

CSCI 4022 Fall 2021

Collaborative Filtering (*k-Nearest Neighbors*)

Recall: Set for a Recommendation System


 X := set of customers S := set of items available to recommend R := set of ratings**Example:**

action script

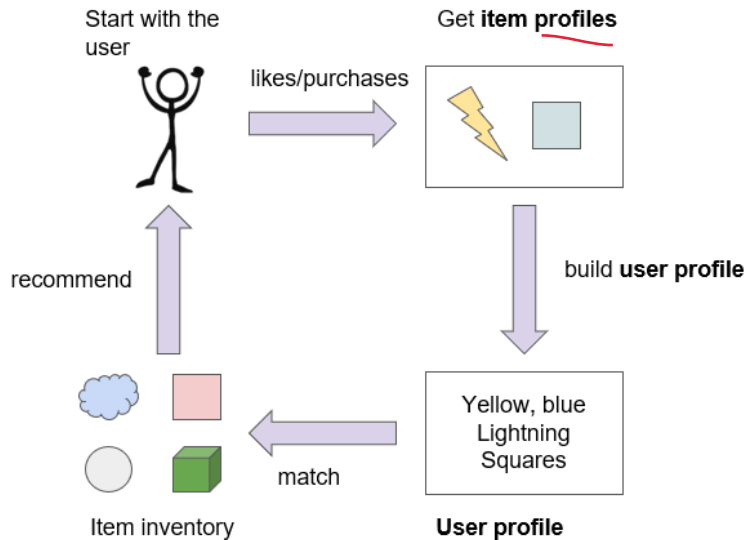
	Avatar	Lord of the Rings	Star Wars	The Matrix
Anya	1	X	0.2	0
Berto	0	0.5	0	0.3
Claudia	0.2	0	1	
Desmond	0		0	0.4

$U =$

Examples:

- © 2014 Ted Goff
KDnuggets Cartoon
- 
- “The machine learning algorithm wants to know if we’d like a dozen wireless mice to feed the Python book we just bought.”

Content-based system map



Creating an Item Profile

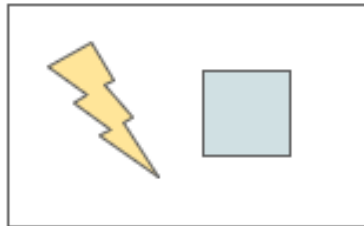
Step 1: Create a *profile* for each item.

Goal: construct a set of *features* for each item.

- ▶ What is important about it?
- ▶ What might make someone like/dislike?

Convenient to think of and represent as a vector.

Can be Boolean, or real-valued



Examples:

Movies: writer, actors, title, director, genre, ...

People: set of friends/connections, location, school/workplace, age, ...

Text: set of important words in the document/web page/etc.

Text Classifications

For text, one profile is set of “important” words (terms = features, docs = items)

So how do we pick which features are important? A Typical heuristic is the *term frequency* \times *inverse document frequency* (TF-IDF)

Denote:

$f_{ij} :=$ frequency of term (feature) i in document (item) j

$n_i :=$ # of documents that mention term i

$N :=$ total # of documents

Then the **term frequency** of term i in doc j is:

$$TF_{i,j} = \frac{f_{ij}}{\max_k f_{k,j}} = \frac{\text{frequency of word } i \text{ in } j}{\text{frequency of most common word in } j}$$

And the **inverse document frequency** of term i is:

$$IDF_i = \log \frac{N}{n_i} > 1$$

N large n_i small; word i is super rare

$$\text{TF-IDF score is } \boxed{TF_{ij} \times IDF_i}$$

$N/n_i \rightarrow$ huge

–document profile could be the set of words with the *highest* TF-IDF scores (possibly including their scores)

Text Classifications

For text, one profile is set of “important” words (terms = features, docs = items)

So how do we pick which features are important? A Typical heuristic is the *term frequency* \times *inverse document frequency* (*TF-IDF*)

Denote:

