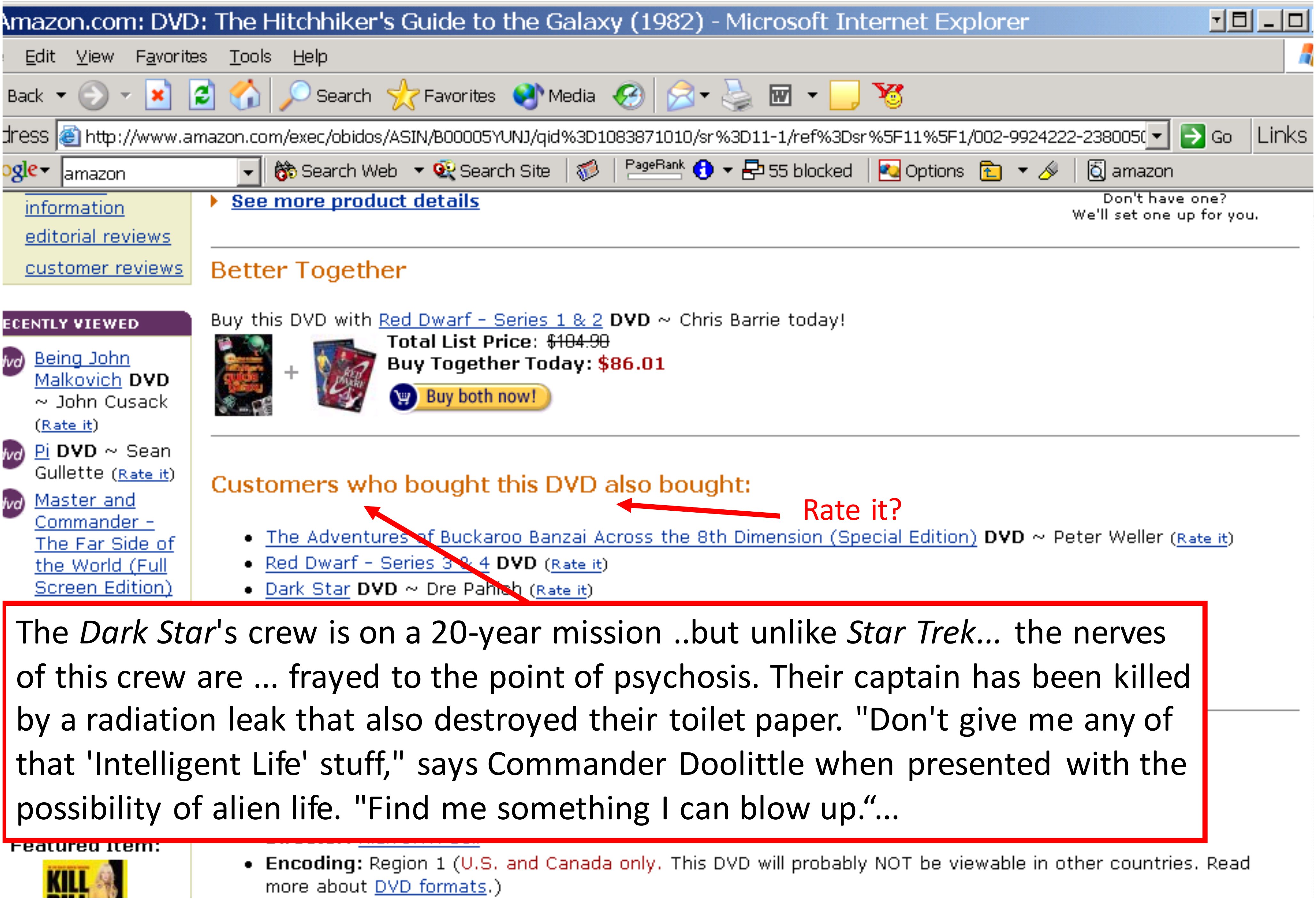
Recommender Systems

ECE 4200

Slides from various online resources …

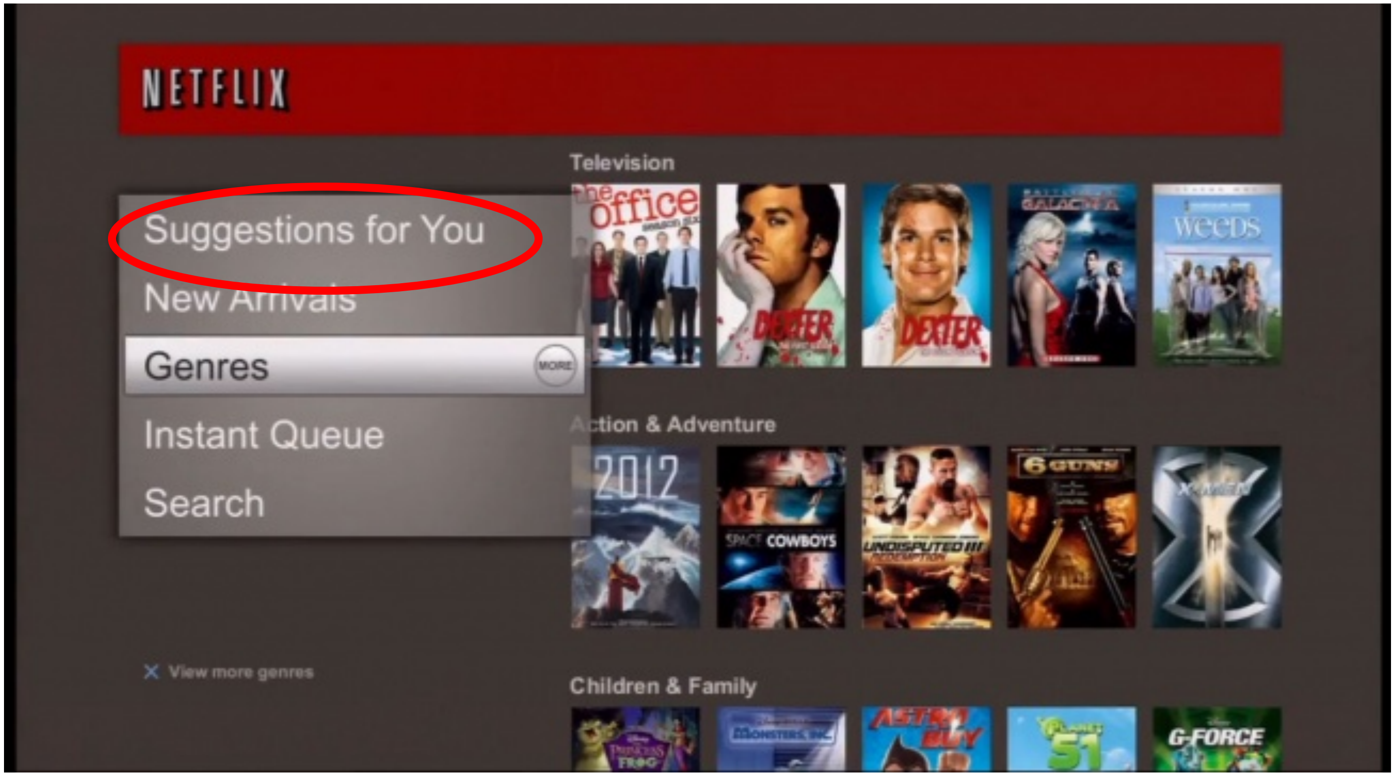
Everyday Examples …



The *Dark Star*'s crew is on a 20-year mission ..but unlike *Star Trek*… the nerves of this crew are … frayed to the point of psychosis. Their captain has been killed by a radiation leak that also destroyed their toilet paper. "Don't give me any of that 'Intelligent Life' stuff," says Commander Doolittle when presented with the possibility of alien life. "Find me something I can blow up.“…

Other Examples …??

