GENERAL ASSEMBLY

INTRO TO DATA SCIENCE
LECTURE 14: RECOMMENDER SYSTEMS

Paul Burkard

08/11/2015

LAST TIME:

- NLP
- LSI

QUESTIONS?

I. RECOMMENDER SYSTEMS
II. CONTENT-BASED RECOMMENDATION
III. COLLABORATIVE FILTERING
HANDS-ON: RECOMMENDER SYSTEMS

- What are Recommender Systems?
    - Why do we need them?
    - What are some common use cases?
- What are the 2 main types of Recommender Systems?
    - How do they differ?
    - What are their respective strengths/weaknesses?

I. RECOMMENDER SYSTEMS

Q: *What are* ***Recommender Systems****?*

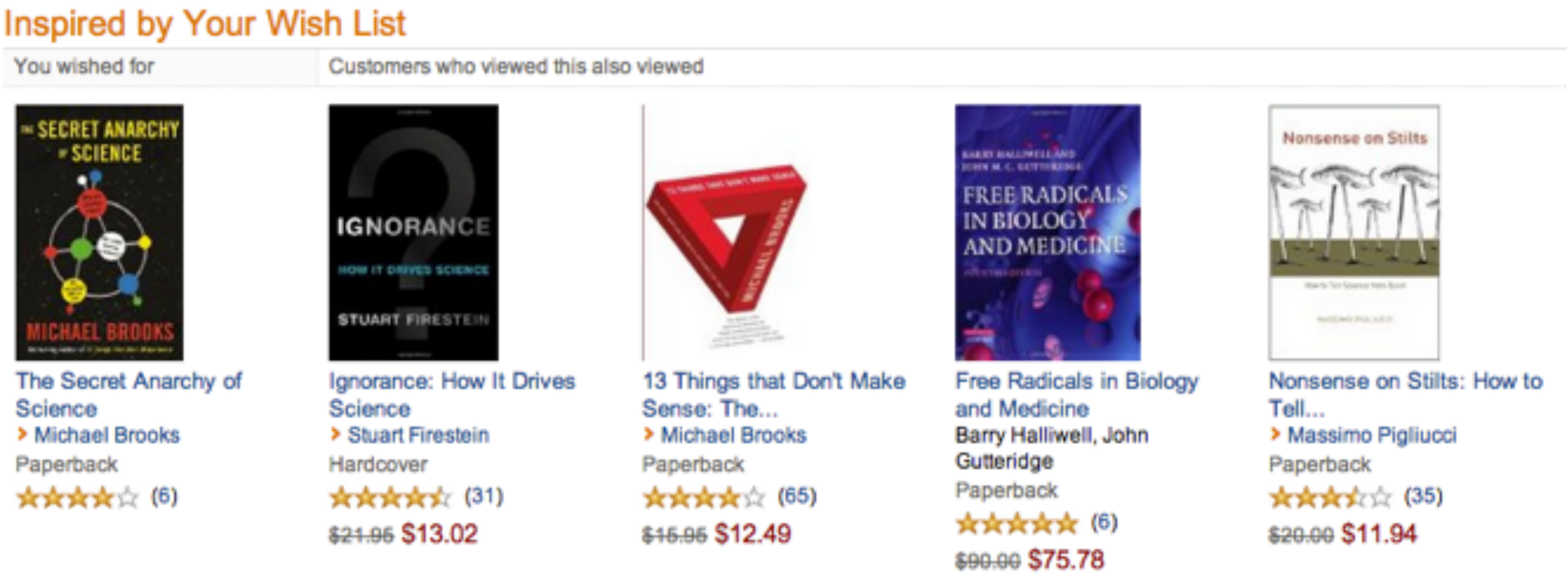
A: *Automated systems that seek to suggest whether a given* ***item*** *(product, event, movie, song, etc) will be desirable to a* ***user****.*

They often build on the back of machine learning concepts we've seen previously.

