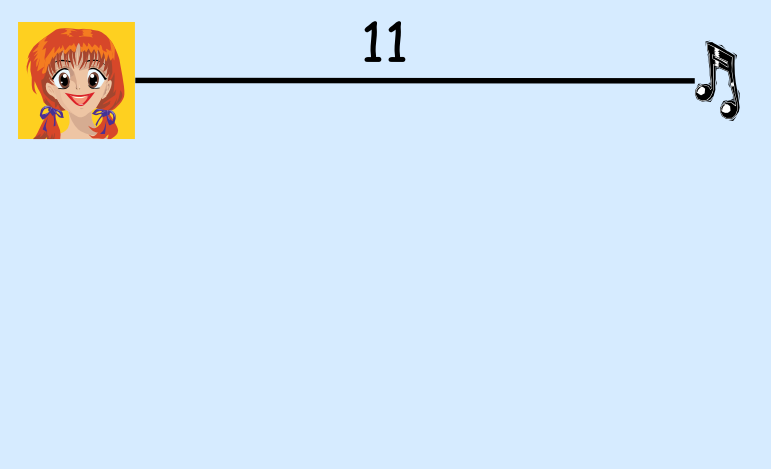
	I1	I2	I3	I4
U1		2	4	3
U2	2			
U3	3			4
U4	1		3	

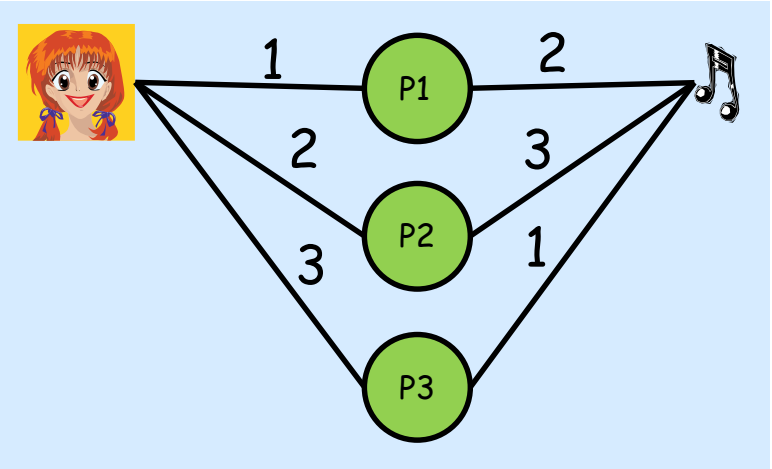
	P1	P2	P3
U1	2	1	3
U2	2	0	2
U3	3	3	5
U4	1	0	3

	I1	I2	I3	I4
P1	1	2	4	0
P2	2	2	0	5
P3	3	0	5	4

Matrix Factorization

- **Assume some latent factors in user preference**





$$1*2 + 2*3 + 3*1 = 11$$

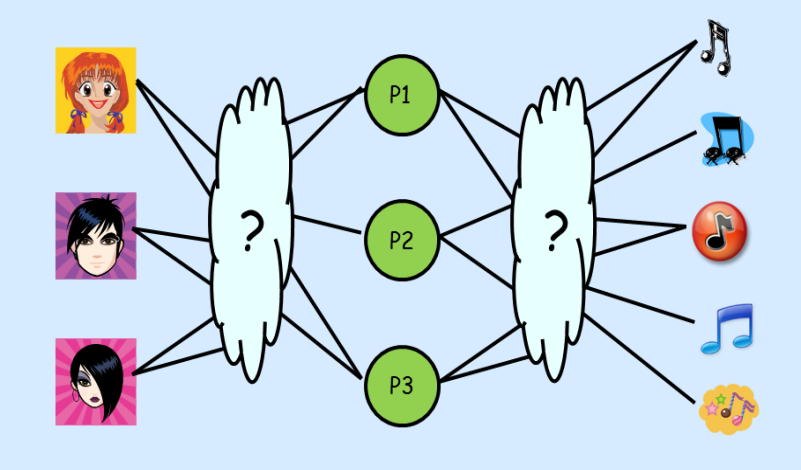
	P1	P2	P3
U1	1	2	3
U2			
U3			
U4			

	I1	I2	I3	I4
P1	2			
P2	3			
P3	1			

Matrix Factorization

- **Assume some latent factors in user preference**

	I1	I2	I3	I4
U1		2	4	3
U2	2			
U3	3			4
U4	1		3	



	I1	I2	I3	I4
U1	-0.061	1.791	4.048	3.062
U2	1.853	-0.493	0.113	0.141
U3	3.052	0.182	-0.041	3.947
U4	1.091	0.307	2.932	-0.088

	P1	P2	P3
U1	-1.915	0.981	-0.524
U2	-0.267	-0.563	0.887
U3	-1.468	-1.581	-0.128
U4	-0.834	0.71	1.146

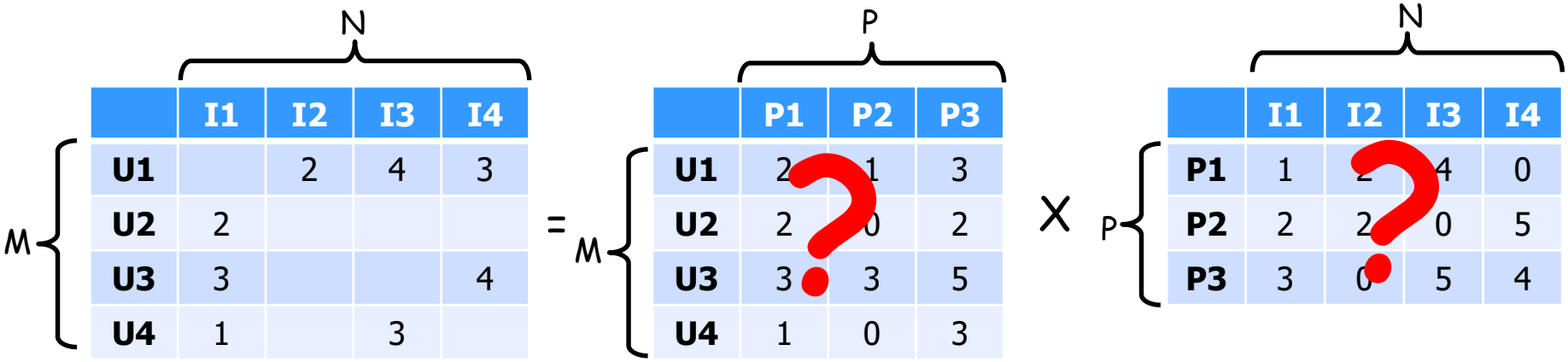
	I1	I2	I3	I4
P1	-0.875	-0.58	-1.543	-1.764
P2	-1.204	0.459	1.413	-0.788
P3	1.061	-0.439	0.56	-0.872

Matrix Factorization

- **However, we do not know about latent factors**
  - We just blindly decompose the matrix

$$R_{M\times N} \Rightarrow U_{M\times P} \bullet I_{P\times N}$$

  - Try to find $U_{M\times P}$ and $I_{P\times N}$ so that $R_{M\times N} = U_{M\times P} \bullet I_{P\times N}$

