

[https://i0.wp.com/thedatascientist.com/wp-content/uploads/2018/05/recommender\\_systems.png](https://i0.wp.com/thedatascientist.com/wp-content/uploads/2018/05/recommender_systems.png)

# Lecture 7: Scalable Matrix Factorisation for Collaborative Filtering in RecSys

[COM6012: Scalable ML](#) with Robert Loftin

Slides courtesy of [Haiping Lu](#)

# Week 7 Contents / Objectives

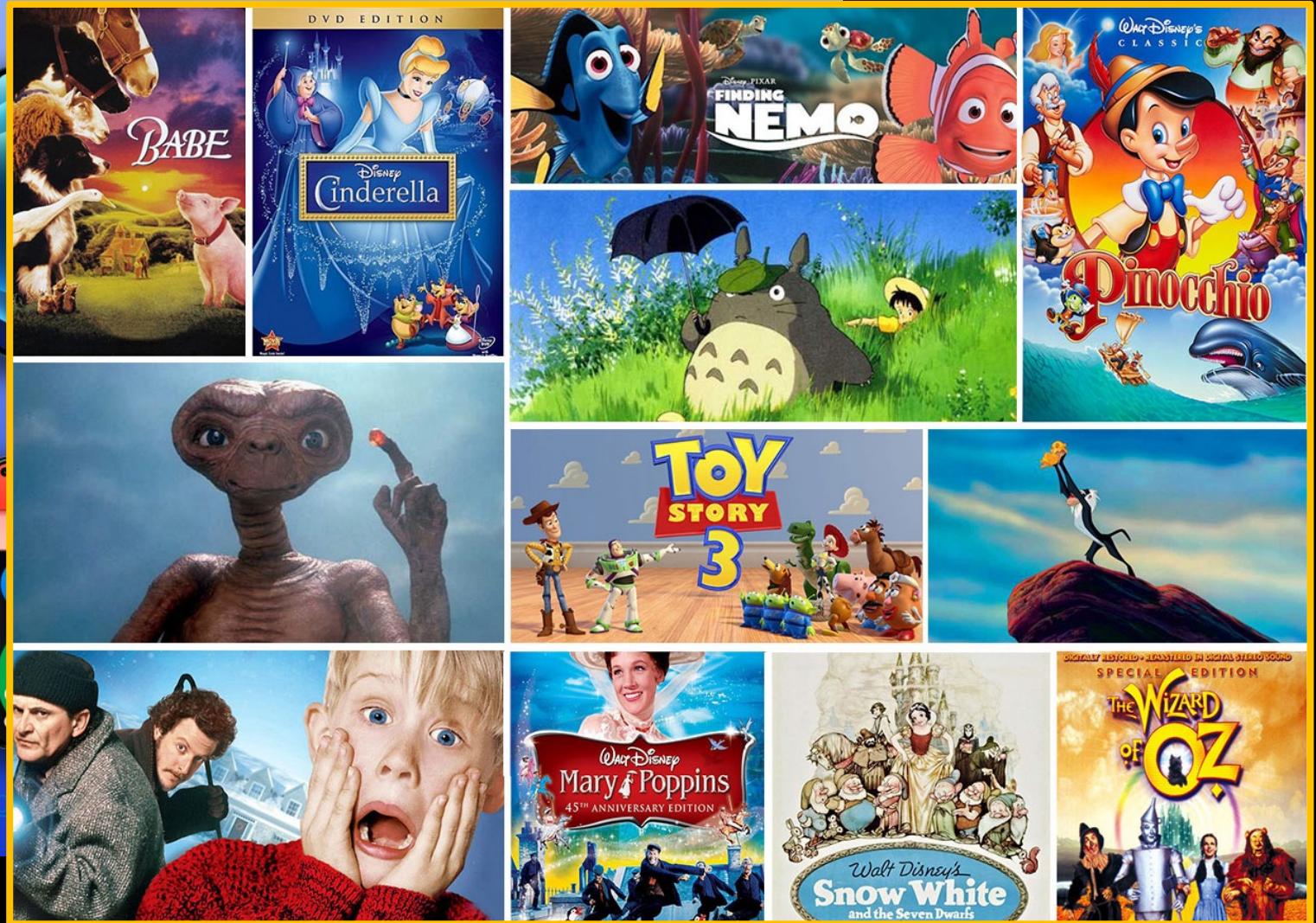
- Recommender Systems
- Collaborative Filtering
- Matrix Factorisation for Collaborative Filtering
- Scalable Collaborative Filtering in Spark

# Many Decisions to Make



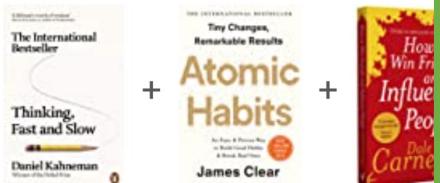
[90 \(1200x800\) \(newatlas.com\)](https://newatlas.com)

# Many Decisions to Make



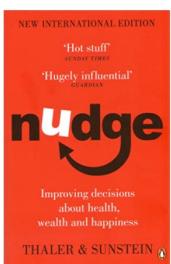
# Recommendations Everywhere

Frequently bought together



- This item: Thinking, Fast and Slow by D
- Atomic Habits: The life-changing millio
- How to Win Friends and Influence Peop

