Recommendation Systems

CSE545 - Spring 2022

Stony Brook University

H. Andrew Schwartz

Big Data Analytics, The Class

Goal: Generalizations
A *model* or *summarization* of the data.

Data Workflow Frameworks

- Hadoop File System ✓
- Spark ✓
- Streaming ✓
- MapReduce ✓
- Deep Learning Frameworks ✓

Analytics and Algorithms

- Similarity Search ✓
- Hypothesis Testing
- Regressions->Transformers ✓
- Recommendation Systems
- Time Series

Big Data Analytics, The Class

Goal: Generalizations
A *model* or *summarization* of the data.

Data Workflow Frameworks

- Hadoop File System ✓
- Spark ✓
- Streaming ✓
- MapReduce ✓
- Deep Learning Frameworks ✓

Analytics and Algorithms

- Similarity Search ✓
- Hypothesis Testing
- Regressions->Transformers ✓
- **Recommendation Systems**
- Time Series

Recommendation Systems

- What other item will this **user** like? (based on previously liked items)

- How much will user like item X?

Recommendation Systems

- What other item will this **user** like? (based on previously liked items)

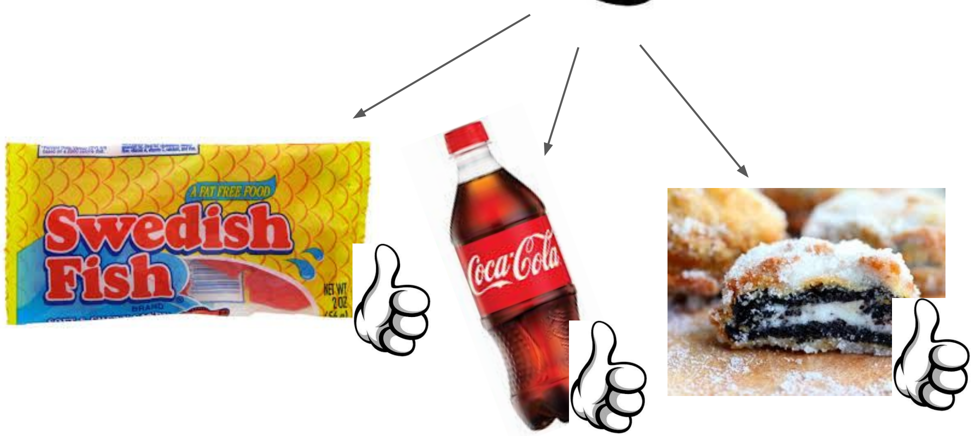
- How much will user like item X?

?

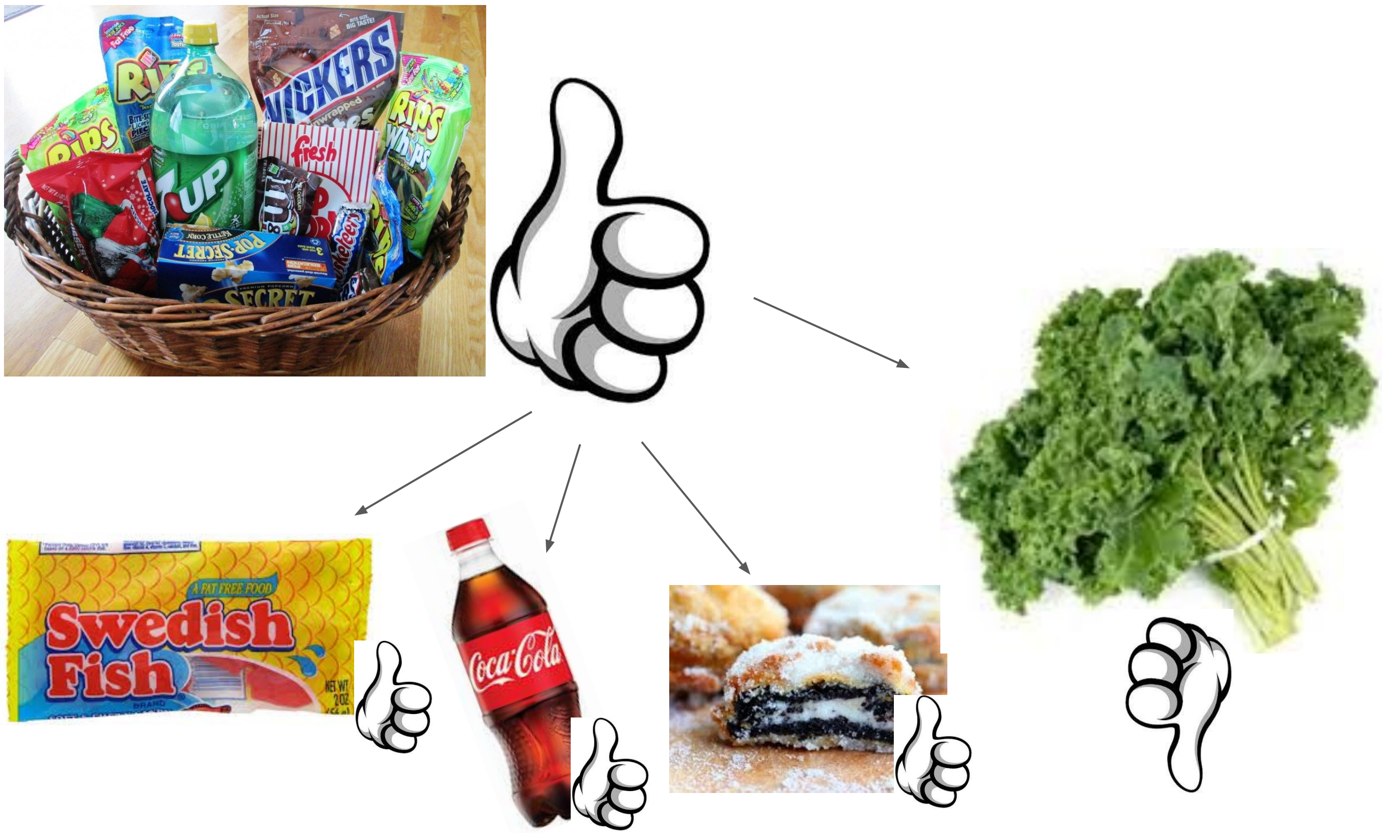
Recommendation Systems

- What other item will this **user** like? (based on previously liked items)

- How much will user like item X?

