GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 14: RECOMMENDATION SYSTEMS

LAST TIME:

- DIMENSIONALITY REDUCTION
- PCA/SVD

QUESTIONS?

SINGULAR VALUE DECOMPOSITION

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \end{array} \\
\begin{array}{c} \text{CS} \\ \\ \\ \\ \text{MD} \\ \\ \\ \end{array} &
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

doc-to-concept similarity matrix

concepts strengths

term-to-concept similarity matrix

Consider a matrix A with n rows and d features.

The **singular value decomposition** *of A is given by:*

$$\underset{(n \times d)}{A} = \underset{(n \times k)}{U} \; \underset{(k \times k)}{\Sigma} \; \underset{(k \times d)}{V^T}$$

st. U, V are **orthogonal** *matrices and Σ is a* **diagonal** *matrix.*

→ $UU^T = I_n, \; VV^T = I_d$ → $\Sigma_{ij} = 0 \;\; (i \neq j)$

I. CONTENT-BASED FILTERING
II. COLLABORATIVE FILTERING
III. A SIMPLE MATRIX FACTORIZATION MODEL

EXERCISE:
IV. RECSYS IN PYTHON

The purpose of a **recommendation system** *is decide whether an item (product, event, movie, song) is something a user is highly likely to be interested*

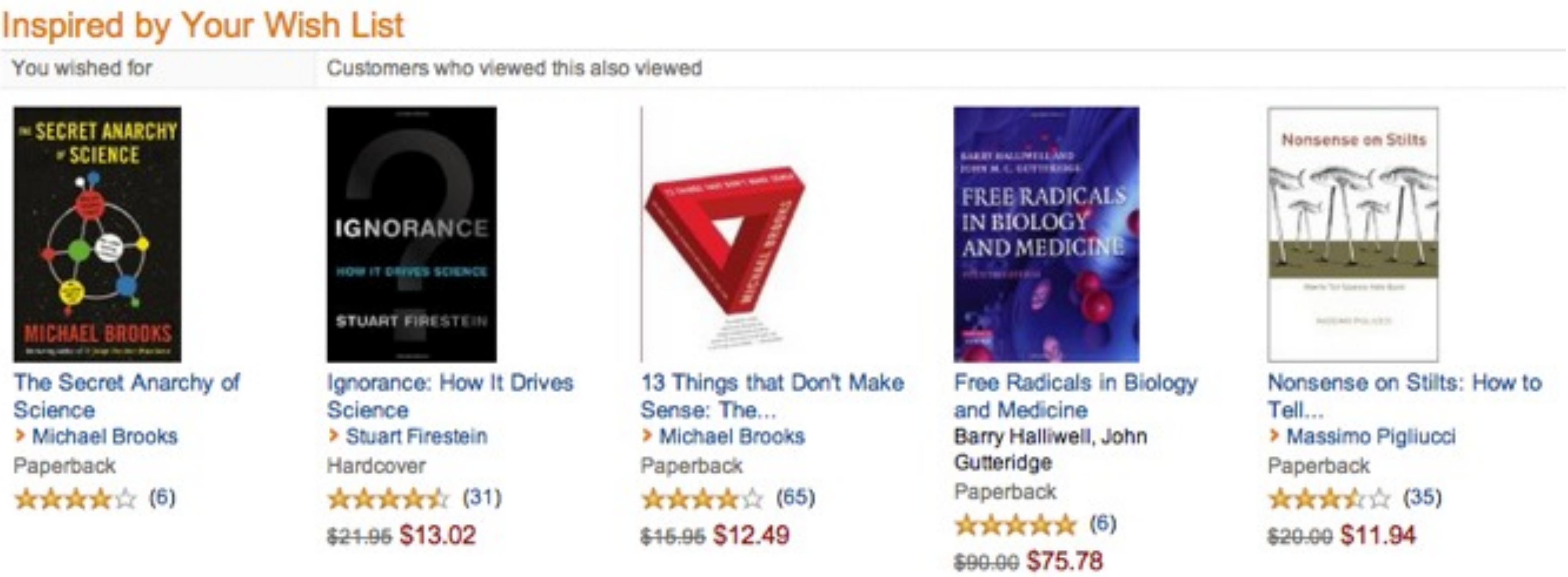
There are two general approaches to recsys design:

There are two general approaches to recsys design:

In **content-based filtering**, *items are mapped into a feature space, and recommendations depend on specified characteristics.*

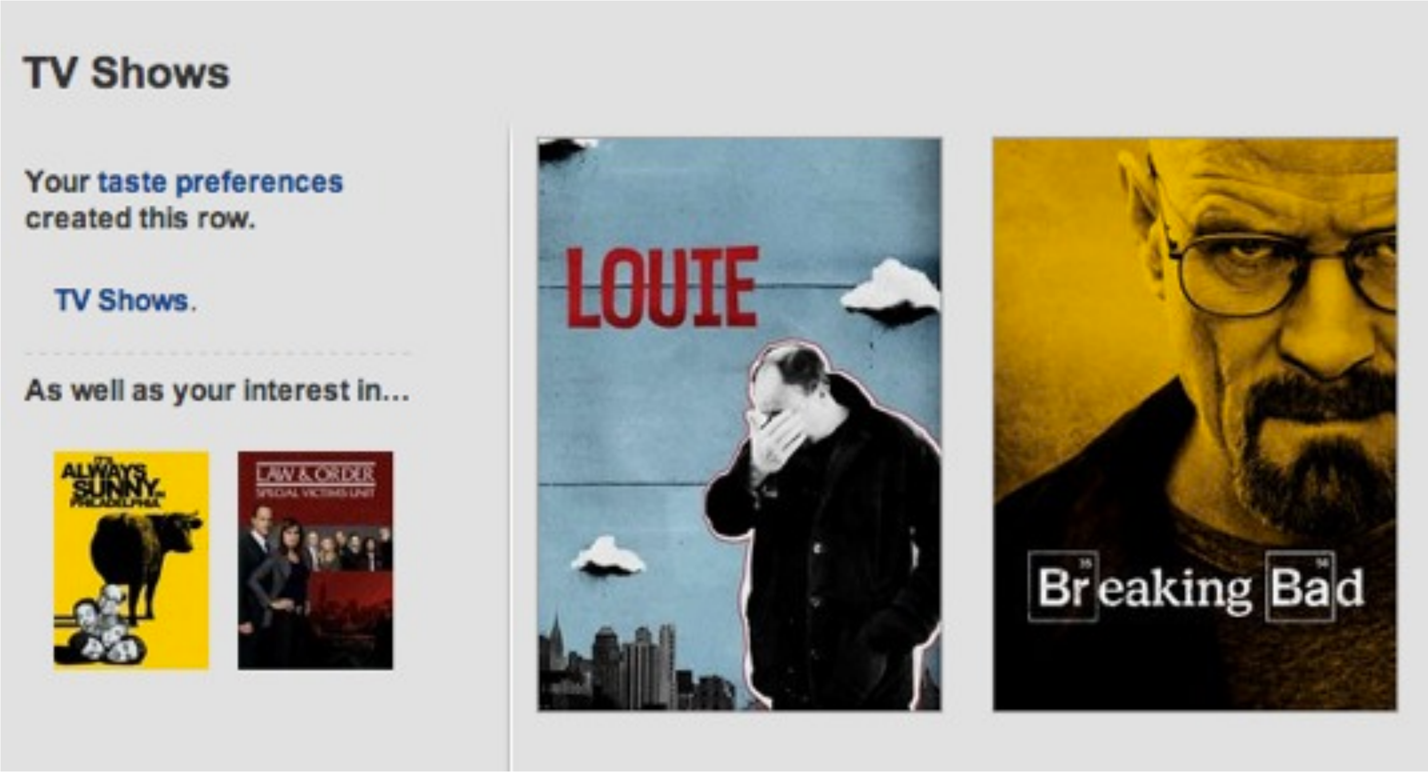
In contrast, the only data under consideration in **collaborative filtering** *are user-item ratings, and recommendations depend on user preferences.*

