



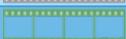
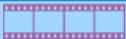
# Matrix Completion

Want to recommend movies based on user ratings for movies.

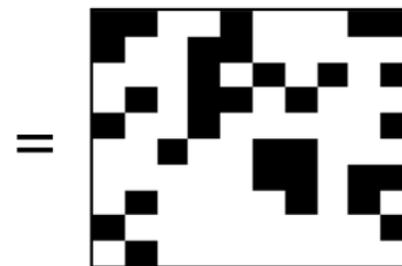
**Challenge:** Users have rated relatively few of the entire catalog

Can think of this as a matrix of users and ratings with missing data!

Input Data

User	Movie	Rating
		★★★★☆
		★★★★★
		★★★☆☆
		★★★★☆
		★★★★★
		★★★☆☆
		★★★★★
		★★★★★

					
User 1	5				3
User 2		2		4	
User 3			3		
User 4	1				
User 5			4		
User 6		5			2



# Matrix Factorization Assumptions

Assume that each item has  $k$  (unknown) features.

e.g.,  $k$  possible genres of movies (action, romance, sci-fi, etc.)

Then, we can describe an item  $v$  with feature vector  $\mathbf{R}_v$

How much is the movie action, romance, sci-fi, ...

e.g.,  $\mathbf{R}_v = [0.3, 0.01, 1.5, \dots]$

We can also describe each user  $u$  with a feature vector  $\mathbf{L}_u$

How much they like action, romance, sci-fi, ....

Example:  $\mathbf{L}_u = [2.3, 0, 0.7, \dots]$

Estimate rating for user  $u$  and movie  $v$  as

$$\widehat{Rating}(\mathbf{u}, \mathbf{v}) = \mathbf{L}_u \cdot \mathbf{R}_v = 2.3 \cdot 0.3 + 0 \cdot 0.01 + 0.7 \cdot 1.5 + \dots$$

# Example

Suppose we have learned the following user and movie features using 2 features

User ID	Feature
1	(2, 0)
2	(1, 1)
3	(0, 1)
4	(2, 1)

Movie ID	Feature vector
1	(3, 1)
2	(1, 2)
3	(2, 1)

Then we can predict what each user would rate each movie

$$\begin{matrix} L \\ \begin{array}{|c|c|} \hline 2 & 0 \\ \hline 1 & 1 \\ \hline 0 & 1 \\ \hline 2 & 1 \\ \hline \end{array} \end{matrix} \begin{matrix} R^T \\ \begin{array}{|c|c|c|} \hline 3 & 1 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \end{matrix} = \begin{matrix} \begin{array}{|c|c|c|} \hline 6 & 2 & 4 \\ \hline 4 & 3 & 3 \\ \hline 1 & 2 & 1 \\ \hline 7 & 4 & 5 \\ \hline \end{array} \end{matrix}$$



# Unique Solution?

Is this problem well posed? Unfortunately, there is not a unique solution ☹

For example, assume we had a solution

$$\begin{array}{|c|c|c|} \hline 6 & 2 & 4 \\ \hline 4 & 3 & 3 \\ \hline 1 & 2 & 1 \\ \hline 7 & 4 & 5 \\ \hline \end{array} = \begin{array}{|c|c|} \hline L & R^T \\ \hline 2 & 0 \\ \hline 1 & 1 \\ \hline 0 & 1 \\ \hline 2 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|} \hline 3 & 1 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Then doubling everything in  $L$  and halving everything in  $R$  is also a valid solution. The same is true for all constant multiples.

$$\begin{array}{|c|c|c|} \hline 6 & 2 & 4 \\ \hline 4 & 3 & 3 \\ \hline 1 & 2 & 1 \\ \hline 7 & 4 & 5 \\ \hline \end{array} = \begin{array}{|c|c|} \hline L & R^T \\ \hline 4 & 0 \\ \hline 2 & 2 \\ \hline 0 & 2 \\ \hline 4 & 2 \\ \hline \end{array} \begin{array}{|c|c|c|} \hline 1.5 & 0.5 & 1.0 \\ \hline 0.5 & 1.0 & 0.5 \\ \hline \end{array}$$



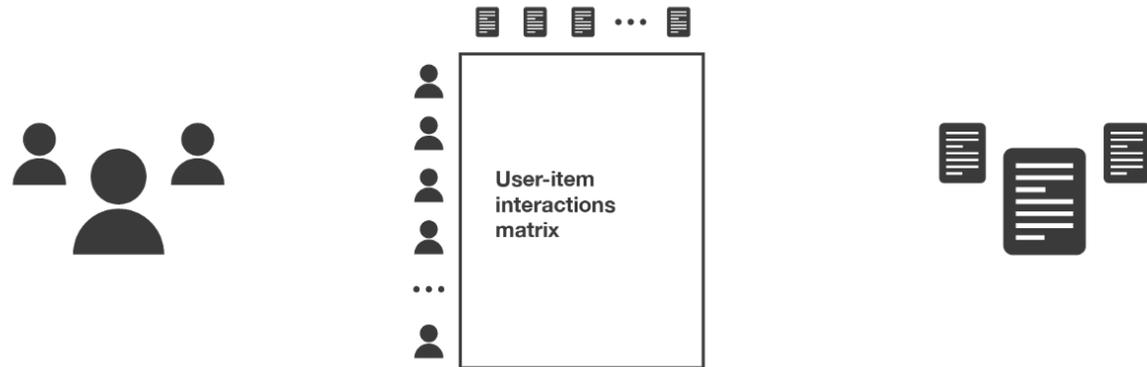
# Recommender Systems Setup

You have  $n$  users and  $m$  items in your system

- Typically,  $n \gg m$ . E.g., Youtube: 2.6B users, 800M videos

Based on the content, we have a way of measuring user preference.