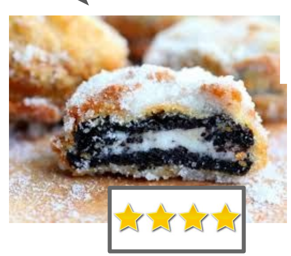


Recommendation Systems



Recommendation Systems

Past User Ratings

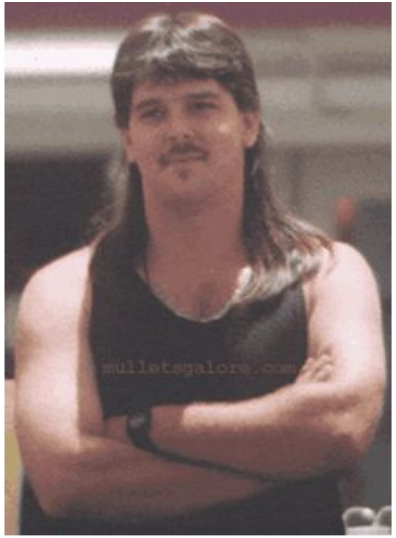


Recommendation Systems

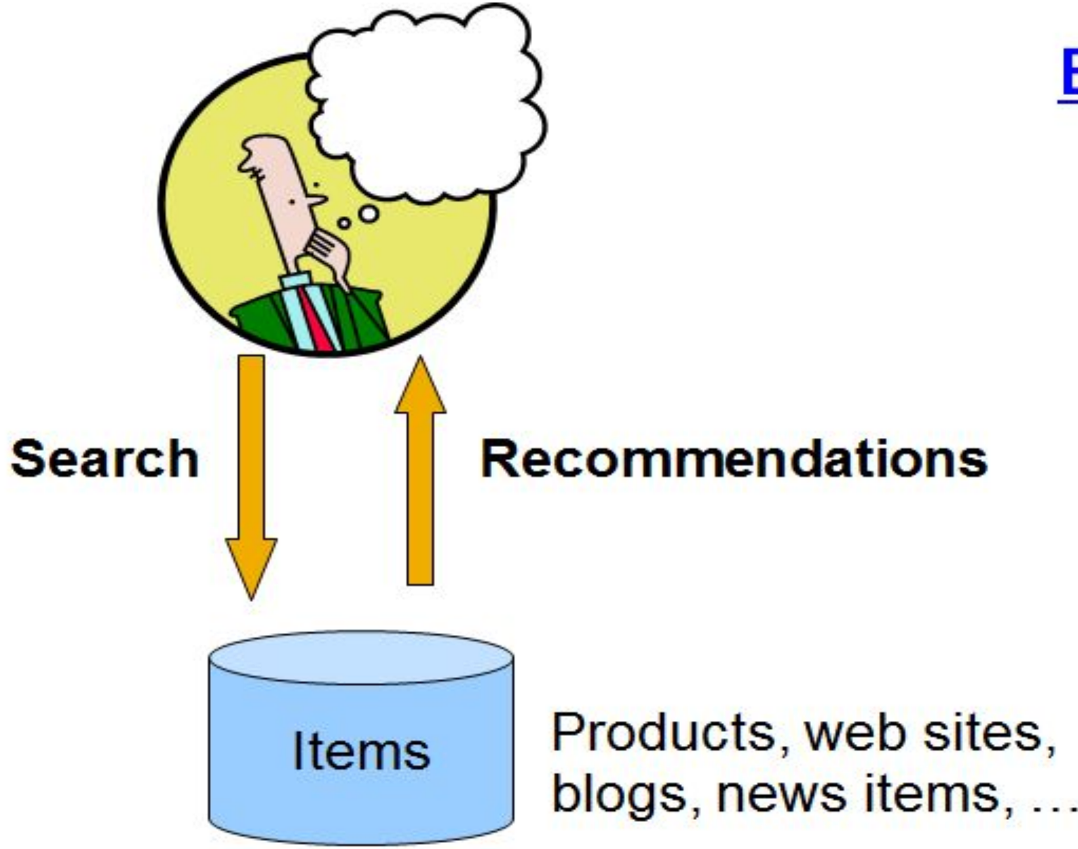
Why Big Data?

- Data with many potential features (and sometimes observations)
- An application of techniques for finding similar items
  - locality sensitive hashing
  - dimensionality reduction

Recommendation Systems: Example



- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD
- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer **X**



Examples:

amazon.com
PANDORA
StumbleUpon
del.icio.us
NETFLIX
movielens
helping you find the *right* movies
last.fm
the social music revolution
Google News
YouTube
XBOX LIVE
TikTok

Origins: Web Shopping

- Does Wal-Mart have everything you need?

Origins: Web Shopping

- Does Wal-Mart have everything you need?

(thelongtail.com)

Origins: Web Shopping

- Does Wal-Mart have everything you need?

- A lot of products are only of interest to a small population (i.e. "long-tail products").
- However, most people buy many products that are from the long-tail.

- Web shopping enables more choices
  - Harder to search
  - Recommendation engines to the rescue

(thelongtail.com)

Origins: Web Shopping

- Does Wal-Mart have everything you need?

- A lot of products are only of interest to a small population (i.e. "long-tail products").
- However, most people buy many products that are fro

- Web shopp
  - Harder t
  - Recomm

Rec Systems Model

Given: *users*, *items, utility matrix*

Rec Systems Model

Given: *users*, *items*, *utility matrix*

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4	5	3		3
B	5			4	2
C			5	2	

Rec Systems Model

Given: *users*, *items*, *utility matrix*

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4	5	3		3
B	5			4	2
C	?	?	5	2	?

