

# CSE/STAT 416

## Recommender Systems

Vinitra Swamy  
University of Washington  
Aug 12, 2020



Last Time...

# Personalization

Personalization is transforming our experience of the world

Youtube

Netflix

Amazon

Spotify

Facebook

Many more...

Almost all have share a common trait where there are users that use the system and items that we want the user to look at.

A recommender system recommends items to a user based on what we think will be the most “useful” for the user.

# Challenges

Types of Feedback (Explicit vs Implicit)

Diverse Outputs

Cold Start

Context (i.e. time)

Scalability

# Solution 0 : Popularity

Simplest Approach: Recommend whatever is popular

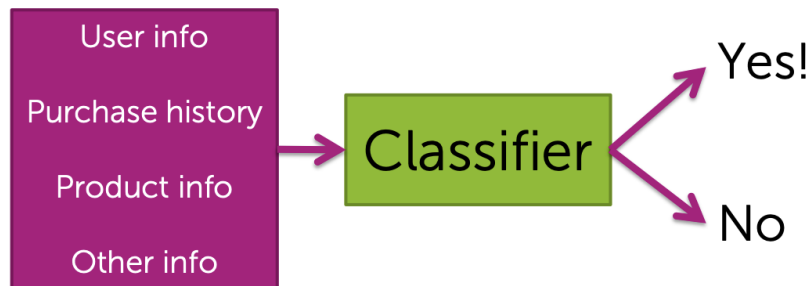
- Rank by global popularity (i.e. Avengers Endgame)

## **Limitations**

- No personalization
- Feedback loop

# Solution 1: Classification Model

Train a classifier to learn whether or not someone will like an item



## Pros

- Personalized
- Features can capture context (time of day, recent history, ...)
- Can even handle limited user history (age of user, location, ...)

## Cons

- Features might not be available or hard to work with
- Often doesn't perform well in practice when compared to more advanced techniques like **collaborative filtering**

# Co-occurrence Matrix

*Solution 2*

# Co-occurrence Matrix

**Idea:** People who bought this, also bought ...

- E.g. people who buy diapers also buy baby wipes

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

$C$  will be symmetric ( $C_{ij} = C_{ji}$ )



# Recommending

Assume a user has purchased diapers.

1. Look at diapers row (or column)
2. Recommend items with largest counts  
Baby wipes, milk, baby food, ...

# Normalization

The count matrix  $C$  needs to be normalized, otherwise popular items will drown out others (will just reduce to popularity).

Normalize the counts by using the **Jaccard similarity** instead

$$S_{ij} = \frac{\# \text{ purchased } i \text{ and } j}{\# \text{ purchased } i \text{ or } j}$$

Could also use something like Cosine similarity, but Jaccard is popular

# Purchase History

What if I know the user  $u$  has bought diapers and milk?

**Idea:** Take the average similarity between items they have bought

$$\text{Score}(u, \text{baby wipes}) = \frac{S_{\text{baby wipes}, \text{diapers}} + S_{\text{baby wipes}, \text{milk}}}{2}$$

Could also weight them differently based on recency of purchase!

Then all we need to do is find the item with the highest average score!

# Analysis

## Pros:

- It personalizes to the user

## Cons

- Does **not** utilize
  - Context (e.g. time of day)
  - User features (e.g. age)
  - Product features (e.g. baby vs electronics)
- Scalability
  - Similarity is size  $m^2$  where  $m$  is the number of items
- Cold start problem

# Matrix Factorization

*Solution 4*








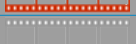



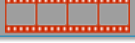








# Matrix Completion





Want to recommend movies based on user ratings for movies.

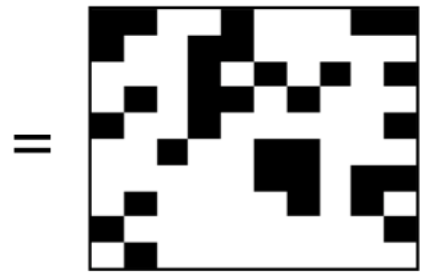
**Challenge:** Users have rated relatively few of the entire catalog

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



# Assumption

Matrix completion is an impossible task without some assumptions on data (unknowns could be anything otherwise).

**Assume:** There are  $k$  types of movies (e.g. action, romance, etc.) which users have various interests in.

This means we can describe a movie  $\mathbf{v}$  with feature vector  $\mathbf{R}_v$ .

