

# Recommendation Systems

## Part II

Devavrat Shah

Massachusetts Institute of Technology

# Recall — Part I

---

Introduction, simple methods

Module 1: background

Recommendation systems: why and what?

Example datasets

Module 2: problem statement

Recommendation systems: a prediction problem

Model: estimating time varying tensor with side information

Module 3: simple solutions

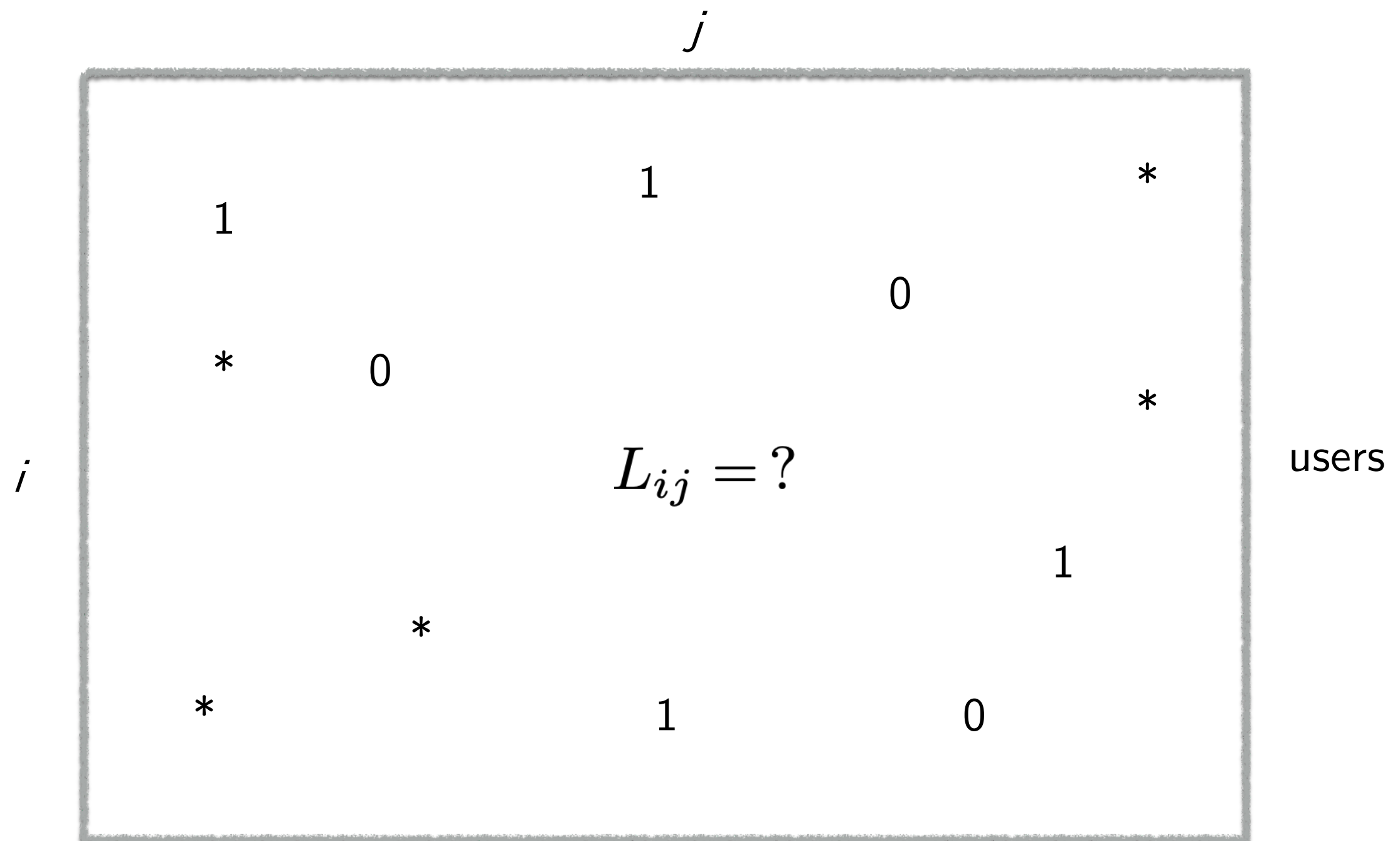
Solution I: averaging

Solution II: content-based

## Recall: problem statement

---

We will start with simple problem statement: complete the matrix



## Recommendation: Problem statement

We will start with simple problem statement: complete the matrix

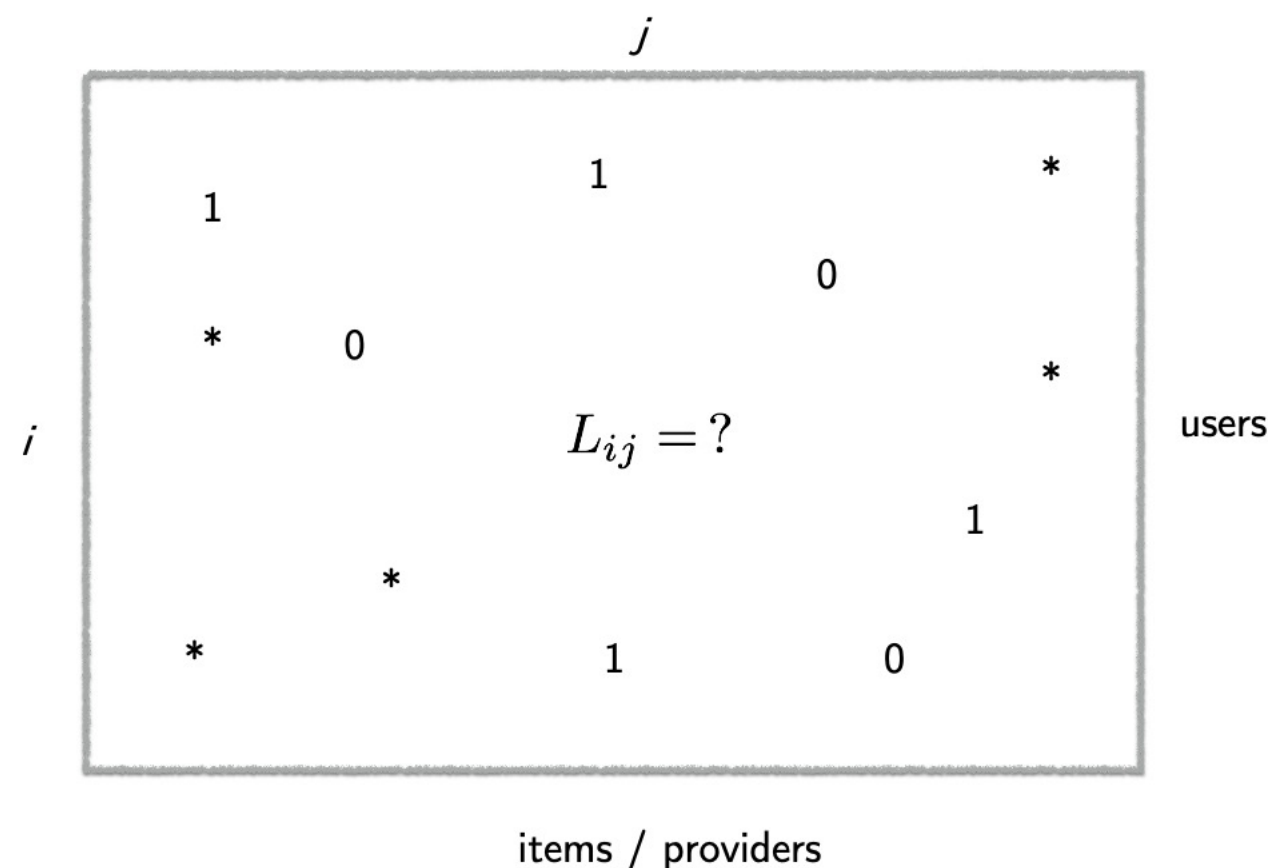
Observations:  $Y_{ij}$  over  $i$  in users,  $j$  in items

