Recommendation Systems

Dr. Edgar Acuna

University of Puerto Rico at Mayaguez

Department of Mathematical Sciences

Recommender Systems

- Systems for recommending items (e.g. books, movies, CD's, web pages, newsgroup messages) to users based on examples of their preferences.

- Many websites provide recommendations (e.g. Amazon, NetFlix, Pandora).

- Recommenders have been shown to substantially increase sales at on-line stores.

Recommender Systems

- Movie/TV recommendation (Netflix, Hulu, iTunes)
- Product recommendation (Amazon, EBay)
- Social recommendation (Facebook, Instagram, Twitter, LinkedIn)
- News content recommendation (Yahoo, Google News)
- Priority inbox & spam filtering (Google)

The value of recommendations

Netflix: According to McKinsey (2013), up to 75% of what consumers watch on Netflix comes from the company's recommender system. Netflix stated that their recommender system saves them about $1 billion each year.

Google News: recommendations generate 38% more click-throughs

Amazon: credits recommender systems with 35% of their revenue,

Pandora: Its recommendation system has allowed the company to grow to 78 million users that listen for an average of 24 hours per month.

Best Buy: In 2016, they reported a 23.7% increase on sales, thanks in part to their recommender system.

The "Recommender problem"

Goal: Estimate a utility function to predict how a user will like an item.

C:= {all users}
 S:= {list of all recommendable items}
u:= utility function, measures the usefulness of item s to user c,
    $u : C \times S \rightarrow R$
where R:= {rating recommended items}. S does not contain items that already user c has expressed interest.

In recommender systems, the utility of an item is usually represented by a rating, which indicates how a particular user liked a particular item, e.g., John Doe gave the movie "Harry Potter" the rating of 7 (out of 10). However, in general, utility can be an arbitrary function, including a profit function.

Properties of a good recommendation System

It must be relevant to the user: personalized

It must be diverse: representing all the possible interests of one user

It should not recommend items the user already knows or would have found anyway.

It should expand the user's taste into neighboring areas.

Personalization

- Recommenders are instances of personalization software.
- Personalization concerns adapting to the individual needs, interests, and preferences of each user.
- Includes:
  - Recommending
  - Filtering
  - Predicting
- From a business perspective, it is viewed as part of Customer Relationship Management (CRM).

Machine Learning and Personalization

- Machine Learning can allow learning a *user model* or *profile* of a particular user based on:
  - Sample interaction
  - Rated examples
- This model or profile can then be used to:
  - Recommend items
  - Filter information
  - Predict behavior

Types of Recommender Systems

- Popularity based recommendation system
- Collaborative Filtering (a.k.a. social filtering, Golberg, et al. 1992, Resnick et al. 1994, Hill et al. 1995)
- Content-based recommendation system
- Demographic-based recommendation system
- Utility-based recommendation system
- Knowledge-based recommendation system
- Hybrid-recommendation systems

1-Recommendation System based on item popularity

As the name suggests Popularity based recommendation system works with the trend. It basically uses the items which are in trend right now. For example, if any product which is usually bought by every new user then there are chances that it may suggest that item to the user who just signed up.
The problem with popularity-based recommendation system is that the personalization is not available with this method i.e., even though you know the behavior of the user you cannot recommend items accordingly.

2-Collaborative Filtering

Predicting (filtering) user preferences on new items is done by collecting taste information from many users (collaborative).
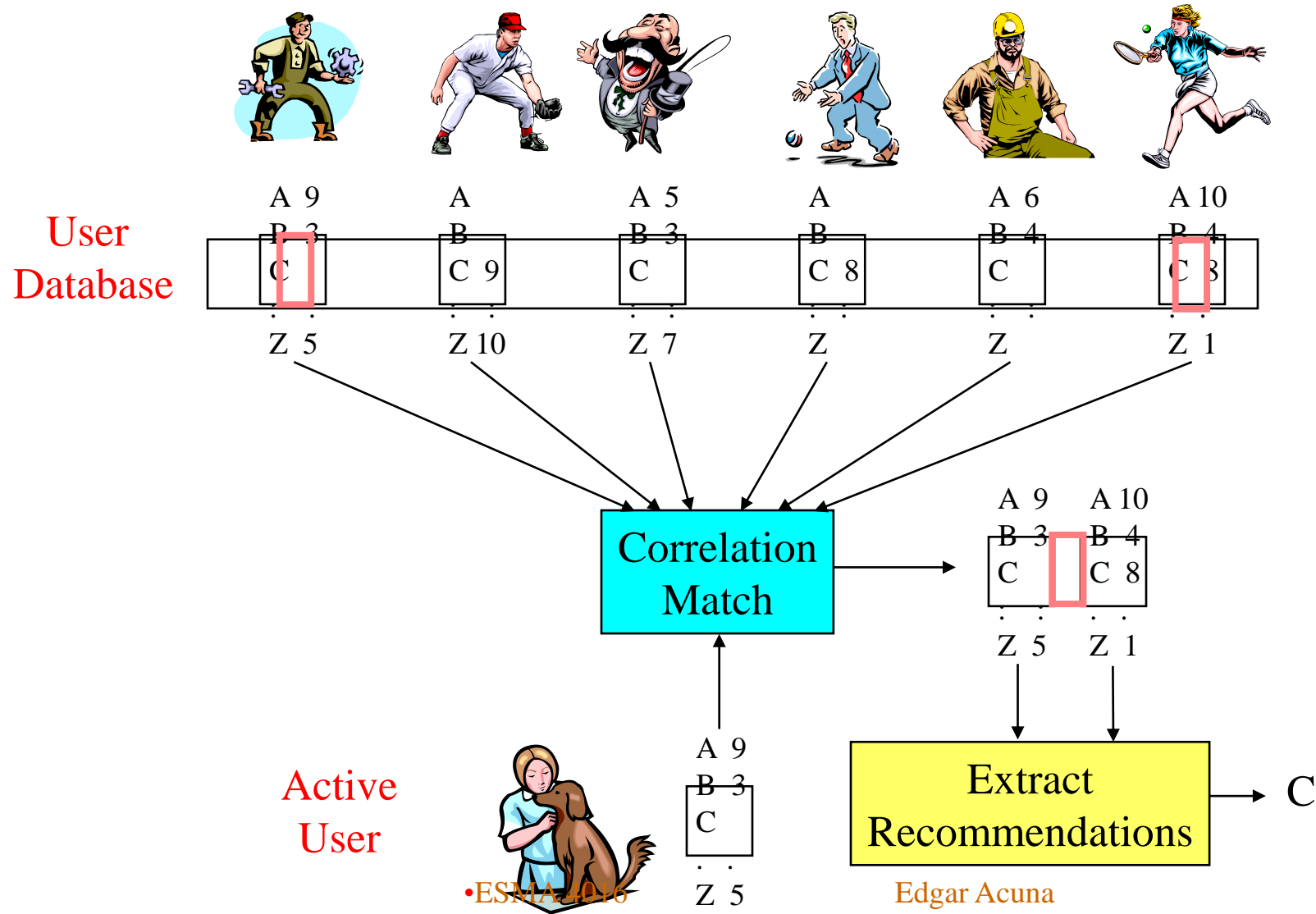
Challenges:
- Many items to choose from very few recommendations to propose
- Few data per user
- No data for new user
- Very large datasets

Collaborative Filtering

- Maintain a database of many users' ratings of a variety of items.

