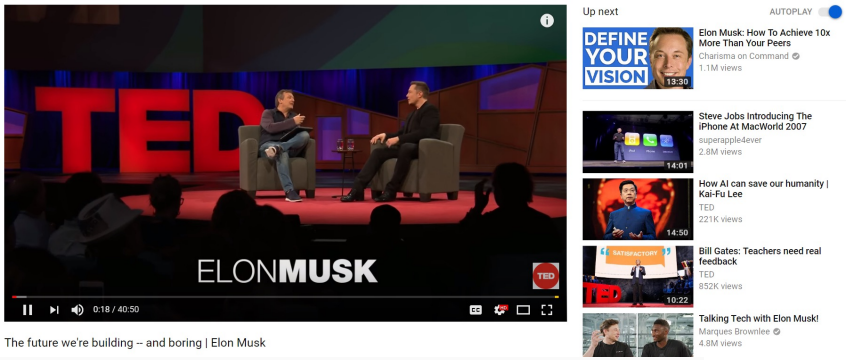
Recommender Systems

From traditional systems to neuronal one

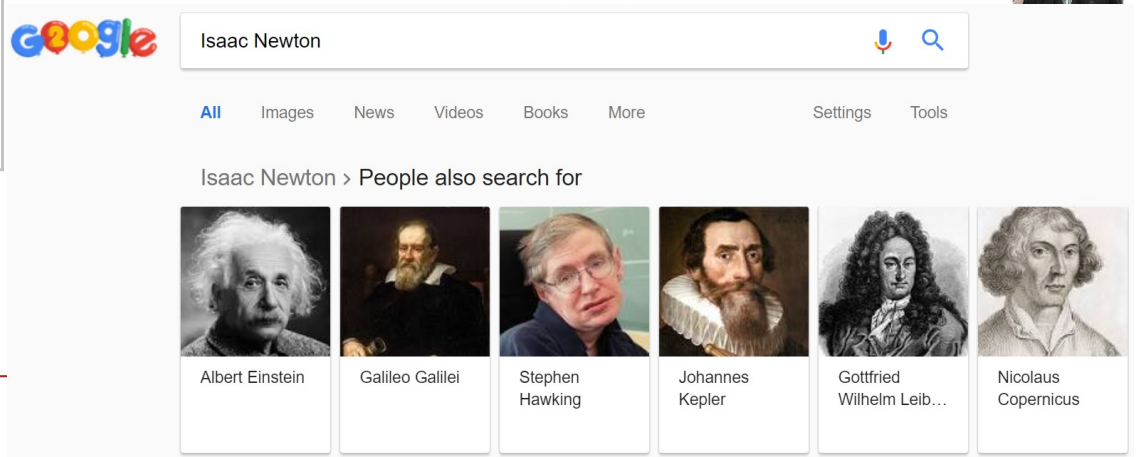
Michel.RIVEILL@univ-cotedazur.fr

Recommender Systems

- A recommender system (RS) helps users that have no sufficient competence or time to evaluate the, potentially overwhelming, number of alternatives offered by a web site.
  - In their simplest form, RSs recommend to their users personalized and ranked lists of items



The future we're building -- and boring | Elon Musk



The Impact of RecSys

- 35% of the purchases on Amazon are the result of their recommender system, according to McKinsey.
- During the Chinese global shopping festival of November 11, 2016, Alibaba achieved growth of up to 20% of their conversion rate using personalized landing pages, according to Alizila.
- Recommendations are responsible for 70% of the time people spend watching videos on YouTube.
- 75% of what people are watching on Netflix comes from recommendations, according to McKinsey

The Age of Recommendation

Search:
User ➡ Items

Recommend:

Items ➡ User
(push model)

Picture from: http://insidebigdata.com/wp-content/uploads/2014/06/Humor_recommender.jpg

Amazon: A personalized online store

Picture from: amazon.com

Recommender Problem

A good recommender

- Show
  - programming titles to a software engineer
  - baby toys to a new mother.

- Don't recommend items user already knows or would find anyway
  - Booking often recommends hotels in places where you have already done research. Is this relevant?

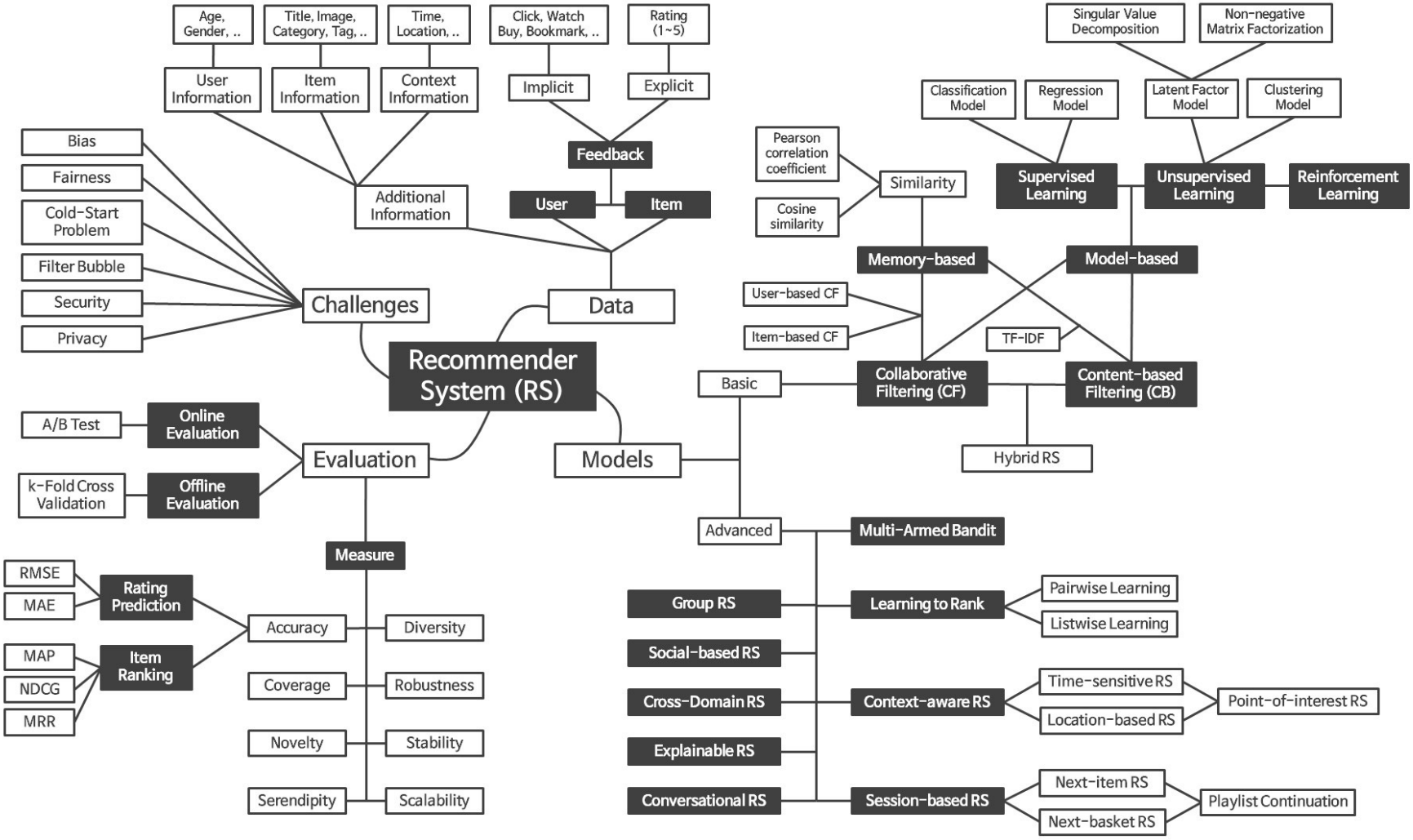
- Expand user's taste without offending or annoying him/her…
  - Try to propose new things, basing your proposals on what you learn from the user
    - is it someone who has varied tastes
    - or on the contrary quite homogeneous?

Recommender Problem

Challenges

- Huge amounts of data
  - tens of millions of customers
  - millions of distinct catalog items.
- Results are required to be returned in real time (<1-2 seconds).
  - You don't have enough time to calculate a lot of things
- New customers have limited information.
  - Some platforms ask questions to build an initial profile
- Old customers can have a lot of information.
  - Customer data is volatile
  - How to know what information is no longer relevant ?

A curated list of awesome Recommender System

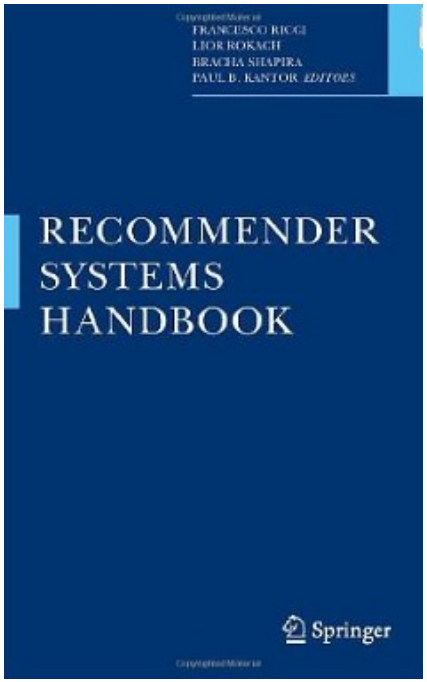


About this lecture

- Will give you a **very short introduction** to the field of Recommender Systems

- **"FIT" → How do recommender systems (RS) work ?**
  - How can you compute recommendations?
    - From the basis
      - Non personalized recommendation
    - To more accurate proposal
      - Content-based filtering
      - Collaborative filtering
    - It's just an introduction…

