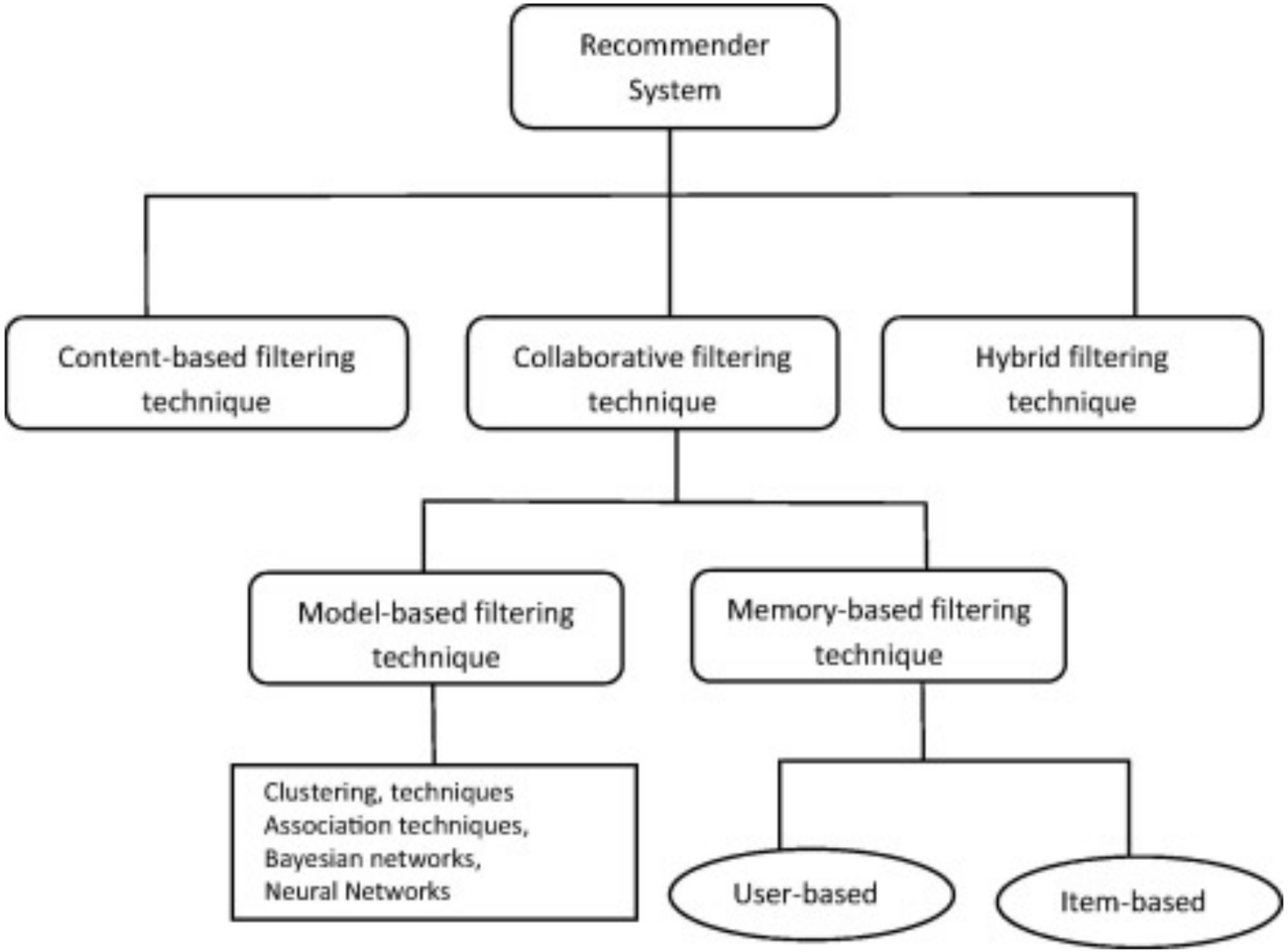
SINGAPORE INSTITUTE OF TECHNOLOGY

Recommender Systems before Deep Learning era

Lucas Vinh Tran

23 March 2023

Overview

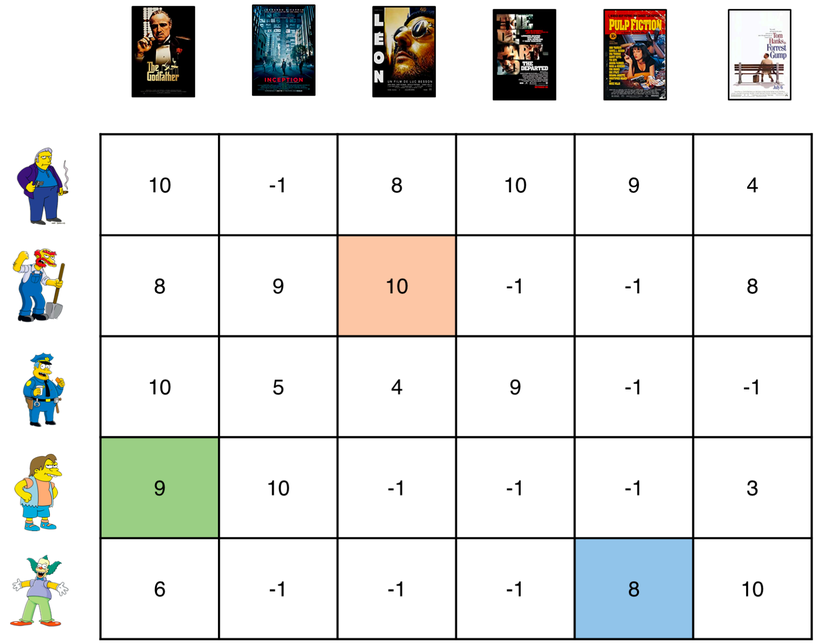

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Traditional Model-based CF

Matrix Factorization

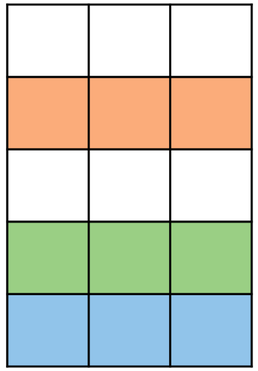
- Users and items are associated with a factor vector
- Dot-product is used to capture user's estimated interest in the item
- Question: How to compute a mapping of users and items to factor vectors?
- Methods:
  - Singular Value Decomposition (SVD)
  - Matrix Factorization (MF)

Matrix Factorization



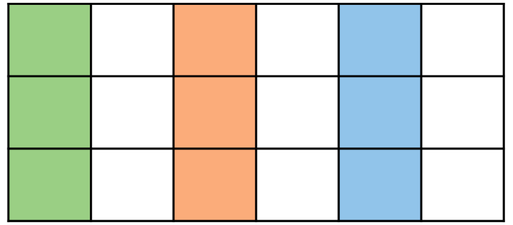
User-item Interaction Matrix (R)

≈



User Matrix (Q)

×



Item Matrix (P)

Source: https://nvidia-merlin.github.io/models/main/models_overview.html

Singular Value Decomposition (SVD)

Dimensionality Reduction

Netflix Prize

2 languages

Article Talk | Read Edit View history

From Wikipedia, the free encyclopedia

The **Netflix Prize** was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films, i.e. without the users being identified except by numbers assigned for the contest.

The competition was held by Netflix, an online DVD-rental and video streaming service, and was open to anyone who is neither connected with Netflix (current and former employees, agents, close relatives of Netflix employees, etc.) nor a resident of certain blocked countries (such as Cuba or North Korea).[1] On September 21, 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%.[2]

Recommender systems

Concepts
Collective intelligence · Relevance · Star ratings · Long tail

Methods and challenges
Cold start · Collaborative filtering · Dimensionality reduction · Implicit data collection · Item-item collaborative filtering · Matrix factorization · Preference elicitation · Similarity search

Implementations
Collaborative search engine · Content discovery platform · Decision support system · Music Genome Project · Product finder

Research
GroupLens Research · MovieLens · **Netflix Prize**

V · T · E

Problem and data sets [edit]

Netflix provided a *training* data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies. Each training rating is a quadruplet of the form `<user, movie, date of grade, grade>`. The user and movie fields are integer IDs, while grades are from 1 to 5 (integer) stars.[3]

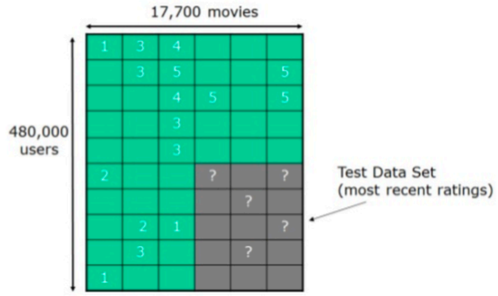
The *qualifying* data set contains over 2,817,131 triplets of the form `<user, movie, date of grade>`, with grades known only to the jury. A participating team's algorithm must predict grades on the entire qualifying set, but they are informed of the score for only half of the data: a *quiz* set of 1,408,342 ratings. The other half is the *test* set of 1,408,789, and performance on this is used by the jury to determine potential prize winners. Only the judges know which ratings are in the quiz set, and which are in the test set—this arrangement is intended to make it difficult to hill climb on the test set. Submitted predictions are scored against the true grades in the form of root mean squared error (RMSE), and the goal is to reduce this error as much as possible. Note that, while the actual grades are integers in the range 1 to 5, submitted predictions need not be. Netflix also identified a *probe* subset of 1,408,395 ratings within the *training* data set. The *probe*, *quiz*, and *test* data sets were chosen to have similar statistical properties.

