

Advanced Recommendation Approaches - Cross-Domain Recommendations

Shaghayegh Sahebi (Sherry)

sahebi@cs.pitt.edu

Outline

- Overview of Collaborative Filtering (CF)
- Advanced Approaches to CF
 - Matrix Factorization Techniques
 - Probabilistic Approaches
- Cross-domain Recommenders

Outline

Overview of Collaborative Filtering (CF)

- Advanced Approaches to CF
 - Matrix Factorization Techniques
 - Probabilistic Approaches
- Cross-domain Recommenders

Collaborative Filtering (CF) Review

- What is Alice's (a) rating (R) for Jane Ayre (j) movie?

$$R_{a,j} = ?$$



Collaborative Filtering (CF) Review

Average rating of *Alice* on all movies

$$R_{a,j} = \overline{V}_a$$

Collaborative Filtering (CF) Review

Average rating of *Alice*

$$R_{a,j} = \overline{v}_a + \alpha \sum_{i=1}^n w(a,i)(v_{i,j} - \overline{v}_i)$$

Adjusted by similar people's ratings on *Jane Ayre*

Collaborative Filtering (CF) Review

Average rating of *Alice* so far

Similarity of *Alice* to user *i*

How much does user *i* like *Jane Eyre* more than other movies?


$$R_{a,j} = \bar{v}_a + \alpha \sum_{i=1}^n w(a,i)(v_{i,j} - \bar{v}_i)$$

Adjusted by similar people's ratings on *Jane Ayre*

Collaborative Filtering (CF) Review

- Predict user ratings on items
- Using the opinions of other (similar) people

	M1	M2	M3	M4	M5
U1	?	?	?	5	?
U2	?	?	?	?	4
U3	3	?	?	?	?
U4	4	?	5	1	?
U5	?	4	4	1	?



Collaborative Filtering (CF) Review

- Predict user ratings on items
- Using the opinions of other (similar) people

	M1	M2	M3	M4	M5
U1	?	?	?	5	?
U2	?	?	?	?	4
U3	3	?	?	?	?
U4	4	4	5	1	?
U5	4	4	4	1	?

More Advanced Approaches to CF

- Dimensionality Reduction Algorithms
 - Matrix Factorization Methods such as Singular Value Decomposition (SVD), SVD++, Factorization Machines, etc.
- Probabilistic Methods
 - Such as Latent Dirichlet Allocation (LDA), Author-Topic Models, Gaussian Processes, etc.
- Other Approaches such as Matrix Completion

Outline

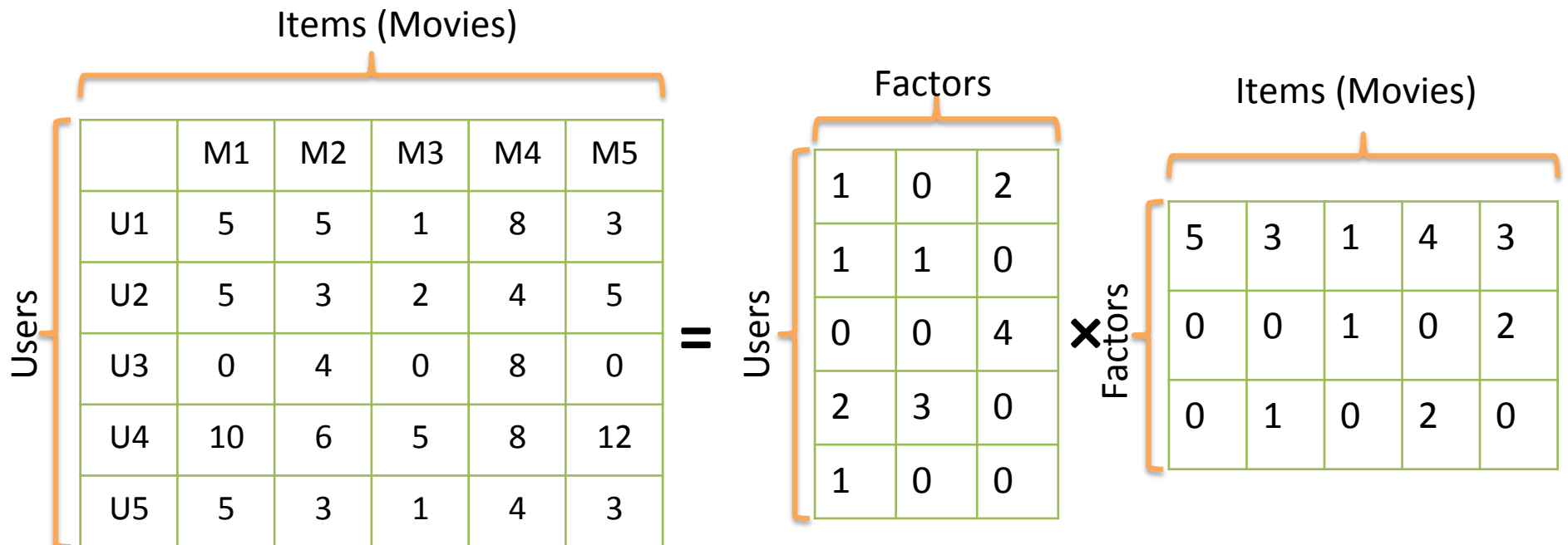
- Overview of Collaborative Filtering (CF)
- Advanced Approaches to CF
 - ➔ – Matrix Factorization Techniques
 - Probabilistic Approaches
- Cross-domain Recommenders

Matrix Factorization

- Any familiar names?
 - SVD, PCA, matrix factorization, spectral decomposition, eigenvalue decomposition
- Factorizing a matrix to products of lower dimensional matrices
 - Represent a big matrix with two smaller matrices
 - Which one takes more memory: a 10×6 matrix or a 10×3 matrix and a 3×6 matrix?
 - Create a common latent space that captures the similarity between features

Factorization Mostly Used in Recommender Systems

$$R_{(m \times n)} = P_{(m \times k)} Q_{(k \times n)}$$



Matrix Factorization Techniques

- Recommendation Goal: to fill in the unknown values of R
 - **consistent** with the existing ratings in the matrix

	M1	M2	M3	M4	M5
U1	?	2	?	5	2
U2	3	4	?	?	4
U3	3	?	3	?	4
U4	4	?	4	1	?
U5	?	4	4	1	?

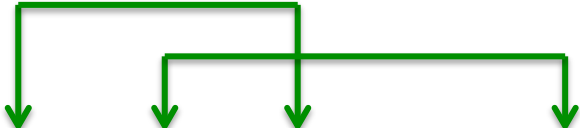
What does it mean “consistent”?

- Some users are more similar to each other

	M1	M2	M3	M4	M5
U1	?	2	?	5	2
U2	3	4	?	?	4
U3	3	?	3	?	4
U4	4	?	4	1	?
U5	?	4	4	1	?

What does it mean “consistent”?

- Items can also be similar to each other



	M1	M2	M3	M4	M5
U1	?	2	?	5	2
U2	3	4	?	?	4
U3	3	?	3	?	4
U4	4	?	4	1	?
U5	?	4	4	1	?

What does it mean “consistent”?

- How about interaction of users and items?
 - There is a common factor (**latent feature**) between “The Matrix” and “Inception” that users either like them both or not like them at all.

	M1	M2	M3	M4	M5
U1	?	2	?	5	2
U2	3	4	?	?	4
U3	3	?	3	?	4
U4	4	?	4	1	?
U5	?	4	4	1	?

What does it mean “consistent”?

- number of factors $<$ number of users
- number of factors $<$ number of items

(WHY?)

	M1	M2	M3	M4	M5
U1	?	2	?	5	2
U2	3	4	?	?	4
U3	3	?	3	?	4
U4	4	?	4	1	?
U5	?	4	4	1	?

Matrix Factorization Techniques

- Matrix Factorization can be used to discover interactions between users and items
 - With help of latent features (factors)

The diagram illustrates a user-item interaction matrix. The matrix has 5 rows (U1 to U5) and 5 columns (M1 to M5). Green arrows point from the top to columns M1, M2, M3, and M5. Orange arrows point from the left to rows U2, U3, U4, and U5.

	M1	M2	M3	M4	M5
U1	?	2	?	5	2
U2	3	4	?	?	4
U3	3	?	3	?	4
U4	4	?	4	1	?
U5	?	4	4	1	?

Matrix Factorization Techniques

- Discovering latent features -> predicting a rating of a certain user on a certain item (WHY?)

Some Math for Matrix Factorization

- A set U of m users, and a set D of n items
- R of size $m \times n$: the rating matrix
- We want to discover k latent factors
- Goal: to find two matrices $P_{m \times k}$ and $Q_{n \times k}$ such that their product approximates R

$$R \approx PQ^T = \hat{R}$$

Some Math for Matrix Factorization

- P : the strength of the associations between a user and the features
- Q : the strength of the associations between an item and the features.

$$R \approx PQ^T = \hat{R}$$

How to predict user ratings?