EXAMPLES – AMAZON

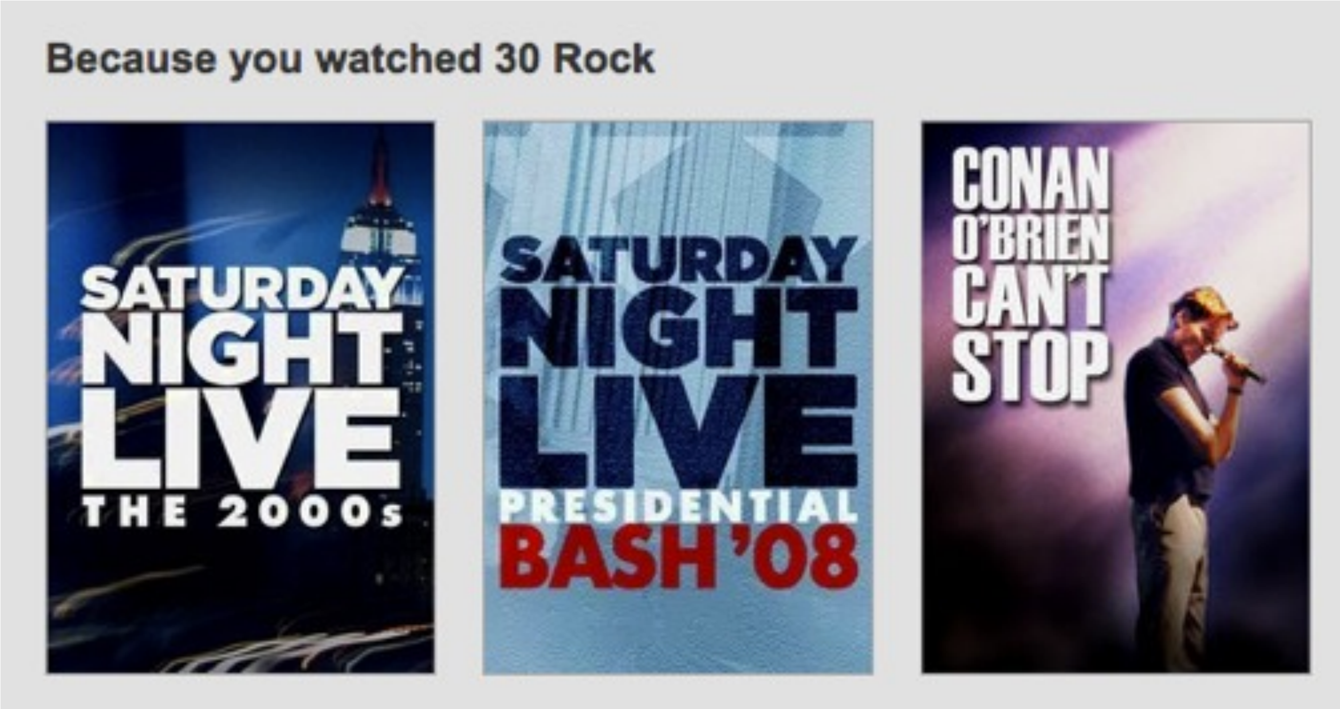


Inspired by Your Wish List
You wished for
Customers who viewed this also viewed
The Secret Anarchy of Science
Michael Brooks
Paperback
(6)
Ignorance: How It Drives Science
Stuart Firestein
Hardcover
(31)
~~$21.95~~ $13.02
13 Things that Don't Make Sense: The...
Michael Brooks
Paperback
(65)
~~$15.95~~ $12.49
Free Radicals in Biology and Medicine
Barry Halliwell, John Gutteridge
Paperback
(6)
~~$90.00~~ $75.78
Nonsense on Stilts: How to Tell...
Massimo Pigliucci
Paperback
(35)
~~$20.00~~ $11.94

