

---

# Matrix Factorization

## ALS: Alternating Least Squares

**Nagiza F. Samatova**, [samatova@csc.ncsu.edu](mailto:samatova@csc.ncsu.edu)

Professor, Department of Computer Science  
North Carolina State University

Senior Scientist, Computer Science & Mathematics Division  
Oak Ridge National Laboratory

# Outline

---

- **Math Preliminaries**
  - **Graph Theory**
  - **Bipartite Graphs**
- **Business Intelligence Use Case: Recommendation Systems**
- **Math Preliminaries**
  - **Matrix Inverse, Transpose, Singular Matrix**
- **Matrix Factorization**
  - **SVD, PCA**
  - **NMF: Non-negative Matrix Factorization**
- **Alternating Least Squares for Matrix Completion**
  - **ALS Algorithm**
  - **ALS vs. SVD**

---

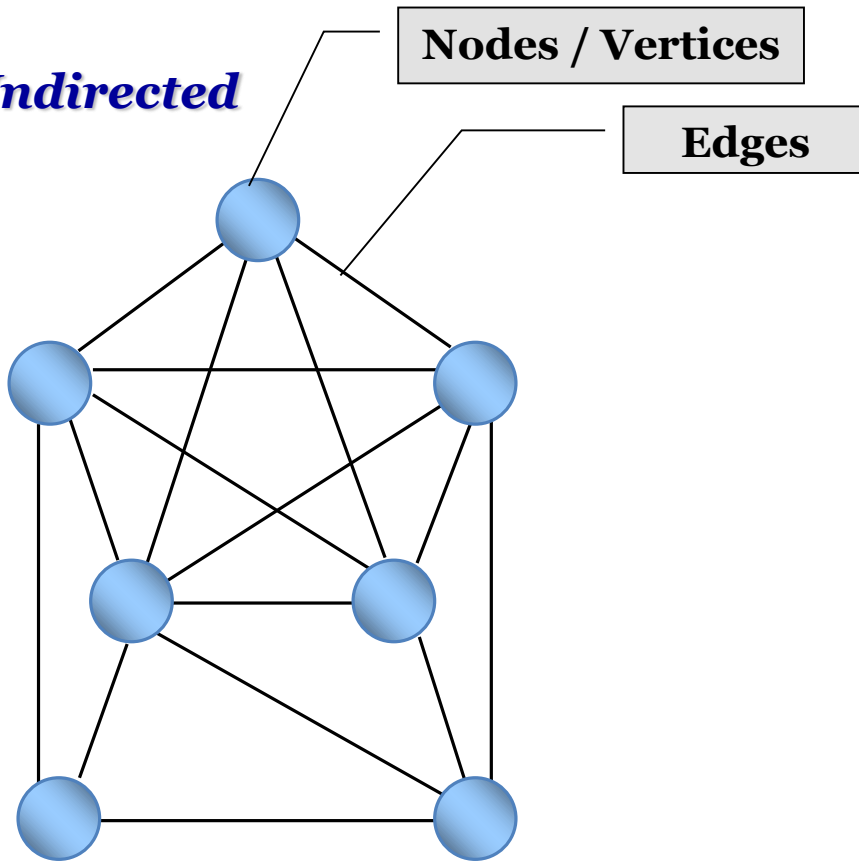
# Mathematical Preliminaries

## **GRAPH THEORY**

# Graphs

Graph with 7 nodes and 16 edges

**Undirected**



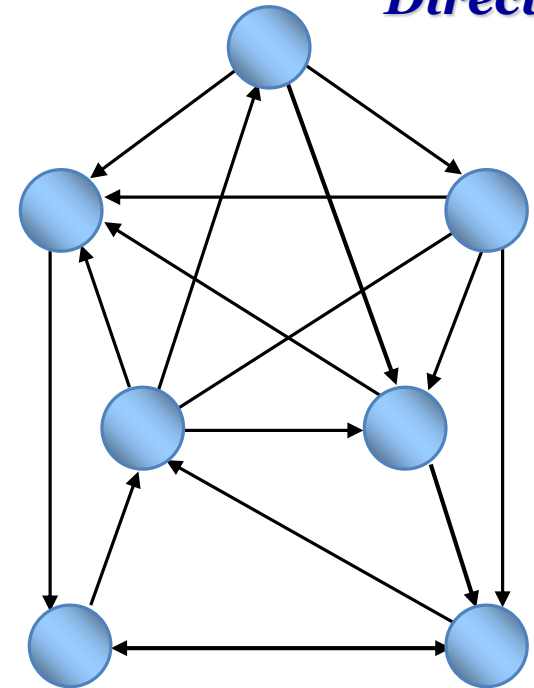
$$(v_i, v_j) = (v_j, v_i)$$

$$G = (V, E)$$

$$V = \{v_1, v_2, \dots, v_n\}$$

$$E = \{e_k = (v_i, v_j) \mid v_i, v_j \in V, k = 1, \dots, m\}$$

**Directed**



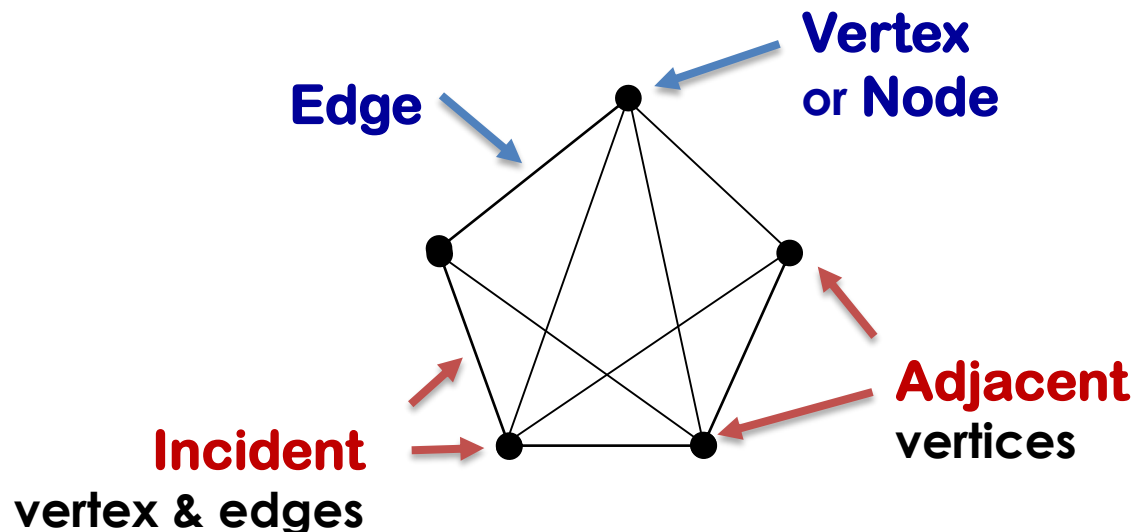
$$(v_i, v_j) \neq (v_j, v_i)$$

# Graph Theory

A **graph** is a collection of vertices (nodes) and edges.