Rec Systems Model

Problems to tackle:

1. Gathering ratings

2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)

3. Evaluation

Rec Systems Model

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Rec Systems Model

Common Approaches

1. **Content-based**
2. Collaborative
3. Latent Factor

Problems to tackle:

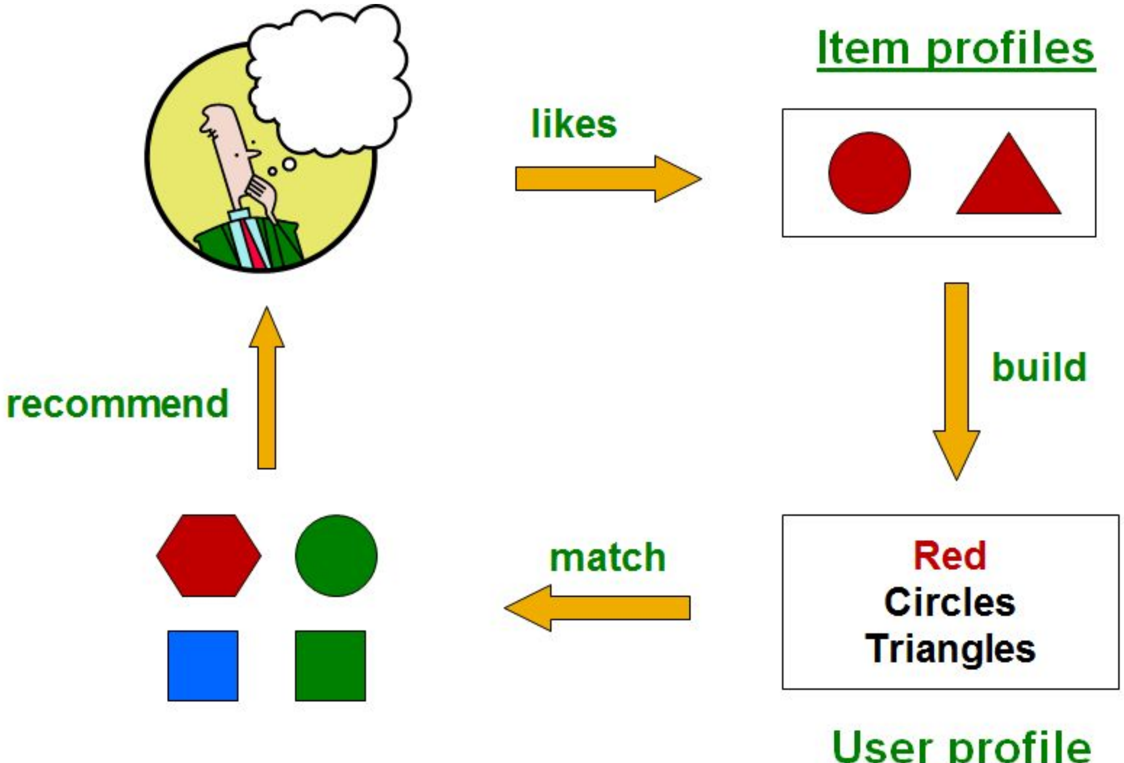
1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Content-Based Rec Systems

Based on similarity of items to past items that they have rated.

Content-Based Rec Systems

Based on similarity of items to past items that they have rated.



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Content-Based Rec Systems

Based on similarity of items to past items that they have rated.

1. Build profiles of items (set of features); examples:

    shows: producer, actors, theme, review

    people: friends, posts

pick words with tf-idf

Content-Based Rec Systems

Based on similarity of items to past items that they have rated.

1. `Build profiles of items (set of features);` examples:
   shows: producer, actors, theme, review
   people: friends, posts
   pick words with tf-idf
2. `Construct user profile from item profiles;` approach:
   average all item profiles of items they've purchased
   variation: weight by difference from their average

Content-Based Rec Systems

Based on similarity of items to past items that they have rated.

1. `Build profiles of items (set of features);` examples:

    shows: producer, actors, theme, review

    people: friends, posts

    pick words with tf-idf

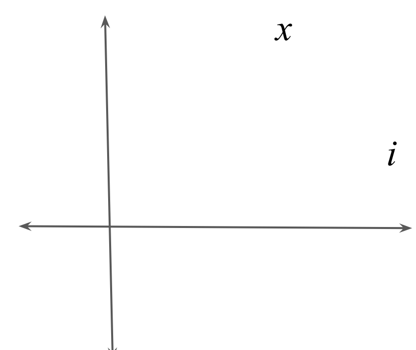
2. `Construct user profile from item profiles;` approach:

    average all item profiles of items they've purchased

    variation: weight by difference from their average ratings

3. `Predict ratings for new items;` approach:

    find similarity between user and items



Content-Based Rec Systems

Based on similarity of items to past items that they have rated.

1. Build profiles of items (set of features); examples:

    shows: producer, actors, theme, review

    people: friends, posts

    pick words with tf-idf

2. Construct user profile from item profiles; approach:

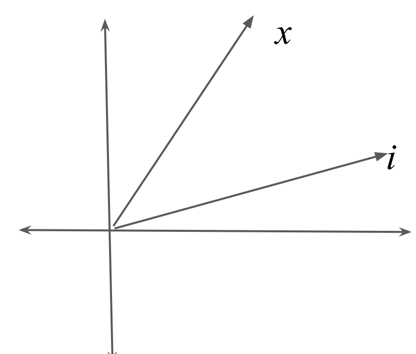
    average all item profiles of items they've purchased

    variation: weight by difference from their average ratings

3. Predict ratings for new items; approach:

    find similarity between user and items

$$utility(user, i) = \cos(x, i) = \frac{x \cdot i}{||x|| \cdot ||i||}$$



Distance Metrics (for Similarity)

finding *near-neighbors* in *high-dimensional space*

There are other metrics of similarity. e.g:

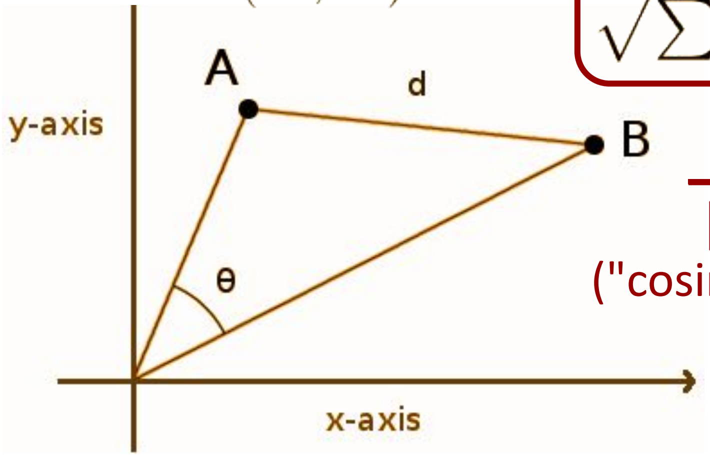
- Euclidean Distance
- Cosine Distance

  …

- Edit Distance
- Hamming Distance

$$distance(X,Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$ ("L2 Norm")

$$distance(X,Y) = 1 - \boxed{\frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}}$$



$$\frac{X \cdot Y}{||X||\ ||Y||}$$

("cosine similarity")