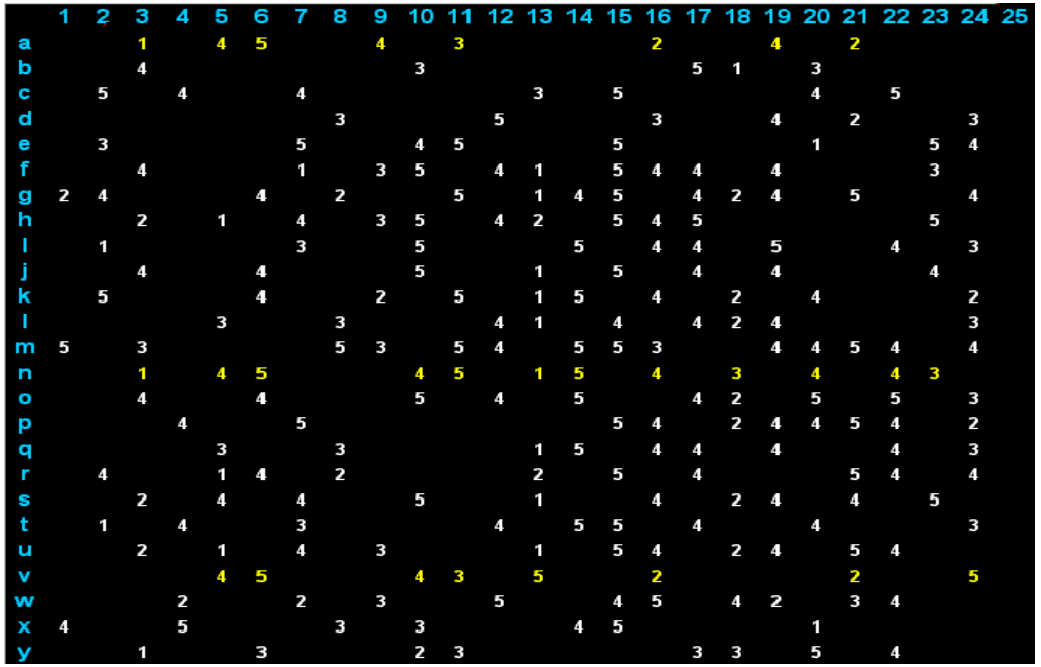
- **"EVALUATE" → How do we measure their success?**
  - How can we know that the recommendations are good?



https://edyaaleh.files.wordpress.com/2016/02/recommendersystemshandbook.pdf

Problem characterization

- Given
  - The profile of the "active" user
  - The description of a set of items
  - And possibly some situational context
- Compute
  - A relevance (ranking) score for each item
  - And select for a user
    - A list of recommendable items
    - The most recommendable items
- The problem …
  - … is mainly a Matrix completion problem



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a			1		4	5			4		3					2			4		2				
b			4							3							5	1		3					
c		5		4			4						3		5					4		5			
d								3				5				3			4		2				3
e		3					5			4	5				5					1			5	4
f			4				1		3	5		4	1		5	4	4		4				3		
g	2	4				4		2			5		1	4	5		4	2	4		5				4
h			2		1		4		3	5		4	2		5	4	5						5		
i		1					3			5				5		4	4		5			4		3	
j			4			4				5			1		5		4		4				4		
k		5				4			2		5		1	5		4		2		4				2	
l					3			3				4	1		4		4	2	4					3	
m	5		3					5	3		5	4		5	5	3			4	4	5	4		4	
n			1		4	5				4	5		1	5		4		3		4		4	3		
o			4			4				5		4		5			4	2		5		5		3	
p				4			5								5	4		2	4	4	5	4		2	
q					3			3					1	5		4	4		4			4		3	
r		4			1	4		2					2		5		4				5	4		4	
s			2		4		4			5			1			4		2	4		4		5		
t		1		4			3					4		5	5		4			4				3	
u			2		1		4		3				1		5	4		2	4		5	4			
v					4	5				4	3		5			2					2			5	
w				2			2		3			5			4	5		4	2		3	4			
x	4			5				3		3				4	5					1					
y			1			3				2	3						3	3		5		4			

The problem

1. Collect the data

User	Item	Score	Time
0	0	5	1
2	1	4	2
4	2	5	3
0	2	2	4
2	0	5	5
Etc.			

The problem

1. Collect the data
2. Construct the pivot matrix

User	Item	Score	Time
0	0	5	1
2	1	4	2
4	2	5	3
0	2	2	4
2	0	5	5
Etc.			

Item / User	0	1	2	3	4
0	5				
1					
2					
3					
4					

The problem

1. Collect the data
2. Eventually construct the pivot matrix

User	Item	Score	Time
0	0	5	1
2	1	4	2
4	2	5	3
0	2	2	4
2	0	5	5
Etc.			

Item / User	0	1	2	3	4
0	5				
1					
2		4			
3					
4					

The problem

1. Collect the data
2. Eventually construct the pivot matrix

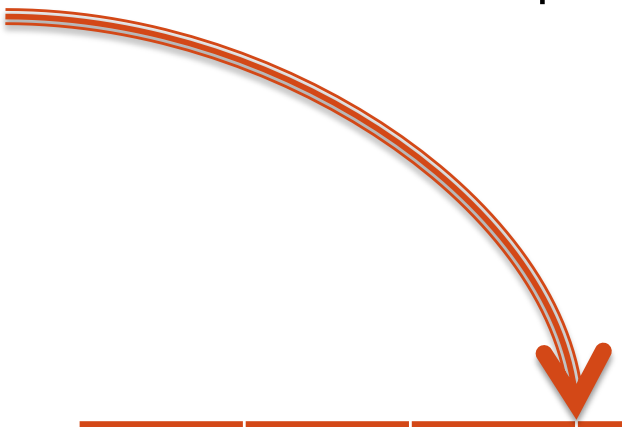
User	Item	Score	Time
0	0	5	1
2	1	4	2
4	2	5	3
0	2	2	4
2	0	5	5
Etc.			

Item / User	0	1	2	3	4
0	5		2		
1					
2	5	4			
3					
4			5		

The problem

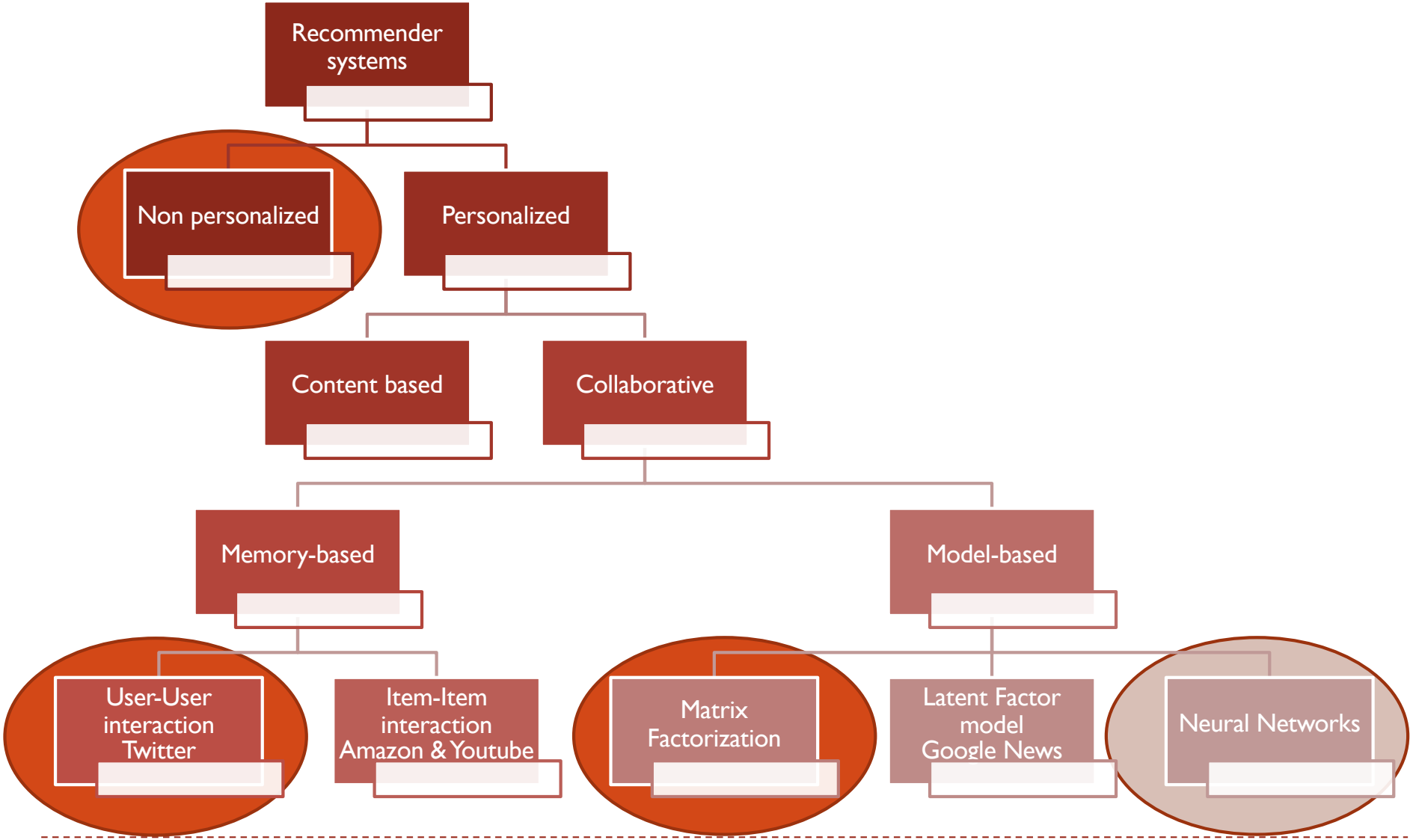
1. Collect the data
2. Eventually construct the pivot matrix
3. Predict missing scores for the desired used
   - Depend of the strategy

User	Item	Score	Time
0	0	5	1
2	1	4	2
4	2	5	3
0	2	2	4
2	0	5	5
Etc.			



