GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 15: RECOMMENDATION SYSTEMS

YUCHEN ZHAO / DAT-14

LAST TIME:

- DIMENSION REDUCTION NONLINEAR METHODS
- IMBALANCED CLASSES
- EVALUATION

QUESTIONS?

I. CONTENT-BASED FILTERING
II. COLLABORATIVE FILTERING
III. A SIMPLE MATRIX FACTORIZATION MODEL
IV. THE NETFLIX PRIZE

EXERCISE:
V. RECSYS IN PYTHON

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

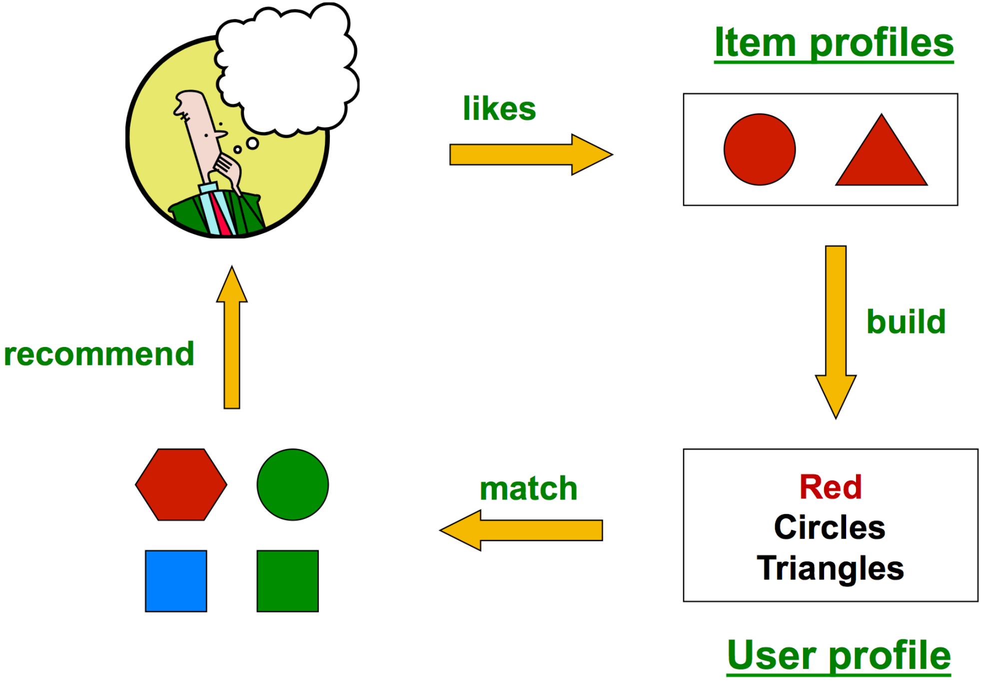
The purpose of a **recommendation system** *is to predict a rating that a user will give an item that they have not yet rated.*

This rating is produced by analyzing other user/item ratings (and sometimes item characteristics) to provide personalized recommendations to users.

Discussion:
Why do we need new methods for recommendation?

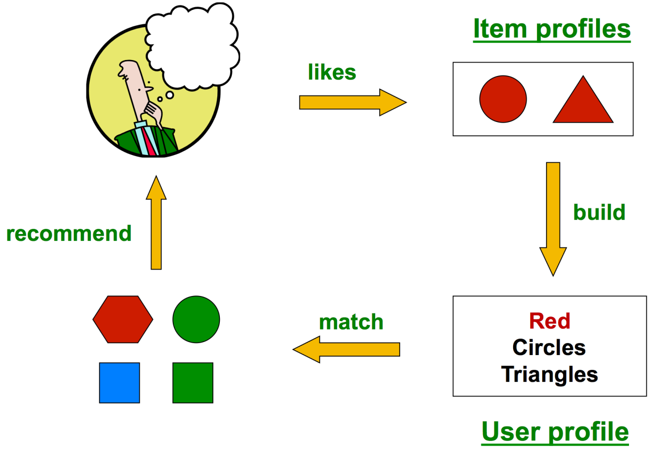
There are two general approaches to recsys design:

content-based filtering:

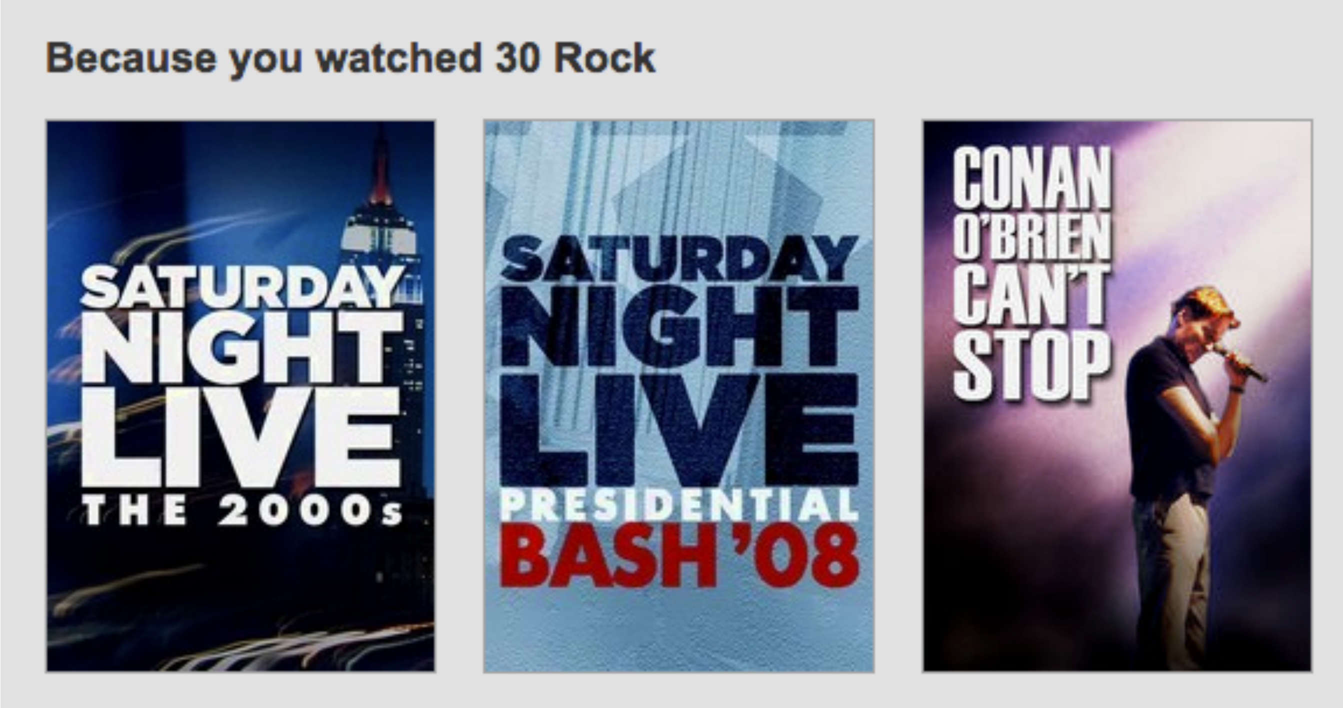


There are two general approaches to recsys design:

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



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


Because you watched 30 Rock
SATURDAY NIGHT LIVE
THE 2000s
SATURDAY NIGHT LIVE
PRESIDENTIAL
BASH '08
CONAN O'BRIEN CAN'T STOP

EXAMPLES – YOUTUBE

Recommended for you because you watched
Sugar Minott - Oh Mr Dc (Studio One)

Mikey Dread - Roots and Culture

by klaxonklaxon · 1,164,133 views

Lyrics:
Now here comes a special request
To each and everyone

Recommended for you because you watched
Thelonious Monk Quartet - Monk In Denmark

Bill Evans Portrait in Jazz (Full Album)

by hansgy1 · 854,086 views

Bill Evans Portrait in Jazz 1960
1. Come Rain or Come Shine - 3.19 (0:00)
2. Autumn Leaves - 5.23 (3:24)

Recommended for you because you watched
Bob Marley One Drop

Bob Marley - She's gone

by Dionysios29 · 1,058,704 views

This is one of the eleven songs of album Kaya that Bob Marley and The Wailers creative in 1978.
Lyrics:

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

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.

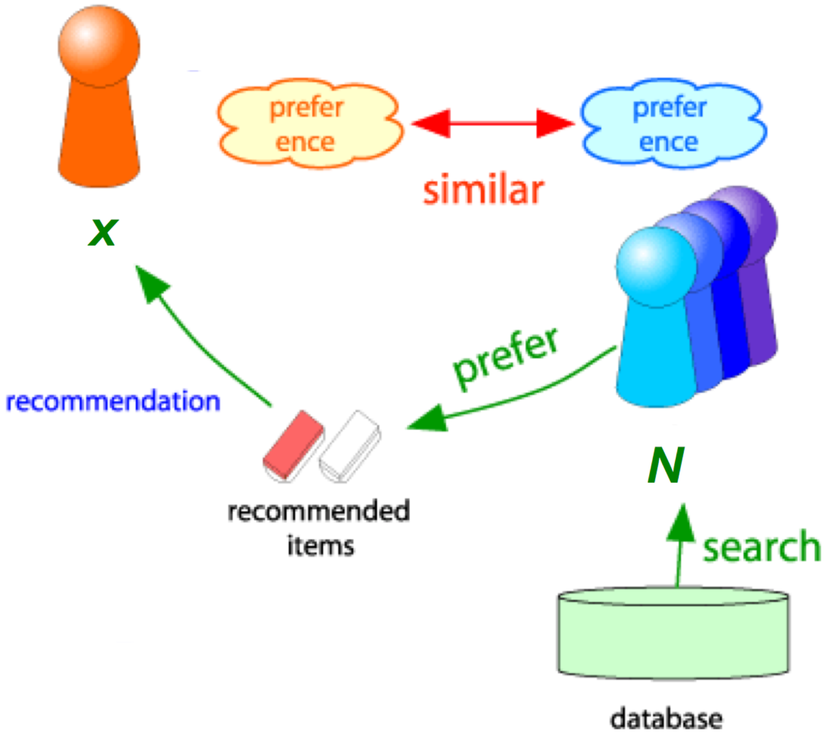
To search for additional articles on one of your most-read topics, click that topic on your personalized Recommendations page. To learn more about topic tags, visit Times Topics.