Customers who viewed this

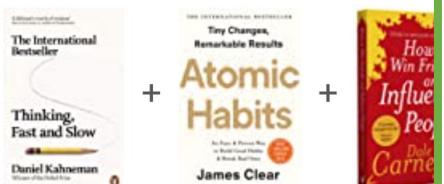


Nudge: Improving Decisions About Health, Wealth and Happiness  
>Richard H. Thaler

A screenshot of a Google search results page. The search bar at the top contains the query "job". Below the search bar is a list of 12 search suggestions, each preceded by a magnifying glass icon. The suggestions are: "jobs at amazon uk", "job in sheffield", "jobs near me", "jobs sheffield", "jobs ecclesall road", "jobs", "jobs ac uk", "jobs at amazon", "Jobcentre Plus", and "jobs indeed uk". At the bottom of the search results are two buttons: "Google Search" and "I'm Feeling Lucky". Below the search results, there is a link to "Report inappropriate predictions" and a note that the site is "Carbon neutral since 2007".

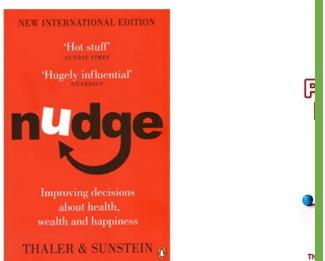
# Recommendations Everywhere

Frequently bought together



- This item: Thinking, Fast and Slow by D...
- Atomic Habits: The life-changing millio...
- How to Win Friends and Influence Peop...

Customers who viewed this



Nudge: Improving Decisions About Health, Wealth and Happiness  
Richard H. Thaler

A screenshot of a Google search results page. The search term "job" is typed into the search bar. Below the search bar, there is a list of suggested search terms:  
jobs at amazon uk  
job in sheffield  
jobs near me  
jobs sheffield  
jobs ecclesall road  
jobs  
jobs ac uk  
jobs at amazon  
Jobcentre Plus  
jobs indeed uk

Google Search

A screenshot of a LinkedIn feed titled "Online events for you". It shows three event cards:

- Scholars Webinar on Drug Discovery...** (Wed, Mar 24 - Thu, M...) Alireza Khatemmoori and 1,616 other... [View](#)
- Accelerating Visual Data Exploration...** Today, 12:30 PM 96 attendees [View](#)
- Live Chat with Salesforce Sr...** Tue, May 11, 5:30 PM Vignesh Srinivasan and 733 other attendees [View](#)

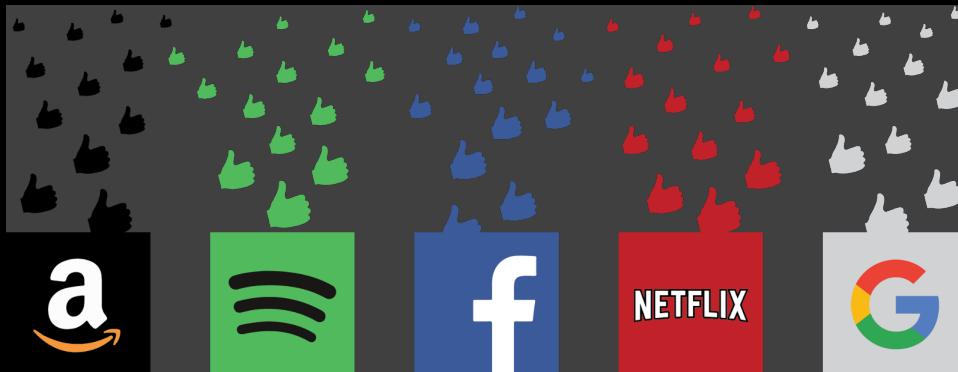
People you may know from The University of Sheffield

A screenshot of a LinkedIn feed titled "People you may know from The University of Sheffield". It shows four profile cards:

- Twin Karma...** Research Software... 9 mutual connections [Connect](#)
- Andrew Stra...** Senior University Teacher... 19 mutual connections [Connect](#)
- Neil Walkins...** Senior Lecturer at The Universi... 30 mutual connections [Connect](#)
- Siobhan No...** Senior Lecturer at University of... 21 mutual connections [Connect](#)

# Recommender Systems (RecSys)

- Predict relevant items for a user, in a given context
- Predict to what extent these items are relevant
- A ranking task (searching as well)
- Implicit, targeted, intelligent advertisement
- Effective, popular marketing