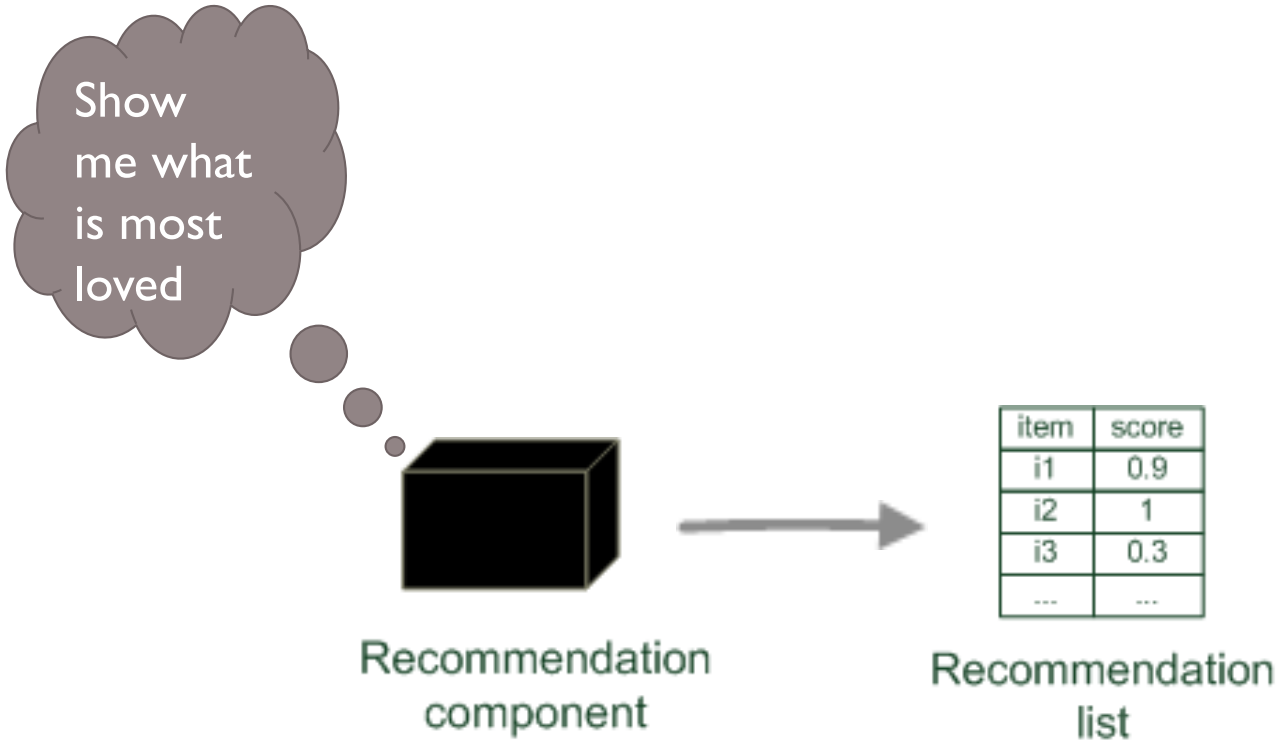
Item / User	0	1	2	3	4
0	5		2		
1					
2	5	4			
3					
4			5		

Paradigms of recommender systems



Non personalized recommendation

Non personalized recommendation



Non personalized recommendation Approach 1 = mean rating

- Goal: provides a list
  - For all user the list is the same
  - But, what do you put at the top of the list?
- A simple approach: rank by score
  - Score for item i : $r_i = \frac{\sum_u r_{ui}}{n}$

Item / User	0	1	2	3	4
0	5	3	2		
1		5	3	3	
2	5	4	2	1	
3		3		1	2
4			5	2	
Item mean grade		3.75			

(3+5+4+3)/4

Non personalized recommendation Approach 1 = mean rating

- Goal: provides a list
  - For all user the list is the same
  - But, what do you put at the top of the list?
- A simple approach: rank by score
  - Score for item i : $r_i = \frac{\sum_u r_{ui}}{n}$

Item / User	0	1	2	3	4
0	5	3	2	1,75	2
1	5	5	3	3	2
2	5	4	2	1	2
3	5	3	3	1	2
4	5	3.75	5	2	2
Item mean grade	5	3.75	3	1,75	2

Ranking list for user 4: [0, 1, 4]

Non personalized recommendation Approach 1 = mean rating

- All unseen items for all users have the same rating
- Ranking list for user 4 = [0, 1, 4]

Item / User	0	1	2	3	4
0	5	3	2	1,75	2
1	5	5	3	3	2
2	5	4	2	1	2
3	5	3	3	1	2
4	5	3.75	5	2	2
Item mean grade	5	3.75	3	1,75	2

Ranking list for user 4: [0, 1, 4]

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_u)}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2			
1		5	3	3		3.66
2	5	4	2	1		
3		3		1	2	
4			5	2		
$\frac{\sum_u (r_{ui} - r_u)}{n}$						

STEP 1: Calculate the mean rating for each user (5+3+3)/3

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user
    - $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2			
1		5	3	3		3.66
2	5	4	2	1		
3		3		1	2	
4			5	2		
$\frac{\sum_u (r_{ui} - r_u)}{n}$						

STEP 1: Calculate the mean rating for each user (5+3+3)/3

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user
    - $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2			3.33
1		5	3	3		3.66
2	5	4	2	1		3
3		3		1	2	2
4			5	2		3.5
$\frac{\sum_u (r_{ui} - r_u)}{n}$						

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user : $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$
  - Step 2: compute the normalized rating for each movie
    - $\bar{r}_i = \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2			3.33
1		5	3	3		3.66
2	5	4	2	1		3
3		3		1	2	2
4			5	2		3.5
$\frac{\sum_u (r_{ui} - r_u)}{n}$	1.835					

STEP 2: compute the normalized rating for each movie
((5-3.33)+(5-3)) / 2 =

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user : $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$
  - Step 2: compute the normalized rating for each movie
    - $\bar{r}_i = \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2			3.33
1		5	3	3		3.66
2	5	4	2	1		3
3		3		1	2	2
4			5	2		3.5
$\frac{\sum_v (r_{vi} - r_v)}{n}$	1.835	0.75	-0.37	-1.29	0	

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user : $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$
  - Step 2: compute the normalized rating for each movie: $\bar{r}_i = \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step 3: add the mean user rating to the item rating
    - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2			3.33
1	5.5	5	3	3		3.66
2	5	4	2	1		3
3	3.8	3		1	2	2
4			5	2		3.5
$\frac{\sum_u (r_{ui} - r_u)}{n}$	1.835	0.75	-0.37	-1.29	0	

STEP 3: add the mean user rating to the item rating
- 3.66+1.835 for item 0/user1
- 2+1.835 for item 3/user1

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user : $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$
  - Step 2: compute the normalized rating for each movie: $\bar{r}_i = \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step 3: add the mean user rating to the item rating
    - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2	3.33-1.29	3.33-0	3.33
1	5.5	5	3	3	3.66	3.66
2	5	4	2	1	3	3
3	3.8	3	2-0.37	1	2	2
4	5.335	4.25	5	2	3.5	3.5
$\frac{\sum_u (r_{ui} - r_u)}{n}$	1.835	0.75	-0.37	-1.29	0	

STEP 3: add the mean user rating to the item rating
- 3.66+1.835 for item 0/user1
- 2+1.835 for item 3/user1

Non personalized recommendation Approach 2 = normalized mean rating

- A simple approach: rank by score with normalisation
  - $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step1 : calculate the mean rating for each user : $\bar{r}_u = \frac{\sum_i (r_{ui})}{n}$
  - Step 2: compute the normalized rating for each movie: $\bar{r}_i = \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$
  - Step 3: add the mean user rating to the item rating: $r_{ui} = \bar{r}_u + \frac{\sum_v (r_{vi} - \bar{r}_v)}{n}$

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2	2.04	3.33	3.33
1	5.5	5	3	3	3.66	3.66
2	5	4	2	1	3	3
3	3.8	3	1,63	1	2	2
4	5.335	4.25	5	2	3.5	3.5
$\frac{\sum_u (r_{ui} - r_u)}{n}$	1.835	0.75	-0.37	-1.29	0	

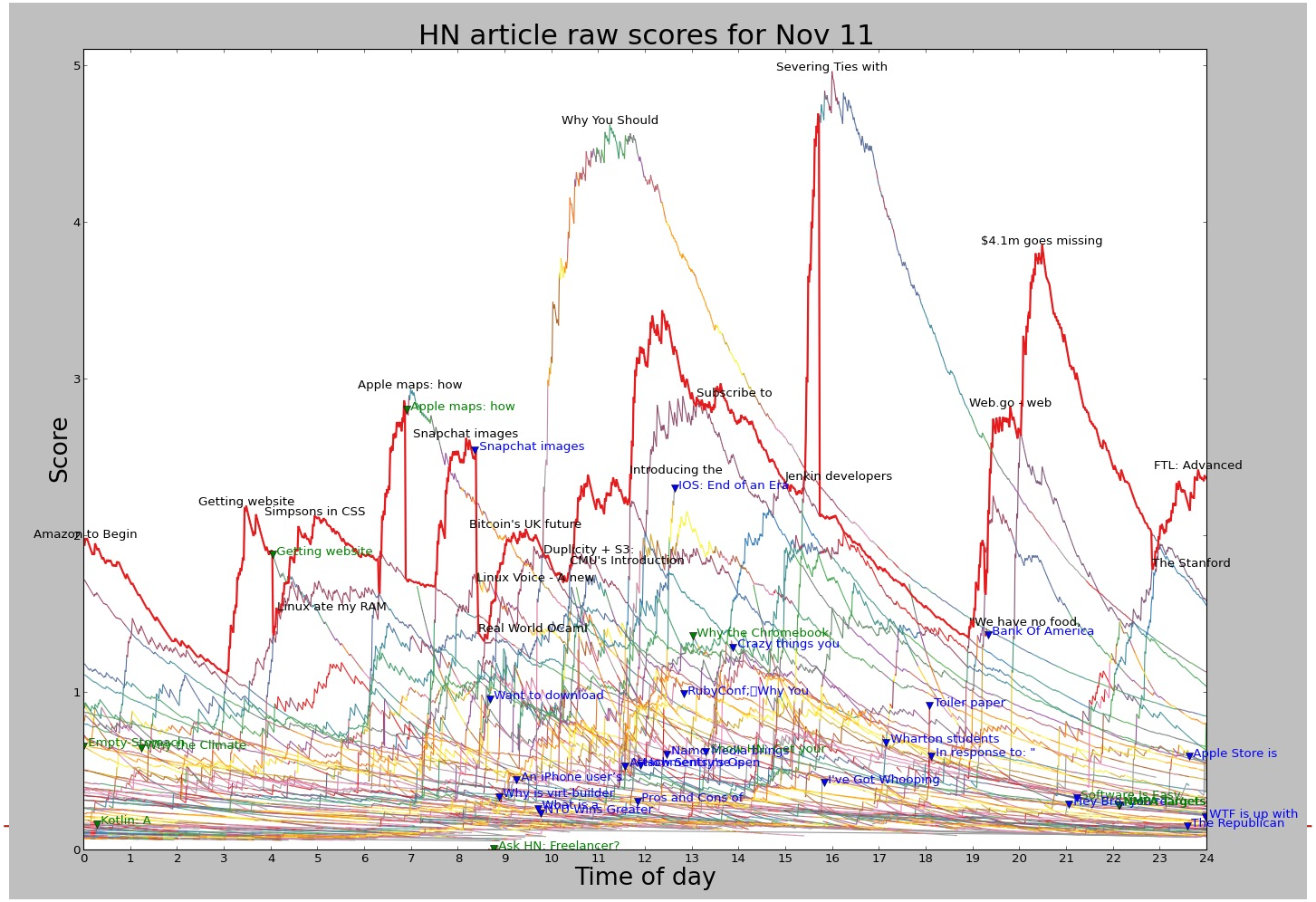
Non personalized recommendation Approach 2 = normalized mean rating