If  $(i, j)$  is observed

$$E[Y_{ij}] = L_{ij}$$

If  $(i, j)$  is *not* observed

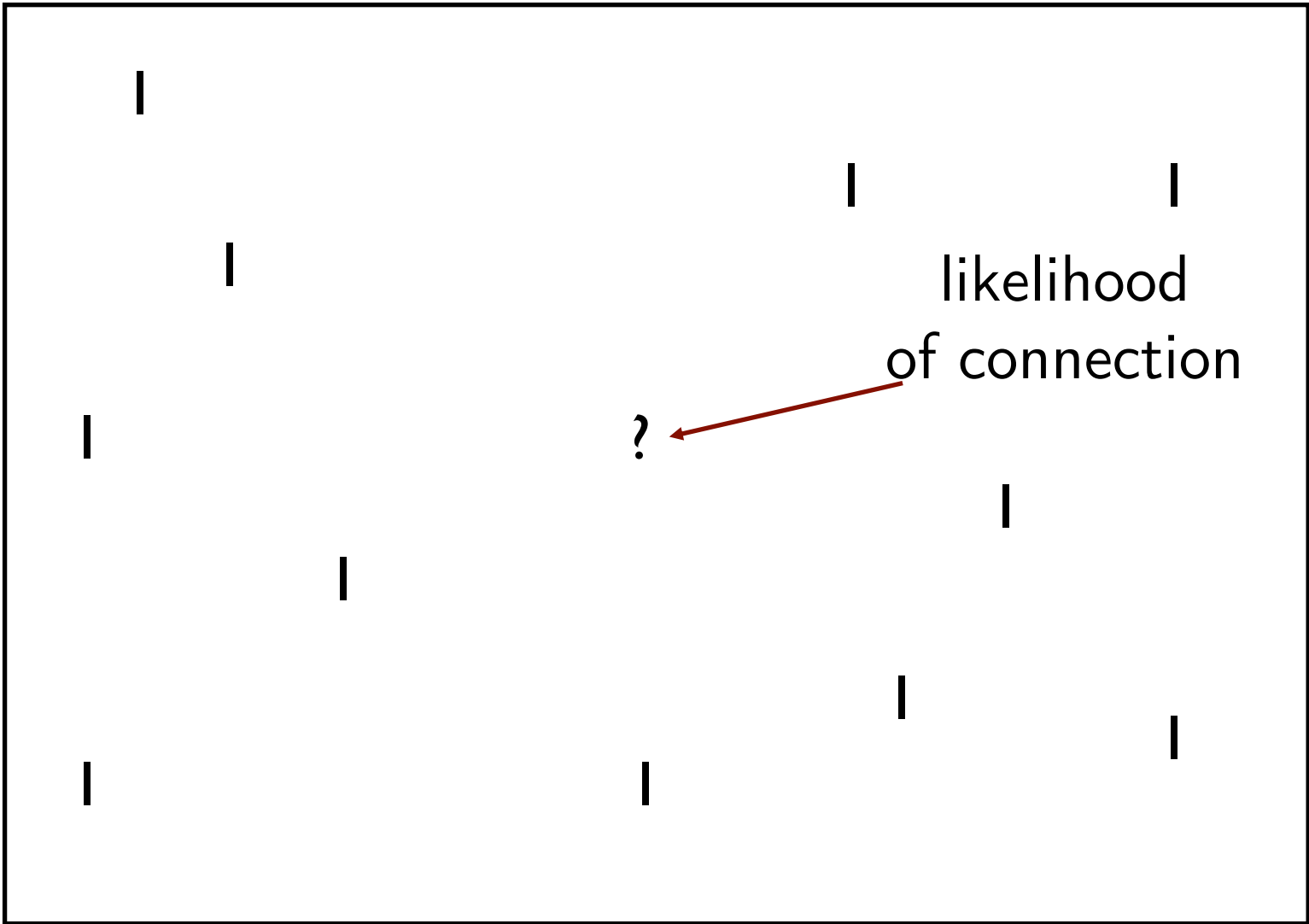
$$Y_{ij} = \star \text{ or } ?$$



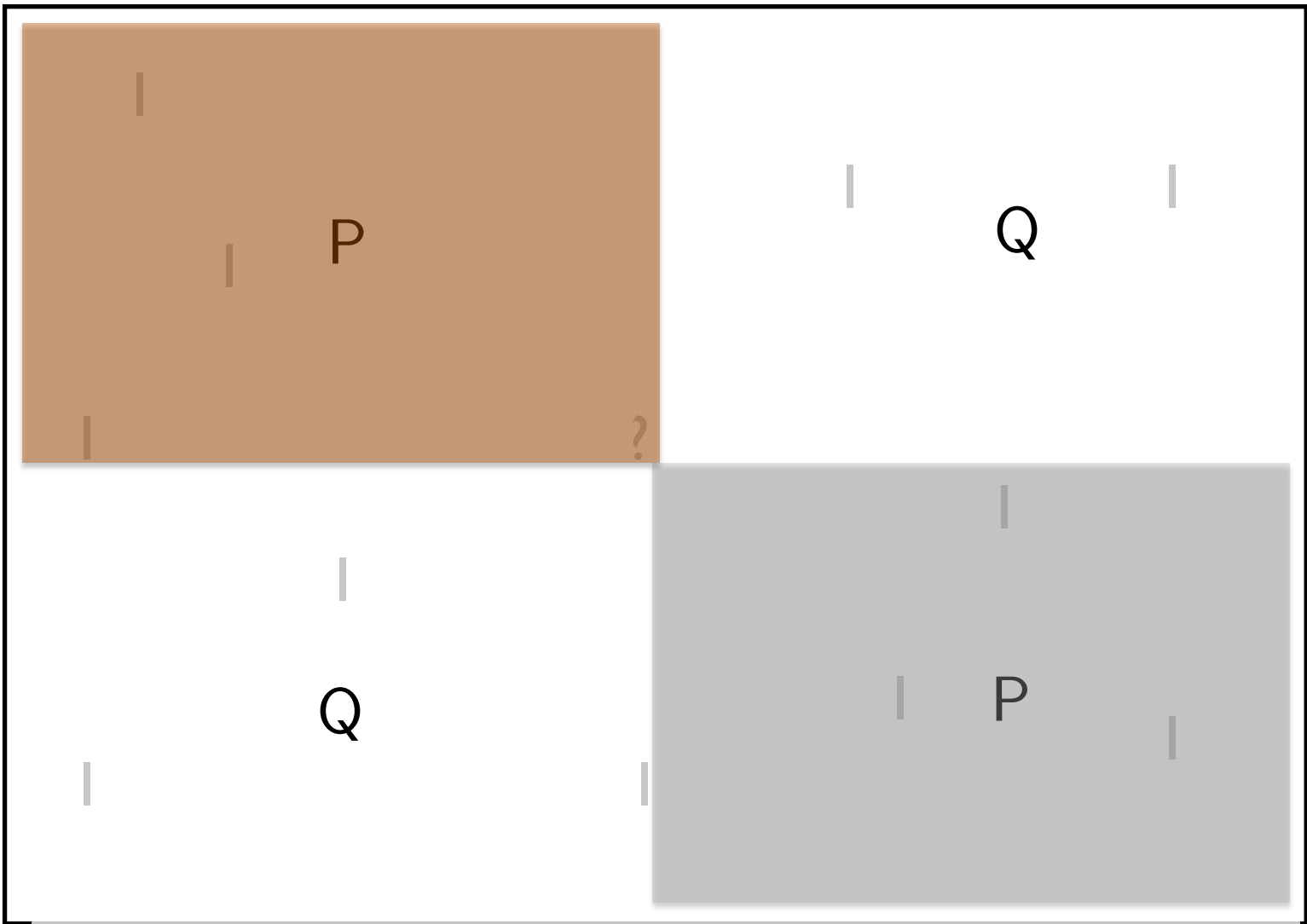
Goal: produce estimation  $\hat{L}_{ij}$  for all  $i, j$