Content-Based Rec Systems

- Only need users history
- Captures unique tastes
- Can recommend new items
- Can provide explanations

Content-Based Rec Systems

- Only need users history
- Captures unique tastes
- Can recommend new items
- Can provide explanations

- Need good features
- New users don’t have history
- Doesn’t venture “outside the box” (Overspecialized)

Content-Based Rec Systems

- Only need users history
- Captures unique tastes
- Can recommend new items
- Can provide explanations

- Need good features
- New users don’t have history
- Doesn’t venture “outside the box” (Overspecialized)

(not exploiting other users judgments)

Collaborative Filtering

(not exploiting other users judgments)

Rec Systems

Common Approaches

1. **Content-based**
2. Collaborative
3. Latent Factor

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Rec Systems

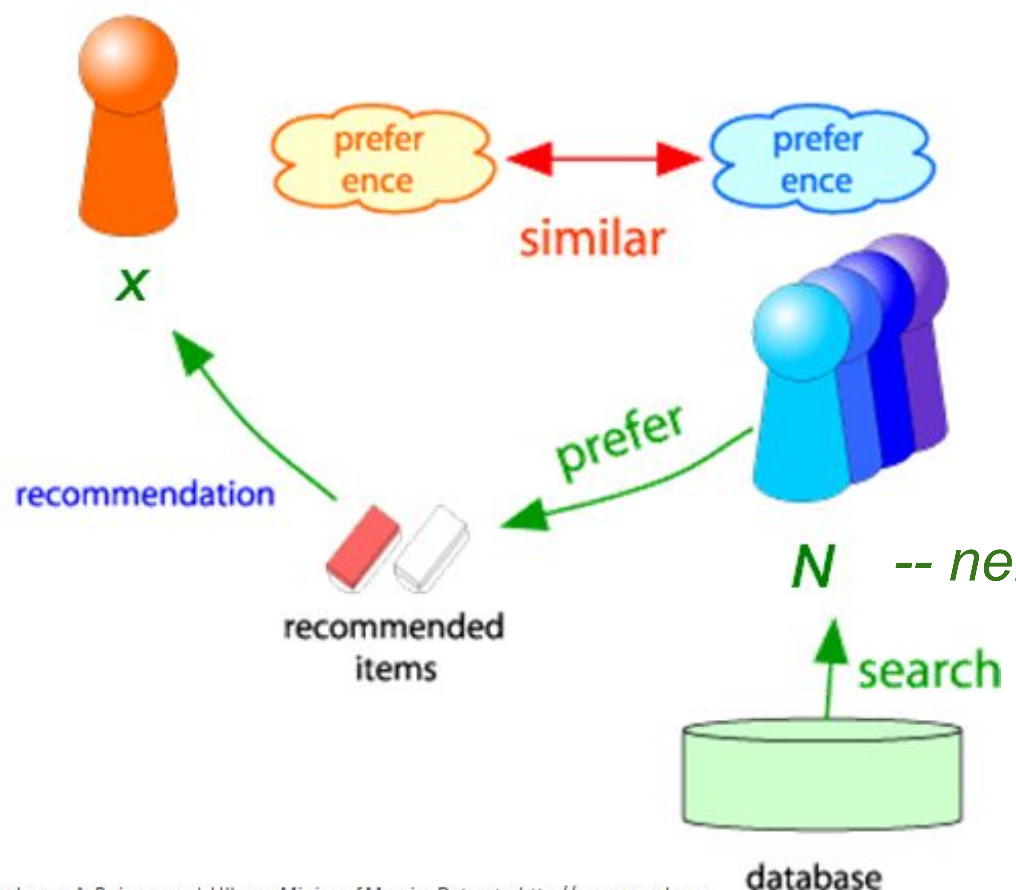
Common Approaches

1. Content-based
2. **Collaborative**
3. Latent Factor

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Collaborative Filtering



-- neighborhood

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4	5	2		3
B	5			4	2
C			5	2	

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4	5	2		3
B	5			4	2
C			5	2	

General Idea:

1) Find similar users = "neighborhood"

2) Infer rating based on how similar users rated

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4	5	2		3
B	5			4	2
C			5	2	

Given: *user*, x; *item*, i; *utility matrix*, u

1. Find neighborhood, N # set of k users most similar to x who have also rated i

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4	5	2		3
B	5			4	2
C			5	2	

Given: *user, x;* *item, i;* *utility matrix, u*

1. Find neighborhood, *N* # set of *k* users most similar to *x* who have also rated *i*

Two Challenges: (1) user bias, (2) missing values

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4 => 0.5	5 => 1.5	2 => -1.5	=> 0	3 => -0.5
B	5			4	2
C			5	2	

Given: *user, x; item, i; utility matrix, u*

1. Find neighborhood, *N* # set of *k* users most similar to *x* who have also rated *i*

   Two Challenges: (1) user bias, (2) missing values

   Solution: subtract user's mean, add zeros for missing

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4 => 0.5	5 => 1.5	2 => -1.5	=> 0	3 => -0.5
B	5			4	2
C			5	2	

```
Given: user, x;   item, i;   utility matrix, u
0. Update u: mean center, missing to 0
1.  Find neighborhood, N # set of k users most similar to
                              x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
```

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4 => 0.5	5 => 1.5	2 => -1.5	=> 0	3 => -0.5
B	5			4	2
C			5	2	

```
Given: user, x;   item, i;   utility matrix, u
0. Update u: mean center, missing to 0
1.  Find neighborhood, N # set of k users most similar to
                              x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
```

Collaborative Filtering

user	Game of Thrones	Fargo	Brooklyn Nine-Nine	Silicon Valley	Walking Dead
A	4 => 0.5	5 => 1.5	2 => -1.5	=> 0	3 => -0.5
B	5			4	2
C			5	2	

```
Given: user, x;   item, i;   utility matrix, u
0. Update u: mean center, missing to 0
1.  Find neighborhood, N # set of k users most similar to
                              x who have also rated i
    -- sim(x, other) = cosine_sim(u[x], u[other])
    -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
    -- average, weighted by sim
```

$$utility(x,i) = \frac{\sum_{y \in N} Sim(x,y) \cdot utility(y,i)}{\sum_{y \in N} Sim(x,y)}$$

