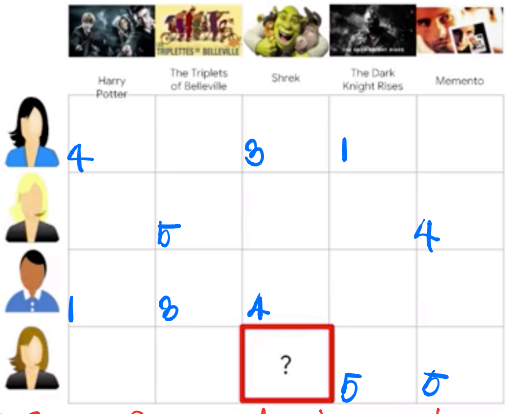
ALS, a matrix factorization algorithm for collaborative filtering.

Types of users feedback data.

- what if we don't know the best factors to compare with?
- collaborative filtering learns latent factors and can explore outside users personal bubble.
  - ↳ use similarities between items & users simultaneously in an embedding space.

Start from a user-interaction matrix where rows are users and items are columns



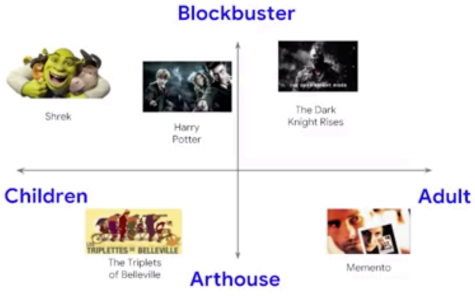
GOAL: Recommend movies to each user that they would like to see.

- "Matrix usually a very sparse one."
  - > because they are large numbers of users and items and most have never interacted.
- To get around this sparsity problem, collaborative filtering usually uses matrix factorization.
- Sometimes these ratings are explicit.
  - → Number of stars, thumbs up, like or dislike clicks "leaving feedback for that item".
  - → There are a lot of blank squares.
- Implicit feedback: watched video time, spend time on a page, clicks, ...

Embedding users & items.

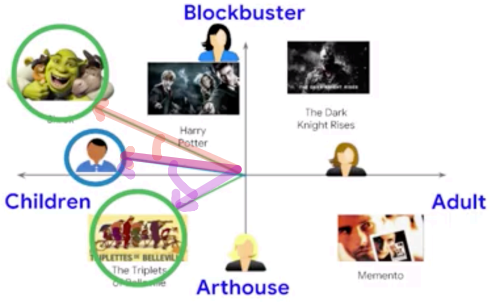
We can organize items by similarity in one dimension

We can organize items by similarity in two dimensions



"To find out the interaction value between users and items, we simply take the dot product between each user item pair.

Simply take the dot product between users and items in embedding space



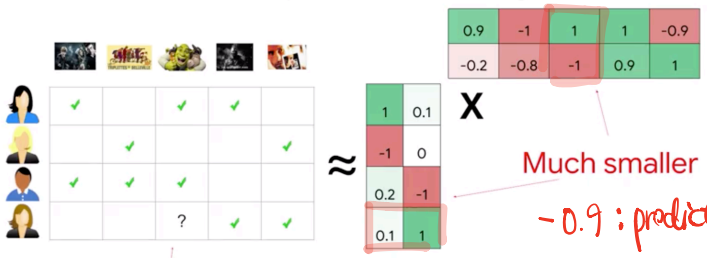
* Each user & item is a d-dimensional point within an embedding space.

- It quickly becomes unscalable as more users & items are added to the system.

∴ Human-derived features may not the best choice.

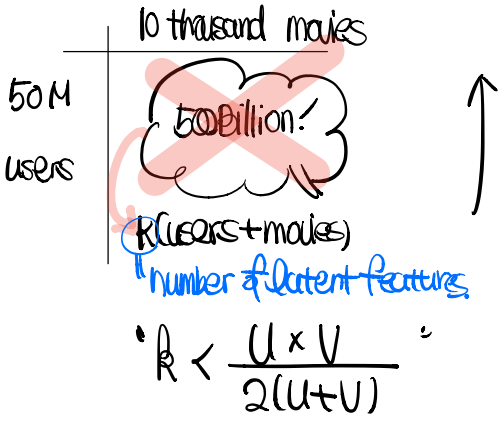
→ embeddings can be learned from data. We're compressing the data to find best generalities to rely on. called "latent factor"

The factorization splits this matrix into row factors and column factors that are essentially user and item embeddings

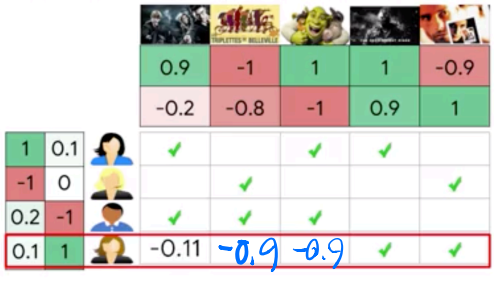


Very large

$A \approx U \times V^T$ #QUIZ



Based on this user-item interaction matrix, which movie should user 4 watch?