There are two general approaches to recsys design:

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

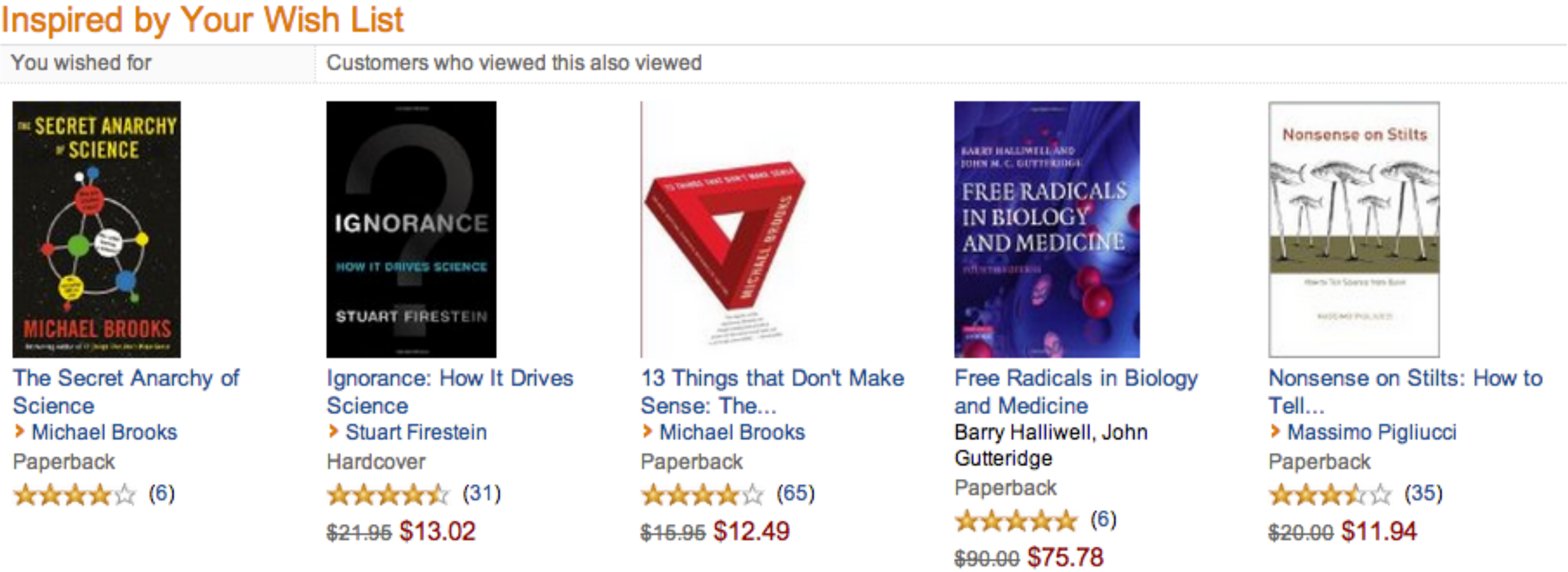
In contrast, the only data under consideration in **collaborative filtering** *are user-item ratings, and recommendations depend on user preferences.*

collaborative filtering:

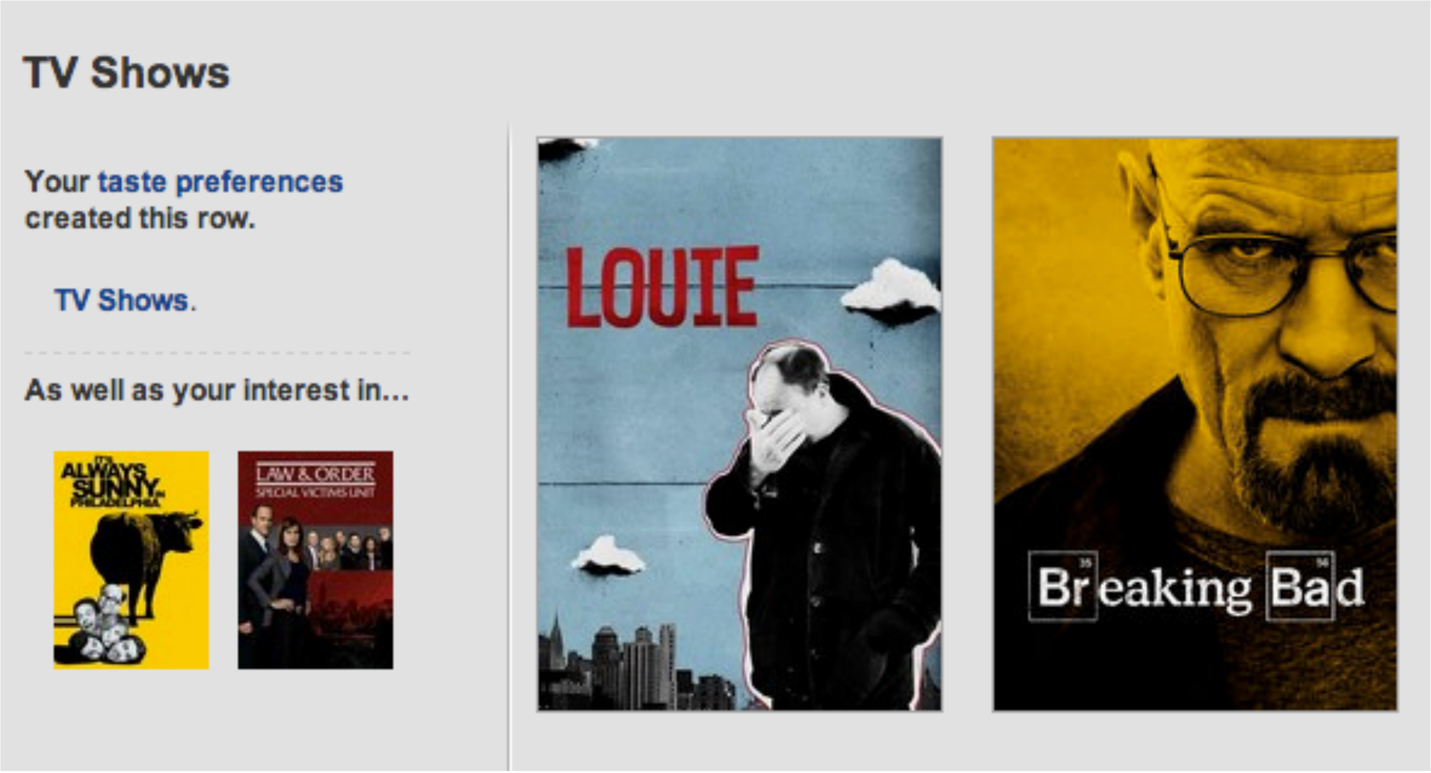


EXAMPLES – AMAZON

EXAMPLES – AMAZON

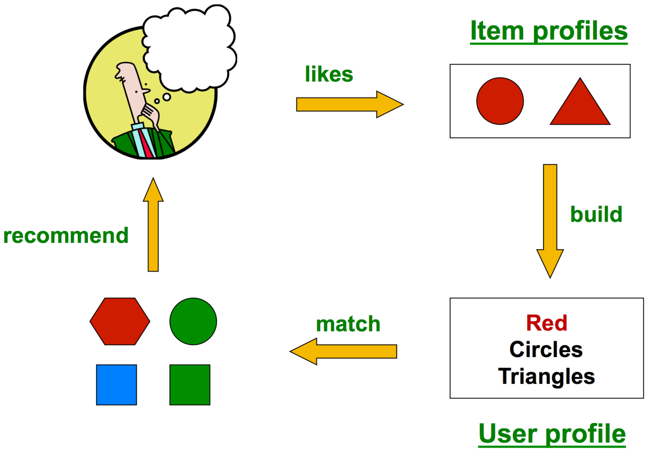


EXAMPLES – NETFLIX



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.*



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

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

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.

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

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.

Ratings are generated by taking dot products of user & item vectors.

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

Item vectors measure the degree to which the item is descr each feature, and user vectors measure a user's preference feature.

Ratings are generated by taking dot products of user & item vectors.

NOTE

The idea is that users like items that are similar to other items they've consumed.

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):

Finding Nemo $= (5, 5, 2)$

Mission Impossible $= (3, -5, 5)$

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

predicted ratings:*

$(-3*5 + 2*5 - 2*2) = -9$

$(-3*3 - 2*5 - 2*5) = -29$

$(3*4 - 2*5 + 2*5) = +12$

users:

Jason $= (-3, 2, -2)$

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)

predicted ratings:*

$(-3*5 + 2*5 - 2*2) = -9$

$(-3*3 - 2*5 - 2*5) = -29$

$(3*4 - 2*5 + 2*5$

NOTE (*)

In practice, these predictions would be proportional to deviations from some global average rating (hence the negative values).

EXAMPLE – CONTENT-BASED FILTERING

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

items (movies):	*predicted ratings*:*
Finding Nemo = (5, 5, 2)	*(4*5 - 3*5 + 5*2) = 15*
Mission Impossible = (3, -5, 5)	*(4*3 + 3*5 +5*5) = 52*
Jiro Dreams of Sushi = (-4, -5, -5)	*(-4*4 + 3*5 - 5*5) = -26*

users:

Bob = (4, -3, 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)

predicted ratings:*

*(4*5 - 3*5 + 5*2) = 15*

*(4*3 + 3*5 +5*5) = 52*

*(-4*4 + 3*5 - 5*5) = -26*

users:

Bob = (4, -3, 5)

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.

About The Music Genome Project®

We believe that each individual has a unique relationship with music – no one else has tastes exactly like yours. So delivering a great radio experience to each and every listener requires an incredibly broad and deep understanding of music. That's why Pandora is based on the Music Genome Project, the most sophisticated taxonomy of musical information ever collected. It represents over ten years of analysis by our trained team of musicologists, and spans everything from this past Tuesday's new releases all the way back to the Renaissance and Classical music.

Each song in the Music Genome Project is analyzed using up to 450 distinct musical characteristics by a trained music analyst. These attributes capture not only the musical identity of a song, but also the many significant qualities that are relevant to understanding the musical preferences of listeners. The typical music analyst working on the Music Genome Project has a four-year degree in music theory, composition or performance, has passed through a selective screening process and has completed intensive training in the Music Genome's rigorous and precise methodology. To qualify for the work, analysts must have a firm grounding in music theory, including familiarity with a wide range of styles and sounds.

source: http://www.pandora.com/about/mgp

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

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.