- To get the prediction of a rating of an item d_j by user u_i

$$R \approx PQ^T$$

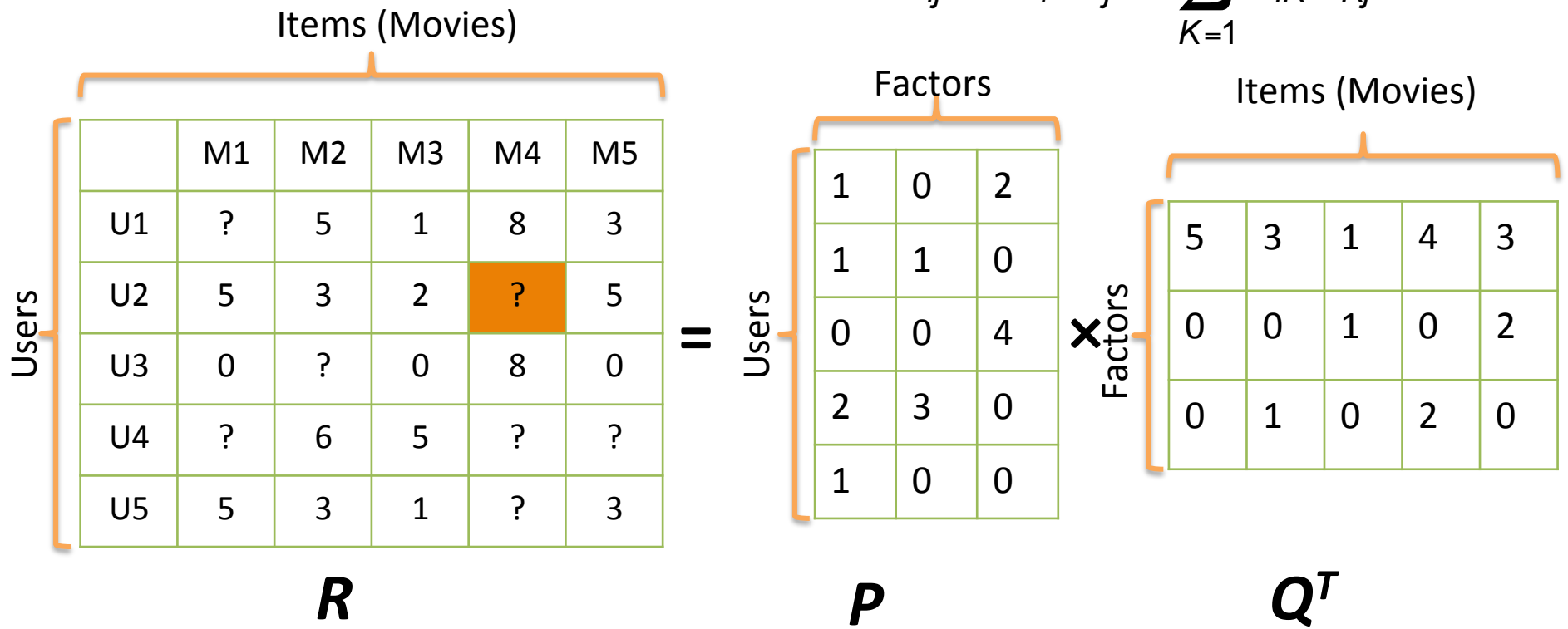


$$\hat{r}_{ij} = p_i^T q_j = \sum_{K=1}^k p_{iK} q_{Kj}$$

Example

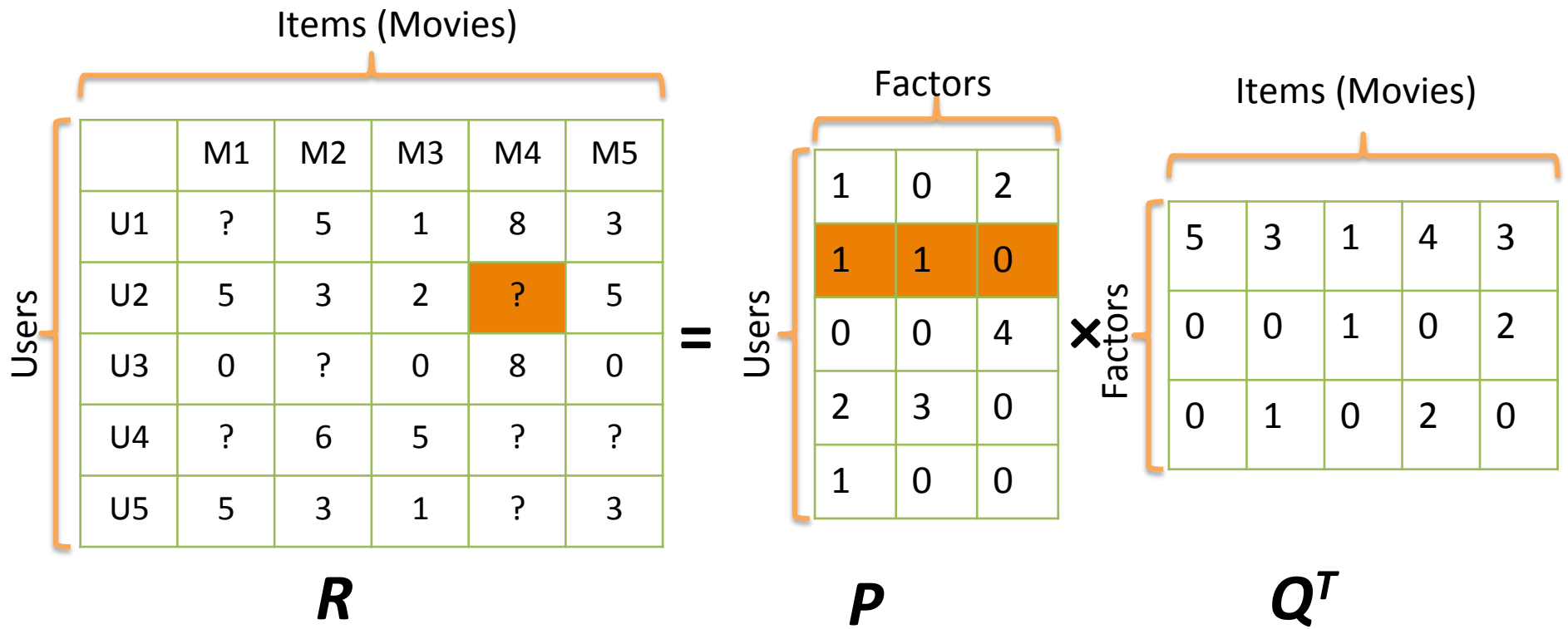
- What is the rating of user U_2 on movie M_4 ?

$$\hat{r}_{ij} = p_i^T q_j = \sum_{K=1}^k p_{iK} q_{Kj}$$



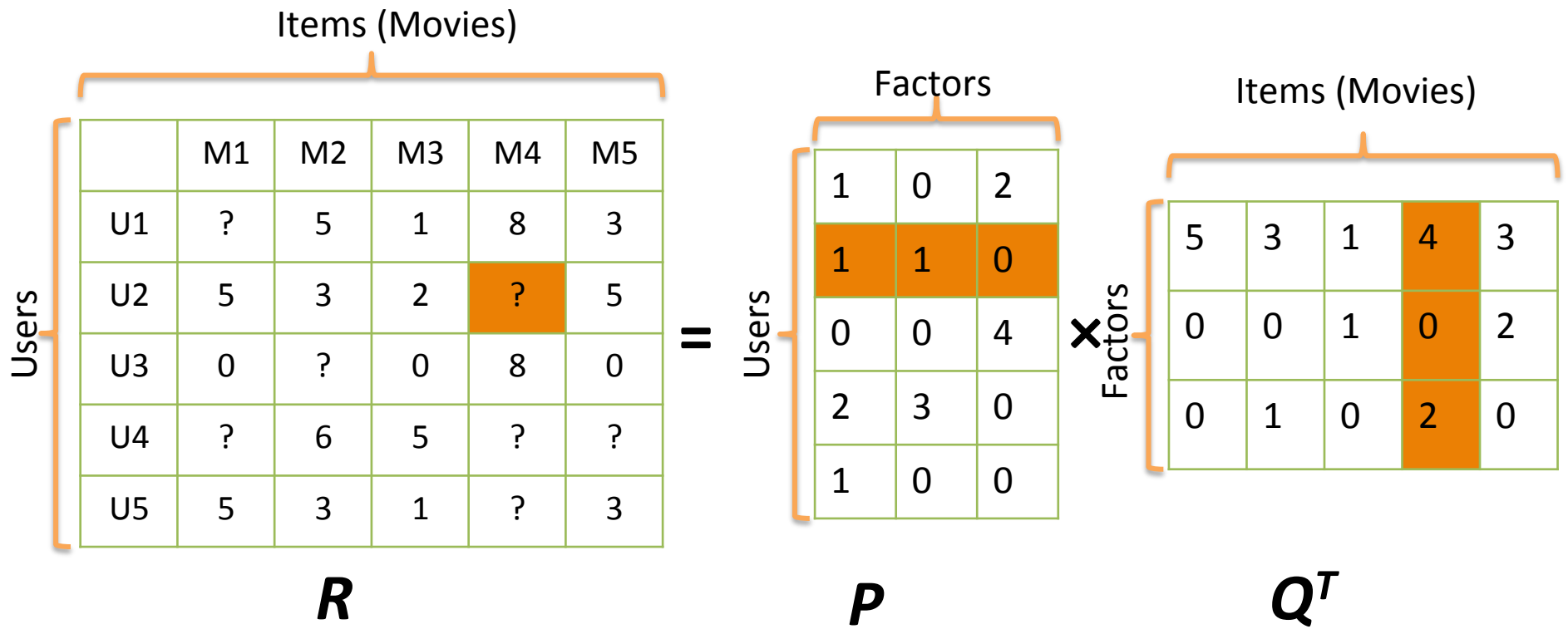
Example

- What is the rating of user U_2 on movie M_4 ?