Source: https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf

Singular Value Decomposition (SVD)

Dimensionality Reduction

- Netflix Prize ($1 Million Contest), 2006-2009



Training data

user	movie	date	score
1	21	5/7/02	1
1	213	8/2/04	5
2	345	3/6/01	4
2	123	5/1/05	4
2	768	7/15/02	3
3	76	1/22/01	5
4	45	8/3/00	4
5	568	9/10/05	1
5	342	3/5/03	2
5	234	12/28/00	2
6	76	8/11/02	5
6	56	6/15/03	4

Test data

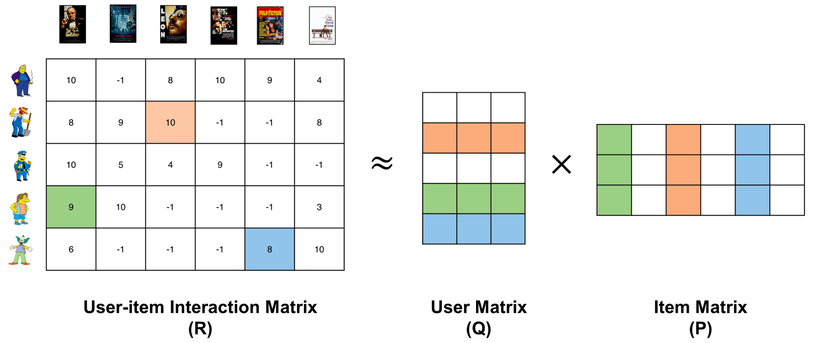
user	movie	date	score
1	62	1/6/05	?
1	96	9/13/04	?
2	7	8/18/05	?
2	3	11/22/05	?
3	47	6/13/02	?
3	15	8/12/01	?
4	41	9/1/00	?
4	28	8/27/05	?
5	93	4/4/05	?
5	74	7/16/03	?
6	69	2/14/04	?
6	83	10/3/03	?

Singular Value Decomposition (SVD)

Dimensionality Reduction

- $R[m \times n] = U[m \times r]S[r \times r](V[n \times r])^T$, where R is the rating matrix with n users and m movies; U is a user-factor matrix; S is a diagonal matrix showing strength of each vector; and V is a item-factor (movie) matrix
- Given the user factor $u_i \in \mathbb{R}^d$ and item factor $v_j \in \mathbb{R}^d$, the predicted rating is $\hat{r}_{ij} = u_i v_j^T$

$$\underset{m \times n}{X} = \underset{m \times r}{U} \; \underset{r \times r}{S} \; \underset{r \times n}{V^\top}$$



Singular Value Decomposition (SVD)

Example

- SVD: $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

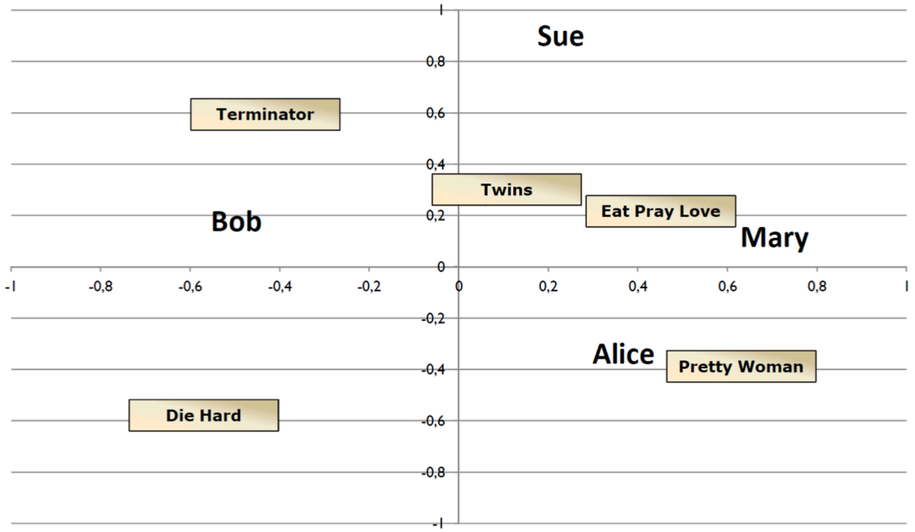
V_k^T	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

  = 3 + 0.84 = 3.84

The projection of U and V^T in the 2 dimensional space (U_2, V_2^T)



Source: https://www.computer.org/csdl/magazine/co/2009/08/mco2009080030/13rRUxBa5fj

Singular Value Decomposition (SVD)

Discussion

- Conventional SVD is undefined for incomplete matrices
- Imputation to fill in missing values
  - Possible issues?
- We need an approach that can simply ignore missing ratings/values

Matrix Factorization (MF)

- How about we model directly observed ratings?

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2$$

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(|| q_i ||^2 + || p_u ||^2)$$

Matrix Factorization (MF)

Learning Strategies

- Stochastic Gradient Descent (SGD)
  - Modification of parameters (q_i, p_u) relative to prediction error
  - Recommended algorithm (better for sparse data)
- Alternating least squares (ALS)
  - Allow massive parallelization
    - Iteratively and independently update q_i and p_u
  - Better for densely filled matrices

Matrix Factorization (MF)

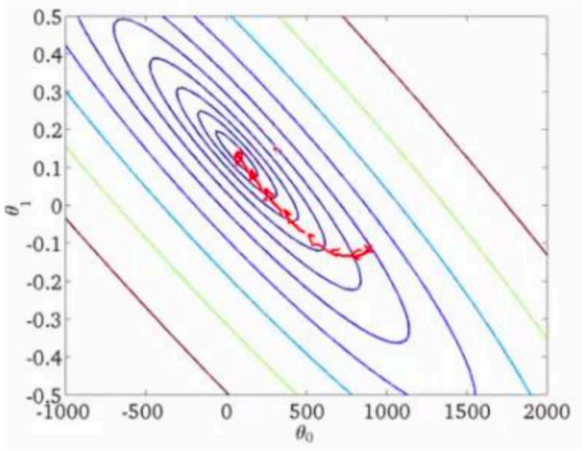
SGD

- $min_{q*,p*} \Sigma_{(u,i) \in k} (r_{ui} - q_i^T p_u)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2)$, for each given training sample
  - Compute construction error
    $e_{ui} = (r_{ui} - q_i^T p_u)^2 + \ldots \Rightarrow e_{ui} = r_{ui} - q_i^T p_u + \ldots$
  - Update in opposite direction to gradient $q_i \leftarrow q_i - \gamma \frac{\partial e_{ui}}{\partial q_i}$
  - γ is the learning rate that controls how much to change the model in response to the estimated error each time the model weights are updated

Matrix Factorization (MF)

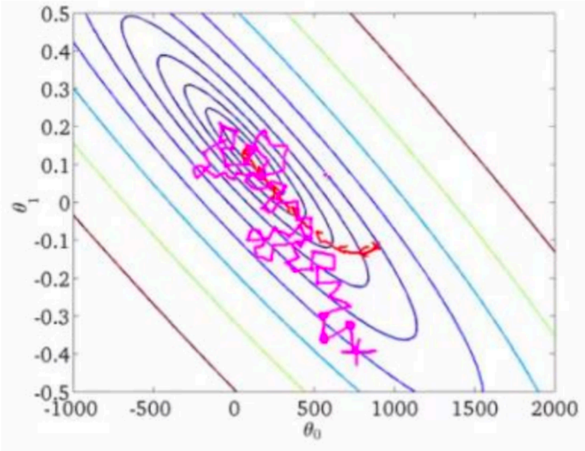
SGD

- In Batch Gradient Descent, all the training data is taken into consideration to take a single step
- We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters



Batch: gradient

$$x \leftarrow x - \eta \nabla F(x)$$



Stochastic: single-example gradient

$$x \leftarrow x - \eta \nabla F_i(x)$$

Source: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Matrix Factorization (MF)

ALS

- $min_{q*,p*}\Sigma_{(u,i)\in k}(r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$; ALS is a multi-regression problem
  - With vectors p fixed, find vectors q that minimize the above function
  - With vectors q fixed, find vectors p that minimize the above function
  - Iterate until convergence

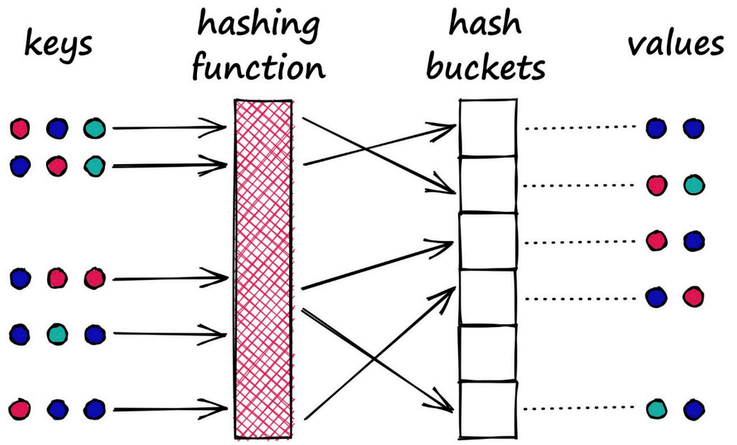
Matrix Factorization (MF)

Enrichment

- Some items always get better (or worse) ratings than others
- Some people always give better (or worse) ratings than others
- Some systems make people give better (or worse) ratings than others
- E.g., time-sensitive user preferences, weather-driven on-device recommendation, etc.

Locality Sensitive Hashing (LSH)

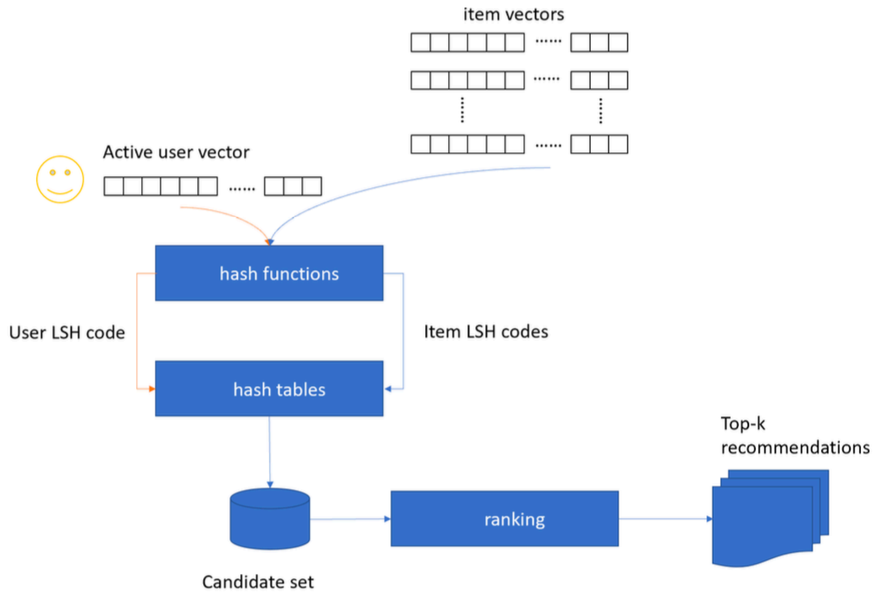
- Pairwise comparisons are commonly essential to construct neighbourhood and item retrieval
  - Computationally expensive and low efficient especially in high dimensional situation (e.g., content features)
  - Increasing time and space complexity for more items
- Solution: hashing!



Locality Sensitive Hashing (LSH)

LSH is efficient for retrieving items that are similar to the active/target item (query)

- Mapping similar objects into same hash bin and dissimilar objects to different bins
- LSH = Hash Function + Hash Tables



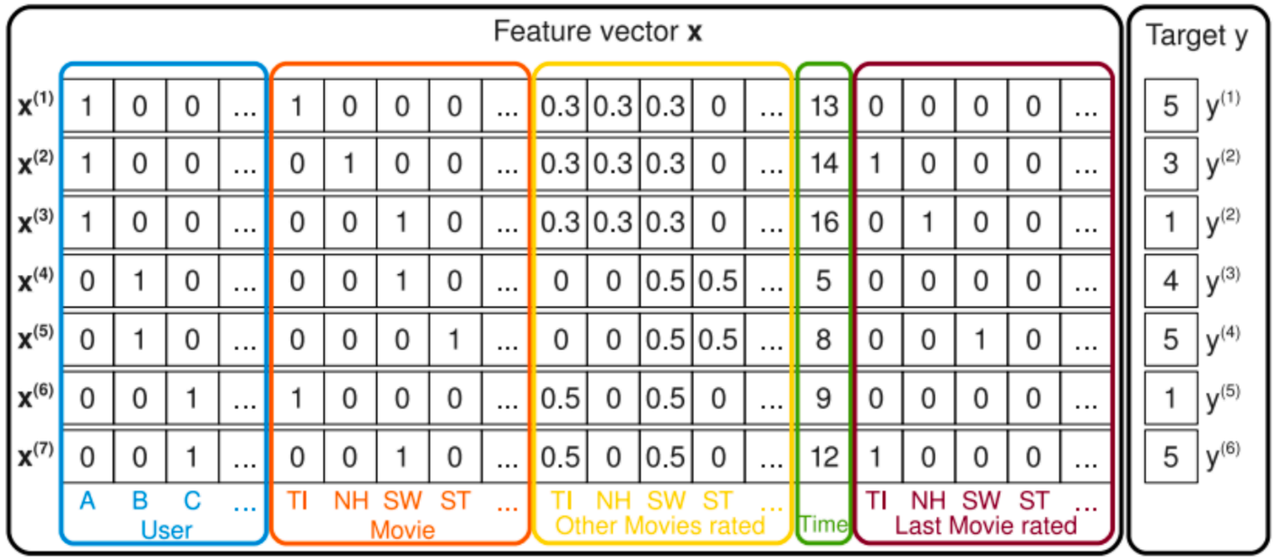
Factorization Machines (FM)

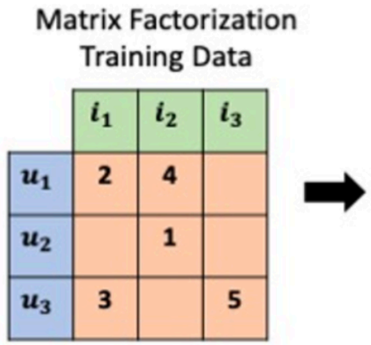
Assume that we have the transaction data of movie review system. Our system records which user $u \in U$ rates a movie (item) $i \in I$ at a certain time $t \in T$ with a rating $r \in \{1,2,3,4,5\}$.