$f_{ij} :=$ frequency of term (feature) i in document (item) j

$n_i :=$ # of documents that mention term i

$N :=$ total # of documents

Then the **term frequency** of term i in doc j is:

$$TF_{i,j} = \frac{f_{ij}}{\max_k f_{k,j}} = \frac{\text{frequency of word } i \text{ in } j}{\text{frequency of most common word in } j}$$

And the **inverse document frequency** of term i is:

$$IDF_i = \log \frac{N}{n_i}$$

TF-IDF score is $\boxed{TF_{ij} \times IDF_i}$

–document profile could be the set of words with the *highest* TF-IDF scores (possibly including their scores)

Ideas:

–rarer words count for more
–density of word in a document matters

User Profiles

Based on **item profiles** of items the user has “liked” (rated, or purchased, or interacted with in some meaningful way), we need to construct a **user profile** for the user.

1. Suppose the user has rated items i_1, i_2, \dots, i_n
2. Simple approach: take the (weighted) average of rate item profiles
 - ▶ Can weight by the user's ratings for each items
 - ▶ Need to baseline/normalize users' ratings by subtracting off their overall mean rating, and divide only by the number of observations of that feature. This is **that users** average score *for each feature*. –Accounts for the fact that some users tend to give higher/lower ratings than others – **Example**: user's ratings of 1 and 5 become $1-(3)=-2$ and $5-(3)=+2$.
3. **Variant**: can *weight* by the difference from average ratings for item

User Profiles

Based on **item profiles** of items the user has “liked” (rated, or purchased, or interacted with in some meaningful way), we need to construct a **user profile** for the user.

1. Suppose the user has rated items i_1, i_2, \dots, i_n
2. Simple approach: take the (weighted) average of rate item profiles
 - ▶ Can weight by the user's ratings for each items
 - ▶ Need to baseline/normalize users' ratings by subtracting off their overall mean rating, and divide only by the number of observations of that feature. This is **that users** average score *for each feature*. –Accounts for the fact that some users tend to give higher/lower ratings than others – **Example**: user's ratings of 1 and 5 become $1-(3)=-2$ and $5-(3)=+2$.
3. **Variant**: can *weight* by the difference from average ratings for item

User Profiles

Example: Suppose Ming's Boolean utility matrix for her movie-viewing history is as follows. The only features in the item (movie) profiles are whether they starred Actor A or Actor B. Compute Ming's user profile using a simple averaging.

Movie	Actor A	Actor B
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1
Profile:	$\frac{2}{5}$	$\frac{3}{5}$

Calculation:

We take the average of each item's feature values to obtain Ming's profile's feature value in that component.

vs. the avg. of .5

$$\left[\begin{array}{cc} \frac{2}{5} - \frac{1}{2} & \frac{3}{5} - \frac{1}{2} \\ -0.1 & 0.1 \end{array} \right]$$

User Profiles

Example: Suppose Ming's Boolean utility matrix for her movie-viewing history is as follows. The only features in the item (movie) profiles are whether they starred Actor A or Actor B. Compute Ming's user profile using a simple averaging.

Movie	Actor A	Actor B
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1
Profile:	2/5	3/5

Calculation:

We take the average of each item's feature values to obtain Ming's profile's feature value in that component.

$$\text{Ming}[\text{Actor A}] = (1 + 1)/5 = 2/5$$

$$\text{Ming}[\text{Actor B}] = (1 + 1 + 1)/5 = 3/5$$

That's great and all, but we didn't incorporate whether she enjoyed the movies or not. Let's try again...

User Profiles

Example: Suppose Ming's Boolean utility matrix for her movie-viewing history is as follows. The only features in the item (movie) profiles are whether they starred Actor A or Actor B. Compute Ming's user profile using a simple averaging.

Movie	Actor A	Actor B
1	3	0
2	0	1
3	0	2
4	5	0
5	0	4
Profile:		

Calculation:

Ming's average rating is $(3 + 1 + 2 + 5 + 4)/5 = 3$, so we normalize all feature values (ratings) by subtracting 3.



mean
→

0 x
x -2
x -1
2 x
x 1

User Profiles

Example: Suppose Ming's Boolean utility matrix for her movie-viewing history is as follows. The only features in the item (movie) profiles are whether they starred Actor A or Actor B. Compute Ming's user profile using a simple averaging.