- The rating for unseen depend of the user
- Ranking list for user 4: [0, 1, 4]

Item / User	0	1	2	3	4	Mean rating r_u
0	5	3	2	2.04	3.33	3.33
1	5.5	5	3	3	3.66	3.66
2	5	4	2	1	3	3
3	3.8	3	1,63	1	2	2
4	5.335	4.25	5	2	3.5	3.5
$\frac{\sum_u (r_{ui} - r_u)}{n}$	1.835	0.75	-0.37	-1.29	0	

If the time is important....

- Hacker News algorithms
  - http://www.righto.com/2013/11/how-hacker-news-ranking-really-works.html
  - Score : $\frac{(votes-1)^{0.8}}{(age+2)^{1.8}}$



Personalized recommendation Collaborative filtering

Memory based

Memory-based collaborative filtering

- Use the rating matrix
  - find users/or items that are similar to the active user/or items
  - and use their rating to predict new rating

- Advantage
  - Quality of precision are rather good
  - Relatively simple algorithm to implement for any situation
  - New data can be added easily and incrementally

- Disadvantage
  - It depend on human ratings
  - Performance decreases when data gets sparse
  - Have scalability problem with large dataset

- We will detail only the user-user approach

User-Based Algorithms (Breese et al, UAI98)

- Predicted vote for "active user" *u* for item j is a weighted sum
  - $\widehat{r_{uj}} = \sum_{v \in U} w(u,v)\ r_{vj}$

Similarity weights between users
- Near 1 : users u and k is similar
- Near 0 : users u and k have no similarity

- As in the previous approaches
  - it is also possible to normalize the ratings.
  - $\widehat{r_{uj}} = \bar{r}_u + \sum_{v \in U} w(u,v)\ (r_{vj} - \bar{r}_v)$

User-Based Algorithms (Breese et al, UAI98)

- weighted sum $\rightarrow w(u,v) \in [0, 1]$ or $[-1, +1]$
  - Many possibilities
  - K-nearest neighbor

$$w(a,i) = \begin{cases} 1 & \text{if } i \in \text{neighbors}(a) \\ 0 & \text{else} \end{cases}$$

  - Pearson correlation coefficient (from Grouplens) → [-1, +1]

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

  - Cosine similarity (from IR) → [0, 1]

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

How to calculate r_{13} ?

- r_{13} = ranking for user1 and item 3 without normalization
- $r_{13}=\alpha * (r_{11} * sim(I_1, I_3) + r_{12} * sim(I_2, I_3) + r_{14} * sim(I_4, I_3) + r_{15} * sim(I_5, I_3))$
- Where α is a normalization factor, which is $\frac{1}{\sum sim(Ii,Ij)}$

Generalization (without normalization)

- User based: $r_{ij} = \frac{\sum_k r_{ik} * sim(Ui,Uk)}{\sum_k sim(U_i,Uk)}, \mathrm{k} \in similar\ User$
- Item based: $r_{ij} = \frac{\sum_k r_{kj} * sim(I_j,I_k)}{\sum_k sim(I_j,I_k)}, \mathrm{k} \in similar\ Item$

- Rating matrix is almost stable

  → pre-calculate all similarity (off line calculation).
  - Between Items for Item based
  - Between Users for User based

Offline computation : Online Recommendation

For Item-based recommendation

1. Offline Computation (each night ? Each week ? Each month ?)
   - builds a similar-items table which is extremely time intensive, O(N2M)
     - M is the number of customers ± 10 millions
     - N is the number of items ± 1 millions
   - In practice, it's closer to O(NM), as most customers have very few purchases
   - Sampling customers can also reduce runtime even further with little reduction in quality.
2. Online Recommendation:
   - Given a similar-items table, the algorithm
     - finds items similar to each of the user's purchases and ratings,
     - aggregates those items, and then
     - recommends the most popular or correlated items.

- For User based, exchange Item and User
  - Item similarity is more stable than User similarity

- M >> N

→ O(N2M) << O(M2N)

→ Item based is more efficient

→ chosen by Amazon / Youtube

User-based or Item-based Approach

- PRO
  - Few modeling assumptions
  - Few tuning parameters to learn
  - Easy to explain to users → linked recommendation
    - Dear Amazon.com Customer, We've noticed **that customers who have purchased** or rated How Does the Show Go On: An Introduction to the Theater by Thomas Schumacher **have also purchased** Princess Protection Program #1: A Royal Makeover (Disney Early Readers).

- CONS
  - Computationally expensive, $O(M^2N)$ for User based in the worst case, where
    - M is the number of customers ± 10 millions
    - N is the number of items ± 1 millions
  - Dimensionality reduction can increase the performance,
    - BUT, also reduce the quality of the recommendation
  - For very large data sets
    - the algorithm encounters severe performance and scaling issues

Work on scalability: Cluster Models

- Approach
  - Divide the customer base into many segments and treat the task as a classification problem
  - Assign the user to the segment containing the most similar customers
  - Uses the purchases and ratings of the customers in the segment to generate recommendations
  - Cluster models have better online scalability and performance than collaborative filtering because they compare the user to a controlled number of segments rather than the entire customer base.
- Problems
  - Quality of the recommendation is low
  - The recommendations are less relevant because the similar customers that the cluster models find are not the most similar customers
  - To improve quality, it needs online segmentation, which is almost as expensive as finding similar customers using collaborative filtering

Comparative results:

- The MovieLens dataset contains
  - 1 million ratings
  - 6,040 users
  - 3,900 movies.
- The best overall results are reached by the item-by-item based approach. It needs 170 seconds to construct the model and 3 seconds to predict 100,000 ratings.

	User Based	Model Based	Item Based
model construction time (sec.)	730	254	170
prediction time (sec.)	31	1	3
MAE	0.6688	0.6736	**0.6382**

Table from: Candillier, L., Meyer, F., & Boull'e Marc. (2007). Comparing state-of-the-art collaborative filtering systems

Personalized recommendation Collaborative filtering

Model based

Model-based collaborative filtering

- Find patterns based on training data, and these are used to make predictions for real data
- Extract some information from dataset, and use that as a "model" to make recommendations without having to use complete dataset every time
- Advantage
  - Handle sparsity better than memory-based ones
  - Scalable with large dataset
  - Improve prediction speed
- Disadvantage
  - Expensive model building
  - Can lose useful information due to reduction models

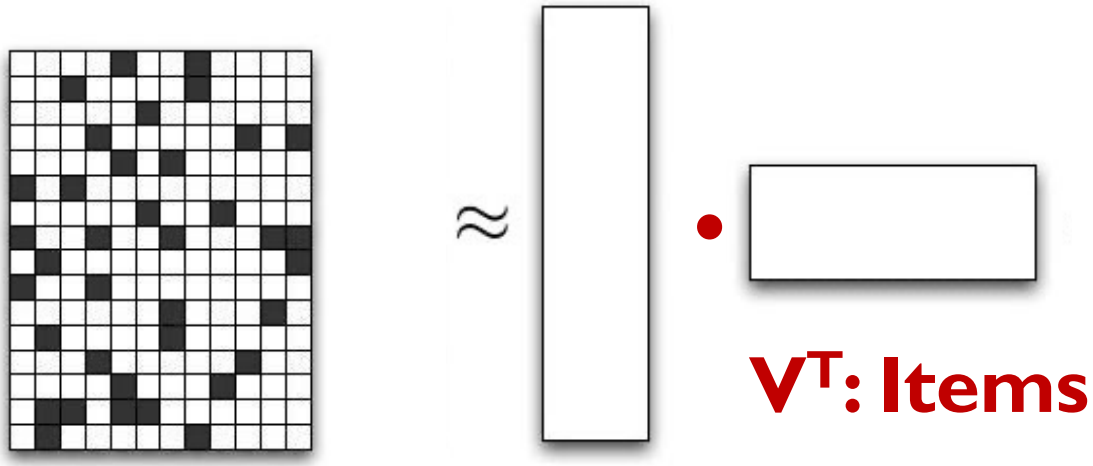
Latent based recommendation

- Models with latent classes of items and users
  - Individual items and users are assigned to either a single class or a mixture of classes