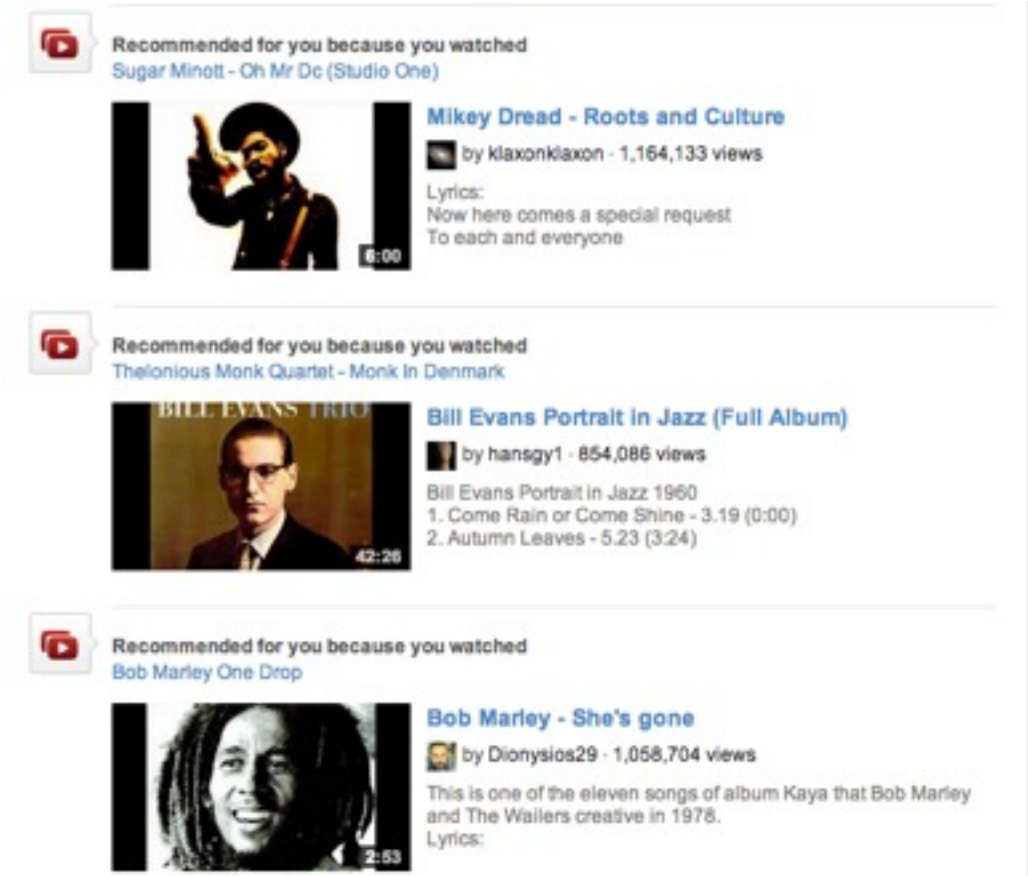
EXAMPLES – NETFLIX



EXAMPLES – NETFLIX



EXAMPLES – YOUTUBE



EXAMPLES – NYTIMES.COM

MOST E-MAILED | RECOMMENDED FOR YOU

1. **How Big Data Is Playing Recruiter for Specialized Workers**
2. SLIPSTREAM
   When Your Data Wanders to Places You've Never Been
3. MOTHERLODE
   The Play Date Gun Debate
4. **For Indonesian Atheists, a Community of Support Amid Constant Fear**
5. **Justice Breyer Has Shoulder Surgery**
6. BILL KELLER
   Erasing History

EXAMPLES – NYTIMES.COM

8. How do you determine my Most Read Topics?

Back to top ▲

Each NYTimes.com article is assigned topic tags that reflect the content of the article. As you read articles, we use these tags to determine your most-read topics.

To search for additional articles on one of your most-read topics, click that topic on your personalized Recommendations page. To learn more about topic tags, visit Times Topics.

I. CONTENT-BASED FILTERING

Content-based filtering *begins by mapping each item into a feature space. Both users and items are represented by vectors in this space.*

Content-based filtering *begins by mapping each item into a feature space. Both users and items are represented by vectors in this space.*

Two approaches:

1) Map users and items to same feature space, compute distance between a user and item

2) Create features from user+item pairs and use ML algorithm to predict like/dislike

2) Create features from user+item pairs and use ML algorithm to predict like/dislike

Each sample/row is a user/item pair with some outcome:

Outcome = Bought
User features - (purchase power, demographics)
Item features - category, metadata
User/Item features - user/item category overlap

1) Map users and items to same feature space, compute distance between a user and item

Item vectors measure the degree to which the item is described by each feature, and user vectors measure a user's preferences for each feature.