- For a given user, find other similar users whose ratings strongly correlate with the current user.

- Recommend items rated highly by these similar users, but not rated by the current user.

- Almost all existing commercial recommenders use this approach (e.g. Amazon).

Collaborative Filtering

User
Database

A 9
B 3
C
Z 5

A
B
C 9
Z 10

A 5
B 3
C
Z 7

A
B
C 8
Z

A 6
B 4
C
Z

A 10
B 4
C 3
Z 1

Correlation
Match

A 9
B 3
C
Z 5

A 10
B 4
C 8
Z 1

Active
User

A 9
B 3
C
Z 5

Extract
Recommendations

C

Edgar Acuna

13dr hc-co=-

Collaborative Filtering Method

- Weight all users with respect to similarity with the active user.

- Select a subset of the users (*neighbors*) to use as predictors.

- Normalize ratings and compute a prediction from a weighted combination of the selected neighbors' ratings.

- Present items with highest predicted ratings as recommendations.

Neighbor Selection

- For a given active user, a, select correlated users to serve as source of predictions.

- Standard approach is to use the most similar n users, u, based on similarity weights, $w_{a,u}$

- Alternate approach is to include all users whose similarity weight is above a given threshold.

Rating Prediction

- Predict a rating, $p_{a,i}$, for each item i, for active user, a, by using the n selected neighbor users, $u \in \{1,2,\ldots n\}$.

- To account for users different ratings levels, base predictions on *differences* from a user's *average* rating.

- Weight users' ratings contribution by their similarity to the active user.

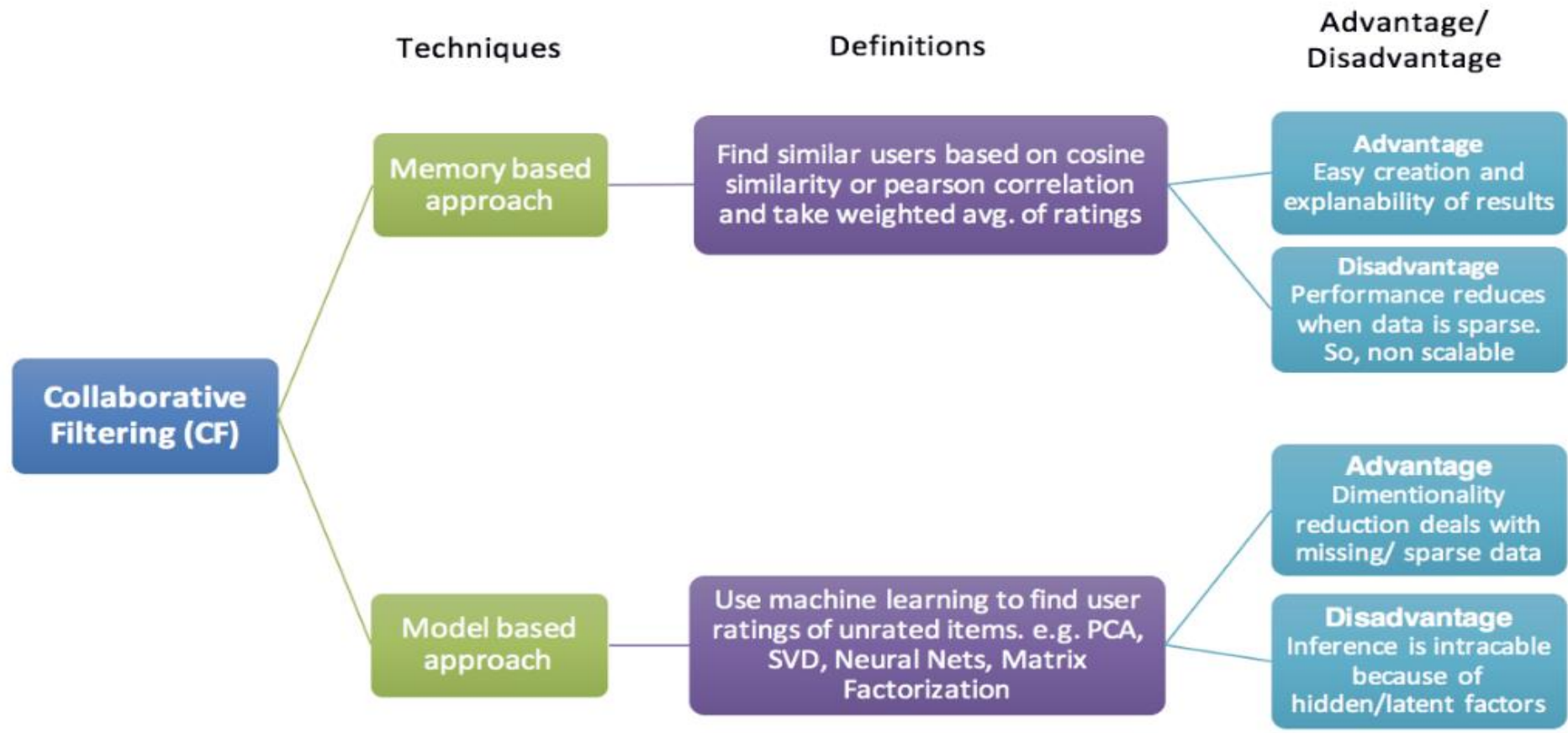$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} w_{a,u}}$$

Example: Collaborative Filtering

2			4	5	
5		4			1
		5		2	
	1		5		4
		4		2	
4	5		1		

Each user has expressed an opinion for some items:

- Explicit opinion: rating score

- Implicit: purchase records or listen to tracks

Collaborative Filtering

Techniques | **Definitions** | **Advantage/ Disadvantage**

Collaborative Filtering (CF)

Memory based approach — Find similar users based on cosine similarity or pearson correlation and take weighted avg. of ratings

