

# CSE 190 – Lecture 8

Data Mining and Predictive Analytics

Recommender Systems

# Why recommendation?

The goal of recommender systems is...

- To help people discover new content

Recommendations for You in Amazon Instant Video [See more](#)



# Why recommendation?

- The goal of recommender systems is...
- To help us find the content we were already looking for



Harry Potter and the Sorcerer's Stone 2001 PG-13 CC

★★★★★ 2,302 IMDb 7.5/10

Based on the wildly popular J.K. Rowling's book about a young boy who on his eleventh birthday discovers, he is the orphaned boy of two powerful wizards and has unique magical powers.

Starring: Richard Harris, Maggie Smith  
Runtime: 2 hours, 33 minutes  
Available to watch on supported devices.

Are these recommendations good or bad?

Share [email] [facebook] [twitter] [pinterest]

amazon

Rent or Buy

Rent HD \$3.99

\$12.99

em a gift card

Add to V

By placing your order, you agree to our [Terms of Use](#). Sold by Amazon Digital Services, Inc.

Customers Who Watched This Item Also Watched



# Why recommendation?

The goal of recommender systems is...

- To discover which things go together



Calvin Klein Men's Relaxed Straight Leg Jean In Cove

★★★★☆ - 20 customer reviews

Price: \$48.16 - \$69.99 & FREE Returns. Details

Size:

Select [Sizing info](#) | [Fit: As expected \(55%\)](#)

Color: Cove

- 98% Cotton/2% Elastane
- Imported
- Button closure
- Machine Wash
- Relaxed straight-leg jean in light-tone denim featuring whiskering and five-pocket styling
- Zip fly with button
- 10.25-inch front rise, 19-inch knee, 17.5-inch leg opening

Frequently Bought Together



Calvin Klein Jeans  
\$57.94 - \$69.50



Calvin Klein Jeans  
\$49.92



Calvin Klein Jeans  
\$50.67 - \$69.99



Levi's  
\$23.99 - \$68.00

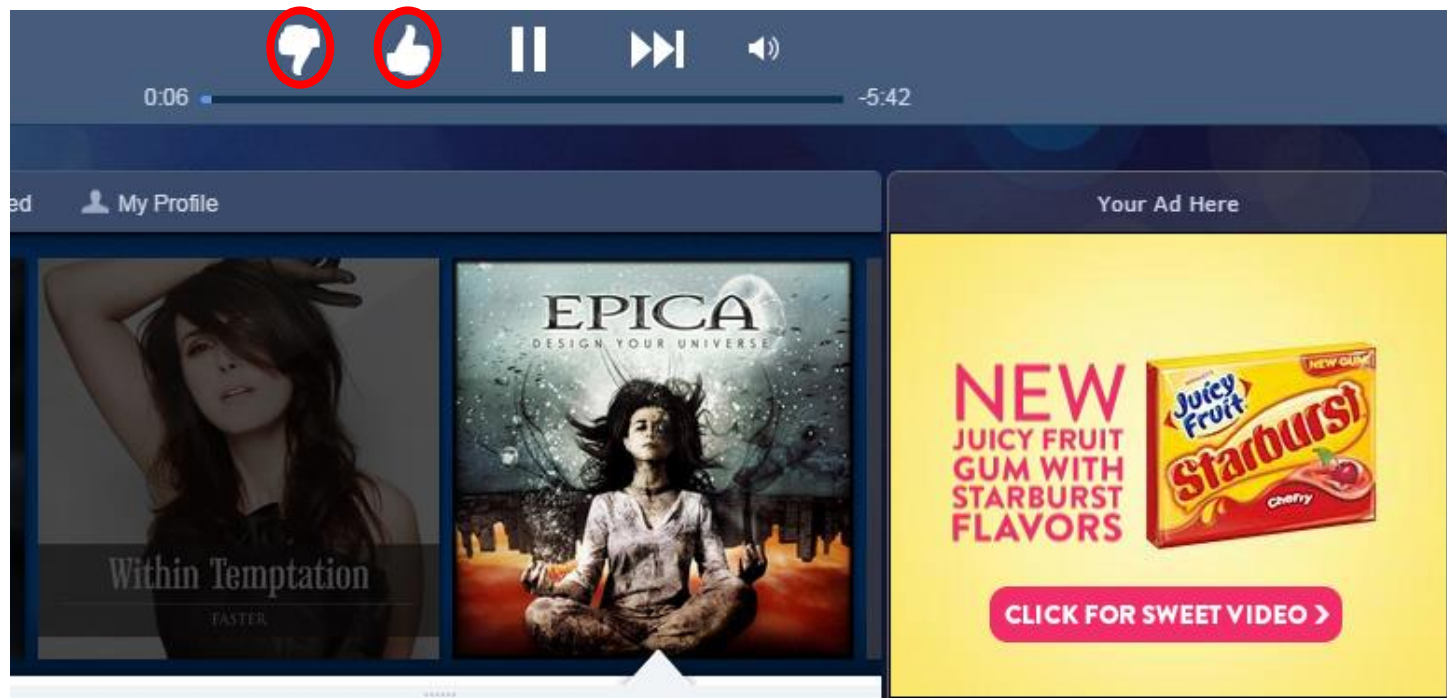
Customers Who Bought This Item Also Bought



# Why recommendation?

The goal of recommender systems is...