1) Toy Story -> (Comedy: 1, Animated: 1, Mafia: 0)
Godfather -> (Comedy: 0, Animated, Mafia: 1)

User 1 -> (Comedy 1, Animated: 0, Mafia: 0)

features = (big box office, aimed at kids, famous actors)

items (movies):

Finding Nemo = (5, 5, 2)

Mission Impossible = (3, -5, 5)

Jiro Dreams of Sushi = (-4, -5, -5)

EXAMPLE – CONTENT-BASED FILTERING

features = (big box office, aimed at kids, famous actors)

items (movies):

Finding Nemo = (5, 5, 2)

Mission Impossible = (3, -5, 5)

Jiro Dreams of Sushi = (-4, -5, -5)

users:

Jason = (-3, 2, -2)

features = (big box office, aimed at kids, famous actors)

items (movies):	predicted ratings*:
Finding Nemo = (5, 5, 2)	(-3*5 + 2*5 -2*2) = -9
Mission Impossible = (3, -5, 5)	(-3*3 - 2*5 -2*5) = -29
Jiro Dreams of Sushi = (-4, -5, -5)	(3*4 - 2*5 + 2*5) = +12

users:

Jason = (-3, 2, -2)

features = (big box office, aimed at kids, famous actors)

items (movies):	predicted ratings*:
Finding Nemo = (5, 5, 2)	$(-3*5 + 2*5 - 2*2) = -9$
Mission Impossible = (3, -5, 5)	$(-3*3 - 2*5 - 2*5) = -29$
Jiro Dreams of Sushi = (-4, -5, -5)	$(3*4 - 2*5 + 2*5) = +12$

users:

Jason = (-3, 2, -2)

One notable example of content-based filtering is Pandora, which maps songs into a feature space using features (or "genes") designed by the Music Genome Project.

Using song vectors that depend on these features, Pandora can create a station with music having similar properties to a song the user selects.

Content-based filtering has some difficulties:

Content-based filtering has some difficulties:

- *need to map each item into a feature space (usually by hand!)*
- *recommendations are limited in scope (items must be similar to each other)*
- *hard to create cross-content recommendations (eg books/music films...this would require comparing elements from different feature spaces!)*

II. COLLABORATIVE FILTERING

The purpose of a **recommendation system** *is decide whether an item (product, event, movie, song) is something a user is highly likely to be interested*

REFRAMED AS:

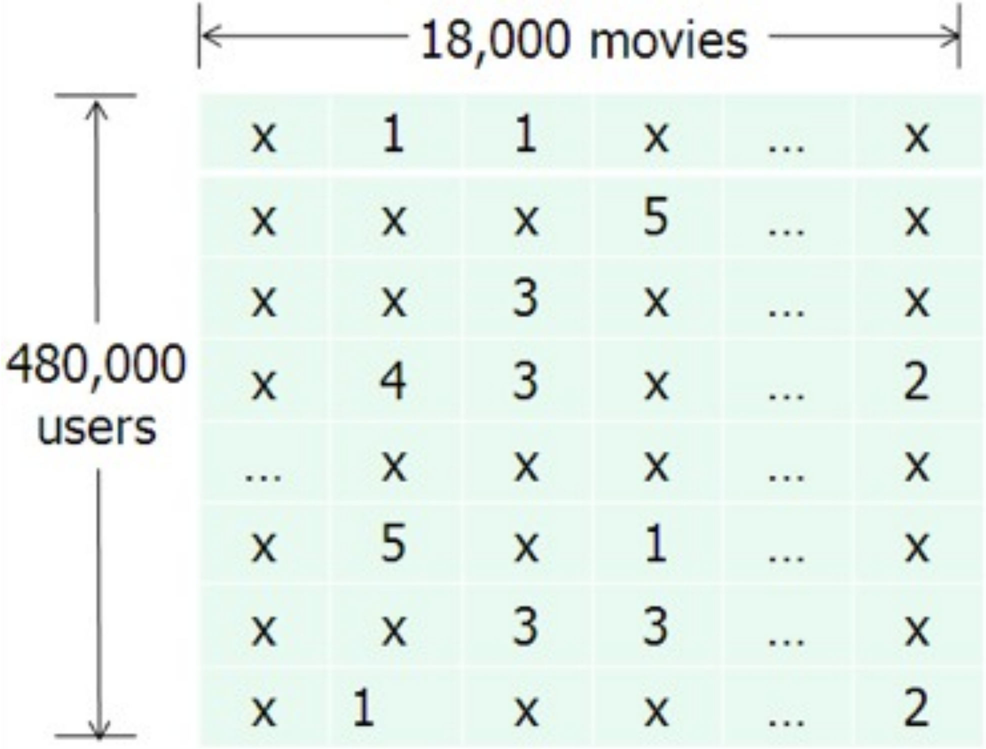
The purpose of a **recommendation system** *is to predict a rating that a user will give an item that they have not yet rated.*

Collaborative filtering *refers to a family of methods for predicting ratings where instead of thinking about users and items in terms of a feature space, we are only interested in the existing user-item ratings themselves.*

Collaborative filtering *refers to a family of methods for predicting ratings where instead of thinking about users and items in terms of a feature space, we are only interested in the existing user-item ratings themselves.*

In this case, our dataset is a ratings matrix whose columns correspond to items, and whose rows correspond to users.

