



# Recommendation Engine





# Introduction -

## **Why Recommender Systems?**

Rise of YouTube, Amazon, Netflix and many other such web services has resulted in recommender systems moving center stage.

## **What is a Recommender System?**

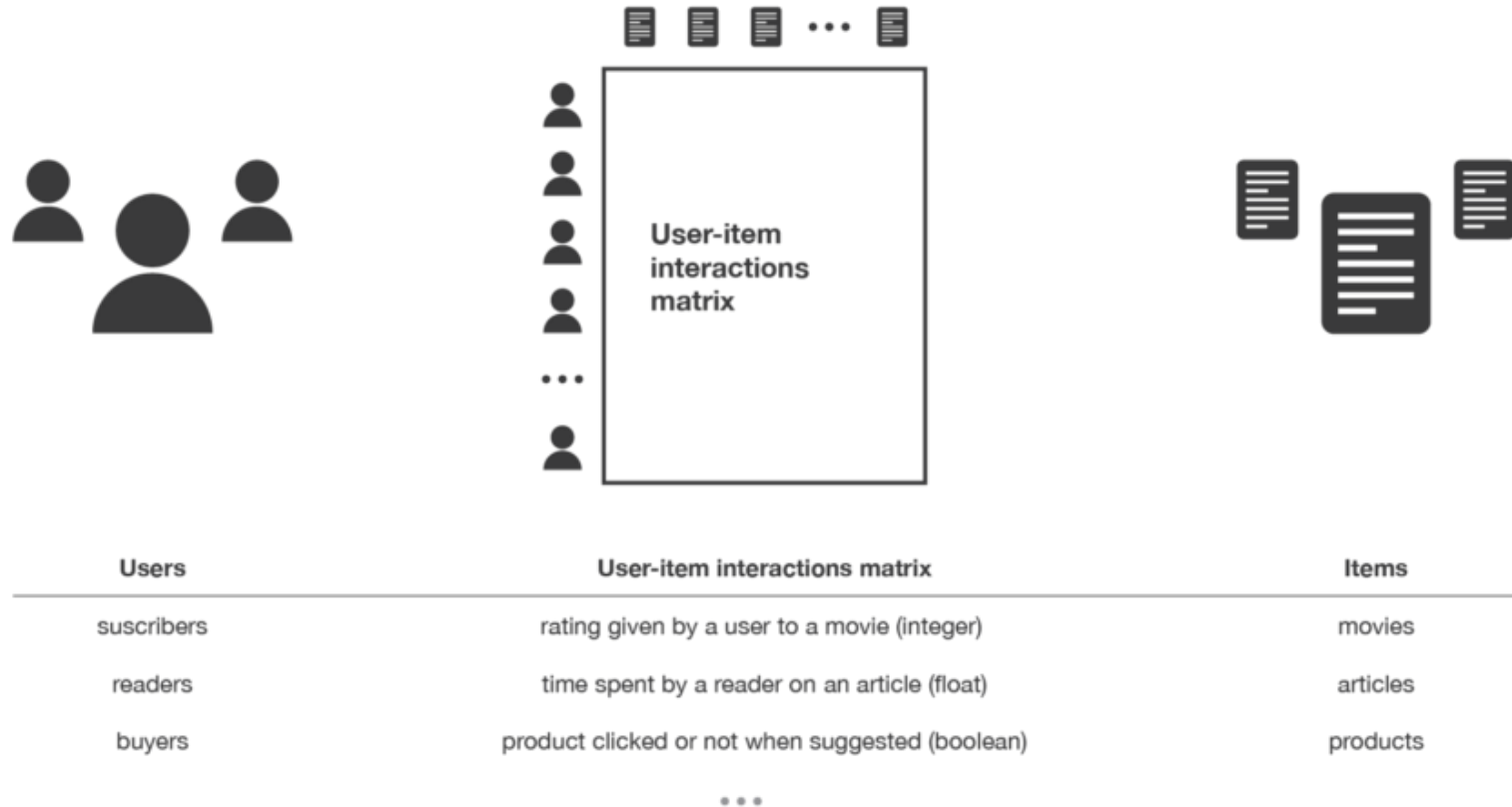
Algorithms aimed at suggesting relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).

## **Two major paradigms of recommender systems :**

- Collaborative Filtering
- Content based methods

# A. Collaborative filtering method -

- Based solely on the past interactions recorded between users and items in order to produce new recommendations.
- Interactions are stored in “user-item interactions matrix”.



*User-item Interactions matrix*

Main idea that rules collaborative methods is that these past user-item interactions are sufficient to detect similar users and/or similar items and make predictions based on these estimated proximities.

## Collaborative Filtering

Memory Based

Model Based

**Memory based approach** - Directly work with values of recorded interactions, assuming no model, and are essentially based on nearest neighbors search (for example, find the closest users from a user of interest and suggest the most popular items among these neighbors).