Advantage Easy creation and explanability of results

Disadvantage Performance reduces when data is sparse. So, non scalable

Model based approach — Use machine learning to find user ratings of unrated items. e.g. PCA, SVD, Neural Nets, Matrix Factorization

Advantage Dimentionality reduction deals with missing/ sparse data

Disadvantage Inference is intracable because of hidden/latent factors

Memory-based Collaborative Filtering

User-based Filtering: These systems recommend products to a user that similar users have liked. For example, let's say Alice and Bob have a similar interest in books (that is, they largely like and dislike the same books). Now, let's say a new book has been launched into the market and Alice has read and loved it. It is therefore, highly likely that Bob will like it too and therefore, the system recommends this book to Bob. User-based Collaborative Filtering: "Users who are similar to you also liked …"

Item-based Filtering: These systems identify similar items based on how people have rated it in the past. For example, if Alice, Bob and Eve have given 5 stars to *The Lord of the Rings* and *The Hobbit*, the system identifies the items as similar. Therefore, if someone buys *The Lord of the Rings*, the system also recommends *The Hobbit* to him or her.
Item-based Collaborative Filtering: "Users who liked this item also liked …"

Example: User-based CF

		MICKEY BLUE EYES			
2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Target (or Active) user for whom the CF recommendation task is performed

Example: User-based CF

2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

1. Identify set of items rated by the target user

Example: User-based CF

2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

1. Identify set of items rated by the target user

2. Identify which other users rated 1+ items in this set (neighborhood formation)

User-based similarity
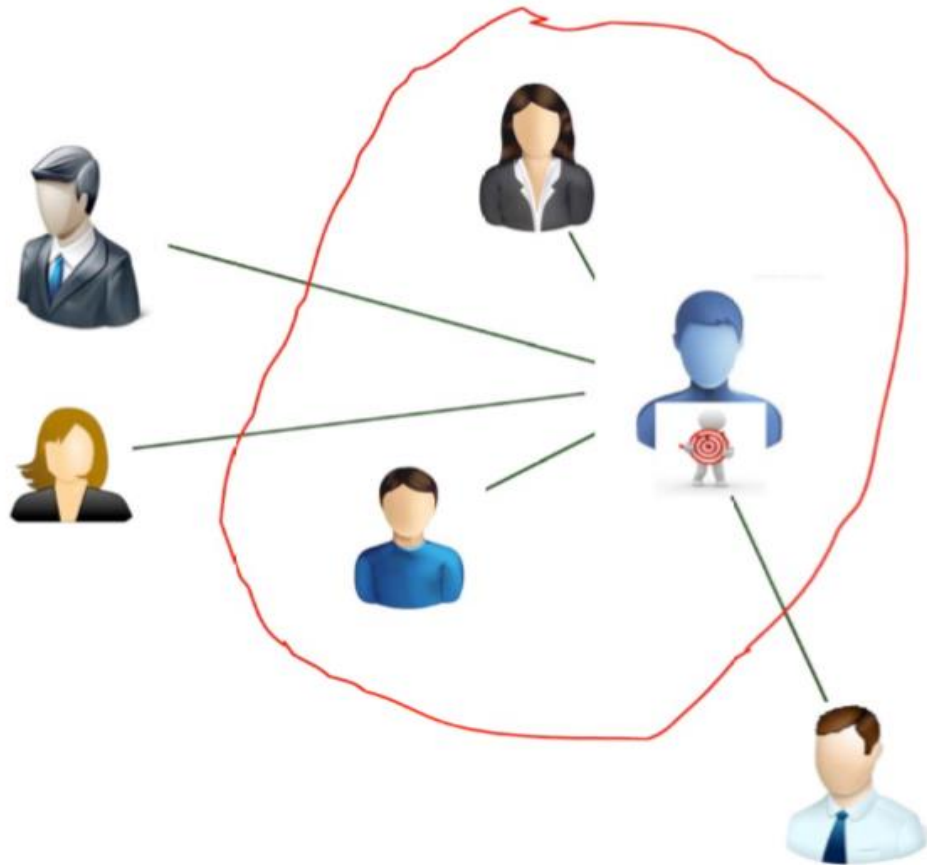
3. Compute how similar each neighbor is to the target user (similarity function)

4. In case, select k most similar neighbors

User-based CF

5. Predict ratings for the target user's unrated items (prediction function)

6. Recommend to the target user the top N products based on the predicted ratings

User-based CF

- Let $r_{u,i}$ be the rating of the ith item under user u, $\bar{r_u}$ be the average rating of user u, and $com(u,v)$ be the set of items rated by both user u and user v

- The similarity between user u and user v is then given by Pearson's correlation coefficient

$$sim(u,v) = \frac{\sum_{i \in com(u,v)}(r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in com(u,v)}(r_{u,i} - \bar{r_u})^2}\sqrt{\sum_{i \in com(u,v)}(r_{v,i} - \bar{r_v})^2}}$$

Example: User-based CF

	Saving Silverman	Date Night	Mickey Blue Eyes	(rainbow)	Star Trek II: Wrath of Khan	Star Trek	sim(u,v)
	2			4	5		NA
	5		4			1	
			5		2		
		1		5		4	
			4			2	
	4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

$0.87 = (4-3)*(4-3.33)+(2-3)(1-3.33)/\text{sqrt}(5.87*2)$

Example: User-based CF

	Saving Silverman	Date Night	Mickey Blue Eyes		Star Trek II: The Wrath of Khan		sim(u,v)
👨	2			4	5		NA
👩	5		4			1	0.87
👨			(5)		2		1
👦		1		5		4	
👦			(4)			2	
👩	4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
		4			2	
4	5		1			NA

Example: User-based CF

$p_{u,i}$, the predicted rating for the ith item of user u, is given by

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in nn(u)} sim(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in nn(u)} |sim(u,v)|}$$

– This is the average rating of user u plus the weighted average of the ratings of u's k nearest neighbors

Example: User-based CF

						sim(∪,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
3.51*	3.81*	4	2.42*	2.48*	2	
4	5		1			NA

$$3.51 = .87*(5-3.33)/(.87+1+1)$$

Example: User-based CF

Cosine metric for similarity

$$\cos(u_i, u_j) = \frac{\displaystyle\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\displaystyle\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

The Sparsity Problem

Typically large product sets & few user ratings
e.g. Amazon:

- in a catalogue of 1 million books, the probability that two users who bought 100 books each, have a book in common is 0.01

- in a catalogue of 10 million books, the probability that two users who bought 50 books each, have a book in common is 0.0002