Example

- What is the rating of user U_2 on movie M_4 ?



Example

- What is the rating of user U_2 on movie M_4 ?

$$\hat{r}_{ij} = p_i^T q_j = \sum_{K=1}^k p_{iK} q_{Kj}$$

$$P_2: \begin{array}{|c|c|c|} \hline 1 & 1 & 0 \\ \hline \end{array}$$

$$\hat{r}_{2,4} = p_2^T q_4 = \sum_{K=1}^3 p_{2K} q_{K4}$$

$$Q_4: \begin{array}{|c|c|c|} \hline 4 & 0 & 2 \\ \hline \end{array}$$

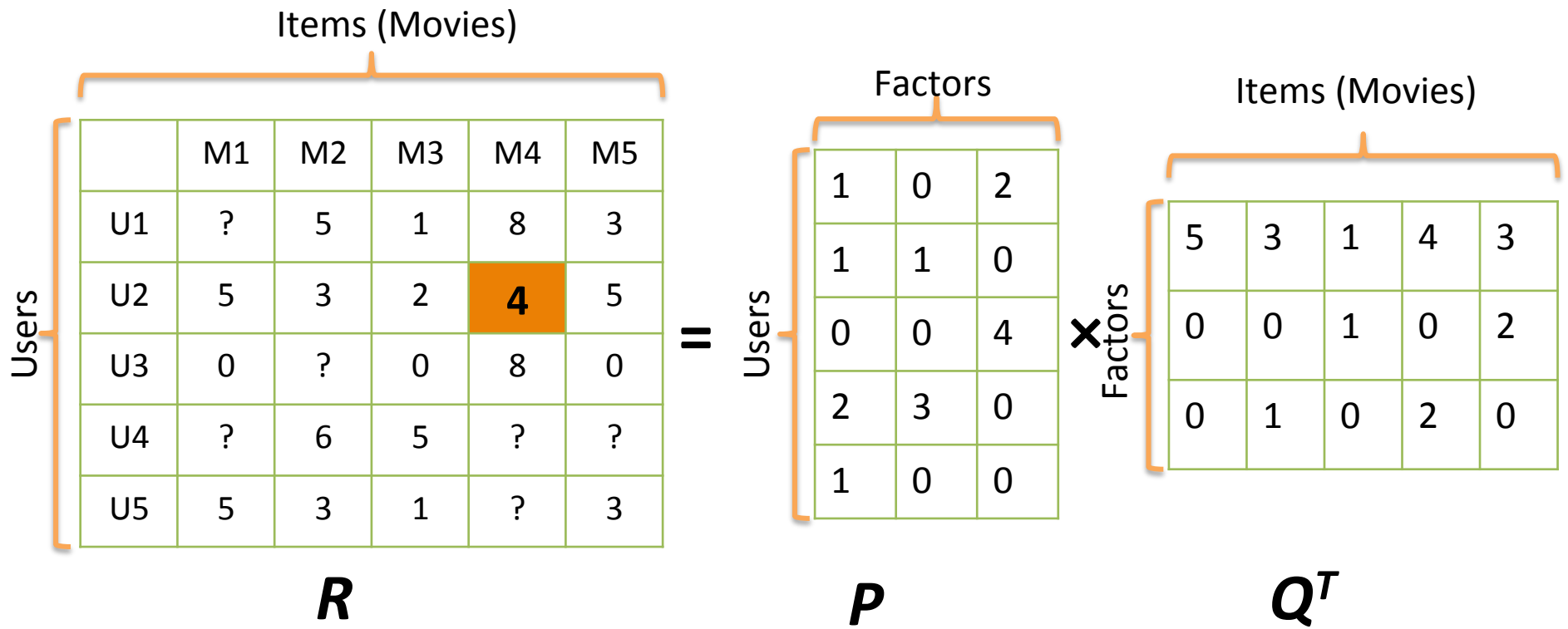
$$= p_{2,1} q_{1,4} + p_{2,2} q_{2,4} + p_{2,3} q_{3,4}$$

$$= 1 \times 4 + 1 \times 0 + 0 \times 2$$

$$= 4$$

Example

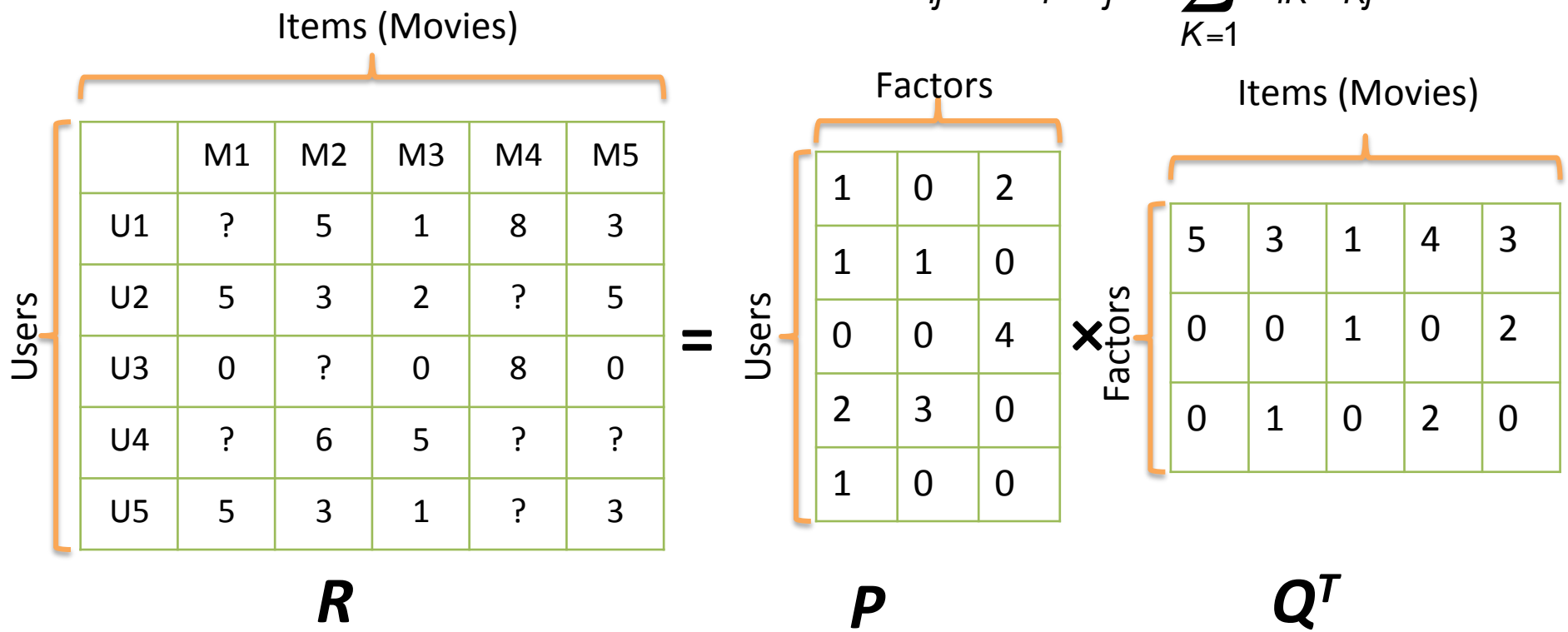
- What is the rating of user U_2 on movie M_4 ?



Activity

- What is the rating of user U_3 on movie M_2 ?