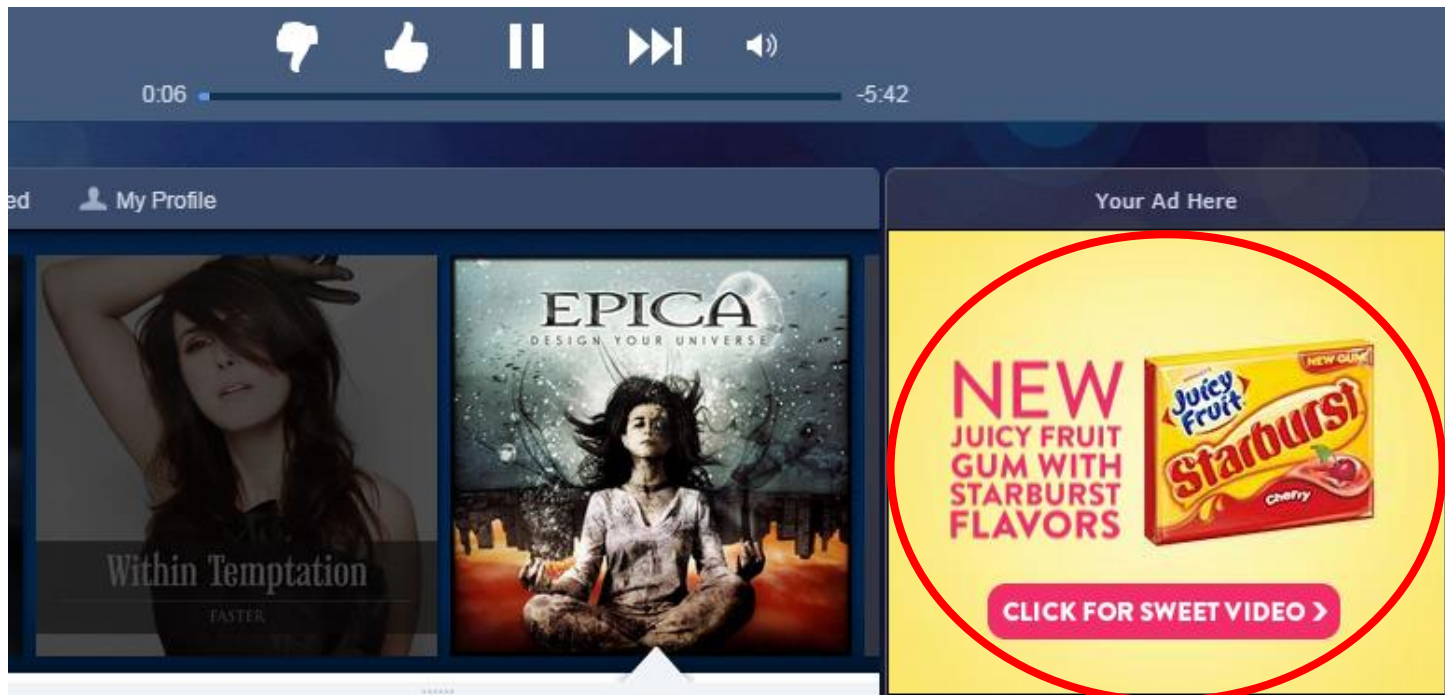
- To personalize user experiences in response to user feedback



# Why recommendation?

The goal of recommender systems is...

- To recommend incredible products that are relevant to our interests



The image shows a screenshot of a video player interface. At the top, there is a video player with a progress bar showing 0:06 out of -5:42. Below the video player, there are two main sections: a profile section on the left and an advertisement on the right. The profile section is titled "My Profile" and features two video thumbnails. The first thumbnail shows a woman with long dark hair, and the second thumbnail is for "EPICA DESIGN YOUR UNIVERSE" featuring a woman in a meditative pose. The advertisement on the right is titled "Your Ad Here" and features a yellow background with a red circle around the central content. The content includes the text "NEW JUICY FRUIT GUM WITH STARBURST FLAVORS" and an image of a box of Juicy Fruit Starburst gum. Below the image is a red button that says "CLICK FOR SWEET VIDEO >".

# Why recommendation?

The goal of recommender systems is...

- To identify things that we **like**

Results for 'mad max'



**Add**

★★★★☆

**Mad Max**  
1979 **R** 93 minutes

In a postapocalyptic future, jaded motorcycle cop Max Rockatansky is ready to retire. But his world is shattered when a malicious gang murders his family as an act of retaliation, forcing a devastated Max to hit the open road seeking vengeance.

**Starring:** Mel Gibson, Hugh Keays-Byrne  
**Director:** George Miller  
**Genre:** Sci-Fi & Fantasy  
**Format:** DVD

★★★★☆ **3.6** Our best guess for Jeremy

# Why recommendation?

The goal of recommender systems is...

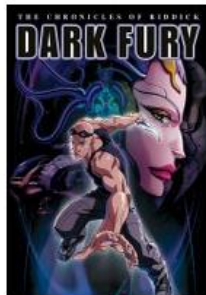
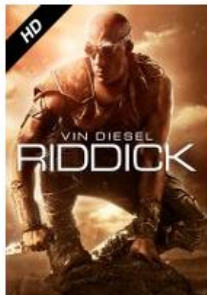
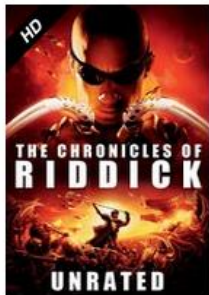
- To help people discover new content
- To help us find the content we were
- To discover preferences, opinions, together
- To provide recommendations in response to user feedback
- To identify things that we **like**

To **model** people's preferences, opinions, and behavior

# Recommending things to people


Suppose we want to build a movie recommender

e.g. which of these films will I rate highest?



# Recommending things to people

We already have a few tools in our "supervised learning" toolbox that may help us



**Pitch Black - Unrated Director's Cut** R CC

★★★★★ 777 **IMDb** 7.1/10

Watch Trailer

When their ship crash-lands on a remote planet, the marooned passengers soon learn that escaped convict Riddick (Vin Diesel) isn't the only thing they have to fear. Deadly creatures lurk in the shadows, waiting to attack in the dark, and the planet is rapidly plunging into the

See More

Starring: Vin Diesel, Radha Mitchell  
Runtime: 1 hour, 53 minutes  
Available to watch on supported devices.

**Product Details**

Genres	Science Fiction, Action, Horror
Director	David Twohy
Starring	Vin Diesel, Radha Mitchell
Supporting actors	Cole Hauser, Keith David, Lewis Fitz-Gerald, Claudia Black, Rhiana Granger, Angela Moore, Peter Chiang, Ken Twohy
Studio	NBC Universal
MPAA Rating	R (Restricted)
Captions and subtitles	English <a href="#">Details</a>
Rental rights	24 hour viewing period. <a href="#">Details</a>
Purchase rights	Stream instantly and download to 2 locations <a href="#">Details</a>
Format	Amazon Instant Video (streaming online video and digital download)

**A. Phillips**  
Reviewer ranking: #17,230,554  
**90% helpful**  
votes received on reviews (151 of 167)

**ABOUT ME**  
Enjoy the reviews...

**ACTIVITIES**  
[Reviews \(16\)](#)  
[Public Wish List \(2\)](#)  
[Listmania Lists \(2\)](#)  
[Tagged Items \(1\)](#)

$f(\text{user features, movie features}) \xrightarrow{?} \text{star rating}$

# Recommending things to people

$f(\text{user features, movie features}) \xrightarrow{?} \text{star rating}$

Movie features: genre, actors, rating, length, etc.