- How much is the movie action, romance, drama, ...
- Example:  $\mathbf{R}_v = [0.3, 0.01, 1.5, \dots]$

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

- How much she likes action, romance, drama, ....
- Example:  $\mathbf{L}_u = [2.3, 0, 0.7, \dots]$

Estimate rating for user  $\mathbf{u}$  and movie  $\mathbf{v}$  as

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



# Brain Break





# Example

Suppose we have learned the following user and movie 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

$$L R^T =$$

2	0
1	1
0	1
2	1

3	1	2
1	2	1

$$=$$

6	2	4
4	3	3
1	2	1
7	4	5



# Unique Solution?

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

For example, assume we had a solution

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

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

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

# Coordinate Descent

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

Remember, our quality metric is

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

Gradient descent is not used much 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

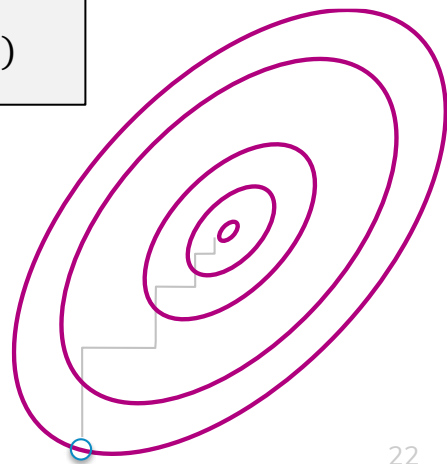
Goal: Minimize some function  $g(w) = g(w_0, w_1, \dots, w_D)$

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

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

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

First key insight:

# Coordinate Descent for Matrix Factorization

Holding movies fixed, we can solve for each user separately!

For each user  $u$

$$\hat{L}_u = \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$$

Second key insight:

Looks like linear regression!



# Overall Algorithm

Want to optimize

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

Fix movie factors, and optimize for user factors separately

- Independent least squares for each user

$$\hat{L}_u = \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2$$

Fix user factors, and optimize for movie factors separately

- Independent least squares for each movie

$$\hat{R}_v = \min_{R_v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2$$

System might be underdetermined: Use regularization

Converges to: local optima

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 residual sum of squares loss? (number)**

**If the next step of coordinate descent updates the user factors, which factors would change?**

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



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 residual sum of squares loss? (number)**

**If the next step of coordinate descent updates the user factors, which factors would change?**

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



# Using Results

Use movie factors  $\hat{R}$  to discover “topics” for movie  $v$ :  $\hat{R}_v$

Use user factors  $\hat{L}$  to discover “topic preferences” for user  $u$ :  $\hat{L}_u$

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

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

Recommendations: Sort movies user hasn't watched by

$\widehat{Rating}(u, v)$

- Recommend movies with highest predicted rating



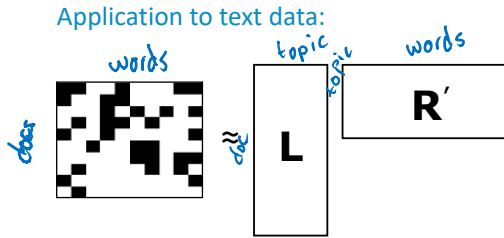
## Brain Break



# Topics

The “features” found by matrix factorization don’t always correspond to something meaningful (like film genre), but sometimes they do!

- Remember, the exact values are meaningless since we can scale them an infinite number of ways, but directions might mean something



partylaw government election court president elected council general minister political national members committee united office federal member house parliament vote public elections democratic vote **son** died his **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 east edward english **school** students university high college schools education year program student campus community programs training center members science national years public academic association officers arts educational inside class middle department teachers colleges classes office activities companies district engineering learning founded faculty gifts sports children boys international board teaching academy secondary established

york county american united city washington john texas served virginia pennsylvania war moved ohio chicago william george james died army centuries dynasty **season** team **game** league games **species** family birds small long large animals bird plants genus giant natural habitat tree fern tropical white black order leaves brown common keeps three arches flowers eggs worldwide feed ocean southwestern wild length male breeding habitat range food female full short means endemic forest group including include moist threatened tall

centuryking roman empire greek **engine** car production built engines speed vehicles designed produced power front system version type series motor rear standard gun company introduced range ford sold five wheel tank fleet factory machine developed based raised wheels time powered small high weight electric body **art** museum work works artists collection design arts painting artist gallery paintings exhibition style **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 farm worn dogs cut popular left generally traditional tall front horses shape tail feet color line coat three specially modern face cross

album band **radio** station news television **channel** broadcast stations network media tv broadcasting time format local program bbc programming live in morning host began sports live or cable call hosted comedian music pre recording daily channels digital also aired changed current broadcast communications programme day broadcasts moved obs week assembly late night

age 18 population income average years median living 65 males females households 100 family people families older town size city household miles density **music** musical opera festival orchestra dance performance works concert studio orchestra symphony performances instruments musicians classical including work composed major a cappella songs folk instrument ballad complexer composers play performing composers playing stage years inside popular choir ensemble sound style time wide tall three chamber recordings string

# Poll Everywhere

Think 

1 min

Which of the following are true about matrix factorization for recommendation systems?