This data is put together into a **user-item interaction matrix**.

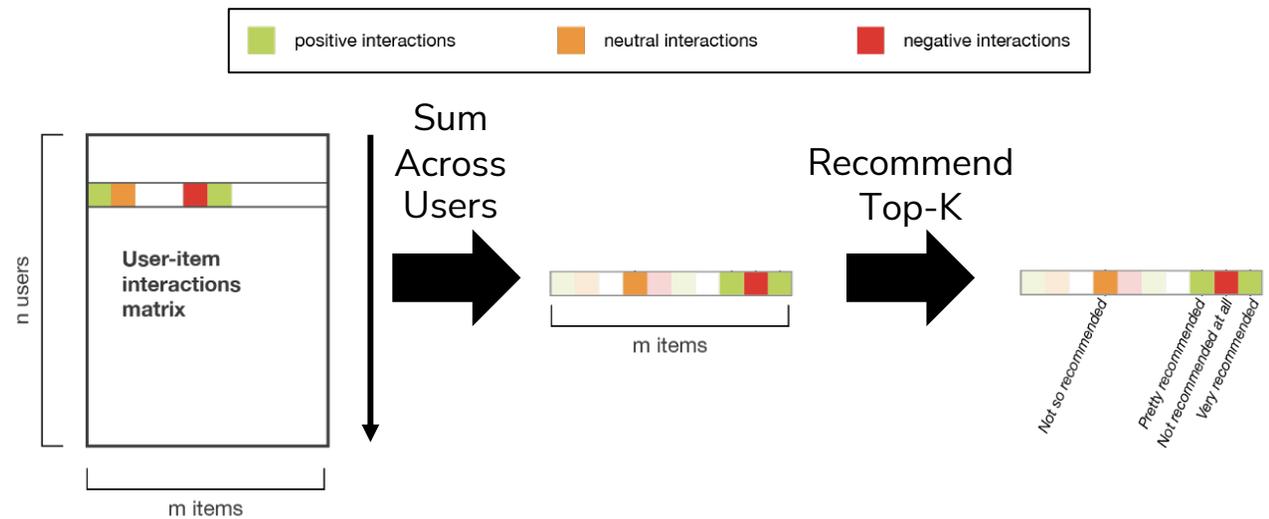


Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products
	...	

**Task:** Given a user  $u_i$  or item  $v_j$ , predict one or more items to recommend.

# Solution 0: Popularity

**Simplest Approach:** Recommend whatever is popular  
Rank by global popularity (i.e., Squid Game)

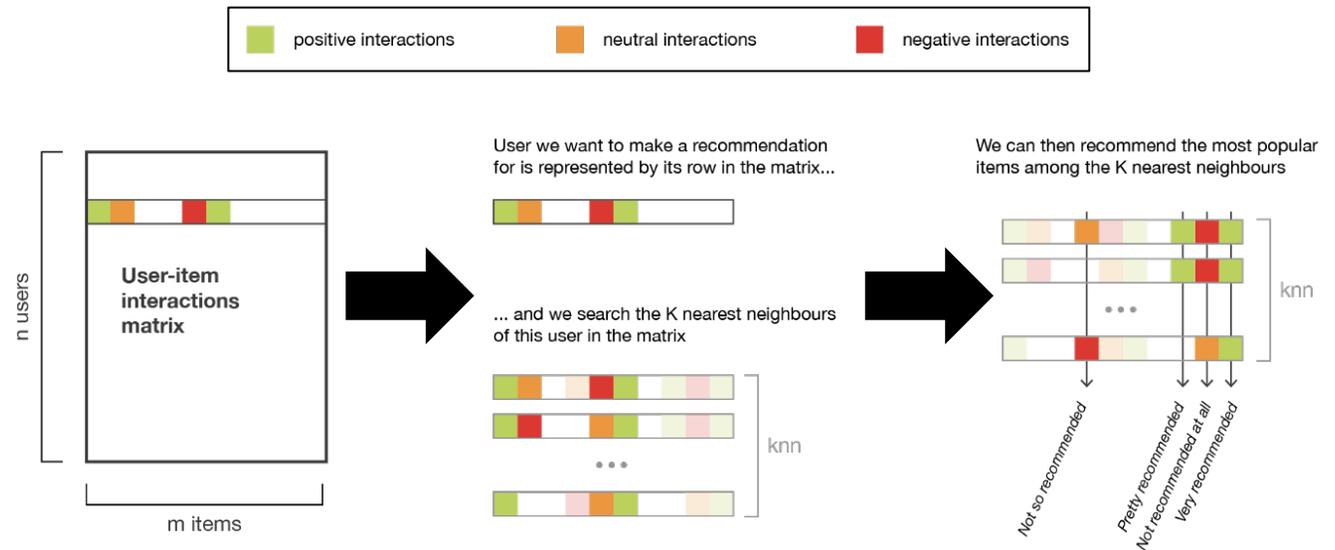


# Solution 1: Nearest User (User-User)

## User-User Recommendation:

Given a user  $u_i$ , compute their  $k$  nearest neighbors.