**Model based approach** - Assume an underlying “generative” model that explains the user-item interactions and try to discover it in order to make new predictions.



### No Model

- users and items are represented directly by their past interactions (large sparse vectors)
- recommendations are done following nearest neighbours information

### Model

- new representations of users and items are build based on a model (small dense vectors)
- recommendations are done following the model information

*Collaborative filtering methods paradigm*




## Benefits -

The main advantage of collaborative approaches is that they require no information about users or items and, so, they can be used in many situations. Moreover, the more users interact with items the more new recommendations become accurate: for a fixed set of users and items, new interactions recorded over time bring new information and make the system more and more effective.

## Drawbacks -

However, as it only consider past interactions to make recommendations, collaborative filtering suffer from the “cold start problem”: it is impossible to recommend anything to new users or to recommend a new item to any users and many users or items have too few interactions to be efficiently handled.

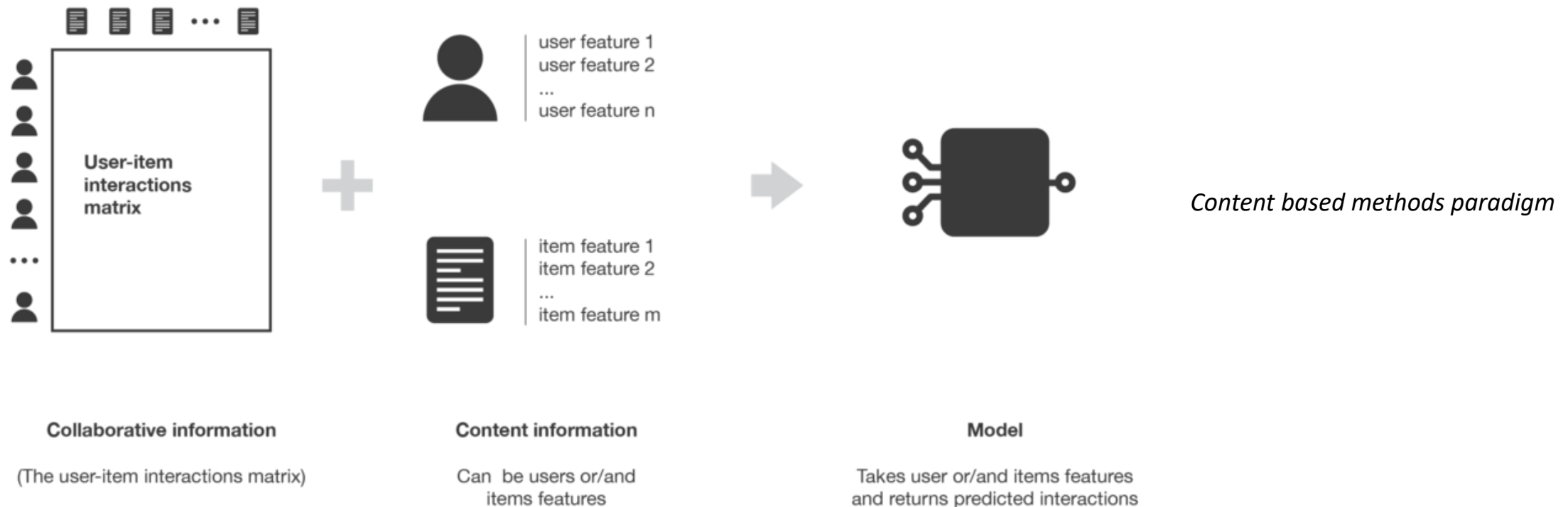
### Ways to overcome ‘cold start problem’–

1. Random Strategy - Recommending random items to new users or new items to random users
  2. Maximum Expectation Strategy - Recommending popular items to new users or new items to most active users
  3. Exploratory Strategy - Recommending a set of various items to new users or a new item to a set of various users
- 

## B. Content based method -

Unlike collaborative methods that only rely on the user-item interactions, content based approaches use additional information about users and/or items. the idea of content based methods is to try to build a model, based on the available “features”, that explain the observed user-item interactions.

For example, In Netflix, to model the fact that young women tend to rate better some movies, that young men tend to rate better some other movies and so on. If we manage to get such model, then, making new predictions for a user is pretty easy: we just need to look at the profile (age, sex, ...) of this user and, based on this information, to determine relevant movies to suggest.