- A. Provides personalization
- B. Captures context (e.g. time of day)
- C. Solves the cold start problem

[pollev.com/cse416](https://pollev.com/cse416)



1:00

# Poll Everywhere

Pair 

2 min

Which of the following are true about matrix factorization for recommendation systems?

- A. Provides personalization
- B. Captures context (e.g. time of day)
- C. Solves the cold start problem

[pollev.com/cse416](https://pollev.com/cse416)

**2:00**



## Blending Models: Featurized Matrix Factorization

*Final Solution*

# Cold Start Again

Consider a new user  $u'$  and we want to predict their ratings

No previous ratings for them so:  $\forall_v r_{u'v} = ?$

Objective

$$\hat{L}, \hat{R} = \operatorname{argmin}_{L, R} \sum_{u, v: r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2 + \lambda_U \|L\|_F^2 + \lambda_V \|R\|_F^2$$

Optimal user factor:  $L_{u'} = 0$  because there is only penalty

Therefore,  $\forall_v \hat{r}_{u'v} = 0$  which seems like a problem

# Blend Models

**Idea:** Learn a classification model to supplement the matrix factorization model!

Create a feature vector for each movie

Define weights on these features for **all users**

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

Fit linear model

## Add Personalization

Of course, not all users have same preferences.

Include a user-specific deviation from global model

Can also add user specific features to model

# 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**

## Evaluating Recommendations

# Accuracy?

Could we use classification accuracy to identify which recommender system is performing best?

- We don't really care to predict what a person **does not** like
- Instead, we want to find the relatively few items from the catalog that they will like
- Sort of a class imbalance

Instead, we want to look at our set of recommendations and ask:

- How many of our recommendations did the user like?
- How many of the items that the user liked did we recommend?

Sound familiar?

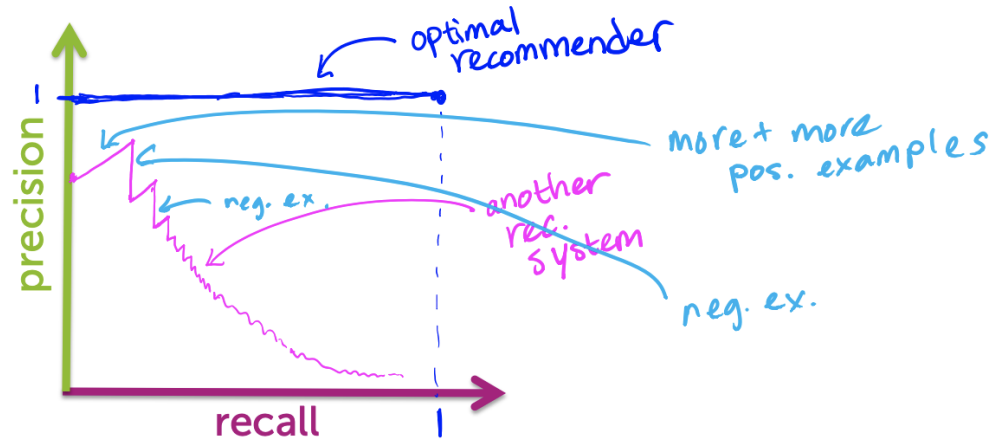
# Precision - Recall

Precision and recall for recommender systems

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

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

For a given recommender system, plot precision and recall for different number of recommended items





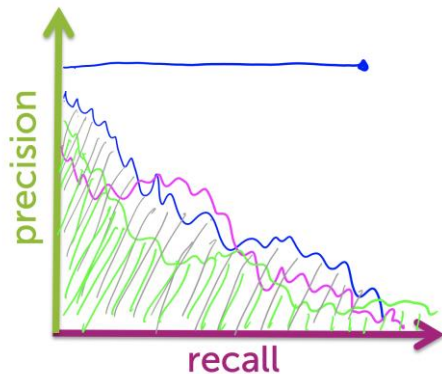
# Which Algorithm is Best?

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 can:

- Describe the goal of a recommender system
- Provide examples of applications where recommender systems are useful
- Implement a co-occurrence based recommender system
- 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
- Exploit estimated “topic” vectors to make recommendations
- 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