Collaborative filtering *refers to a family of methods for predicting ratings where instead of thinking about users and items in t*

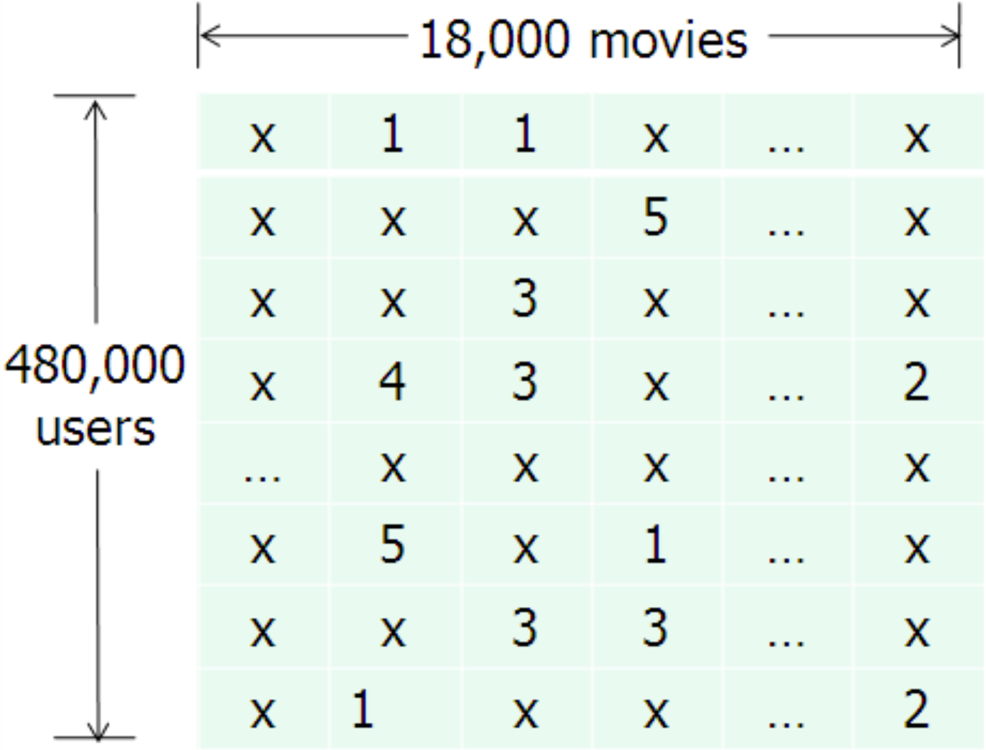
feature space, we are only interested in the existing user-it

themselves.

NOTE

The idea here is that users get value from recommendations based on other users with similar tastes.

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



x	1	1	x	...	x
x	x	x	5	...	x
x	x	3	x	...	x
x	4	3	x	...	2
...	x	x	x	...	x
x	5	x	1	...	x
x	x	3	3	...	x
x	1	x	x	...	2

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**.

Collaborative filtering can be done in two different ways.

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

> **NOTE**
>
> This is equivalent to a clustering problem in the space of column vectors (items).
>
> Item-based CF is a neighborhood method.

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

This is also called **memory-based CF**.

Collaborative filtering can be done in two different ways.

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

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

This is also called **memory-based CF***.*

NOTE

Th
cl
th
ve

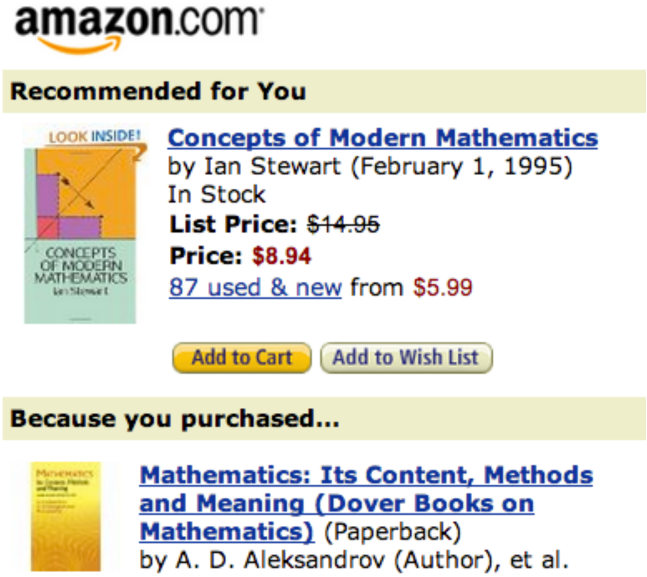
Ite
ne

NOTE

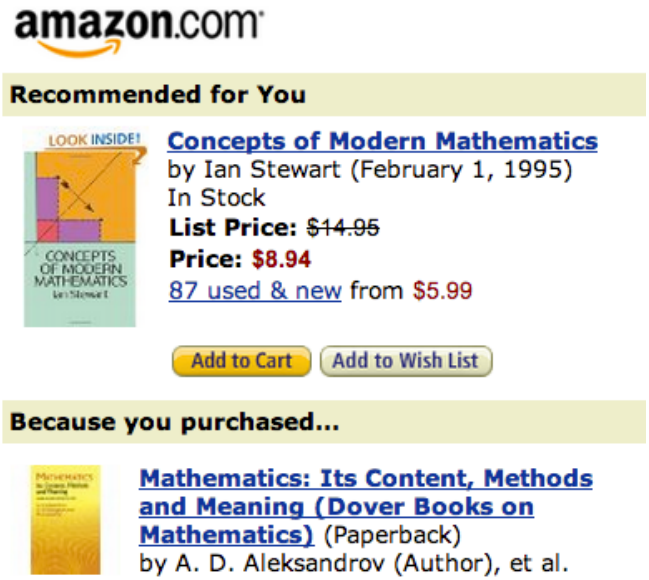
User-based collaborative filtering is possible but less efficient, since there are typically more users than items.