so that  $\hat{L}_{ij} \approx L_{ij}$  for all  $i, j$

# Social Networks



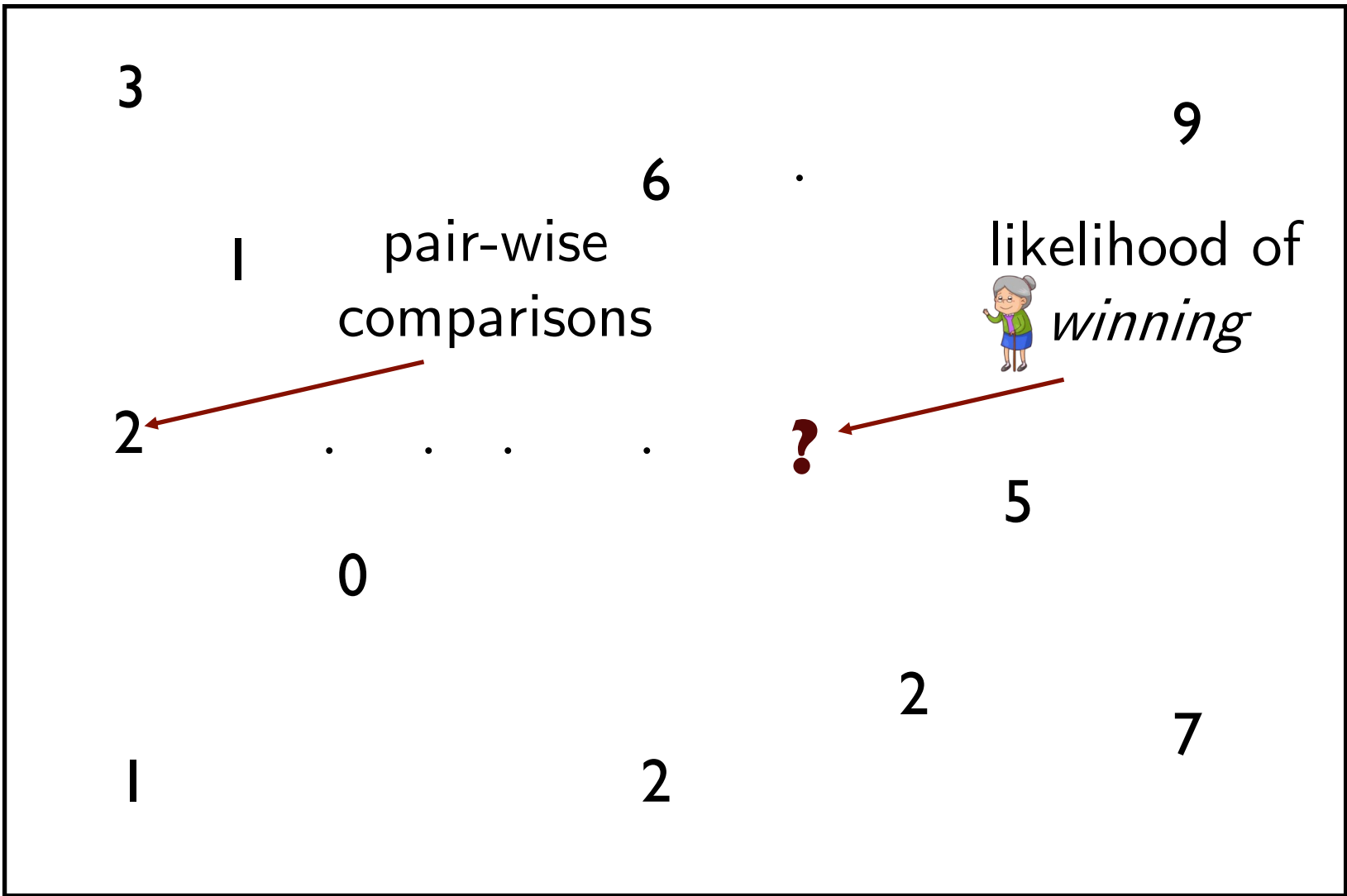
# Community Detection



$P > Q$

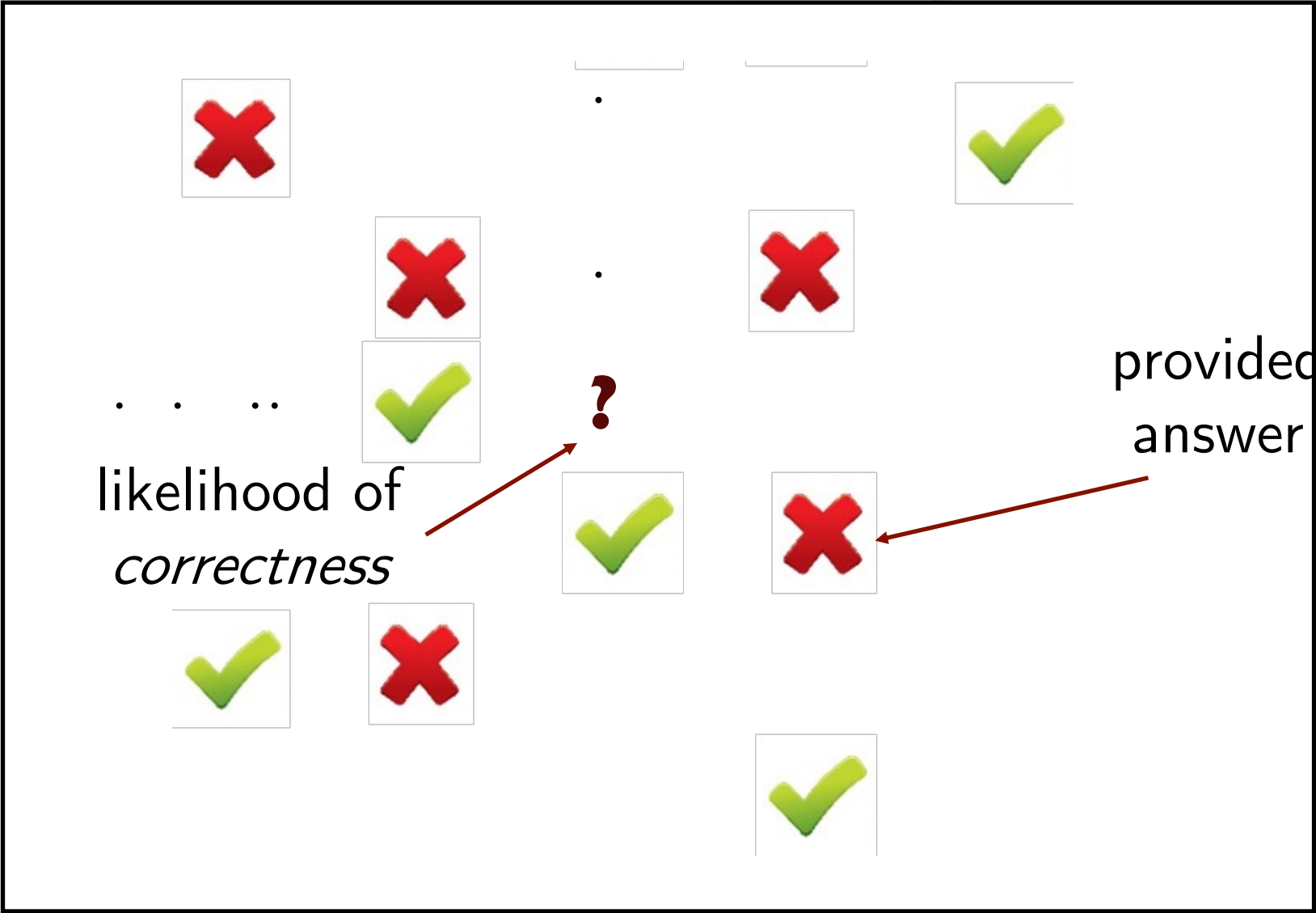


# Ranking Players, Teams



# Crowd-sourcing

*Is the Website Suitable for Children?*





# Recommendations

---

May be *data* can help. What data?

Example: Yelp data

Businesses: attributes (locations, category), hours

Users: attributes, friends

Reviews: rating, description, time

Check-ins: time

Tip

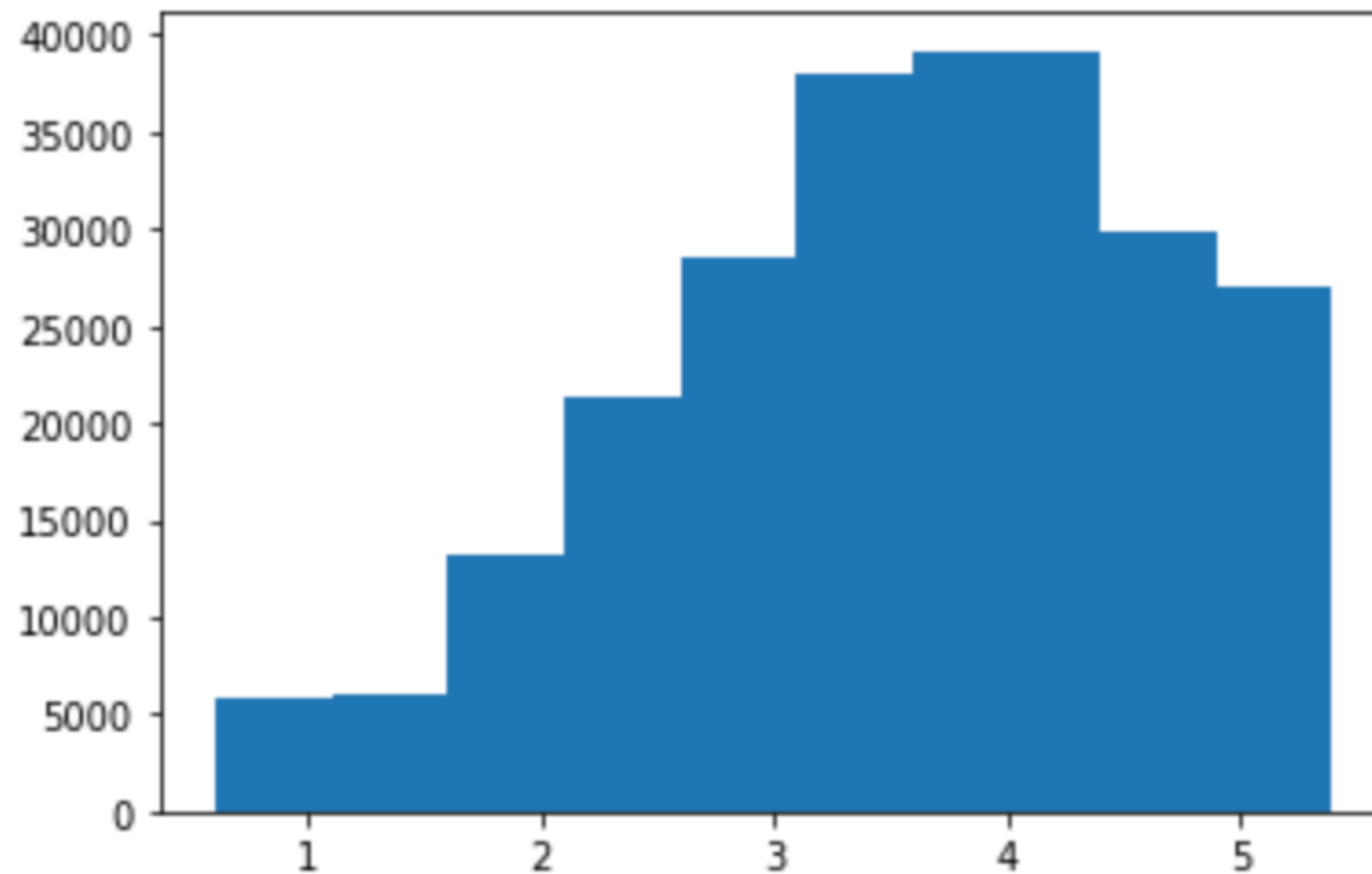
**Exercise:** go to link below, explore data and reproduce statistics reported

<https://www.kaggle.com/jagangupta/what-s-in-a-review-yelp-ratings-eda>

## Explore: Yelp Data

---

(Aggregate) Star Rating Distribution



Data URL:

<https://www.kaggle.com/jagangupta/what-s-in-a-review-yelp-ratings-eda>

## Explore: Yelp Data

---

What fraction of reviews are known?

Users =  $\sim 2\text{M}$

Businesses =  $\sim 200\text{k}$

Total possible reviews =  $\sim 2\text{M} \times \sim 200\text{k} = \sim 0.4\text{T}$

Known reviews =  $\sim 8\text{M}$

Fraction known =  $\sim 8\text{M} / 0.4\text{T} = 2 \times 10^{-5}$

i.e. 2 in every 100k reviews is known, rest are *unknown*

Finding these *unknown* reviews is the primary goal of Rec Sys

Data URL:

<https://www.kaggle.com/jagangupta/what-s-in-a-review-yelp-ratings-eda>

# Explore: MovieLens Data

---

## MovieLens Data

Movies: attributes including title, release date, genre, actors, director

Users: demographics including age, gender, occupation, zip code

Reviews: ratings, timestamp

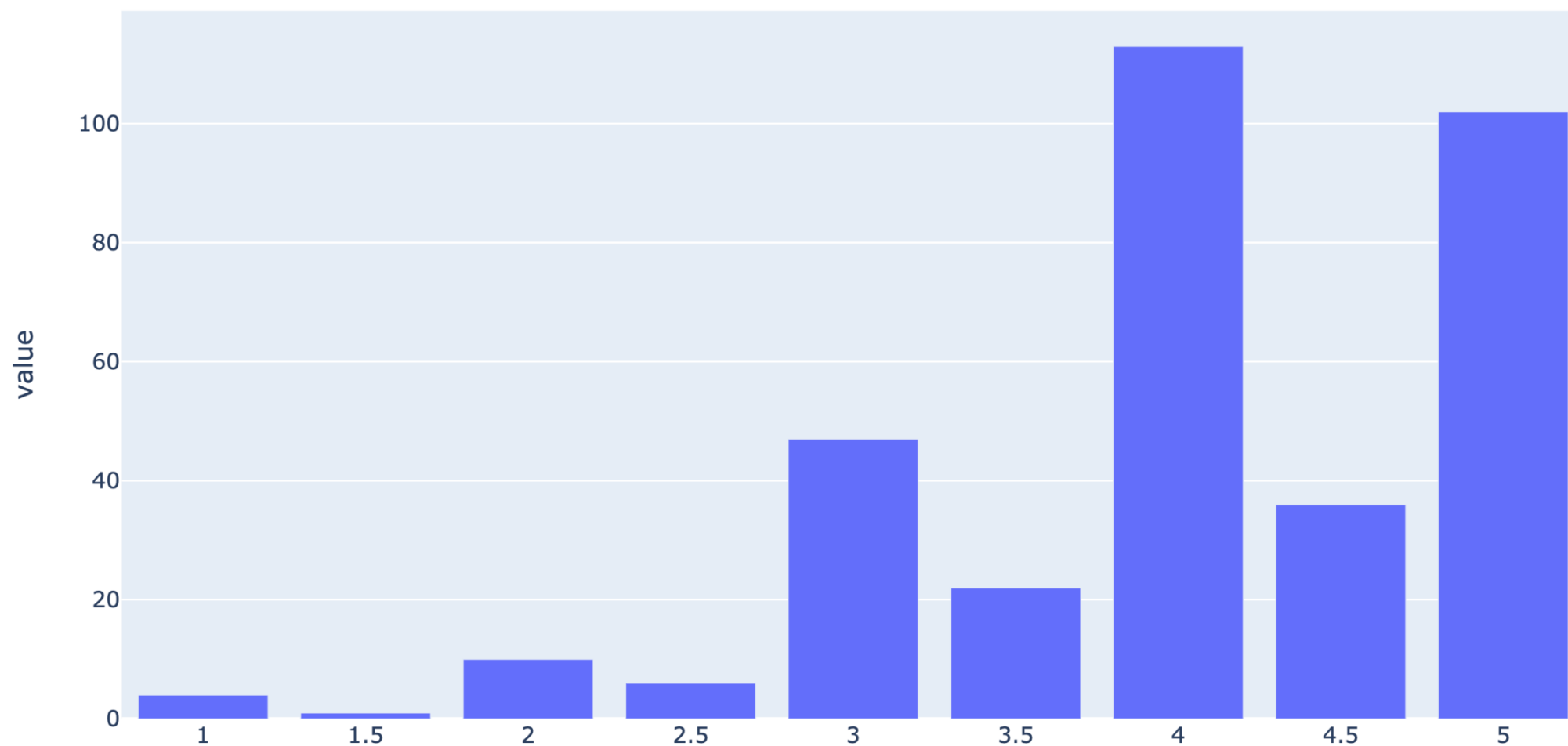
**Exercise:** go to link below, explore data and reproduce statistics reported

<https://grouplens.org/datasets/movielens/100k>

# Explore: MovieLens Data

---

Distribution of top-rated movie (356)



Data URL:

<https://grouplens.org/datasets/movielens/100k>

## Explore: MovieLens Data

---

What fraction of reviews are known?