Netflix Viewing Recommendations



Recommender Systems

DOMAIN: some field of activity where *users* buy, view, consume, or otherwise experience *items*

PROCESS:

1. *users* provide *ratings* on *items* they have experienced
2. Take all < *user*, *item*, *rating* > data and build a predictive model
3. For a *user* who hasn't experienced a particular *item*, use model to *predict* how well they will like it (i.e. *predict rating*)

Roles of Recommender Systems

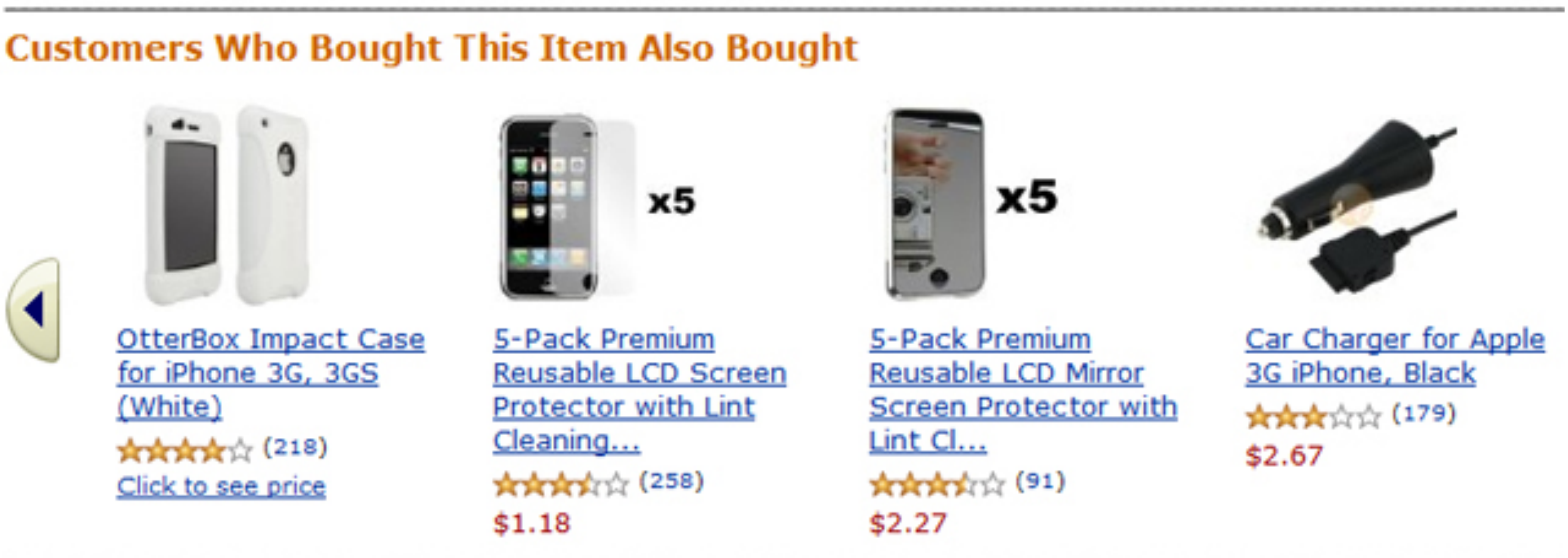
- Help users deal with *paradox of choice*

- Allow online sites to:
  - Increase likelihood of sales
  - Retain customers by providing positive search experience

- Considered essential in operation of:
  - Online retailing, e.g. Amazon, Netflix, etc.
  - Social networking sites

Amazon.com Product Recommendations

“If I have 3 million customers on the Web, I should have 3 million stores on the Web” *(1999)*