## Product Details

Genres	<a href="#">Science Fiction</a> , <a href="#">Action</a> , <a href="#">Horror</a>
Director	<a href="#">David Twohy</a>
Starring	<a href="#">Vin Diesel</a> , <a href="#">Radha Mitchell</a>
Supporting actors	<a href="#">Cole Hauser</a> , <a href="#">Keith David</a> , <a href="#">Lewis Fitz-Gerald</a> , <a href="#">Claudia Black</a> , <a href="#">Rhiana Gr Angela Moore</a> , <a href="#">Peter Chiang</a> , <a href="#">Ken Twohy</a>
Studio	NBC Universal
MPAA rating	R (Restricted)
Captions and subtitles	English <a href="#">Details</a> ▾
Rental rights	24 hour viewing period. <a href="#">Details</a> ▾
Purchase rights	Stream instantly and download to 2 locations <a href="#">Details</a> ▾
Format	Amazon Instant Video (streaming online video and digital download)

User features: age, gender, location, etc.

## A. Phillips

Reviewer ranking: #17,230,554

**90%** helpful

votes received on reviews  
(151 of 167)

## ABOUT ME

Enjoy the reviews...

## ACTIVITIES

[Reviews](#) (16)

[Public Wish List](#) (2)

[Listmania Lists](#) (2)

[Tagged Items](#) (1)

# Recommending things to people

$f(\text{user features, movie features}) \xrightarrow{?}$  star rating

With the models we've seen so far, we can build predictors that account for...


- Do women give higher ratings than men?
- Do Americans give higher ratings than Australians?
- Do people give higher ratings to action movies?
- Are ratings higher in the summer or winter?
- Do people give high ratings to movies with Vin Diesel?

So what **can't** we do yet?

# Recommending things to people

$f(\text{user features}, \text{movie features}) \xrightarrow{?}$  star rating

Consider the following linear predictor  
(e.g. from week 1):

$$\begin{aligned} f(\text{user features}, \text{movie features}) &= \\ &\langle \phi(\text{user features}); \phi(\text{movie features}), \theta \rangle \\ &= \langle \phi(\text{user features}), \theta_{\text{user}} \rangle + \langle \phi(\text{movie features}), \theta_{\text{movie}} \rangle \end{aligned}$$


# Recommending things to people

But this is essentially just two separate predictors!

$$f(\text{user features}, \text{movie features}) =$$
$$= \underbrace{\langle \phi(\text{user features}), \theta_{\text{user}} \rangle}_{\text{user predictor}} + \underbrace{\langle \phi(\text{movie features}), \theta_{\text{movie}} \rangle}_{\text{movie predictor}}$$

That is, we're treating user and movie features as though they're **independent**

# Recommending things to people

But these predictors should (obviously?)  
**not** be independent

$$f(\text{user features, movie features}) = f(\text{user}) + f(\text{movie})$$

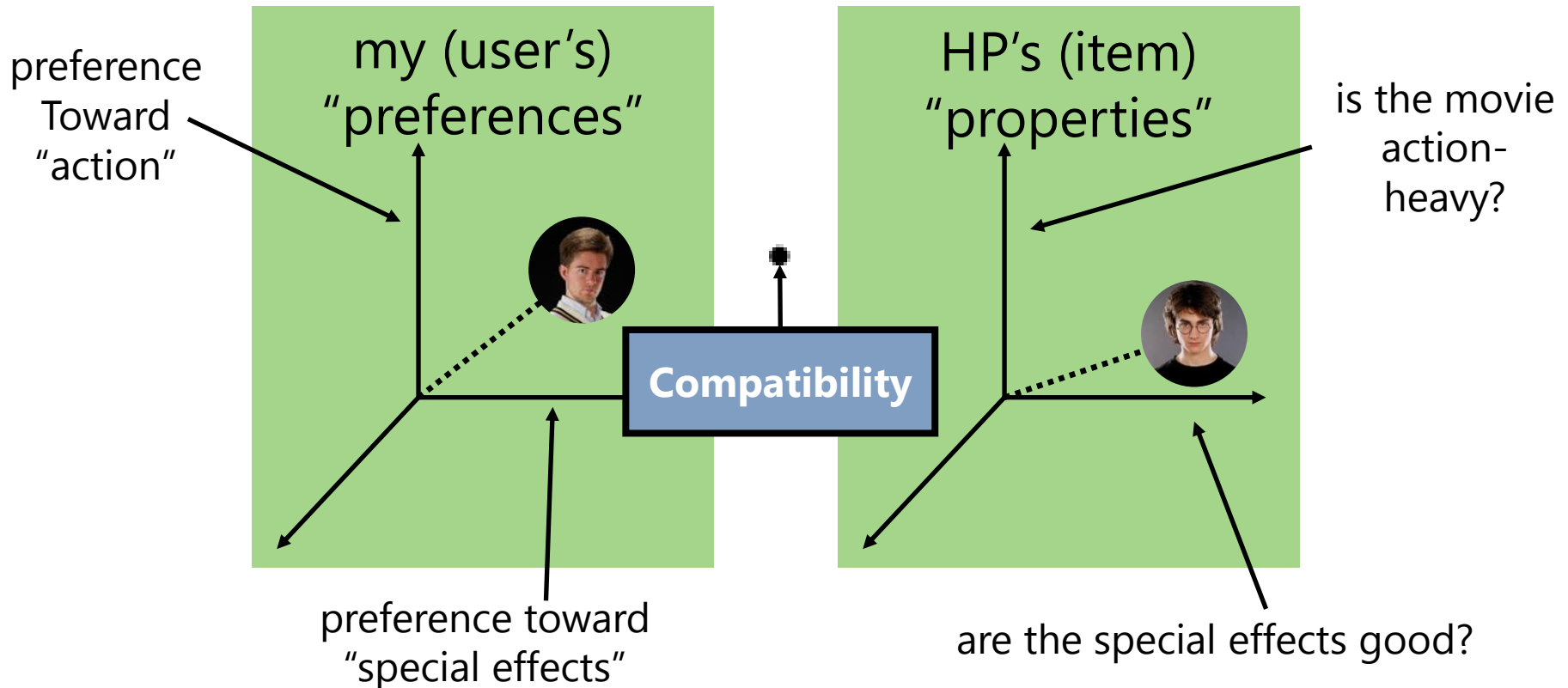
do I tend to give high ratings?

does the population tend to give high ratings to this genre of movie?

But what about a feature like "do **I** give high ratings to **this genre** of movie"?