# Comparison Between Models –

## I. Memory based collaborative method

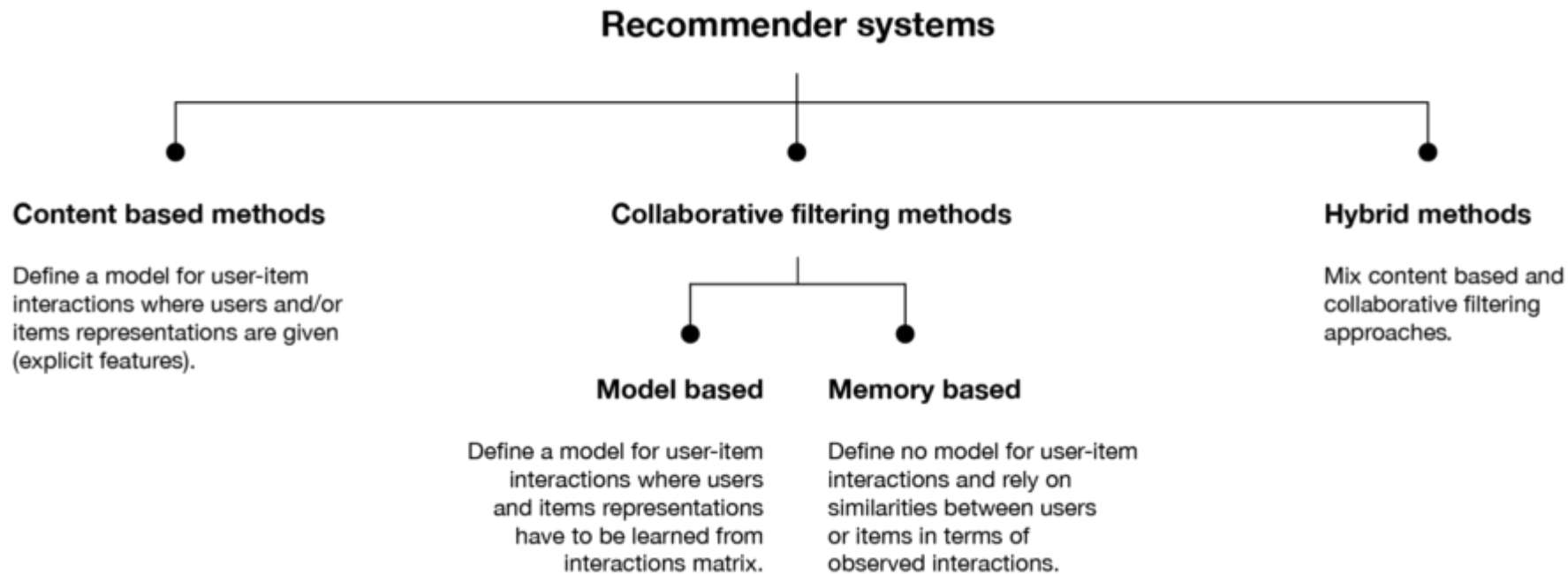
- No latent model is assumed. The algorithms directly works with the user-item interactions.
- Users are represented by their interactions with items and a nearest neighbors search on these representations is used to produce suggestions.
- Low Bias
- High Variance

## II. Model based collaborative method

- Some latent interaction model is assumed. The model is trained to reconstruct user-item interactions values from its own representation of users and items.
- New suggestions can then be done based on this model.
- The users and items latent representations extracted by the model have a mathematical meaning that can be hard to interpret for a human being.
- Higher bias
- Low variance

## III. Content based method

- Some latent interaction model is also assumed. However, here, the model is provided with content that define the representation of users and/or items
- Users are represented by given features and we try to model for each item the kind of user profile that likes or not this item.
- Model is more constrained because representation of users and/or items are given
- Highest Bias
- Lowest Variance