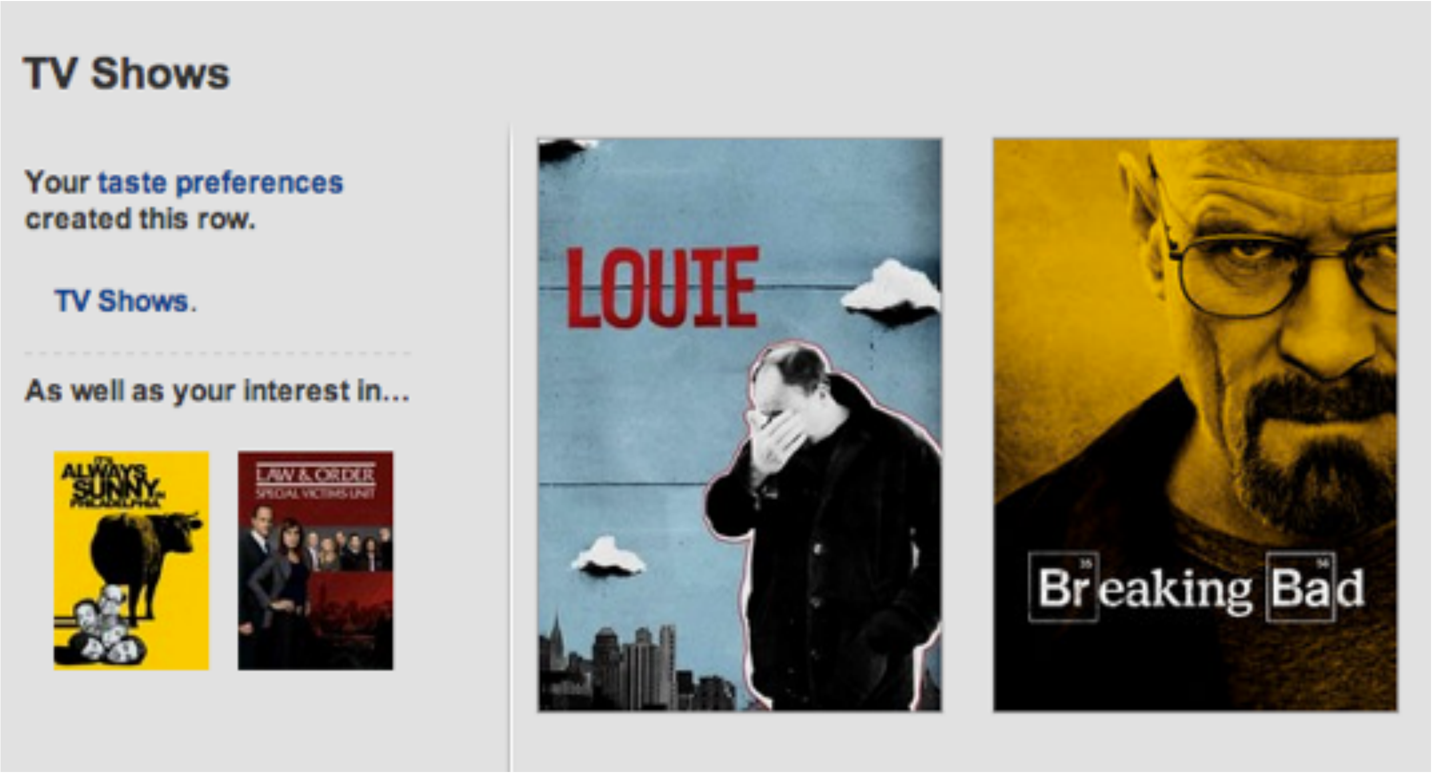
They've become ubiquitous in today's web-based world, so there are many different applications…