Movie	Actor A	Actor B
1	0	0
2	0	-2
3	0	-1
4	2	0
5	0	1
Profile:	1	-2/3

pick movies
+
(0)

Calculation:

Ming's average rating is $(3 + 1 + 2 + 5 + 4)/5 = 3$, so we normalize all feature values (ratings) by subtracting 3.

$$\text{Ming}[\text{Actor A}] = (0 + 2)/2 = 1 \quad \text{Ming}[\text{Actor B}] = (-2 + -1 + 1)/3 = -2/3$$

This is more helpful! All relative to 0, so we see that Ming has a positive view of movies with Actor A, and negative view for Actor B.

User Profiles

Ming's rating: $\text{Base line (3)} + 1 \cdot \text{"is actor"} + (-2) \cdot \text{"is actor"} + \text{B log}^n$

If this sounds a lot like a linear regression or an ANOVA... it can be!

One way to construct a user profile is to construct a *linear regression* of the features F_1, F_2, \dots, F_n **for each user**.

$$(\text{Log?}) \text{Prediction} = \beta_0 + \beta_1 \cdot F_1 + \beta_2 \cdot F_2 + \dots + \beta_n \cdot F_n$$

User Profiles

If this sounds a lot like a linear regression or an ANOVA... it can be!

One way to construct a user profile is to construct a *linear regression* of the features F_1, F_2, \dots, F_n **for each user**.

$$(Log?-)Prediction = \beta_0 + \beta_1 \cdot F_1 + \beta_2 \cdot F_2 + \dots + \beta_n \cdot F_n$$

...but this is pretty costly! This is a **for each user** n loop, and within that is a **for each feature** m model, which tends to be an $\mathcal{O}(n \cdot m^3)$ kind of thing. So reserve it for when m is small.

Also, what to do if a user has less than m total ratings? Sounds scary (e.g. overfitted, non-unique) to fit a linear model with more features than observations!

User Profiles

If this sounds a lot like a linear regression or an ANOVA... it can be!

One way to construct a user profile is to construct a *linear regression* of the features F_1, F_2, \dots, F_n **for each user**.

$$(Log?-)Prediction = \beta_0 + \beta_1 \cdot F_1 + \beta_2 \cdot F_2 + \dots + \beta_n \cdot F_n$$

Best practice: use only the **largest** β terms for each person. Not discussed here, but this is common in machine learning and prediction. Many classifiers/predictors *penalize* extra terms. Both Logistic and least-squares regression have common penalties to avoid using too many terms as well: see LASSO, ridge regression.

Making Predictions

Goal: Given a user profile x and item profile i , we want to compute *similarities* in high-dimensional space.

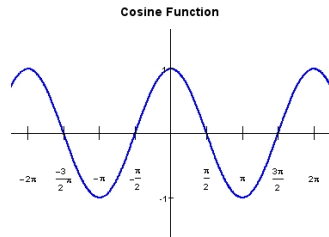
Recall the cosine distance between x and i is given by θ from:

$$\cos \theta = \frac{\vec{x} \cdot \vec{i}}{|\vec{x}| \cdot |\vec{i}|}$$

Handwritten notes: "similarity $x \cdot i$ " (in red) and "lies in $[-1, 1]$ " (in blue) with a bracket under the fraction.

But also:

- As $\theta \rightarrow 0$, $\cos \theta \rightarrow 1$, so x and i are more similar.
- As θ increases, $\cos \theta \rightarrow -1$, so x and i are less similar.
- So this is a *similarity* measure between vector objects.



Making Predictions

Prediction: The degree to which a user likes an item is the similarity between that item's profile and their user profile!