Users =  $\sim 1.7k$

Movies =  $\sim 1k$

Total possible reviews =  $\sim 1.7k \times \sim 1k = \sim 1.7M$

Known reviews =  $\sim 100k$

Fraction known =  $\sim 100k / 1.7M = \sim 0.058$  or  $\sim 6\%$

i.e. 6 in every 100 reviews is known, rest are *unknown*

Finding these *unknown* reviews is the primary goal of Rec Sys

Data URL:

<https://www.kaggle.com/jagangupta/what-s-in-a-review-yelp-ratings-eda>



# Recall — Part I

---

Introduction, simple methods

Module 1: background

Recommendation systems: why and what?

Example datasets

Module 2: problem statement

Recommendation systems: a prediction problem

Model: estimating time varying tensor with side information

Module 3: simple solutions

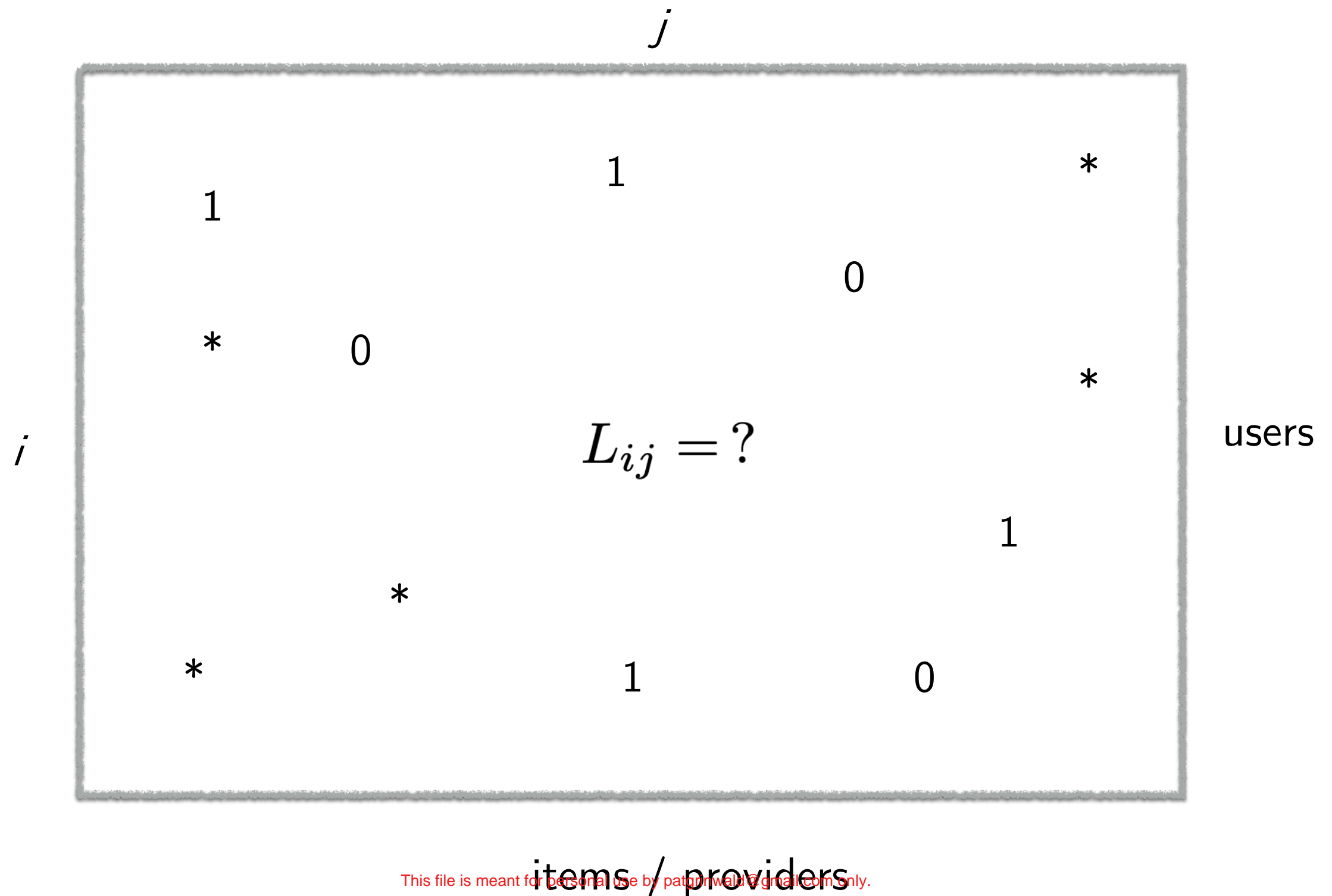
Solution I: averaging

Solution II: content-based

## Recall: problem statement

---

We will start with simple problem statement: complete the matrix



# Recommendation: Problem statement

Complete the matrix

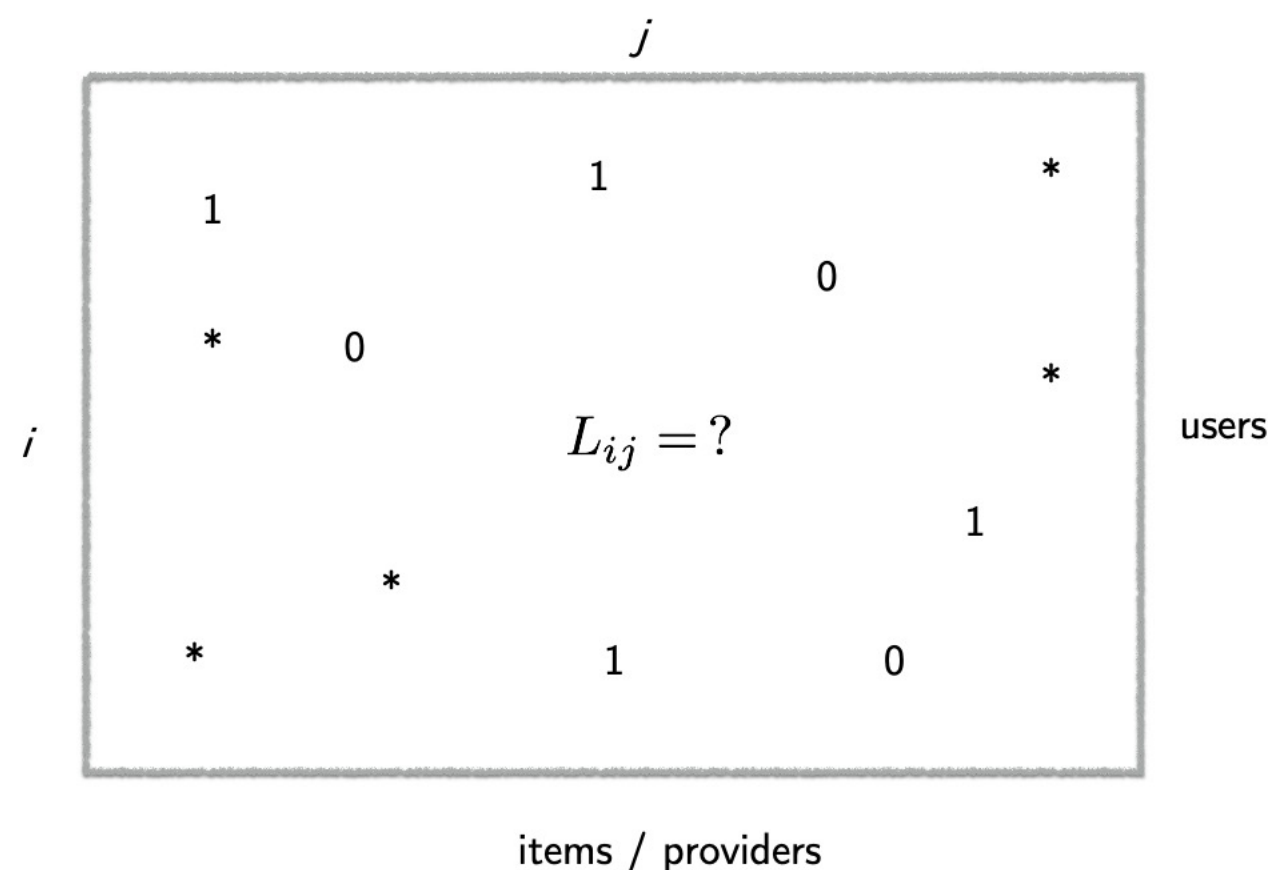
Observations:  $Y_{ij}$  over  $i$  in users,  $j$  in items

If  $(i, j)$  is observed

$$E[Y_{ij}] = L_{ij}$$

If  $(i, j)$  is *not* observed

$$Y_{ij} = \star \text{ or } ?$$



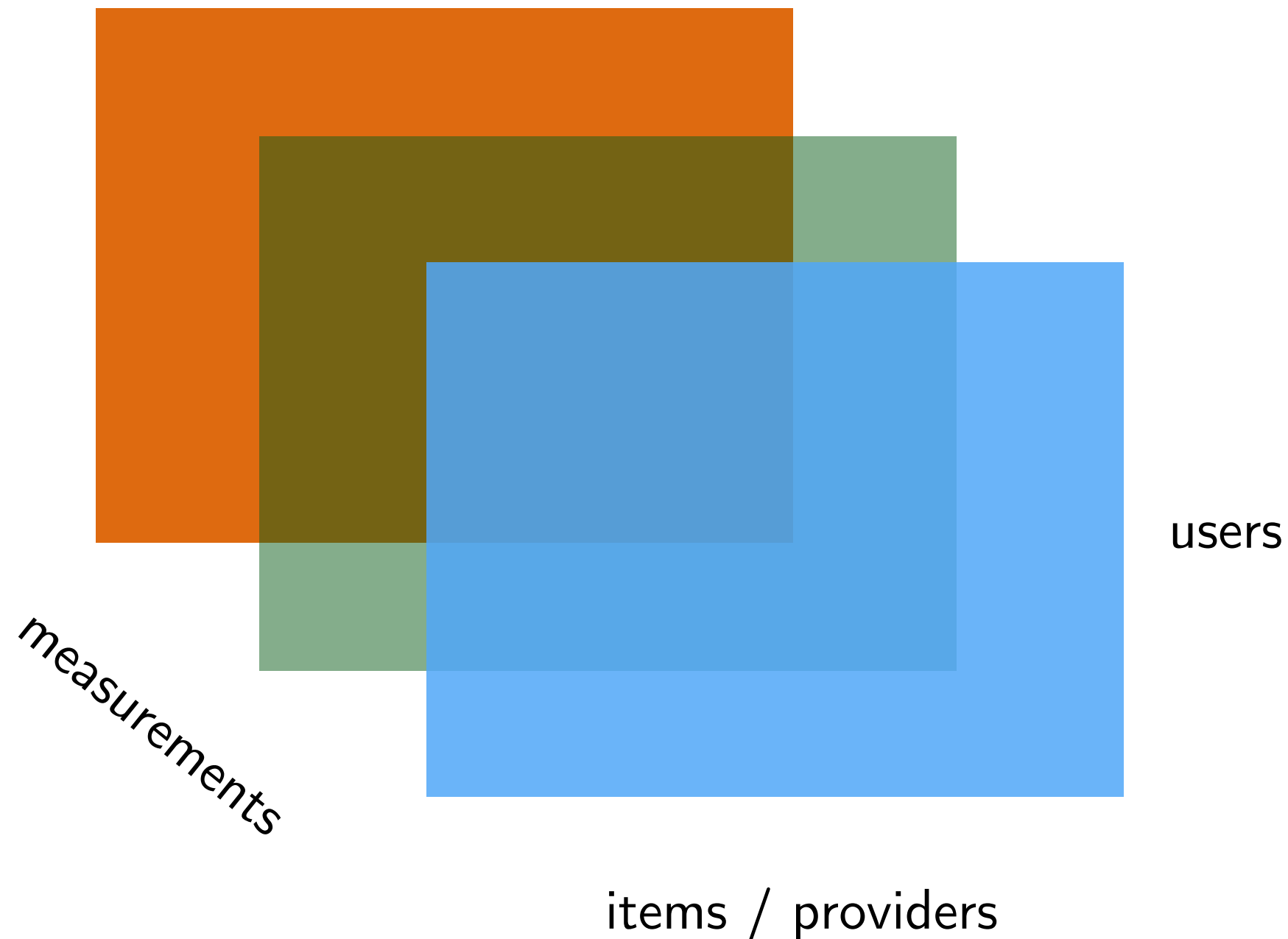
Goal: produce estimation  $\hat{L}_{ij}$  for all  $i, j$