# Recommending things to people

**Recommender Systems** go beyond the methods we've seen so far by trying to model the **relationships** between people and the items they're evaluating



Today

## **Recommender Systems**

### **1. Collaborative filtering**

(performs recommendation in terms of user/user and item/item similarity)

### **2. Assignment 1**

### **3. (next week?) Latent-factor models**

(performs recommendation by projecting users and items into some low-dimensional space)

### **4. (next week) The Netflix Prize**

# Defining similarity between users & items

**Q:** How can we measure the **similarity** between two **users**?

**A:** In terms of the **items** they purchased!

**Q:** How can we measure the similarity between two **items**?

**A:** In terms of the users who purchased them!

# Defining similarity between users & items

e.g.:  
Amazon



## Calvin Klein Men's Relaxed Straight Leg Jean In Cove

★★★★☆ 20 customer reviews

Price: \$48.16 - \$69.99 & FREE Returns. Details

Size:

Select Sizing info | Fit: As expected (55%)

Color: Cove

- 98% Cotton/2% Elastane
- Imported
- Button closure
- Machine Wash
- Relaxed straight-leg jean in light-tone denim featuring whiskering and five-pocket styling
- Zip fly with button
- 10.25-inch front rise, 19-inch knee, 17.5-inch leg opening

### Frequently Bought Together



Calvin Klein Jeans  
\$57.94 - \$69.50



Calvin Klein Jeans  
\$49.92



Calvin Klein Jeans  
\$50.67 - \$69.99



Levi's  
\$23.99 - \$68.00

### Customers Who Viewed This Item Also Viewed



### Customers Who Bought This Item Also Bought



# Definitions

## Definitions

$I_u$  = set of items purchased by user  $u$

$U_i$  = set of users who purchased item  $i$

# Definitions

Or equivalently...

$$R = \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{pmatrix}$$

items

users

$R_u$  = binary representation items purchased by  $u$

$R_{.,i}$  = binary representation of users who purchased  $i$

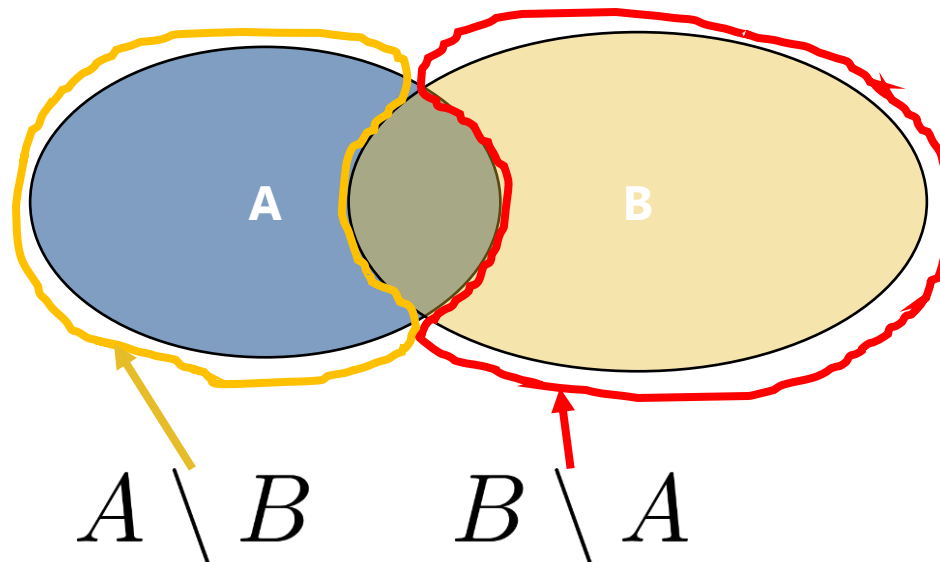
$$I_u = \{i | R_{u,i} = 1\} \quad U_i = \{u | R_{u,i} = 1\}$$

# 0. Euclidean distance

## Euclidean distance:

e.g. between two items  $i, j$  (similarly defined between two users)

$$|U_i \setminus U_j| + |U_j \setminus U_i| = \|R_i - R_j\|$$



# 0. Euclidean distance

## Euclidean distance:

$$\begin{aligned} \text{e.g.: } U_1 &= \{1,4,8,9,11,23,25,34\} \\ U_2 &= \{1,4,6,8,9,11,23,25,34,35,38\} \\ U_3 &= \{4\} \\ U_4 &= \{5\} \end{aligned}$$

$$|U_1 \setminus U_2| + |U_2 \setminus U_1| = 3$$

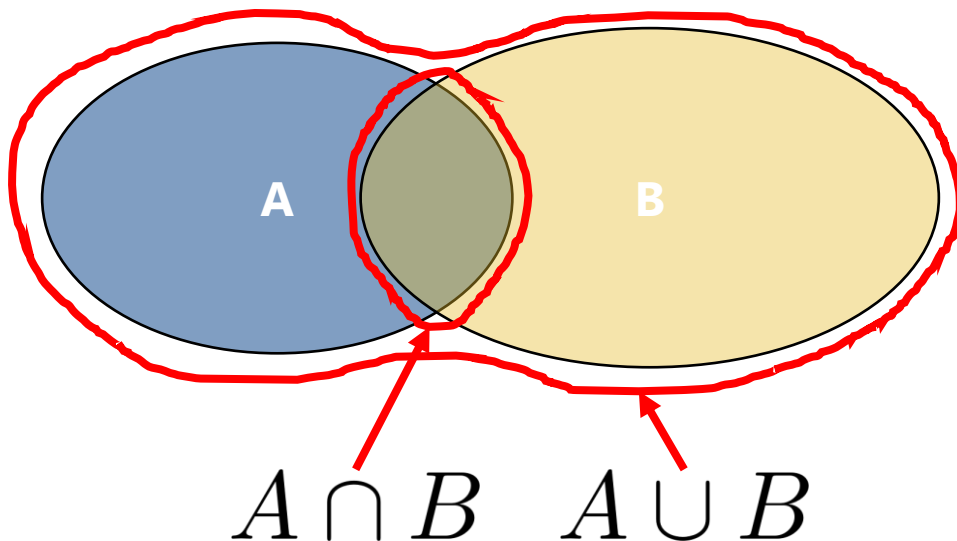
$$|U_3 \setminus U_4| + |U_4 \setminus U_3| = 2$$

**Problem:** favors small sets, even if they have few elements in common

# 1. Jaccard similarity

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$\text{Jaccard}(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$