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.*

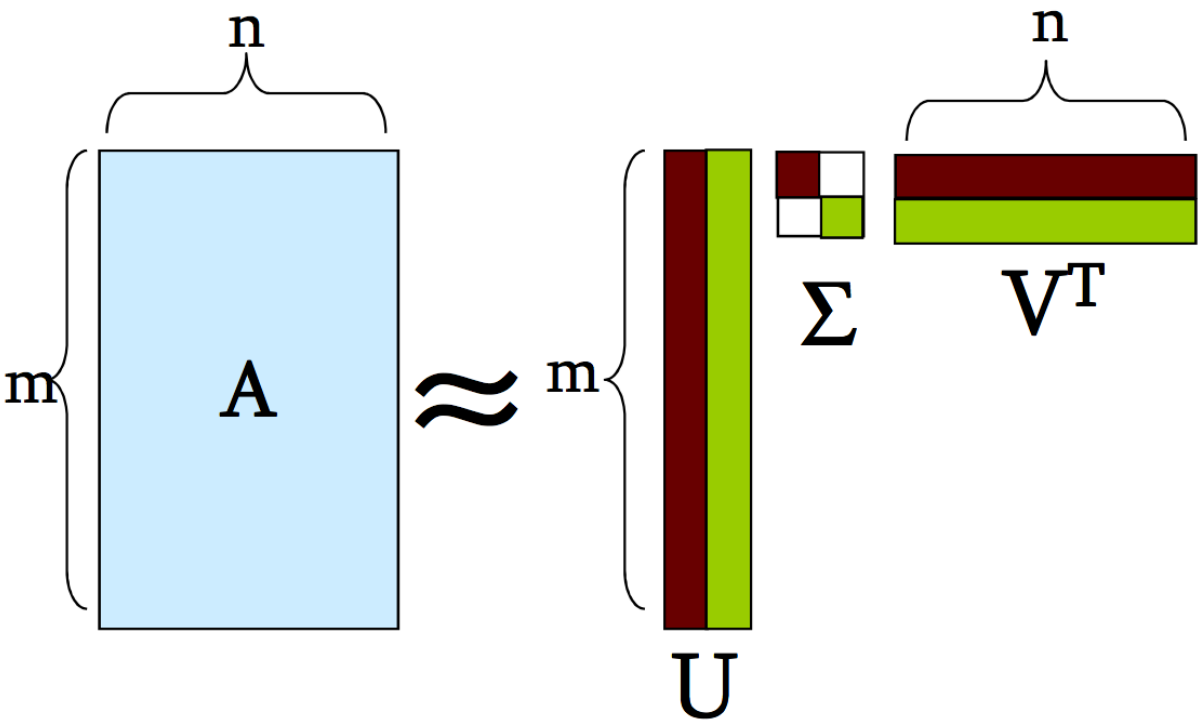
The most popular model-based CF techniques use matrix decomposition techniques to find deeper structure in the ratings data.

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

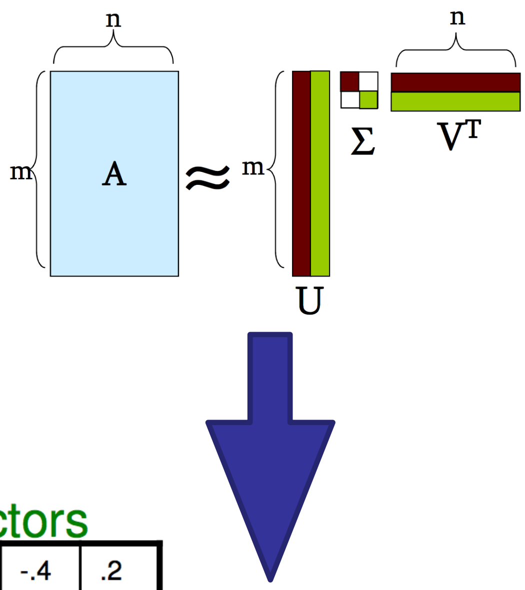
The most popular model-based CF techniques use matrix decomposition techniques to find deeper structure in the ratings data.

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

remember SVD?



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



R (items × users)

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

≈

Q (items × factors)

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

$\boldsymbol{P^T}$ (factors × users)

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6

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

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

R

≈

factors

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

Q

users

factors

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6

$\boldsymbol{P^T}$

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.