Movie	Actor A	Actor B	Rating
1	1	0	3
2	0	1	1
3	0	1	2
4	1	0	5
5	0	1	4
6	1	0	?
7	0	1	?
8	1	1	?
Profile:	1	-2/3	

Recommendation: We can compare the user's profile x to the item profiles i in our database, and recommend items that have the highest cosine similarity.

Calculation:

Suppose movies 6-8 become available. What are their similarities with Ming, based on the appearance of Actors A and B in those movies.

Making Predictions

Prediction: The degree to which a user likes an item is the similarity between that item's profile and their user profile!

Movie	Actor A	Actor B	Rating
1	1	0	3
2	0	1	1
3	0	1	2
4	1	0	5
5	0	1	4
6	1	0	A
7	0	1	B
8	1	1	C
Profile:	1	-2/3	

Calculation:

$$\begin{aligned} \cos \theta_A &= \frac{\langle 1, -2/3 \rangle \cdot \langle 1, 0 \rangle}{\sqrt{1 + 4/9} \cdot \sqrt{1}} = \frac{1}{\sqrt{13/9} \cdot \sqrt{1}} \\ \cos \theta_B &= \frac{\langle 1, -2/3 \rangle \cdot \langle 0, 1 \rangle}{\sqrt{1 + 4/9} \cdot \sqrt{1}} = \frac{-2/3}{\sqrt{13/9} \cdot \sqrt{1}} \\ \cos \theta_C &= \frac{\langle 1, -2/3 \rangle \cdot \langle 1, 1 \rangle}{\sqrt{1 + 4/9} \cdot \sqrt{2}} = \frac{1/3}{\sqrt{13/9} \cdot \sqrt{2}} \end{aligned}$$

$$\sqrt{1 + 4/9} = \sqrt{13/9}$$

Of course, computing *tons* of cosine similarities between the user profile and each item profile can be a big task in itself. There are some approaches to speed this up that we won't get into here, including locality-sensitive-hashing, and random hyperplane searches.

Costs of Item-Item

Pros:

1. No need for data on other users.
2. Able to recommend to users with unique tastes (even without similar other users)
3. Able to recommend new or unpopular items (don't depend on others' ratings)
4. Readily available explanations for recommended items (using the features)

Cons:

1. Finding appropriate features can be difficulty (e.g., music, images, movies – genres!)
2. Overspecialization – might never recommend items outside the user's profile
3. Doesn't exploit quality judgment by other users – even wildly popular content won't be recommended if it doesn't fit the user's profile
4. Cold-start problem – new users need to start with some kind of “average” profile

User-User recommendations

The big alternative to item-based recommendations is to instead recommend content based on similar other *users'* preferences. Sounds familiar?

Plan: k "nearest neighbors"

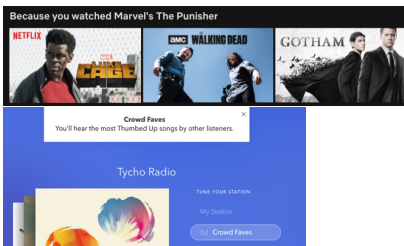
Find a set N of other users whose ratings are *similar* to user x 's ratings
 k of them, $|N| = k$

Define: N = the neighborhood of user x

Estimate x 's unknown ratings for new items based on the ratings of other users in N

Example: Consider users A and B, with ratings vectors
 $r_A = [1, 0, 0, 1, 3]$ and $r_B = [1, 0, 2, 2, 0]$ (where 0 := unknown). We need a notion of similarity between users, $sim(A, B)$.

$$2/4 = .5$$



Option 1: Jaccard similarity

Drawback: ... ?

User-User recommendations

The big alternative to item-based recommendations is to instead recommend content based on similar other *users'* preferences. Sounds familiar?

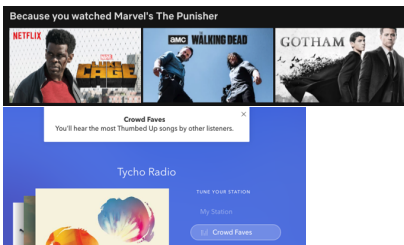
Plan:

Find a set N of other users whose ratings are *similar* to user x 's ratings

Define: N = the neighborhood of user x

Estimate x 's unknown ratings for new items based on the ratings of other users in N

Example: Consider users A and B, with ratings vectors $r_A = [1, 0, 0, 1, 3]$ and $r_B = [1, 0, 2, 2, 0]$ (where 0 := unknown). We need a notion of similarity between users, $sim(A, B)$.