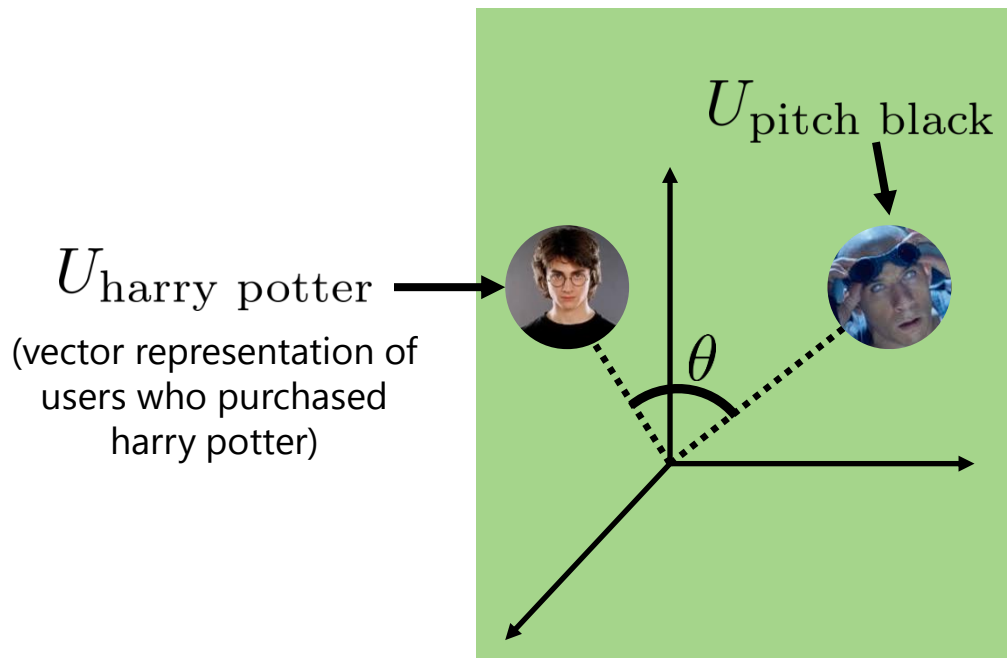
→ Maximum of 1 if the two users purchased **exactly the same** set of items  
(or if two items were purchased by the same set of users)

→ Minimum of 0 if the two users purchased **completely disjoint** sets of items  
(or if the two items were purchased by completely disjoint sets of users)

## 2. Cosine similarity

$$\text{Cosine}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

$$\theta = \cos^{-1} \left( \frac{A \cdot B}{\|A\| \|B\|} \right)$$



$$\cos(\theta) = 1$$

(theta = 0)  $\rightarrow$   $A$  and  $B$  point in exactly the same direction

$$\cos(\theta) = -1$$

(theta = 180)  $\rightarrow$   $A$  and  $B$  point in opposite directions (won't actually happen for 0/1 vectors)

$$\cos(\theta) = 0$$

(theta = 90)  $\rightarrow$   $A$  and  $B$  are orthogonal

## 2. Cosine similarity

### Why cosine?

- Unlike Jaccard, works for arbitrary vectors
- E.g. what if we have **opinions** in addition to purchases?

$$R = \begin{pmatrix} 1 & 0 & \dots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \dots & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} -1 & 0 & \dots & 1 \\ 0 & 0 & & -1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \dots & -1 \end{pmatrix}$$

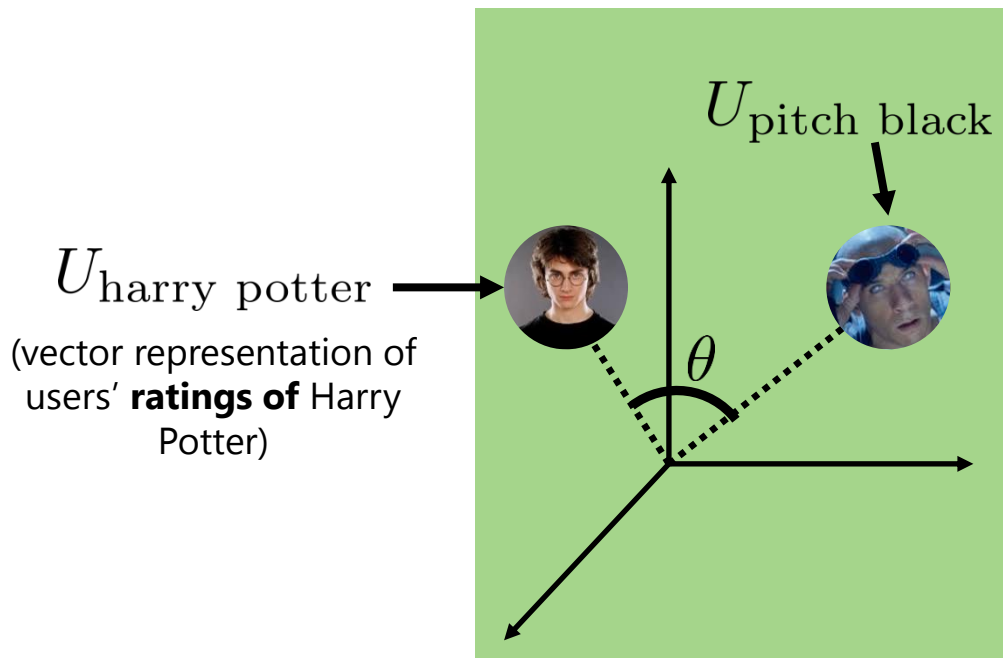
bought and **liked**

didn't buy

bought and **hated**

## 2. Cosine similarity

E.g. our previous example, now with “thumbs-up/thumbs-down” ratings



$$\cos(\theta) = 1$$

(theta = 0) → Rated by the same users, and they all agree

$$\cos(\theta) = -1$$

(theta = 180) → Rated by the same users, but they **completely disagree** about it

$$\cos(\theta) = 0$$

(theta = 90) → Rated by different sets of users

## 4. Pearson correlation

What if we have numerical ratings  
(rather than just thumbs-up/down)?

$$R = \begin{pmatrix} -1 & 0 & \dots & 1 \\ 0 & 0 & & -1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \dots & -1 \end{pmatrix} \longrightarrow \begin{pmatrix} 4 & 0 & \dots & 2 \\ 0 & 0 & & 3 \\ \vdots & & \ddots & \vdots \\ 5 & 0 & \dots & 1 \end{pmatrix}$$

bought and **liked**      didn't buy      bought and **hated**

## 4. Pearson correlation

What if we have numerical ratings (rather than just thumbs-up/down)?

- We wouldn't want 1-star ratings to be parallel to 5-star ratings
- So we can subtract the average – values are then **negative** for below-average ratings and **positive** for above-average ratings

$$\text{Sim}(u, v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R}_v)^2}}$$

items rated by both users      average rating by user v

## 4. Pearson correlation

Compare to the cosine similarity:

Pearson similarity (between users):

items rated by both users      average rating by user  $v$

$$\text{Sim}(u, v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R}_v)^2}}$$

Cosine similarity (between users):

$$\text{Sim}(u, v) = \frac{\sum_{i \in I_u \cap I_v} R_{u,i} R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} R_{u,i}^2 \sum_{i \in I_u \cap I_v} R_{v,i}^2}}$$

# Collaborative filtering in practice

How did Amazon generate their ground-truth data?