Social Network Recommendations

- Recommendations on essentially every category of interest known to mankind
    - Friends
    - Groups
    - Activities
    - Media (TV shows, movies, music, books)
    - News stories
    - Ad placements
- All based on connections in underlying social network graph and your expressed ‘likes’ and ‘dislikes’

Types of Recommender Systems

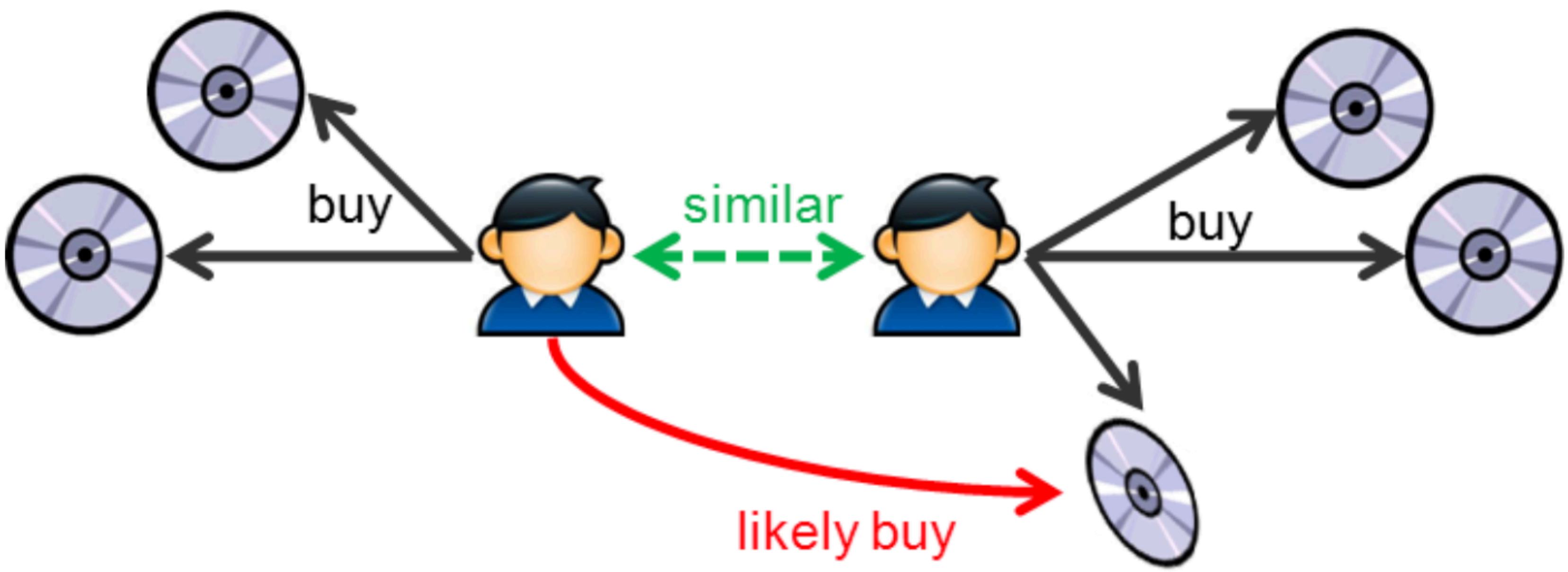
Base predictions on either:

- content-based approach
  - *explicit* characteristics of users and items
- collaborative filtering approach
  - *implicit* characteristics based on similarity of users' preferences to those of other users

Collaborative Filtering

- collaborative filtering approach
  - *implicit* characteristics based on similarity of users' preferences to those of other users
  - Come up with similarity measures between users (and perhaps items)
  - How much a user likes an item?
    - Find another user like this one,
    - See how much they like it.
  - Another way to do it??