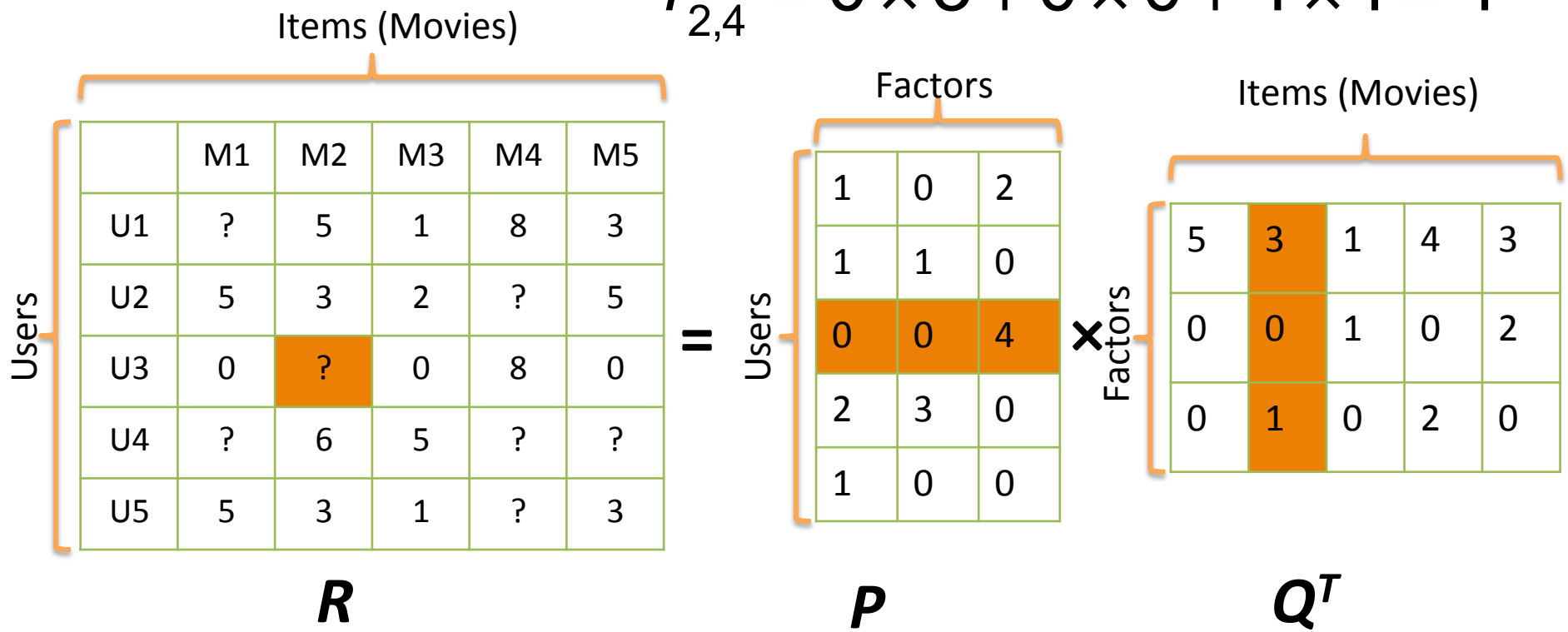
$$\hat{r}_{ij} = p_i^T q_j = \sum_{K=1}^k p_{iK} q_{Kj}$$



Activity

$$\hat{r}_{ij} = p_i^T q_j = \sum_{K=1}^k p_{iK} q_{Kj}$$

$$\hat{r}_{2,4} = 0 \times 3 + 0 \times 0 + 4 \times 1 = 4$$



How to obtain P and Q ?

- One way: Gradient Descent method
 - Initialize the two P and Q matrices with some values
 - Calculate how 'different' their product is to R
 - Try to minimize this difference iteratively
- **Difference**: error between the estimated rating (\hat{r}) and the real rating (r)

How to calculate the error?

- The common error calculation for each user-item pair:

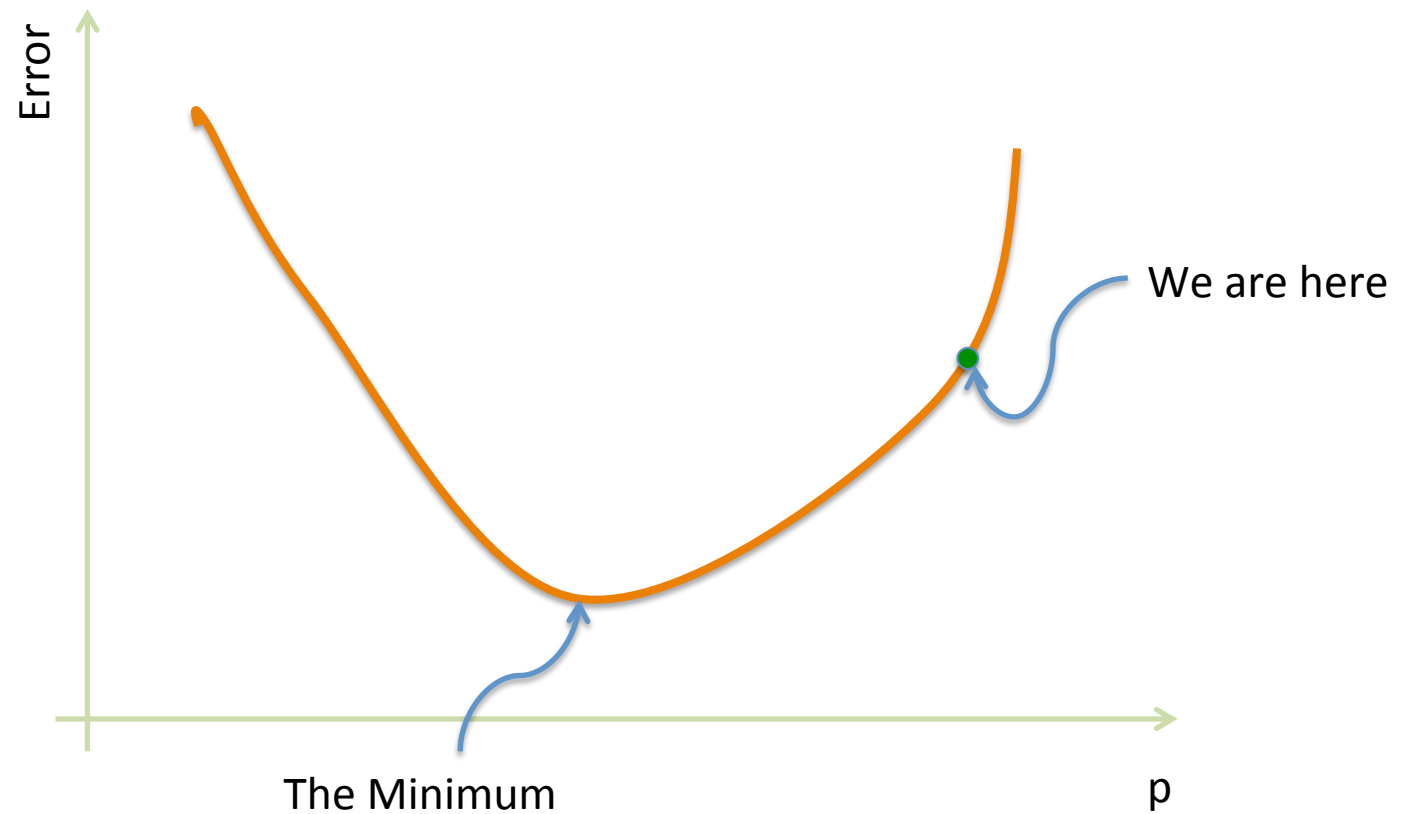
$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{K=1}^k p_{iK} q_{Kj})^2$$

- Why do we use squared error?

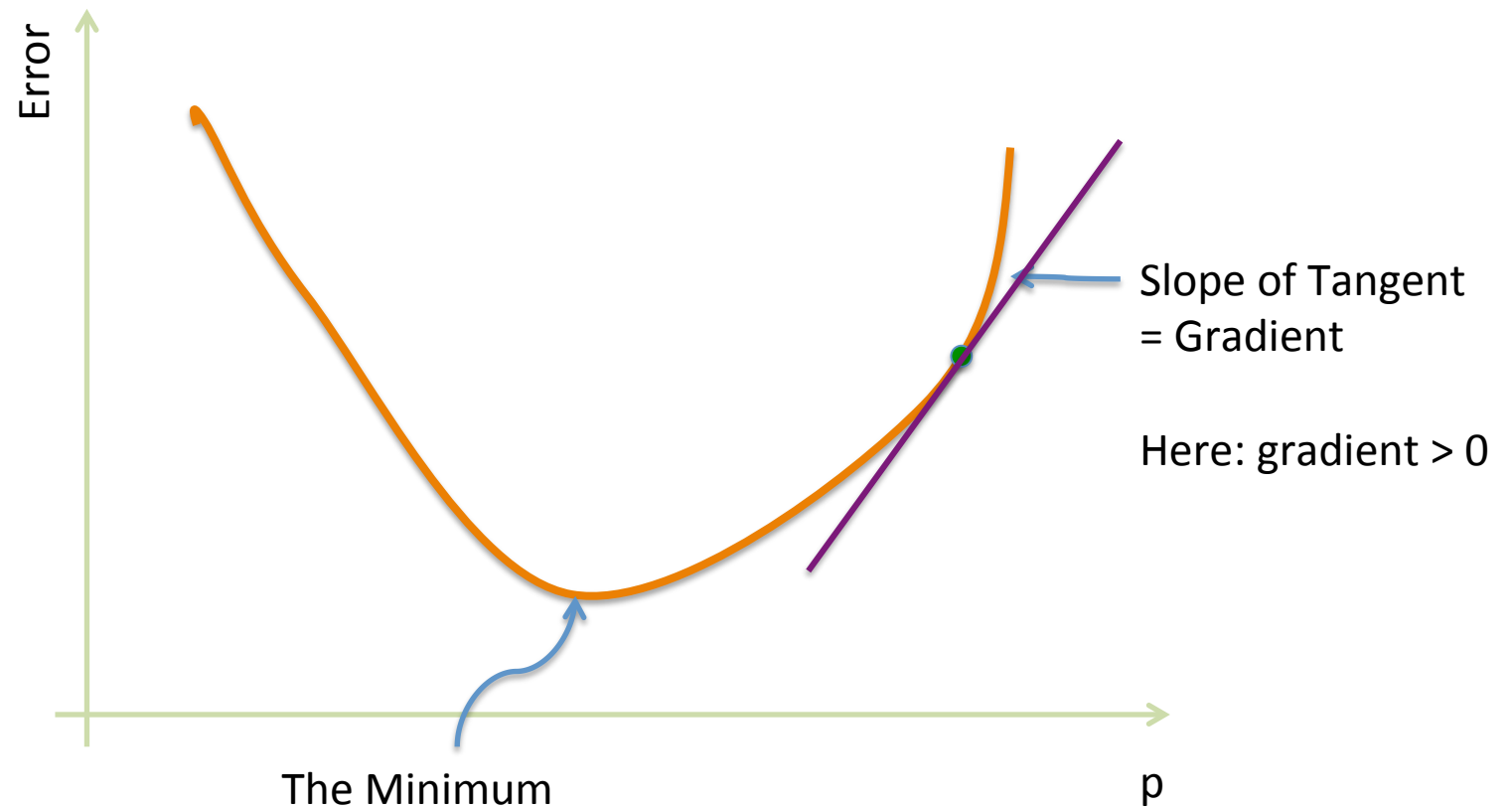
How to obtain P and Q ? (Cont'd)