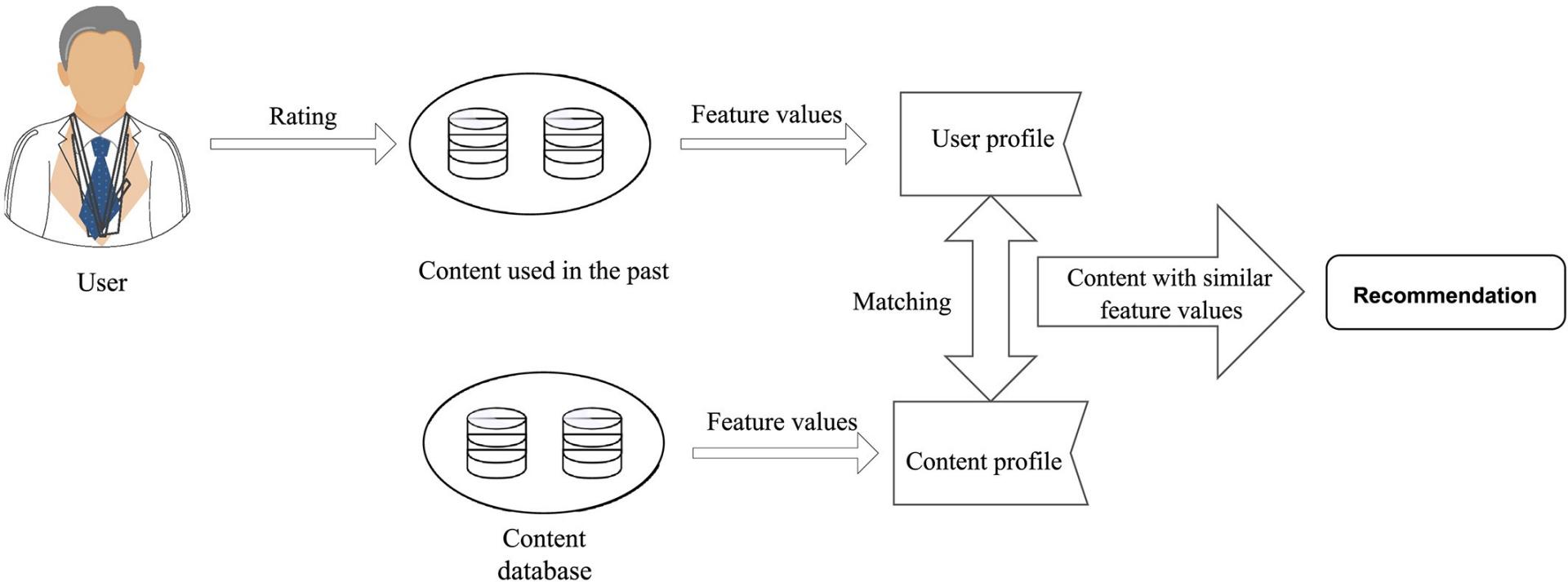


# Two Classes of RecSys

- Content-based – use known features of users/items
- Collaborative filtering – leverage other users