A. Harry Potter **-0.11**
B. Triplets of Belleville **-0.9**
C. Shrek **-0.9**
D. The Dark Knight Rises **1.0**

Factorization Approaches

- collaborative filtering is usually carried out using matrix factorization.
  - i) factorize user-interactions matrix into user-factors and item-factors.
  - ii) Given user ID, multiply by item factors to get predicted ratings for all items.
  - iii) Return top k rated items for this user.

User factor Mat.

- $A \approx U \times V^T$ item factor Mat.

user interaction matrix.

$\min_{U,V} \sum_{(i,j) \in obs} (A_{ij} - U_i V_j)^2$: minimize square error!

i) SGD (Stochastic gradient descent)

flexible (can be used with many types of problems & loss f°)

Parallel (can scale out our Matrix-factorization. quickly tackle much larger problem.)

Slower

Hard to handle unobserved interaction pairs.

ii) ALS (Alternating least squares)

least squares only -

parallel.

Faster!

Easy to handle unobserved interaction pairs.

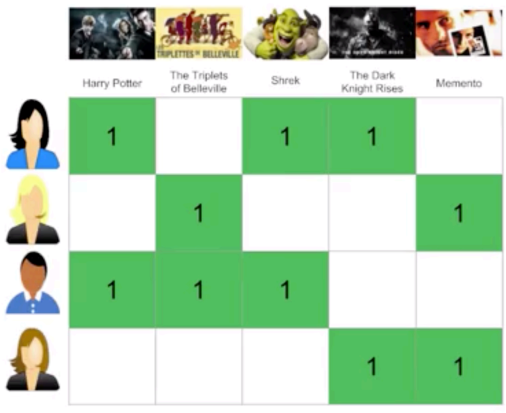
- Unobserved pairs.

$|A - UV^T|^2$

(unobserved pair = 0) ⇒ This can lead to poor performance.

ALS

$\sum_{(i,j) \in obs} (A_{ij} - U_i V_j)^2$ (only summing the observed instances)



"minor tweaks"

Weighted ALS

$$\sum_{(i,j)\in obs}(A_{ij}-U_iV_j)^2+w_0*\sum_{(i,j)\neq obs}(0-U_iV_j)^2$$

Quiz

There are many ways to handle unobserved user-interaction matrix pairs. ___ explicitly sets all missing values to zero. ___ simply ignores missing values. ___ uses weights instead of zeros that can be thought of as representing _____.

A. WALS, SVD, ALS, low confidence
B. SVD, ALS, WALS, low confidence
C. SVD, ALS, WALS, high confidence
D. SVD, ALS, WSVD, low confidence

The ALS algorithm

Initialize U.V.
repeat
  for i=1 to n do
    $u_i = (\sum_{r_{ij}\in r_{i*}} v_i v_j^T + \lambda I_k)^{-1} \sum_{r_{ij}\in r_{i*}} r_{ij} v_j$
  end for
  for j=1 to m do
    $v_j = (\sum_{r_{ij}\in r_{*j}} u_i u_j^T + \lambda I_k)^{-1} \sum_{r_{ij}\in r_{*j}} r_{ij} u_i$
  end for
until convergence

* GOAL: Calculate those two embeddings simultaneously.

We need column factors and the ratings entries for this row.

We need row factors and the ratings entries for this column.

The loop makes it possible to batch the rows or columns.

QUIZ

ALS is an alternating least squares algorithm. In ordinary least squares we have the analytic solution of the normal equation:

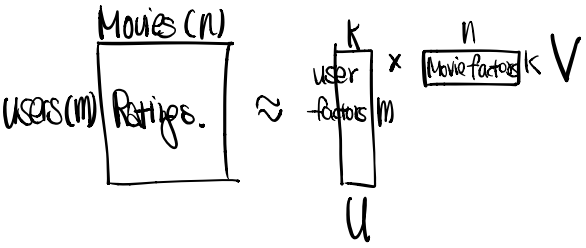
$$\beta = (X^T X + \lambda I_k)^{-1} X^T y$$

In ALS, during the row-factor solving phase, where we fix column-factors:
Row-factors U are analogous to β.
Column-factors V are analogous to X.
Ratings matrix R is analogous to y.