- CF must have a number of users ~ 10% of the product catalogue size

Model-based Collaborative Filtering
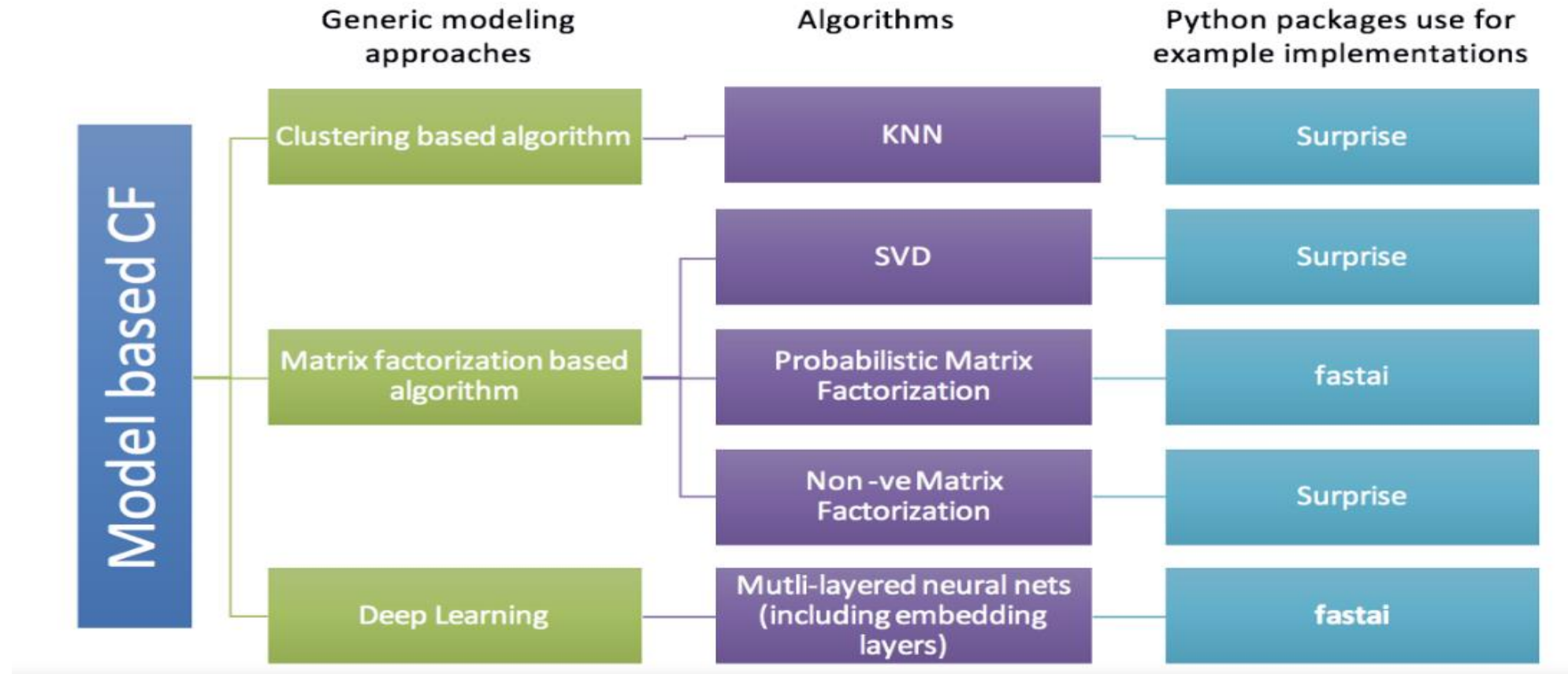
Model-based Collaborative Filtering

In this approach, models are developed using different data mining, machine learning algorithms to predict users' rating of unrated items.

There are many model-based CF algorithms. Bayesian networks, clustering models, singular value decomposition, probabilistic latent semantic analysis, multiple multiplicative factor, and Markov decision process based models.

Through this approach, dimensionality reduction methods are mostly being used as complementary technique to improve robustness and accuracy of memory-based approach. In this sense, methods like singular value decomposition, principal component analysis, known as latent factor models, compress user-item matrix into a low-dimensional representation in terms of latent factors.

 One advantage of using this approach is that instead of having a high dimensional matrix containing abundant number of missing values we will be dealing with a much smaller matrix in lower-dimensional space.

This approach handles the sparsity of the original matrix better than memory based ones.

Recommendation systems using clustering

- **Cluster** customers into categories based on preferences & past purchases

- **Compute** recommendations at the cluster level:

all customers within a cluster receive the same recommendations

Recommendation systems using clustering

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

Customers B,C and D form one cluster and customers A and E another cluster
Typical preferences for members of the cluster are:
Book 2 is highly recommended,
Book 3 is recommended
Books 5 and 6 may be recommended.
Books 1 and 4 are not recommended.

Recommendation systems using clustering

+ It can also be applied for selecting the k most relevant neighbours in a CF algorithm

+ Faster: recommendations are per cluster

- less personalized: recommendations are per cluster vs. in CF they are per user

Recommendation systems using association rules

Past purchases used to find relationships of common purchases

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	
CUSTOMER F			X		X	

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
BOOK 1				1	1	
BOOK 2			2		1	1
BOOK 3		2			2	
BOOK 4	1					
BOOK 5	1	1	2			
BOOK 6		1				

Recommendation systems using association rules

+ Fast to implement

+ Fast to execute

+ Not much storage space required

+ Not « individual » specific

+ Very successful in broad applications for large populations, such as shelf layout in retail stores

- Not suitable if preferences change rapidly

- Rules can be used only when enough data validates them. False associations can arise

Recommendation Systems using Matrix Factorization

- There could be a number of latent factor (variables that are not directly observed) that affect the recommendation:

  Style of movie: Drama, Comedy, Fiction,
  Demographic: It is preferred more for men or women