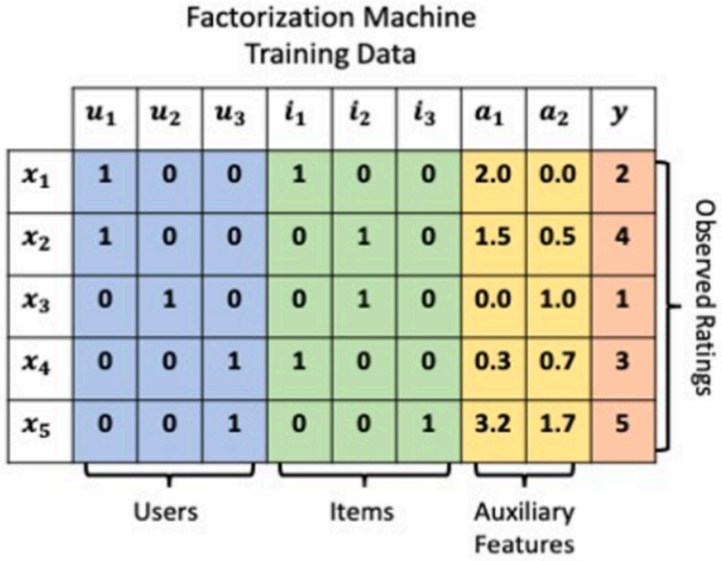
$$S = \{(\text{A}, \text{TI}, \text{2010-1}, 5), (\text{A}, \text{NH}, \text{2010-2}, 3), (\text{A}, \text{SW}, \text{2010-4}, 1)$$
$$(\text{B}, \text{SW}, \text{2009-5}, 4), (\text{B}, \text{ST}, \text{2009-8}, 5),$$
$$(\text{C}, \text{TI}, \text{2009-9}, 1), (\text{C}, \text{SW}, \text{2009-12}, 5)\}$$

	Titanic (TI)	Notting Hill (NH)	Star Wars (SW)	Star Trek (ST)
Alice (A)	5	3	1	
Bob (B)			4	5
Charlie (C)	1		5	

Factorization Machines (FM)







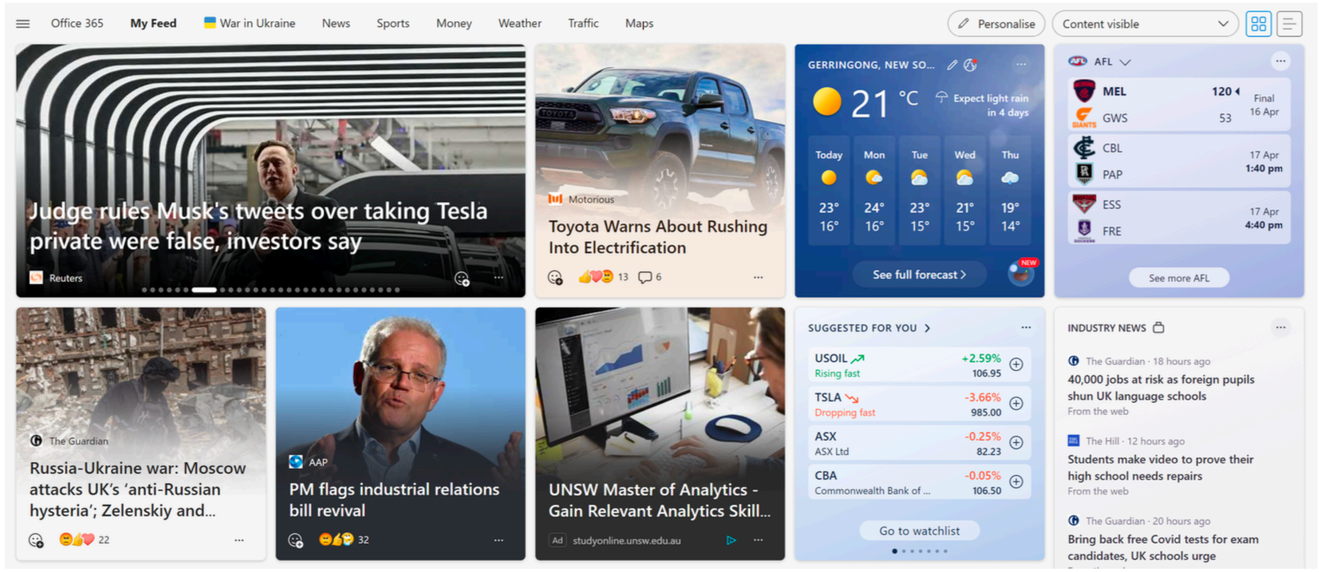
Source: https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf

Factorization Machines (FM)

- Combine linear regression and MF
- A general predictor, can be used for different tasks (i.e., classification, regression and ranking)
- Generalised Factorization Models, mostly used in recommender systems
- Main goal is to model interactions between features (i.e., attributes, explanatory variables) using factorized parameters

Other Methods

- Other types of MFs: Hybrid Neighbourhood-based MF, Probabilistic Matrix Factorization (PMF), etc.
- Association Rules



- Bayesian CF, Clustering CF

Learning To Rank

Why Learning To Rank?

- No need to predict category labels, e.g., mapping to an unordered set of classes in ordinal classification
- No need to predict value of $f(x)$, e.g., rating score
- Relative ranking order is more important, e.g., top-k recommendation

Problem Formulation

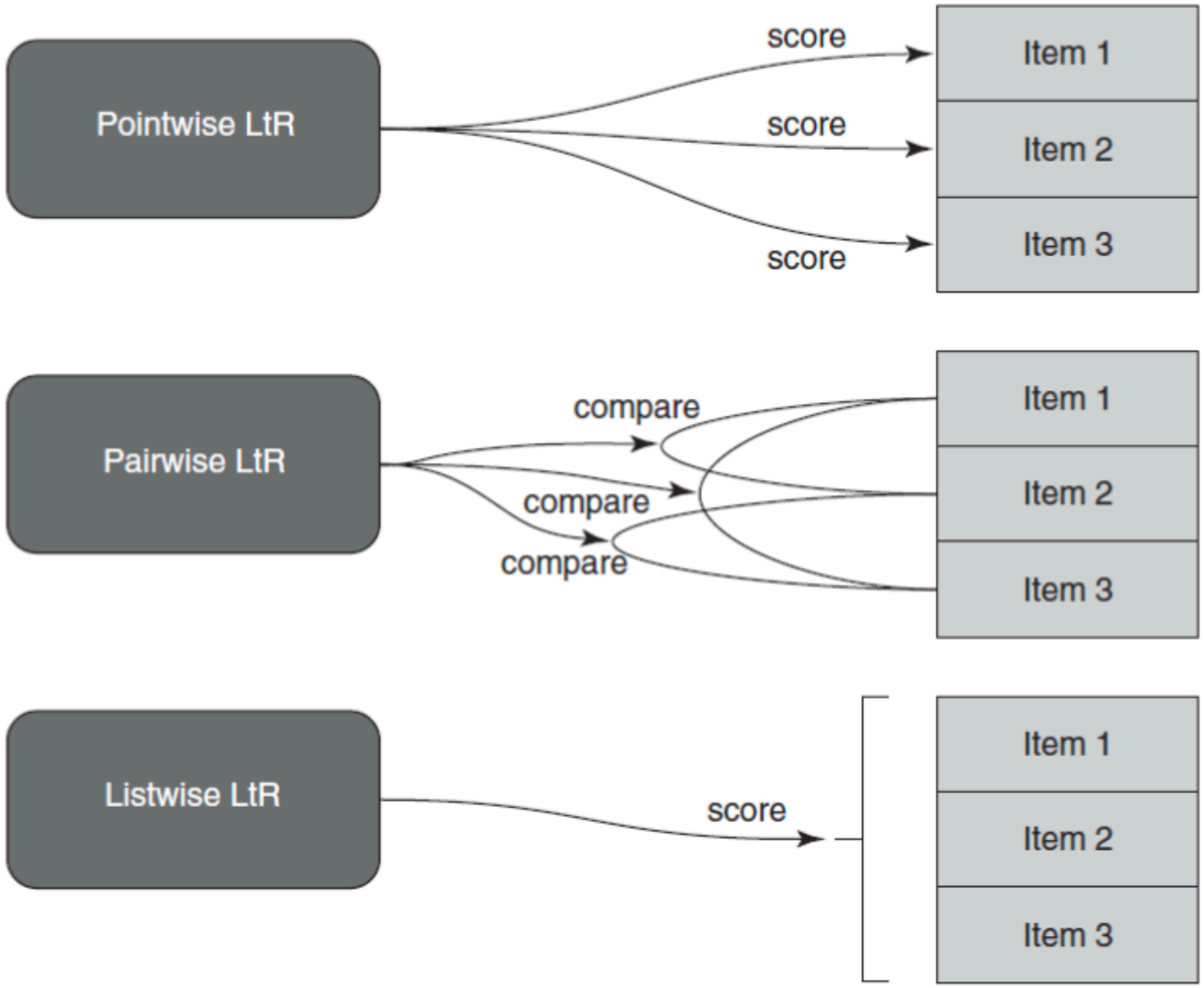
To create a ranking list of objects using the features of the objects

- Given a set of input vectors and corresponding labels with specified order
- Learn a function that specifies a ranking over the input vectors, which minimizes the cost/loss

Ranking as a recommendation

- Most recommendations are presented in a sorted list
- Recommendation can be understood as a ranking problem
- Implicit feedback (will discuss more in the last lecture)
- Given a set of ordered pairs of instances as training data
  - Learn an item scoring function
  - Learn a classifier for classifying item pairs into two types of relations (correctly ordered vs. incorrectly ordered)