- We should minimize the error in each iteration
 - Which direction to go?
 - Increase the values or decrease them?
 - How much to go?
- Decide based on the gradient at the current values

Gradient Descent

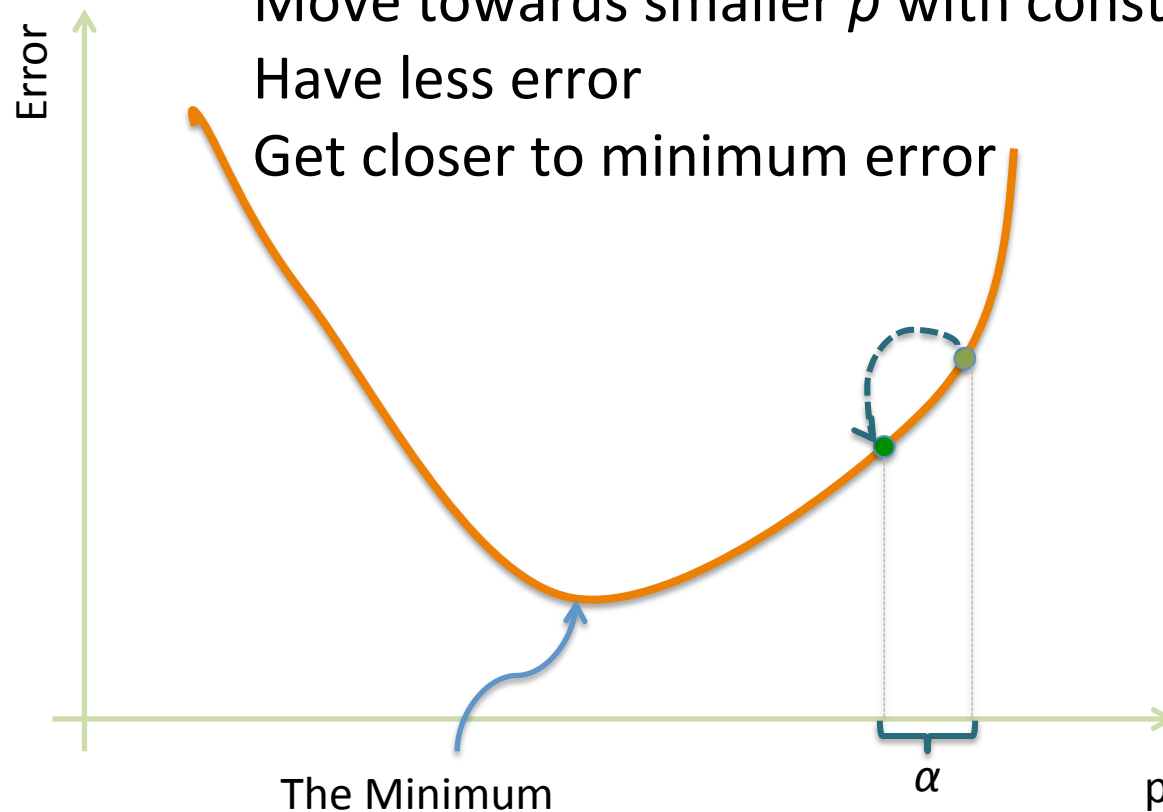


Gradient Descent



Gradient Descent

Move towards negative of gradient
Move towards smaller p with constant jump α
Have less error
Get closer to minimum error



How to obtain P and Q ? (Cont'd)

- Differentiate the error based on p and q

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = \left(r_{ij} - \sum_{k \in I_{ij}} p_{ik} q_{kj} \right)^2$$

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik}$$

- Update p and q values

$$p'_{ik} = p_{ik} - \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} - 2\alpha e_{ij} q_{kj}$$

$$q'_{kj} = q_{kj} - \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} - 2\alpha e_{ij} p_{ik}$$

The final Algorithm

1. Initialize the two P and Q matrices with random values
2. Calculate the **total** error of **observed** data

$$E = \sum_{u_i, d_j, r_{i,j}} e_{ij}^2 = \sum_{u_i, d_j, r_{i,j}} \left(r_{ij} - \sum_{K=1}^k p_{iK} q_{Kj} \right)^2$$

3. If reached the stop criterion
 - Stop
4. Otherwise
 - Update P and Q values and return to step 2

$$p'_{ik} = p_{ik} - \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} - 2\alpha e_{ij} q_{kj}$$
$$q'_{kj} = q_{kj} - \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} - 2\alpha e_{ij} p_{ik}$$

Note

- We should give the number of latent features (k) as an input to the algorithm.
 - Usually found by cross-validation over evaluation dataset

The final Algorithm - Problems

- Can over-fit
 - Not generalizable enough
- Does not consider user and item biases
 - Alice is picky: always rates lower than other users
 - The Godfather is a good movie: average rating is higher than other movies
 - Most of the ratings are 3 or higher in our data

The final Algorithm - Solutions

- Can over-fit
 - Not generalizable enough
 - Solution: **Regularize** the error
- Does not consider user and item biases
 - Alice is picky: always rates lower than other users
 - The Godfather is a good movie: average rating is higher than other movies
 - Most of the ratings are 3 or higher in our data
 - Solution: consider **bias** to estimated rating

Regularization

- To avoid overfitting
- Add some factors to the error to control the values of P and Q
 - Only allow some important factors to have value bigger than zero

$$e_{ij}^2 = \left(r_{ij} - \sum_{K=1}^k p_{iK} q_{Kj} \right)^2 + \beta \sum_{K=1}^k (\|P\|^2 + \|Q\|^2)$$

Adding Biases

- Include user and item biases in estimated rating

deviation of item i


overall average rating

deviation of user u

$$\hat{r}_{ij} = \mu + b_i + b_u + p_i^T q_j$$

$$e_{ij}^2 = (r_{ij} - \mu - b_i - b_u - p_i^T q_j)^2 + \beta(\|p_u\|^2 + \|q_i\|^2 + b_i^2 - b_u^2)$$

Outline

- Overview of Collaborative Filtering (CF)
- Advanced Approaches to CF
 - Matrix Factorization Techniques
 -  – Probabilistic Approaches
- Cross-domain Recommenders

Probabilistic Methods - LDA

- Origin from information retrieval
- Each **document** consists of multiple **topics** and each **word** in the document represents one of those topics

LDA-An Example

Topics

gene 0.04
dna 0.02
genetic 0.01
...

life 0.02
evolve 0.01
organism 0.01
...

brain 0.04
neuron 0.02
nerve 0.01
...

data 0.02
number 0.02
computer 0.01
...

Documents

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

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

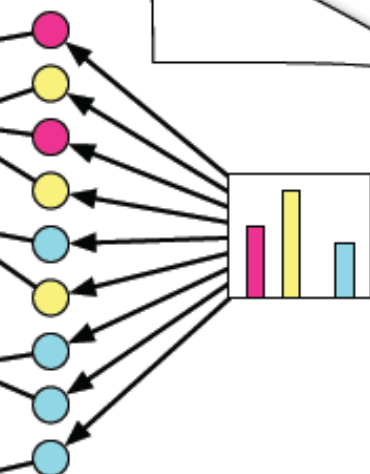
"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson at Uppsala University in Sweden. "We arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes 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



Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

Topic proportions and assignments



LDA-An Example

- Suppose we have these sentences
 - I like to eat broccoli and bananas.
 - I ate a banana and spinach smoothie for breakfast.
 - Chinchillas and kittens are cute.
 - My sister adopted a kitten yesterday.
 - Look at this cute hamster munching on a piece of broccoli.
- We want to discover 2 topics for them.

LDA-An Example

1. I like to eat broccoli and bananas.
 2. I ate a banana and spinach smoothie for breakfast.
 3. Chinchillas and kittens are cute.
 4. My sister adopted a kitten yesterday.
 5. Look at this cute hamster munching on a piece of broccoli.
- **Sentences 1 and 2:** 100% Topic A
 - **Sentences 3 and 4:** 100% Topic B
 - **Sentence 5:** 60% Topic A, 40% Topic B
 - **Topic A:** 30% broccoli, 15% bananas, 10% breakfast, 10% munching, ... (at which point, you could interpret topic A to be about food)
 - **Topic B:** 20% chinchillas, 20% kittens, 20% cute, 15% hamster, ... (at which point, you could interpret topic B to be about cute animals)

How does LDA do that?