EXAMPLES — AMAZON

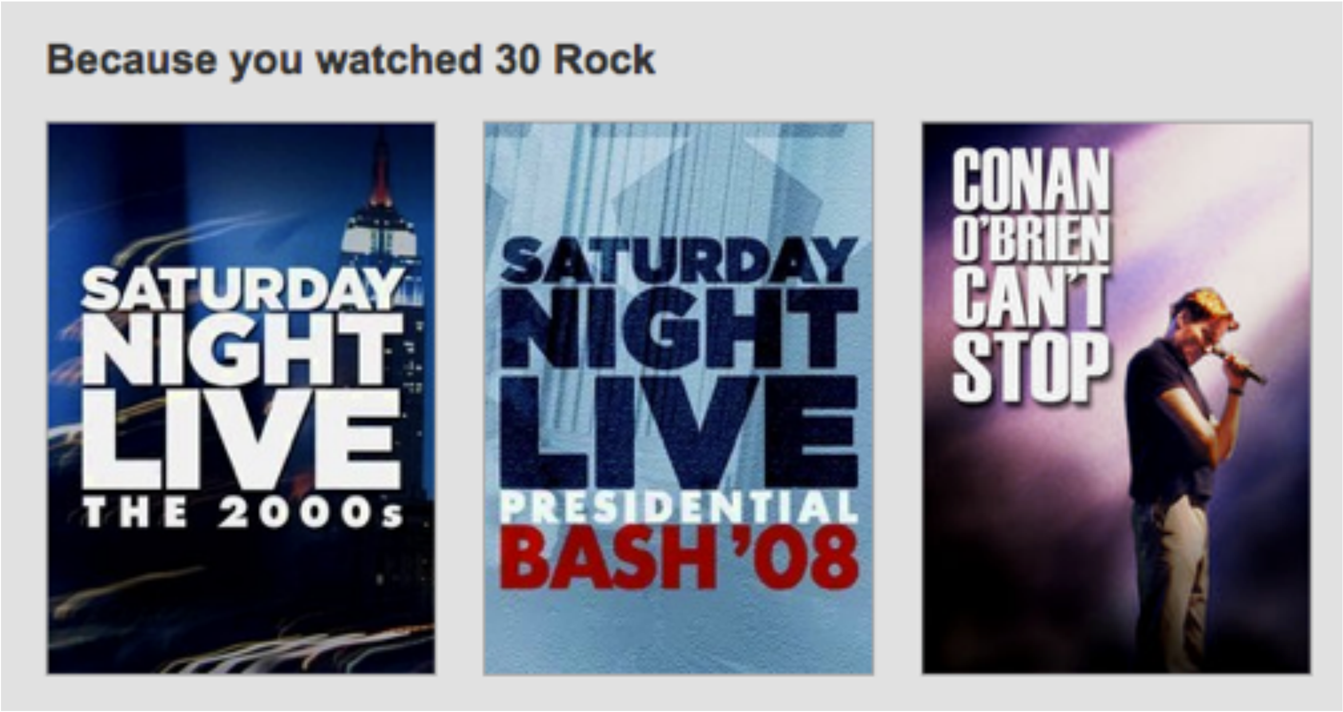
EXAMPLES – AMAZON



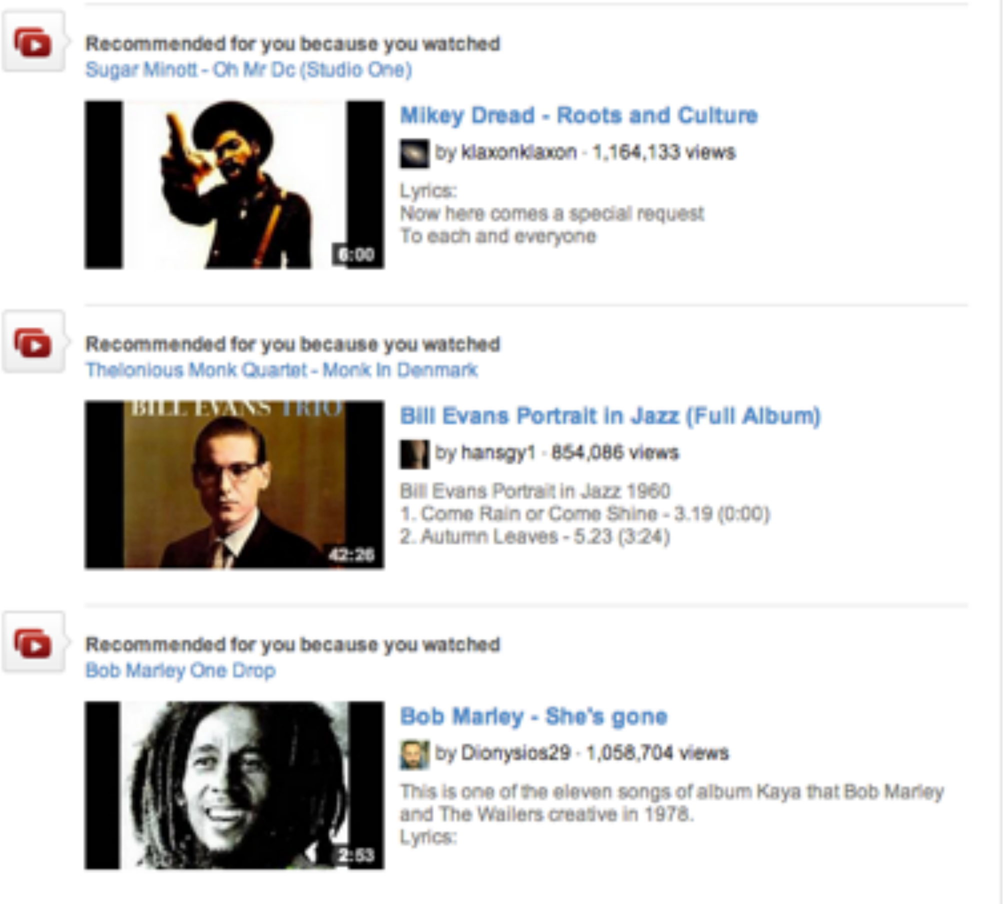
EXAMPLES – NETFLIX



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.

There are two general approaches to their 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.*

II. CONTENT-BASED RECOMMENDERS

Content-based recommendation *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

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):	*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)

2) Create features from user+item pairs and use ML algorithm (classifier for instance) 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