Recommend the items that are most popular amongst the nearest neighbors.



## Solution 2: “People Who Bought This Also Bought...” (Item-Item)

### Item-Item Recommendation:

Create a **co-occurrence matrix**  $C \in \mathbb{R}^{m \times m}$  ( $m$  is the number of items).  $C_{ij} = \#$  of users who bought both item  $i$  and  $j$ .

For item  $i$ , predict the top-k items that are bought together.

	Sunglasses	Baby Bottle	...	Diapers	Swim Trunks	Baby Formula
Sunglasses	500	15	...	9	130	20
Baby Bottle	15	45	...	6	10	10
...	...	...	...	...	...	...
Diapers	9	6	...	30	9	6
Swim Trunks	130	10	...	9	200	8
Baby Formula	20	10	...	6	8	50

# Normalizing Co- Occurrence Matrices

**Problem:** popular items drown out the rest!

**Solution:** Normalizing using Jaccard Similarity.

$$S_{ij} = \frac{\# \text{ purchased } i \text{ and } j}{\# \text{ purchased } i \text{ or } j} = \frac{C_{ij}}{C_{ii} + C_{jj} - C_{ij}}$$

	Sunglasses	Baby Bottle	...	Diapers	Swim Trunks	Baby Formula
Sunglasses	1.00	0.03	...	0.02	0.23	0.04
Baby Bottle	0.03	1.00	...	0.09	0.04	0.12
...	...	...	...	...	...	...
Diapers	0.02	0.09	...	1.00	0.04	0.08
Swim Trunks	0.23	0.04	...	0.04	1.00	0.03
Baby Formula	0.04	0.12	...	0.08	0.03	1.00

Solution 3:  
Feature-  
Based

## Solution 3: Feature- Based

What if we know what factors lead users to like an item?

**Idea:** Create a feature vector for each item. Learn a regression model!

Genre	Year	Director	...
Action	1994	Quentin Tarantino	...
Sci-Fi	1977	George Lucas	...

Define weights on these features for **all users** (global)

$$w_G \in \mathbb{R}^d$$

Fit linear model

## Solution 3: Feature- Based

What if we know what factors lead users to like an item?

**Idea:** Create a feature vector for each item. Learn a regression model!

Genre	Year	Director	...
Action	1994	Quentin Tarantino	...
Sci-Fi	1977	George Lucas	...

Define weights on these features for **all users** (global)

$$w_G \in \mathbb{R}^d$$

Fit linear model

$$\hat{r}_{uv} = w_G^T h(v) = \sum_i w_{G,i} h_i(v)$$

$$\hat{w}_G = \underset{w}{\operatorname{argmin}} \frac{1}{\# \text{ ratings}} \sum_{u,v:r_{uv} \neq ?} (w_G^T h(v) - r_{uv})^2 + \lambda \|w_G\|$$

## Personalization: Option A

Add user-specific features to the feature vector!

Genre	Year	Director	...	Gender	Age	...
Action	1994	Quentin Tarantino	...	F	25	...
Sci-Fi	1977	George Lucas	...	M	42	...



## Personalization: Option B

Include a user-specified deviation from the global model.

$$\hat{r}_{uv} = (\hat{w}_G + \hat{w}_u)^T h(v)$$

Start a new user at  $\hat{w}_u = 0$ , update over time.

OLS on the residuals of the global model

Bayesian Update (start with a probability distribution over user-specific deviations, update as you get more data)



## Solution 3 (Feature- Based) Pros / Cons

### Pros:

No cold-start issue!

- Even if a new user/item has no purchase history, you know features about them.

Personalizes to the user and item.

Scalable (only need to store weights per feature)

Can add arbitrary features (e.g., time of day)

### Cons:

Hand-crafting features is very tedious and unscalable 😞



## Solution 4: Matrix Factorization

*Can we learn the  
features of items?*

# Matrix Factorization Assumptions

Assume that each item has  $k$  (unknown) features.

e.g.,  $k$  possible genres of movies (action, romance, sci-fi, etc.)

Then, we can describe an item  $v$  with feature vector  $\mathbf{R}_v$

How much is the movie action, romance, sci-fi, ...

e.g.,  $\mathbf{R}_v = [0.3, 0.01, 1.5, \dots]$

We can also describe each user  $u$  with a feature vector  $\mathbf{L}_u$

How much they like action, romance, sci-fi, ....