- Alternative approach: View CF as a matrix factorization problem

Recommendation Systems using Matrix Factorization

$$f_{ij} = \langle U_{i*}, M_{*,j} \rangle$$

$$E(U, M) = L(y_{i,j}, f_{ij}) + \Omega(U, M)$$
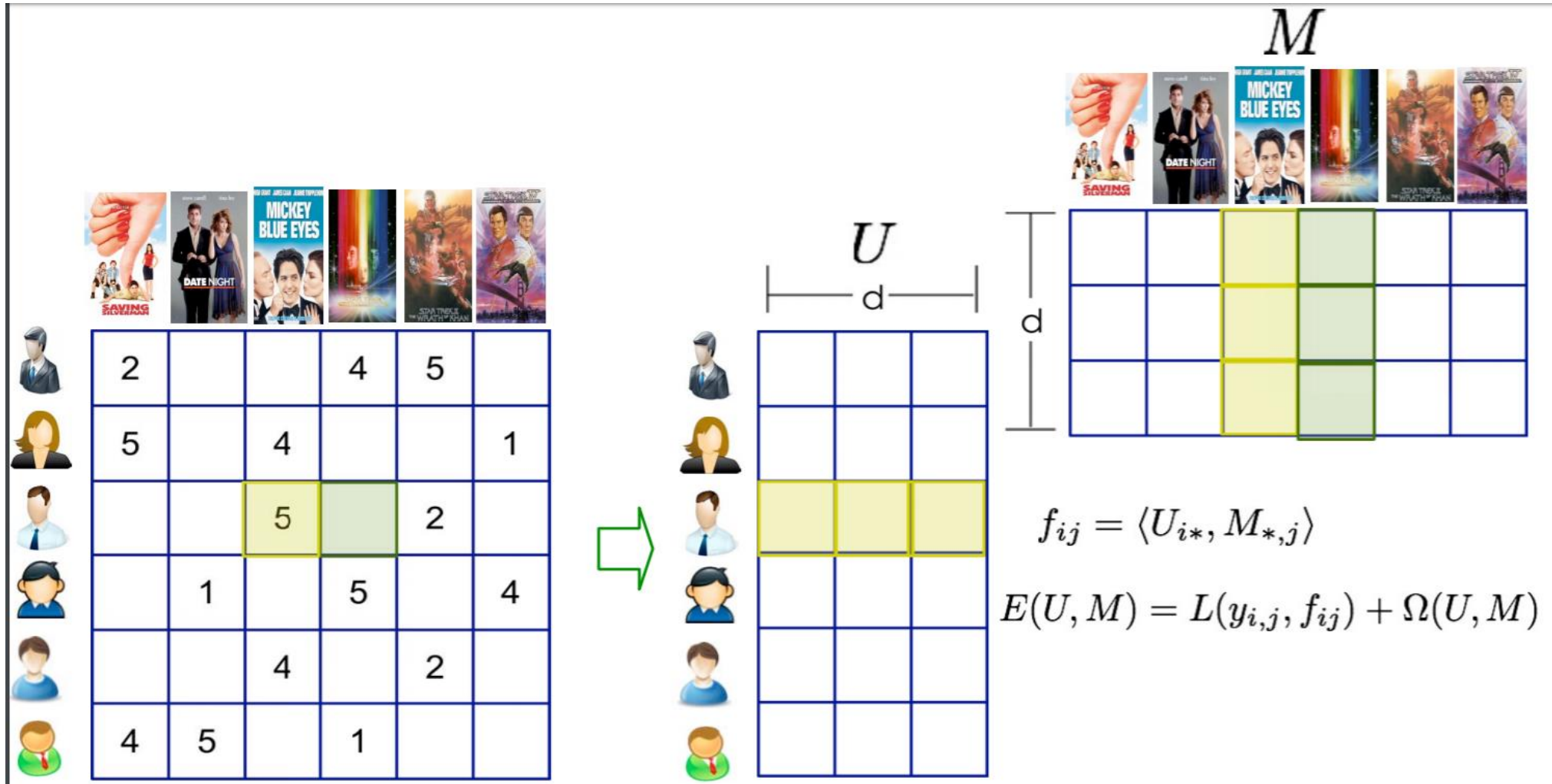
Matrix Factorization

- Express a matrix $M \in \mathbb{R}^{m \times n}$ approximately as a product of factors $A \in \mathbb{R}^{m \times p}$ and $B \in \mathbb{R}^{p \times n}$

$$M \sim A \cdot B$$

- Approximate the user $\times$ items matrix as a product of matrices in this way

  - Similar to SVD decompositions that we saw earlier (SVD can't be used for a matrix with missing entries)

  - Think of the entries of M as corresponding to an inner product of latent factors