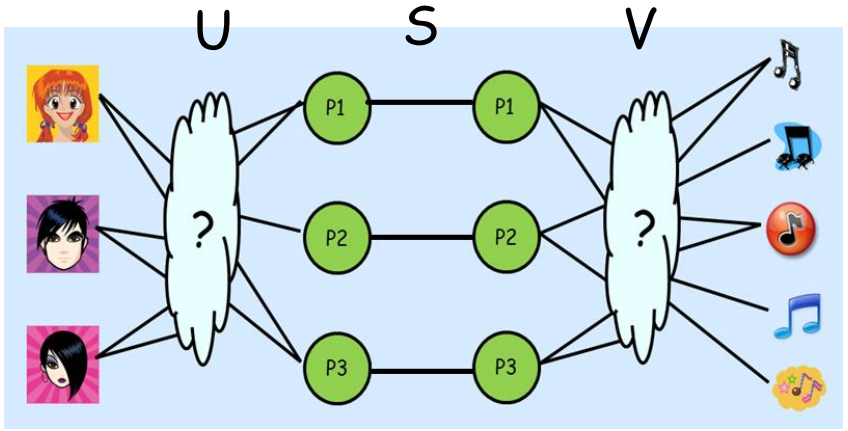


Singular Value Decomposition

- **Any matrix can be decomposed into**

$$\underset{m \times n}{X} = \underset{m \times r}{U} \underset{r \times r}{S} \underset{r \times n}{V^{\mathsf{T}}}$$

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix} \begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix} \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}$$



Singular Value Decomposition

- **Example**

$$
\begin{bmatrix}
4.000 & 1.000 & 2.000 & 3.000 & 4.000 \\
0.000 & 1.000 & 6.000 & 4.000 & 8.000 \\
9.000 & 4.000 & 3.000 & 6.000 & 1.000 \\
3.000 & 9.000 & 6.000 & 5.000 & 2.000 \\
9.000 & 2.000 & 5.000 & 8.000 & 2.000
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
-0.277 & -0.094 & 0.301 & -0.701 & 0.577 \\
-0.333 & -0.821 & 0.304 & 0.098 & -0.335 \\
-0.502 & 0.438 & 0.041 & -0.373 & -0.645 \\
-0.484 & -0.187 & -0.833 & 0.014 & 0.189 \\
-0.570 & 0.300 & 0.347 & 0.600 & 0.323
\end{bmatrix}
\times
\begin{bmatrix}
22.366 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 9.168 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 7.076 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 2.153 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.070
\end{bmatrix}
\times
\begin{bmatrix}
-0.546 & -0.363 & -0.439 & -0.544 & -0.286 \\
0.621 & -0.027 & -0.374 & 0.056 & -0.686 \\
0.311 & -0.853 & -0.101 & 0.138 & 0.382 \\
-0.332 & -0.358 & 0.533 & 0.428 & -0.542 \\
-0.330 & 0.106 & -0.611 & 0.707 & 0.089
\end{bmatrix}
$$

Can be interpreted as importance of factors
We can drop some less important ones !!

Singular Value Decomposition

- **Row rank approximation**

$$
\begin{bmatrix}
4.000 & 1.000 & 2.000 & 3.000 & 4.000 \\
0.000 & 1.000 & 6.000 & 4.000 & 8.000 \\
9.000 & 4.000 & 3.000 & 6.000 & 1.000 \\
3.000 & 9.000 & 6.000 & 5.000 & 2.000 \\
9.000 & 2.000 & 5.000 & 8.000 & 2.000
\end{bmatrix}
\approx
\begin{bmatrix}
-0.277 & -0.094 & 0.301 & -0.701 & 0.577 \\
-0.333 & -0.821 & 0.304 & 0.098 & -0.335 \\
-0.502 & 0.438 & 0.041 & -0.373 & -0.645 \\
-0.484 & -0.187 & -0.833 & 0.014 & 0.189 \\
-0.570 & 0.300 & 0.347 & 0.600 & 0.323
\end{bmatrix}
\times
\begin{bmatrix}
22.366 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 9.168 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 7.076 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000
\end{bmatrix}
\times
\begin{bmatrix}
-0.546 & -0.363 & -0.439 & -0.544 & -0.286 \\
0.621 & -0.027 & -0.374 & 0.056 & -0.686 \\
0.311 & -0.853 & -0.101 & 0.138 & 0.382 \\
-0.332 & -0.358 & 0.533 & 0.428 & -0.542 \\
-0.330 & 0.106 & -0.611 & 0.707 & 0.089
\end{bmatrix}
=
\begin{bmatrix}
3.510 & 0.455 & 2.827 & 3.616 & 3.177 \\
0.061 & 1.072 & 5.868 & 3.927 & 8.115 \\
8.715 & 3.720 & 3.398 & 6.373 & 0.567 \\
3.013 & 9.003 & 5.989 & 4.979 & 2.020 \\
9.432 & 2.460 & 4.320 & 7.428 & 2.698
\end{bmatrix}
$$

Singular Value Decomposition

- **Row rank approximation**

$$\begin{bmatrix} 4.000 & 1.000 & 2.000 & 3.000 & 4.000 \\ 0.000 & 1.000 & 6.000 & 4.000 & 8.000 \\ 9.000 & 4.000 & 3.000 & 6.000 & 1.000 \\ 3.000 & 9.000 & 6.000 & 5.000 & 2.000 \\ 9.000 & 2.000 & 5.000 & 8.000 & 2.000 \end{bmatrix} \approx \begin{bmatrix} 3.510 & 0.455 & 2.827 & 3.616 & 3.177 \\ 0.061 & 1.072 & 5.868 & 3.927 & 8.115 \\ 8.715 & 3.720 & 3.398 & 6.373 & 0.567 \\ 3.013 & 9.003 & 5.989 & 4.979 & 2.020 \\ 9.432 & 2.460 & 4.320 & 7.428 & 2.698 \end{bmatrix}$$

$$\begin{bmatrix} -0.277 & -0.094 & 0.301 & -0.701 & 0.577 \\ -0.333 & -0.821 & 0.304 & 0.098 & -0.335 \\ -0.502 & 0.438 & 0.041 & -0.373 & -0.645 \\ -0.484 & -0.187 & -0.833 & 0.014 & 0.189 \\ -0.570 & 0.300 & 0.347 & 0.600 & 0.323 \end{bmatrix} \times \begin{bmatrix} 22.366 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 9.168 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 7.076 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix} \times \begin{bmatrix} -0.546 & -0.363 & -0.439 & -0.544 & -0.286 \\ 0.621 & -0.027 & -0.374 & 0.056 & -0.686 \\ 0.311 & -0.853 & -0.101 & 0.138 & 0.382 \\ -0.332 & -0.358 & 0.533 & 0.428 & -0.542 \\ -0.330 & 0.106 & -0.611 & 0.707 & 0.089 \end{bmatrix}$$

$$\begin{bmatrix} -0.277 & -0.094 & 0.301 \\ -0.333 & -0.821 & 0.304 \\ -0.502 & 0.438 & 0.041 \\ -0.484 & -0.187 & -0.833 \\ -0.570 & 0.300 & 0.347 \end{bmatrix} \times \begin{bmatrix} 22.366 & 0.000 & 0.000 \\ 0.000 & 9.168 & 0.000 \\ 0.000 & 0.000 & 7.076 \end{bmatrix} \times \begin{bmatrix} -0.546 & -0.363 & -0.439 & -0.544 & -0.286 \\ 0.621 & -0.027 & -0.374 & 0.056 & -0.686 \\ 0.311 & -0.853 & -0.101 & 0.138 & 0.382 \end{bmatrix}$$

Low dimensional representation !!