*Summary of different types of recommender systems algorithms*

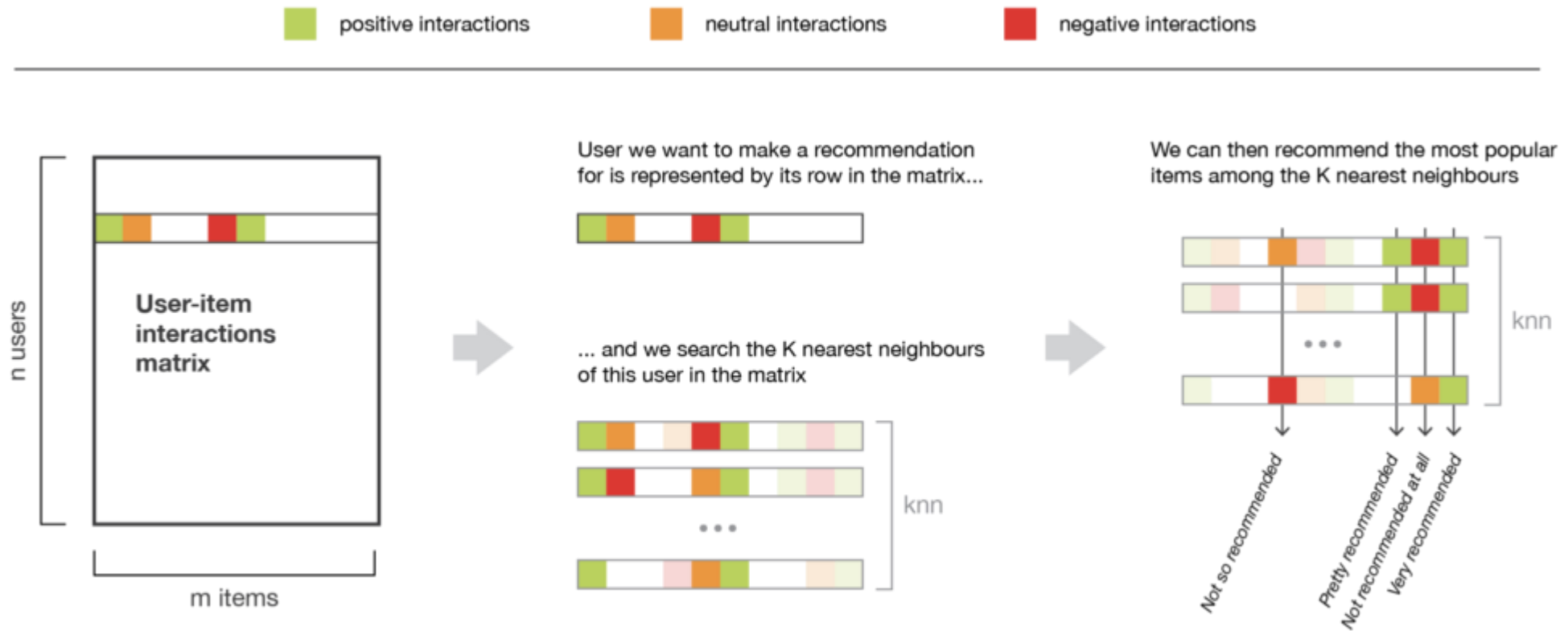
## I. Memory based -

**a) User – User** - In order to make a new recommendation to a user, user-user method roughly tries to identify users with the most similar “interactions profile” (nearest neighbors) in order to suggest items that are the most popular among these neighbors (and that are “new” to our user). This method is said to be “user-centred” as it represent users based on their interactions with items and evaluate distances between users.



## To make a recommendation for a given user.

- Represent every user by its vector of interactions with the different items (“its line” in the interaction matrix).
- Compute “similarity” between our user of interest and every other users. That similarity measure is such that two users with similar interactions on the same items should be considered as being close.
- Keep the k-nearest-neighbors to our user and then suggest the most popular items among them (only looking at the items that our reference user has not interacted with yet).

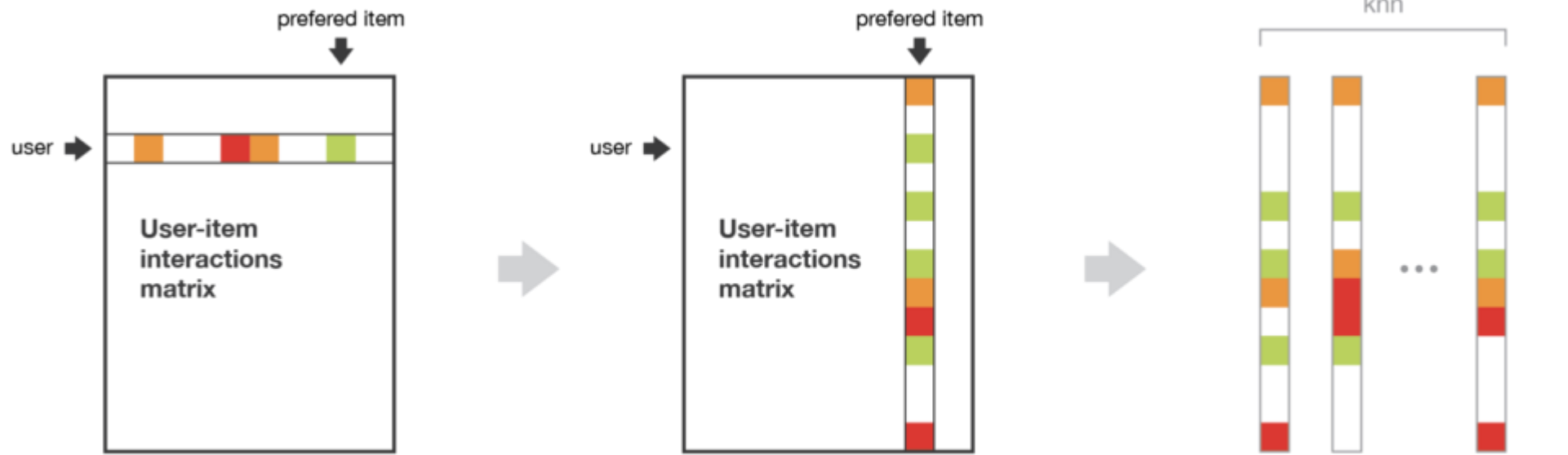


*User-user method*

**b) Item – Item** - Find items similar to the ones the user already “positively” interacted with. Two items are considered to be similar if most of the users that have interacted with both of them did it in a similar way. This method is said to be “item-centred” as it represent items based on interactions users had with them and evaluate distances between those items.