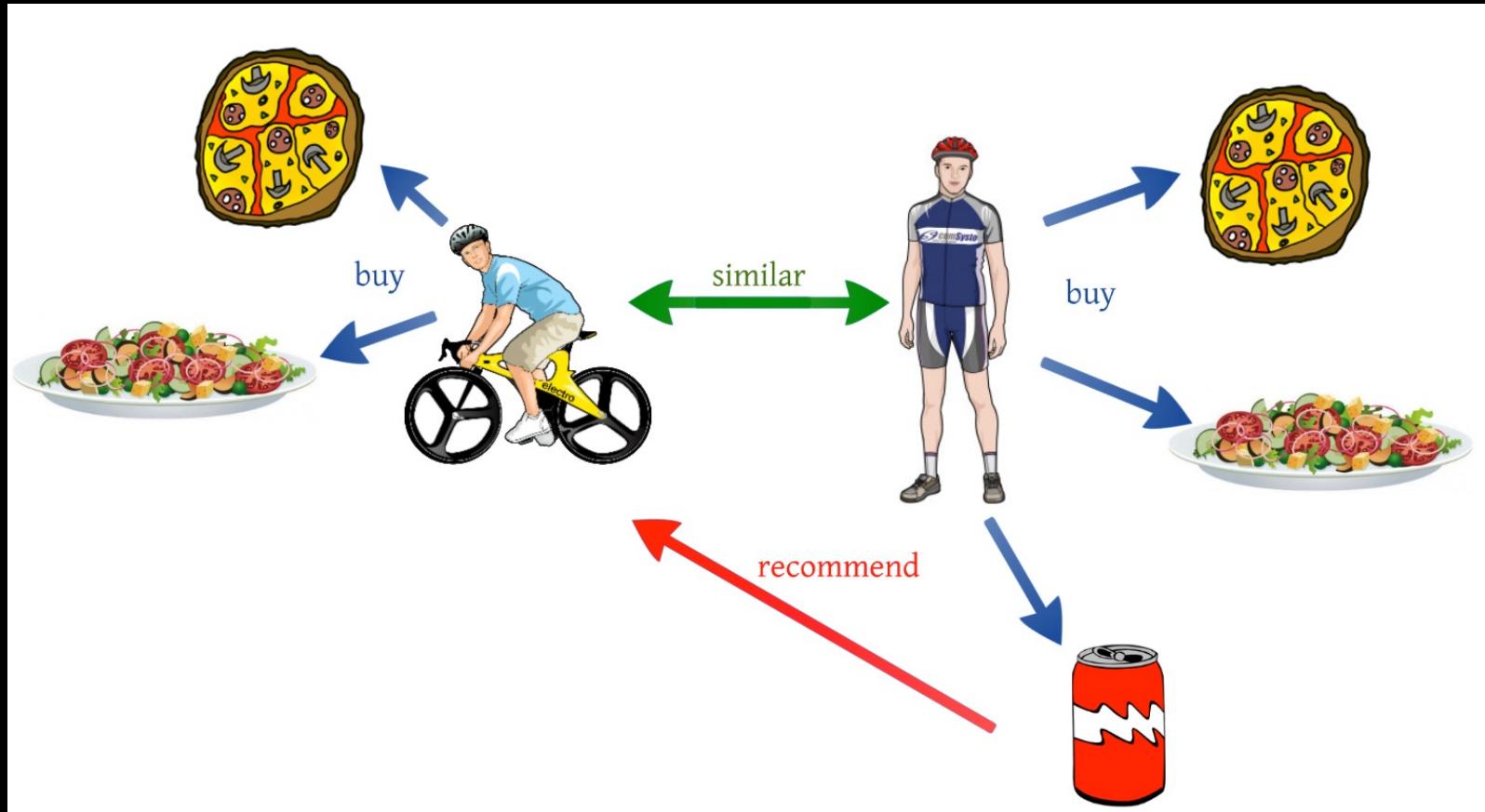


# Content-based RecSys



[dac4058-fig-0010-m.jpg \(2128x789\) \(wiley.com\)](#)

# Collaborative Filtering RecSys



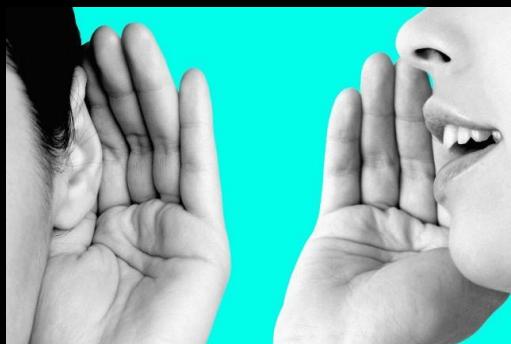
[https://miro.medium.com/max/2656/1\\*6\\_NIX6CJYhtxzRM-t6ywkQ.png](https://miro.medium.com/max/2656/1*6_NIX6CJYhtxzRM-t6ywkQ.png)

# Week 7 Contents / Objectives

- Recommender Systems
- Collaborative Filtering
- Matrix Factorisation for Collaborative Filtering
- Scalable Collaborative Filtering in Spark

# What is Collaborative Filtering?

- Base recommendations on the ratings of other users with similar tastes
- Components
  - Users (customers): who provide ratings
  - Items (products): to be rated
  - Ratings: the training data



	ARGO	SEVEN	RUMBLE	THE KILL
John	5	1	3	5
Tom	?	?	?	2
Alice	4	?	3	?

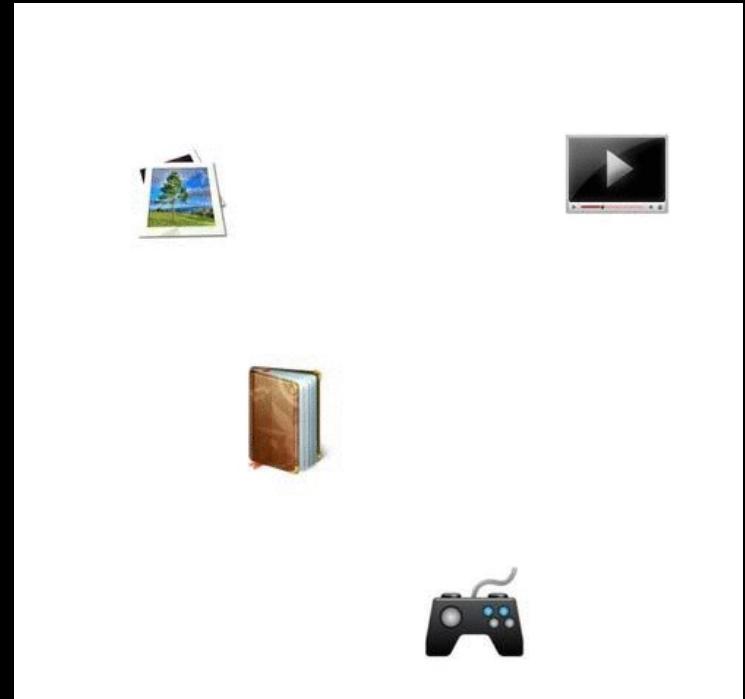
# Collaborative Filtering (CF)

- Many users, many items, many ratings
- Users rate multiple items
- Other users with similar needs/tastes
- Item evaluation requires personal taste
- Taste persists



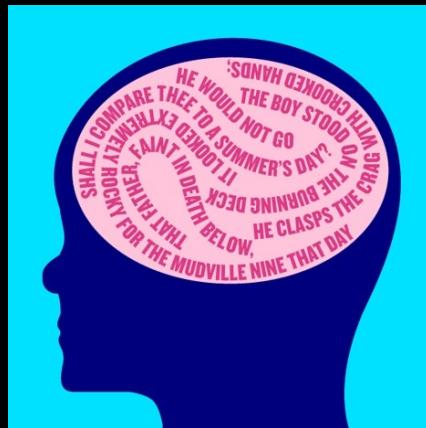
# Collaborative Filtering (CF)