Prediction with SVD

- **One Problem**
  - How to deal with empty cells in user-item matrix

	I1	I2	I3	I4
U1		2	4	3
U2	2			
U3	3			4
U4	1		3	

  - We need to fill out the empty cells to apply SVD
  - How?
    - With 0's ?
    - With the average of the whole users?
    - With the average of each user ?

Prediction with SVD

- **One Problem**
  - Considering both user bias and item bias
    - There are users whose ratings are higher or lower than average
    - Also, there are items of which ratings are higher or lower than average
  - Example: Estimate John's rating of Titanic
    - The average rating over all movies is 3.7 starts
    - Titanic is rated 0.5 stars above the average movie
    - John tends to rate 0.3 starts lower than the average

    =>Titanic's rating by John would be (3.7+0.5-0.3)

Prediction with SVD

- **One Problem**
  - We may assume that

$$r_{u,i} = \mu + b_u + b_i + (\text{some unknown factors})$$

$$b_u = \mu_u - \mu$$

$$b_i = \mu_i - \mu$$

or

$$b_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - \mu_u)$$

μ : average of the whole users

μ_u : average of user u

μ_i : average of item i

U_i : users who watched item i

  - Two things from the assumption
    - We may fill out the empty cells with $r_{u,i} = \mu + b_u + b_i$
    - Rather than estimating $r_{u,i}$ than $(\text{some unknown factors})$

Prediction with SVD

- **Steps**
  - Normalizing rating matrix *R*
  - Decompose *R* with SVD
  - Reconstruct an approximated matrix
  - De-normalizing the approximated matrix

Prediction with SVD

- **Steps: Normalization**
  - Construct a matrix ***b***
  - Subtract ***b*** from ***R***

$$b_u = \frac{1}{|I_u|}\sum_{i \in I_u}(r_{u,i} - \mu) \qquad b_i = \frac{1}{|U_i|}\sum_{u \in U_i}(r_{u,i} - b_u - \mu)$$

$$b_{u,i} = \mu + b_u + b_i$$

R

	I1	I2	I3	I4	I5
U0	5	3			4
U1		3	4	3	
U2	2	2		5	4
U3	4		5	3	
U4	1		3	2	

$\mu + b_u + b_i + (\text{some unknown factors})$

b

	I1	I2	I3	I4	I5
U0	3.7	3.1			4.4
U1		2.5	4.2	3.4	
U2	2.9	2.4		3.4	3.6
U3	3.7		4.9	4.1	
U4	1.7		2.9	2.1	

$\mu + b_u + b_i$

Prediction with SVD

- **Steps: Low rank approximation**
  - Fill out empty cells with 0
  - Decompose and approximate ***R-b*** with SVD

R-b (unknown factors)

	I1	I2	I3	I4	I5
U0	1.3	-0.1	0.0	0.0	-0.4
U1	0.0	0.5	-0.2	-0.4	0.0
U2	-0.9	-0.4	0.0	1.6	0.4
U3	0.3	0.0	0.1	-1.1	0.0
U4	-0.7	0.0	0.1	-0.1	0.0

(some unknown factors)

Approximation

	I1	I2	I3	I4	I5
U0	1.3	-0.1	0.0	0.0	-0.3
U1	0.0	0.1	0.0	-0.5	0.0
U2	-0.9	-0.4	0.0	1.6	0.3
U3	0.3	0.2	0.0	-1.0	-0.1
U4	-0.7	0.1	0.0	-0.1	0.2

Estimation of (some unknown factors)

Prediction with SVD

- **Steps: De-normalization**
  - Predict by adding b_{ui} into the approximated matrix

Approximation

	I1	I2	I3	I4	I5
U0	1.3	-0.1	0.0	0.0	-0.3
U1	0.0	0.1	0.0	-0.5	0.0
U2	-0.9	-0.4	0.0	1.6	0.3
U3	0.3	0.2	0.0	-1.0	-0.1
U4	-0.7	0.1	0.0	-0.1	0.2

Estimation of (some unknown factors)

Prediction

	I1	I2	I3	I4	I5
U0	5.0	3.0	4.9	4.1	4.0
U1	3.0	2.6	4.2	2.9	3.7
U2	2.0	2.0	4.1	5.0	3.9
U3	4.0	3.4	4.9	3.1	4.2
U4	1.0	1.2	2.9	2.0	2.5

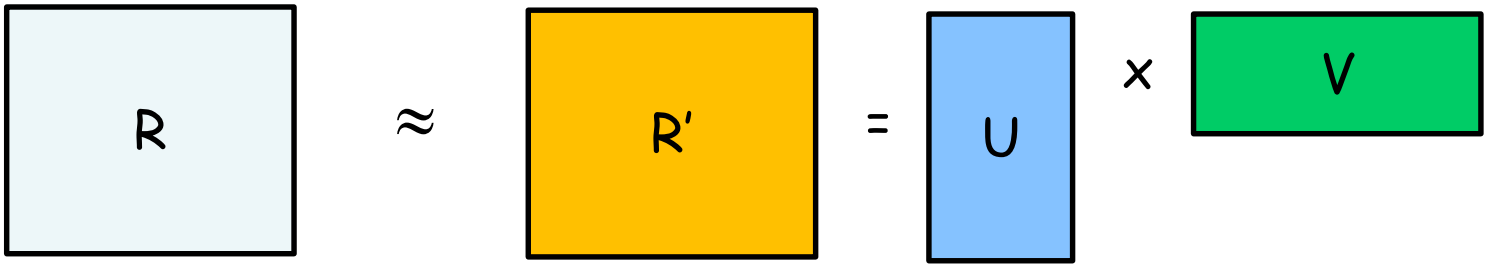
$$r_{u,i} = \mu + b_u + b_i$$

$+$ Estimation of (some unknown factors)

Matrix Factorization through Optimization

- **For given $R_{M\times N}$, we want to find $U_{M\times P}, V_{P\times N}$ such that**

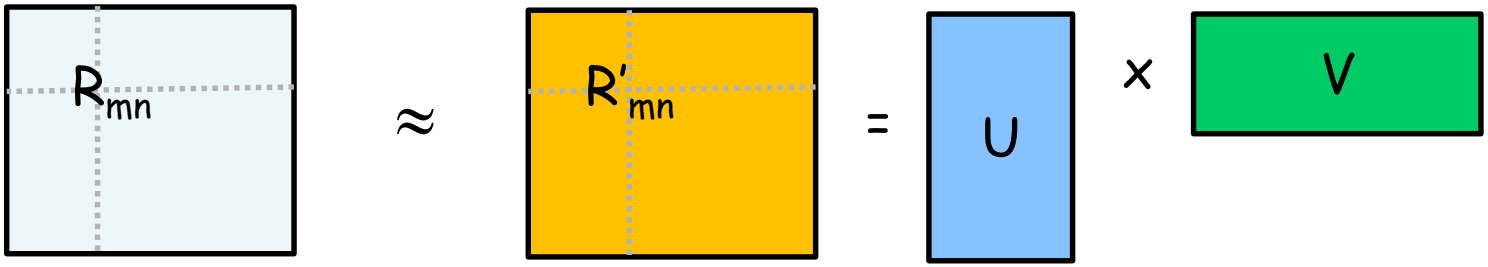
$$R_{M\times N} \approx R'_{M\times N} \text{ where } R'_{M\times N} = U_{M\times P} \bullet V_{P\times N}$$