Matrix Factorization

users

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

items

~

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

~

●

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1
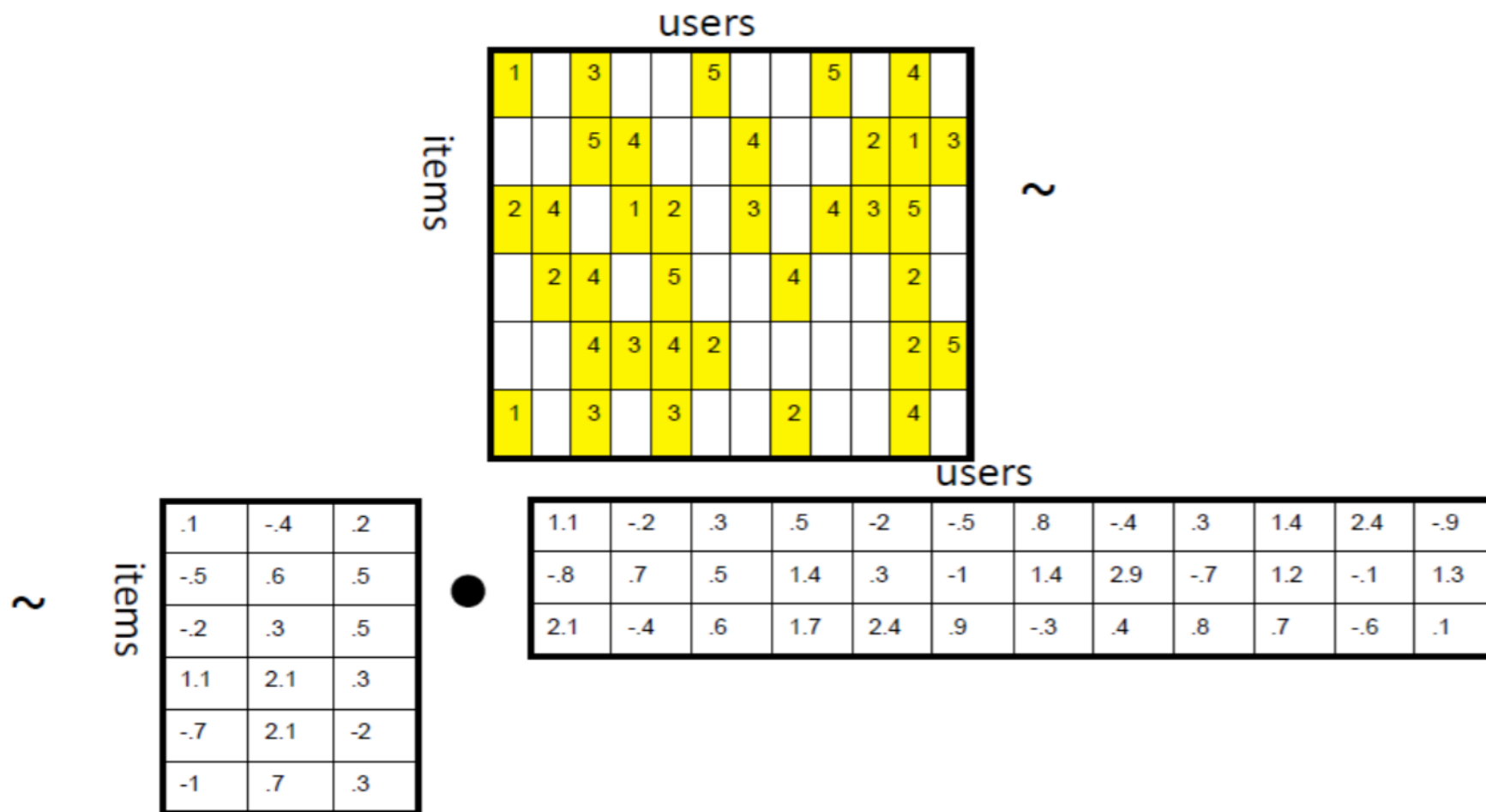
Matrix Factorization
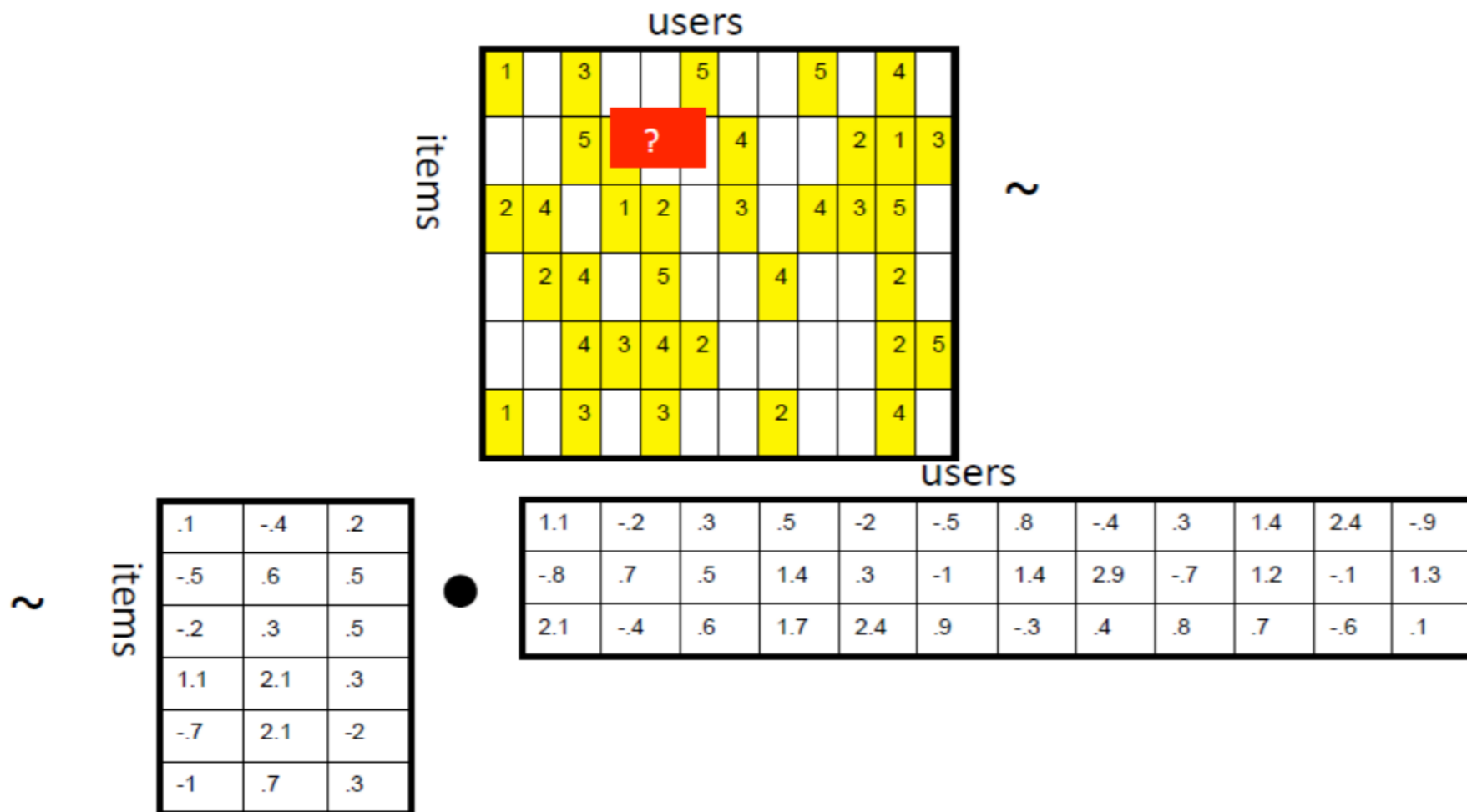
Matrix Factorization
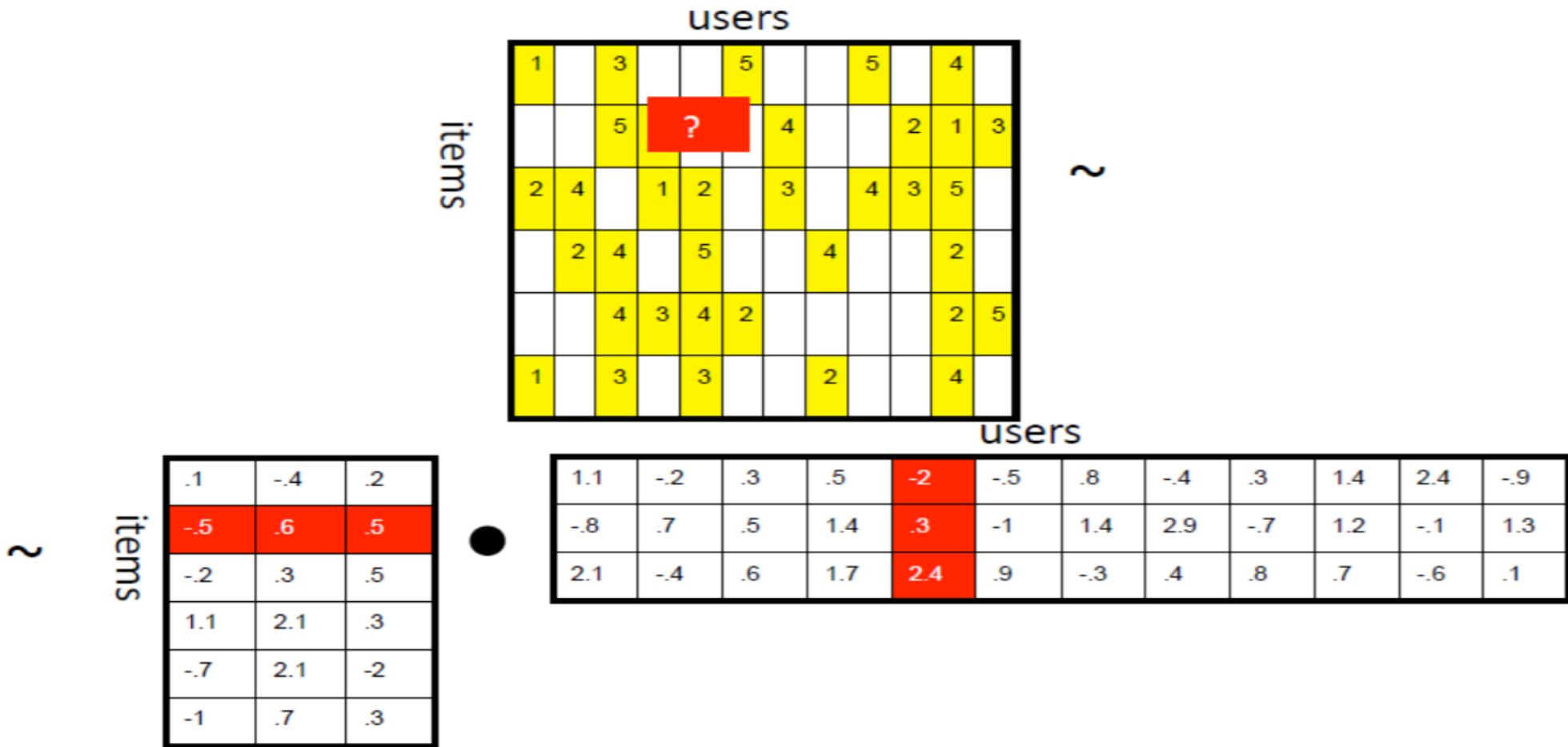
Matrix Factorization