so that  $\hat{L}_{ij} \approx L_{ij}$  for all  $i, j$

# Recommendation: Problem statement

---

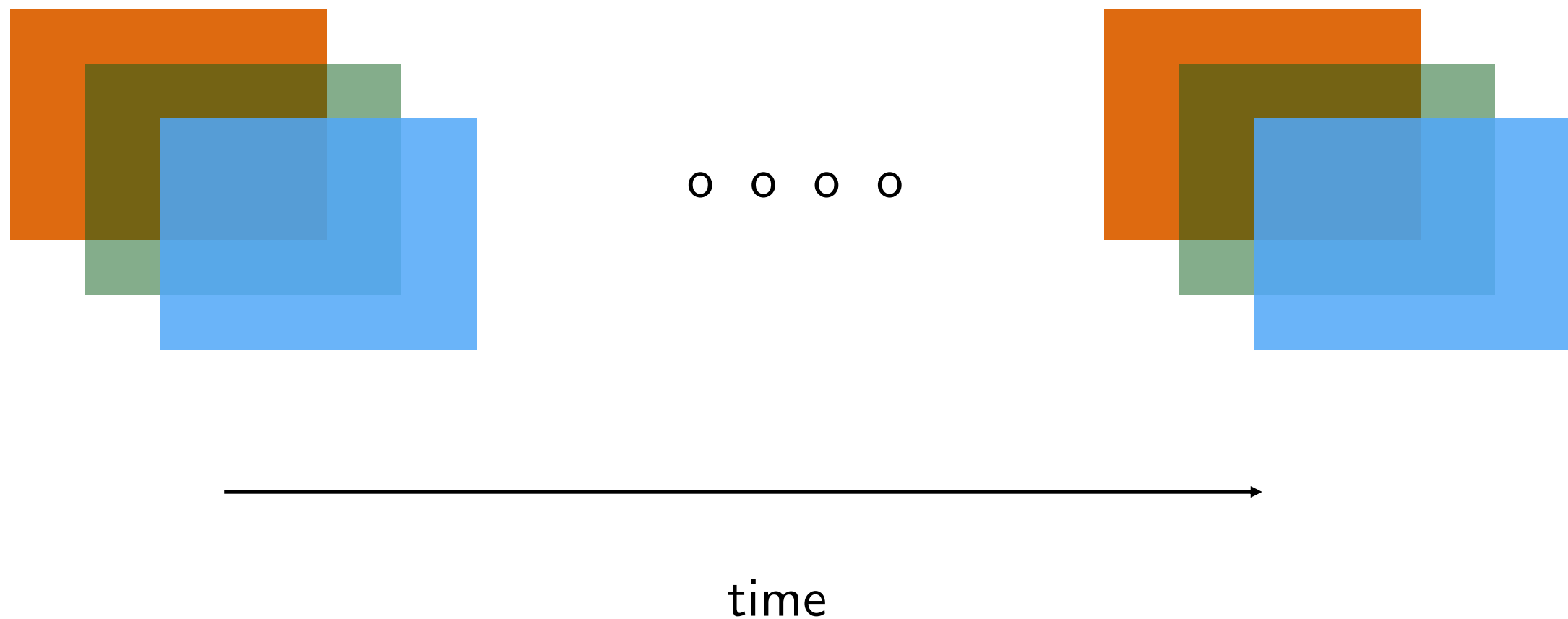
Prediction problem: complete the tensor



# Recommendation: Problem statement

---

Prediction problem: complete the time varying tensor



Introduction, simple methods

Module 1: background

Recommendation systems: why and what?

Example datasets

Module 2: problem statement

Recommendation systems: a prediction problem

Model: estimating time varying tensor with side information

Module 3: simple solutions

Solution I: averaging

Solution II: content-based



## Solution 1: Averaging

---

What if, instead we assume

All items or providers are identical

Then estimate: row average (+ correction for number of observations)

How to put these two simple estimators together?

$$2L_{ij} = L_{i.} + \frac{1}{\sqrt{n_{i.}}} + L_{.j} + \frac{1}{\sqrt{n_{.j}}}$$

where  $L_{i.}$  is average of observed entries in row  $i$

$n_{i.}$  is number of observed entries in row  $i$

$L_{.j}$  is average of observed entries in column  $j$

$n_{.j}$  is number of observed entries in column  $j$

## Solution 2: Content Based

---

This is *supervised learning* problem we have already seen

Labeled data:

each observed entry in matrix  $(i, j)$  corresponds to labeled data

$$((x_i, y_j); L_{ij})$$

Learning problem:

learn the model / function that maps features to label

For likelihood setting with observations being 0 or 1, it is *classification*

### Exercise:

What method would you use for classification?

What if observations were not 0/1 but continuous numbers?

# Appendix: Converting Content to Features

---

Challenge: Content is *not* structured

e.g. recall user information from MovieLens data

```
user id | age | gender | occupation | zip code
```

```
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
```

How do we convert these “attributes” or “content” to features

## Appendix: Converting Content to Features

---

Challenge: Content is *not* structured

How do we convert these “attributes” or “content” to features

Age: It's a number. That's easy.

Gender: Two classes or binary. Convert into 0 / 1.

Occupation:

Treat as a class. Use one-hot encoding.

```
user id | age | gender | occupation | zip code
```

```
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
```

# Appendix: Converting Content to Features

---

Challenge: Content is *not* structured

What about “Tip” data. It has free-form text.

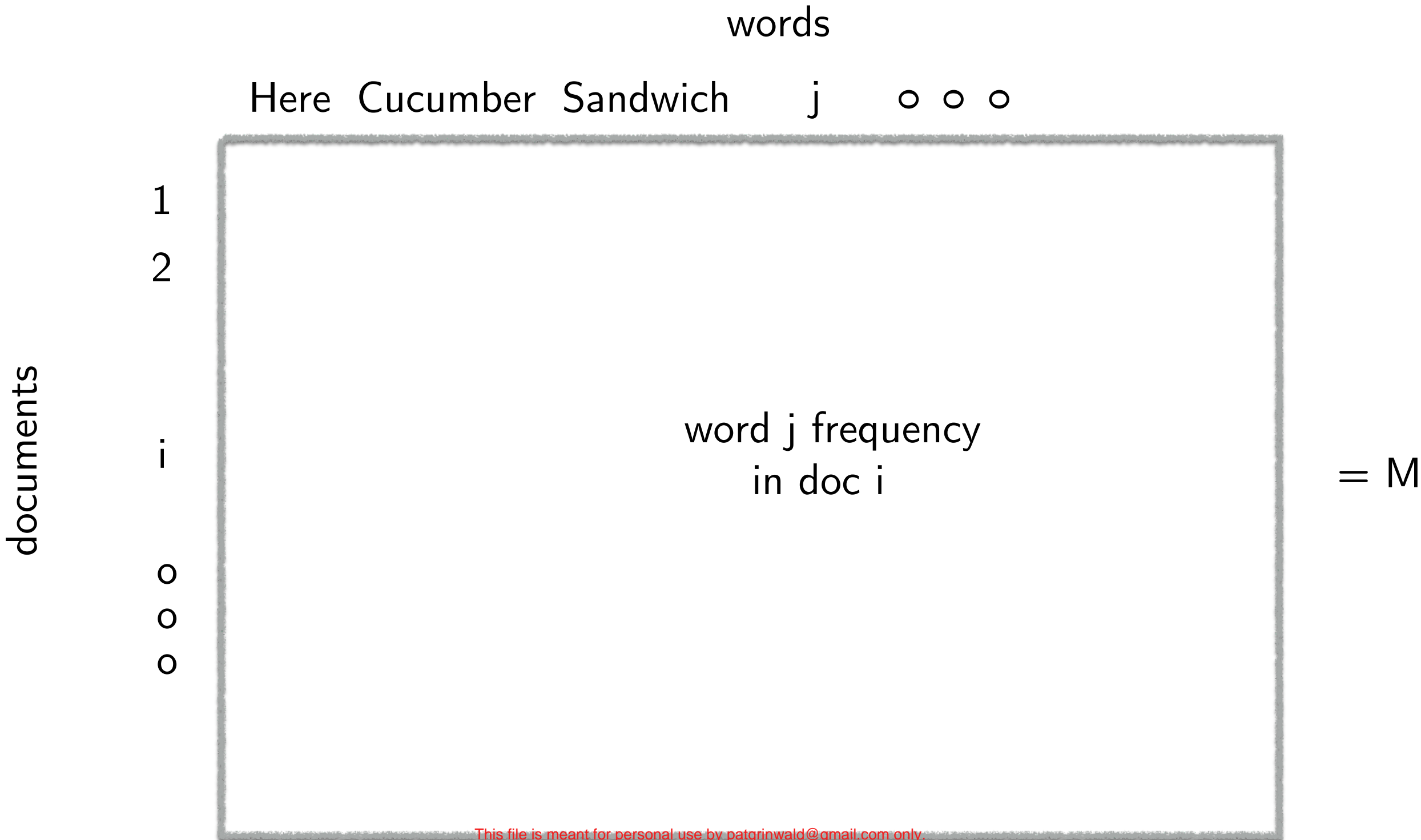
	business_id	compliment_count	date	text	user_id
0	UYX5zL_Xj9WEc_Wp-FrqHw	0	2013-11-26 18:20:08	Here for a quick mtg	hf27xTME3EiCp6NL6VtWZQ
1	Ch3HkwQYv1YKw_FO06vBWA	0	2014-06-15 22:26:45	Cucumber strawberry refresher	uEvusDwoSymbJJ0auR3muQ
2	rDoT-MgxGRiYqCmi0bG10g	0	2016-07-18 22:03:42	Very nice good service good food	AY-lalws3S7YXNI_f_D6rQ
3	OHXnDV01gLokiX1ELaQufA	0	2014-06-06 01:10:34	It's a small place. The staff is friendly.	Ue_7yUlKEbX4AhnYdUfL7g
4	GMrwDXRIAzu2zj5nH6l4vQ	0	2011-04-08 18:12:01	8 sandwiches, \$24 total...what a bargain!!! An...	LltbT_fUMqZ-ZJP-vJ84IQ