A **vertex** represents some object.

An **edge** connects two vertices and represents some **relationship** between them.



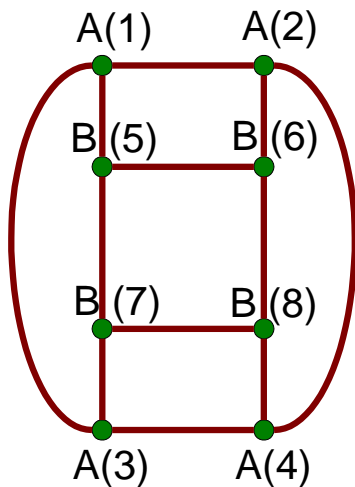
# Graph Representation as a Matrix

---

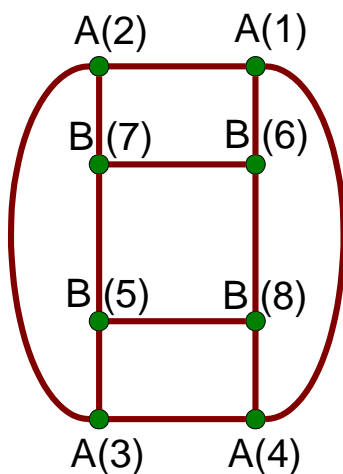
- **Adjacency Matrix** (vertex vs. vertex)
- **Incidence Matrix** (vertex vs. edge)
- **Sparse** (lots of zero's) vs. **Dense** Matrices

# Adjacency Matrix Representation

The representation is *NOT* unique.  
Some algorithms are *order-sensitive*.



	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	1	0	1	0	0	0
A(2)	1	1	0	1	0	1	0	0
A(3)	1	0	1	1	0	0	1	0
A(4)	0	1	1	1	0	0	0	1
B(5)	1	0	0	0	1	1	1	0
B(6)	0	1	0	0	1	1	0	1
B(7)	0	0	1	0	1	0	1	1
B(8)	0	0	0	1	0	1	1	1



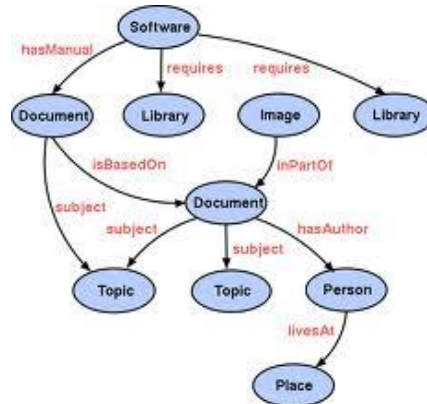
	A(1)	A(2)	A(3)	A(4)	B(5)	B(6)	B(7)	B(8)
A(1)	1	1	0	1	0	1	0	0
A(2)	1	1	1	0	0	0	1	0
A(3)	0	1	1	1	1	0	0	0
A(4)	1	0	1	1	0	0	0	1
B(5)	0	0	1	0	1	0	1	1
B(6)	1	0	0	0	0	1	1	1
B(7)	0	1	0	0	1	1	1	0
B(8)	0	0	0	1	1	1	0	1

# What apps naturally deal w/ graphs?

## Social Networks



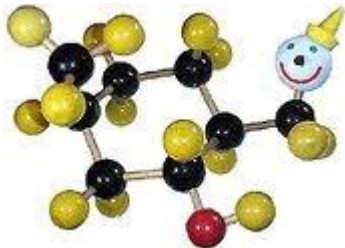
## Semantic Web



## World Wide Web



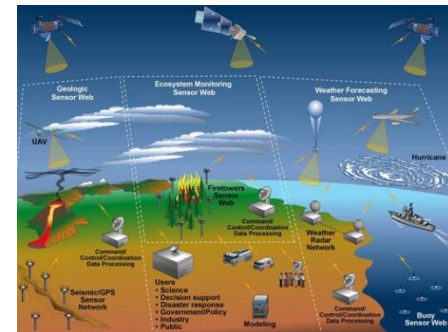
## Drug Design, Chemical compounds



## Computer networks



## Sensor networks



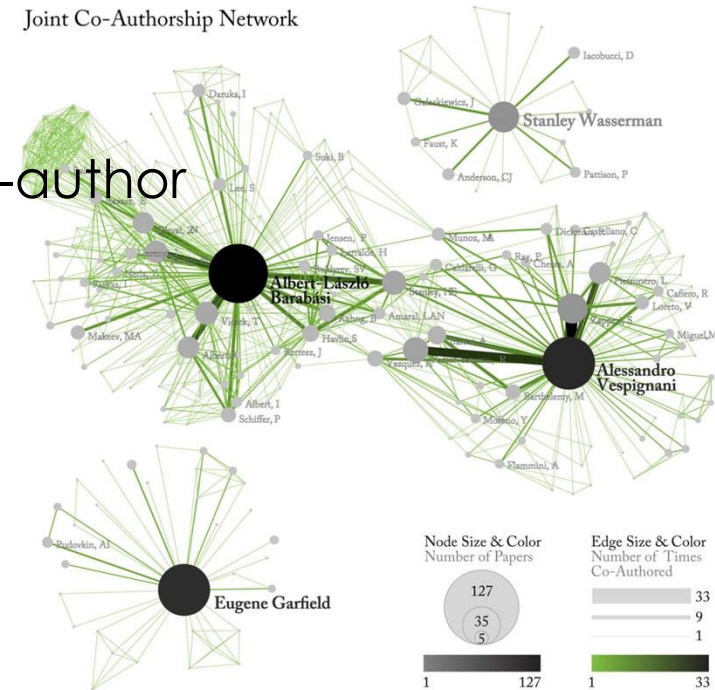


# Real-world Graphs, or Networks

- **Graphs:** consist of objects (vertices) & their relationships (edges/links)
  - **Social:** Facebook, Twitter, LinkedIn
  - **Physical:** Internet, power-grid
  - **Biological:** Protein-protein interactions
  - **Academic Collaboration:** Citation, co-author

**Can you think of real-world graphs/networks?**

- What are their nodes/vertices?
- What are their relationships/edges/links?