- Objective: predict how a user will rate an item, given a set  $D$  of past ratings by a community of users
- CF for RecSys:
  - Input: many user-item-rating tuples in  $D$
  - Model: similar users  $\leftarrow$  ratings strongly correlate
  - Recommend items rated highly by similar users



# Collaborative Filtering (CF)

- **Memory-based:** predict using past ratings **directly**
  - Weighted ratings given by other similar users
  - User-based & item-based (non-ML)
- **Model-based:** model users based on past ratings
  - Predict ratings using the learned model



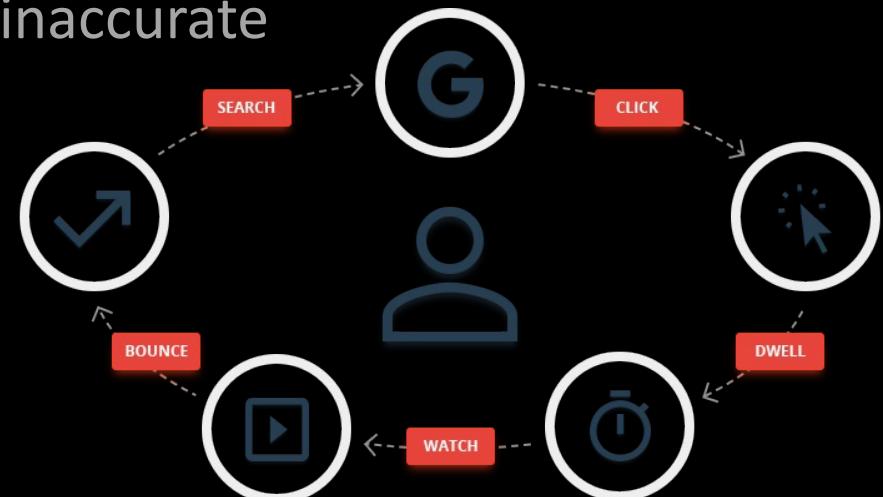
[iforget-465.jpg \(465x465\) \(newyorker.com\)](#)



[neural-header.jpg \(756x503\) \(utsouthwestern.edu\)](#)

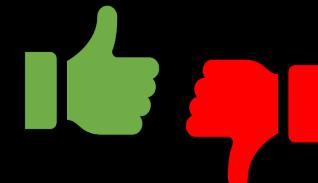
# Explicit vs Implicit Ratings

- **Explicit:** users directly indicate levels of interest
  - Most accurate descriptions of a user's preference
  - Challenging to collect such data
- **Implicit:** infer from observed user behavior
  - Can be collected with little or no cost to user
  - Inference ratings may be inaccurate



# Rating Scales

- Scalar ratings
  - Numerical scales
  - 1-5, 1-7, etc.
- Binary ratings
  - Agree/Disagree, Good/Bad, etc.
- Unary ratings
  - Presence/absence of an event, e.g., purchase/browsing history, search patterns, mouse movements
  - No explicit “negative” examples



# Prediction Accuracy

- Mean absolute error (MAE)

- Where  $n = |D|$

$$MAE = \frac{\sum_{i,j} |p_{i,j} - r_{i,j}|}{n}$$

- Normalized MAE

$$NMAE = \frac{MAE}{r_{max} - r_{min}}$$

- Root mean squared error (RMSE)

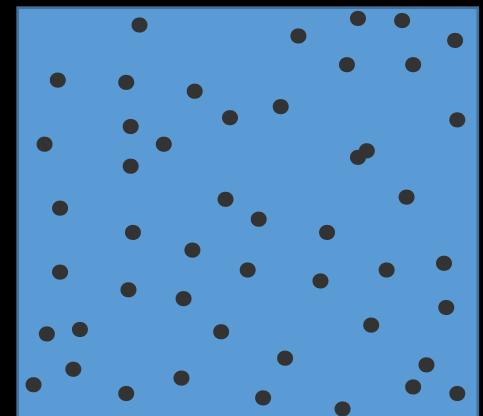
$$RMSE = \sqrt{\frac{1}{n} \sum_{i,j} (p_{i,j} - r_{i,j})^2}$$

# Challenges

- **Cold Start**
  - New user
    - Rate some initial items
    - Non-personalized rec.
    - Describe tastes
    - Demographic info
  - New item
    - Randomly selecting items
    - Content analysis, metadata (non-CF)
- **Sparsity:** sparse user-item matrix
- **Scalability:** millions of users and items



[isbil+2.jpg \(490×303\) \(squarespace-cdn.com\)](#)



# Week 7 Contents / Objectives

- Recommender Systems
- Collaborative Filtering
- Matrix Factorisation for Collaborative Filtering
- Scalable Collaborative Filtering in Spark