Need an approach to convert text into number or vector of numbers

# Appendix: Converting Content to Features

Text to vector of number:

Create word-frequency in documents matrix  $M$





## Appendix: Converting Content to Features

---

Text to vector of number:

Create word-frequency in documents matrix  $M$

Perform Principal Component Analysis of  $M$

Each document receives  $k$  co-ordinates  
via  $k$  principal components

This is the vector representing the text features (restricted to data)

Another (more classical) option:

TF-IDF vector

But it can be very large

## Outline — Part II

---

Solution evolves, Matrix estimation

Module 1: clustering

Finding user and item clusters

Averaging *within* clusters

Module 2: collaborative filtering aka personalized clustering

Finding users and items similar to a given user, item

Averaging *amongst* user-item specific *similar* users, items

Module 3: singular value thresholding, optimization

It's a Matrix! find Singular Value Decomposition

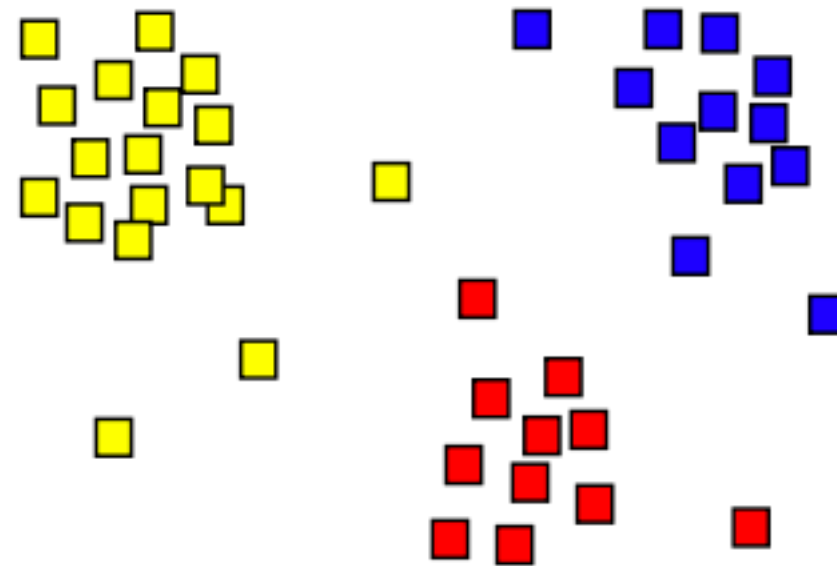
Solve for least-squares

# Module 1: clustering

# Clustering

---

What is a cluster?



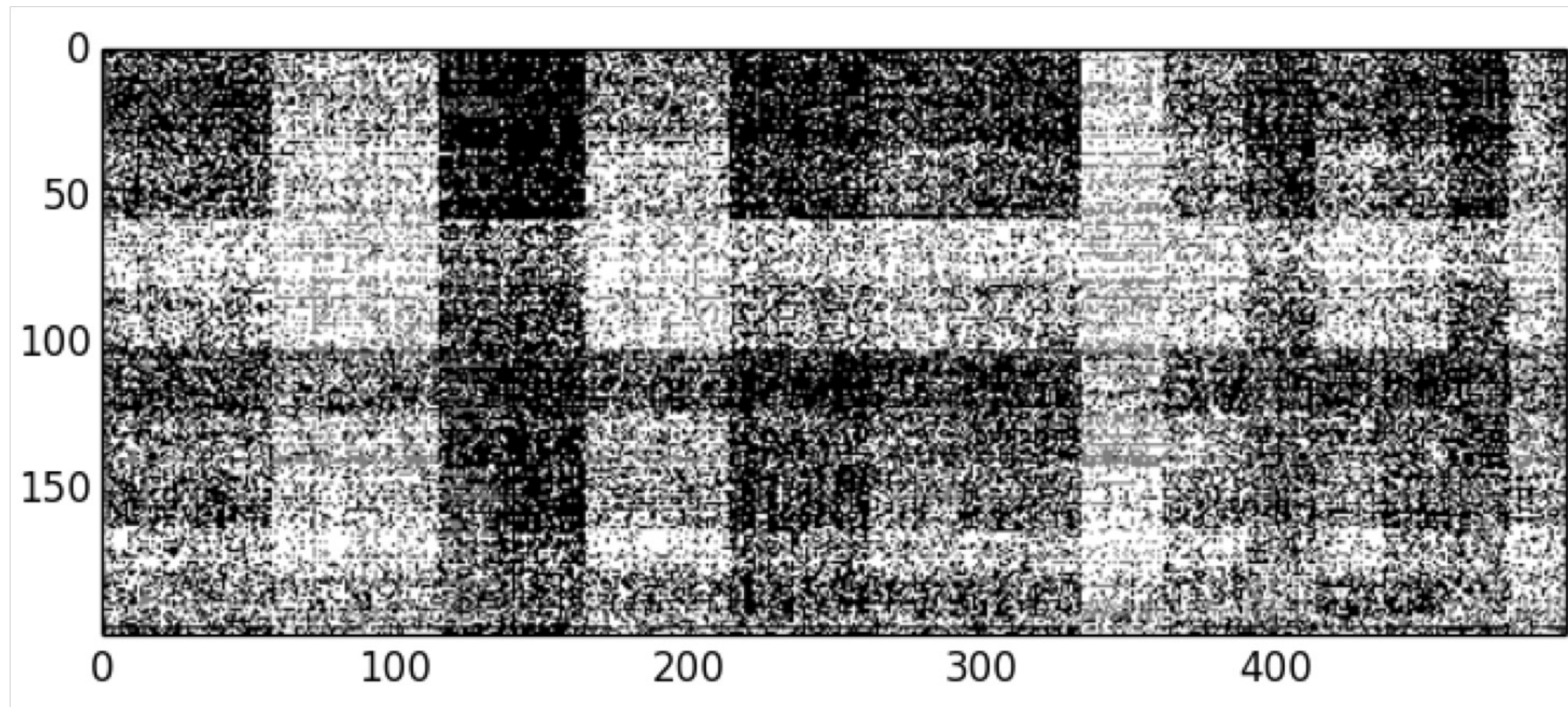
An example of Three clusters

Loosely speaking, a *cluster* is a collection of observations that are *more* similar to each other than the *rest*

# Clustering: MovieLens

---

Visual representation (after re-ordering) of rating matrix



top 200 users x top 500 items

# Clustering

---

We are interested in clusters of *users* and *items*

Why?

To estimate  $L_{ij}$ , we compute average over *all* rows, columns

This assumes *all* users (or items) being *homogenous*

Most likely that may not be true

However, it may be that users (and items) form  
multiple homogenous enough groups or clusters

Then find these clusters and

restrict averaging method to the cluster  
in which user/item of interest belongs



# Clustering

---

How to cluster users (or items)?

Step 1. Compute similarity between each pair of  $N$  users [How?]

This gives  $N \times N$  similarity matrix (that is symmetric)

Step 2. Obtain representation of each users in low-dim space [How?]

This assigns co-ordinates to each user in  $d$  dim space

Step 3. Perform k-means clusterings [How?]

Iteratively find  $k$  clusters till they make sense

# Explore: Yelp Data

---

## Exercise:

Apply clustering algorithm for Yelp Data

Compare the estimation error with respect to global averaging

Did it work better?

Data URL:

<https://www.kaggle.com/jagangupta/what-s-in-a-review-yelp-ratings-eda>

## Module 2: collaborative filtering

# Personalized Clustering

---

## Aggregate clustering

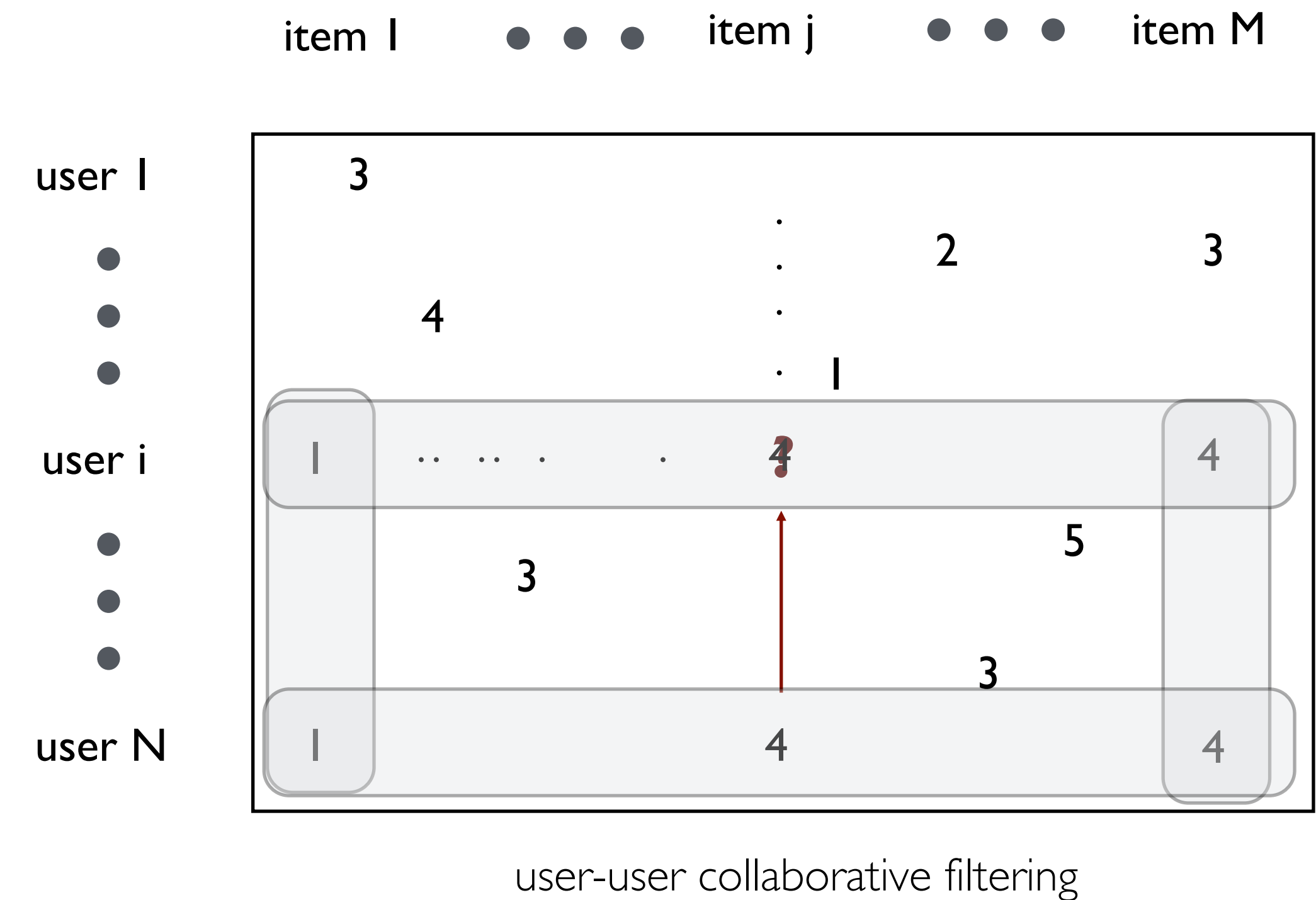
leads to many user (or item) being closer to 'boundary' of clusters  
that is the cluster utilized for its estimation  
may not be representative enough