Collaborative Filtering

"User-User collaborative filtering"

```
Given: user, x;   item, i;   utility matrix, u
0. Update u: mean center, missing to 0
1.  Find neighborhood, N # set of k users most similar to
                                   x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
   -- average, weighted by sim
```

$$utility(x,i) = \frac{\sum_{y \in N} Sim(x,y) \cdot utility(y,i)}{\sum_{y \in N} Sim(x,y)}$$

Collaborative Filtering

"User-User collaborative filtering"

Item-Item:

Flip rows/columns of utility matrix and use same methods.
(i.e. estimate rating of item i, by finding similar items, j)

```
Given: user, x;   item, i;   utility matrix, u
0. Update u: mean center, missing to 0
1.  Find neighborhood, N # set of k users most similar to
                              x who have also rated i
    -- sim(x, other) = cosine_sim(u[x], u[other])
    -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
    -- average, weighted by sim
```

$$utility(x,i) = \frac{\sum_{y \in N} Sim(x,y) \cdot utility(y,i)}{\sum_{y \in N} Sim(x,y)}$$

Collaborative Filtering

"User-User collaborative filtering"

Item-Item:

Flip rows/columns of utility matrix and use same methods.
(i.e. estimate rating of item i, by finding similar items, j)

```
Given: user, x;   item, i;   utility matrix, u
0. Update u: mean center, missing to 0
1.  Find neighborhood, N # set of k items most similar to
                                   i also rated by x
   -- sim(i, other) = cosine_sim(u[i], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) by x based on N
   -- average, weighted by sim
```

$$\text{utility}(x,i) = \frac{\sum_{j \in N} Sim(i,j) \cdot \text{utility}(x,j)}{\sum_{j \in N} Sim(i,j)}$$

item-item vs user-user

Item-item often works better than user-user. Why?

Users tend to be more different from each other than items are from other items.

e.g. Mary likes jazz + rock, Coleman likes classical + rock,
but Mary may still have same rock preferences as Coleman

item-item vs user-user

Item-item often works better than user-user. Why?

Users tend to be more different from each other than items are from other items.

e.g. Mary likes jazz + rock, Coleman likes classical + rock,
but Mary may still have same rock preferences as Bob

In other words, users span genres but items usually do not.

Item-Item: Example

movies

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- unknown rating - rating between 1 to 5

Item-Item: Example

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie **1** by user **5**

Item-Item: Example

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	**sim(1,m)**
1	1		3		?	5			5		4		**1.00**
2			5	4			4			2	1	3	**-0.18**
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		**-0.10**
5			4	3	4	2					2	5	**-0.31**
6	1		3		3			2			4		**0.59**

Same as cosine sim when subtracting the mean

Neighbor selection:
Identify movies similar to movie **1, rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute cosine similarities between rows

Item-Item: Example

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

Item-Item: Example

movies

	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

***utility*(1, 5) = (0.41*2 + 0.59*3) / (0.41+0.59)**

$$\text{utility}(x,i) = \frac{\sum_{j \in N} Sim(i,j) \cdot \text{utility}(x,j)}{\sum_{j \in N} Sim(i,j)}$$

item-item vs user-user

	user1	user2	user3	user4	user5	*mu, std*
item1	1 => -1.5	2 => -0.5	=> 0	2 => -0.5	5 => 2.5	*2.5*
item2	2 => -1	3 => 0	2 => -1	=> 0	5 => 2	*3*
item3	5 => 1	=> 0	4 => 0	=> 0	3 => -1	*4*

item-item vs user-user

	user1	user2	user3	user4	user5	*mu, std*
item1	1 => -1.5	2 => -0.5	=> 0	2 => -0.5	5 => 2.5	*2.5*
item2	2 => -1	3 => 0	2 => -1	=> 0	5 => 2	*3*
item3	5 => 1	=> 0	4 => 0	=> 0	3 => -1	*4*