Option 1: Jaccard similarity

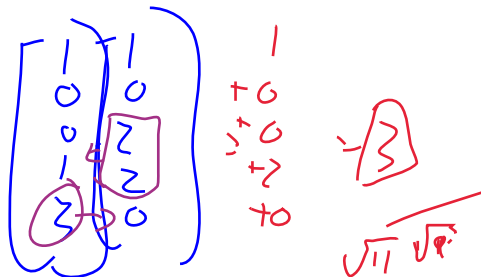
Drawback: ... ? Jaccard tells us *what* is similar, not *ratings*

Collaborative Filtering

Example: Consider users A and B, with ratings vectors $r_A = [1, 0, 0, 1, 3]$ and $r_B = [1, 0, 2, 2, 0]$ (where $0 := \text{unknown}$). We need a notion of similarity between users, $\text{sim}(A, B)$.

Option 2: cosine similarity

Drawback:



Collaborative Filtering

Example: Consider users A and B, with ratings vectors $r_A = [1, 0, 0, 1, 3]$ and $r_B = [1, 0, 2, 2, 0]$ (where $0 := \text{unknown}$). We need a notion of similarity between users, $\text{sim}(A, B)$.

Option 2: cosine similarity

Drawback: Treats missing ratings as negative (1 is the lowest rating, so 0 is terrible!)

Option 3:

Collaborative Filtering

Example: Consider users A and B, with ratings vectors $r_A = [1, 0, 0, 1, 3]$ and $r_B = [1, 0, 2, 2, 0]$ (where $0 := \text{unknown}$). We need a notion of similarity between users, $\text{sim}(A, B)$.

Option 2: cosine similarity

Drawback: Treats missing ratings as negative (1 is the lowest rating, so 0 is terrible!)

Option 3:

Collaborative Filtering

Example: Consider users A and B, with ratings vectors $r_A = [1, 0, 0, 1, 3]$ and $r_B = [1, 0, 2, 2, 0]$ (where $0 := \text{unknown}$). We need a notion of similarity between users, $\text{sim}(A, B)$.

Option 2: cosine similarity

Drawback: Treats missing ratings as negative (1 is the lowest rating, so 0 is terrible!)

Option 3: *centered* cosine distance: normalize ratings by subtracting off the user's mean and still treating unknowns as 0

Centered Cosine Similarity

Calculation: normalize ratings by subtracting off the user's mean and still treating unknowns as 0. Now if we compute $r_A \cdot r_B$, we're multiplying their ratings-versus-personal-average against one another. Each multiplication is positive if both user's agree that the movie is **either** "above average" **or** "below average"

1. Now missing values are treated as neutral! (0 = the average)
2. And handles differences in how different users tend to rate items (*personal averages*)
3. Sum of ratings in any user's row vector of ratings is 0

Does this look familiar? It's equivalent to the equivalent to Pearson correlation coefficient:

$$r_{XY} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

= dot $(X - \bar{X}) \cdot (Y - \bar{Y})$

In other words, this is a measure of how well the scores for one user *correlate* to the scores of another user.

Collaborative Filtering

Prediction based on the nearest (by ratings) neighbors is called *collaborative filtering*.

Goal: To predict user x 's unknown rating for item i .

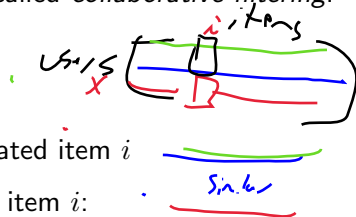
Let $r_x =$ vector of user x 's ratings *row*

Let $N =$ set of k users most similar to user x who have rated item i

Prediction: Option 1: Simple average of ratings in N for item i :

$$\hat{r}_{x,i} = \frac{1}{k} \sum_{y \in N} r_{y,i}$$

*x is similar user to x :
 $y \in N$*



Collaborative Filtering

Prediction based on the nearest (by ratings) neighbors is called *collaborative filtering*.