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, genre, artist etc. the user selects.

Content-based recommendation 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!)

III. COLLABORATIVE FILTERING

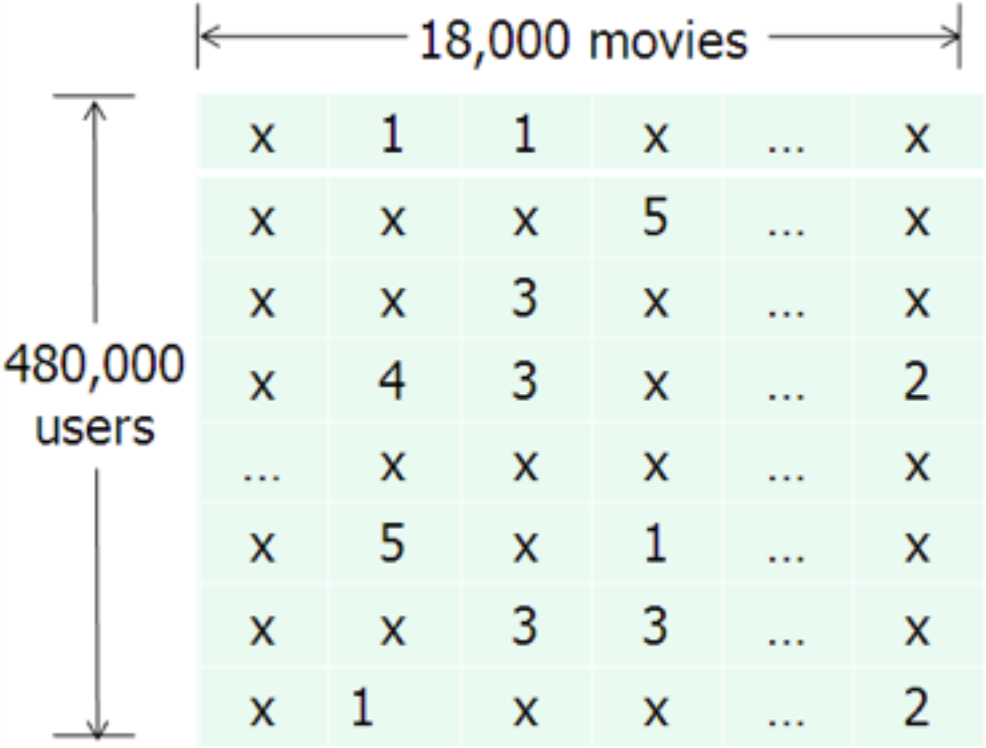
The purpose of a **recommendation system** *is to decide whether an item (product, event, movie, song) is something a user is highly likely to be interested in*

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.*

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



NOTE

This matrix will always be *sparse*!