- Consider a machine that represents **documents** as **mixtures of topics** that spit out **words** with certain probabilities

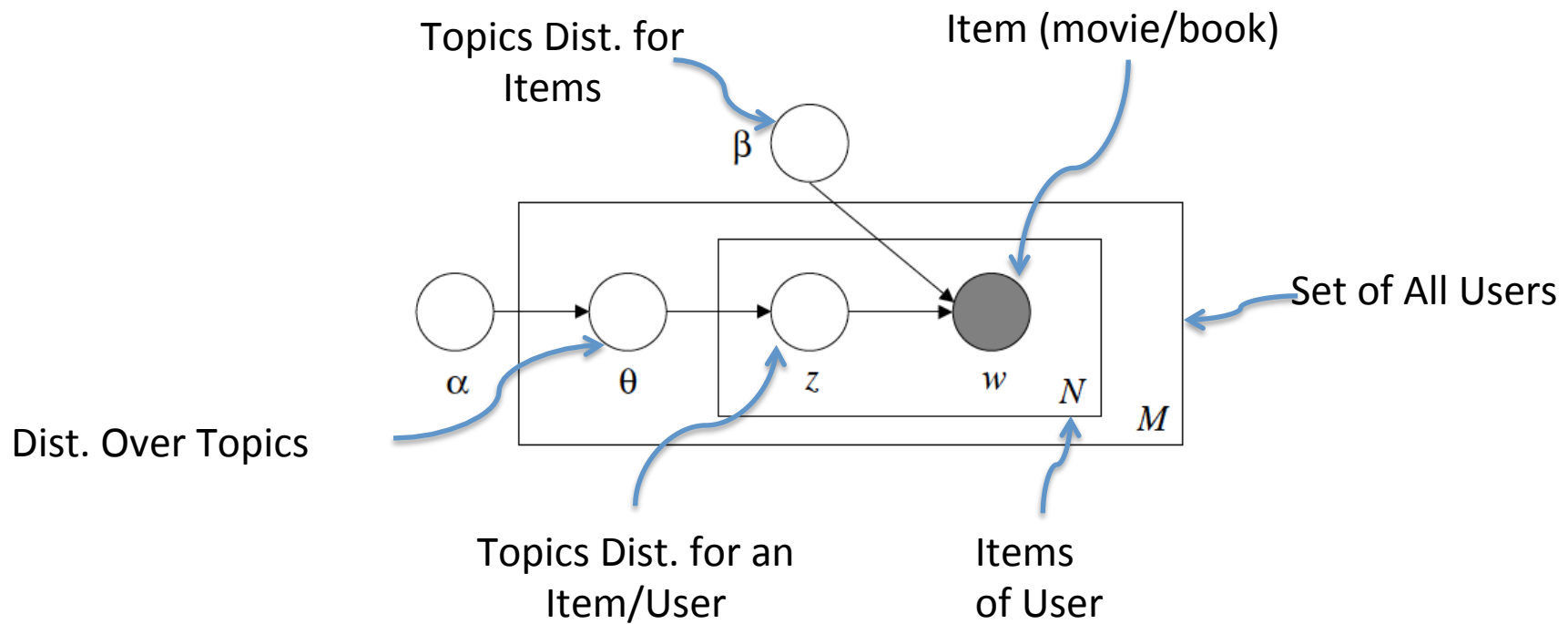
How does LDA do that? (Cont'd)

- Decide on the number of **words** N the **document** will have (say, according to a Poisson distribution).
- Choose a **topic** mixture for the document (according to a Dirichlet distribution over a fixed set of K topics).
 - E.g. choose the document to consist of 1/3 food and 2/3 cute animals.
- Generate each **word** w_i in the **document** by:
 - First picking a **topic** (according to the distribution that you sampled above; e.g. pick the food topic with 1/3 probability).
 - Using the **topic** to generate the **word** itself (according to the topic's multinomial distribution). E.g. generate the word “broccoli” with 30% probability

How is LDA related to recommender system?

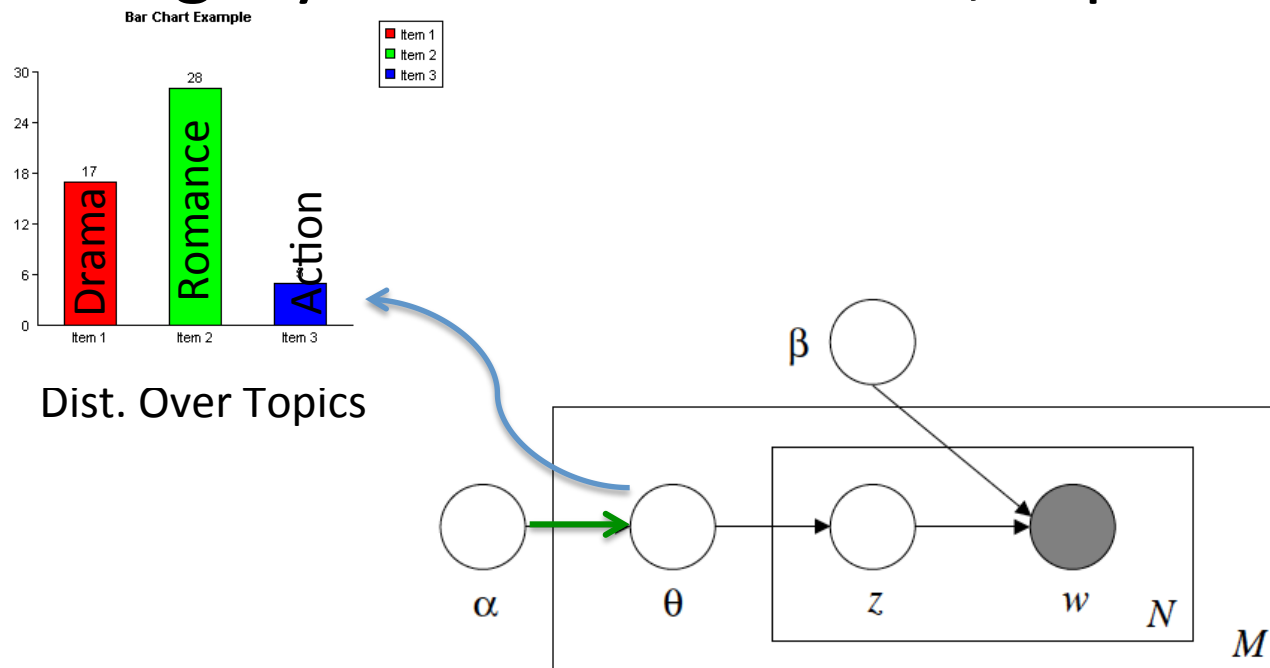
- CF Analogy: Each **user** is interested in multiple **topics** and each **item** in user profile represents one of those topics
 - **Alice** is interested in **crime** and **drama**
 - “**The God Father**” is 50% crime and 50% drama
 - “**Casablanca**” is 40% drama, 30% war, and 30% romance
 - **Alice** likes “**The God Father**”, and “**Casablanca**”

Graphical Representation of LDA



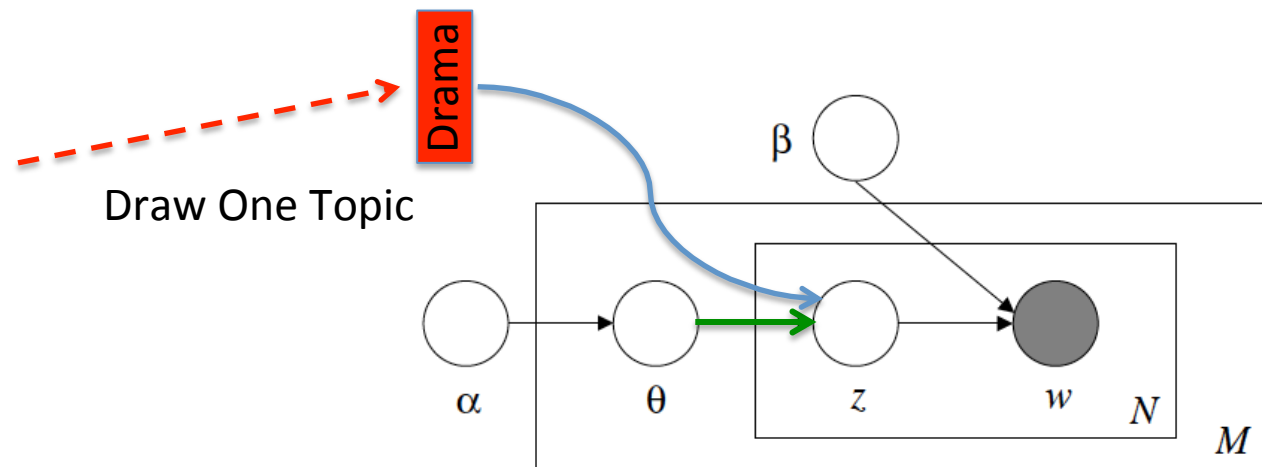
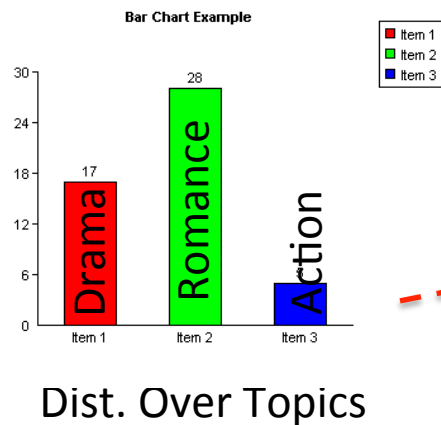
Graphical Representation of LDA

- Draw topic/category proportions: *Dirichlet*(α)
- Category: Genre for movies, topic for books, ...



