Recommender System.

Labeling your data:

- Explicit feedbacks: Ratings. thumbs up/down, purchase/conversion.
- Implicit feedbacks: consumption of an item

Usually explicit feedbacks is better for understand a user's true taste, but implicit feedbacks is often more abundant due to the ease of collecting it.

(using only explicit feedback give us low bias. but we have high variance due to little data, so we need implicit feedback)

Bias in data collection:

- Rank bias:
  items higher up in the higher rank are more likely to get more engagement, independent of how relevant they are.
  1. can randomize top K results to average out rank effect
  2. can use rank as a feature and control for this
     (give all the result a rank of 1)
- Exploration Bias
  We only receive feedback on the items shown, so we risk only being able to learn on items we've ranked highly.
  1. we can insert less relevant or newer items into the

shown lists to get feedbacks on them.. (more exploration.)

(otherwise we never get feedbacks on these items, even if they are relevant)

Limit the scale of exploration: Because with more exploration, you are likely to affect the user experience and reduce the KPI.

Collaborative filtering:

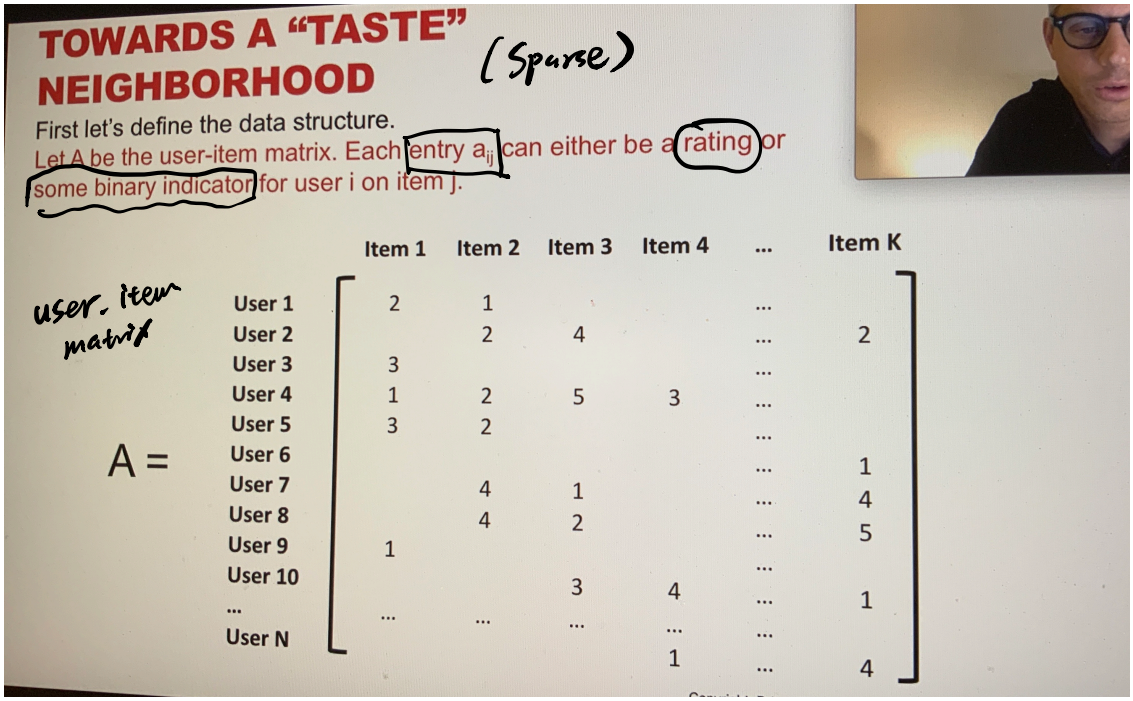
User Based

Intuition: 1. find a group of people who like things similarly.