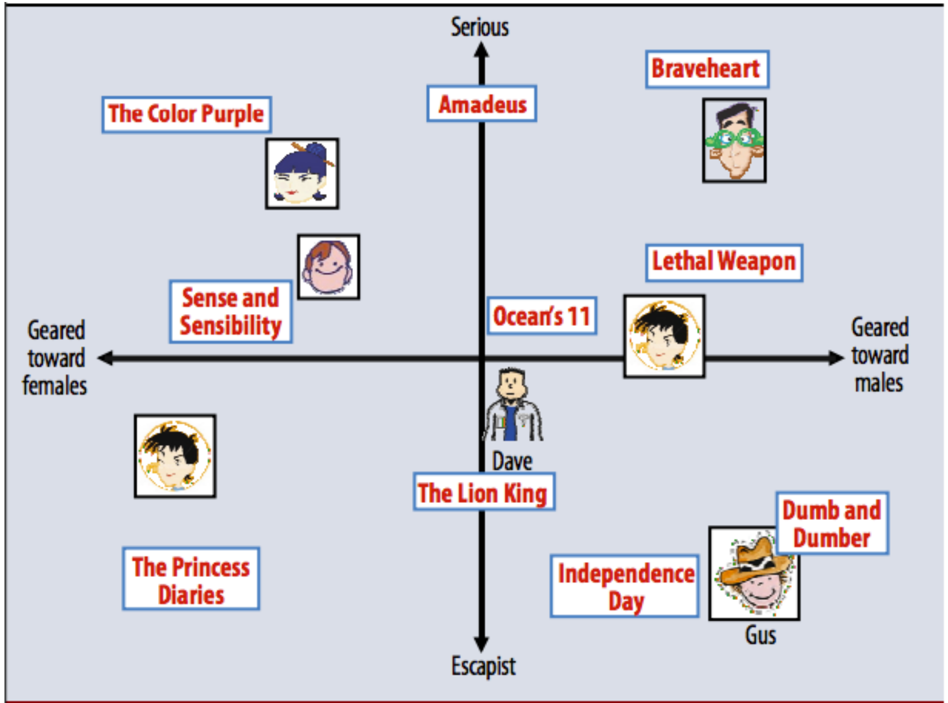


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

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)*
- *susceptible to fraud (eg shilling attacks)*
- **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.

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

$$\hat{r}_{xi} = q_i \cdot p_x$$
$$= \sum q_{if} \cdot p_{xf}$$

users

items

1		3			5			5		4	
		5	4	2.4		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

≈

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

f factors

Q

•

users

f factors

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

P^T

SIMPLE MATRIX FACTORIZATION MODEL

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.

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:

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

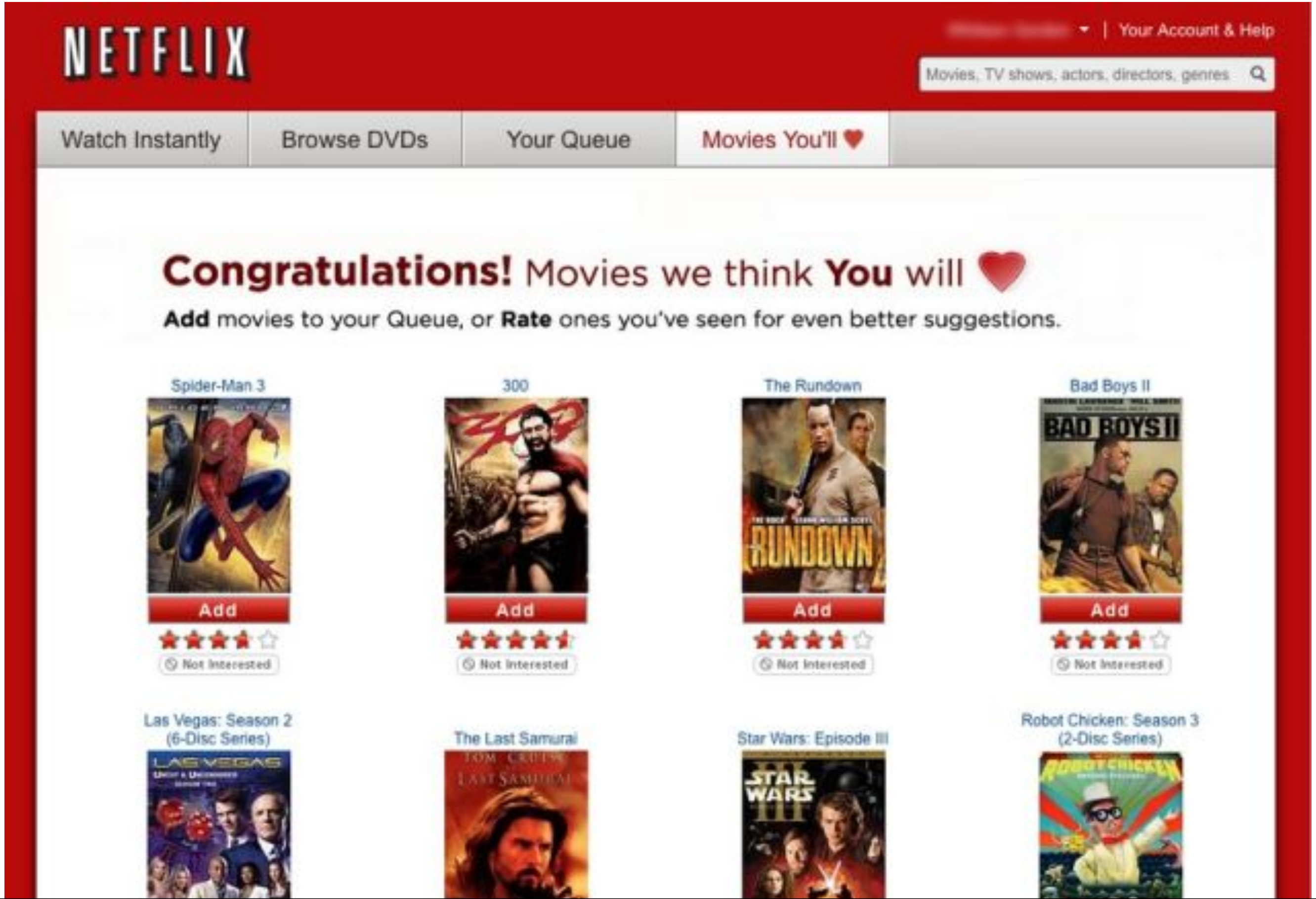
With this generalization, our minimization problem becomes:

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

Further modifications can be made to this model (incorporating implicit feedback, capturing temporal effects, attaching confidence scores to predictions), and you can look up the details in the references.