Example:  $\mathbf{L}_u = [2.3, 0, 0.7, \dots]$

Estimate rating for user  $u$  and movie  $v$  as

$$\widehat{Rating}(\mathbf{u}, \mathbf{v}) = \mathbf{L}_u \cdot \mathbf{R}_v = 2.3 \cdot 0.3 + 0 \cdot 0.01 + 0.7 \cdot 1.5 + \dots$$

# Matrix Factorization Learning

**Goal:** Find  $L_u$  and  $R_v$  that when multiplied, achieve predicted ratings that are close to the values that we have data for.

Our quality metric will be (could use others)

$$\hat{L}, \hat{R} = \operatorname{argmin}_{L,R} \frac{1}{\# \text{ ratings}} \sum_{u,v:r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

This is the MSE, but we are learning both “weights” (how much the user likes each feature) and item features!



# slido

Think 

1 min

Suppose we have learned the following user and movie features using 2 features

User ID	Feature
1	(2, 0)
2	(1, 1)
3	(0, 1)
4	(2, 1)

Movie ID	Feature vector
1	(3, 1)
2	(1, 2)
3	(2, 1)

What is the predicted rating user 1 will have of movie 2?

What is the highest predicted rating from this matrix factorization model? Which user made the prediction, for which movie?

# slido

Group



2 min

Suppose we have learned the following user and movie features using 2 features

User ID	Feature
1	(2, 0)
2	(1, 1)
3	(0, 1)
4	(2, 1)

Movie ID	Feature vector
1	(3, 1)
2	(1, 2)
3	(2, 1)

What is the predicted rating user 1 will have of movie 2?

What is the highest predicted rating from this matrix factorization model? Which user made the prediction, for which movie?

# Example

Suppose we have learned the following user and movie features using 2 features

User ID	Feature
1	(2, 0)
2	(1, 1)
3	(0, 1)
4	(2, 1)

Movie ID	Feature vector
1	(3, 1)
2	(1, 2)
3	(2, 1)

Then we can predict what each user would rate each movie

$$\begin{matrix} L & R^T \\ \begin{matrix} 2 & 0 \\ 1 & 1 \\ 0 & 1 \\ 2 & 1 \end{matrix} & \begin{matrix} 3 & 1 & 2 \\ 1 & 2 & 1 \end{matrix} \end{matrix} = \begin{matrix} \begin{matrix} 6 & 2 & 4 \\ 4 & 3 & 3 \\ 1 & 2 & 1 \\ 7 & 4 & 5 \end{matrix} \end{matrix}$$

# Coordinate Descent

## Find $\hat{L}$ & $\hat{R}$

Remember, our quality metric is

$$\hat{L}, \hat{R} = \operatorname{argmin}_{L,R} \frac{1}{\# \text{ ratings}} \sum_{u,v:r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

Gradient descent is not used in practice to optimize this, since it is much easier to implement **coordinate descent** (i.e., Alternating Least Squares) to find  $\hat{L}$  and  $\hat{R}$



# Coordinate Descent

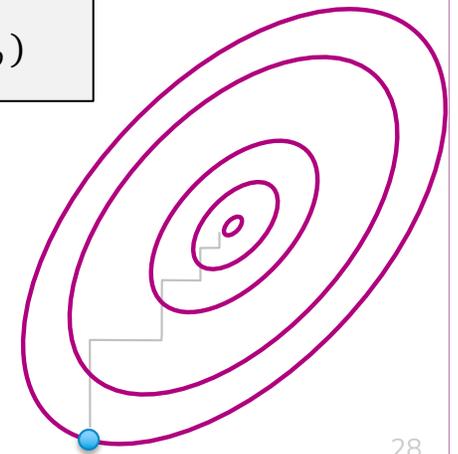
**Goal:** Minimize some function  $g(w) = g(w_0, w_1, \dots, w_D)$

Instead of finding optima for all coordinates, do it for one coordinate at a time!

*Initialize  $\hat{w} = 0$  (or smartly)*  
*while not converged:*  
    *pick a coordinate  $j$*   
     $\hat{w}_j = \underset{w}{\operatorname{argmin}} g(\hat{w}_0, \dots, \hat{w}_{j-1}, w, \hat{w}_{j+1}, \dots, \hat{w}_D)$

To pick coordinate, can do round robin or pick at random each time.

Guaranteed to find an optimal solution under some constraints



# Coordinate Descent for Matrix Factorization

$$\hat{L}, \hat{R} = \operatorname{argmin}_{L, R} \frac{1}{\# \text{ ratings}} \sum_{u, v: r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

Fix movie factors  $R$  and optimize for  $L$

$$\hat{L} = \operatorname{argmin}_L \frac{1}{\# \text{ ratings}} \sum_{u, v: r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

**First key insight:** users are independent!

$$\hat{L}_u = \operatorname{argmin}_{L_u} \frac{1}{\# \text{ ratings for } u} \sum_{v: r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

# Coordinate Descent for Matrix Factorization

$$\hat{L}_u = \operatorname{argmin}_{L_u} \frac{1}{\# \text{ ratings for } u} \sum_{v:r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

**Second key insight:** this looks a lot like linear regression!

$$\hat{w} = \operatorname{argmin}_w \frac{1}{n} \sum_{i=1}^n (w \cdot h(x_i) - y_i)^2$$

**Takeaway:** For a fixed  $R$ , we can learn  $L$  using linear regression, separately per user.

Conversely, for a fixed  $L$ , we can learn  $R$  using linear regression, separately per movie.