Goal: To predict user x 's unknown rating for item i .

Let r_x = vector of user x 's ratings

Let N = set of k users most similar to user x who have rated item i

Prediction: Option 1: Simple average of ratings in N for item i :

$$\hat{r}_{x,i} = \frac{1}{k} \sum_{y \in N} r_{y,i}$$

Option 2: Average of ratings in N for item i , *weighted* by user similarity to x :

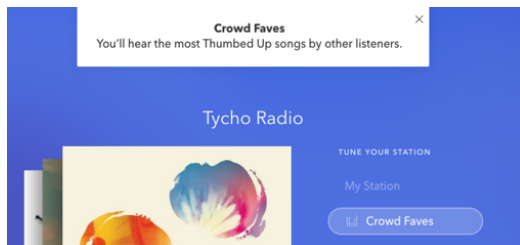
$$\hat{r}_{x,i} = \frac{1}{\sum_{y \in N} \text{sim}(x, y)} \sum_{y \in N} \text{sim}(x, y) r_{y,i}$$

Handwritten notes: ".75" above the sum in the denominator and ".75" below the sum in the numerator, with a red box around the similarity term in the numerator.

This is good! Accounts for similarities between users. Many other possibilities too. . .

Item-Item Filtering

So far, we have been talking about user-user collaborative filtering (using other similar users' ratings to predict whether a given user will like something)



There is an analogous view we could take: item-item collaborative filtering

For item i , find other similar items

Estimate rating for i based on ratings for similar items of that user

Can use all the same similarity and prediction functions as in our user-user model!

Filtering Example

Example: Suppose the utility matrix for users A-F and Movies 1-12 is given below. Ratings are between 1 (bad) and 5 (good), and blanks indicate unknown ratings. Use neighborhood of size $|N| = 2$ and centered cosine similarity to estimate $r_{A,5}$.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3			5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

option use-use:

find 2 rows like row

A: out of C, D, E, F

find 2 that maximize

$\text{Cosine sim}(A, X)$

Filtering Example

Process to estimate $r_{A,5}$

- User-User** Check set of *rows* with entries for 5. See how similar they are to user *A*. Pick most similar 2, and *weighted average* their scores for 5.
- Item-Item** Check set of *columns* with ratings by user *A*. See how similar their ratings are to item 5's. Pick most similar 2, and *weighted average* their scores by user *A*.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3		X	5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

cols: 1, 3, 6, 9, 11
 $\text{sim}(\text{col}, 5)$
 find max 2, then avg.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3		X	5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

User-user example:

1. Demean each row.
2. Find the $k = 2$ "nearest neighbors" N_A to user A that have rated 5.
3. Predict their $r_{A,5}$ by averaging these.

A $\begin{matrix} 1 \text{ high} \\ 5 \end{matrix} \rightarrow \begin{matrix} 2 \text{ low} \\ 5 \end{matrix}$
 B $\begin{matrix} 3 \text{ low} \\ 4 \text{ high} \end{matrix}$
 C $\begin{matrix} 1 \text{ low} \\ 5 \text{ high} \end{matrix}$

$\rightarrow \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix} \cdot \begin{bmatrix} -5/3 \\ 1/3 \\ 4/3 \end{bmatrix} = \text{very negative}$

$r_{A,5}$ $\rightarrow 3$

$\bar{A} - 13/5$ $2/5$ $+7/5$ $+7/5$ $+1/5$

	1	2	3	4	5	6	7	8	9	10	11	12
A	1 ⁻⁻⁻		3 [→]		X	5 ⁺⁺			5 ⁺⁺		4 ⁺	←
B			5	4			4			2	1	3
C	2 ⁻⁻⁻	4		1	2		3		4	3	5	
D		2	4 ⁺		5			4			2	
E			4 ⁺	3	4	2					2	5
F	1 ⁻⁻⁻		3 ⁻⁻⁻		3			2			4	

User-user example:

1. Demean each row.
2. Find the $k = 2$ "nearest neighbors" N_A to user A that have rated 5.
3. Predict their $r_{A,5}$ by averaging these.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3		X	5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

User-user example:

1. Demean each row.
2. Find the $k = 2$ “nearest neighbors” N_A to user A that have rated 5.
3. Predict their $r_{A,5}$ by averaging these.