Given a product:



Let  $U_i$  be the set of users who viewed it

Rank products according to:  $\frac{|U_i \cap U_j|}{|U_i \cup U_j|}$  (or cosine/pearson)



# Collaborative filtering in practice

**Note:** (surprisingly) that we built something pretty useful out of **nothing but rating data** – we didn't look at any features of the products whatsoever

# Collaborative filtering in practice

**But:** we still have  
a few problems left to address...

1. This is actually kind of slow given a huge enough dataset – if one user purchases one item, this will change the rankings of **every other item that was purchased by at least one user in common**
2. Of no use for **new users** and **new items** (“cold-start” problems)
3. Won’t necessarily encourage diverse results

# Questions

# CSE 190 – Lecture 8

Data Mining and Predictive Analytics

Latent-factor models

# Latent factor models

So far we've looked at approaches that try to define some definition of user/user and item/item **similarity**

**Recommendation** then consists of

- Finding an item  $i$  that a user likes (gives a high rating)
- Recommending items that are similar to it (i.e., items  $j$  with a similar rating profile to  $i$ )

# Latent factor models

What we've seen so far are **unsupervised** approaches and whether the work depends highly on whether we chose a "good" notion of similarity

So, can we perform recommendations via **supervised** learning?

# Latent factor models

e.g. if we can model

$f(\text{user features, movie features}) \rightarrow \text{star rating}$

Then recommendation  
will consist of identifying

$$\textit{recommendation}(u) = \arg \max_{i \in \text{unseen items}} f(u, i)$$

# The Netflix prize


In 2006, Netflix created a dataset of **100,000,000** movie ratings  
Data looked like:

(userID, itemID, time, rating)

The goal was to reduce the (R)MSE at predicting ratings:

$$\text{RMSE}(f) = \sqrt{\frac{1}{N} \sum_{u,i,t \in \text{test set}} (f(u, i, t) - r_{u,i,t})^2}$$

model's prediction                      ground-truth



Whoever first manages to reduce the RMSE by **10%** versus  
Netflix's solution wins **\$1,000,000**

# The Netflix prize

This led to **a lot** of research on rating prediction by minimizing the Mean-Squared Error

**NETFLIX**

(it also led to a lawsuit against Netflix, once somebody managed to de-anonymize their data)

We'll look at a few of the main approaches

# Rating prediction

Let's start with the simplest possible model:

$$f(u, i) = \alpha$$

↑     ↑  
user  item

$$\alpha = \frac{1}{N} \sum_{u, i \in \text{training data}} r_{u, i}$$

Here the RMSE is just equal to the **standard deviation** of the data

(and we cannot do any better with a 0<sup>th</sup> order predictor)

# Rating prediction

What about the **2<sup>nd</sup>** simplest model?

$$f(u, i) = \alpha + \beta_u + \beta_i$$

user item

how much does  
this user tend to  
rate things above  
the mean?

does this item tend  
to receive higher  
ratings than others

e.g.

$$\alpha = 4.2$$



$$\beta_{\text{pitch black}} = -0.1$$

$$\beta_{\text{julian}} = -0.2$$



# Rating prediction

The optimization problem becomes:

$$\arg \min_{\alpha, \beta} \underbrace{\sum_{u,i} (\alpha + \beta_u + \beta_i - R_{u,i})^2}_{\text{error}} + \lambda \underbrace{[\sum_u \beta_u^2 + \sum_i \beta_i^2]}_{\text{regularizer}}$$

Jointly convex in  $\beta_i, \beta_u$ . Can be solved by iteratively removing the mean and solving for beta

# Rating prediction

Iterative procedure – repeat the following updates until convergence:

$$\alpha = \frac{\sum_{u,i \in \text{train}} (R_{u,i} - (\beta_u + \beta_i))}{N_{\text{train}}}$$

$$\beta_u = \frac{\sum_{i \in I_u} R_{u,i} - (\alpha + \beta_i)}{\lambda + |I_u|}$$

$$\beta_i = \frac{\sum_{u \in U_i} R_{u,i} - (\alpha + \beta_u)}{\lambda + |U_i|}$$

(exercise: write down derivatives and convince yourself of these update equations!)

# Rating prediction

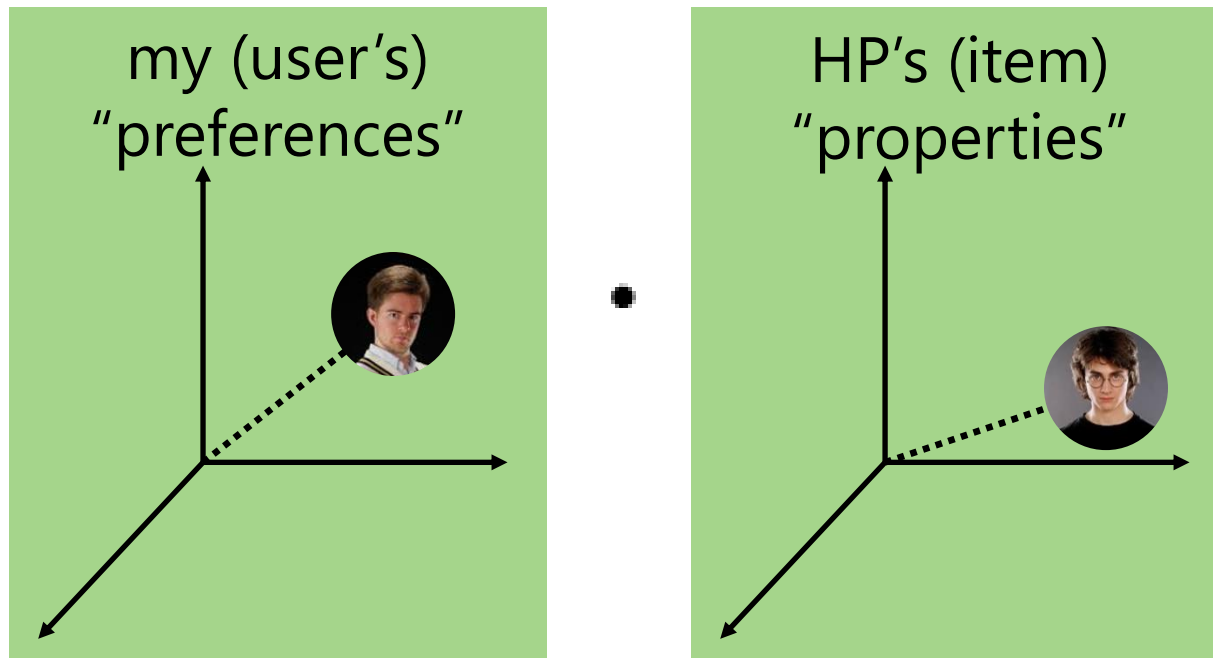
Looks good (and actually works surprisingly well), but doesn't solve the basic issue that we started with

$$\begin{aligned} f(\text{user features}, \text{movie features}) &= \\ &= \underbrace{\langle \phi(\text{user features}), \theta_{\text{user}} \rangle}_{\text{user predictor}} + \underbrace{\langle \phi(\text{movie features}), \theta_{\text{movie}} \rangle}_{\text{movie predictor}} \end{aligned}$$