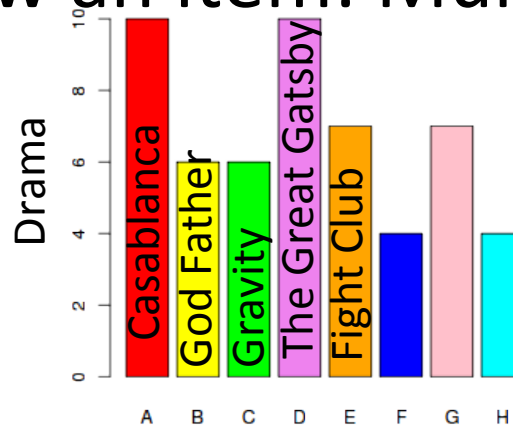
Graphical Representation of LDA

- For each user:
 - Draw a topic/category z for user from its own ϑ over topics: $Mult(\vartheta_j)$

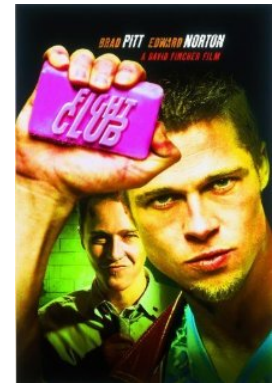


Graphical Representation of LDA

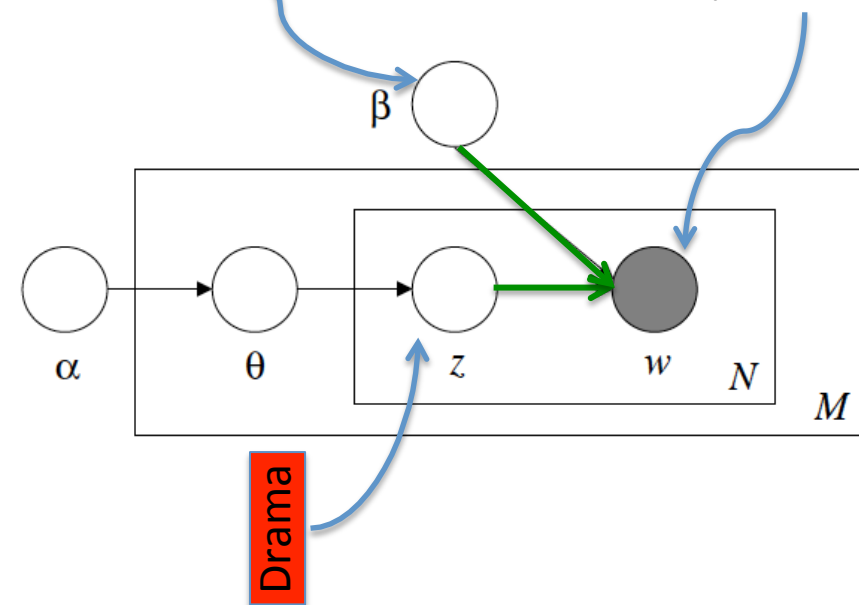
- Draw an item: $\text{Mult}(z_{jn}, \beta_{zjn})$



Topics Dist. for Items



Item (movie/book)



Probabilistic Methods - LDA

$$p(w|\mathbf{w}_{\text{obs}}) = \int \sum_z p(w|z)p(z|\theta)p(\theta|\mathbf{w}_{\text{obs}})d\theta,$$

How to Learn Parameters?

- Collapsed Gibbs sampling
- Go through each document, and randomly assign each word in the document to one of the K topics.
- To improve on topic assignments, for each document d
 - Go through each word w in d (assume that all topic assignments except for the current word are correct)
 - for each topic t , compute:
 - 1) $p(\text{topic } t \mid \text{document } d)$ and
 - 2) $p(\text{word } w \mid \text{topic } t)$.
 - Reassign w a new topic (topic t with probability $p(\text{topic } t \mid \text{document } d) * p(\text{word } w \mid \text{topic } t)$)
 - Repeat the previous step till you reach a stable assignment

Note

- We should give the number of topics as an input to the algorithm.
 - Usually found by cross-validation over evaluation dataset

Outline

- Overview of Collaborative Filtering (CF)
- Advanced Approaches to CF
 - Matrix Factorization Techniques
 - Probabilistic Approaches

 Cross-domain Recommenders

Cross-Domain Recommender Systems

- Current research studies on recommender systems: single-domain recommendations
- Commercial systems such as Amazon, Google Play: more than one domain of items: books, movies, perfumes, etc.
- Cross-Domain Recommendations: Item recommendations in a multi-domain environment

Cross-Domain Recommender Systems

- How can cross-domain recommenders help?
 - provide the relationship between two sets of items from various domains.
 - provide extra information about the new users of a target domain (targeting the cold-start problem)
 - Increase serendipity and novelty in results

Cross-Domain Recommender Systems-

Domain

- types of items (e.g. movies vs. books) or groups of similar items with common characteristics (e.g. movies vs. TV shows)
- system domains: different datasets upon which the recommender systems are built
- data domains: different representations of user preferences, which can be implicit (e.g. clicks, purchases) or explicit (e.g. ratings)
- temporal domains: subsets in which a dataset is split based on timestamps

Cross-Domain Recommender Systems- Task

- U_A, U_B : sets of users in domains A and B
- J_A, J_B : sets of items with “characteristics” (user preferences and item attributes) in the domains A and B
- Exploit knowledge about users and items in the source domain A for **improving the quality of the recommendations** for items in the target domain B
- Making **joint recommendations** for items belonging to different domains, i.e., suggesting items in $J_A \cup J_B$ to users in $U_A \cup U_B$.

Cross-Domain Recommender Systems-

Types of Explicit Relations

- No overlap

$$U_A \cap U_B = \emptyset \wedge J_A \cap J_B = \emptyset$$

- User overlap

$$U_A \cap U_B \neq \emptyset$$

- Item overlap

$$J_A \cap J_B \neq \emptyset$$

- Full overlap

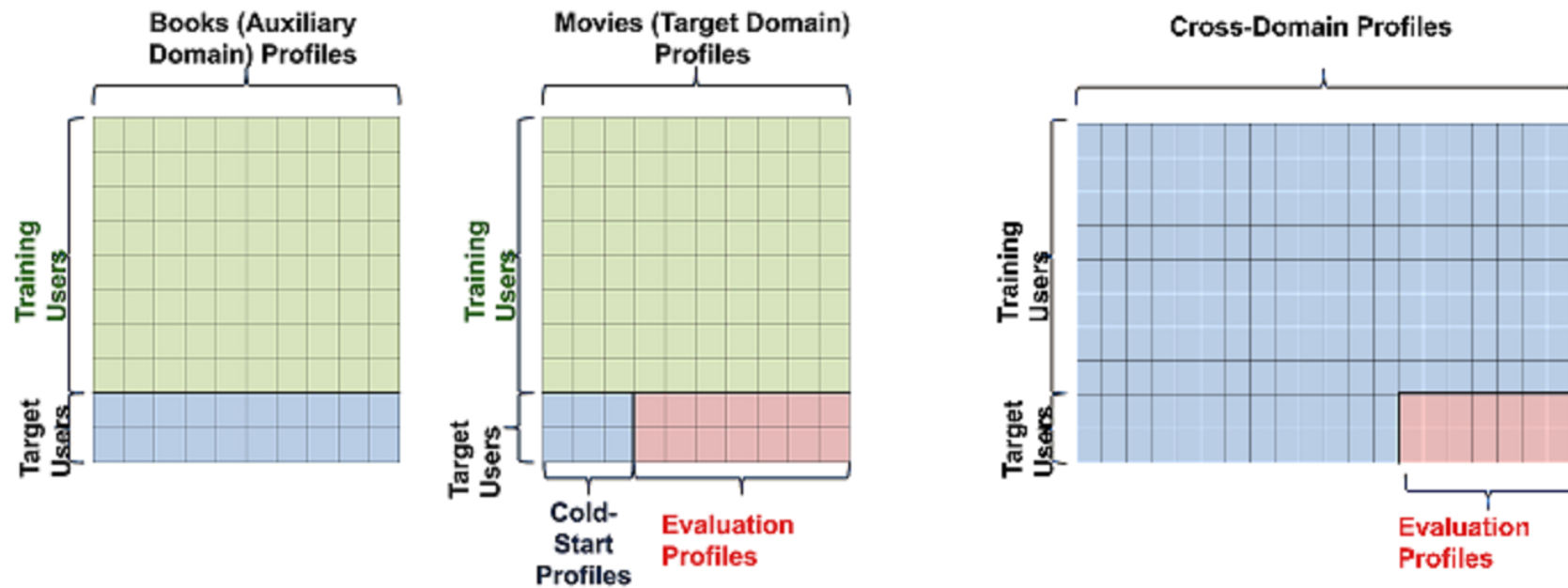
$$U_A \cap U_B \neq \emptyset \wedge J_A \cap J_B \neq \emptyset$$

Cross-Domain Recommender Systems-

Types of Explicit Relations

- Content-based relations
 - Both item and user content is described as a set of features $F = \{F_1, F_2, \dots, F_n\}$ (content, tags, etc)
 - an overlap between the domains A and B occurs when
$$F_A \cap F_B \neq \emptyset$$
- Collaborative Filtering-based relations
 - Building joint matrices to achieve relations between items and users
- Other relations
 - Other features such as time, user mood, etc.