2. Not everyone will like exact same set of things

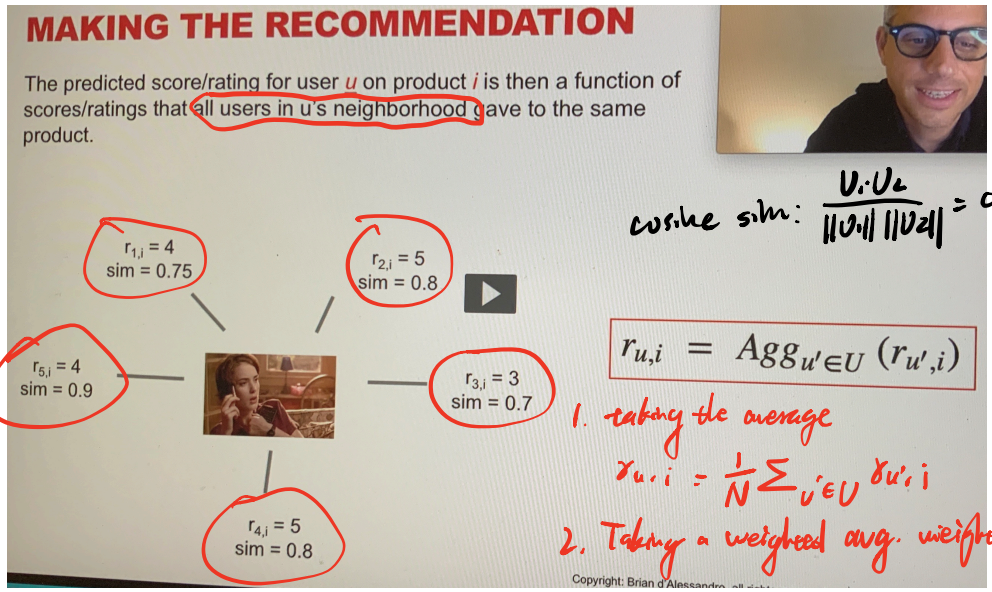
3. Recommend the non-overlapping items.

Find a "taste" neighborhood



the similarity between 2 users:

1. cosine similarity. 2. pearson correlation.



cosine sim: $\frac{U_1 \cdot U_2}{\|U_1\| \|U_2\|} = \cos\theta$

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

1. taking the average

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

2. Taking a weighted avg. weighted by similarity

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

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

Average: $\frac{1}{5} \times (4+5+3+4+5) = 4.2$

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

$= 0.8 + 0.7 + 0.8 + 0.75 + 0.9 = 3.95$

$\frac{1}{3.95} \times (4 \times 0.75 + 5 \times 0.8 + \cdots + 4 \times 0.9) = 4.22$

Item Based

Given the user is consuming/viewing an item, find a set of items that are most similar to the current one and recommend those.

Reasons: 1. users >> items

2. we know nothing about new user (cold start), but it's important to build relationship at the beginning.

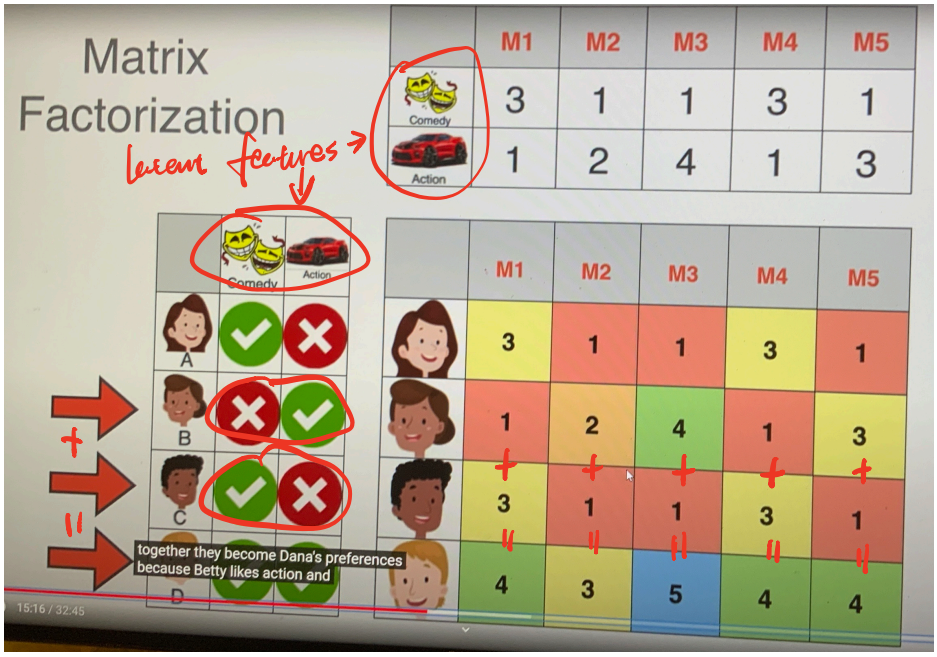
From a user-item matrix

↓

similarity between items

Derive a user-item matrix.