Matrix Factorization through Optimization

- For given $R_{M\times N}$, we want to find $U_{M\times P}, V_{P\times N}$ such that

minimize the error between $R_{M\times N}$ and $R'_{M\times N}$



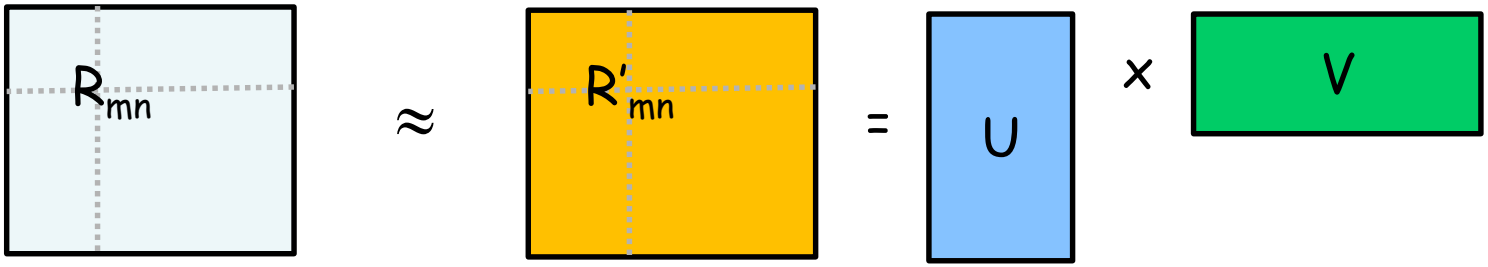
$$E(U,V)=\sum_{m,n}\left(R_{mn}-R'_{mn}\right)^2$$

Matrix Factorization through Optimization

- For given $R_{M\times N}$,

  we want to find as simple $U_{M\times P}, V_{P\times N}$ as possible

  such that minimize the error between $R_{M\times N}$ and $R'_{M\times N}$



$$E(U,V)=\sum_{m,n}\left(R_{mn}-R'_{mn}\right)^2+\lambda\left(\sum_{m,p}U_{mp}{}^2+\sum_{p,n}V_{pn}{}^2\right)$$

Matrix Factorization through Optimization

- **For given $R_{M\times N}$,**

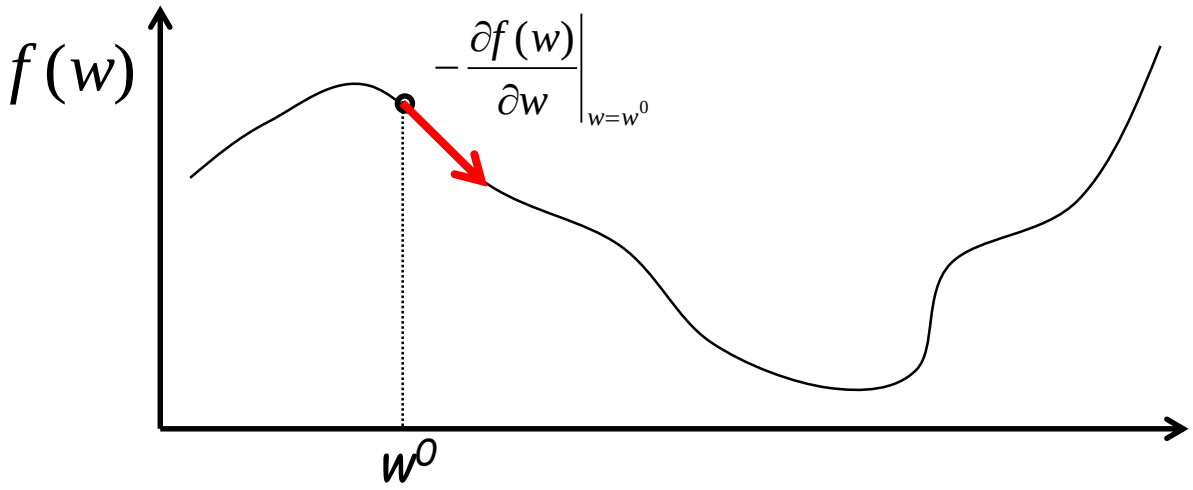
$$\arg\min_{U,V} E(U,V) = \sum_{m,n}\left(R_{mn} - R'_{mn}\right)^2 + \lambda\left(\sum_{m,p} U_{mp}{}^2 + \sum_{p,n} V_{pn}{}^2\right)$$

$$\arg\min_{U,V} E(U,V) = \left\|R - R'\right\|_F^2 + \lambda\left(\left\|U\right\|_F^2 + \left\|V\right\|_F^2\right)$$

- **But how?**
  - Any optimization technique is OK
  - Mostly gradient descent methods is used

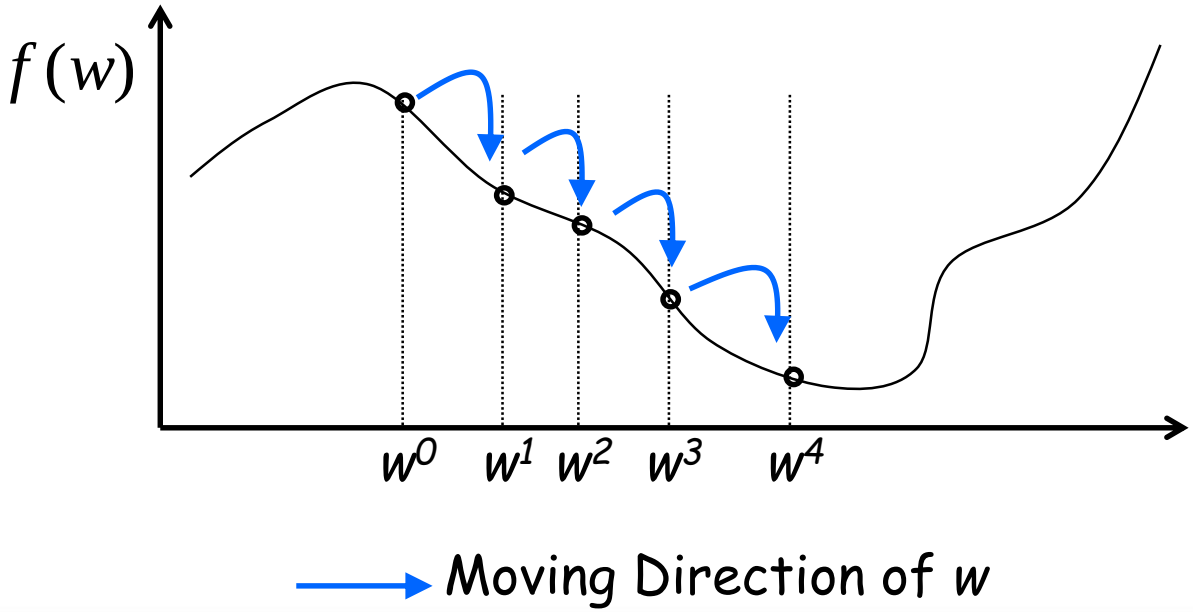
Matrix Factorization through Optimization

- **Gradient Descent Method**
  - Randomly choose an initial solution
  - Descend using the gradient information



Matrix Factorization through Optimization

- **Gradient Descent Method**
  - Randomly choose an initial solution
  - Descend using the gradient information



Matrix Factorization through Optimization

- **Gradient Descent Method**

Randomly choose an initial solution, w^0

Repeat

$$w^{t+1} = w^t - f'(w^t) \times \eta$$ → learning rate

Until stopping condition is satisfied

- $|w^{t+1} - w^t|$ is very small
- f(w) little moves
- fixed number of iterations

Matrix Factorization through Optimization

- **Find U and V to minimize**

$$E(U,V)=\sum_{m,n}\left(R_{mn}-R'_{mn}\right)^2+\lambda\left(\sum_{m,p}U_{mp}{}^2+\sum_{p,n}V_{pn}{}^2\right)$$