**To make a recommendation for a given user.**

- Consider the item this user liked the most and represent it (as all the other items) by its vector of interaction with every users (“its column” in the interaction matrix).
- Compute similarities between the “best item” and all the other items.
- Keep the k-nearest-neighbours to the selected “best item” that are new to our user of interest and recommend these items.



We identify the preferred item of user we want to make recommendation for.

The preferred item is represented by its column in the matrix.

We can search and recommend the K nearest items to this “preferred item”

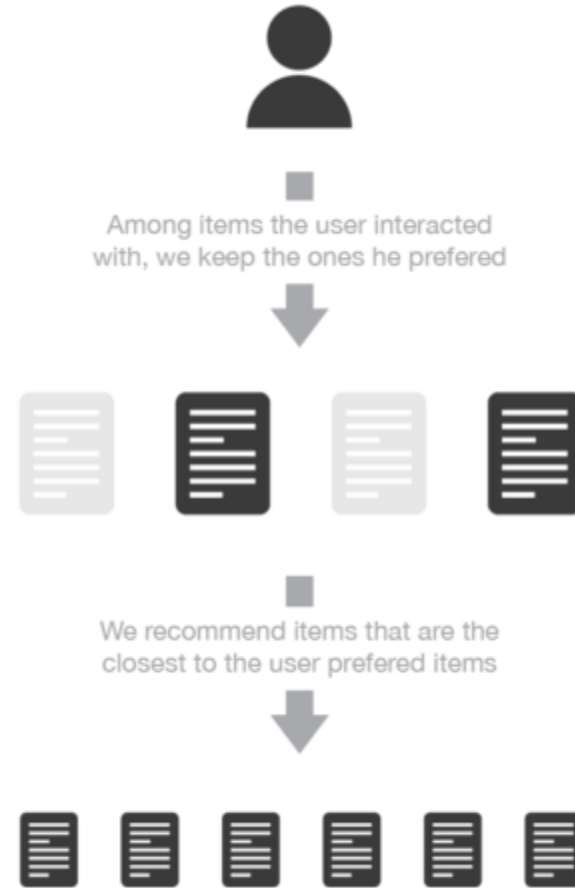
*Item-item method*

# Difference between User –User & Item- Item -

user-user



item-item



## II. Model based -

Model based collaborative approaches only rely on user-item interactions information and assume a latent model to explain these interactions.