Collaborative Filtering



Source: https://dzone.com/articles/recommendation-engine-models

As a matrix completion problem

Set of users $\{1, 2, \ldots, |U|\}$

Set of items $\{1, 2, \ldots, |I|\}$

r_{ui}: rating of user u for item i

	Star Wars	**Frozen 2**	**Avatar**	**Titanic**
A	5	4	5	2
B	2	2	?	4
C	5	?	5	2
D	5	4	?	1

Will D like Avatar?

- Segue to Netflix competition

How would you solve it?

Set of users $\{1, 2, \ldots, |U|\}$

Set of items $\{1, 2, \ldots, |I|\}$

r_{ui}: rating of user u for item i

	Star Wars	Frozen 2	Avatar	Titanic
A	5	4	5	2
B	2	2	?	4
C	5	?	5	2
D	5	4	?	1

Will D like Avatar?

k-NN user-user CF

To predict r_{ui}:

- Compute similarity between u and other users
- Find k users closest to u
- Output the *average* of these users for i

- Similar users vote similarly, and we can **learn** this similarity from a few ratings

k-NN item-item CF

To predict r_{ui}:

- Compute similarity between i and other items
- Find k items closest to i
- Output the *average* of these items by u

- Similar items will have similar ratings

User-user CF

How to decide similarity between users?

$\bar{r}_u$: vector or ratings of user u

$\bar{r}_{\mathrm{B}} = (2,2,?,4)$

	Star Wars	Frozen 2	Avatar	Titanic
A	5	4	5	2
B	2	2	?	4
C	5	?	5	2
D	5	4	?	1

Similarity between u and v

Cosine similarity:

$$s_{\mathbf{c}}(u, v) = \frac{\bar{r}_u \cdot \bar{r}_v}{\| \bar{r}_u \| \| \bar{r}_v \|}$$

Inverse Euclidean similarity:

$$s_{\mathbf{e}}(u, v) = \frac{1}{\| \bar{r}_u - \bar{r}_v \|}$$

Similarity between u and v

Some are inherently conservative than others

e.g., grading across different subjects

$\bar{r}_e = (4,2,5,4,1,2,3,5)$

$\bar{r}_f = (3,1,4,4,0,1,1,4)$

User e gives higher ratings than f but across users they are similar

$\hat{r}_u$: vector of means of u's ratings,

$\hat{r}_e = (3.25, 3.25, \dots, 3.25)$

$\hat{r}_e = (2.25, 2.25, \dots, 2.25)$

Similarity between u and v

Pearson similarity:

$$s_{\mathbf{p}}(u,v) = \frac{(\bar{r}_u - \hat{r}_u) \cdot (\bar{r}_v - \hat{r}_v)}{\| \bar{r}_u - \hat{r}_u \| \| \bar{r}_v - \hat{r}_v \|}$$

EXERCISES:

1. Fill values you like in the table above and derive the similarity between all pairs of users.
2. Derive the similarity between items for the same table.

Missing values user-user CF

Some values might be missing

$\bar{r}_A = (5,4,5,2)$

$\bar{r}_D = (5,4,?,1)$

- Ignore the missing values (replace by 0) and compute the similarity measures as before
- Need to only do it for users who actually have rated item i

How to choose k-NN and rate?

Some values might be missing

$\bar{r}_A = (5,4,5,2)$

$\bar{r}_D = (5,4,?,1)$

- Ignore the missing values (replace by 0) and compute the similarity measures as before
- Need to only do it for users who actually have rated item i

k-NN user-user CF

To predict r_{ui}:

- Compute similarity between u and other users
- Find k users closest to u among all who rated i
- Output the *weighted average* of these users for i