- **Solve with Gradient Descent Method**

for t = 1 to infinite
{
  for all m and p, $U_{mp}^{t+1}=U_{mp}^{t}-\lambda\left(\frac{\partial E(U,V)}{\partial U_{mp}}\right)_{U_{mp}=U_{mp}^{t}}$

  for all p and n, $V_{pn}^{t+1}=V_{pn}^{t}-\lambda\left(\frac{\partial E(U,V)}{\partial V_{pn}}\right)_{V_{pn}=V_{pn}^{t}}$
}

Matrix Factorization through Optimization

- **Advantage**
  - Very flexible
  - Easy to parallelize the optimization process (with GPUs)
  - Easy to combine with other factors

Matrix Factorization through Optimization

- **Prediction**
  - Find U and V approximating R

$$r_{u,i} \approx u_u \bullet v_i$$

  - Find P and Q approximating the following

$$r_{u,i} \approx \mu + b_u + b_i + p_u \bullet q_i$$

  - Find everything except μ approximating the following

$$r_{u,i} \approx \mu + b_u + b_i + p_u \bullet q_i$$

Matrix Factorization through Optimization

- **Target function**

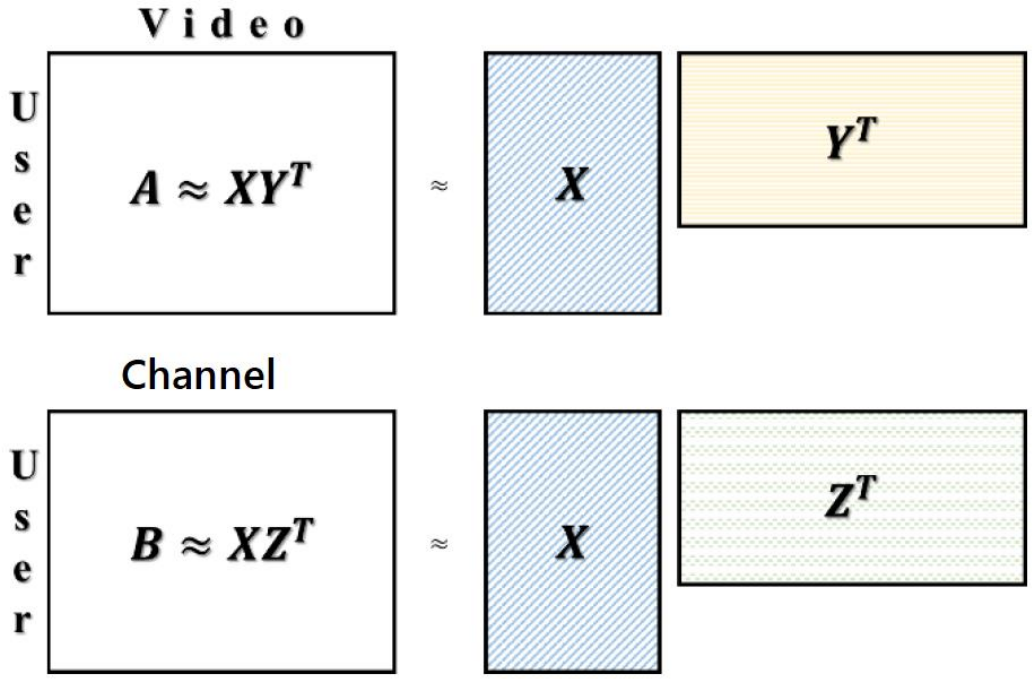
$$\min_{b_*, p_*, q_*} \sum_{(u,i)\in R} \left(r_{u,i} - \left(\mu + b_u + b_i + p_u \bullet q_i\right)\right)^2 + \lambda\left(b_u^2 + b_i^2 + \left\|p_u\right\|^2 + \left\|q_i\right\|^2\right)$$

- **Stochastic gradient descent learning rule**

$$b_u \leftarrow b_u + \eta\left(e_{u,i} - \lambda b_u\right)$$
$$b_i \leftarrow b_i + \eta\left(e_{u,i} - \lambda b_i\right)$$
$$p_u \leftarrow p_u + \eta\left(e_{u,i} \cdot p_u - \lambda q_i\right)$$
$$q_i \leftarrow b_u + \eta\left(e_{u,i} \cdot q_i - \lambda p_u\right)$$

Matrix Factorization through Optimization

- **Collective Matrix Factorization**



$$L(X,Y,Z) = \frac{1}{2}\|I_1 \circ (A - XY^T)\|_F^2 + \frac{\alpha}{2}\|I_2 \circ (B - XZ^T)\|_F^2 + \frac{\beta}{2}(\|X\|_F^2 + \|Y\|_F^2 + \|Z\|_F^2),$$