# Examples of Real-World Graphs

Graph / Network	Node / Vertex	Edge / Arc / Relationship
Internet	Computer/router	Cable/wireless data connection
WWW	Web page	Hyperlink
Facebook	Person	Friendship
Power Grid	Generating station	Transmission line
Airport Network	Airport in a city	Flight connection
US Roads	City	Roads
Neural Network	Neuron	Synapse

# How Big Are These Graphs?

Graphs encode rich relationships

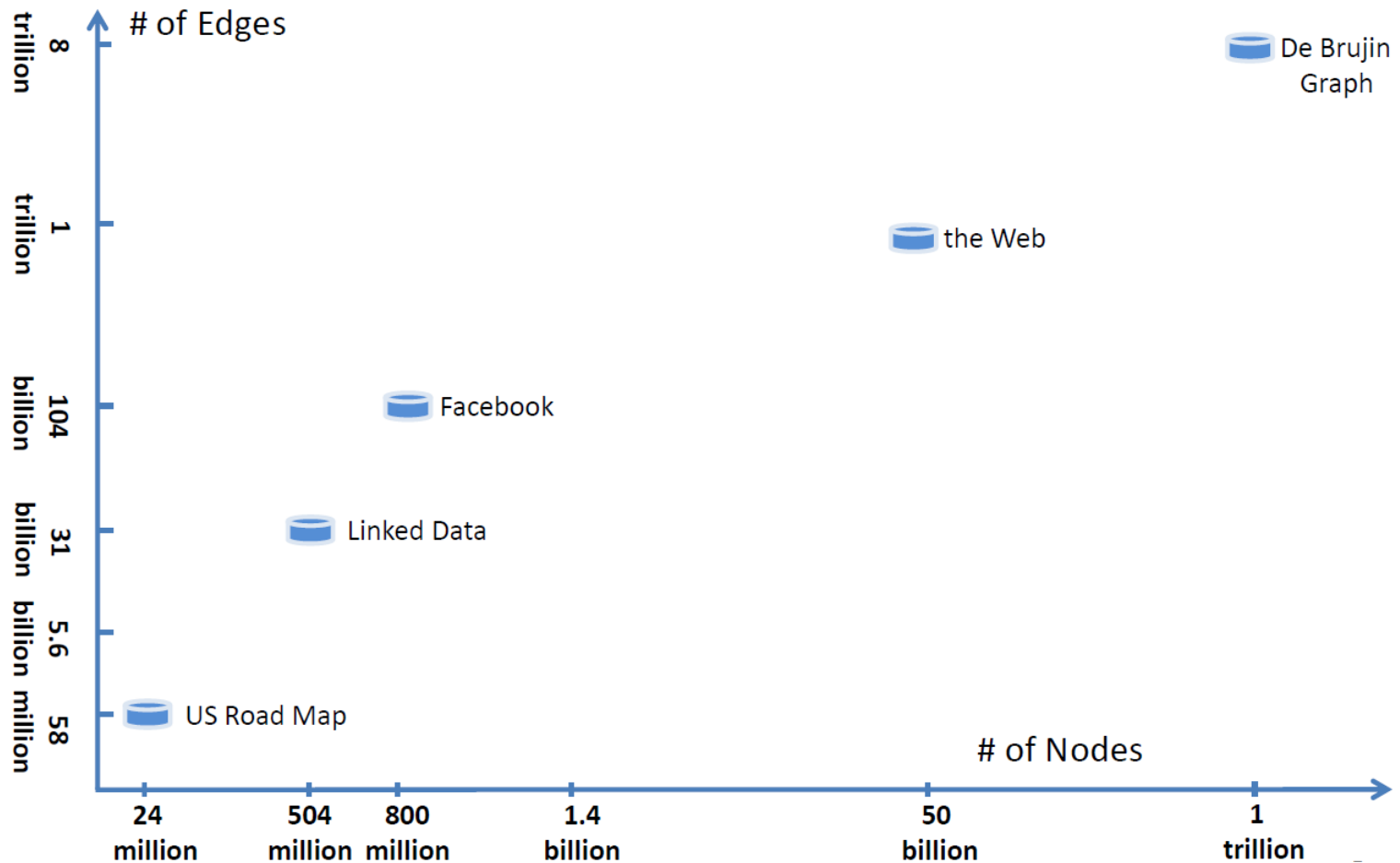


Image source: Haixun Wang, KDD 2012

# Graphs Challenge Many Algorithms

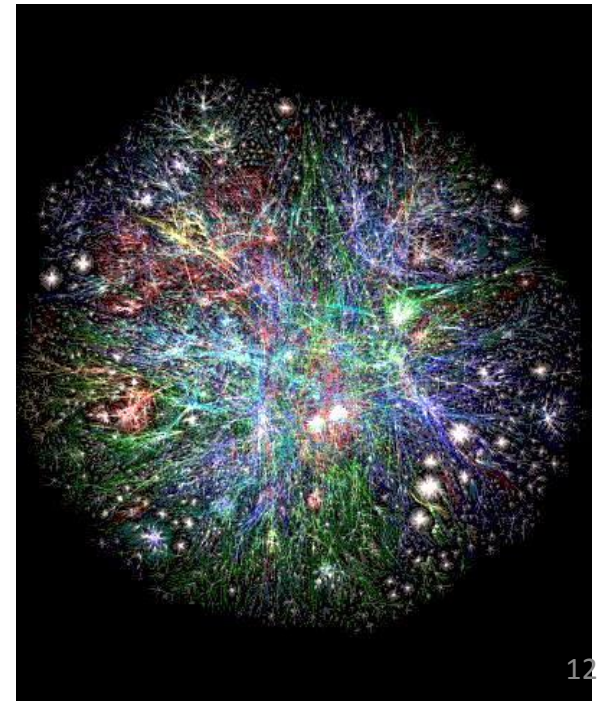
## Massive scale

- Can't use conventional algorithms on large graphs
- The graph itself may not even fit in memory