- Could be implemented by
  - Latent Factor Models
    - Items and users described by unobserved factors
    - **Matrix factorization** or SVD decomposition
  - Probabilistic latent semantic analysis (PLSA)
  - **Neural networks**

Matrix factorization

- Item and users are associated with a factor vector •
  - Dot product captures the user's estimated interest in the item
  - $\widehat{r_{ui}} = ViTU_u$



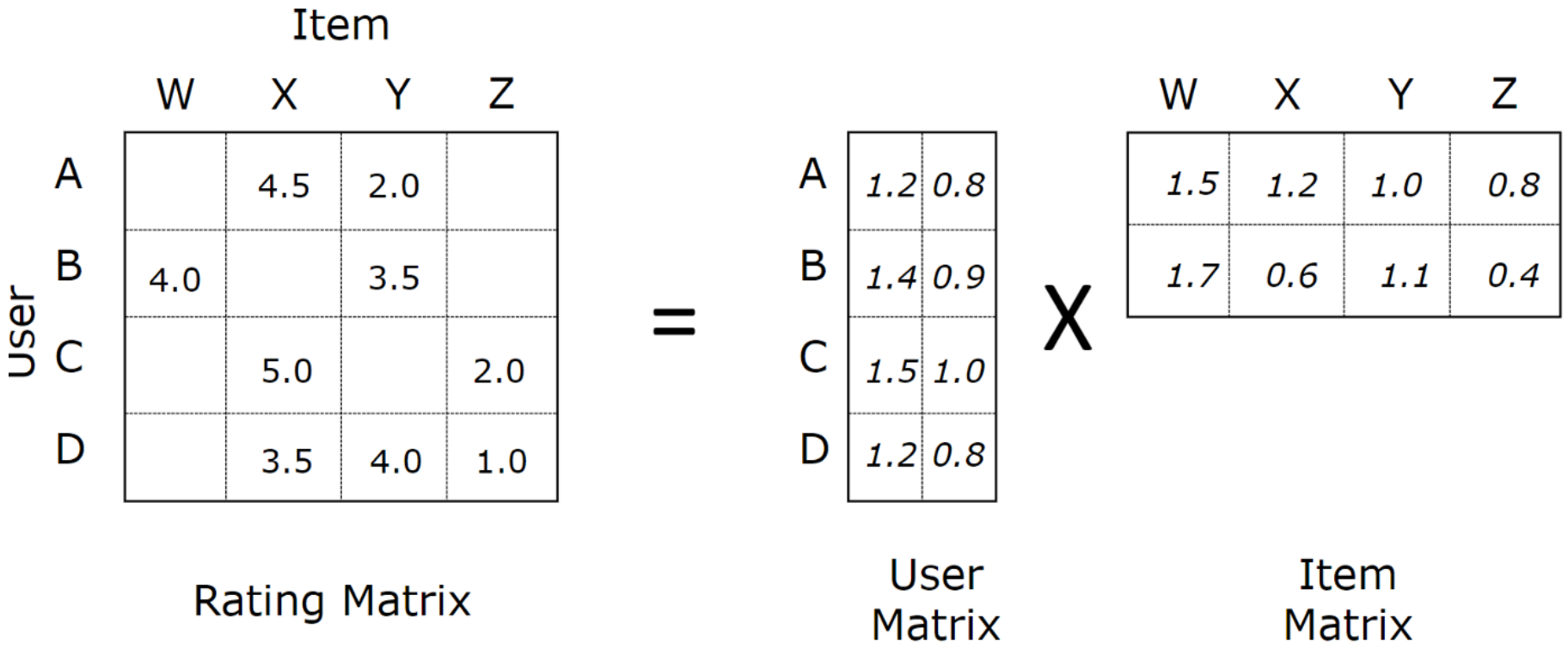
- Challenge: How to compute a mapping of users and items to factor vectors?

Matrix factorization
Leaning algorithms

- Stochatic gradient descent
  - Calculation of the prediction error
    - Error = actual rating – predicted rating: $e_{ui} = rui - ViTU_u$
  - Modification of parameters relative to prediction error
    - $V_i \leftarrow Vi + \gamma(euiUu - \lambda Vi)$
    - $U_u \leftarrow U_u + \gamma(euiVi - \lambda U_u)$
  - By magnitude proportional to γ
  - In the opposite direction of the gradient

- Main problem
  - Fix the size of the factor vectors

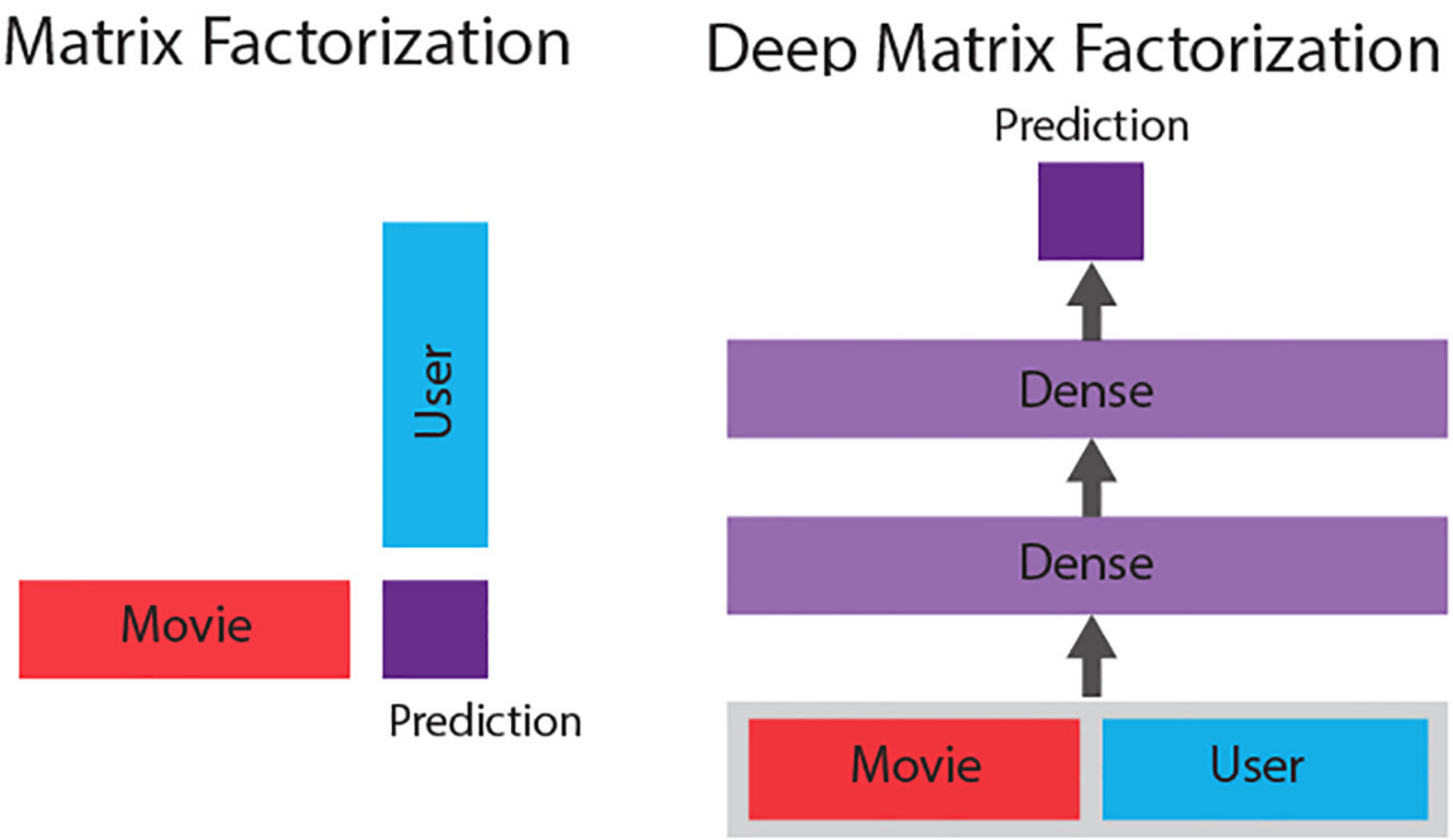
Matrix factorization



[[3.16 1.92 2.08 1.28]
[3.63 2.22 2.39 1.48]
[3.95 2.4 2.6 1.6]
[3.16 1.92 2.08 1.28]]