That is, we're **still** fitting a function that treats users and items independently

# Recommending things to people

How about an approach based on **dimensionality reduction**?



i.e., let's come up with low-dimensional representations of the users and the items so as to best explain the data

# Dimensionality reduction

We already have some tools that ought to help us, e.g. from lecture 3:

$$R = \begin{pmatrix} 5 & 3 & \cdots & 1 \\ 4 & 2 & & 1 \\ 3 & 1 & & 3 \\ 2 & 2 & & 4 \\ 1 & 5 & & 2 \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \cdots & 1 \end{pmatrix}$$

What is the best low-rank approximation of  $R$  in terms of the mean-squared error?

# Dimensionality reduction

We already have some tools that ought to help us, e.g. from lecture 3:

$$R = \begin{pmatrix} 5 & 3 & \cdots & 1 \\ 4 & 2 & & 1 \\ 3 & 1 & & 3 \\ \text{Singular Value Decomposition} \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \cdots & 1 \end{pmatrix}$$

(square roots of)  
eigenvalues of  $RR^T$

$$R = U \Sigma V^T$$

eigenvectors of  $RR^T$

eigenvectors of  $R^T R$

The “best” rank-K approximation (in terms of the MSE) consists of taking the eigenvectors with the highest eigenvalues

# Dimensionality reduction

**But!** Our matrix of ratings is only partially observed; and it's **really big!**

$$R = \begin{pmatrix} 5 & 3 & \cdots & \cdot \\ 4 & 2 & & 1 \\ 3 & \cdot & & 3 \\ \cdot & 2 & & 4 \\ 1 & 5 & & \cdot \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \cdots & \cdot \end{pmatrix}$$

Missing ratings

SVD is **not defined** for partially observed matrices, and it is **not practical** for matrices with 1Mx1M+ dimensions

# Latent-factor models

Instead, let's solve approximately using gradient descent

$$R = \begin{pmatrix} 5 & 3 & \dots & \cdot \\ 4 & 2 & & 1 \\ 3 & \cdot & & 3 \\ \cdot & 2 & & 4 \\ 1 & 5 & & \cdot \\ \vdots & & \ddots & \vdots \\ 1 & 2 & \dots & \cdot \end{pmatrix} \left. \vphantom{\begin{pmatrix} 5 \\ 4 \\ 3 \\ \cdot \\ 1 \\ \vdots \\ 1 \end{pmatrix}} \right\} \text{users}$$

items

K-dimensional representation of each item

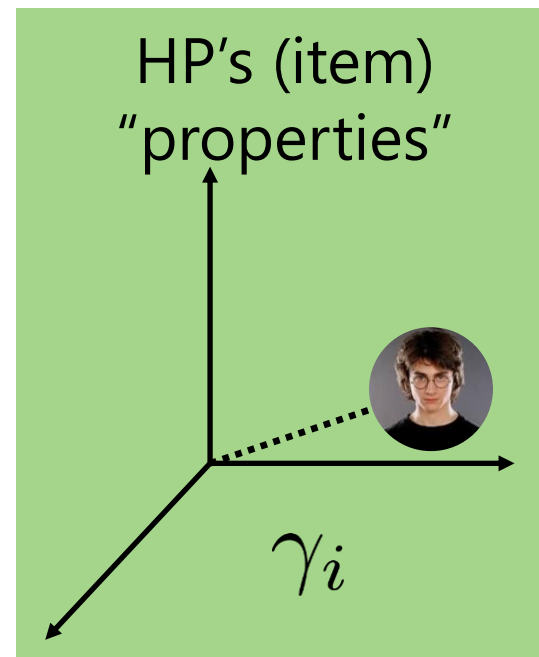
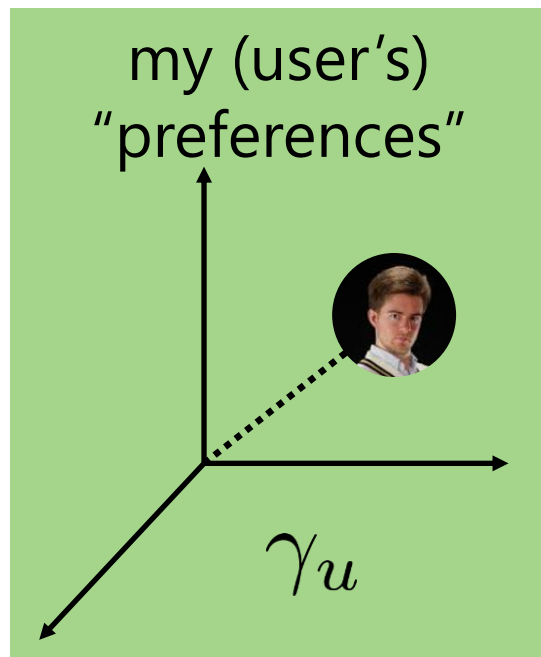
$$R \simeq UV^T$$

K-dimensional representation of each user

# Latent-factor models

Let's write this as:

$$f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$



•

# Latent-factor models


Let's write this as:

$$f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

Our optimization problem is then

$$\arg \min_{\alpha, \beta, \gamma} \underbrace{\sum_{u,i} (\alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i - R_{u,i})^2}_{\text{error}} + \lambda \underbrace{[\sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_i \|\gamma_i\|_2^2 + \sum_u \|\gamma_u\|_2^2]}_{\text{regularizer}}$$

**Problem:** this is certainly not convex

(proof is easy: (1) it is smooth; (2) permuting the columns of gamma preserves the objective; (3) therefore it has multiple local optima and cannot be convex; (4) in other words it must look like this: )

permutations of local minima

# Latent-factor models

Oh well. We'll just solve it approximately

Observation: if we know either the user or the item parameters, the problem becomes easy

$$f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

e.g. fix  $\gamma_i$  – pretend we're fitting parameters for features

# Latent-factor models

This gives rise to a simple (though approximate) solution

**objective:**

$$\arg \min_{\alpha, \beta, \gamma} \sum_{u,i} (\alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i - R_{u,i})^2 + \lambda [\sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_i \|\gamma_i\|_2^2 + \sum_u \|\gamma_u\|_2^2]$$

$$= \arg \min_{\alpha, \beta, \gamma} \text{objective}(\alpha, \beta, \gamma)$$

1) fix  $\gamma_i$ . Solve  $\arg \min_{\alpha, \beta, \gamma_u} \text{objective}(\alpha, \beta, \gamma)$

2) fix  $\gamma_u$ . Solve  $\arg \min_{\alpha, \beta, \gamma_i} \text{objective}(\alpha, \beta, \gamma)$

3,4,5...) repeat until convergence

Each of these subproblems is “easy” – just regularized least-squares, like we’ve been doing since week 1. This procedure is called **alternating least squares**.