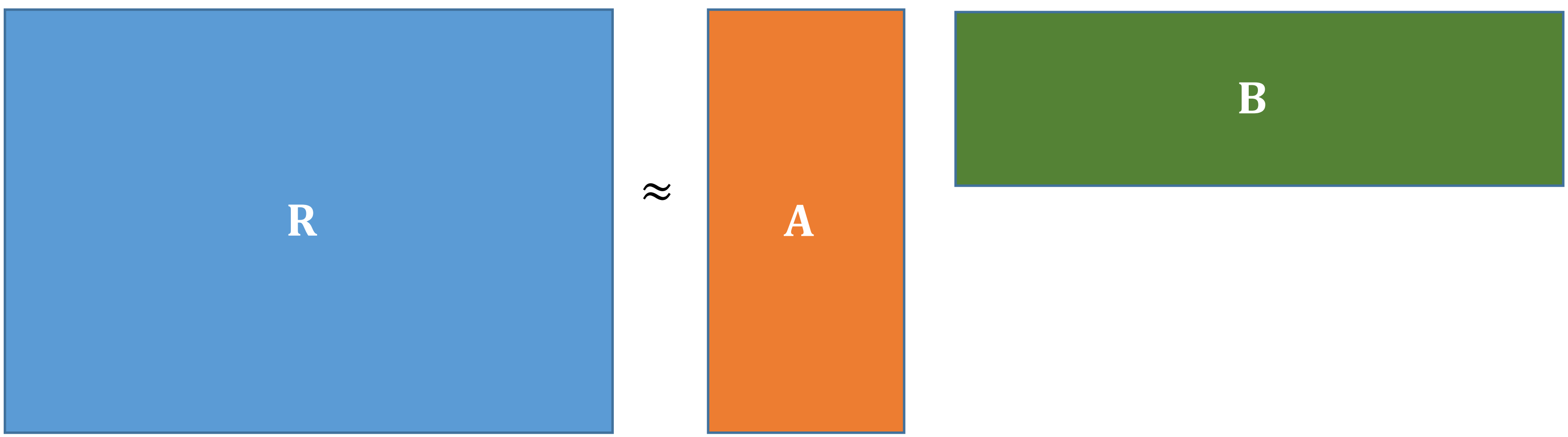
Let $u_1, \ldots, u_k$ be the k closest neighbors, then predict

$$\frac{\sum_{j=1} s_{\mathbf{c}}(u, u_j) \cdot r_{u_j i}}{\sum_{j=1} s_{\mathbf{c}}(u, u_j)}$$

Low dimensional matrix factorization

Let $\mathbf{R}$ be the ratings matrix, and $m = |U|, n = |I|$

Goal: $\mathbf{R} \approx \mathbf{AB}$, where $\mathbf{A}$ is $|U| \times k$, $\mathbf{B}$ is $k \times |I|$



What is k?

k: Latent variables describing items and users

Example: Movies and Viewers

(crime, politics, age, …):

uth row of $\mathbf{A}$: low-dim repn of user u's attributes

ith col of $\mathbf{B}$: low-dim repn of item i's attributes

$$\mathbf{R}_{ui} \approx \mathbf{A}_u \cdot \mathbf{B}_i$$

Low dimensional matrix factorization

Given $\mathbf{R}$ and k find $\mathbf{A}$, $\mathbf{B}$ *to*

$$\textbf{minimize} \quad \sum_{ui} (\mathbf{R}_{ui} - \mathbf{A}_u \cdot \mathbf{B}_i)^2$$

Minimize the mean-squared error.

Question: Write PCA in this form

Advantages: smaller dimensional problem

SVD

Any $m \times n$ real matrix $\mathbf{R}$ can be decomposed as:

$$\mathbf{R} = \mathbf{U\Sigma V}^{\mathrm{T}}$$

$\mathbf{U}$ is $m{\times}m$ orthonormal matrix

$\mathbf{V}$ is $n{\times}n$ orthonormal matrix

$\mathbf{\Sigma}$ is $m{\times}n$ diagonal matrix (sorted singular values)

Solution using **SVD:** $\mathbf{A} = \mathbf{U}_{m\times k}\mathbf{\Sigma}_{k\times k}$, $\mathbf{B} = \mathbf{V}_{n\times k}$

The PCA answer should be the same as this

What about missing values??