## Personalized clustering

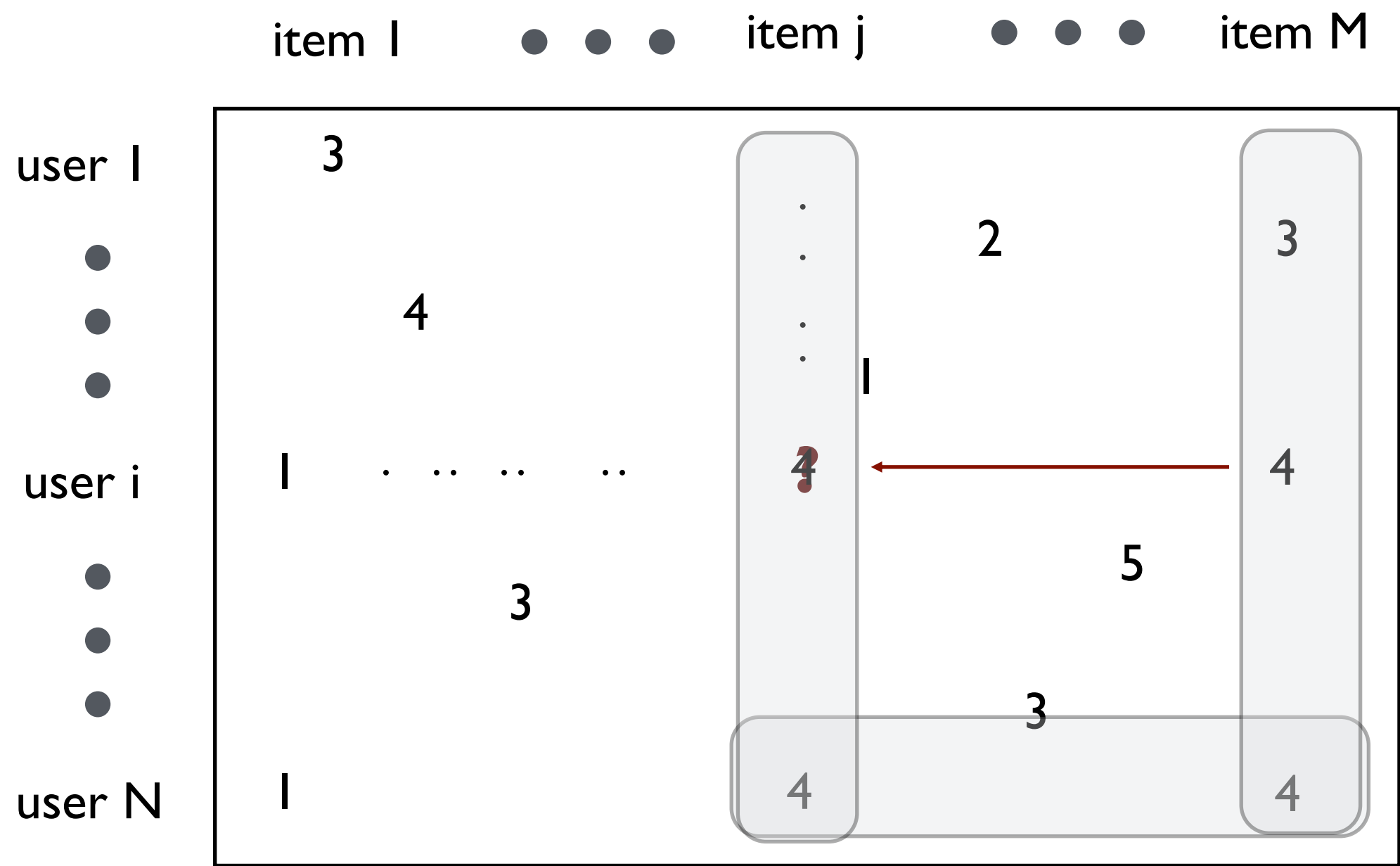
for each user (or item), find other users that are very similar to it  
declare them as it's *personal* cluster  
use the average over such a cluster to produce the estimate

This is precisely what collaborative filtering attempts to do

# Collaborative filtering



# Collaborative filtering



item-item collaborative filtering

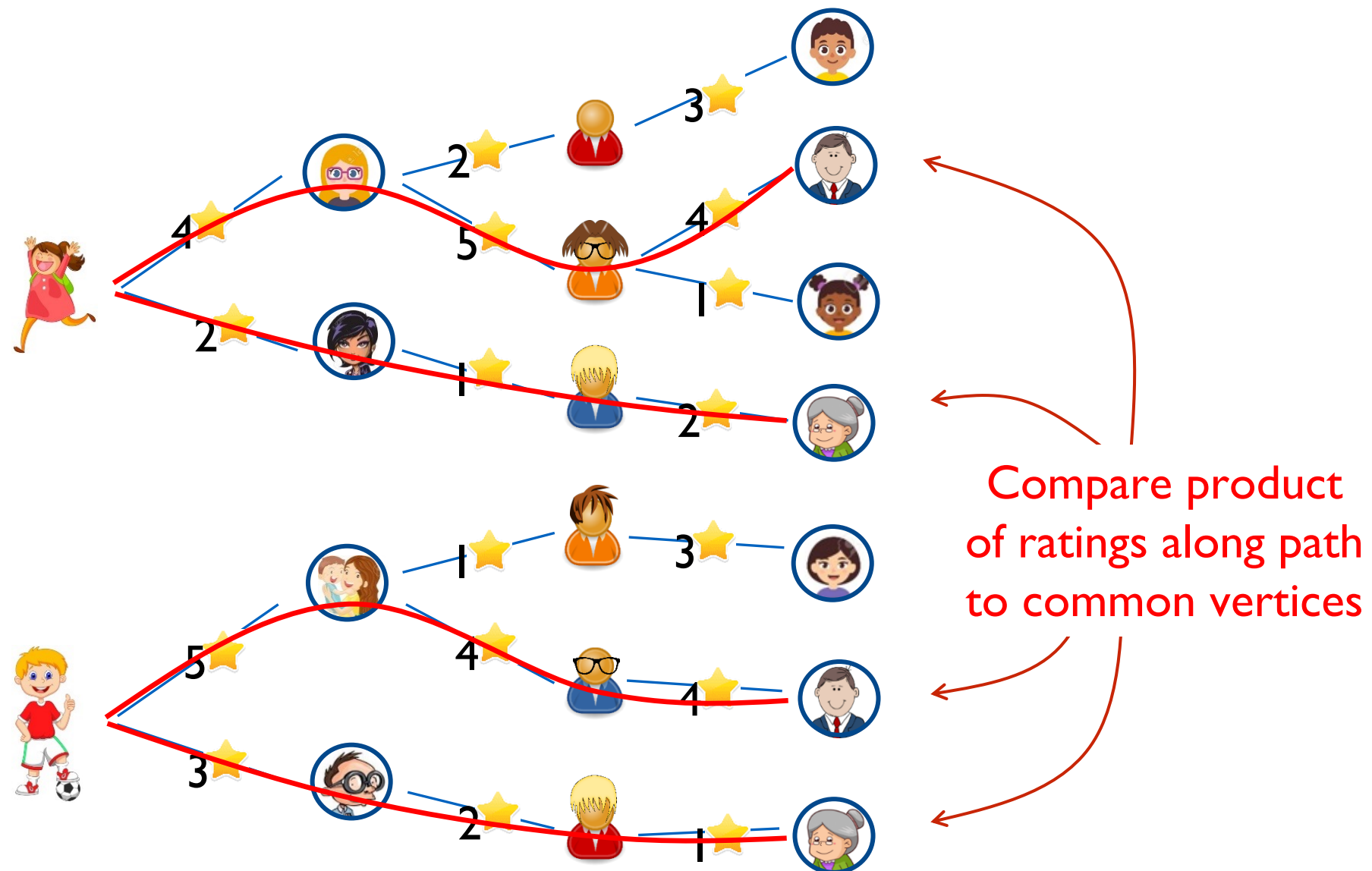
# Collaborative filtering



Computing similarity requires common observations  
(Birthday Paradox!)

What if observations are too sparse?

# Iterative collaborative filtering



Thy Friend is My Friend!



# Iterative collaborative filtering: In Summary

---

What if there are very few *neighbors* ?

Or they have not yet experienced the item of interest?

To overcome such sparse data challenge

Find users *similar* to user of interest

Next find users *similar* to these users

And continue *iterating* this procedure to find more *similar* users

till *enough similar* users are found

Use the experiences of such users to obtain the estimate

# Collaborative filtering

---

## Collaborative filtering and its variations

Extensively used in practice

Scalable implementation using “approximate nearest neighbors”

Closely related to non-parametric nearest neighbor method

Incremental and hence robust

Interpretable:

You are being recommended *GoodFellas* because you liked *Godfather*

And, those who liked *Godfather* also liked *GoodFellas*

# Explore: MovieLens Data

---

## Exercise:

Apply collaborative filtering algorithm for MovieLens Data

Compare the estimation error of

User-user collaborative filtering

Item-item collaborative filtering

Iterative collaborative filtering

Data URL:

<https://grouplens.org/datasets/movielens/100k>

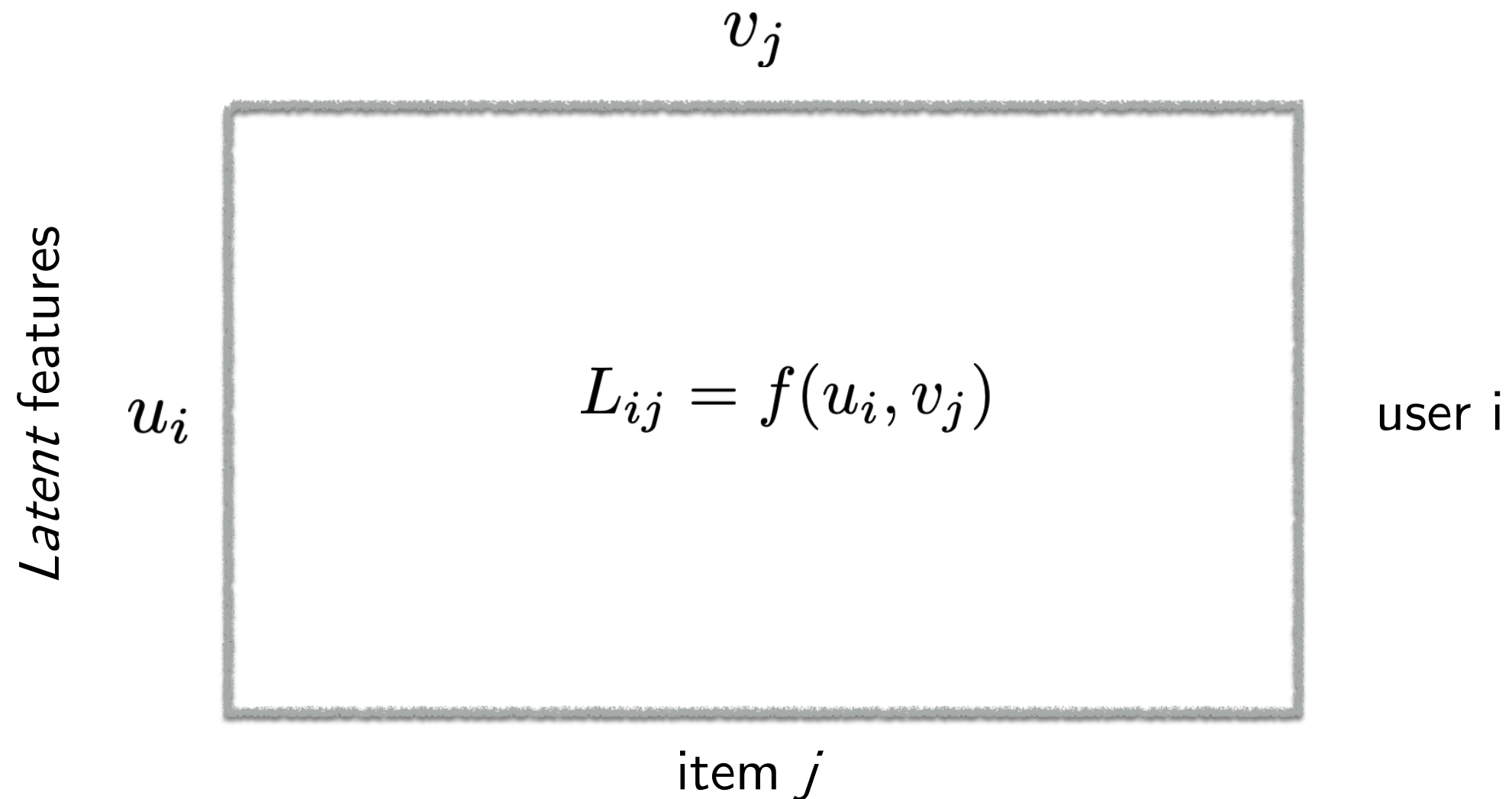
## Module 3: Singular Value Thresholding

# Matrix Estimation: Generic Model

---

Prediction problem: complete the matrix

*Latent* features



Collaborative filtering is solving such a problem

Using effectively “nearest neighbors” approach!

# Hey, L is a Matrix! Let's do Singular Value Decomposition

To estimate  $L_{ij}$  for any user  $i$  and item  $j$

We need to *fill* missing values in a matrix

Now, any matrix obeys singular value decomposition:  $\text{rank}(X)=r$

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n}^X = \begin{pmatrix} u_{11} & \dots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}_{m \times r}^U \begin{pmatrix} s_{11} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}_{r \times r}^S \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}_{r \times n}^{V^T}$$

An example:

$$\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -2 & -1 \end{bmatrix}$$

# Singular Value Thresholding

---

How to use SVD to estimate  $L_{ij}$  ?

Let  $L = U\Sigma V^T$  (assuming we know it)

That is,

$$L_{ij} = \sum_{k \leq r} s_k u_{ik} v_{jk}$$

Therefore, if we know U, S and V

Then we can estimate all missing values

And de-noise observed values

Question: how do we find SVD of L?

# Singular Value Thresholding

---

A natural algorithm: singular value thresholding

Step 1. Fill missing values in  $Y$  by 0

[better way to fill missing values?]

Step 2. Compute SVD

$$Y_{ij} = \sum_{k=1}^{\min(M,N)} s_k u_{ik} v_{jk}, \quad \text{for all } i, j$$

Step 3. Truncate SVD by keeping on top  $r$  components (+ normalize)

$$\hat{L}_{ij} = \frac{1}{\hat{p}} \sum_{k=1}^r s_k u_{ik} v_{jk}, \quad \text{for all } i, j$$

where  $\hat{p}$  is fraction of observed entries

[how to choose  $r$ ?]



# Explore: MovieLens Data

---

## Exercise:

Apply singular value thresholding algorithm for MovieLens Data

Compare the estimation error

Across different thresholds

How close is it to collaborative filtering?

Data URL:

<https://grouplens.org/datasets/movielens/100k>

# Optimization perspective

---

Singular value decomposition: optimization perspective

Let  $U, V$  be solution of

$$\begin{aligned} &\text{minimize} \sum_{i,j} \left( X_{ij} - \sum_{k=1}^r u_{ik} v_{jk} \right)^2 \\ &\text{over } u_{ik} \in \mathbb{R}, \ 1 \leq i \leq N, \ 1 \leq k \leq r \\ &\text{over } v_{jk} \in \mathbb{R}, \ 1 \leq j \leq M, \ 1 \leq k \leq r. \end{aligned}$$

Then,  $U$  and  $V$  are (effectively) left/right singular vectors

And,  $UV^T$  provides the best rank  $r$  approximate of  $X$

## Optimization perspective

---

This suggests the following algorithm

Find solution  $U, V$  of optimization problem

$$\begin{aligned} &\text{minimize} && \sum_{(i,j):\text{obs}} \left( Y_{ij} - \sum_{k=1}^r u_{ik} v_{jk} \right)^2 \\ &\text{over} && u_{ik} \in \mathbb{R}, \quad 1 \leq i \leq N, \quad 1 \leq k \leq r \\ &\text{over} && v_{jk} \in \mathbb{R}, \quad 1 \leq j \leq M, \quad 1 \leq k \leq r. \end{aligned}$$

For any  $i, j$ , produce estimate  $\hat{L}_{ij} = \sum_{k=1}^r u_{ik} v_{jk}$

## Optimization perspective

---

The algorithm uses *only* observed entries

and does not require filling missing values as in for SVD

The optimization problem can be solved

via iteratively solving for  $U$  and  $V$

also known as *Alternating Least Squares*

Food for thought:

Will it converge?

# Optimization perspective

---

**Exercise:** Singular Value Thresholding meets Alternative Least Squares

Initialize by filling missing values with 0

Singular Value Thresholding to obtain an estimate.

Use outcome to fill missing values and then perform Alt. Least. Sqs.

Iterate.

What are the advantages?

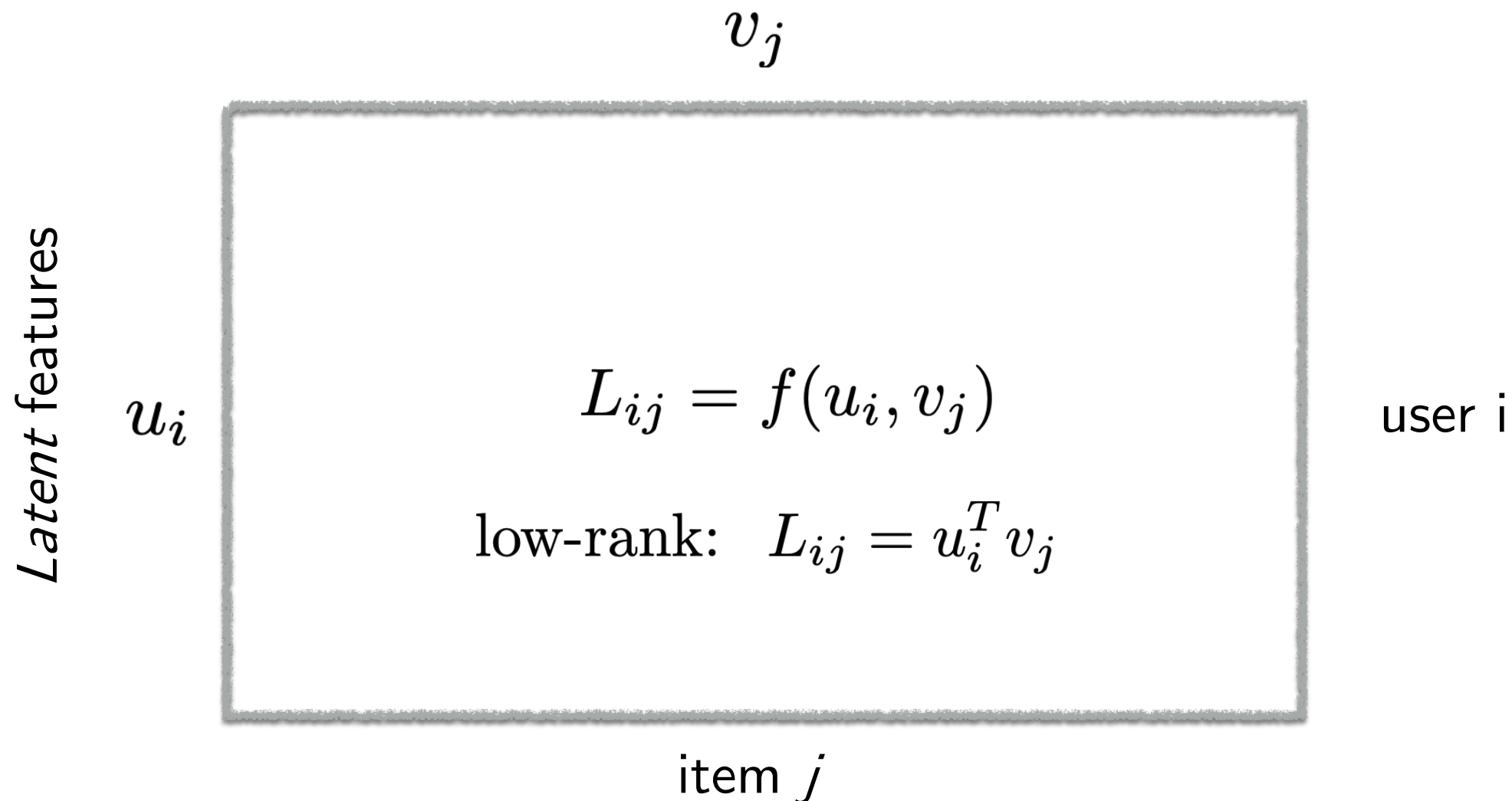
Use MovieLens and/or Yelp data to answer.

# Matrix Estimation: Generic Model

---

Prediction problem: complete the matrix

*Latent* features



Problem reduces to learning “factorization” of the matrix  
either through similarities or algebraic approaches

# Appendix: Matrix Estimation with Neural Networks

---

Singular Value Thresholding

bi-linear function of latent features

“Generalized” Singular Value Thresholding

generic “activation” function of latent features

multiple layers

this provides neural network implementation

**Exercise:** compare performance with other methods