# Latent Factor Models

- Map users and items to a **joint latent factor** space of dimensionality  $k$
- Describe each user/item by a vector of  $k$  “factors”
- We can think of each factor as describing some “topic” that a user might be interested in
- High correlation between user and item factors → *good recommendation*

# Latent Factor Models

- To learn – item factors  $\{q_i\}$  and user factors  $\{p_u\}$ :
  - Each user  $u \rightarrow$  vector  $p_u$ : the user's degree of interest in each of the  $k$  factors
  - Each Item  $i \rightarrow$  vector  $q_i$ : the degree to which the item possesses those  $k$  factors
- Define the prediction  $\hat{r}_{u,i}$  of user  $u$ 's rating of item  $i$  as

$$\hat{r}_{u,i} = q_i^T p_u$$

# Matrix Factorisation (MF) for CF

- Let  $R$  be the user x item rating matrix

John	5	1	3	5
Tom	?	?	?	2
Alice	4	?	3	?

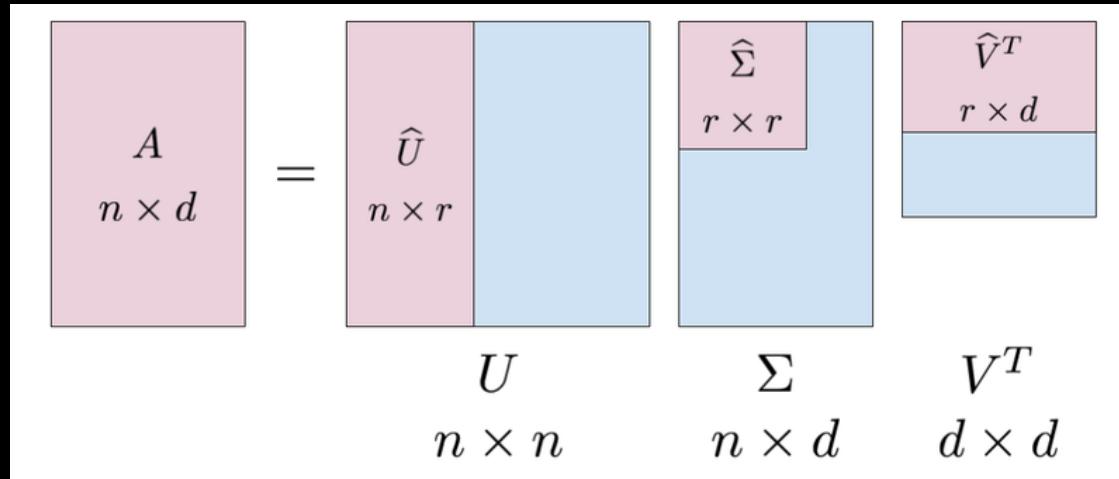
- Approximate rating matrix as  $R \approx PQ^T$  where
  - $P$  is the matrix with rows  $p_u$
  - $Q$  is the matrix with rows  $q_i$

# Singular Value Decomposition

- Factorise the **full** rating matrix  $R$  using SVD
  - Obtain matrices  $P, S, Q$  such that

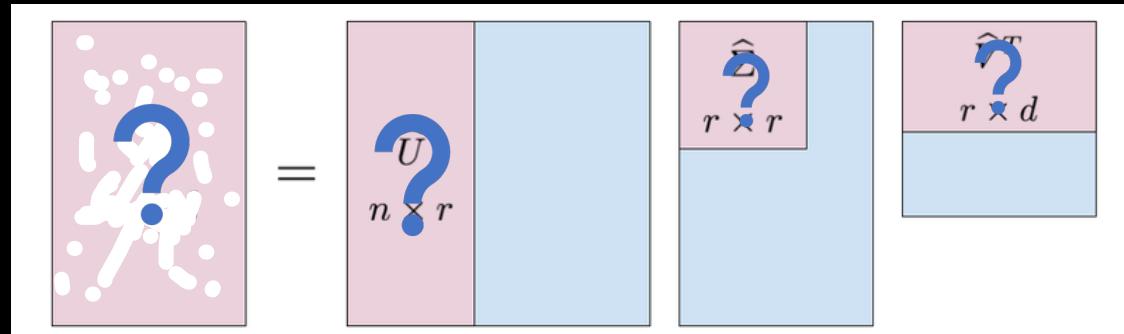
$$R = PSQ^T$$

- Reduce the diagonal matrix  $S$  to the  $k \times k$  matrix  $S_k$ 
  - Likewise Reduce  $P \rightarrow P_k$  and  $Q \rightarrow Q_k$
  - Then  $u$ th row of  $P_k S_k = p_u$  and  $i$ th row of  $Q_k = q_i$



# Singular Value Decomposition

- The SVD is undefined when the rating matrix is not completely known – we only know ratings in  $D$
- Imputation – fill in ratings with user's average
  - Potentially expensive as this creates a lot more rating data
  - Inaccurate imputed values could distort the data



# Factorisation with Missing Values

- Learn to predict the **observed ratings only**
  - Avoid ***overfitting*** through  $\ell_2$ regularisation
  - Minimize the regularised squared error on the set of known ratings to learn the factor vectors  $p_u$  and  $q_i$

$$\min_{Q,P} \sum_{(u,i) \in D} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