# Overall Algorithm

Want to optimize

$$\hat{L}, \hat{R} = \operatorname{argmin}_{L, R} \frac{1}{\# \text{ ratings}} \sum_{u, v: r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2$$

Fix movie factors  $R$ , and optimize for user factors separately

**Step 1:** Independent least squares for each user

$$\hat{L}_u = \operatorname{argmin}_{L_u} \frac{1}{\# \text{ ratings for } u} \sum_{v \in U_u} (L_u \cdot R_v - r_{uv})^2$$

Fix user factors, and optimize for movie factors separately

**Step 2:** Independent least squares for each movie

$$\hat{R}_v = \operatorname{argmin}_{R_v} \frac{1}{\# \text{ ratings for } v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2$$

Repeatedly do these steps until convergence (to local optima)

System might be underdetermined: Use regularization

# slido

Think 

1.5 minutes

Consider we had the ratings matrix

	Movie 1	Movie 2
User 1	4	?
User 2	?	2

During one step of optimization, user and movie factors are

	User Factors
User 1	[1, 2, 1]
User 2	[1, 1, 0]

	Movie Factors
Movie 1	[2, 1, 0]
Movie 2	[0, 0, 2]

Two questions:

**What is the current MSE loss? (number)**

**Assume the next step of coordinate descent updates the *user factors*. Which factors would change?**

- User 1
- User 2
- User 1 and 2
- None

# slido

Group



3 minutes

Consider we had the ratings matrix

	Movie 1	Movie 2
User 1	4	?
User 2	?	2

During one step of optimization, user and movie factors are

	User Factors
User 1	[1, 2, 1]
User 2	[1, 1, 0]

	Movie Factors
Movie 1	[2, 1, 0]
Movie 2	[0, 0, 2]

Two questions:

**What is the current MSE loss? (number)**

**Assume the next step of coordinate descent updates the *user factors*. Which factors would change?**

- User 1
- User 2
- User 1 and 2
- None



## Brain Break



# What Has Matrix Factorization Learnt?

Matrix Factorization is a very versatile technique!

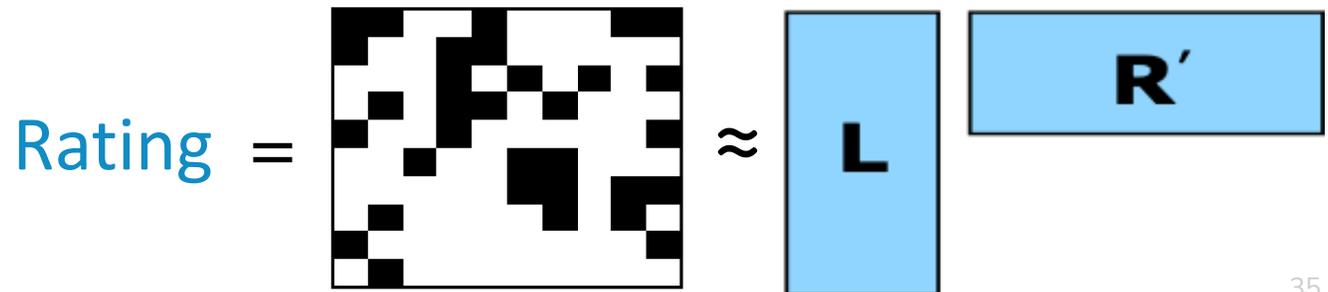
Learns a latent space of topics that are most predictive of user preferences.

Learns the “topics” that exist in movie  $v$ :  $\hat{R}_v$

Learns the “topic preferences” for user  $u$ :  $\hat{L}_u$

Predict how much a user  $u$  will like a movie  $v$

$$\widehat{Rating}(u, v) = \hat{L}_u \cdot \hat{R}_v$$

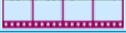
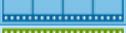
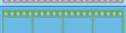


# Applications: Recommender Systems

## Recommendations: (Semi-Supervised)

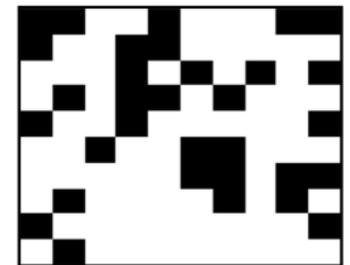
Use matrix factorization to predict user ratings on movies the user hasn't watched

Recommend movies with highest predicted rating

User	Movie	Rating
		★ ★ ★ ☆ ☆
		★ ★ ★ ★ ★
		★ ★ ☆ ☆ ☆
		★ ★ ☆ ☆ ☆
		★ ★ ★ ★ ☆
		★ ☆ ☆ ☆ ☆
		★ ★ ★ ☆ ☆
		★ ★ ★ ★ ★
		★ ★ ★ ★ ☆

					
User 1	5				3
User 2		2		4	
User 3			3		
User 4	1				
User 5			4		
User 6		5			2

=



# Applications: Topic Modeling

## Topic Modeling: (Unsupervised)

Treat the TF-IDF matrix as the user-item matrix

- Documents are "users"
- Words are "items"

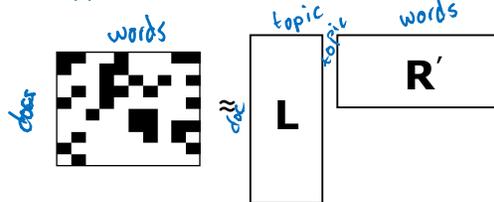
$L$  tells us which topics are present in each document

$R$  tells us what words each topic is composed of

Oftentimes, the topics are interpretable!