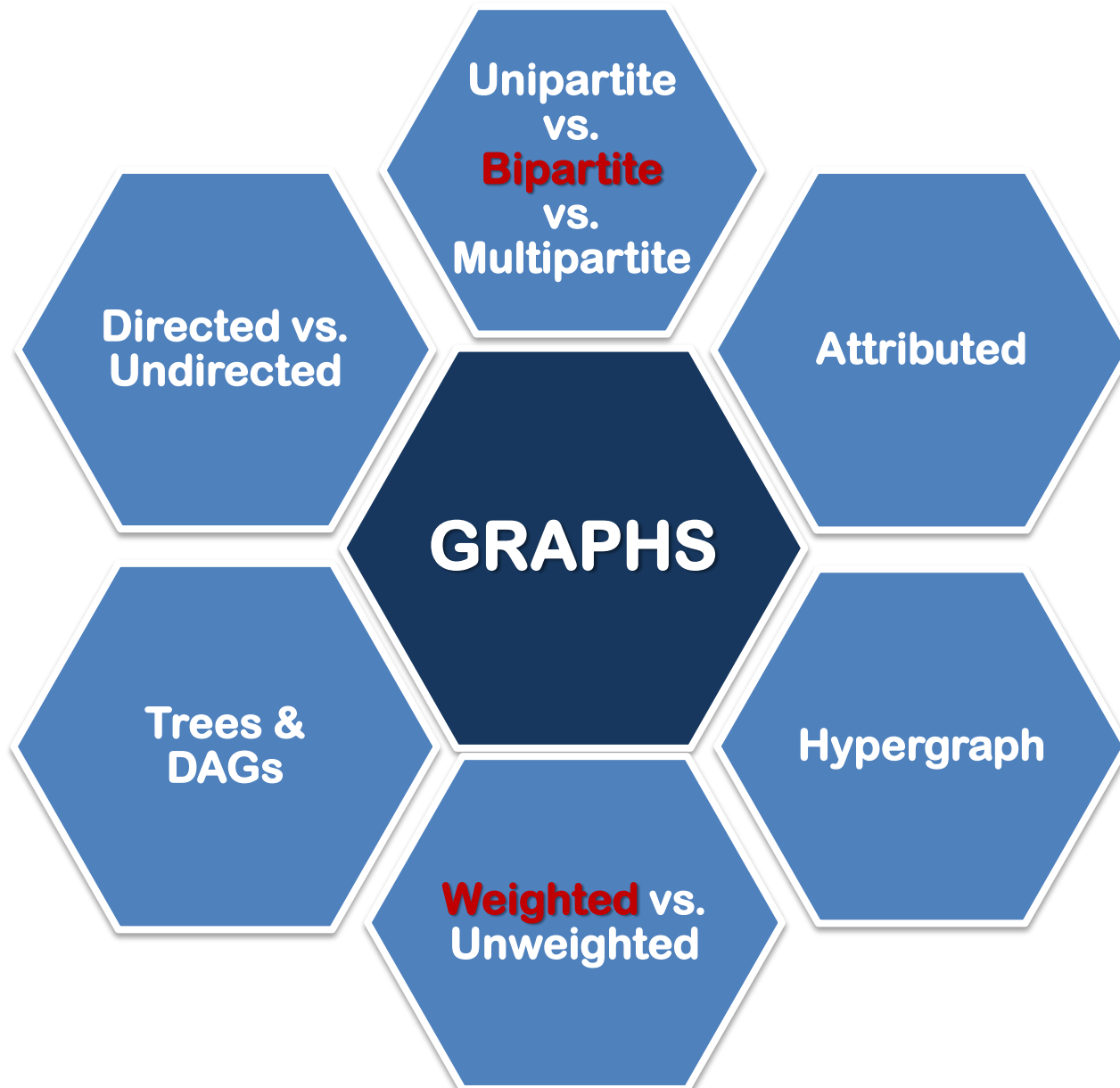
Facebook:  $|V| = 721\text{M}$ ,  $|E| = 137\text{B}$

Common crawl:  $|V| = 3.5\text{B}$ ,  $|E| = 128\text{B}$

$$O(n^2) \text{ ☹}$$



# Types of Graphs

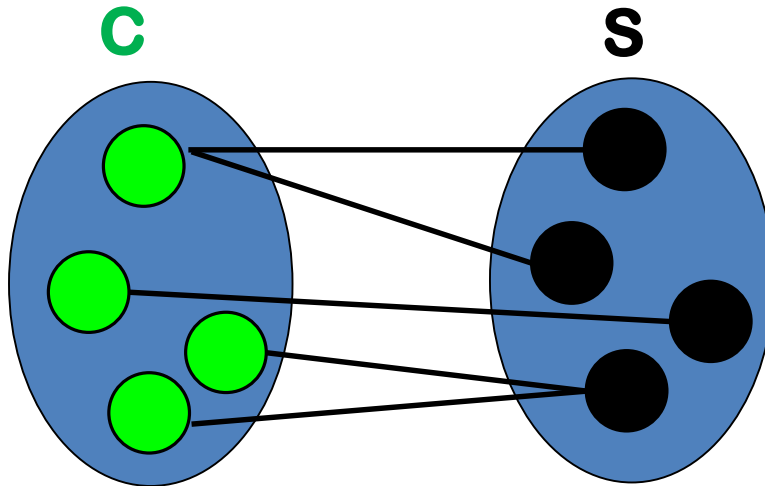


# Bipartite Graphs

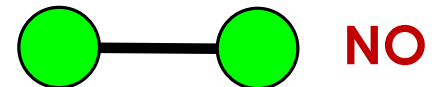
A graph  $B$  is **bipartite** if its vertices  $V$  can be partitioned into two non-intersecting sets  $C$  and  $S$  such that all relationships/edges go between  $C$  and  $S$  (no edges go from  $C$  to  $C$  or from  $S$  to  $S$ ).

$B = (V, R)$  is **bipartite** graph such that:

- (1)  $V = C \cup S$  and  $C \cap S = \emptyset$
- (2)  $R = \{r_{cs} = (c, s): c \in C \text{ and } s \in S\}$



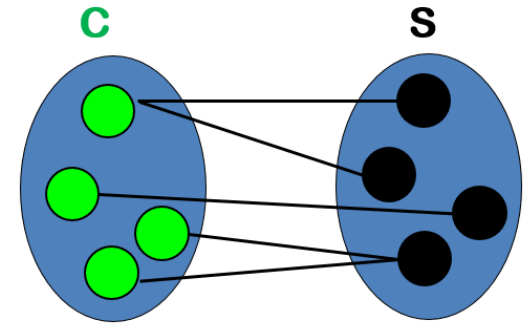
Relation/Edge Types



# Bipartite Graphs in Business Intelligence (BI)

$B = (V, R)$  is **bipartite** graph such that:

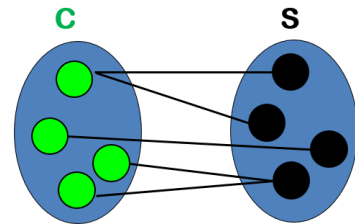
- (1)  $V = C \cup S$  and  $C \cap S = \emptyset$
- (2)  $R = \{r_{cs} = (c, s): c \in C \text{ and } s \in S\}$



Business, <b>B</b>	Customers, <b>C</b>	Services, <b>S</b>	Relationships, <b>R</b>
Amazon	Customers	Products	Purchasing
Netflix	Subscribers	Movies	Watching
Pandora	Subscribers	Songs	Listening to
CISCO	Manufacturers	Parts	Supplying to CISCO