The concept of matrix factorization can be written mathematically to look something like below.

$$\hat{r}_{ui} = q_i^T p_u.$$

Then we can create an objective function (that we want to minimize) with respect to q and p, which are (m,k) and (k,n) matrices.

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

Matrix Factorization

The last term of the equation is called the regularization.
The method became very well known during the Netflix Prize competition(2006-2009).
Using a Least Absolute loss function rather than a Quadratic Loss Function lead to the Alternating Least Square (ALS) Method, that it runs faster.

Deep Learning also can be used for matrix factorization. The input layer considers the users and items units.

Problems with Collaborative Filtering

- **Cold Start**: There needs to be enough other users already in the system to find a match.
- **Sparsity**: If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is hard to find users that have rated the same items.
- **First Rater**: Cannot recommend an item that has not been previously rated.
  – New items
  – Esoteric items
- **Popularity Bias**: Cannot recommend items to someone with unique tastes.
  – Tends to recommend popular items.

Content-Based Recommending

- Recommendations are based on information on the content of items rather than on other users' opinions.

- Uses a machine learning algorithm to induce a profile of the users preferences from examples based on a featural description of content.

- Some previous applications:
  - Newsweeder (Lang, 1995)
  - Syskill and Webert (Pazzani et al., 1996)

Advantages of Content-Based Approach

- No need for data on other users.

  - No cold-start or sparsity problems.

- Able to recommend to users with unique tastes.

- Able to recommend new and unpopular items

  - No first-rater problem.

- Can provide explanations of recommended items by listing content-features that caused an item to be recommended.

Disadvantages of Content-Based Method

- Requires content that can be encoded as meaningful features.

- Users' tastes must be represented as a learnable function of these content features.

- Unable to exploit quality judgments of other users.
  - Unless these are somehow included in the content features.

Combining Content and Collaboration

- Content-based and collaborative methods have complementary strengths and weaknesses.

- Combine methods to obtain the best of both.

- Various hybrid approaches:
  - Apply both methods and combine recommendations.
  - Use collaborative data as content.
  - Use content-based predictor as another collaborator.
  - **Use content-based predictor to complete collaborative data.**

Metrics

- ## Mean Absolute Error (MAE)

  – Compares numerical predictions with user ratings

- ## ROC sensitivity [Herlocker 99]

  – How well predictions help users select *high-quality* items

  – Ratings ≥ 4 considered "good"; < 4 considered "bad"

- ## Recall

- ## Precision

Metrics

RMSE (Root Mean Squared Error):

- It measures the error in the predicted ratings:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

-
    - Here, <u>Predicted</u> is the rating predicted by the model and Actual is the original rating
    - If a user has given a rating of 5 to a movie and we predicted the rating as 4, then RMSE is 1
    - Lesser the RMSE value, better the recommendations

The above metrics tell us how accurate our recommendations <u>are</u> but they do not focus on the order of recommendations, i.e. they do not focus on which product to recommend first and what follows after that.

Metcis

Mean Reciprocal Rank:

- Evaluates the list of recommendations

$$MRR = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{r(Q_i)}$$

-

  o Suppose we have recommended 3 movies to a user, say A, B, C in the given order, but the user only liked movie C. As the rank of movie C is 3, the reciprocal rank will be 1/3
  o Larger the mean reciprocal rank, better the recommendations

Conclusions

- Recommending and personalization are important approaches to combating information over-load.
- Machine Learning is an important part of systems for these tasks.
- Collaborative filtering has problems.
- Content-based methods address these problems (but have problems of their own).
- Integrating both is best.

Recommendation systems using Deep Learning

Taking from https://towardsdatascience.com/building-a-recommendation-system-using-neural-network-embeddings-1ef92e5c80c9
We will see how to use an embedding neural network to create a book recommendation system.
For our data, we will use the goodbooks-10k dataset which contains ten thousand different books and about one million ratings. It has three features the book_id, user_id and rating.
Our recommendation system will be built on the idea that books which has the same rating are similar to one another. We can represent this similarity and hence make recommendations by *learning embeddings* of books using a neural network. The end result is an effective recommendation system and a practical application of deep learning.

Recommendation systems using Deep Learning

Embedding

An embedding is a mapping from discrete objects, such as words or ids of books in our case, to a vector of continuous values. This can be used to find similarities between the discrete objects, that wouldn't be apparent to the model if it didn't use embedding layers.