RATINGS MATRIX



NOTE

This matrix will always be *sparse*!

source: http://www.eecs.berkeley.edu/~zhanghao/main/publications/subfolder/netflix.png

Collaborative filtering can be done in two different ways.

Collaborative filtering can be done in two different ways.

Item-based CF *uses ratings data to create an item-item similarity matrix.*

Collaborative filtering can be done in two different ways.

Item-based CF *uses ratings data to create an item-item similarity matrix.*

Recommendations are then made to a user for items most similar to those that the user has already rated highly.

Collaborative filtering can be done in two different ways.

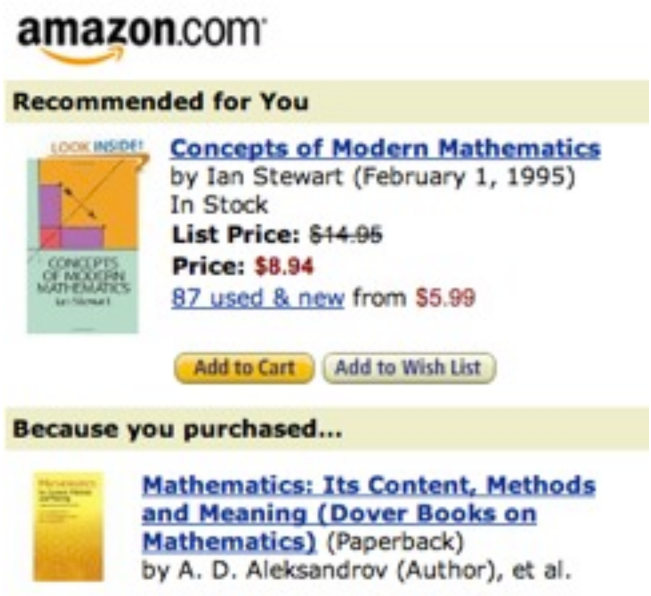
Item-based CF *uses ratings data to create an item-item similarity matrix.*

Recommendations are then made to a user for items most similar to those that the user has already rated highly.

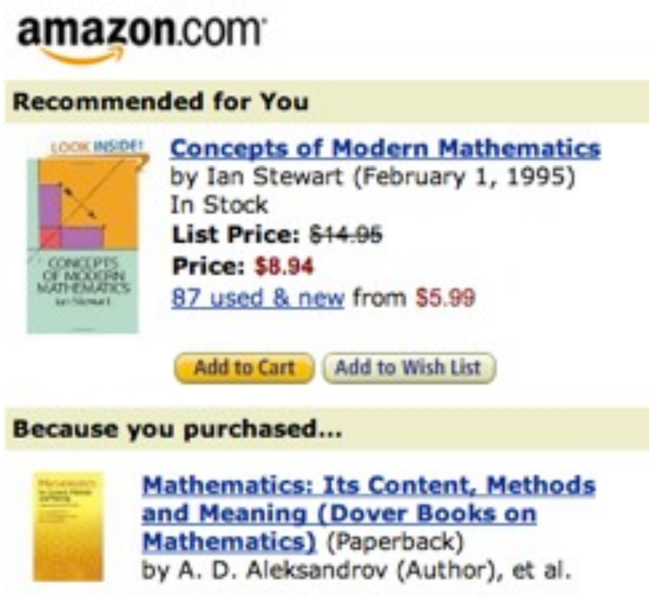
This is also called **memory-based CF** *or* **neighborhood** methods


Customers Who Bought This Item Also Bought
LOOK INSIDE!
Pitch Dark (NYRB Classics)
Renata Adler
Paperback
$11.54
LOOK INSIDE!
How Literature Saved My Life
David Shields
(60)
Hardcover
$18.08
No image available
Bleeding Edge
Thomas Pynchon
Hardcover
$18.05
LOOK INSIDE!
THE FLAME THROWERS
RACHEL KUSHNER
The Flamethrowers: A Novel
Rachel Kushner
(17)
Hardcover
$15.79

Neighborhood methods such as item-based CF are popular and easy to understand, but they don't scale well.