Cross-Domain Recommender Systems- Collaborative Filtering-based relations



Cross-Domain Recommender Systems

- techniques

- integrating and exploiting explicit user preferences distributed in various systems
- recording user behavior and actions aiming to learn user preferences, and use them for generating joint recommendations on multiple domains
- combining recommendations from different domains to build a single system

Cross-Domain Recommender Systems

- techniques

- adaptive models: exploit information from a source domain to make recommendations in a target domain
- collective models: are built with data from several domains and potentially can make joint recommendations for such domains

Cross-Domain Recommender Systems

- evaluation

- Methods
 - varying the sparsity level in the data
 - varying the degree of overlap between domains
- Measures
 - Rating-based: MAE and RMSE
 - Ranking-based: Precision, Mean Reciprocal Rank, F-measure, etc.
 - novelty and diversity metrics

- Do you want to have a project in cross-domain recommender systems? Email me:
sahebi@cs.pitt.edu

references

- SVD++: Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '08). ACM, New York, NY, USA, 426-434. DOI=10.1145/1401890.1401944 <http://doi.acm.org/10.1145/1401890.1401944>
- LDA: David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993-1022.
- Cross-Domain Recommenders: I. F. Tobias, I. Cantador, M. Kaminskas, F. Ricci. 2011. Cross-domain recommender systems: A survey of the State of the Art. Proceedings of the 2nd Spanish Conference on Information Retrieval ([CERI 2012](#)). Valencia, Spain, June 2012
- <http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/>
- <http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/>

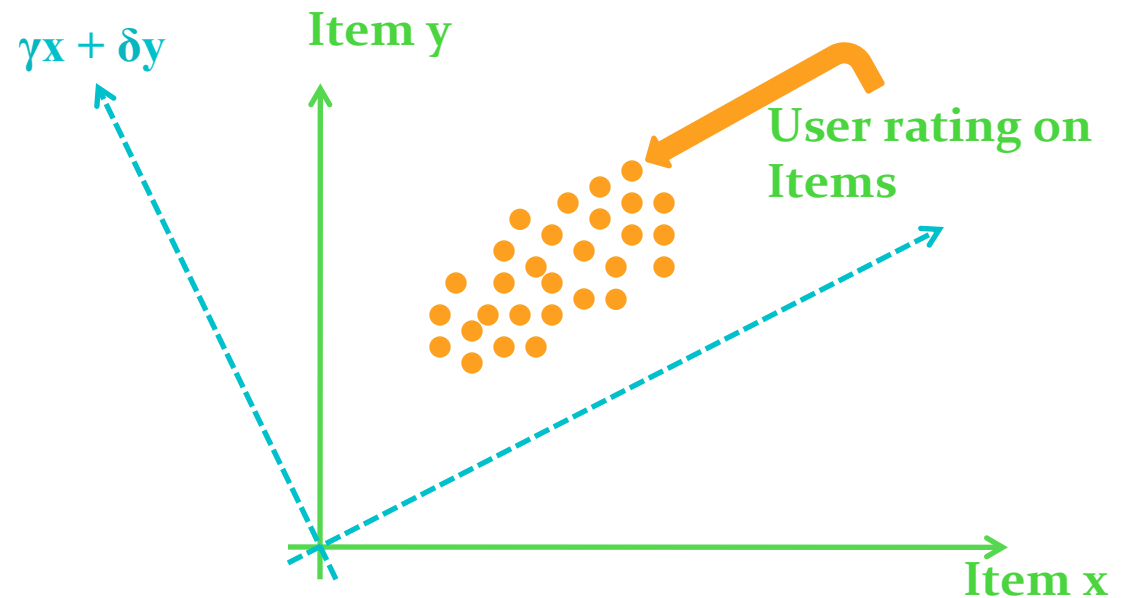
Collaborative Filtering (CF) Review

- $P_{a,j}$: Predicted rating of user a on item j
- V_a : Average rating of user a
- $V_{i,j}$: Rating of user i on item j
- $W(a,i)$: Similarity between users a and i

$$P_{a,j} = \bar{v}_a + \alpha \sum_{i=1}^n w(a,i)(v_{i,j} - \bar{v}_i)$$

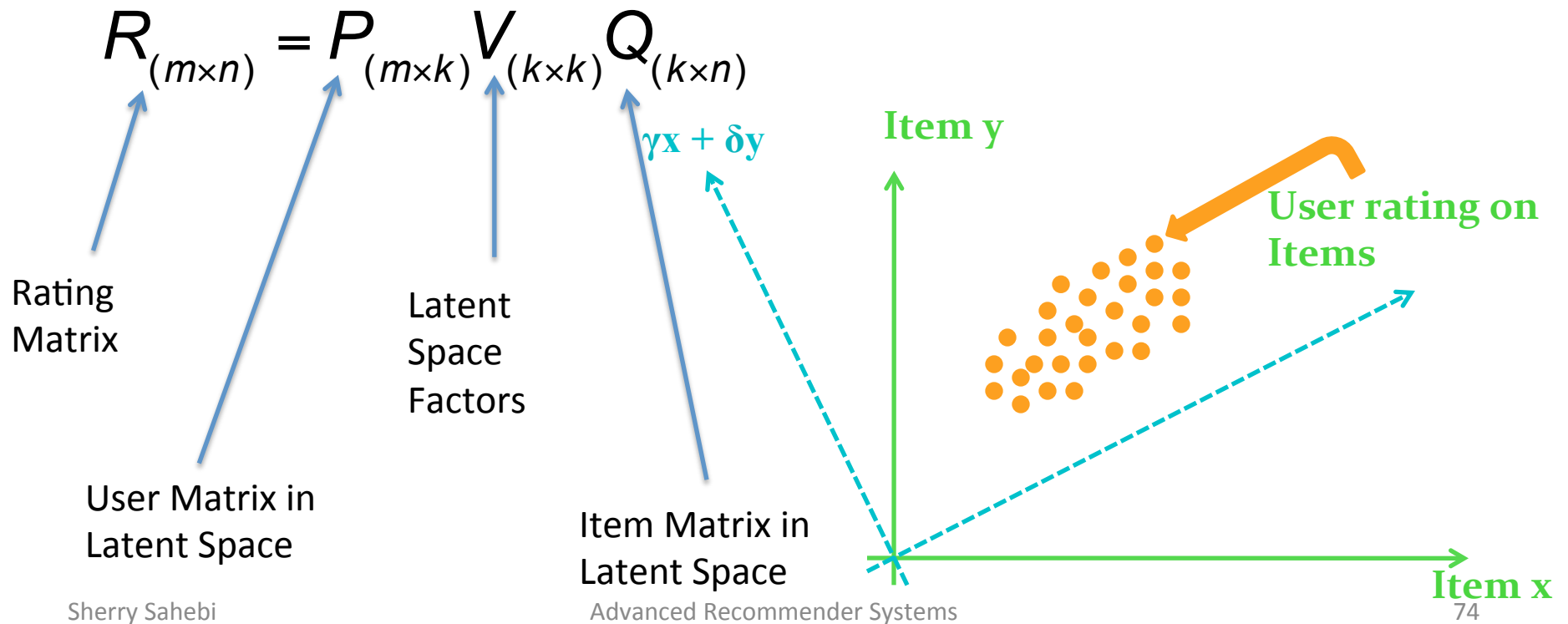
Dimensionality Reduction Algorithms - SVD

- Map item space to a smaller number of underlying “dimensions”
- Building a new orthogonal space based on the old space dimensions



Dimensionality Reduction Algorithms - SVD

- Groups items into similar sets
- Decomposes the rating matrix into 3 matrices:



Dimensionality Reduction Algorithms -

R

	M1	M2	M3	M4	M5
U1	?	?	?	5	?
U2	?	?	?	?	4
U3	3	?	?	?	?
U4	4	?	5	1	?
U5	?	4	4	1	?

SVD

P

	L1	L2	L3
U1	-0.1	-0.9	0.3
U2	0	0	0
U3	-0.1	0.2	0.4
U4	-0.7	0.2	0.3
U5	-0.5	-0.5	-0.7

Q

	M1	M2	M3	M4	M5
L1	-0.4	0.3	0.6	0	0.4
L2	-0.2	-0.1	-0.6	0	0.6
L3	-0.7	0.1	-0.2	0	-0.5

Dimensionality Reduction Algorithms - SVD

- Gives user-factors vector (p_u) and item-factors vector (q_i)
 - b_u and b_i : the observed deviations of user u and item i , respectively, from the average
 - μ : Overall average rating

$$r_{ui} = \mu + b_u + b_i + p_u^T q_i$$