Reconstructed matrix

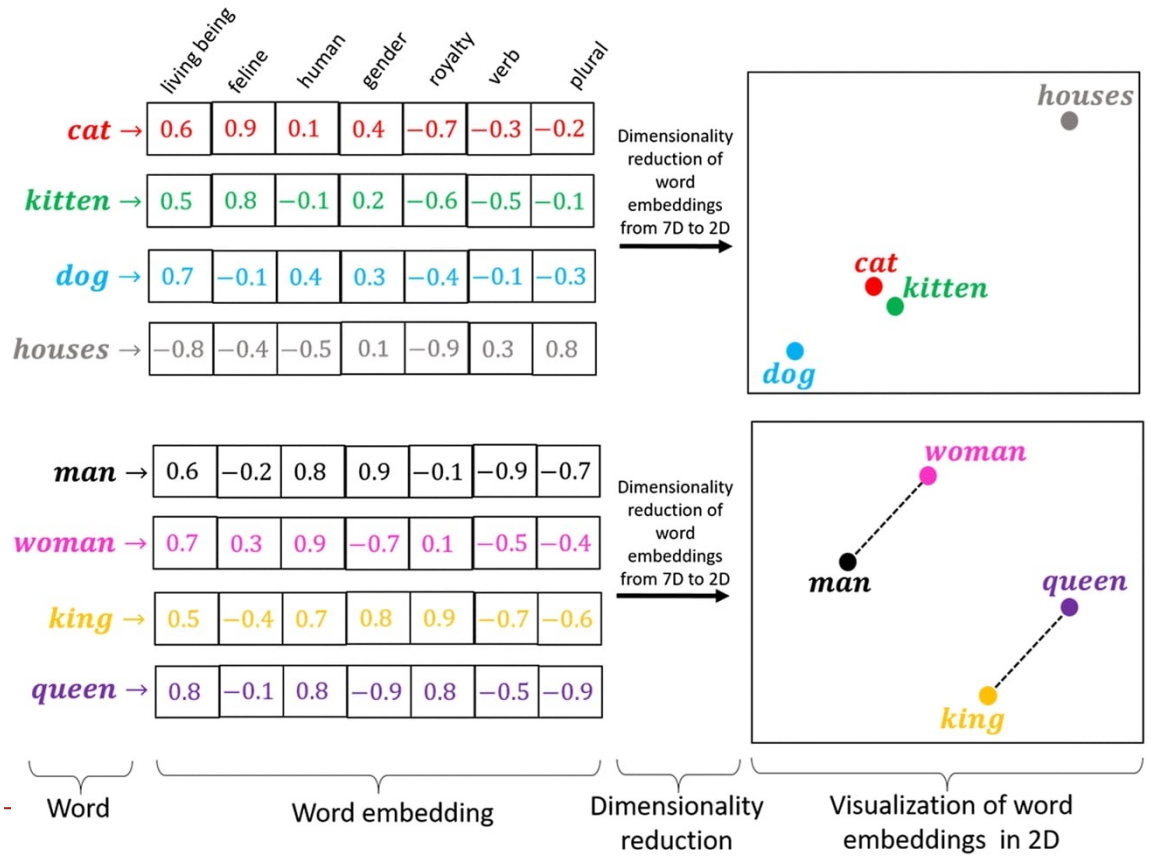
Matrix factorization by neural network



Embedding

- Transform a « label » to a « vector »
  - Until now : Label → integer → OneHotVector
  - OneHotVector
    - Very sparse: a lot of zeros, only one one
    - No semantic: Sim(l1, l2) = 0 if l1 != l2

- Vector embedding
  - Integrating "semantics » into vectors
  - More compact vector
    - Size << vocabulary size
  - For example, in NLP
    - If two words have a similar meaning
    - They will have a similar vector



In Keras, use Embedding layer

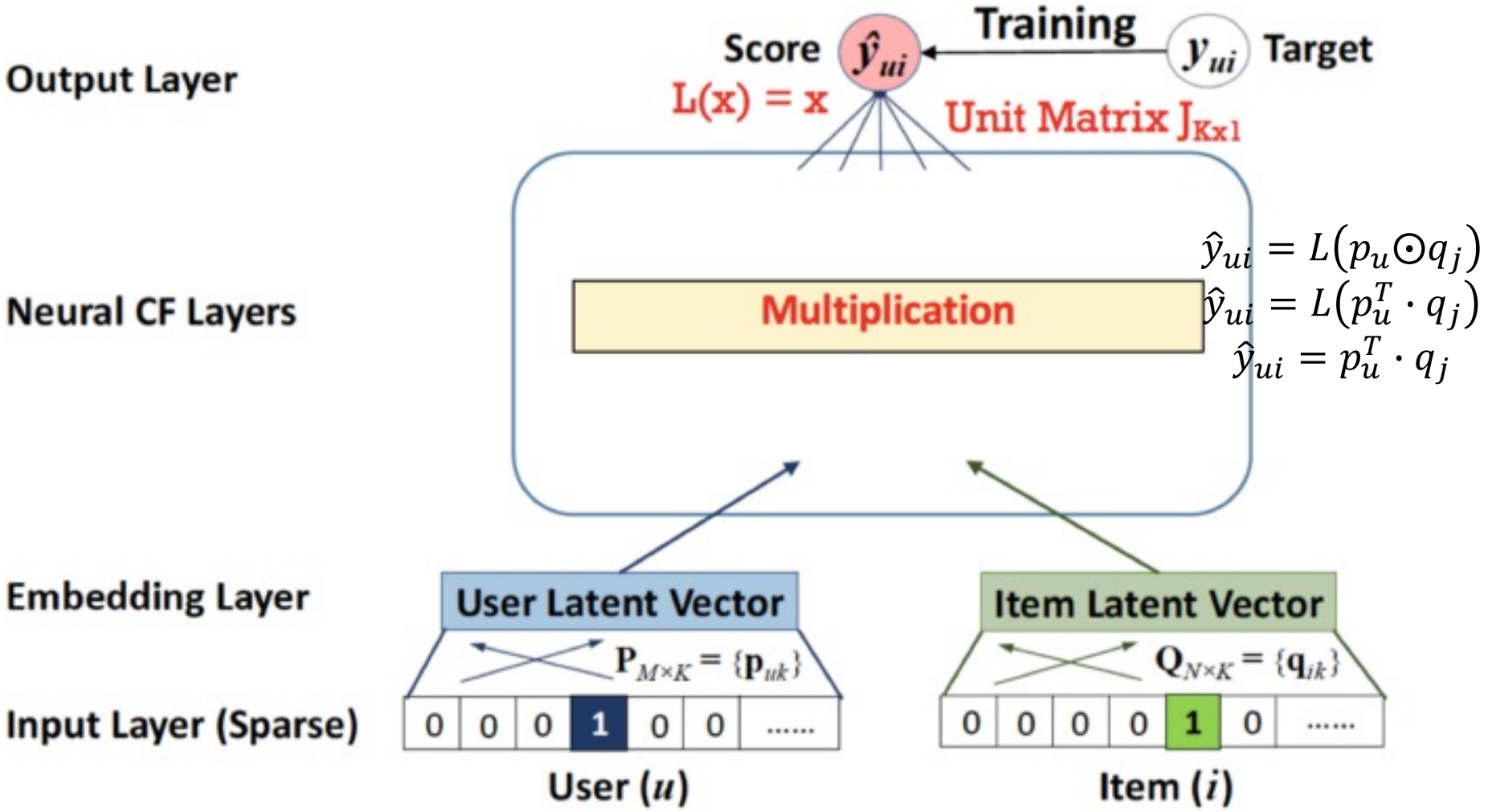
```
model = Sequential([
        layers.Embedding(len(tokenizer.index_word)+1,
                embed_len,
                input_length=max_tokens),
        layers.Flatten(),
        layers.Dense(128, activation="relu"),
        layers.Dense(len(target_classes), activation="softmax") ] )
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 25)            1800075
_________________________________________________________________
flatten (Flatten)            (None, 1250)              0
_________________________________________________________________
dense (Dense)                (None, 128)               160128
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
```

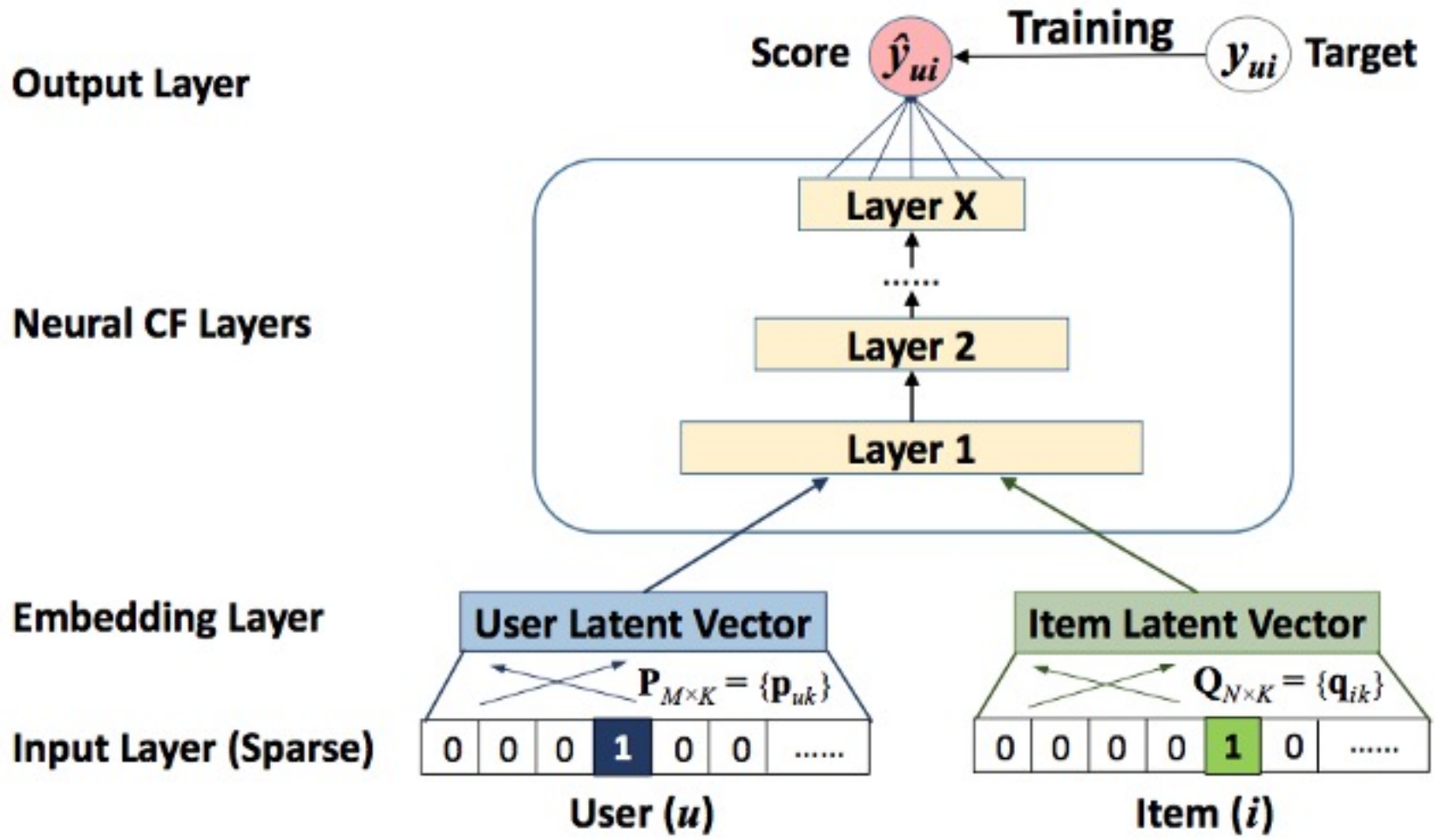
Matrix factorization with Neural Network

GMF (Generalized Matrix Factorization).