- $D$ : the training set of the  $(u,i)$  pairs with known ratings
- $\lambda$ : the regularisation parameter

# Alternating Least Squares for CF

$$\mathcal{L}(Q, P ; D) = \sum_{(u,i) \in D} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

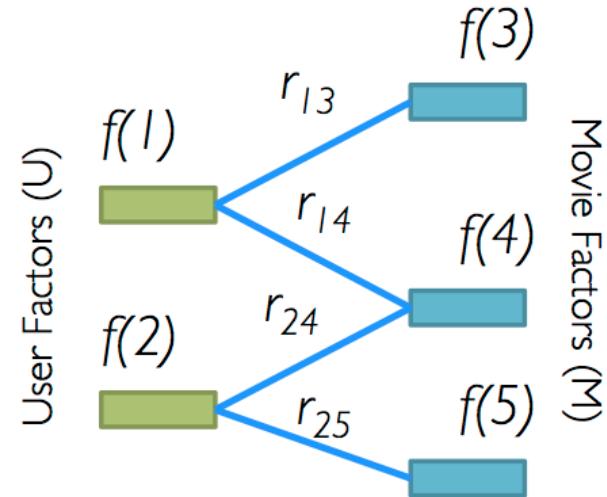
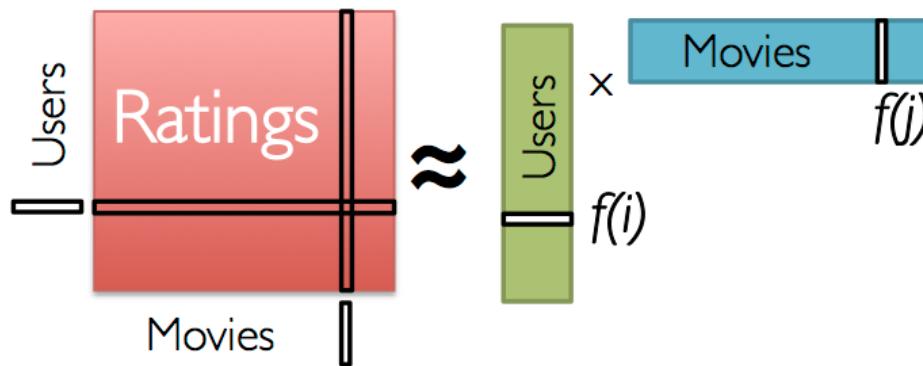
- Both  $p_u$  and  $q_i$  are unknown (loss is non-convex)
- Fix either  $p_u$  or  $q_i \rightarrow$  loss becomes quadratic
- Alternating Least Squares (ALS) – alternate between fixing the  $q_i$ 's and fixing  $p_u$ 's
  - Fix  $P$  and find  $Q$  by solving a ***least-squares*** problem  $\|R - PQ^T\|_F$  - (the Frobenius norm)

$$Q^T = (P^T P)^{-1} P^T R$$

- Fix  $Q$  and update  $P$  as  $P = R \hat{Q} (\hat{Q}^T \hat{Q})^{-1}$

# MF for Movie Recommendations

Low-Rank Matrix Factorization:



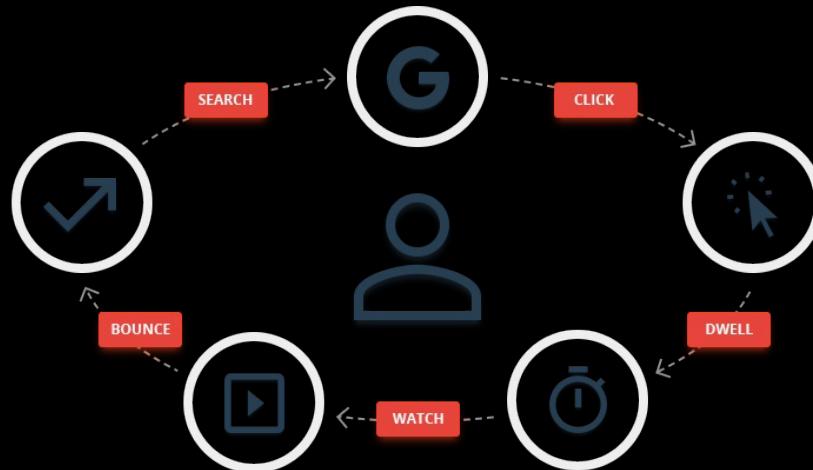
Iterate:

$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda ||w||_2^2$$

<https://github.com/databricks/spark-training>

# Implicit Feedback

- Implicit feedback: views, clicks, purchases, shares
  - Rating  $r$  = “strength” of actions (no. of clicks, viewing duration) → confidence level of observed preference
  - Preference matrix  $P$  with  $p = 1$  if  $r > 0$  and  $0$  if  $r = 0$
  - Factorise of  $P$  with ratings **weighted** by their confidence, which is a function of  $r$  (for details see this [ICDM08 paper](#))



# Week 7 Contents / Objectives

- Recommender Systems
- Collaborative Filtering
- Matrix Factorisation for Collaborative Filtering
- Scalable Collaborative Filtering in Spark