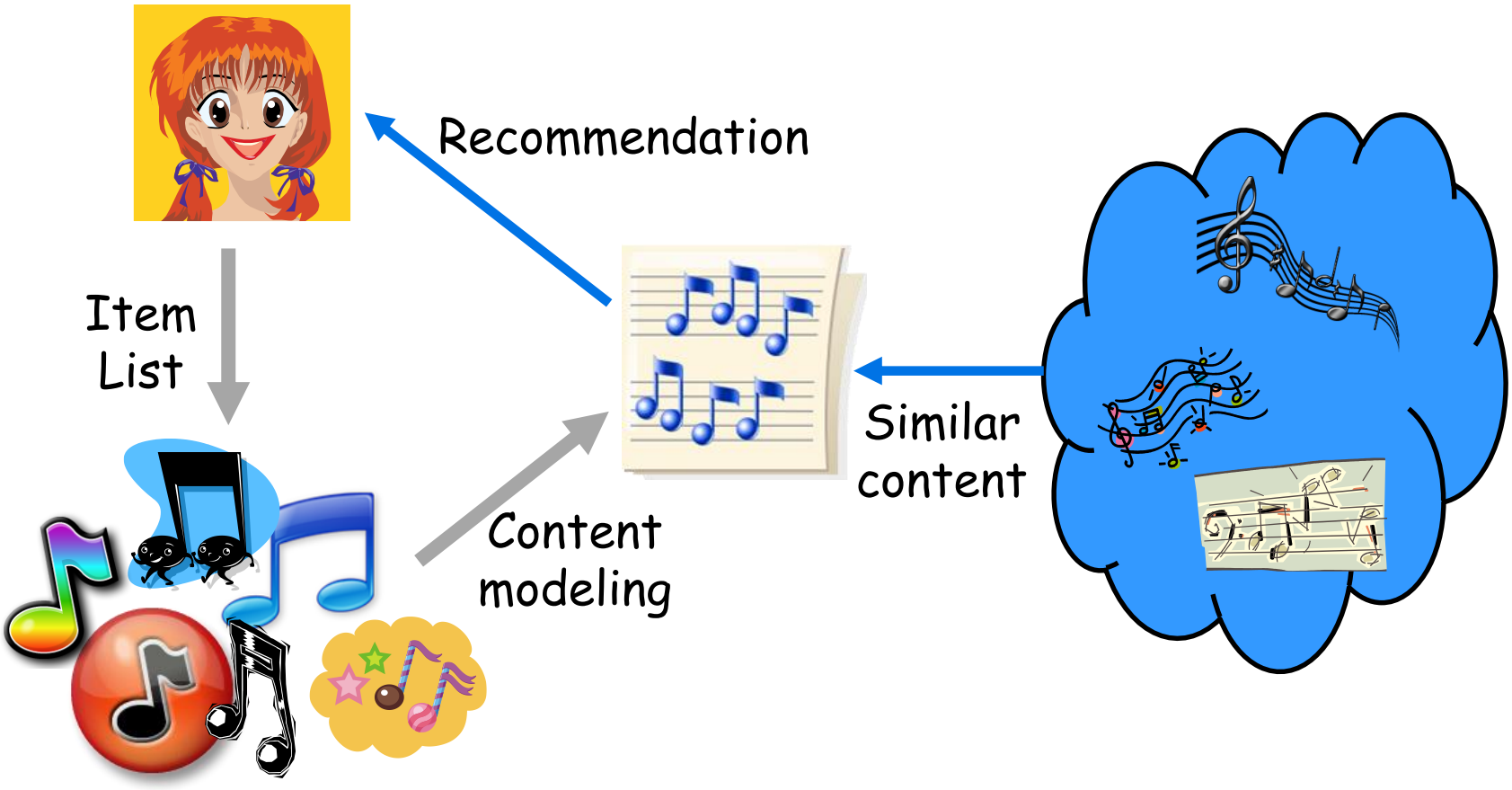
Summary of CF

- **Pros**
  - Requires minimal knowledge engineering efforts
  - No need of any internal structure or characteristics
- **Cons**
  - Requires a large number of reliable ratings
  - Assumes that prior behavior determines current behavior
  - Cold start problems: New user, new items
  - Sparsity problems

Outline

1. Introduction
2. Collaborative Filtering Overview
3. Memory based Collaborative Filtering
4. Model based Collaborative Filtering
5. **Content based Recommender System**
   - **5-1. Overview**
   - **5-2. Vector Model**
   - **5-3. tf-idf weighting**
   - **5-4. Similarity between Documents**
   - **5-5. Advantage and Disadvantage**
6. Summary

Overview


Recommendation
Item
List
Content
modeling
Similar
content

Overview

- **What's content?**
  - Explicit attributes or chracteristics (Eg for a movie)
    - Genre : Action / adventure
    - Feature : Bruce Willis
    - Year : 1995
  - Textual content (Eg for a book)
    - Title
    - Description
    - Table of content
  - **Any features or keywords which can describe items**

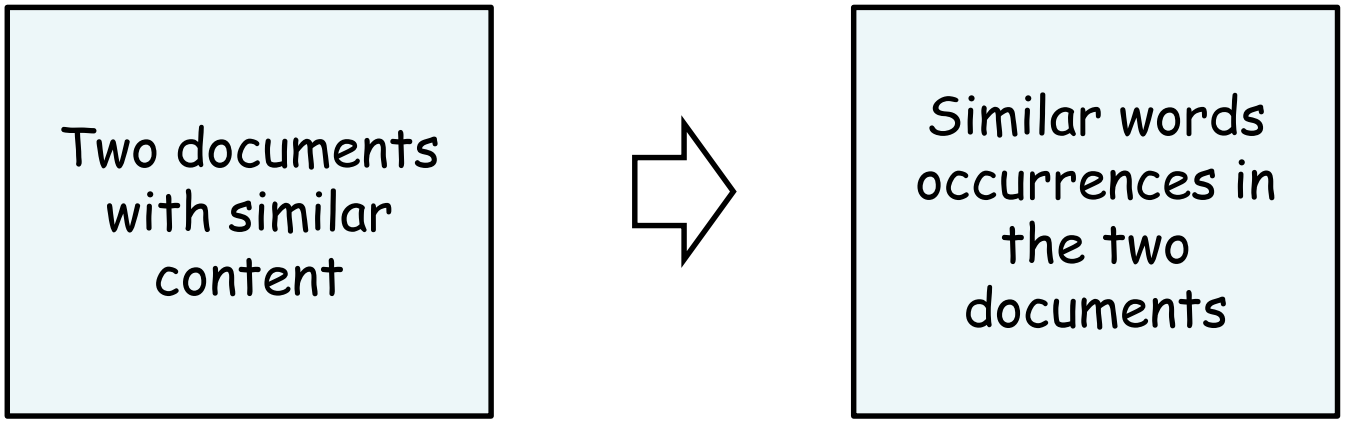
Overview

- **Basic assumption and idea**
  - Customers will like similar content which they liked in the past
- **Suitable for text-based products (web pages, book)**
  - Items are "described" by their features (e.g. keywords)
  - Users are described by the keywords in the items they bought
- **Characteristic**
  - Easy to apply to text-based products or products with text description
  - Based on match between the content (item keywords) and user keywords
  - Many machine learning approaches are applicable
    - Neural Networks, Naive Bayesian, Decision Tree, …

Overview

- **Two Questions**
  - How to represent the content of documents
  - How to compare the content similarity between documents

- **An assumption**



Vector Model

- **Bag of Words Model**
  - Counting the occurrences of a word in a document:
  - Each document is a count vector in $\mathbb{N}^{v}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	1
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Vector Model

- **Term Frequency Vector**

$$d = \left(\mathrm{tf}_{t_1,d}, \mathrm{tf}_{t_2,d}, \mathrm{tf}_{t_3,d}, \cdots, \mathrm{tf}_{t_N,d} \right)$$

$\mathrm{tf}_{w,d}$: term frequency of word t in d

N : the number of different words in the set of documents

  - More frequent terms are more relevant to the content of the document
  - But, it does not increase proportionally with term frequency

Vector Model

- **Log-frequency weighting**

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

tf	0	1	2	10	1000
w	0	1	1.3	2	4

Vector Model

- **But frequent words may be less informative**
  - Usually "a", "the", "is", etc. have high term frequencies
  - They are NOT so relevant to any content
- **Some less frequent words are more informative**
- **Which words are important?**
  - Words frequent in the document, but infrequent in the document collection
  - It is very probable that such words are strongly relevant to the content of the document

Vector Model

- **Term Frequency**
  - Represents how frequent a word in a document

- **Document Frequency (df_t)**
  - Represents how frequent a word *t* in a document collection
  - Defined as the number of documents that contain the word

  - Higher Document Frequency -> Less Informative
  - Lower Document Frequency -> More Informative

Vector Model

- **Inverse Document Frequency (idf_t)**

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

df_t = the number of documents containing word t

N = the number of documents in the collection

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Vector Model

- **Why Not Collection Frequency?**
  - Collection frequency of t is the number of occurrences of *t* in the collection

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

  - Which word is more informative?

tf-idf weighting

- **Definition**

$$w_{t,d} = \left(1 + \log \mathrm{tf}_{t,d}\right) \times \log \frac{N}{\mathrm{df}_t}$$

  - Increases with the number of occurrences within a document
  - Increases with the rarity of the term in the collection

tf-idf weighting

- **Term frequency vectors**

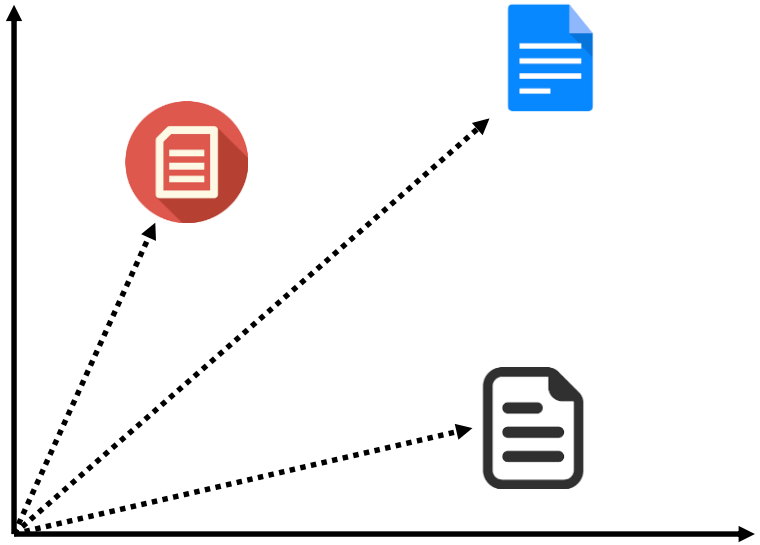
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	1
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- **tf-idf vectors**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