# Observations about Bipartite BI Graphs

- Adjacency matrix is very **sparse**: (  $r_{cs} = 0$  or  $r_{cs} = ???$  )
- Relationships are not always binary, but **weighted**:
  - $r_{cs} = *****$ : How customer rated a product
  - $r_{cs} = 25$ : How many times subscriber listened to the song
  - $r_{cs} = High$ : How reliable manufacturer was in supplying the part
- Missing** relationships (  $r_{cs} = 0$  or  $r_{cs} = ???$  ) are **business opportunities**:
  - Recommend new products to customers
  - Find reliable manufacturers
  - Ensure continued customer subscriptions to services



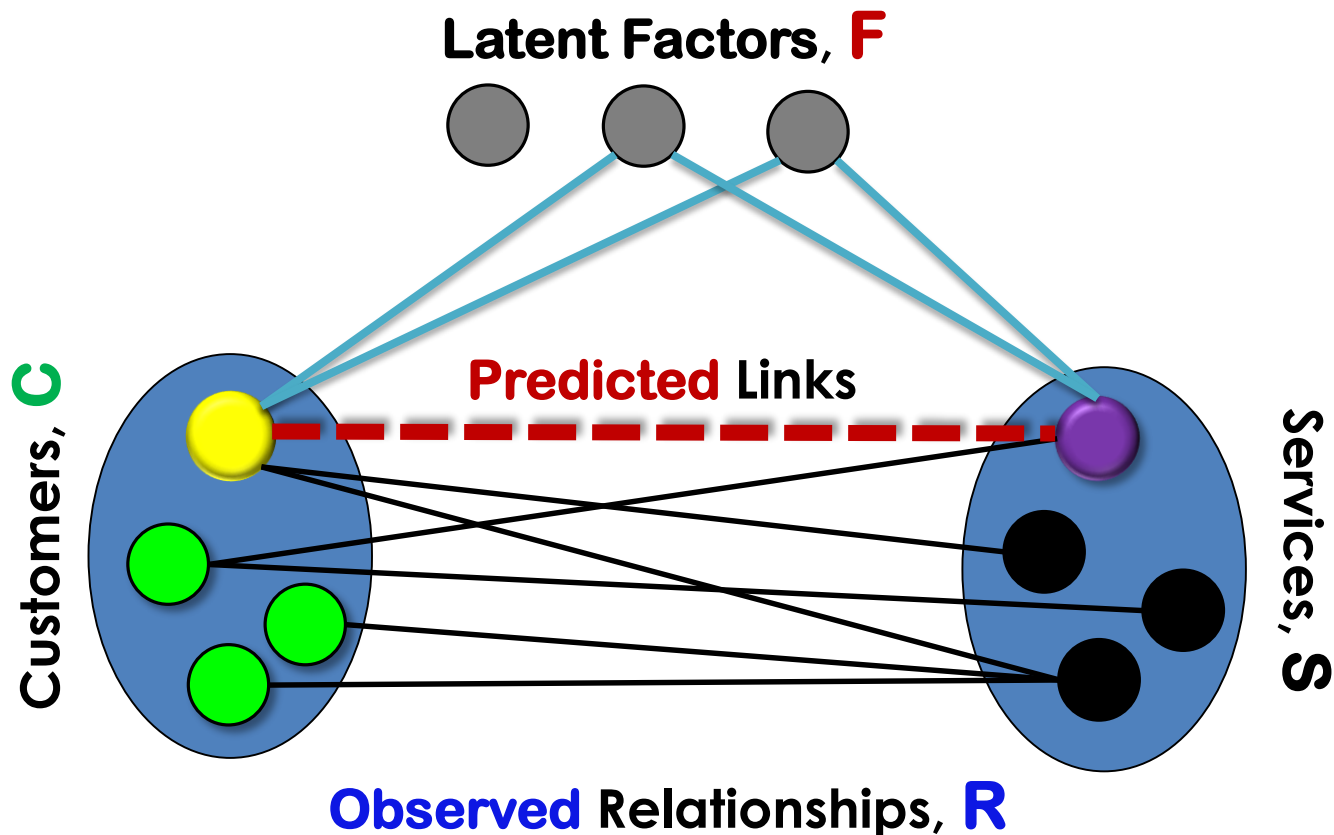
Business, <b>B</b>	Customers, <b>C</b>	Services, <b>S</b>	Relationships, <b>R</b>
Amazon	Customers	Products	Purchasing /Rating
Netflix	Subscribers	Movies	Watching
Pandora	Subscribers	Songs	Listening to
CISCO	Manufacturers	Parts	Supplying to CISCO



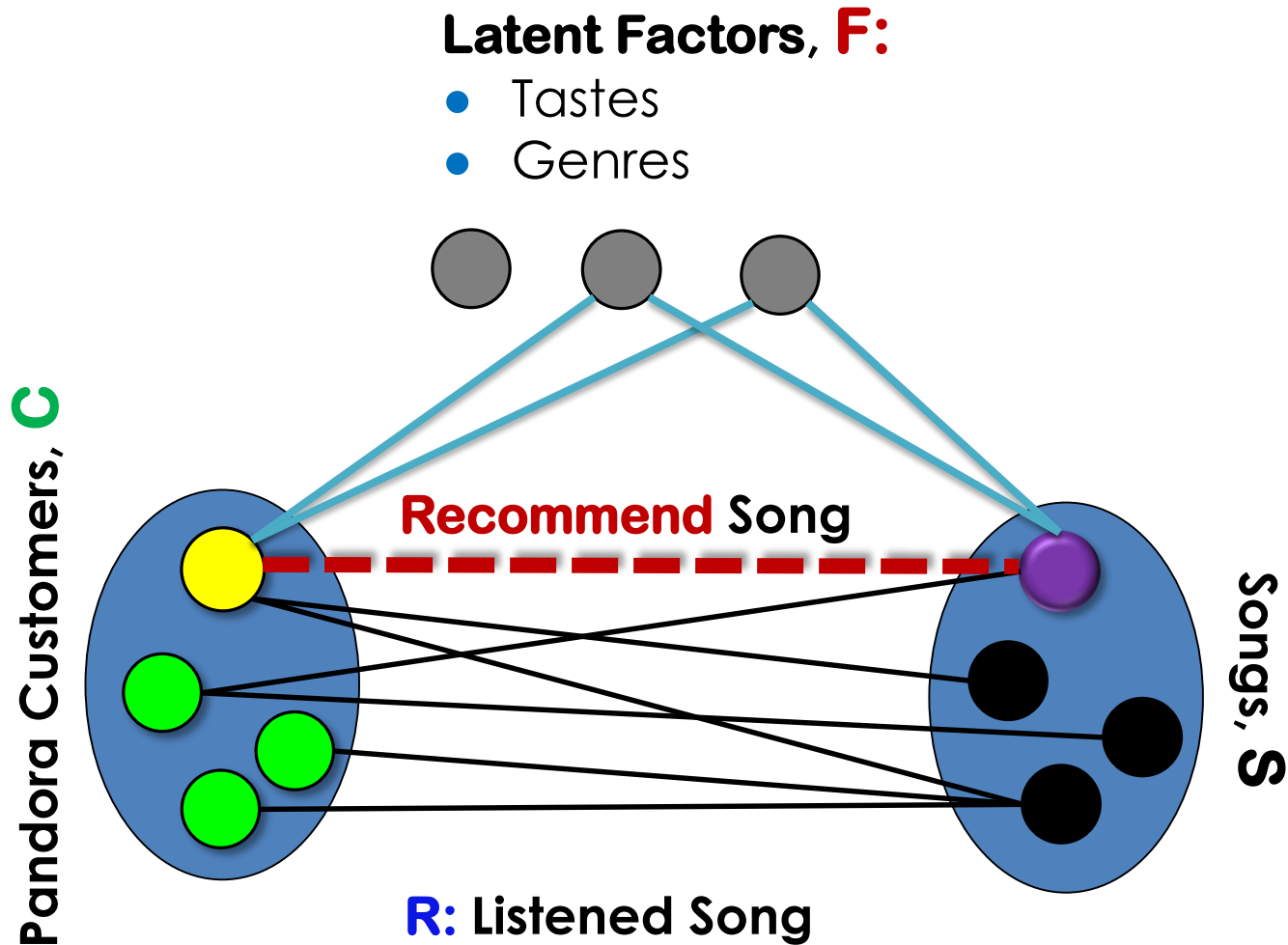
# Assumptions: Hidden Factors & Missing Links