Neighborhood methods such as item-based CF are popular and easy to understand, but they don't scale well.



NOTE

Item-based CF is different than content-based filtering!

Though we're making recommendations based on items, we are *not* embedding the items in a feature space.

Model-based *collaborative filtering abandons the neighborhood approach and applies other techniques to the ratings matrix.*

Model-based *collaborative filtering abandons the neighborhood approach and applies other techniques to the ratings matrix.*

The most popular model-based CF techniques use matrix decomposition techniques to find deeper structure in the ratings data.

Model-based *collaborative filtering abandons the neighborhood approach and applies other techniques to the ratings matrix.*

The most popular model-based CF techniques use matrix decomposition techniques to find deeper structure in the ratings data.

For example, we could decompose the ratings matrix via SVD to reduce the dimensionality and extract **latent variables***.*

Once we identify the latent variables in the ratings matrix, we can express both users and items in terms of these latent variables.

Once we identify the latent variables in the ratings matrix, we can express both users and items in terms of these latent variables.

As before, values in the item vectors represent the degree to which an item exhibits a given feature, and values in the user vectors represent user preferences for a given feature.

Once we identify the latent variables in the ratings matrix, we can express both users and items in terms of these latent variables.

As before, values in the item vectors represent the degree to which an item exhibits a given feature, and values in the user vectors represent user preferences for a given feature.

Ratings are constructed by taking dot products of user & item vectors in the latent feature space.

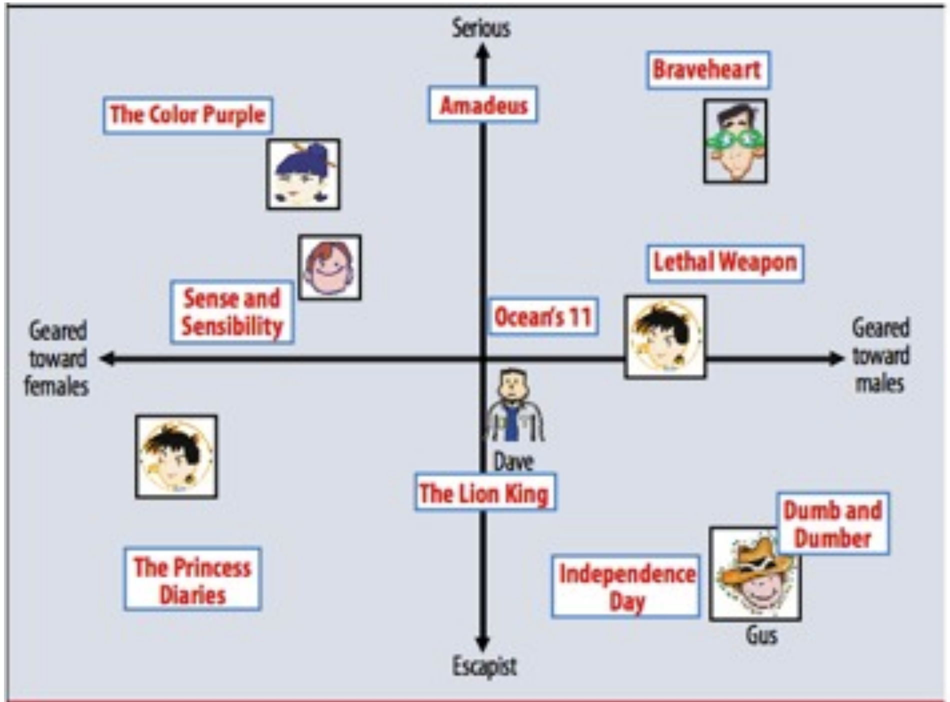


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

source: http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf

This approach is domain independent, and requires no explicit user or item profiles to be created.

This approach is domain independent, and requires no explicit user or item profiles to be created.

It combines predictive accuracy, scalability, and enough flexibility for practical modeling (we'll see what this means in a moment).

This approach is domain independent, and requires no explicit user or item profiles to be created.

It combines predictive accuracy, scalability, and enough flexibility for practical modeling (we'll see what this means in a moment).