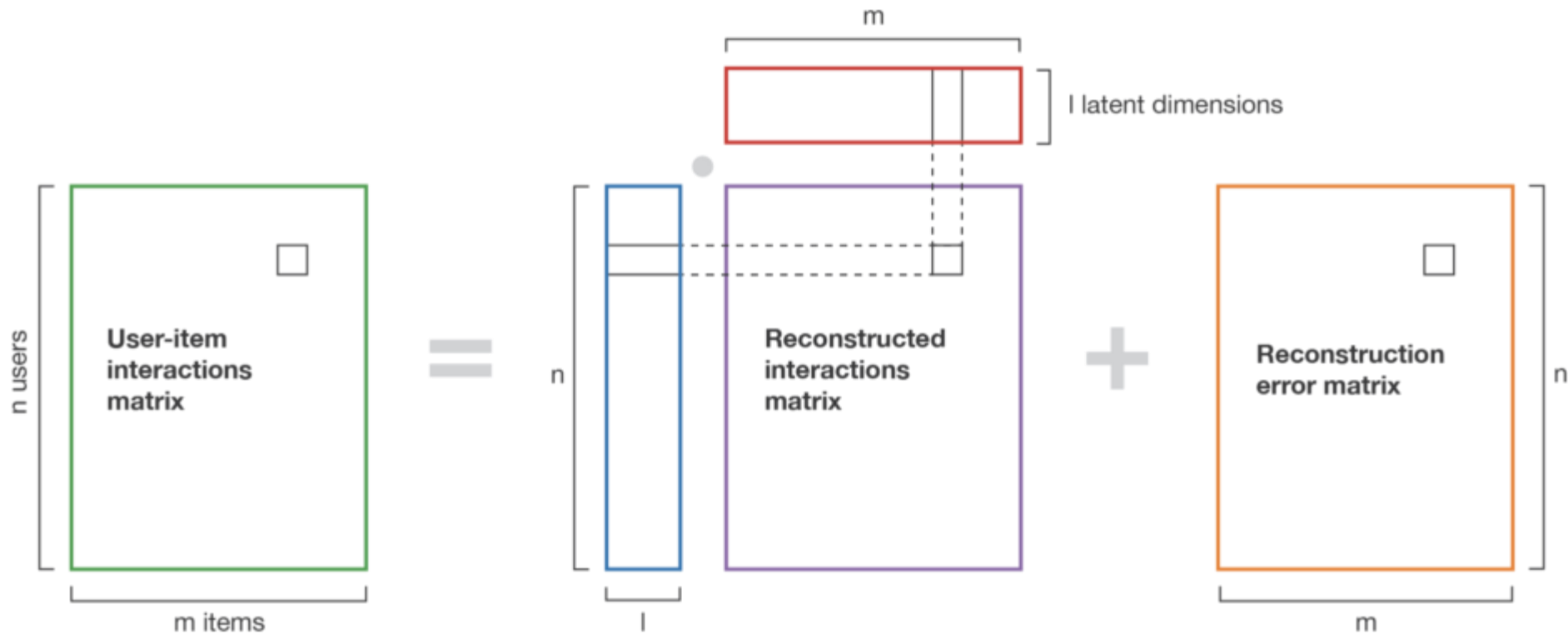
### a) **Matrix Factorisation** -

- There exists a pretty low dimensional latent space of features in which we can represent both users and items
- Interaction between a user and an item can be obtained by computing the dot product of corresponding dense vectors in that space.

For example, consider that we have a user-movie rating matrix. In order to model the interactions between users and movies, we can assume that:

- there exists some features describing (and telling apart) pretty well movies.
- these features can also be used to describe user preferences (high values for features the user likes, low values otherwise)

However we don't want to give explicitly these features to our model (as it could be done for content based approaches that we will describe later). Instead, we prefer to let the system discover these useful features by itself and make its own representations of both users and items



The **user-item interactions matrix** is assumed to be equal to...

... the **dot product** of a **user matrix** and a **transposed item matrix**...

... plus some **reconstruction error**

*Matrix factorisation method*

# Mathematics of Matrix Factorisation -

Let's consider an interaction matrix  $M$  ( $n \times m$ ) of ratings where only some items have been rated by each user (most of the interactions are set to None to express the lack of rating). We want to factorise that matrix such that

$$M \approx X \cdot Y^T \quad \text{where } X \text{ is the "user matrix" } (n \times l) \text{ whose rows represent the } n \text{ users and} \\ \text{where } Y \text{ is the "item matrix" } (m \times l) \text{ whose rows represent the } m \text{ items:}$$

$$\text{user}_i \equiv X_i \quad \forall i \in \{1, \dots, n\}$$

$$\text{item}_j \equiv Y_j \quad \forall j \in \{1, \dots, m\}$$

Here  $l$  is the dimension of the latent space in which users and item will be represented. So, we search for matrices  $X$  and  $Y$  whose dot product best approximate the existing interactions. Denoting  $E$  the ensemble of pairs  $(i, j)$  such that  $M_{ij}$  is set (not None), we want to find  $X$  and  $Y$  that minimise the "rating reconstruction error"

$$(X, Y) = \underset{X, Y}{\operatorname{argmin}} \sum_{(i, j) \in E} [(X_i)(Y_j)^T - M_{ij}]^2$$

Adding a regularisation factor and dividing by 2, we get

$$(X, Y) = \underset{X, Y}{\operatorname{argmin}} \frac{1}{2} \sum_{(i, j) \in E} [(X_i)(Y_j)^T - M_{ij}]^2 + \frac{\lambda}{2} \left( \sum_{i, k} (X_{ik})^2 + \sum_{j, k} (Y_{jk})^2 \right)$$

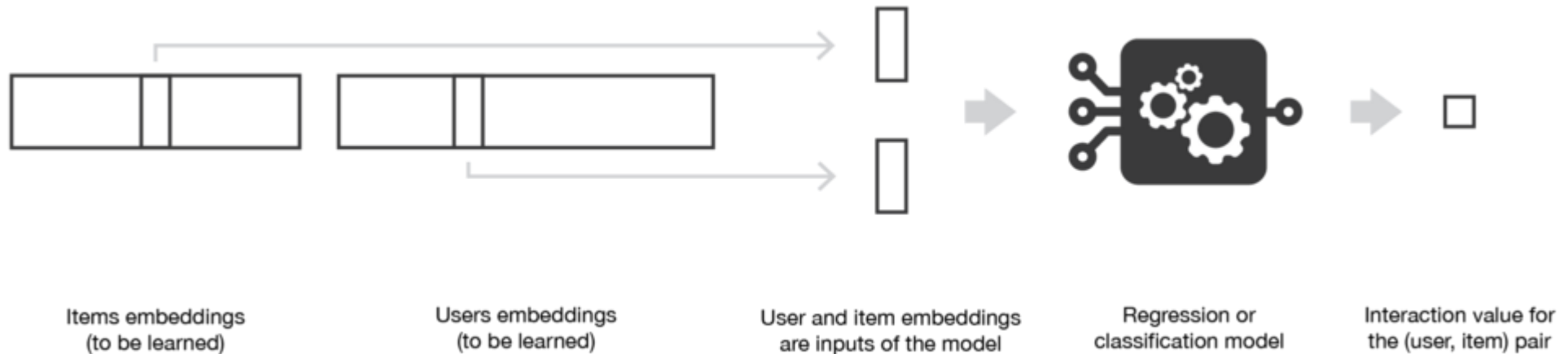
The matrices  $X$  and  $Y$  can then be obtained following a gradient descent optimisation process for which we can notice two things. First, the gradient do not have to be computed over all the pairs in  $E$  at each step and we can consider only a subset of these pairs so that we optimise our objective function "by batch". Second, values in  $X$  and  $Y$  do not have to be updated simultaneously and the gradient descent can be done alternatively on  $X$  and  $Y$  at each step (doing so, we consider one matrix fixed and optimise for the other before doing the opposite at the next iteration). Once the matrix has been factorised, we have less information to manipulate in order to make a new recommendation: we can simply multiply a user vector by any item vector in order to estimate the corresponding rating.