The two most similar users are C and F , with $s_{A,C} = 0.41$ and $s_{A,F} = 0.59$. Then the estimate is

$$\begin{aligned}\hat{r}_{A,5} &= \frac{\sum_N \text{sim}(A, N_A) r_{N_A,5}}{\sum_N \text{sim}(A, N_A)} \\ &= \frac{1}{0.41 + 0.59} (0.41 \cdot 2 + 0.59 \cdot 3)\end{aligned}$$

$= 2.6$.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3		X	5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

Item-item example:

1. Demean each column.
2. Find the $k = 2$ “nearest neighbors” to item 5 that have been purchased by user A .
3. Predict their $r_{A,5}$ by averaging these.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3		X	5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

Item-item example:

1. Demean each column.
2. Find the $k = 2$ “nearest neighbors” to item 1 that have been purchased by user A .
3. Predict their $r_{A,5}$ by averaging these.

	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3		X	5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

Item-item example:

1. Demean each column.
2. Find the $k = 2$ “nearest neighbors” to item 1 that have been purchased by user A .
3. Predict their $r_{A,5}$ by averaging these.

The two most similar items are 3 and 9, so...

$$\hat{r}_{A,5} = \frac{\text{sim}(5, 3) \cdot r_{A,3} + \text{sim}(5, 9) \cdot r_{A,9}}{\text{sim}(5, 3) + \text{sim}(5, 9)}$$

Rec' System Wrapup

We have discussed 3 options to choose between, which we'll briefly discuss/finish up next lecture.

1. Item/user profile based recommendations (predicted cosine similarity)
2. User-user collaborative filtering (k-nearest neighbors by rating)
3. Item-item collaborative filtering (k-nearest neighbors by rating)

Item-item vs user-user? In practice, item-item typically works better. Why? Items are simpler than users. Items belong to small genres, whereas users are diverse. Item similarity is more meaningful than user similarity.

Choices may also depend on fundamental assumptions:

1. Do users buy items similar to other items in the catalogue, or are they exclusive/self-avoiding (things you only buy *once*).
2. Do we have more users or more items? Usually safer to use the ratings we have more of for prediction!

Advanced Recs

Often, the best results are found by implementing two or more different recommender engines and combining predictions

1. Can combine using a linear model (e.g. $pred = wgt_1 * pred_1 + wgt_2 * pred_2 + \dots$)
2. Can combine a global baseline estimate with collaborative filtering

This can combining content-based methods to collaborative filtering!

1. Get item profiles to solve the new item problem
2. Get demographic information/average profiles to solve the new user problem

Rule of thumb and common practice: *always* combine some notion of overall mean ratings into user recommendations!

Hybrid Recs

Suppose Janice has never rated any movie similar to The Sixth Sense, but we want to estimate her rating for it.

Define:

μ := overall mean movie rating

b_x := rating deviation of user x = (average rating of user x) - μ

b_i = rating deviation of item i = (average rating of item i) - μ

Baseline estimate of $r_{x,i}$, user x 's unknown rating of item i . Suppose...

Janice's mean movie rating overall: 3.7 stars

The Sixth Sense is 0.5 stars above average

Janice rates 0.2 stars below average

Baseline estimate is:

$$b_{x,i} = \mu + b_x + b_i$$

Hybrid Recs

Suppose Janice has never rated any movie similar to The Sixth Sense, but we want to estimate her rating for it.