Ranking as a recommendation



Ranking as a recommendation

Pointwise methods $f(u, i) \rightarrow \mathbb{R}$

- Minimizing the loss function between predicted value and ground-truth value
- Ranking score is based on regression or classification

Ranking as a recommendation

Pairwise methods $f(u, i^+, i^-) \rightarrow \mathbb{R}$

- Loss function is defined on pairwise preferences
- The observed entries should be ranked higher than the unobserved ones
- Maximizes the margin between observed entry and unobserved entry

Ranking as a recommendation

Listwise methods $f(u, i_1, i_2, \ldots, i_n) \rightarrow \mathbb{R}$

- Indirect loss functions, e.g., similarity between ranking list and ground-truth as loss function; KL-divergence as loss function by defining a probability distribution
- Directly optimizing the ranking measures, like NDCG (Normalized Discounted Cumulative Gain), MRR (Mean Reciprocal Rank)

Bayesian Personalized Ranking

- Recommending the top k mostly relevant items only
- A ranked list - the top k is orderly arranged according to relevance
- Reasons of missing values are complex
  - Explicit feedback is not available in many cases, e.g., users may not rate an item that they don’t like
  - Explicit feedback may not capture the nature of data, e.g., a clickstream dataset may only reveal how frequent a user visit an item, but that may not be equivalent to say the user like this item

Source: BPR: Bayesian Personalized Ranking from Implicit Feedback. UAI 2009.

Bayesian Personalized Ranking

- Training data consists of both positive and negative pairs and missing values. The missing values between two non-observed items are exactly the item pairs that have to be ranked in the future. That means from a pairwise point of view the training data and the test data is disjoint.
- The training data is created for the actual objective of ranking.

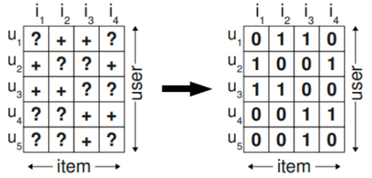


Figure 1: On the left side, the observed data S is shown. Learning directly from S is not feasible as only positive feedback is observed. Usually negative data is generated by filling the matrix with 0 values.

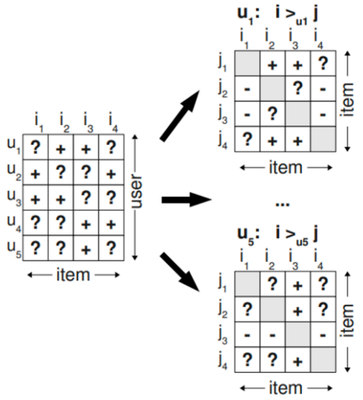


Figure 2: On the left side, the observed data S is shown. Our approach creates user specific pairwise preferences $i >_u j$ between a pair of items. On the right side, plus (+) indicates that a user prefers item i over item j; minus (−) indicates that he prefers j over i.

Source: BPR: Bayesian Personalized Ranking from Implicit Feedback. UAI 2009.

Learning to Rank for recommendation

Recap

- Memory-based learning to rank (e.g., EigenRank)
- Model-based learning to rank
  - Model-based pairwise learning to rank (e.g., BPR)
  - Model-based listwise learning to rank
    - Pointwise ranking can’t be directly interpreted into a measure of ranking quality
    - Pairwise ranking is computationally intensive, and hard to run at scale

Break

Sequential and Session-based/aware

Background

Input: an ordered and timestamped list of past user actions (e.g., user downloaded Angry Bird, Doddle Jump, Temple Run, … from the App Store)

- Anonymous user actions (e.g., GDPR)
- Order of recommended items matter (e.g., next item to buy)
- Decide how many similar users' opinions should be considered

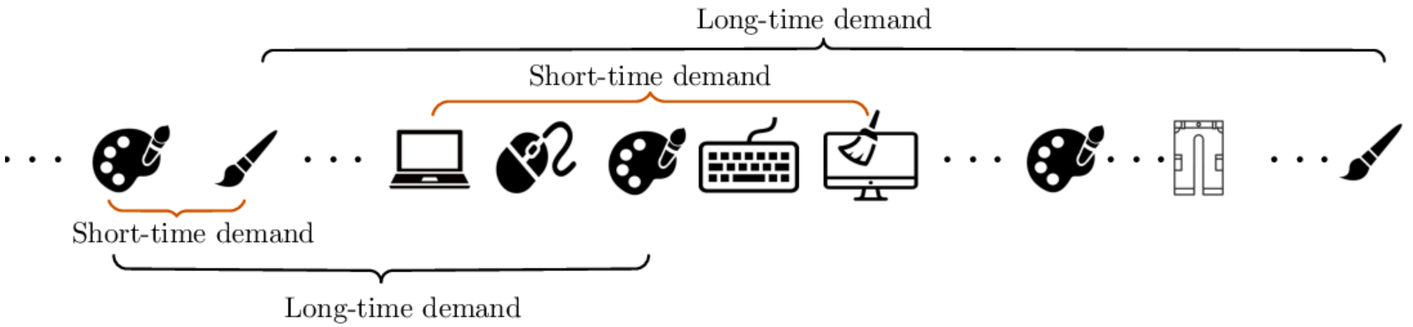
Output: next item, next set of items, next location, etc.

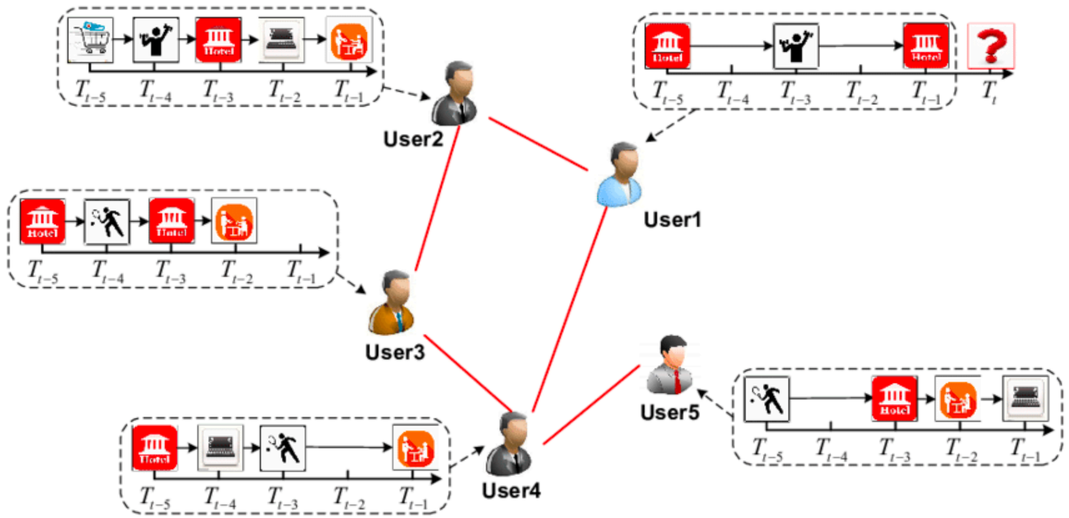
Background

Examples

Keywords:

- Next-item recommendation
- Sequential recommendation
- Session-based recommendation
- Next-POI recommendation
- …



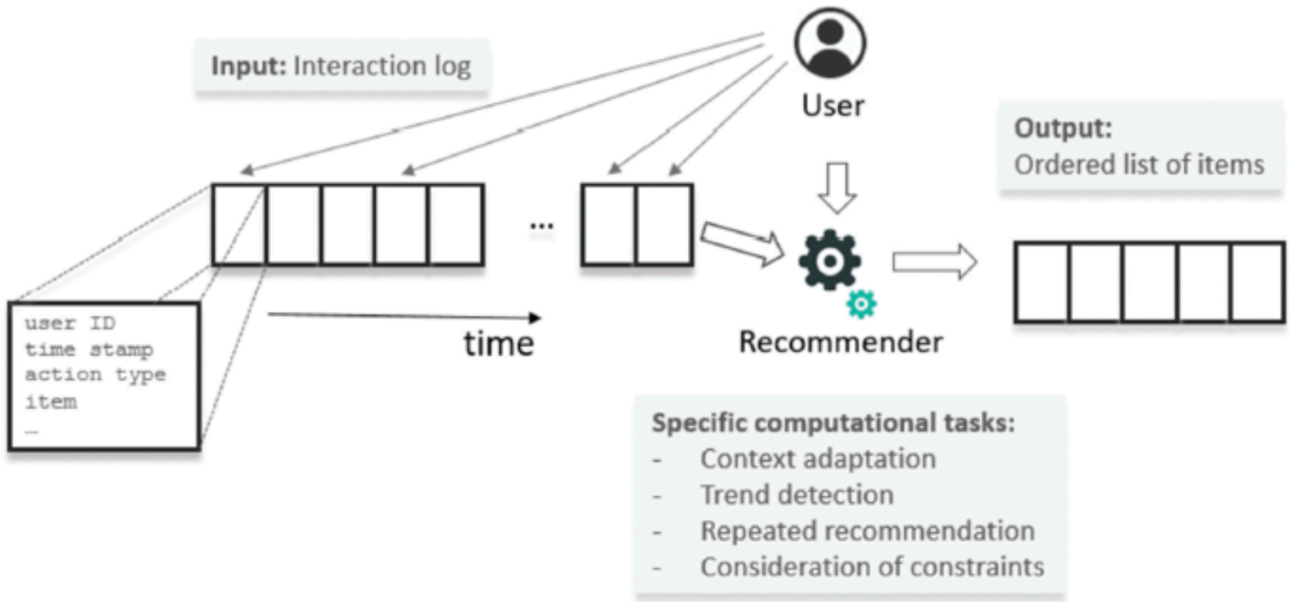


POI: Point of Interest

Background

Depending on the availability of historical data for individual users and the importance of focusing on the most recent interactions