There are a few **hidden** or **latent factors**  $F = \{f_1, f_2, \dots, f_h\}$  that define/drive the **relationships** between **C** and **S**.

These **latent / hidden factors** can be used **to complete missing relationships** between **C** and **S**.



# Example: Hidden Factors & Missing Links

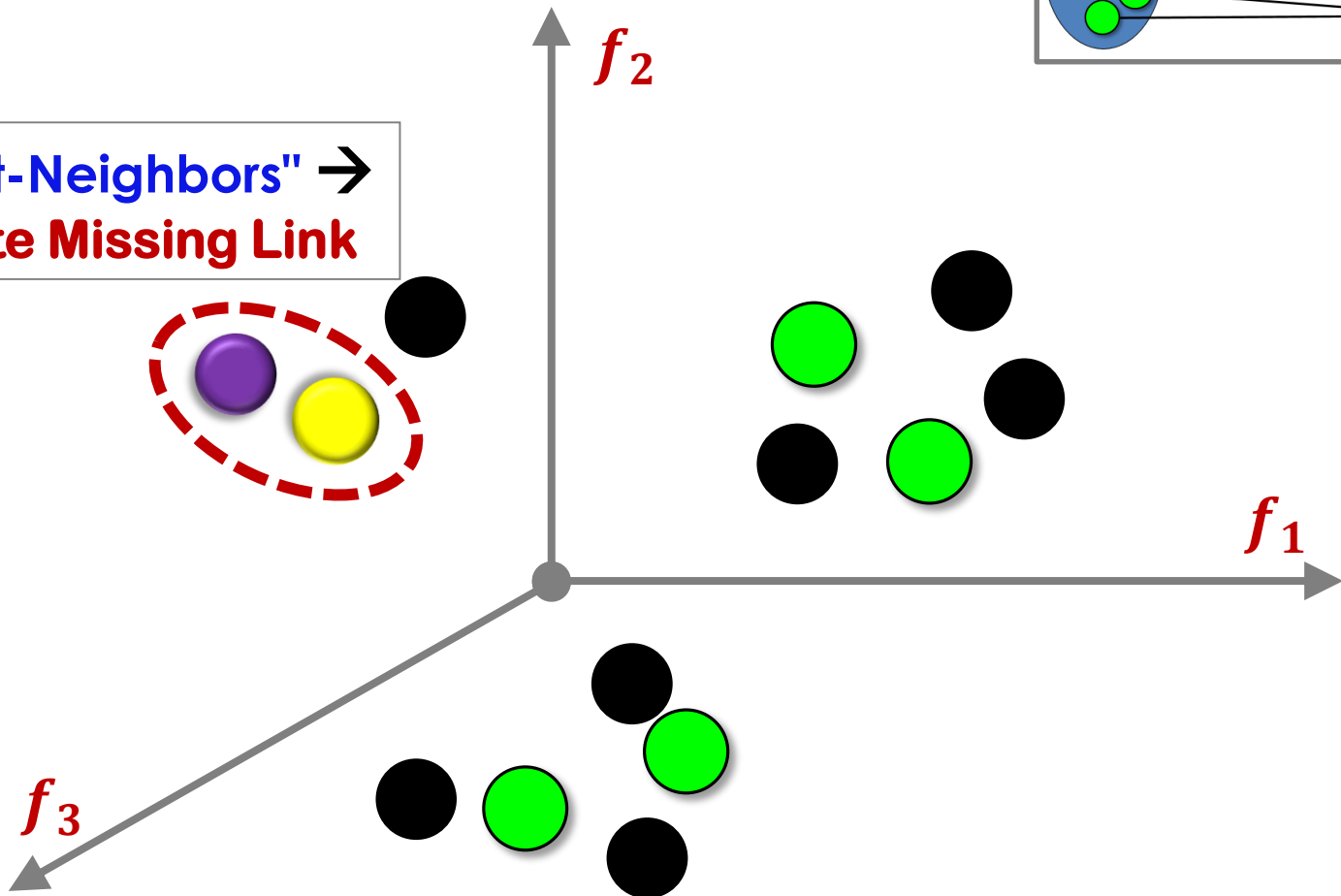


# Latent Factor Space View

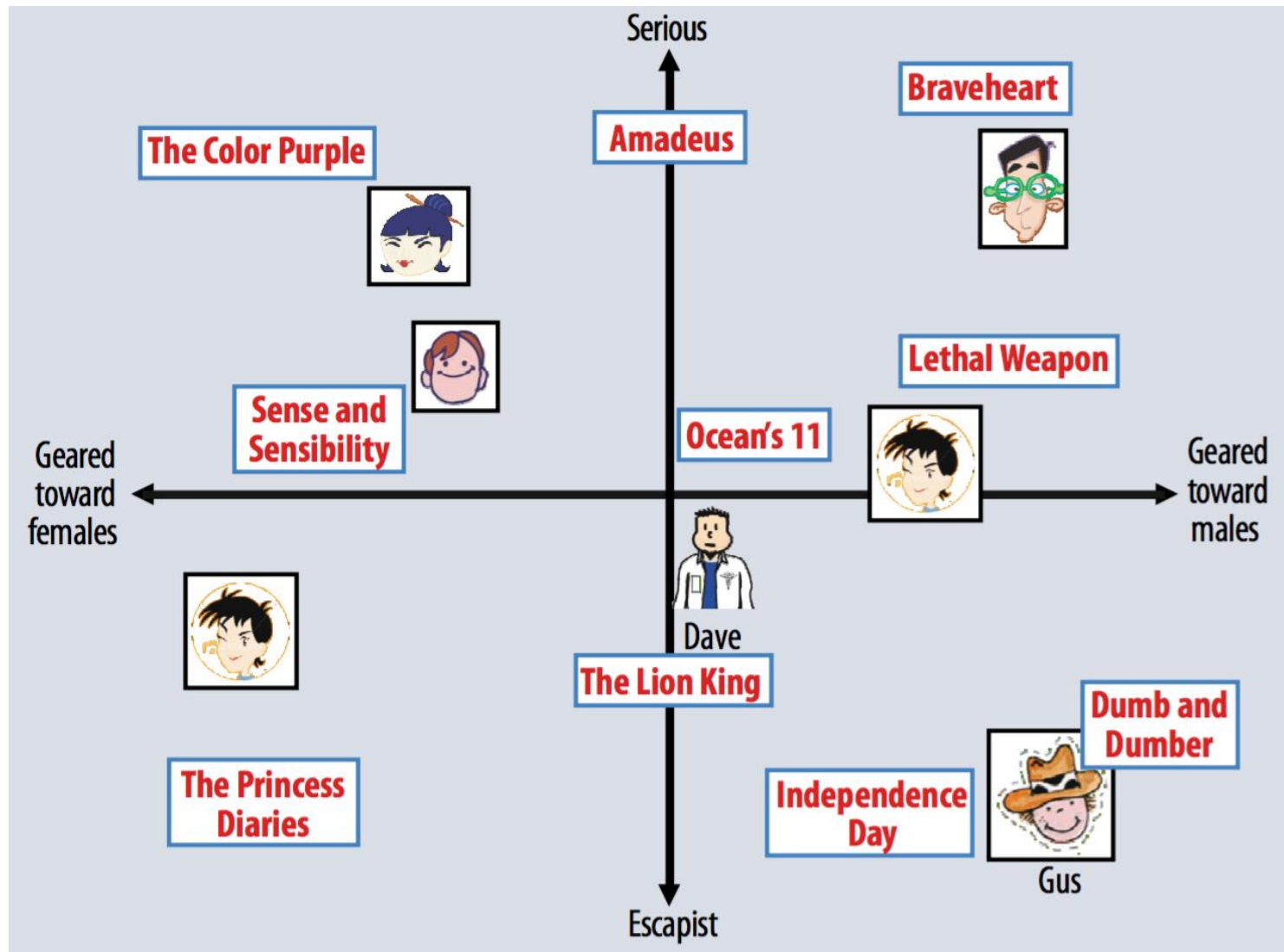
Customer-Service Relations in the **Latent Factor Space**:

$$F = \{f_1, f_2, \dots, f_h\}$$

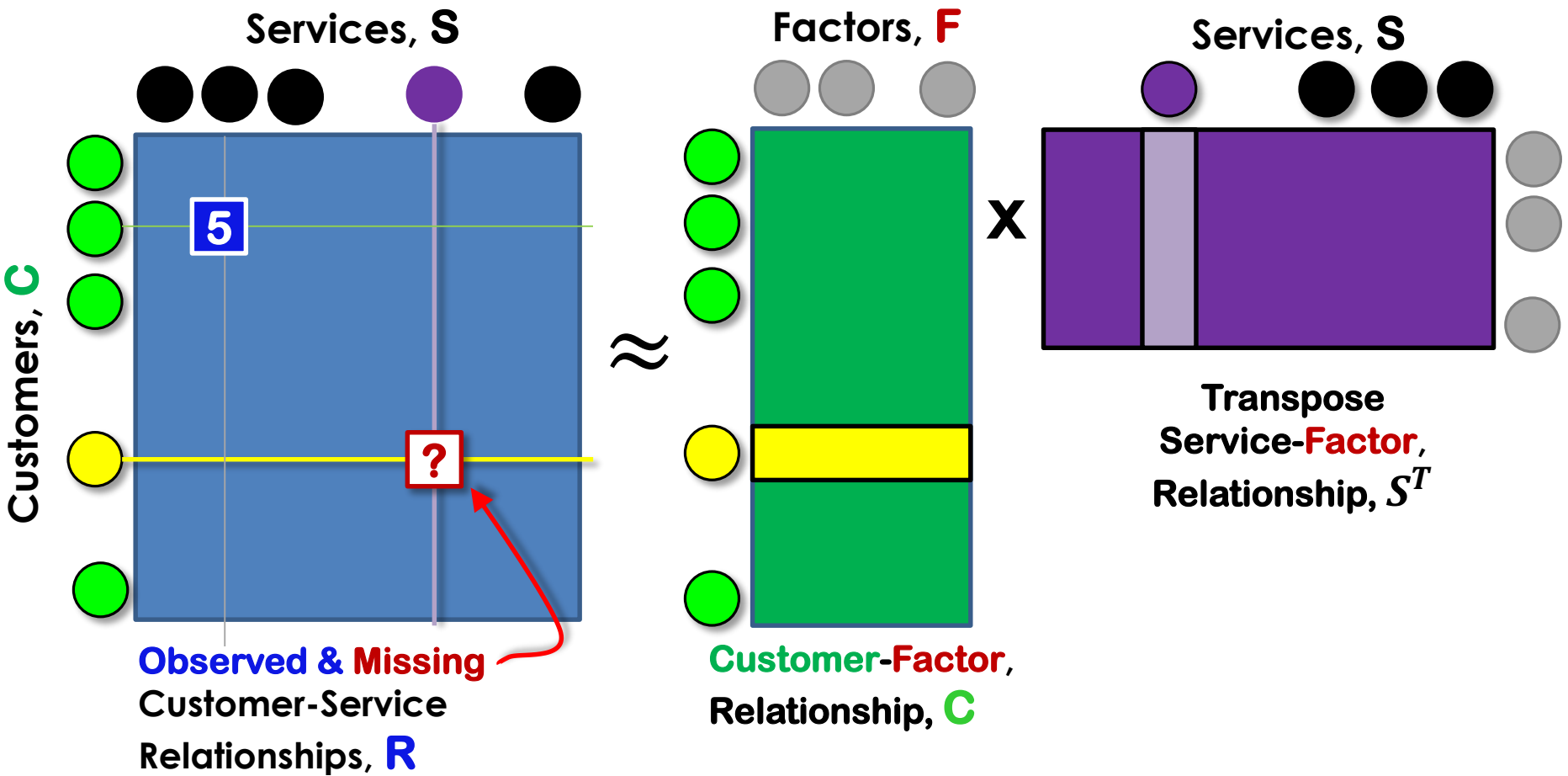
"Nearest-Neighbors" →  
**Complete Missing Link**