Let $\mathbf{Z}$ be a $\mathrm{m}\times n$ matrix with $\mathbf{Z}_{ui} = \mathrm{I}\{\mathbf{R}_{ui} \neq ?\}$

Given $\mathbf{R}$ and k find $\mathbf{A}$, $\mathbf{B}$ *to*

$$\mathbf{minimize}_{\mathbf{A},\ \mathbf{B}} \quad \sum_{ui} \mathbf{Z}_{ui}(\mathbf{R}_{ui} - \mathbf{A}_u \cdot \mathbf{B}_i)^2$$

Only penalize over the observed ratings

How to solve?

What about missing values??

Is a hard problem, svd no longer possible

Local minima might exist

Insight from k-means clustering

Iterate between cluster and cluster means

- If we know missing values in $\mathbf{R}$, we can apply SVD
- If we have $\mathbf{A}$, $\mathbf{B}$, we can fill missing entries

An iterative (EM) algorithm

- If we know missing values in $\mathbf{R}$, we can apply SVD
- If we have $\mathbf{A}$, $\mathbf{B}$, we can fill missing entries

Algorithm

- **E-**step: $\mathbf{X} = \mathbf{Z} * \mathbf{R} + (1 - \mathbf{W}) * \widehat{\mathbf{R}}$

Filling with best guess

- **M-**step: $\mathrm{SVD}(\mathbf{X}) = (\mathbf{U}, \mathbf{\Sigma}, \mathbf{V})$, $\widehat{\mathbf{R}} = \mathbf{U}_{m\times k}\mathbf{\Sigma}_{k\times k}\mathbf{V}_{n\times k}^{\mathrm{T}}$

See this as $\mathbf{A}\times\mathbf{B}$

Space: $O(mn)$, time per iteration: $O(mnk)$

Complete ratings may not fit … $mn = 8.5B$

Regularized squares objective

Given $\mathbf{R}$ and k find $\mathbf{A}$, $\mathbf{B}$:

$$\mathbf{min}_{\mathbf{A},\mathbf{B}} \sum_{ui} \mathbf{Z}_{ui} (\mathbf{R}_{ui} - \mathbf{A}_u \cdot \mathbf{B}_i)^2 + \lambda\left(\sum_u \| \mathbf{A}_u \|^2 + \sum_u \| \mathbf{B}_i \|^2 \right)$$

Fights overfitting with regularization

- Chooses small norm of $\mathbf{A}$, $\mathbf{B}$

Good generalization performance

Key: If you know $\mathbf{A}$, you can find best $\mathbf{B}$ (ridge regression)

ALS algorithm

- Alternate between $\mathbf{A}$ and $\mathbf{B}$

- Fix $\mathbf{A}$ find best $\mathbf{B}$
- Fix $\mathbf{B}$ find best $\mathbf{A}$

Algorithm: Alternate:

- For each user

$$\mathbf{A}_u = \left(\sum_{i:\mathbf{Z}_{ui}=1} \mathbf{B}_i \mathbf{B}_i^{\mathrm{T}} - \lambda \mathbf{I}\right)^{-1} \left(\sum_{i:\mathbf{Z}_{ui}=1} \mathbf{R}_{ui} \mathbf{B}_i\right)$$

- For each item

$$\mathbf{B}_i = \left(\sum_{u:\mathbf{Z}_{ui}=1} \mathbf{A}_u \mathbf{A}_u^{\mathrm{T}} - \lambda \mathbf{I}\right)^{-1} \left(\sum_{u:\mathbf{Z}_{ui}=1} \mathbf{R}_{ui} \mathbf{A}_u\right)$$

ALS algorithm

Time: $O(mk + nk)$ space, $O(mk^3 + nk^3)$ time per step

Can be done in parallel across the users and items

No need to store entire matrix

A Gradient Descent algorithm can make dependence on k to linear.

Recap