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

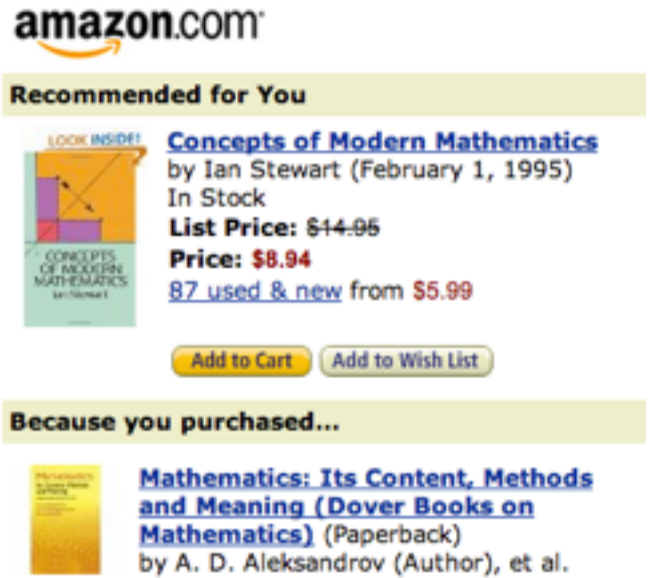
Collaborative filtering can be done in two different ways.

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

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.*

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.

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.

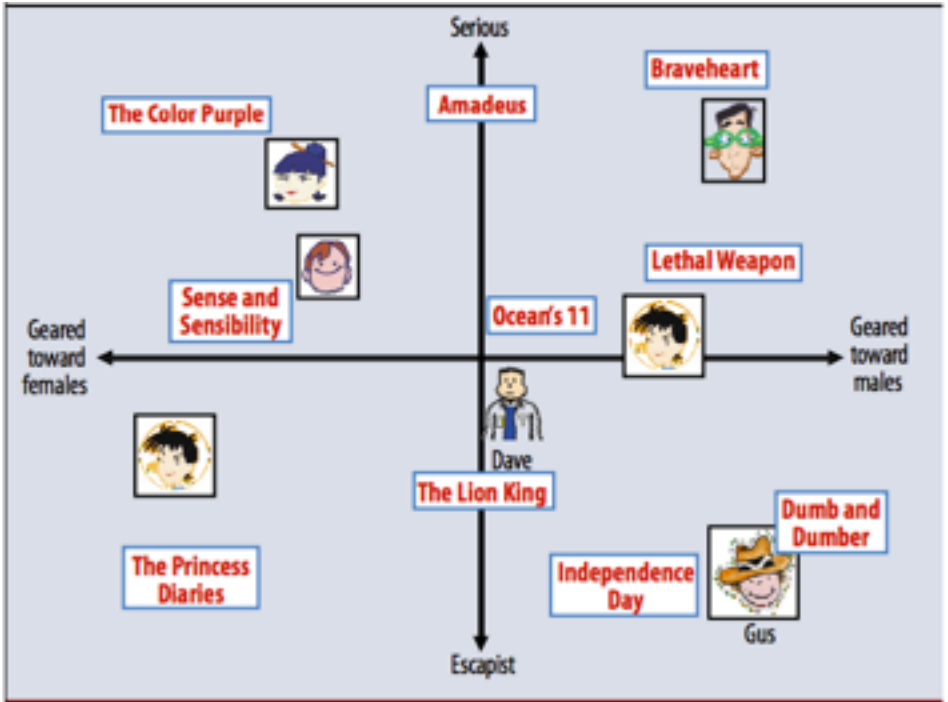


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

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.

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.*

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.

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).*

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.

HANDS-ON: RECOMMENDERS