Introduction to Data Science

COLLABORATIVE FILTERING

BRIAN D'ALESSANDRO

2 TYPES

User based

Find like users and make recommendations based on ratings/scores of like users

Item Based

Find similar items and recommend items similar to an item a user has shown interest in.

USER BASED

INTUITION

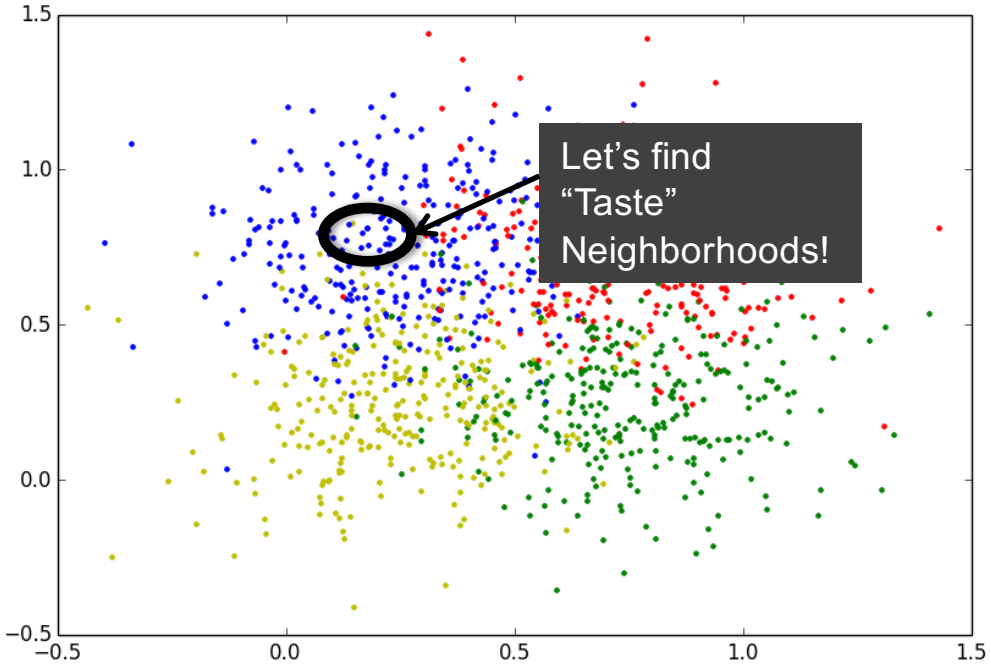
1. Find a group of people who like the same things similarly
2. Not everyone will like the exact same set of things

INTUITION

1. Find a group of people who like the same things similarly
2. Not everyone will like the exact same set of things
3. Recommend the non-overlapping items.

DEFINING A MECHANISM FOR "WE HAVE SIMILAR TASTES"

How do we translate "Find a group of people who like the same things" into a data science algorithm?



TOWARDS A "TASTE" NEIGHBORHOOD

First let's define the data structure.

Let A be the user-item matrix. Each entry a_{ij} can either be a rating or some binary indicator for user i on item j.

$A =$

	Item 1	Item 2	Item 3	Item 4	…	Item K
User 1	2	1			…	
User 2		2	4		…	2
User 3	3				…	
User 4	1	2	5	3	…	
User 5	3	2			…	
User 6					…	1
User 7		4	1		…	4
User 8		4	2		…	5
User 9	1				…	
User 10			3	4	…	1
…	…	…	…	…	…	
User N				1	…	4

TOWARDS A "TASTE" NEIGHBORHOOD

Second let's create a neighborhood for user i.

$A =$

	Item 1	Item 2	Item 3	Item 4	…	Item K
User 1	2	1			…	
User 2		2	4		…	2
User 3	3				…	
User 4	1	2	5	3	…	
User 5	3	2			…	
User 6					…	1
User 7		4	1		…	4
User 8		4	2		…	5
User 9	1				…	
User 10			3	4	…	1
…	…	…	…	…	…	
User N				1	…	4

TOWARDS A "TASTE" NEIGHBORHOOD

To do this, we need to define user-user similarity or distance.

$A =$

	Item 1	Item 2	Item 3	Item 4	…	Item K
User 1	2	1			…	
User 2		2	4		…	2
User 3	3				…	
User 4	1	2	5	3	…	
User 5	3	2			…	
User 6					…	1
User 7		4	1		…	4
User 8		4	2		…	5
User 9	1				…	
User 10			3	4	…	1
…	…	…	…	…	…	
User N				1	…	4

TOWARDS A "TASTE" NEIGHBORHOOD

Each user i corresponds to a row in our user-item matrix A. There are two popular ways to define similarity between similar rows of A.