Matrix factorization:

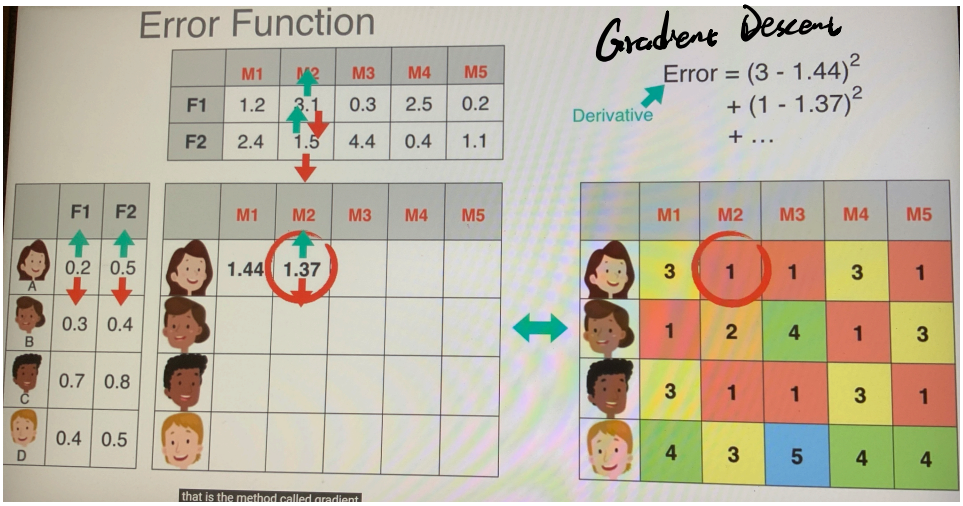


We express the user-item matrix in terms of two smaller matrices, and we can see the dependency in the smaller matrix (user preference of B and C = preference of D.) translate into the large matrix (the rating of B and C = the rating of D)

Benefit:

1. Storage (store two smaller matrices instead of large one)

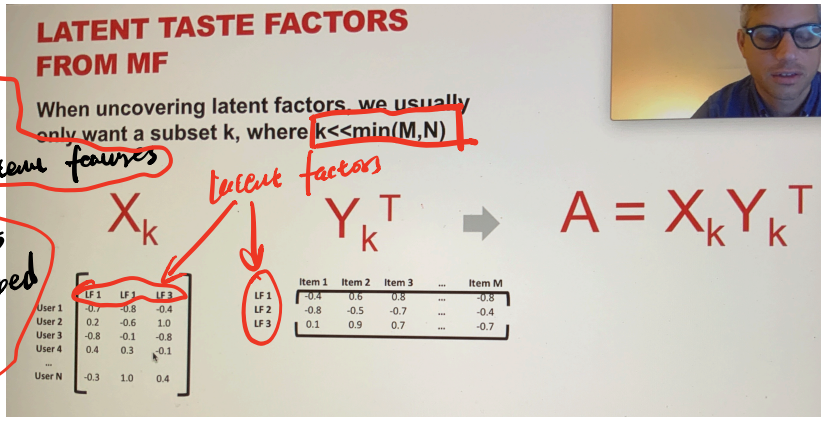
find the matrix factorization



$A = XY^T$

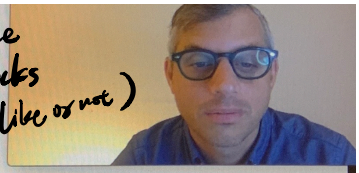
Each row of X shows users affinity for the latent features

Each row of Y shows how movie can be described by the latent features.



LEARNING WITH IMPLICIT FEEDBACK

Because we don't have much explicit feedbacks (rating, like or not)



In many (if not most) cases we don't have explicit user ratings of an item. We only know if a user consumed or viewed the item.

implicit feedback: consumption

To adapt to this, we introduce a new variable:

$$p_{ui} = \begin{cases} 1 & \text{if user consumed item } i \\ 0 & \text{if user did not consume item } i \end{cases}$$

Because consumption isn't always just binary (i.e., how many seconds did they listen to song or watch video), we can create a weighting factor.

$$c_{ui} = 1 + \alpha r_{ui}$$

amount of consumption (metric for affinity)

- r_{ui} is a non-zero observed consumption amount
- α is a weight that we set

hyperparameter

Then we redefine our loss function using p_{ui} and c_{ui}

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

error function: sum of square + regularization

THE RATING PREDICTION

The rating prediction for user i on item j is then an inner product between the the user's preferences for each latent factor and the item's strength on that factor.

y_{jt} indicates how much factor **t** describes item **j**

$r = x_i \cdot y_j$

$$r_{ij} = x_{i1} * y_{j1} + x_{i2} * y_{j2} + \ldots + x_{ik} * y_{jk} = \Sigma_{t=1,k}\, x_{it} * y_{jt}$$

x_{it} indicates how user **i** prefers factor **t**

The learned vectors X_k & Y_k can be used as embedding vectors for users and items (lower dimension and perserve most information)

we can clustering and nearest neighbors for recommendations. we can find most similar users/items. for the purpose of recommendations.