# Extensions of Matrix Factorisation -

If we want to reconstruct boolean interactions, a simple dot product is not well adapted. If, however, we add a logistic function on top of that dot product, we get a model that takes its value in  $[0, 1]$  and, so, better fit the problem. In such case, the model to optimise is

$$(X, Y) = \underset{X, Y}{\operatorname{argmin}} \frac{1}{2} \sum_{(i, j) \in E} [f((X_i)(Y_j)^T) - M_{ij}]^2 + \frac{\lambda}{2} \left( \sum_{i, k} (X_{ik})^2 + \sum_{j, k} (Y_{jk})^2 \right)$$

with  $f(\cdot)$  our logistic function. Deeper neural network models are often used to achieve near state of the art performances in complex recommender systems.



*Matrix factorization can be generalized with the use of a model on top of users and items embeddings*

### III. Content based -

The recommendation problem is cast into either

- A classification problem (predict if a user “likes” or not an item) or
- A regression problem (predict the rating given by a user to an item).

#### **Item Centered Models:**



- Classification (or regression) is based on users features,
- Optimisations and computations can be done “by item”.
- We build and learn one model by item based on users features trying to answer the question “what is the probability for each user to like this item?”
- Robust models but less personalised (more biased) than the user-centred method.

#### **User-centred models:**



- Based on items features
- Optimisations and computations can be done “by user”.
- We can then attach a model to each user that is trained on its data:
- More personalised than its item-centred counterpart as it only takes into account interactions from the considered user.
- However, most of the time a user has interacted with relatively few items and, so, the model we obtain is a far less robust than an item-centred one.



# Difference between Item-Centred and User-Centred Models

 Model for a given user based on items features  
 Items that the concerned user has interacted with (dataset)



 Model for a given item based on user features  
 Users that have interacted with the concerned item (dataset)



**a) Item-centred Bayesian Classifier** - For each item we want to train a Bayesian classifier that takes user features as inputs and output either “like” or “dislike”. So, to achieve the classification task, we want to compute -

$$\frac{\mathbb{P}_{item}(like|user\_features)}{\mathbb{P}_{item}(dislike|user\_features)}$$

the ratio between the probability for a user with given features to like the considered item and its probability to dislike it. This ratio of conditional probabilities that defines our classification rule (with a simple threshold) can be expressed by following the Bayes formula

$$\mathbb{P}_{item}(like|user\_features) = \frac{\mathbb{P}_{item}(user\_features|like) \times \mathbb{P}_{item}(like)}{\mathbb{P}_{item}(user\_features)}$$

$$\mathbb{P}_{item}(dislike|user\_features) = \frac{\mathbb{P}_{item}(user\_features|dislike) \times \mathbb{P}_{item}(dislike)}{\mathbb{P}_{item}(user\_features)}$$

$$\frac{\mathbb{P}_{item}(like|user\_features)}{\mathbb{P}_{item}(dislike|user\_features)} = \frac{\mathbb{P}_{item}(user\_features|like) \times \mathbb{P}_{item}(like)}{\mathbb{P}_{item}(user\_features|dislike) \times \mathbb{P}_{item}(dislike)}$$

Where,

$$\mathbb{P}_{item}(like) \quad \text{and} \quad \mathbb{P}_{item}(dislike)(= 1 - \mathbb{P}_{item}(like))$$

are priors computed from the data whereas

$$\mathbb{P}_{item}(.|like) \quad \text{and} \quad \mathbb{P}_{item}(.|dislike)$$

are likelihoods assumed to follow Gaussian distributions with parameters to be determined also from data.

Various hypothesis about the covariance matrices of these two likelihood distributions (no assumption, equality of matrices, equality of matrices and features independence) leads to various well known models (quadratic discriminant analysis, linear discriminant analysis, naive Bayes classifier).