Let A_i and A_k be two row vectors corresponding to the items i and k have rated. Let S be the set of items both users have rated/selected. Then,

Cosine Similarity

$$sim(A_i, A_k) = cos(A_i, A_k) = \frac{A_i \cdot A_k}{\|A_i\|_2 \|A_k\|_2}$$

Pearson Correlation

$$sim(A_i, A_k) = \frac{\sum_{j \in S} (A_{ij} - \mu_i)(A_{kj} - \mu_k)}{\sqrt{\sum_{j \in S} (A_{ij} - \mu_i)^2 \sum_{j \in S} (A_{kj} - \mu_k)^2}}$$

A PRACTICAL ASIDE

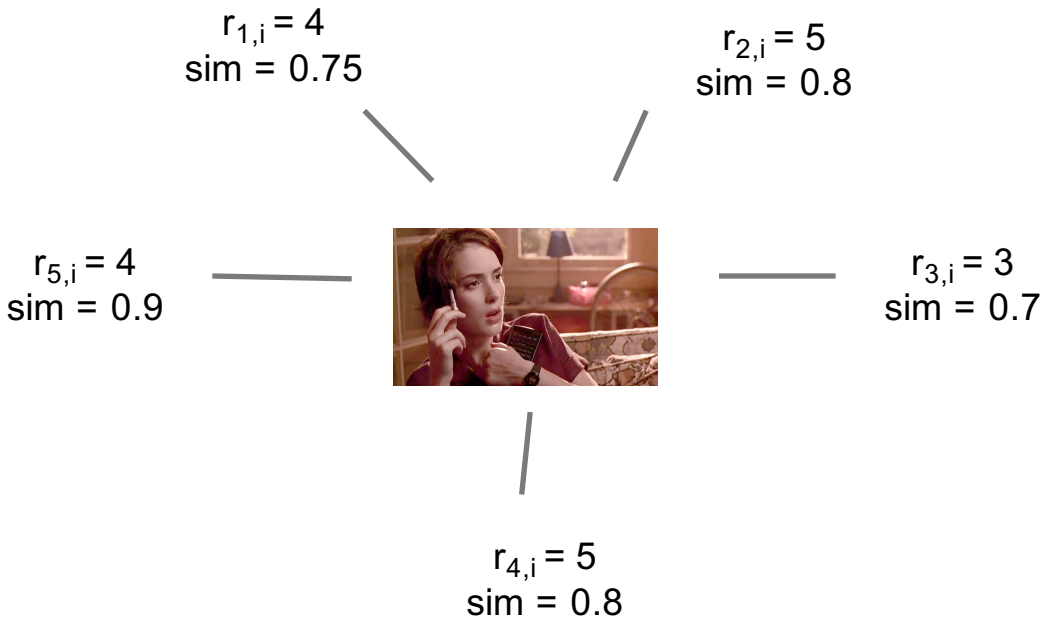
Things to consider when choosing a similarity measure:

- Which metric is better for ratings vs. binary indicators?
- Should you mean normalize? I.e., subtract user and/or item average rating from each rating.
- Should you take Similarity over all items for each user, or just those in common. I.e, should S be the intersection or union of A_i and A_k.

The right answer will likely depend on your problem.
Testing and experimentation is important in each case.

MAKING THE RECOMMENDATION

The predicted score/rating for user *u* on product *i* is then a function of scores/ratings that all users in u's neighborhood gave to the same product.



$$r_{u,i} = Agg_{u' \in U}(r_{u',i})$$

DIFFERENT WAYS TO AGGREGATE

Take a simple average.

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

Take a weighted avg, weighted by similarity…

$$r_{u,i} = \frac{1}{k} \sum_{u' \in U} sim(u, u') * r_{u',i}$$

$$k = \sum_{u' \in U} sim(u, u')$$

There are many other ways to define the aggregation function. Other variants use averages but normalize out the means of the individual users to account for user-specific biases. The right method is an empirical choice that is likely dependent on the application

THE FINAL PREDICTION

Once you have defined the neighborhood, aggregation is pretty straightforward.

$r_{1,i} = 4$
sim = 0.75

$r_{2,i} = 5$
sim = 0.8

$r_{3,i} = 3$
sim = 0.7

$r_{4,i} = 5$
sim = 0.8

$r_{5,i} = 4$
sim = 0.9

User	ri,j	sim	ri,j*sim
1	4	0.75	3.0
2	5	0.8	4.0
3	3	0.6	1.8
4	4	0.8	3.2
5	5	0.9	4.5