HW7 Programming: Tweet Topic Modeling

Application to text data:



party law government election court president elected council general minister national members committee united office federal member house parliament vote public elections democratic held

**son died** is a **married family** **king daughter john death william father born wife royal ireland irish henry house lord charles sir prince brother children england queen duke thomas years marriage george earl edward english**

**school students** university high college schools education year program student center members science national peers public academic association division arts educational include dean institute department teachers colleges classes office activities universities district engineering hearing founded faculty girls sports children boys international board teaching academy secondary established

**album band** **song released** **music** songs single records recorded rock bands release live tour video record albums label group recording guitar track cover version discography ranked top chart hit uk top performed studio played singles sound flow top artist sale cd digital singer artists members included early second base

**radio station** **news television** **channel broadcast** **stations network media tv** broadcasting time format local program bbc programming live streaming live stream sports for ar media daily channels digital also aired changed current launched communications programs day broadcasts moved six years satursday talk night

**york county** **american united** city washington john texas served virginia pennsylvania war moved ohio florida illinois george james died massachusetts president named jersey born boston michigan for years philadelphia white

**season team** **game league games** **species** family birds small long large animals **bird plants genus** plant natural habitat tree fish mammals whale shark octopus leaves brown common forests trees animal flowers eggs webbed feet social subspiral web longh mark leaping habitats single food female foot about needs endemic lower group including include most threatened list

**century king** **roman empire greek** **design model cars** production built engines vehicle class models speed vehicles designed produced power front system version type series motor rear standard gun company introduced range ford sold fuel also wheel tires fitted history reprints developed based replaced wheels time occurred small high weight electric body styling early

**art museum work** works artists collection design arts painting artist gallery paintings exhibition style fine including painted architecture york fashion painter 20 early modern sculpture artist history contemporary collections years museuma egypted images time photography figures academy exhibitions modern period photographs began studio drawing include exhibited produced designed period about

**war army** military forces battle force british command general navy ship division ships troops corps service naval regiment commander infantry attack men officer head soldiers units officers operations unit june august brigade july fighting march battalion guard cavalry captain september three enemy united british sea royal garrisons marine major

**white red** **black blue** called color will head green gold side small hand long arms top flag horse wear silver common light dog wood body type large yellow hair worn change last popular left generally traditional had three business shape hair feet color time coat three typically modern hair color

**age 18 population** **music** musical opera festival orchestra dance performed jazz piano theatre performance works concert symphony composer played performances instruments musicians classical including work composed major songs folk festival performance concerts playing stage years include piano other ensemble sound age time victor had piece chamber recordings songs

## Solution 4 (Matrix Factorization) Pros / Cons

### Pros:

Personalizes to item and user!

Learns latent features that are most predictive of user ratings.

### Cons:

Cold-Start Problem

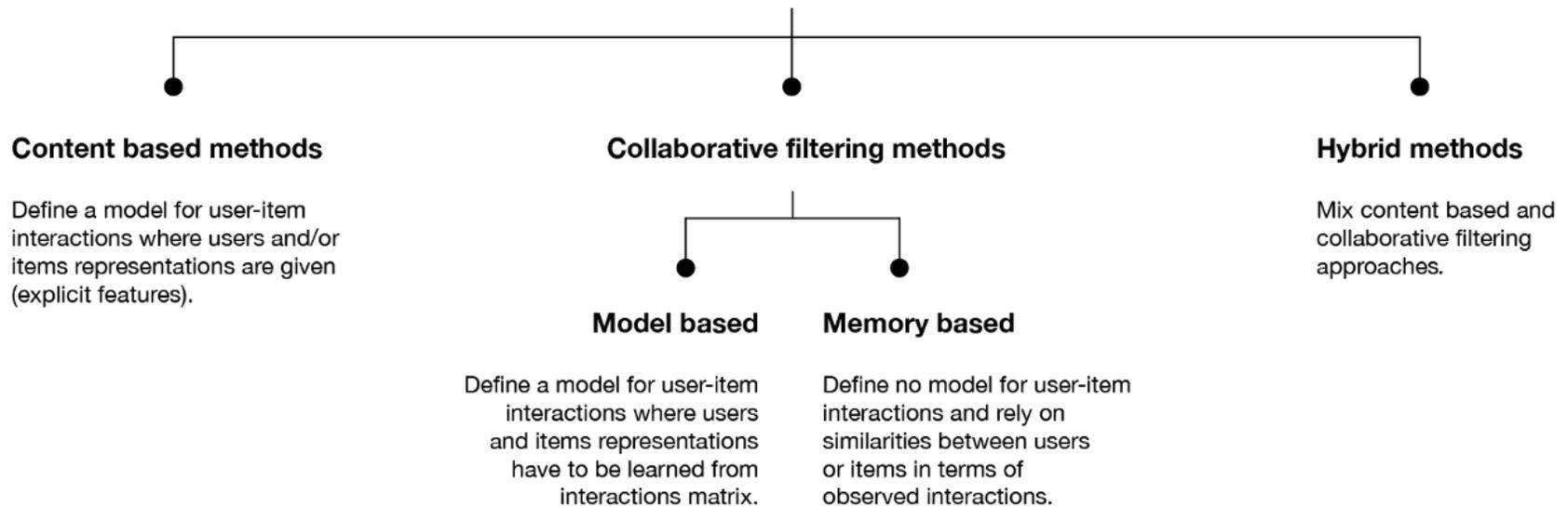
- What do you do about new users or items, with no data?