A. X, y, β
B. y, X, β
C. β, X, y
D. β, y, X

Preparing Input Data for ALS.

- Feed the ALS algorithm whole raws (or columns) at a time, but because knowing which stage it's in difficult, feed both.



- iterative algorithm
  - → fix V, compute U.
  - → fix U, compute V.

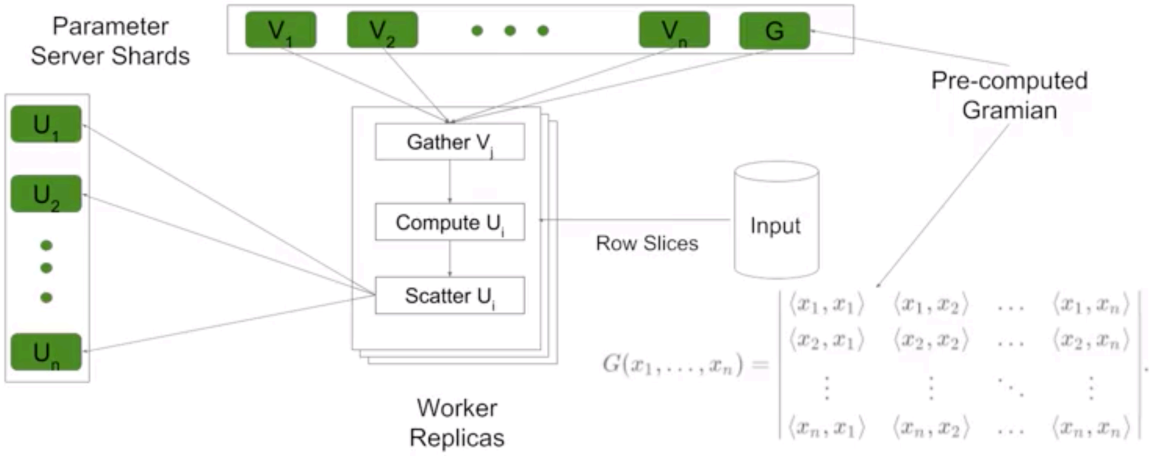
	visitorId	contentId	session_duration
0	7337153711992174438	100074831	44652
1	5190801220865459604	100170790	1214205
2	5874973374932455844	100510126	32109
3	2293633612703952721	100510126	47744
4	1173698801255170595	100676857	10512

map

	userId	itemId	rating
0	0	0	0.231208
1	1	1	1.000000
2	2	2	0.166260
3	3	2	0.247218
4	4	3	0.054431

> save the mapping to persistent storage because you'll need to map input values to the mapped values.

Even though the update depends on the full column factor, it is possible to distribute WALS by precomputing a the Gramian G



#QWIZ

When using the WALS Estimator, it is important to have the inputs in the correct format. What should we do with the table below in the input function before it is used by the Estimator?

	clientID	productID	sentiment
0	s98h2oknsa8	87198471098	Like
1	xnl891nbjk21	95218970198	Love
2	kn0iokjs0i2n	12719850987	Neutral
3	mnal9n2oino	89713598759	Dislike

A. Map clientID to int in [0, num_clients)
B. Map productID to int in [0, num_products)
C. Map sentiment from string to numeric
D. A & B
E. A & C
F. All of the above

The client ID is represented as an alphanumeric string in this data. So, we need to map each string into an integer representing that client's user index. The product ID at least isn't a string, but it is a long integer that is not representative of the actual matrix index. So, we need to map each to an integer representing that products item index. As for the rating, we probably have an example of explicit feedback. However, it is a string, so we will need an ordinal mapping perhaps from the lowest sentiment to highest sentiment by integers, and we can even scale that to be between zero or one or some other range. The main goal is to eventually get the radians into a numeric format. Therefore, we had to create all three mappings and save them in persistent storage to be used for future training and inference.

ALS and WALS create embedding tables for both users and items. Because these are held in memory, it's important to plan for their size. How big would you expect the embedding table for the users to be?

- [x] Proportional to k, the number of dimensions in your embedding space

Correct

- [x] Proportional to the number of users

Correct

- [ ] Proportional to the number of users squared

Un-selected is correct

- [ ] Proportional to the number of users multiplied by the number of items

Un-selected is correct