Average	**4.20**
Weighted Average	**4.29**

ITEM BASED

DEVELOPED BY AMAZON

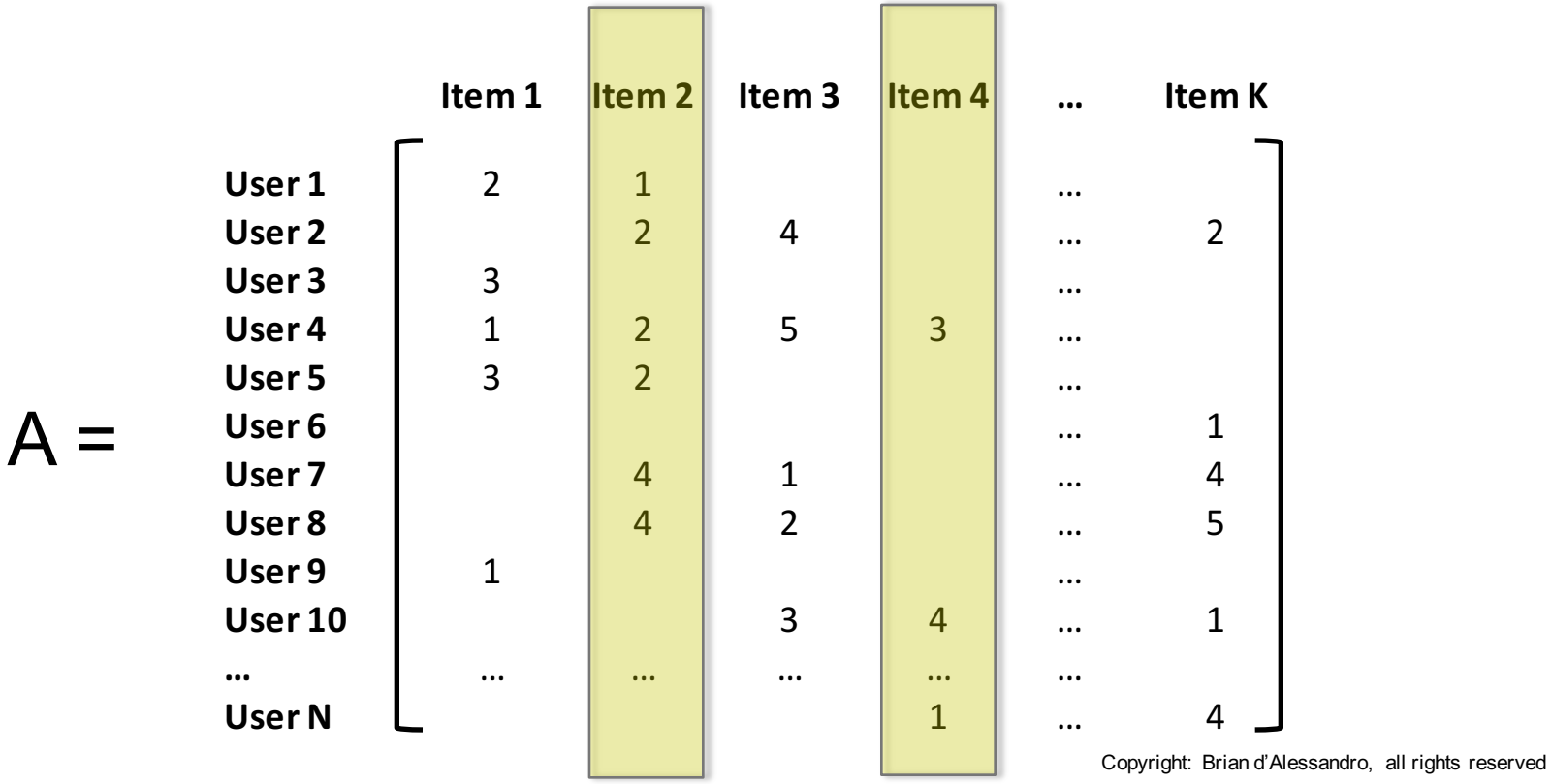
User based methods can be unscalable as user-item matrix grows, as there are usually more users than items.

As a general tactic, item based methods are useful for recommending based on the current, or last few items a user has consumed/viewed.

I.e., given the user is consuming/viewing an item (like a video), find a set of items that are most similar to the current one and recommend those.

START WITH USER-ITEM MATRIX

But this time we care about similarity between columns, not rows. We can use the same type of similarity functions that we used in the user based system.



DERIVE THE ITEM-ITEM MATRIX

2 approaches to recommendation:

1. If a user has selected/purchased an item, find the k most similar items

2. For each item the user hasn't selected/purchased, predict user's rating/score for that product as a function of the user's rating/score on similar items (similar to the user based kNN approach)

M =

	Item 1	Item 2	Item 3	Item 4	…	Item K
Item 1		.8	0.2	0	…	0.1
Item 2			0.5	0.11	…	0.6
Item 3				0.3	…	0.4
Item 4					…	0.8
…	…	…	…	…	…	0.3
Item K					…	