sim(item1, item2) = 0.89

sim(item1, item3) = -0.94 X

$$\text{score(user3)} = \frac{(0.89 * 2)}{0.89} = 2$$

Rec Systems

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Rec Systems

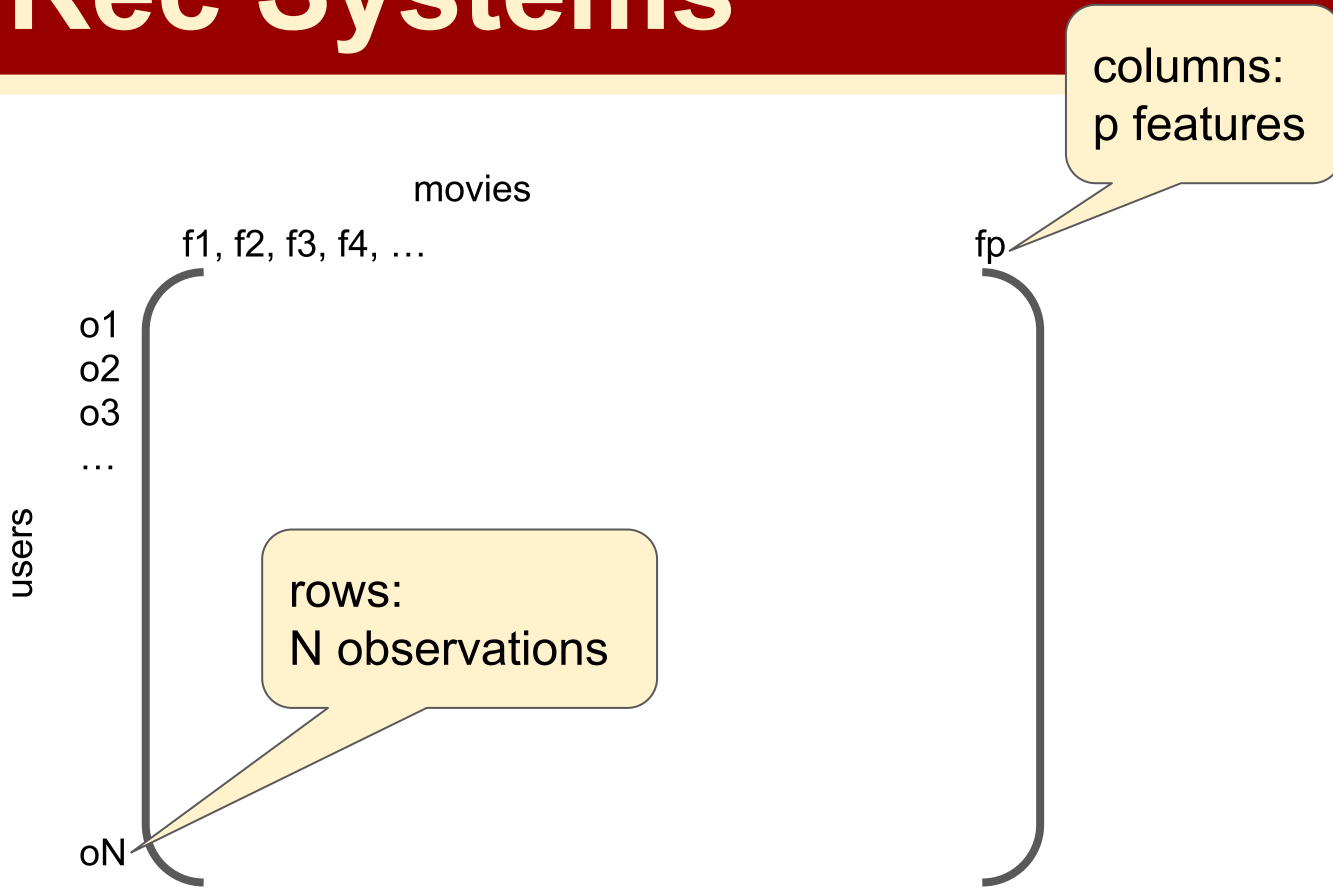
Common Approaches

1. Content-based
2. Collaborative
3. **Latent Factor**

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Rec Systems


columns:
p features
movies
f1, f2, f3, f4, …
fp
o1
o2
o3
…
users
rows:
N observations
oN