# Common Issues with Recommender Systems

*(and some solutions)*

## Recommender systems



# slido

Think 

1 min

You are a software engineer at Spotify and have developed a matrix-factorization based recommendation system. The system works very well on existing users and songs, but does not work on new users or new songs.

How can you augment, extend, and/or modify your recommender system to handle new songs/users?

# slido

Group



2 min

You are a software engineer at Spotify and have developed a matrix-factorization based recommendation system. The system works very well on existing users and songs, but does not work on new users or new songs.

How can you augment, extend, and/or modify your recommender system to handle new songs/users?

# Comparing Recommender Systems

	Efficiency (Space, Deploy)	Efficiency (Time, Deploy)	Addresses Cold-Start?	Personalizes to User?	Discovers Latent Features?
User-User					
Item-Item					
Feature-Based					
Matrix Factorization					
Hybrid (Feature-Based + Matrix Factorization)					



# Featurized Matrix Factorization

## **Feature-based approach**

Feature representation of user and movie fixed

Can address cold start problem

## **Matrix factorization approach**

Suffers from cold start problem

User & Movie features are learned from data

## **A unified model**



# Cold-Start Problem

When a new user comes in, we don't know what items they like!  
When a new item comes into our system, we don't know who likes it! This is called the **cold start** problem.

Addressing the cold-start problem (for new users):

- Give random predictions to a new user.

- Give the globally popular recommendations to a new user.

- Require users to rate items before using the service.

- Use a feature-based model (or a hybrid between feature-based and matrix factorization) for new users.



# Top-K versus Diverse Recommendations

Top-k recommendations might be very redundant

Someone who likes Rocky I also will likely enjoy Rocky II, Rocky III, Rocky IV, Rocky V

Diverse Recommendations

Users are multi-faceted & we want to hedge our bets

Maybe recommend: Rocky II, Always Sunny in Philadelphia, Robin Hood

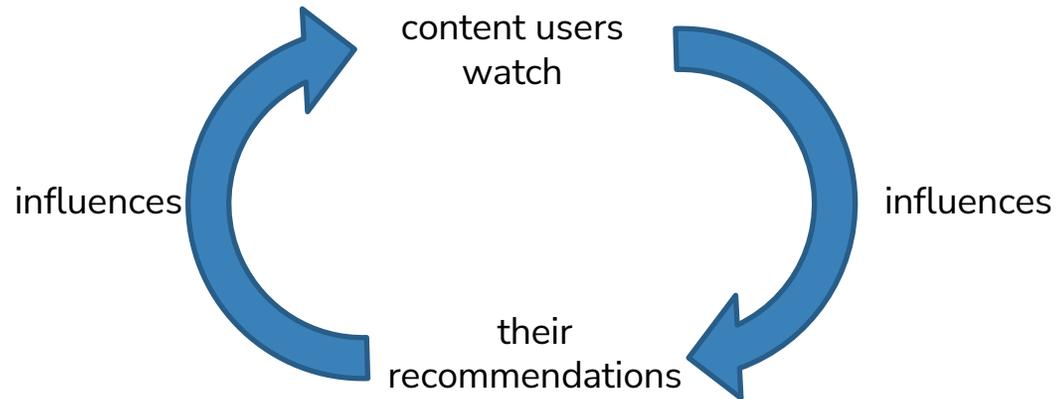
**Solution:** Maximal Marginal Relevance

Pick recommendations one-at-a-time.

Select the item that the user is most likely to like and that is most dissimilar from existing recommendations.

- Hyperparameter  $\lambda$  to trade-off between those objectives.

## Feedback Loops / Echo Chambers



Users always get recommended similar content and are unable to discover new content they might like.

### Exploration-Exploitation Dilemma

- Common problem in “online learning” settings

Pure Exploration: show users random content

- Users can discover new interests, but will likely be unsatisfied

Pure Exploitation: show users content they’re likely to like

- Users can’t discover new interests.

**Solution:** Multi-Armed Bandit Algorithms (beyond the scope of 416)

# Radicalization Pathways

In the real-world, recommender systems guide us along a path through the content in a service.

If watch video 1, recommend video 2

If watch video 2, recommend video 3

[A 2019 study](#) found that YouTube's algorithms lead users to more and more radical content.