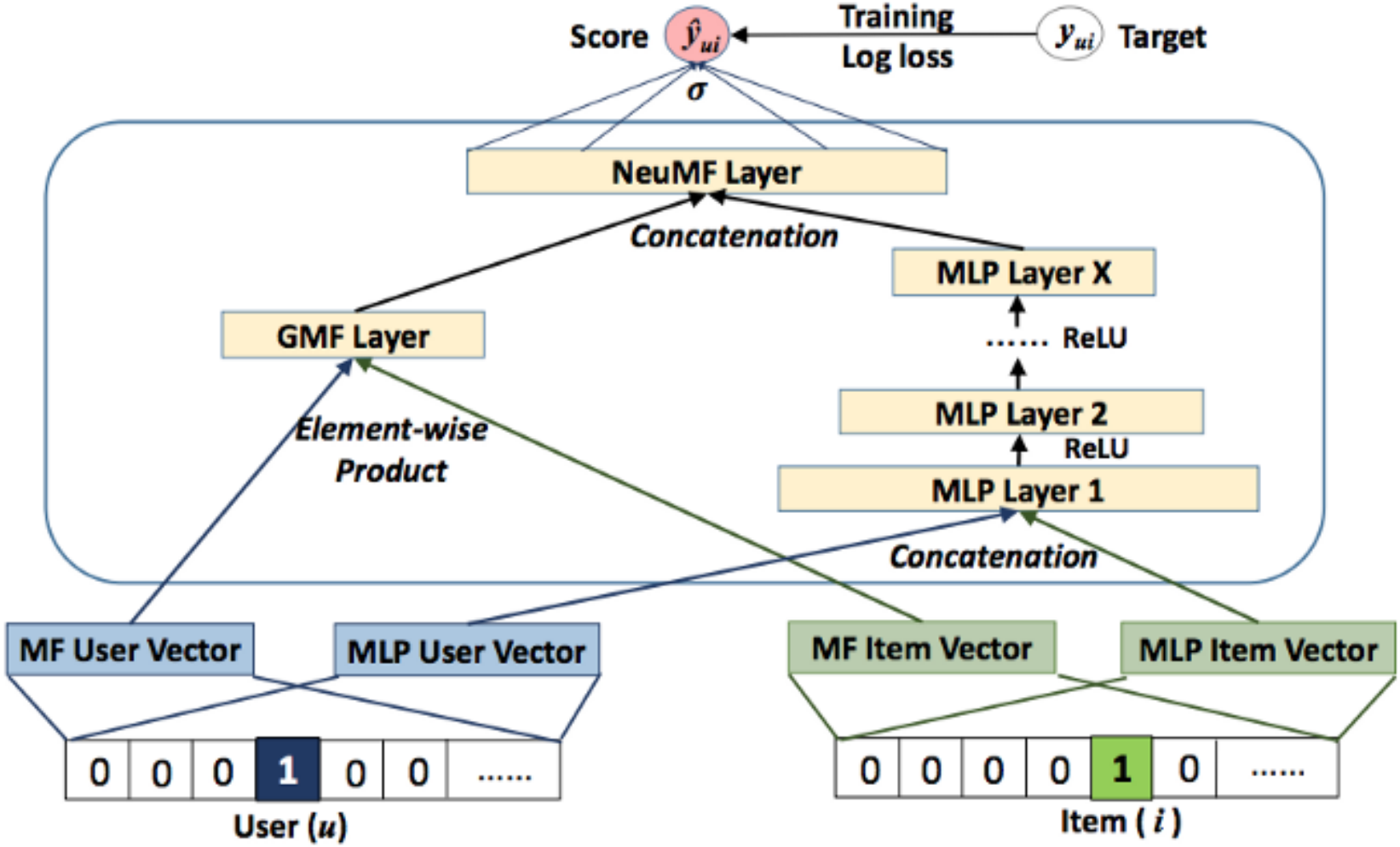
Neural collaborative filtering (NCF)
More detail on the lab

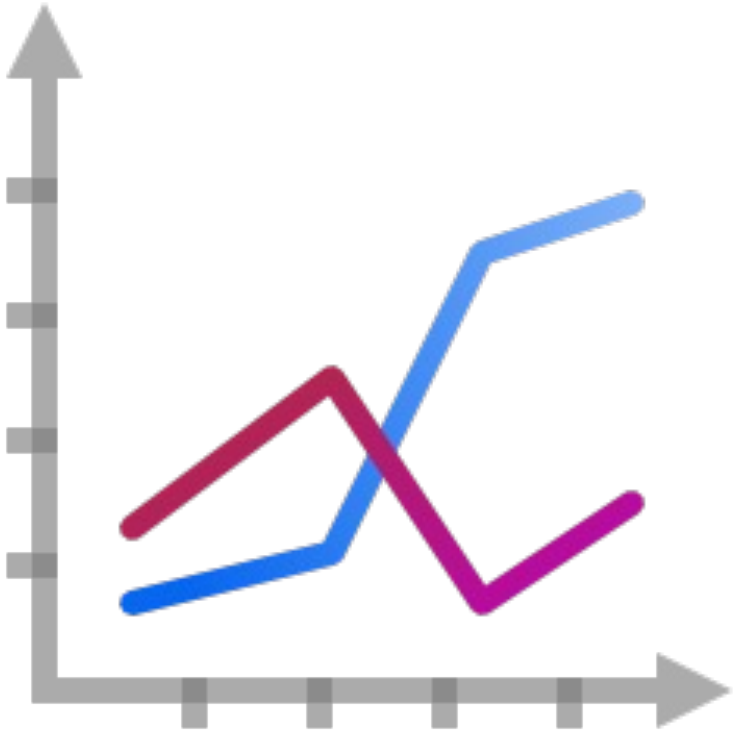
The non-linearity of the MLP allows the estimation of complex interactions between users and items



Neural collaborative filtering (NeuMF)
More detail on the lab

In order to introduce an additional non-linearity, we can use both approaches : one MLP and one GMF (Generalized Matrix Factorization).





Evaluating recommender systems

How to evaluate RS ?

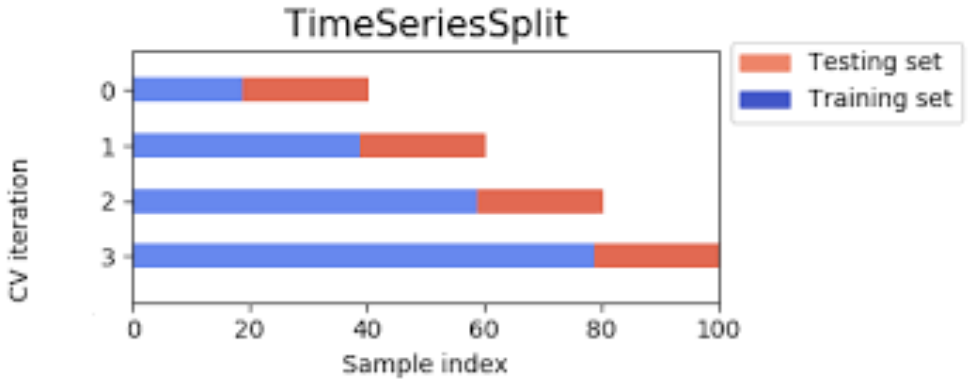
- Test with real users
  - A/B tests
  - Example measures: sales increase, click through rates
- Laboratory studies
  - Controlled experiments
  - Example measures: satisfaction with the system (questionnaires)

- Offline experiments
  - Based on historical data
  - Example measures: prediction accuracy, coverage

Data set splitting

- We are exactly in the case of the times series.
- We can't share the train/test in a random way.

- sklearn.model_selection.TimeSeriesSplit



Traditional accuracy metrics

- Prediction accuracy metrics include the mean absolute error (MAE), root mean square error (RMSE).
  - Use in regression task
  - Focus on comparing the actual vs predicted ratings
  - Easy to calculate and easy to use

  - Measure the difference between ground truth and prediction

Record

- MAE: Mean absolute error
  - sklearn.metrics.mean_absolute_error
- MSE: Mean square error
  - sklearn.metrics.mean_squared_error
- RMSE: Root MSE
  - √MSE

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

New metrics are needed

- When we search for a document, on google for example:
  - Not interested in the exact score of a document
    - Evaluated by MAE, RMSE
  - More interested in its place: first page, second page, second page
    - Evaluated by MAP@K, MAR@K
      - MAP = Mean Average Precision
      - MAR = Mean Average Recall

- $Recommender\ system\ precision = \frac{\#\ \text{of our recommendations that are relevant}}{\#\ \text{of items we recommended}}$