Rec Systems

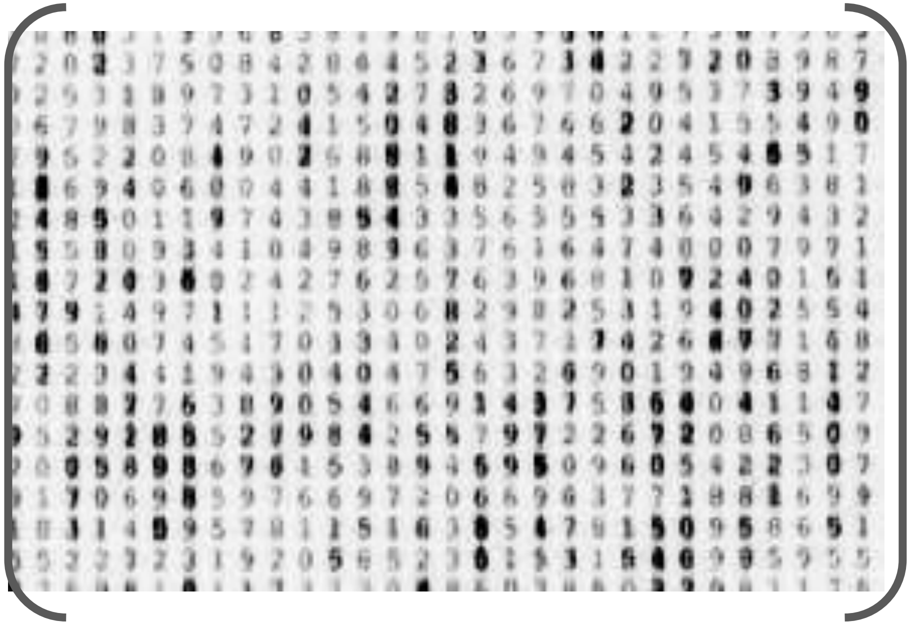
Goal: Complete Matrix

movies

f1, f2, f3, f4, … fp

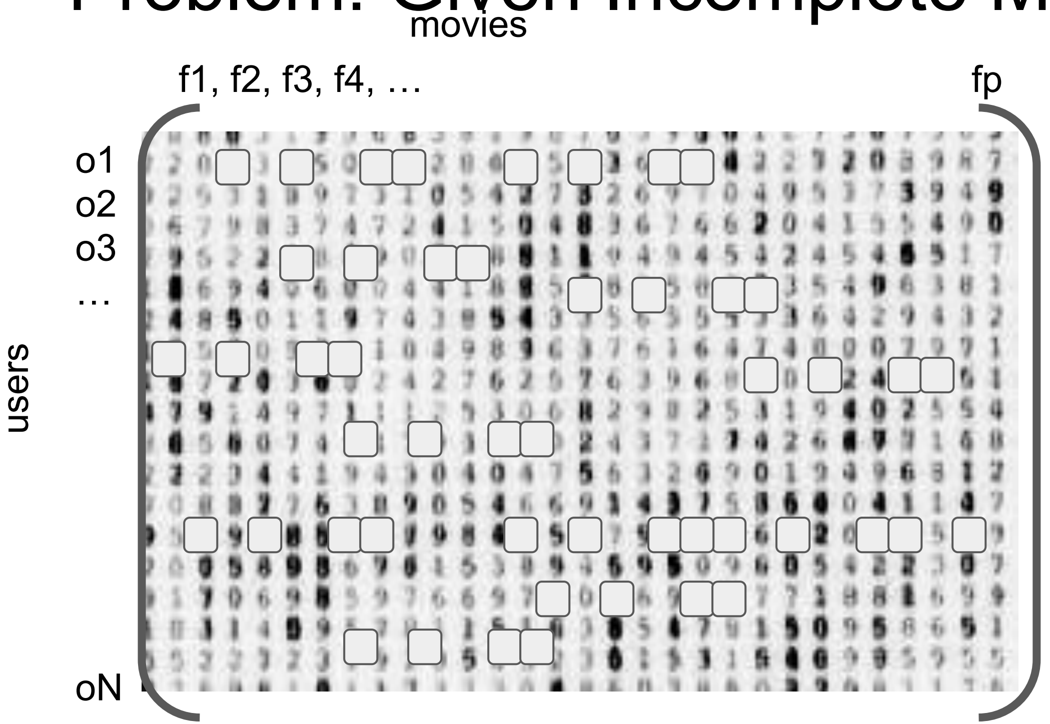
users

o1
o2
o3
…
oN



Rec Systems

Problem: Given Incomplete Matrix



Rec Systems

Complete Matrix using Latent Factors

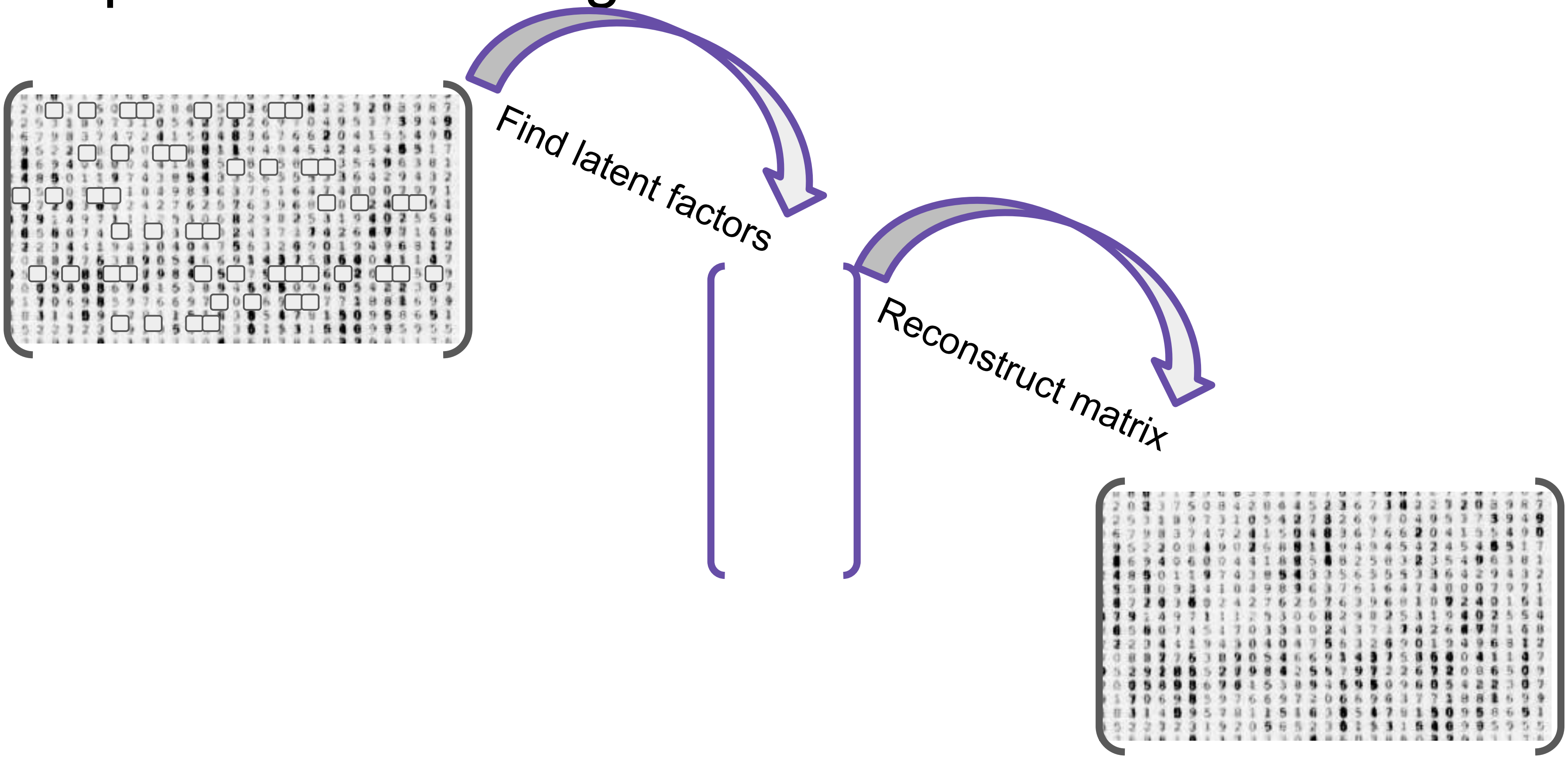
$$
\begin{array}{c|c}
 & f1, f2, f3, f4, \ldots \quad fp \\
\hline
\begin{array}{c} o1 \\ o2 \\ o3 \\ \ldots \\ \\ oN \end{array} & \left(\begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}\right)
\end{array}
\qquad \longrightarrow \qquad
\begin{array}{c|c}
 & c1, c2, c3, c4, \ldots \quad cp' \\
\hline
\begin{array}{c} o1 \\ o2 \\ o3 \\ \ldots \\ \\ oN \end{array} & \left(\begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}\right)
\end{array}
$$

Dimensionality reduction

Try to best represent but with on p’ columns.

Rec Systems

Complete Matrix using Latent Factors



Dimensionality Reduction PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} = U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix,
U: "left singular vectors",
D: "singular values" (diagonal),
V: "right singular vectors"

Projection (dimensionality reduced space) in 3 dimensions:

$$(U_{[nx3]} D_{[3x3]} V_{[px3]}^T)$$

To reduce features in new dataset:

$$X_{new} DV = X_{new\_small}$$

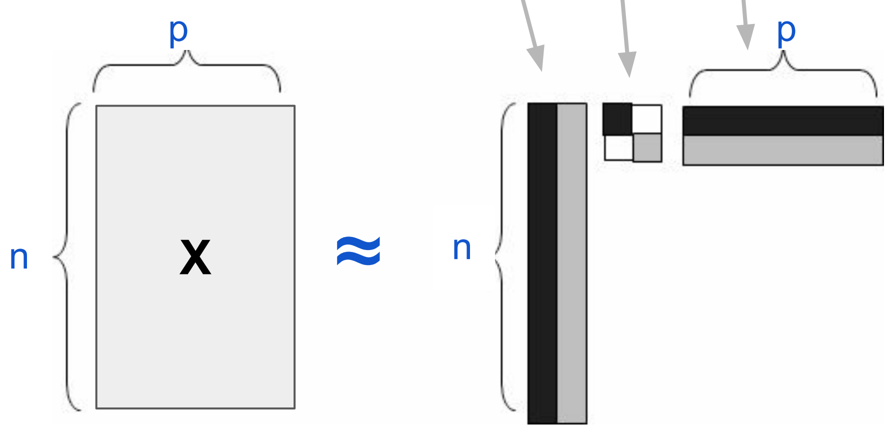
Dimensionality Reduction PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[n\times p]} = U_{[n\times r]} D_{[r\times r]} V_{[p\times r]}^{T}$$