# Key in Scalable ML

- Computation and storage should be **linear** (in  $D$ ,  $k$ )  
→ Low-cost computation (time + space)
- Perform **parallel** and **in-memory** computation  
→ Many working + reduce disk I/O
- Minimise network **communication** → Reduce overhead in parallelisation, not the more the better

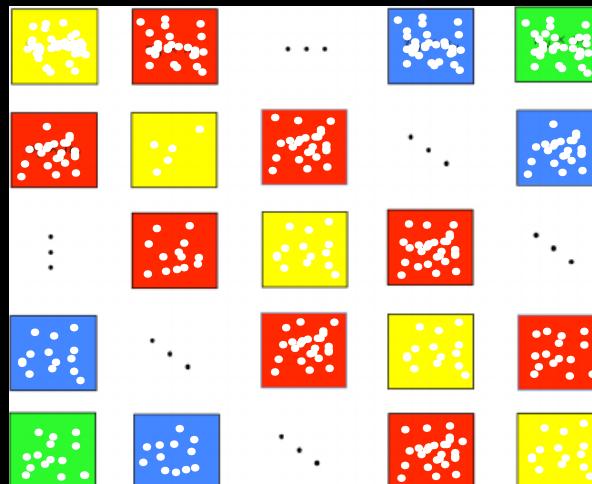
# Blocked Implementation of ALS

$$\mathcal{L}(Q, P ; D) = \sum_{(u,i) \in D} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

*How can we parallelize this?*

# Blocked Implementation of ALS

- Group users and items into blocks
  - Reduce **communication**: only send one copy of each user vector to each item block on each iteration, and only for the item blocks that need that user's feature vector
  - Pre-compute info: **out-links** of each user (which blocks of items it will contribute to); **in-links** for each item (which of the feature vectors it receives from each user block it will depend on)



[Color-online-A-symmetric-block-Toeplitz-matrix-Each-block-is-also-a-symmetric-Toeplitz.png \(488x369\) \(researchgate.net\)](#)

# The ALS API in Spark

- `numUserBlocks/numItemBlocks`: the number of blocks the users/items will be partitioned into to parallelize computation (defaults to 10)
- `rank`: the number of latent factors in the model (defaults to 10)
- `regParam`: the regularization parameter in ALS (defaults to 0.1)
- `implicitPrefs`: whether to use the explicit feedback ALS variant or one adapted for implicit feedback data (defaults to false: explicit ratings)
- `alpha`: the baseline confidence of implicit feedback (defaults to 1.0)
- `nonnegative`: whether to use nonnegative constraints (defaults to false)
- `coldStartStrategy`: “drop” → drop any rows in the DataFrame of predictions that contain NaN values (defaults to “nan”: assign NaN to a user and/or item factor is not present in the model)
- `blockSize`: the size of the user/product blocks in the blocked implementation of ALS to reduce communication

```

944 def train[ID: ClassTag]( // scalastyle:ignore
945     ratings: RDD[Rating[ID]],
946     rank: Int = 10,
947     numUserBlocks: Int = 10,
948     numItemBlocks: Int = 10,
949     maxIter: Int = 10,
950     regParam: Double = 0.1,
951     implicitPrefs: Boolean = false,
952     alpha: Double = 1.0,
953     nonnegative: Boolean = false,
954     intermediateRDDStorageLevel: StorageLevel = StorageLevel.MEMORY_AND_DISK,
955     finalRDDStorageLevel: StorageLevel = StorageLevel.MEMORY_AND_DISK,
956     checkpointInterval: Int = 10,
957     seed: Long = 0L)(
958     implicit ord: Ordering[ID]): (RDD[(ID, Array[Float])], RDD[(ID, Array[Float])]) = {
959
960     require(!ratings.isEmpty(), s"No ratings available from $ratings")
961     require(intermediateRDDStorageLevel != StorageLevel.NONE,
962             "ALS is not designed to run without persisting intermediate RDDs.")
963
964     val sc = ratings.sparkContext
965
966     // Precompute the rating dependencies of each partition
967     val userPart = new ALSPartitioner(numUserBlocks)
968     val itemPart = new ALSPartitioner(numItemBlocks)
969     val blockRatings = partitionRatings(ratings, userPart, itemPart)
970         .persist(intermediateRDDStorageLevel)
971     val (userInBlocks, userOutBlocks) =
972         makeBlocks("user", blockRatings, userPart, itemPart, intermediateRDDStorageLevel)
973     userOutBlocks.count()    // materialize blockRatings and user blocks

```

# CF in Spark ML

- [Scala code](#) (1800+ lines)
- Documentation: [Collaborative Filtering in Spark](#)
- [DataBricks movie recommendations tutorial](#)
- [DataBricks](#): founded by the creators of Apache Spark
  - Their latest packages at [their GitHub page](#)
  - [Databricks community edition](#): 14 days free on AWS, Microsoft Azure or Google Cloud.



# References

- Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 8 (2009): 30-37 (Yahoo & AT&T)
- Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." *Eighth IEEE International Conference on Data Mining*, 2008
- Charu C. Aggarwal, Recommender Systems: The Textbook, April 2016