- Last-N Interaction-based Recommendation
- Session-based Recommendation
- Session-aware Recommendation



Source: Sequence-aware recommender systems.

Background

Specifically,

- Last-N Interaction-based Recommendation: only consider the last N actions
- Session-based Recommendation: only the last sequence of actions is known and limited to a session (i.e., limited within a period of time)
- Session-aware Recommendation: have knowledge about both actions in the last session and historical behaviours (i.e., users can be identified)

Source: Sequence-aware recommender systems.

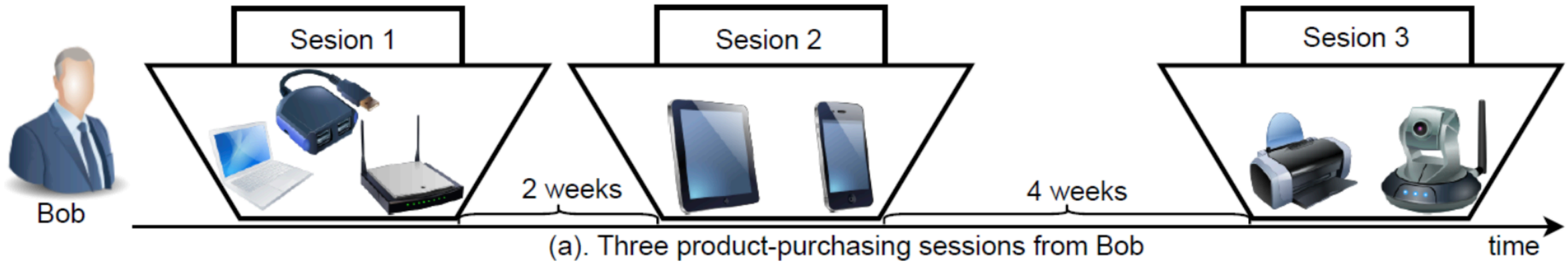
Session-based RS

- A session: a list of interactions with a clear boundary, while the interactions may be chronologically ordered (in ordered sessions), or unordered (in unordered sessions).
- A boundary: refers to the starting-ending interaction pair to start and end a specific session in a transaction event. An ordered (unordered) session refers to a session in which the interactions are (not) chronologically ordered.
- The session data from a given user usually consists of multiple sessions happening at different time and separated by multiple boundaries with non-identical time intervals between sessions

Source: UNSW

Session-based RS

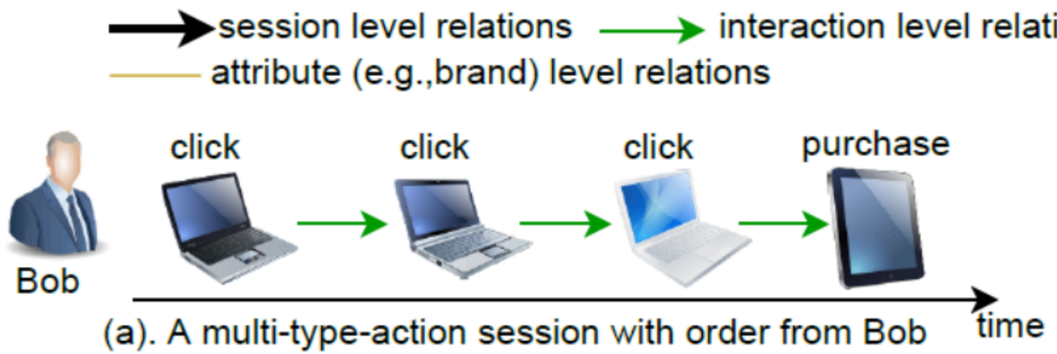
- An SBRS aims to predict either the unknown part of a session given the known part; or the future session given the historical session via learning the intra- or inter-session dependencies



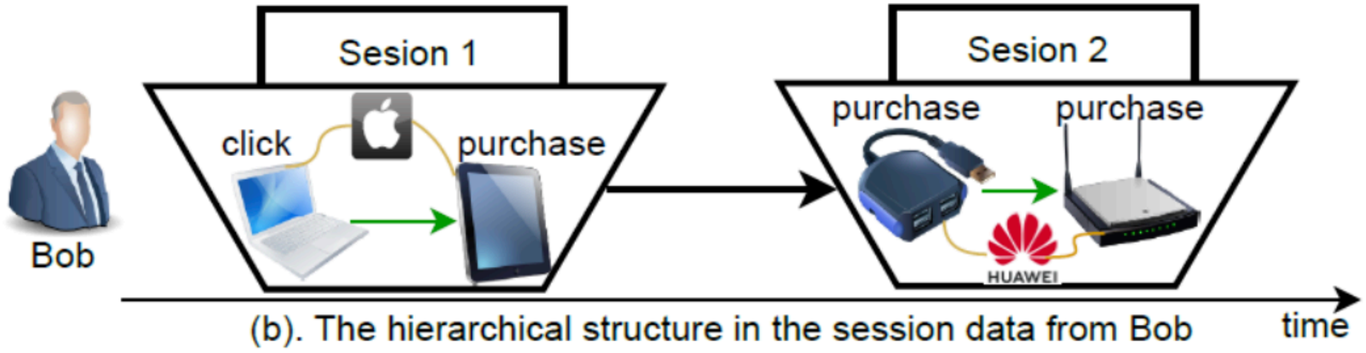
Source: A survey on session-based recommender systems

Session-based RS

- A session is a non-empty bounded list of interactions generated in a period of continuous time which may be connected with some user- (e.g., user ID) or session-specific (e.g., a session-ID or a cookie) information.