Since the conclusion of the Netflix prize, these latent factor methods for collaborative filtering have been regarded as the state-of-the-art in recsys technology.

But they do have some drawbacks:

- *lots of (high-dimensional) ratings data needed*
- *data is typically very sparse (in the Netflix prize dataset, ~99% of possible ratings were missing)*

- **cold start problem***: need lots of data on new user or item before recommendations can be made*

The cold start problem arises because we've been relying only on ratings data, or on **explicit feedback** *from users.*

The cold start problem arises because we've been relying only on ratings data, or on **explicit feedback** *from users.*

Until a user rates several items, we don't know anything about her preferences!

The cold start problem arises because we've been relying only on ratings data, or on **explicit feedback** *from users.*

Until a user rates several items, we don't know anything about her preferences!

We can get around this by enhancing our recommendations using **implicit feedback**, *which may include things like item browsing behavior, search patterns, purchase history, etc.*

While explicit feedback (ratings, likes, purchases) leads to high quality ratings, the data is sparse and cold starts are problematic.

While explicit feedback (ratings, likes, purchases) leads to high quality ratings, the data is sparse and cold starts are problematic.

Meanwhile implicit feedback (browsing behavior, etc) leads to less accurate ratings, but the data is much more dense (and less invasive to collect).

While explicit feedback (ratings, likes, purchases) leads to high quality ratings, the data is sparse and cold starts are problematic.

Meanwhile implicit feedback (browsing behavior, etc) leads to less accurate ratings, but the data is much more dense (and less invasive to collect).

Implicit feedback can help to infer user preferences when explicit feedback is not available, therefore easing the cold start problem.

Hybrid filtering methods *provide another way to get around the cold start problem by combining filtering methods (eg, by using content-based info to "boost" a collaborative model).*

Hybrid filtering methods *provide another way to get around the cold start problem by combining filtering methods (eg, by using content-based info to "boost" a collaborative model).*

This content-based info can be item-based as above, or even user-based (eg, demographic info).

Hybrid filtering methods *provide another way to get around the cold start problem by combining filtering methods (eg, by using content-based info to "boost" a collaborative model).*

This content-based info can be item-based as above, or even user-based (eg, demographic info).

Hybrid methods can also make the data sparsity issue easier to deal with, by broadening the set of features under consideration.

III. A SIMPLE MATRIX FACTORIZATION MODEL

Matrix factorization decomposes the ratings matrix and maps users and items into a low-dimensional vector space spanned by a basis of latent factors.

Matrix factorization decomposes the ratings matrix and maps users and items into a low-dimensional vector space spanned by a basis of latent factors.

Predicted ratings are given by inner products in this space, so for user u and item i we can write:

$$\hat{r}_{ui} = q_i^T r_u$$

Factoring the ratings matrix via SVD leads to difficulty, since the matrix is typically sparse and therefore our information about the data is incomplete.

Factoring the ratings matrix via SVD leads to difficulty, since the matrix is typically sparse and therefore our information about the data is incomplete.

Interpolating missing values is an expensive process and can lead to inaccurate predictions, so we need another way to perform this factorization.

One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings matrix, we have to be careful to avoid overfitting.

One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings matrix, we have to be careful to avoid overfitting.

We can learn these feature vectors by minimizing the loss function:

$$\min_{q,p} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

where κ denotes the set of known ratings, and λ is a hyperparameter.

One possibility is to learn the feature vectors using the observed ratings only. Since this dramatically reduces the size of the ratings have to be careful to avoid overfitting.

We can learn these feature vectors by minimizing the loss f

$$\min_{q,p} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where κ denotes the set of known ratings, and λ is a hyperparameter.

NOTE

The loss function has two unknowns (q, p) and so is not convex!

This can be minimized using a method called *alternating least squares*.

It turns out that much of the variation in observed ratings is due to user or item biases (eg, some users are very critical, or some items are universally popular).

It turns out that much of the variation in observed ratings is due to user or item biases (eg, some users are very critical, or some items are universally popular).

We can capture these biases in our model by generalizing $\hat{r}_{ui}$ ·

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T r_u$$

Here μ *is a global average rating,* b_i *is the item bias,* b_u *is the user bias, and* $q_i^T r_u$ *is the user-item interaction.*

With this generalization, our minimization problem becomes:

$$\min_{q,p,b} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2 + b_u^2 + b_i^2)$$