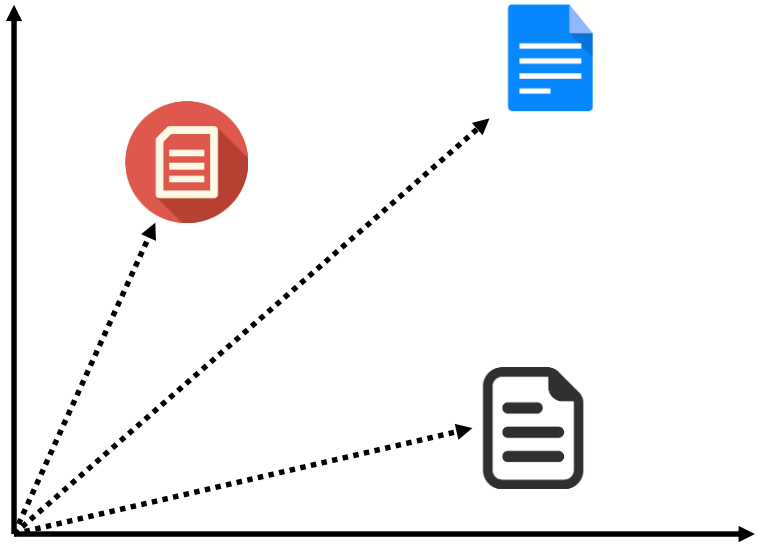
tf-idf weighting

- **So we have a |V|-dimensional vector space**
  - Terms are axes of the space
  - Documents are points or vectors in this space
  - Very high-dimensional and very sparse vectors



Similarity between Documents

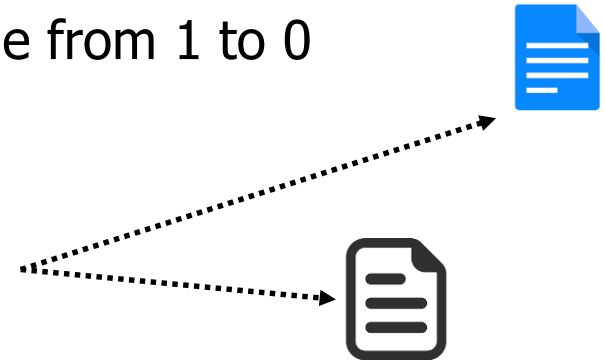
- **Meaning of vectors**
  - Length of a vector ≈ Length of a document
  - Orientation of a vector ≈ Content of a document



Similarity between Documents

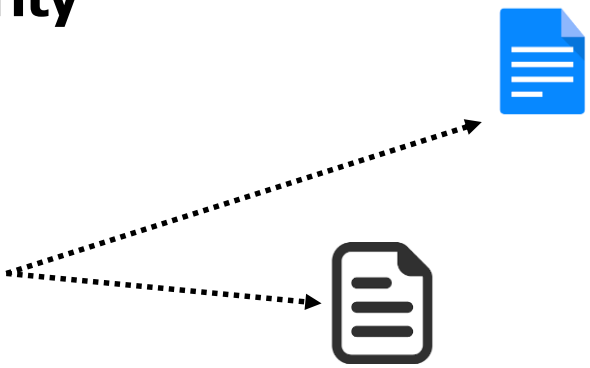
- **Similarity**
  - Angle between two vectors is close to 0^o -> two documents have very similar content
  - Angle between two vectors is close to 90^o -> two documents have irrelevant content each other

- **Angle from 0^o to 90^o**
  - Similarity 1 to 0
  - Cosine of the angle from 1 to 0



Similarity between Documents

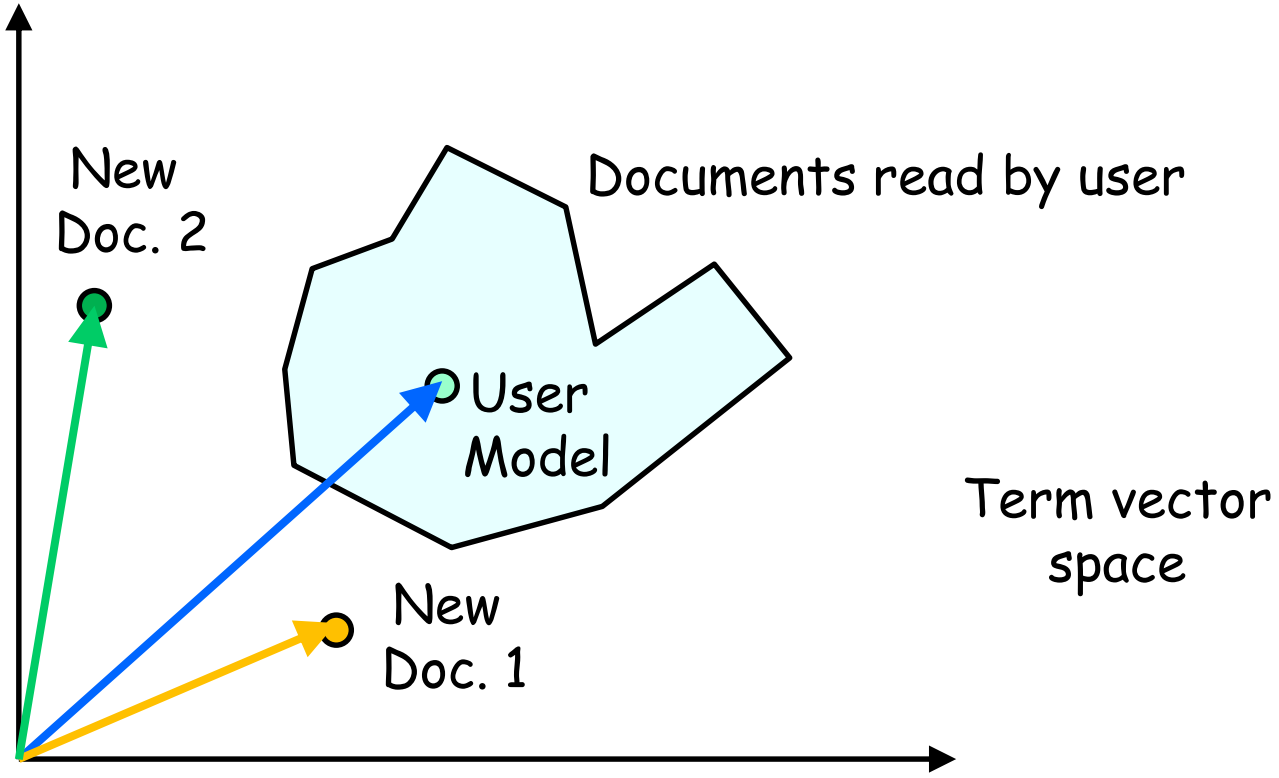
- **Cosine Similarity**



$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Similarity between Documents

- **Cosine Similarity**



Advantages of CBR

- **No need for data on other users**
  - No first-rater problem or sparsity problems
  - Able to recommend new and unpopular items
- **Able to recommend to users with unique preference**
- **Can provide explanations why it is recommended**
  - by listing content-features that caused an item to be recommended
- **Good to dynamically created items**
  - News, email, events, etc.