(a). A multi-type-action session with order from Bob



(b). The hierarchical structure in the session data from Bob

Source: A survey on session-based recommender systems

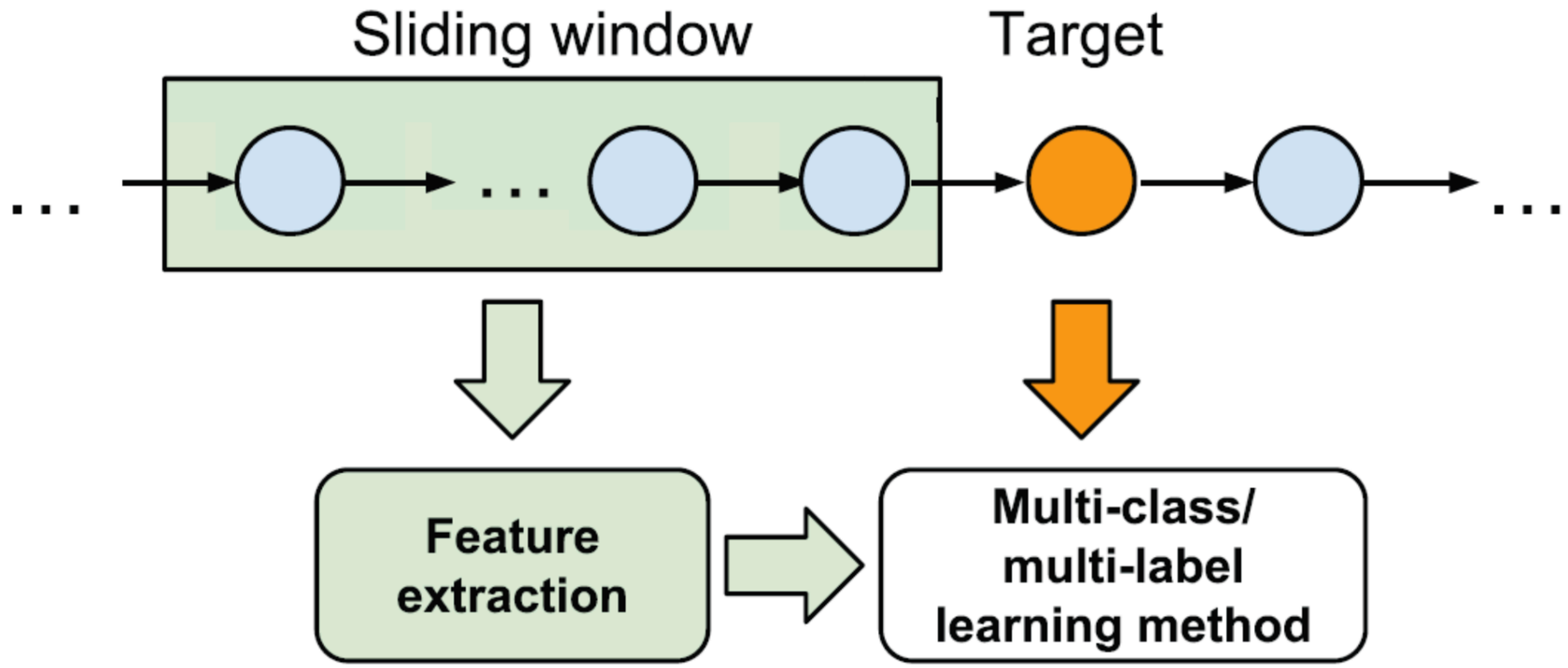
Sequence-aware RS

Some methods

- Neighbourhood based
- Association Rule Mining
- Supervised Learning
- Sequence Modeling
  - Markov chain; Markov Decision Process
- …

Sliding Window

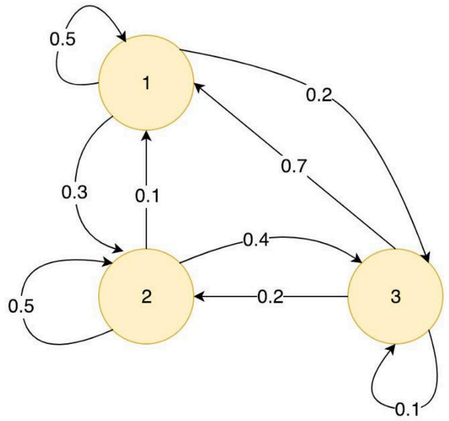
- Sliding window models convert the next-in-sequence prediction problem into a traditional supervised learning problem that can be solved with any classifier



Recommendation as Sequential Decision

At each time point, the recommender system makes a decision

- Recommend the user directions as to how to get from Location A to Location B
- Recommend instructions for mitigating a cyber threat (e.g., Apple Watch?)
- Recommend what accessories to purchase and how to find for the newly bought device

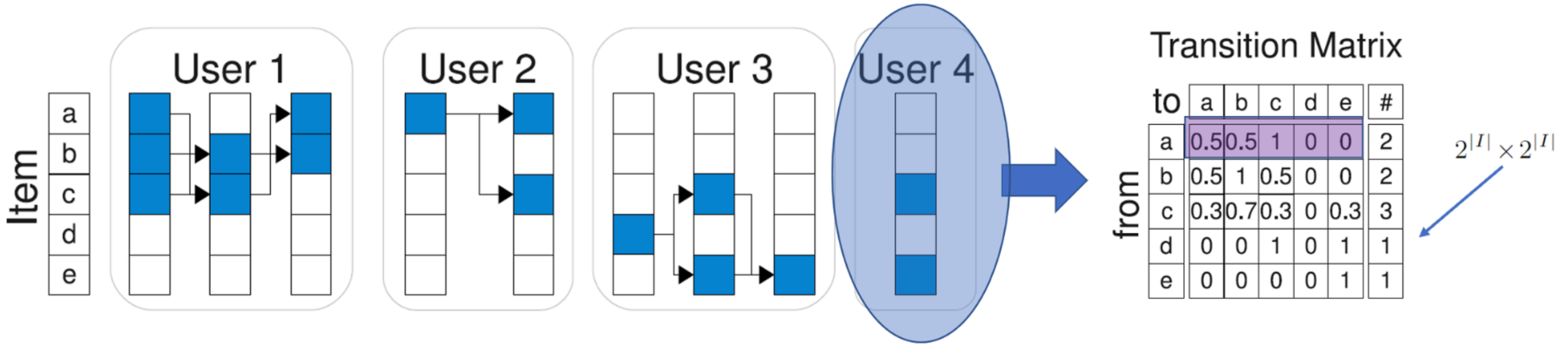


Markov process

	S_1	S_2	S_3
S_1	0.5	0.1	0.7
S_2	0.3	0.5	0.2
S_3	0.2	0.4	0.1

Transition matrix

MC-based RS



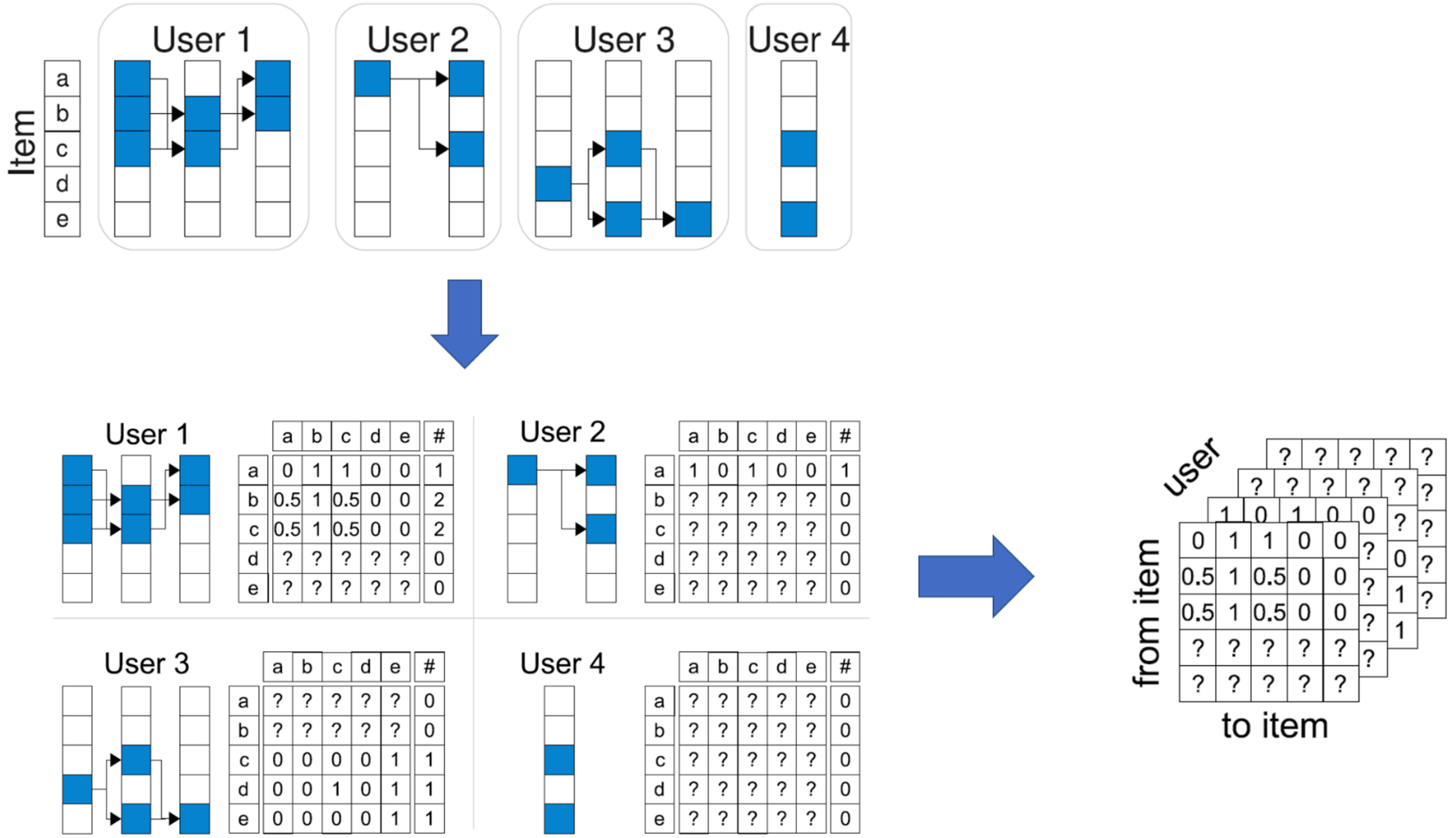
$$p(i \in B_t | B_{t-1}) := \frac{1}{|B_{t-1}|} \sum_{l \in B_{t-1}} p(i \in B_t | l \in B_{t-1})$$