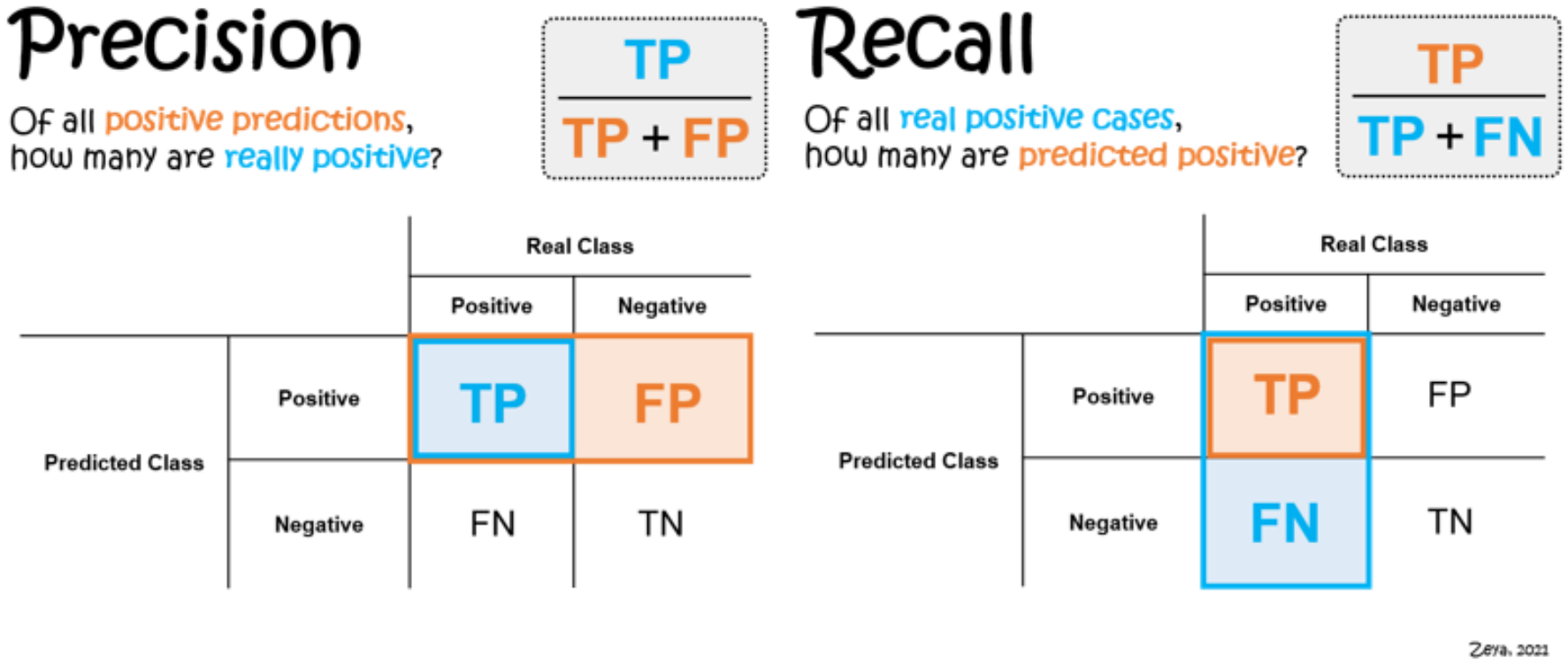
- $Recall\ system\ precision = \frac{\#\ \text{of our recommendations that are relevant}}{\#\ \text{of all the possible relevant items}}$

- Products: 3 positives and 7 negatives for the user
- 5 products recommended but only 2 are positive for the user

- Precision = $2/5$ (2 positive products out of a selection of 5)
- Recall = $2/3$ (2 positive products out of 3 positive)

Precision / Recall

Based on the full list of items / order have no importance



Precision and Recall at Cutoff k

Based on the partial list of items / order have an importance

P(k=3) = 1/3

Algorithm Outputs...

Rank	Product Recommended	Result
1	Credit card	Correct positive
2	Christmas Fund	False positive
3	Debit Card	False positive
4	Auto loan	False positive
5	HELOC	Correct Positive
6	College Fund	Correct positive
7	Personal loan	False positive

Algorithm Outputs...

Rank	Product Recommended	Result
1	Credit card	Correct positive
2	Christmas Fund	False positive
3	Debit Card	False positive
4	Auto loan	False positive
5	HELOC	Correct Positive
6	College Fund	Correct positive
7	Personal loan	False positive

P(k=6) = 3/6

Average Precision at Cutoff k

m *is the total number of relevant items at rank* k

$$AP@k = \frac{1}{m}\sum_{n=1}^{k}(P(k)\ if\ k^{th}\ item\ was\ relevant)$$

$$AP@k = \frac{1}{m}\sum_{n=1}^{k}P(k)\,rel(k)$$

$$AP@6 = \frac{1}{m}\left(P(1) + P(5) + P(6)\right)$$

$$AP@6 = \frac{1}{3}\left(1 + 2/5 + 3/6\right)$$

Algorithm Outputs...

P(k=3) = 1/3

Rank	Product Recommended	Result
1	Credit card	Correct positive
2	Christmas Fund	False positive
3	Debit Card	False positive
4	Auto loan	False positive
5	HELOC	Correct Positive
6	College Fund	Correct positive
7	Personal loan	False positive

Algorithm Outputs...

Rank	Product Recommended	Result
1	Credit card	Correct positive
2	Christmas Fund	False positive
3	Debit Card	False positive
4	Auto loan	False positive
5	HELOC	Correct Positive
6	College Fund	Correct positive
7	Personal loan	False positive

P(k=6) = 3/6

Mean Average Precision at Cutoff k

$$MAP@k = \frac{1}{|U|}\sum_{u=1}^{|U|}(AP@k)_u$$

$$MAP@k = \frac{1}{|U|}\sum_{u=1}^{|U|}\frac{1}{m}\sum_{n=1}^{k}P_u(k)rel_u(k)$$

Algorithm Outputs...

Rank	Product Recommended	Result
1	Credit card	Correct positive
2	Christmas Fund	False positive
3	Debit Card	False positive
4	Auto loan	False positive
5	HELOC	Correct Positive
6	College Fund	Correct positive
7	Personal loan	False positive

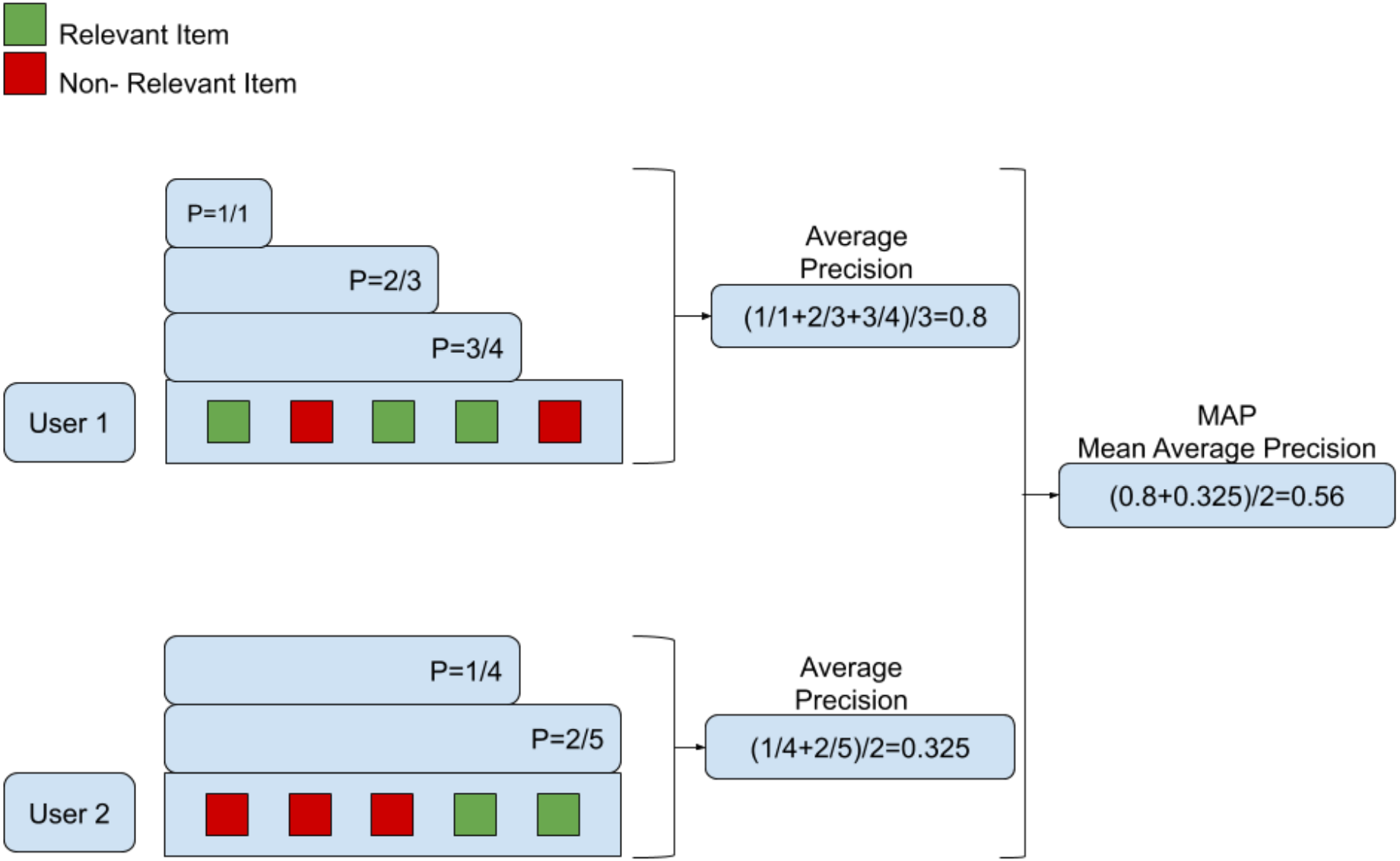
P(k=3) = 1/3

Algorithm Outputs...

Rank	Product Recommended	Result
1	Credit card	Correct positive
2	Christmas Fund	False positive
3	Debit Card	False positive
4	Auto loan	False positive
5	HELOC	Correct Positive
6	College Fund	Correct positive
7	Personal loan	False positive

P(k=6) = 3/6

Mean Average Precision at Cutoff k



MAP Pros and Cons

- Advantages of MAP
  - A value that represents the area under the precision-recall curve
  - Manages the ranking of recommended lists of items
  - Gives more weight to errors that occur at the beginning of lists
  - Suitable if relevant/not relevant

- Disadvantages of MAP
  - Not suitable if rating (e.g. 1 to 5)

Summary

- Very active field
- A large number of machine learning approaches
  - Effective if you can group users or products
  - Effective if part of the calculation is done offline
- Possibility to use deep models
  - Then the question arises as to when to re-train them
- Need for specific metrics
  - For example MAP@k