*Item-centred content based Bayesian classifier*

**a) User-centred Bayesian Classifier** - for a user-centred regression: for each user we want to train a simple linear regression that takes item features as inputs and output the rating for this item. We still denote  $M$  the user-item interaction matrix, we stack into a matrix  $X$  row vectors representing users coefficients to be learned and we stack into a matrix  $Y$  row vectors representing items features that are given. Then, for a given user  $i$ , we learn the coefficients in  $X_i$  by solving the following optimisation problem

$$X_i = \underset{X_i}{\operatorname{argmin}} \frac{1}{2} \sum_{(i,j) \in E} [(X_i)(Y_j)^T - M_{ij}]^2 + \frac{\lambda}{2} \left( \sum_k (X_{ik})^2 \right)$$

where one should keep in mind that  $i$  is fixed and, so, the first summation is only over (user, item) pairs that concern user  $i$ . We can observe that if we solve this problem for all the users at the same time, the optimisation problem is exactly the same as the one we solve in “alternated matrix factorisation” when we keep items fixed.



*User-centred content based regression*




# Evaluation -

As for any machine learning algorithm, we need to be able to evaluate the performances of our recommender systems in order to decide which algorithm fit the best our situation.

## I. Metrics based evaluation -

If our recommender system is based on a model that outputs numeric values such as ratings predictions or matching probabilities, we can assess the quality of these outputs in a very classical manner using an error measurement metric such as, for example, mean square error (MSE). In this case, the model is trained only on a part of the available interactions and is tested on the remaining ones.

Still if our recommender system is based on a model that predicts numeric values, we can also binarize these values with a classical thresholding approach and evaluate the model in a more “classification way”.



Finally, if we now consider a recommender system not based on numeric values and that only returns a list of recommendations (such as user-user or item-item that are based on a knn approach), we can still define a precision like metric by estimating the proportion of recommended items that really suit our user. To estimate this precision, we can not take into account recommended items that our user has not interacted with and we should only consider items from the test dataset for which we have a user feedback.


## II. Human based evaluation -

We absolutely want to avoid having a user being stuck in what we called earlier an information confinement area. The notion of “serendipity” is often used to express the tendency a model has or not to create such a confinement area (diversity of recommendations).

Serendipity, that can be estimated by computing the distance between recommended items, should not be too low as it would create confinement areas, but should also not be too high as it would mean that we do not take enough into account our users interests when making recommendations (exploration vs exploitation).


Thus, in order to bring diversity in the suggested choices, we want to recommend items that both suit our user very well and that are not too similar from each others. For example, instead of recommending a user “Star Wars” 1, 2 and 3, it seems better to recommend “Star wars 1”, “Star trek into darkness” and “Indiana Jones and the raiders of the lost ark”: the two later may be seen by our system as having less chance to interest our user but recommending 3 items that look too similar is not a good option.

Explainability is another key point of the success of recommendation algorithms. Indeed, it has been proven that if users do not understand why they had been recommended as specific item, they tend to loose confidence into the recommender system. So, if we design a model that is clearly explainable, we can add, when making recommendations, a little sentence stating why an item has been recommended (“people who liked this item also liked this one”, “you liked this item, you may be interested by this one”, ...).



Finally, on top of the fact that diversity and explainability can be intrinsically difficult to evaluate, we can notice that it is also pretty difficult to assess the quality of a recommendation that do not belong to the testing dataset.

As the goal of the recommender system is to generate an action (watch a movie, buy a product, read an article etc...), we can indeed evaluate its ability to generate the expected action. For example, the system can be put in production, following an A/B testing approach, or can be tested only on a sample of users. Such processes require, however, having a certain level of confidence in the model.



## Key Takeaways -

- ✓ Recommendation algorithms can be divided in two great paradigms: collaborative approaches (such as user-user, item-item and matrix factorisation) that are only based on user-item interaction matrix and content based approaches (such as regression or classification models) that use prior information about users and/or items
- ✓ Memory based collaborative methods do not assume any latent model and have then low bias but high variance ; model based collaborative approaches assume a latent interactions model that needs to learn both users and items representations from scratch and have, so, a higher bias but a lower variance ; content based methods assume a latent model build around users and/or items features explicitly given and have, thus, the highest bias and lowest variance
- ✓ Recommender systems are more and more important in many big industries and some scales considerations have to be taken into account when designing the system (better use of sparsity, iterative methods for factorisation or optimisation, approximate techniques for nearest neighbours search...)
- ✓ Recommender systems are difficult to evaluate: if some classical metrics such that MSE, accuracy, recall or precision can be used, one should keep in mind that some desired properties such as diversity (serendipity) and explainability can't be assessed this way ; real conditions evaluation (like A/B testing or sample testing) is finally the only real way to evaluate a new recommender system but requires a certain confidence in the model





Source -

<https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>

THANK YOU



Corporate: [info@sonata-software.com](mailto:info@sonata-software.com) | [www.sonata-software.com](http://www.sonata-software.com)