$$p(a \in B_t | \{c, e\}) = 0.5(0.3 + 0.0) = 0.15$$
$$p(b \in B_t | \{c, e\}) = 0.5(0.7 + 0.0) = 0.35$$
$$p(c \in B_t | \{c, e\}) = 0.5(0.3 + 0.0) = 0.15$$
$$p(d \in B_t | \{c, e\}) = 0.5(0.0 + 0.0) = 0.00$$
$$p(e \in B_t | \{c, e\}) = 0.5(0.3 + 1.0) = 0.65$$

$$p(a \in B_t | \{c, e\}) = \frac{1}{2}(p(a|c) + p(a|e))$$

Source: Factorizing Personalized Markov Chains for Next-Basket Recommendation. WWW 2010.

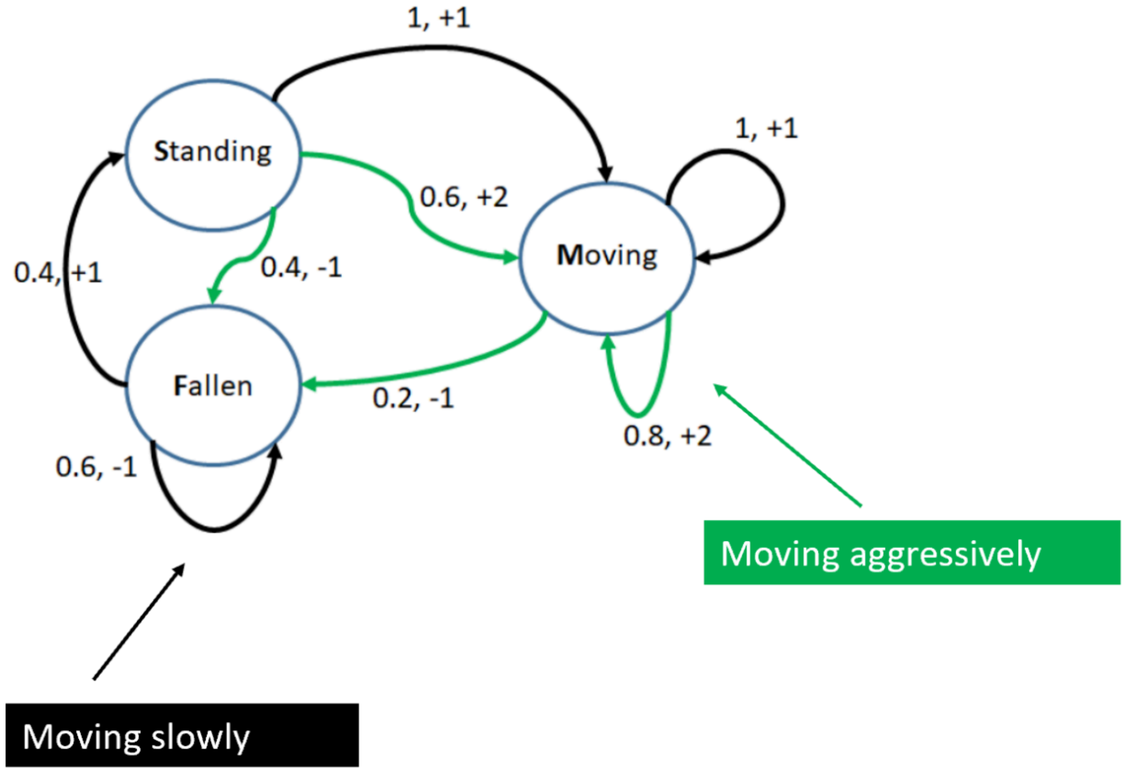
MC-based RS



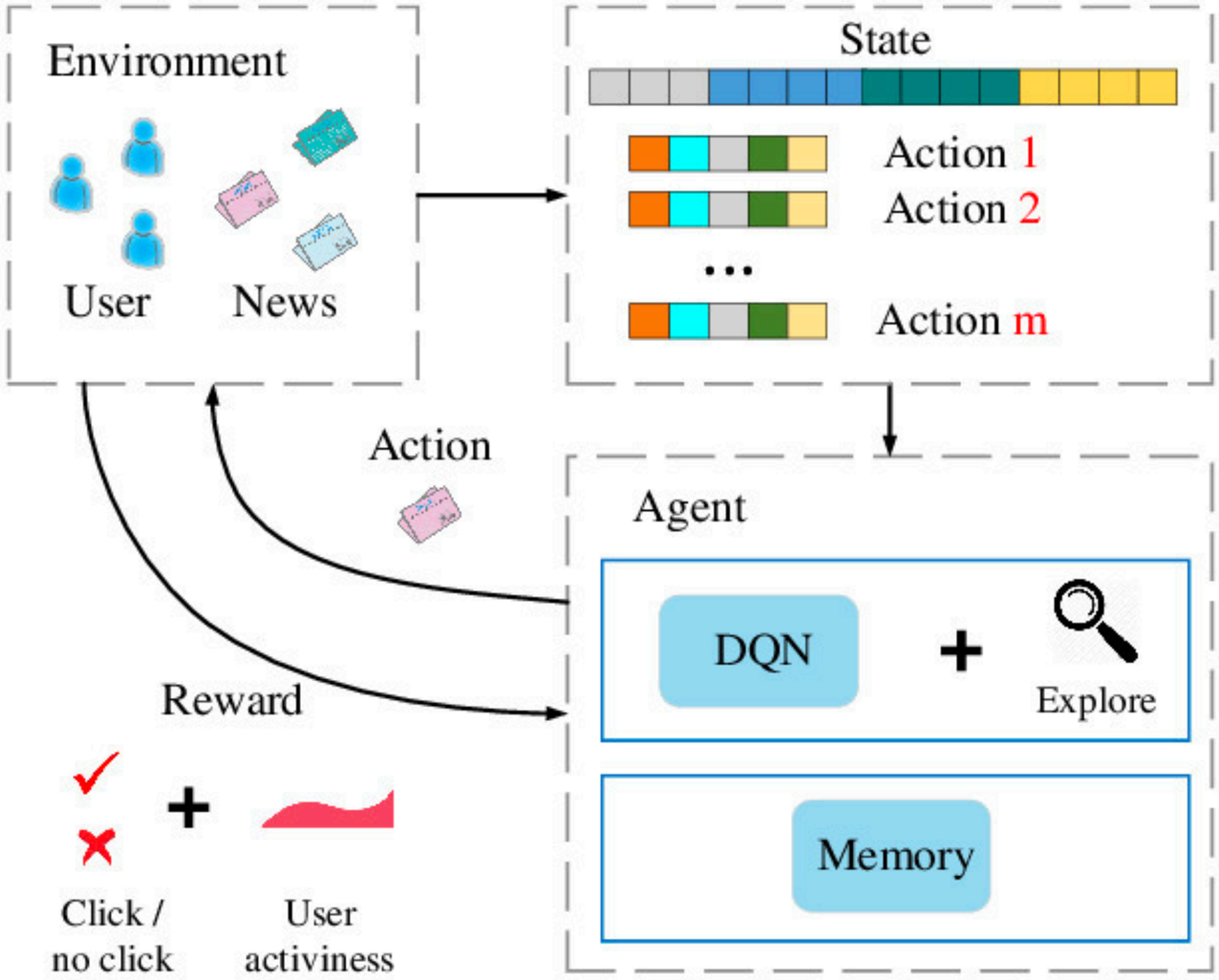
Source: Factorizing Personalized Markov Chains for Next-Basket Recommendation. WWW 2010.

MDP-based RS

- State: Standing, Moving, and Fallen
- Action: Moving slowly, moving aggressively



Reinforcement Learning in RS



Source: DRN: A Deep Reinforcement Learning Framework for News Recommendation. WWW 2018.

Summary

Summary

- Traditional model-based methods: MF, SVD, FM, LSH
- Learning to Rank: pointwise, pairwise, listwise, BPR
- Sequential recommendation; Session-based recommendation

Thank You!