# Latent-factor models

**Observation:** we went from a method which uses **only** features:

$f(\text{user features, movie features}) \rightarrow \text{star rating}$

**User features:**  
age, gender,  
location, etc.

**Movie features:** genre,  
actors, rating, length, etc.

Product Details	
Genres	Science Fiction, Action, Horror
Director	David Twohy
Starring	Vin Diesel, Radha Mitchell
Supporting actors	Cole Hauser, Keith David, Lewis Fitz-Gerald, Claudia Black, Rhiana Gr Angela Moore, Pieter Chang, Ken Twohy
Studio	NBC Universal
MPAA rating	R (Restricted)
Captions and subtitles	English Details ▾
Rental rights	24 hour viewing period Details ▾
Purchase rights	Stream instantly and download to 2 locations Details ▾
Format	Amazon Instant Video (streaming online video and digital download)

to one which **completely ignores** them:

$$\arg \min_{\alpha, \beta, \gamma} \sum_{u, i} (\alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i - R_{u, i})^2 + \lambda [\sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_i \|\gamma_i\|_2^2 + \sum_u \|\gamma_u\|_2^2]$$

# Latent-factor models

Should we use features or not?

1) Argument **against** features:

Imagine incorporating features into the model like:

$$f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i + \langle \phi(u), \theta_u \rangle + \langle \phi(i), \theta_i \rangle$$

which is equivalent to:

$$f(u, i) = \alpha + \beta_u + \beta_i + \underbrace{(\phi(u); \phi(i); \gamma_u)}_{\text{knowns}} \cdot \underbrace{(\theta_u; \theta_i; \gamma_i)}_{\text{unknowns}}$$

but this has fewer degrees of freedom than a model which replaces the knowns by unknowns:

$$f(u, i) = \alpha + \beta_u + \beta_i + (\gamma'_i; \gamma'_u; \gamma_u) \cdot (\theta_u; \theta_i; \gamma_i)$$

# Latent-factor models

Should we use features or not?

1) Argument **against** features:

So, the addition of features adds **no expressive power** to the model. We **could** have a feature like “is this an action movie?”, but if this feature were useful, the model would “discover” a latent dimension corresponding to action movies, and we wouldn’t need the feature anyway

**In the limit**, this argument is valid: as we add more ratings per user, and more ratings per item, the latent-factor model should automatically discover any useful dimensions of variation, so the influence of observed features will disappear

# Latent-factor models

Should we use features or not?

2) Argument **for** features:

But! Sometimes we **don't** have many ratings per user/item

Latent-factor models are next-to-useless if **either** the user or the item was never observed before

$$f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

reverts to zero if we've never seen the user before  
(because of the regularizer)

# Latent-factor models

Should we use features or not?

2) Argument **for** features:

This is known as the **cold-start** problem in recommender systems. Features are not useful if we have many observations about users/items, but are useful for **new** users and items.

We also need some way to handle users who are **active**, but don't necessarily rate anything, e.g. through **implicit feedback**

# Overview & recap

Tonight we've followed the programme below:

1. Measuring similarity between users/items for **binary** prediction (e.g. Jaccard similarity)
2. Measuring similarity between users/items for **real-valued** prediction (e.g. cosine/Pearson similarity)
3. Dimensionality reduction for **real-valued** prediction (latent-factor models)
4. **Finally** – dimensionality reduction for **binary** prediction

# One-class recommendation

How can we use **dimensionality reduction** to predict **binary outcomes**?

- In weeks 1&2 we saw **regression** and **logistic regression**. These two approaches use the same type of linear function to predict real-valued and binary outputs
- We can apply an analogous approach to binary recommendation tasks

# One-class recommendation

This is referred to as “**one-class**” recommendation

- In weeks 1&2 we saw **regression** and **logistic regression**. These two approaches use the same type of linear function to predict real-valued and binary outputs
- We can apply an analogous approach to binary recommendation tasks

# One-class recommendation

Suppose we have binary (0/1) observations (e.g. purchases) or positive/negative feedback (thumbs-up/down)

$$R = \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 0 & 0 & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & 0 & \cdots & 1 \end{pmatrix} \text{ or } \begin{pmatrix} -1 & ? & \cdots & 1 \\ ? & ? & & -1 \\ \vdots & & \ddots & \vdots \\ 1 & ? & \cdots & -1 \end{pmatrix}$$

purchased    didn't purchase    liked    didn't evaluate    didn't like

# One-class recommendation

So far, we've been fitting functions of the form

$$R \simeq UV^T$$

- Let's change this so that we maximize the **difference** in predictions between positive and negative items
- E.g. for a user who likes an item  $i$  and dislikes an item  $j$  we want to maximize:

$$\max \ln \sigma(\gamma_u \cdot \gamma_i - \gamma_u \cdot \gamma_j)$$

# One-class recommendation

We can think of this as maximizing the probability of correctly predicting pairwise preferences, i.e.,

$$p(i \text{ is preferred over } j) = \sigma(\gamma_u \cdot \gamma_i - \gamma_u \cdot \gamma_j)$$

- As with logistic regression, we can now maximize the likelihood associated with such a model by gradient ascent
  - In practice it isn't feasible to consider all pairs of positive/negative items, so we proceed by stochastic gradient ascent – i.e., randomly sample a (positive, negative) pair and update the model according to the gradient w.r.t. that pair

## Recap

1. Measuring similarity between users/items for **binary** prediction  
*Jaccard similarity*
2. Measuring similarity between users/items for **real-valued** prediction  
*cosine/Pearson similarity*
3. Dimensionality reduction for **real-valued** prediction  
*latent-factor models*
4. Dimensionality reduction for **binary** prediction  
*one-class recommender systems*

# Questions?

## Further reading:

One-class recommendation:

<http://goo.gl/08Rh59>

Amazon's solution to collaborative filtering at scale:

<http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf>

An (expensive) textbook about recommender systems:

<http://www.springer.com/computer/ai/book/978-0-387-85819-7>

Cold-start recommendation (e.g.):

<http://wanlab.poly.edu/recsys12/recsys/p115.pdf>