Define:

μ := overall mean movie rating

b_x := rating deviation of user x = (average rating of user x) - μ

b_i = rating deviation of item i = (average rating of item i) - μ

Baseline estimate of $r_{x,i}$, user x 's unknown rating of item i . Suppose...

Janice's mean movie rating overall: 3.7 stars

The Sixth Sense is 0.5 stars above average

Janice rates 0.2 stars below average

Baseline estimate is:

$$b_{x,i} = \mu + b_x + b_i = 3.7 - 0.2 + 0.5 = 4$$

Coding Best Practice

Note that we can seamlessly put this into our full model: *demean* each user, then *recenter* at the end.

Baseline estimate: $b_{x,i} = \mu + b_x + b_i$

Combine this baseline estimate with collaborative filtering:

Old (CF-only):

$$\hat{r}_{x,i} = \frac{1}{\sum_{y \in N} \text{sim}(x, y)} \sum_{y \in N} \text{sim}(x, y) r_{y,i}$$

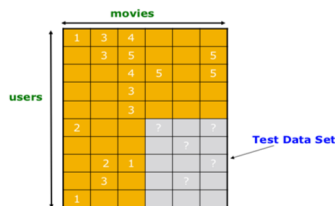
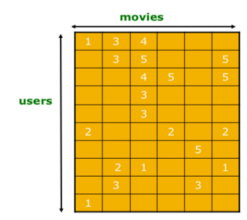
New (combined method):

$$\hat{r}_{x,i} = b_{x,i} \frac{1}{\sum_{y \in N; x} \text{sim}(x, y)} \sum_{y \in N; x} \text{sim}(x, y) (r_{y,i} - b_{x,j})$$

Evaluations

For any system where we try to *predict* missing information, a common approach in machine learning is to create a **holdout set**.

Process: take a piece of the utility matrix (user-item-rating matrix) and withhold it from the rating estimation process as a test set, T . Then, compare predictions in T against the withheld known ratings T .



We need a *distance* between the predictions and the true values as a score of *goodness*. One is:
root-mean-squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(x,i) \in T} (r_{x,i} - \hat{r}_{x,i})^2}$$

Scoring Evaluations

Narrow focus on prediction accuracy can miss the point:

- ▶ Prediction diversity

Example: If a user likes Star Wars a lot, they might only be recommended Star Wars and very similar movies.

- ▶ Prediction context

Example: If a user buys a bunch of travel guides for an upcoming trip, then goes on the trip, no longer needs the travel recommendations.

- ▶ Order of predictions

Example: Should recommend users to watch Star Wars I: The Phantom Menace, before Star Wars II: Attack of the Clones, but Phantom Menace was terrible, so might not get recommended much.

In practice, we only care to predict high ratings anyway. So we care more about measuring how well our system does for matching high ratings

Scoring Evaluations

There are alternative to RMSE.

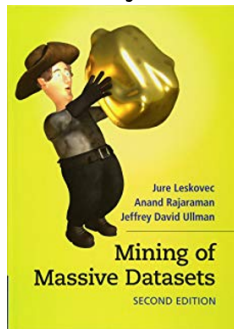
1. Other *metrics*: **precision at top 10%**, where we measure how often we correctly predict users' top 10 lists. Also **rank** correlation, e.g. $\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$ where d_i is difference in *rank* between predicted/actual.
2. Other *validation schema*. These include cross-validation like "Leave One Out (LOO)," where we fit the model many time. Remove a datum. Fit your model, then ask how well you predicted exactly that one datum without using it. Repeat over *all* datum. This is expensive but *very* appealing in theory compared to breaking the data into regular/holdout or training/prediction sets.

That concludes our discussion of recommendation systems, but we're not going far. The questions of predicting missing values or similarities on *matrix* structures is coming along. And we want some asides in linear algebra before we go any deeper.

Acknowledgments

Next time: Working with these item profiles, and user-based recommendations!

Some material is adapted/adopted from Mining of Massive Data Sets, by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University) <http://www.mmds.org>



Special thanks to Tony Wong for sharing his original adaptation and adoption of slide material.