X: original matrix,
D: "singular values" (diagonal),
U: "left singular vectors",
V: "right singular vectors"



Dimensionality Reduction PCA

$$X_{[nxp]} = U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

$$\begin{array}{c} \\ \text{SciFi} \\ \\ \text{Romnce} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \end{array} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

Users to movies matrix

Dimensionality Reduction PCA

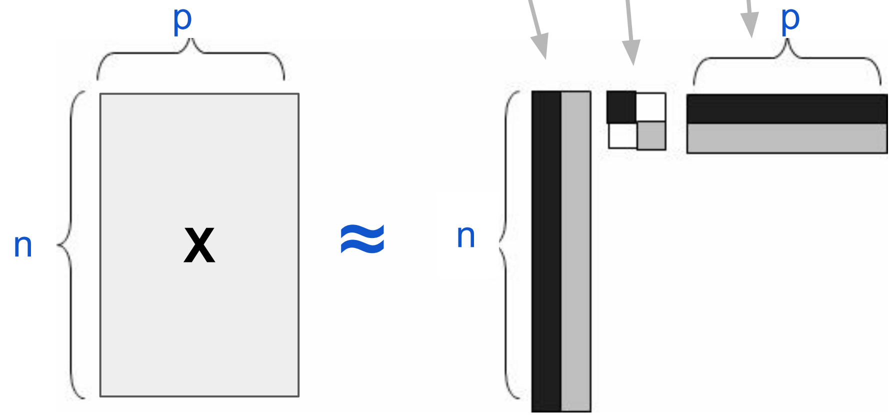
Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[n\times p]} = U_{[n\times r]} D_{[r\times r]} V_{[p\times r]}^T$$

X: original matrix,
D: "singular values" (diagonal),

U: "left singular vectors",
V: "right singular vectors"



Dimensionality Reduction PCA

- Goal: **Minimize the sum of reconstruction errors:**

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\|x_{ij}-z_{ij}\right\|^2$$

  - where $\boldsymbol{x_{ij}}$ are the "old" and $\boldsymbol{z_{ij}}$ are the "new" coordinates

s.

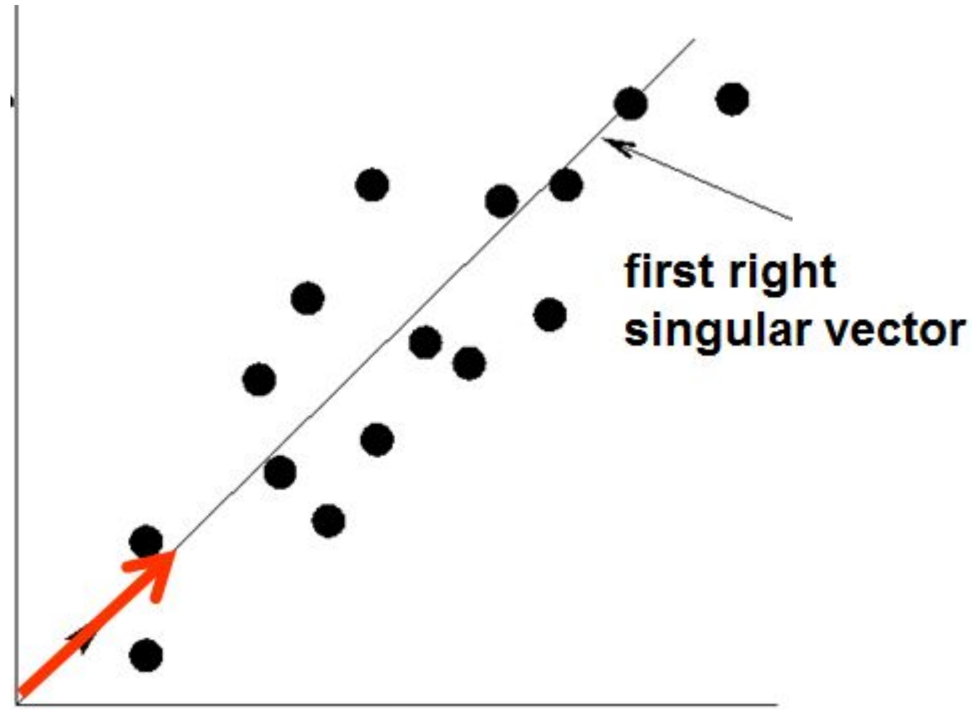
X: original mat lar vectors",
D: "singular va ular vectors"

To check how well the original matrix can be reproduced:

$Z_{[nxp]}$ = U D V^T , How does Z compare to original X?

Dimensionality Reduction PCA

$$X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$



PCA - Parallelized

1. Approximate solutions to PCA (very large speedups with little drawback!):
    a. **Stochastic Sampling** (also sometimes called "randomized" which is ambiguous): Only using a sample of rows (i.e. only some users for recommendation systems)

    b. **Truncated SVD:** Only optimizing for minimizing reconstruction error based on up to r dimensions (full SVD solves for up to min(n, p) dimensions and then you just truncate the result for the lower rank version). Positive side effect: using a smaller sample also can be sped up with less loss of power.

    c. **Limiting power iterations to a few iterations:** Power iterations from pagerank solves for the first principle component. This can be extended to multiple components.

    (more here.)

PCA - Parallelized

1. Approximate solutions to PCA (very large speedups with little drawback!):
    a. **Stochastic Sampling**
    b. **Truncated SVD**
    c. **Limiting power iterations to a few iterations**

2. Distribute the matrix operations. Complex; not as flexible (usually done across processors within node)
3. Data Parallelism: As in other instances stochastic or mini-batch gradient descent.

Rec Systems

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)
3. Evaluation

Rec Systems

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor

Problems to tackle:

1. Gathering ratings

2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks) (problem: hard to learn low ratings)

3. Evaluation