Disadvantages of CBR

- **Not easy to create content model for any products**
  - Book, web pages, news articles, music, video
- **Over-specialization**
  - Users are recommended with items similar to what they watched
  - No serendipity

Outline

1. Introduction
2. Collaborative Filtering Overview
3. Memory based Collaborative Filtering
4. Model based Collaborative Filtering
5. Content based Recommender System
6. **Summary**

Summary

- **Recommendation**
  - Collaborative Filtering
  - User-based Collaborative Filtering
  - Item-based Collaborative Filtering
  - Model-based Collaborative Filtering
  - Content based Recommender System

- **RS are fairly new but already grounded on well-proven technology**

- **However, there are still many open questions and a lot of interesting research to do**

Thank you for your attention

Q&A

Prediction with SVD

- **Steps: Fill out empty cells of sparse matrix**
  - In order to apply SVD, the matrix should be dense
  - Construct a matrix ***b***

	I1	I2	I3	I4	I5
U0	5	3			4
U1		2	4	3	
U2	2	2		5	3
U3	3		5	4	
U4	1		3	2	

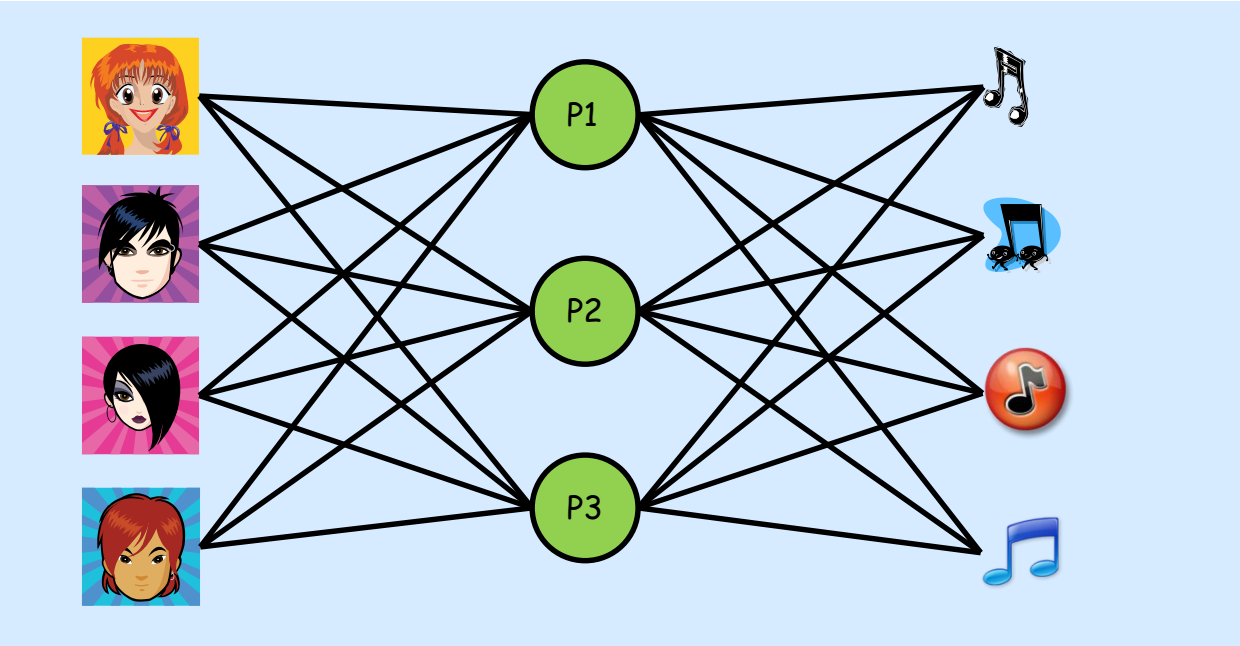
b

	I1	I2	I3	I4	I5
U0	3.6	3.1	4.8	4.3	4.3
U1	2.6	2.1	3.8	3.3	3.3
U2	2.6	2.1	3.8	3.3	3.3
U3	3.6	3.1	4.8	4.3	4.3
U4	1.6	1.1	2.8	2.3	2.3

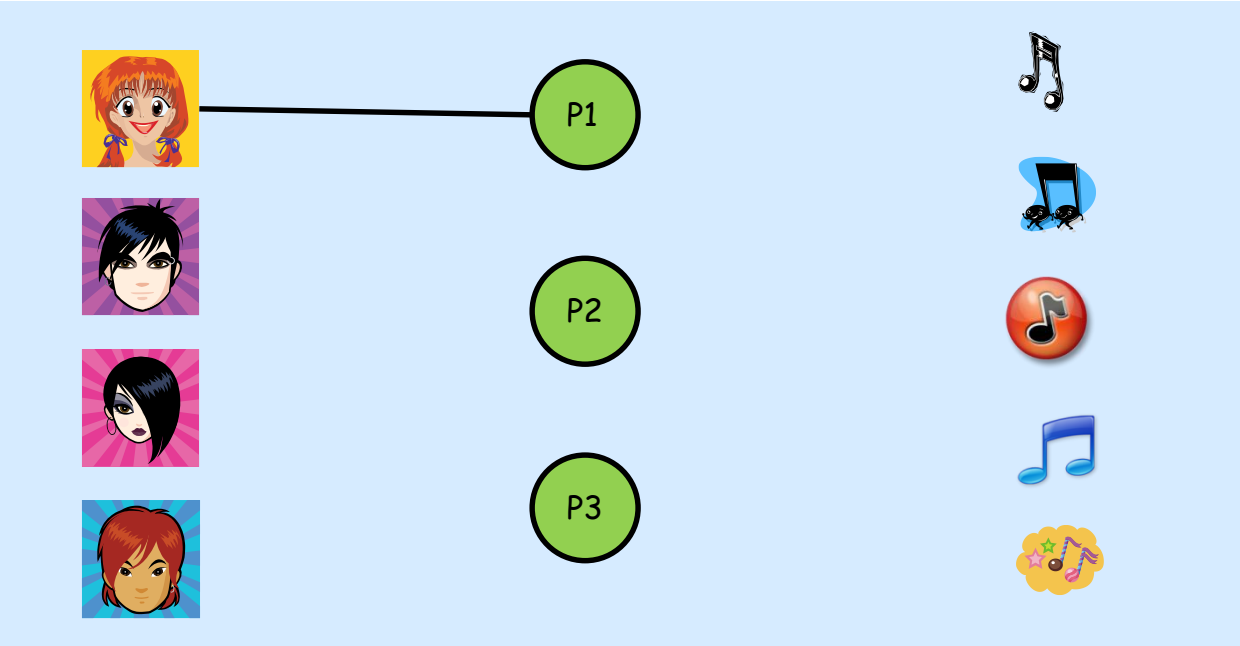
$$b_{u,i} = \mu + b_u + b_i$$

$$b_u = \frac{1}{|I_u|}\sum_{i \in I_u}(r_{u,i} - \mu) \qquad b_i = \frac{1}{|U_i|}\sum_{u \in U_i}(r_{u,i} - b_u - \mu)$$

Matrix Factorization



Matrix Factorization



Matrix Factorization