“Intellectual Dark Web” → Alt-Lite → Alt-Right

See more: iSchool 2021 Spring Lecture on [Algorithmic Bias & Governance](#)

Youtube's response [has been whack-a-mole](#). (Remove the content, manually tweak the recommendations for that topic)



# TikTok

[2021 experiment](#) on time-to-seeing radical alt-right content



Source: <https://www.tiktok.com/@tofology/video/7016081760643534085?lang=en>

# Evaluating Recommender Systems

## MSE / Accuracy?

It is possible to evaluate recommender systems using existing metrics we have learnt:

- MSE (if predicting ratings)
- Accuracy (if predicting like/dislike, or click/ignore)

However, we don't really care about accurately predicting what a user **won't like**.

Rather, we care about finding the few items they will like.

Instead, we focus on the following metrics:

How many of our recommendations did the user like?

How many of the items that the user liked did we recommend?

Sound familiar?

# Precision - Recall

Precision and recall for recommender systems

$$precision = \frac{\# \text{ liked \& shown}}{\# \text{ shown}}$$

$$recall = \frac{\# \text{ liked \& shown}}{\# \text{ liked}}$$

What happens as we vary the number of recommendations we make?

Best Recommender System:

**Top-1:** high precision, low recall

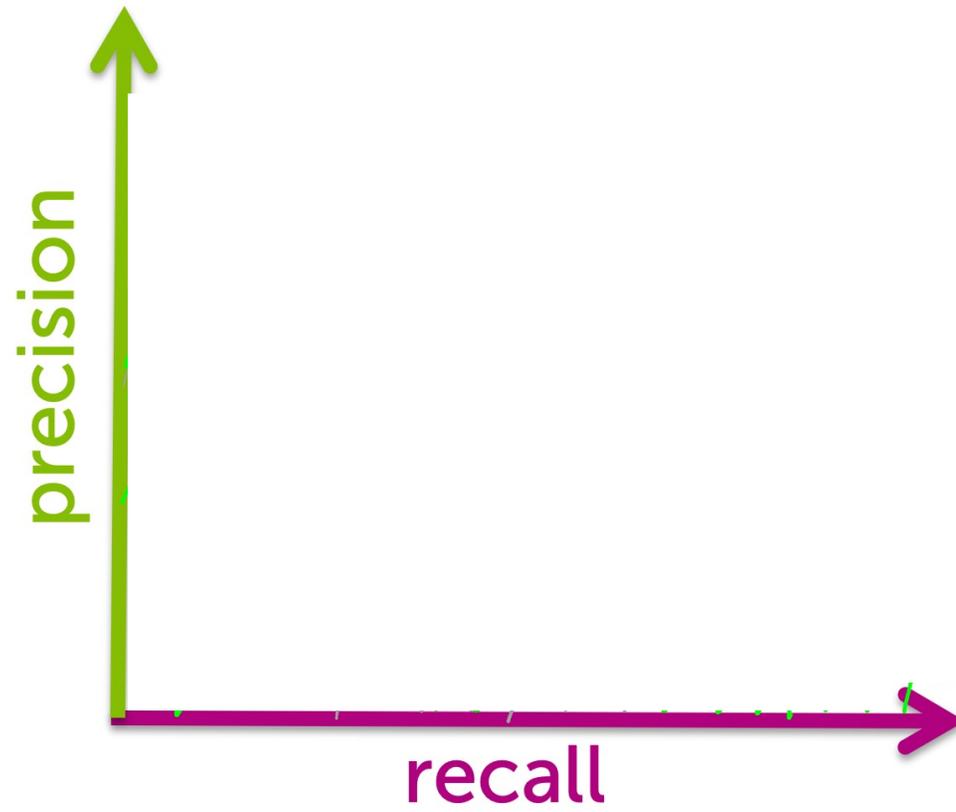
**Top-k (large k):** high precision, high recall

Average Recommender System:

**Top-1:** average precision, low recall

**Top-k (large k):** low precision, high recall

# Precision - Recall Curves



# Comparing Recommender Systems

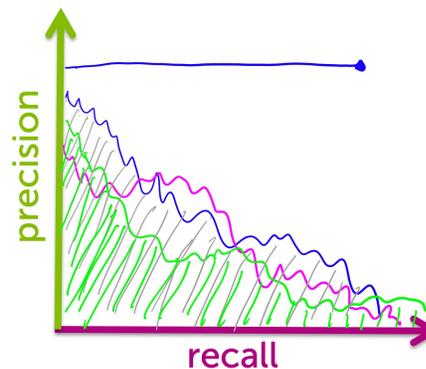
In general, it depends

What is true always is that for a given precision, we want recall to be as large as possible (and vice versa)

What target precision/recall depends on your application

One metric: area under the curve (**AUC**)

Another metric: Set desired recall and maximize precision (**precision at k**)



# Recap

Now you know how to:

Describe the input (observations, number of “topics”) and output (“topic” vectors, predicted values) of a matrix factorization model

Implement a coordinate descent algorithm for optimizing the matrix factorization objective presented

Compare different approaches to recommender systems

Describe the cold-start problem and ways to handle it (e.g., incorporating features)

Analyze performance of various recommender systems in terms of precision and recall

Use AUC or precision-at-k to select amongst candidate algorithms