The embedding vectors are low-dimensional and get updated while training the network. Neural network embeddings are useful because they can *reduce the dimensionality* of categorical variables and *meaningfully represent* categories in the transformed space. Neural network embeddings have 3 primary purposes:

- Finding nearest neighbors in the embedding space. These can be used to make recommendations based on user interests or cluster categories.
- As input to a machine learning model for a supervised task.
- For visualization of concepts and relations between categories.

Recommendation systems using Deep Learning

Neural network embeddings overcome the two limitations of a common method for representing categorical variables: one-hot encoding.

Limitations of One Hot Encoding

The operation of one-hot encoding categorical variables is actually a simple embedding where each category is mapped to a different vector. This process takes discrete entities and maps each observation to a vector of 0s and a single 1 for a specific category.

The one-hot encoding technique has two main drawbacks:

For high-cardinality variables—those with many unique categories—the dimensionality of the transformed vector becomes unmanageable.

The mapping is completely uninformed: "similar" categories are not placed closer to each other in embedding space.

Recommendation systems using Deep Learning

```
# One Hot Encoding Categoricals
books = ["War and Peace", "Anna Karenina",
        "The Hitchhiker's Guide to the Galaxy"]
books_encoded = [[1, 0, 0],
                 [0, 1, 0],
                 [0, 0, 1]]
Similarity (dot product) between First and Second = 0
Similarity (dot product) between Second and Third = 0
Similarity (dot product) between First and Third = 0
```

Recommendation systems using Deep Learning

```
# Idealized Representation of Embeddingbooks = ["War and Peace",
"Anna Karenina",
         "The Hitchhiker's Guide to the Galaxy"]
books_encoded_ideal = [[0.53,  0.85],
                       [0.60,  0.80],
                       [-0.78, -0.62]]Similarity (dot product)
between First and Second = 0.99
Similarity (dot product) between Second and Third = -0.94
Similarity (dot product) between First and Third = -0.97
```
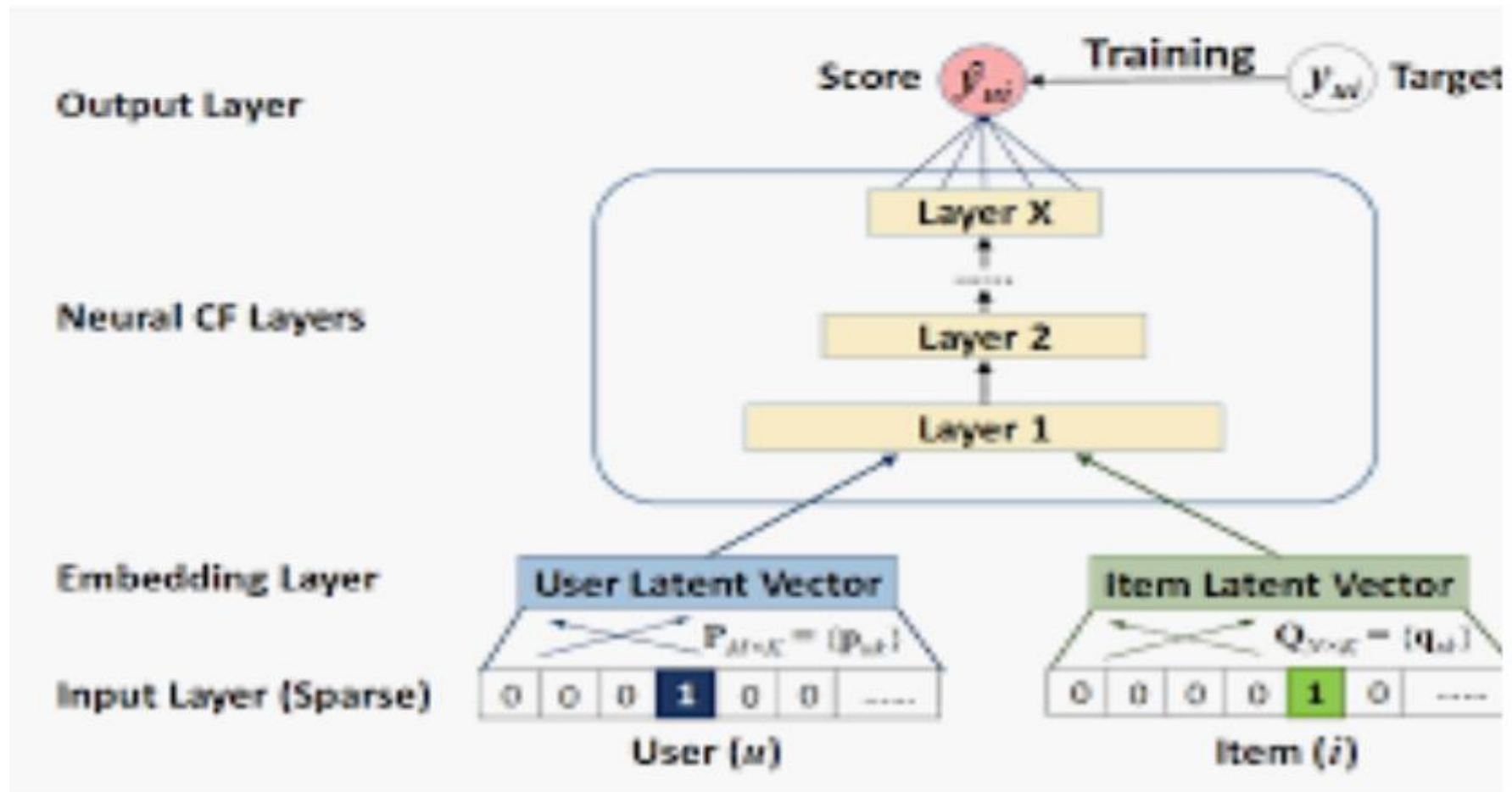
Recommendation systems using Deep Learning

The embedding neural network model used in our example has 3 layers:

Input: parallel inputs for the book (item) and users
Embedding: parallel length 50 embeddings for the book and users.
Dot: merges embeddings by computing dot product

Recommendation systems using Deep Learning

In an embedding neural network, the embeddings are the parameters(weights) of the neural network that are adjusted during training in order to minimize loss on the objective. The neural network takes in a book and a user as integers and outputs a prediction between 0 and 1 that is compared to the true value. The model is compiled with the Adam optimizer (a variant on Stochastic Gradient Descent) which, during training, alters the embeddings to minimize the binary_crossentropy for this binary classification problem.