IV. THE NETFLIX PRIZE

NETFLIX PRIZE



award $1 million to anyone who can improve movie recommendation by 10%

The competition began in 2006, and the grand prize was eventually awarded in 2009. The winning entry was a stacked ensemble of 100's of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.

The competition began in 2006, and the grand prize was eventually awarded in 2009. The winning entry was a stacked ensemble of 100's of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.

Ultimately, the competition ended in a photo finish. The winning strategy came down to last-minute team mergers & creative blending schemes to shave 3rd & 4th decimals off RMSE (concerns that would not be important in practice).

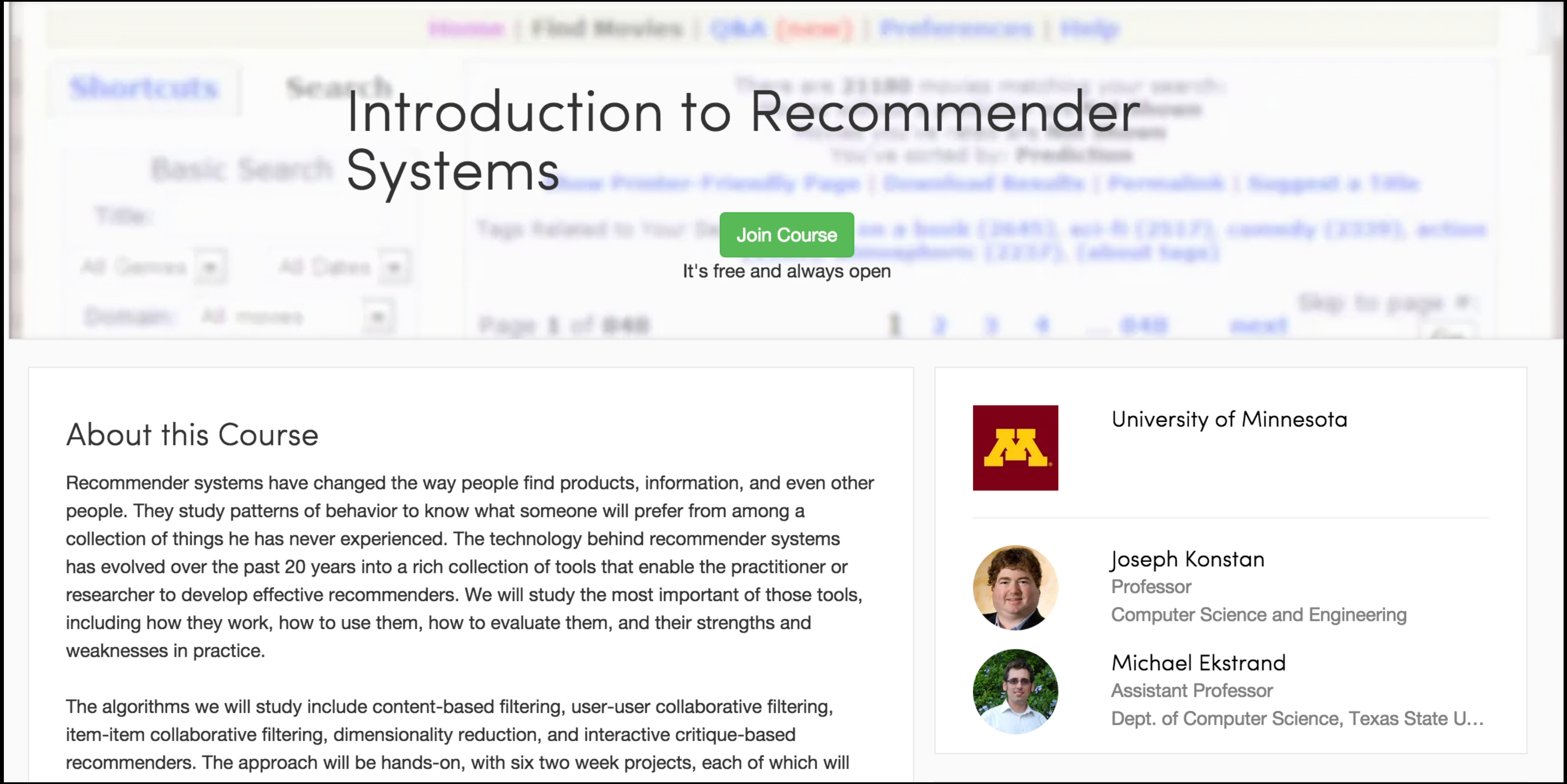
The competition began in 2006, and the grand prize was eventually awarded in 2009. The winning entry was a stacked ensemble of 100's of models (including neighborhood & matrix factorization models) that were blended using boosted decision trees.

Ultimately, the competition ended in a photo finish. The winning strategy came down to last-minute team mergers & creative blending schemes to shave 3^{rd} & 4^{th} decimals off RMSE (concerns that would not be important in practice).

The competition did much to spur interest and research advances in recsys technology, and the prize money was donated to charity.

Though they adopted some of the modeling techniques that emerged from the competition, Netflix never actually implemented the prizewinning solution.

Why do you think that's true?



Further learning material:
https://www.coursera.org/learn/recommender-systems