# Ex: User-Movie Relations in a Latent Factor Space



# Matrix Factorization / Completion View

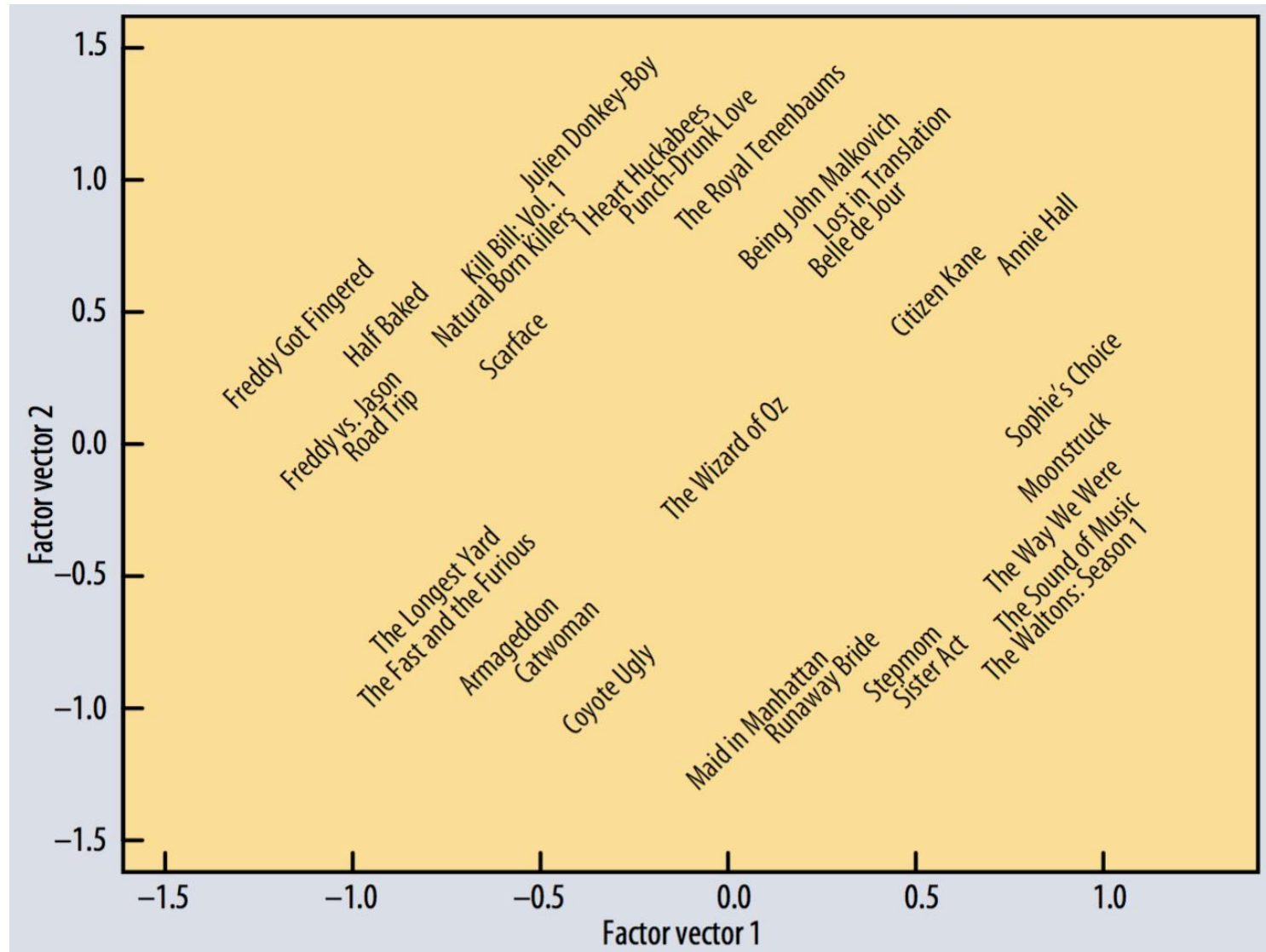


$$R \approx C \times S^T$$

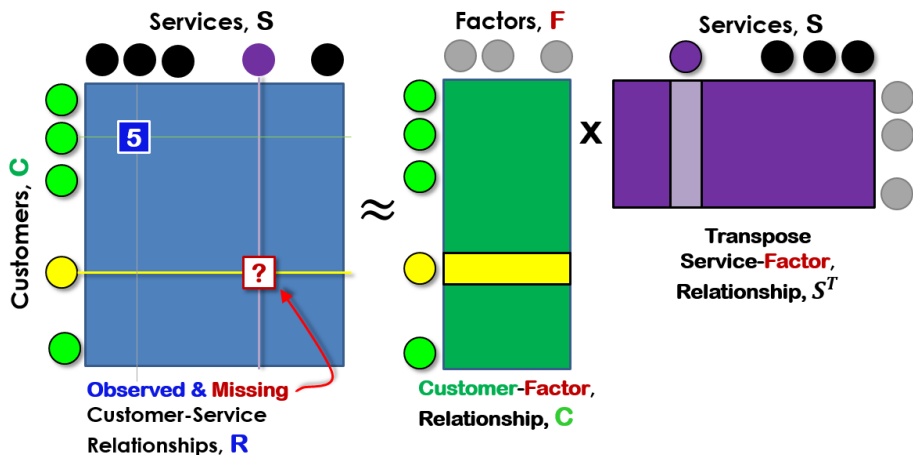
Complete Missing Values

$$[?] \approx [\text{Yellow Box}] \times [\text{Purple Box}]$$

# First Two Vectors from Matrix Decomposition



# Matrix Notation & Observations



Customers:  $C = \{c_1, c_2, \dots, c_n\}$

Services:  $S = \{s_1, s_2, \dots, s_m\}$

Hidden Factors:  $F = \{f_1, f_2, \dots, f_h\}$

$$R_{n \times m} \approx C_{n \times h} \times S_{h \times m}^T$$

$$r_{i,j} \approx \vec{c}_i \times \vec{s}_j^T$$

vector product

$R$ : sparse matrix

$C \times S^T$ : dense matrix

$C$ : dense matrix

$S$ : dense matrix

$h = O(\text{constant})$

$h \ll n$

$h \ll m$

---

# Business Intelligence Use Case

## **RECOMMENDATION SYSTEMS**

*In our case [Netflix], more than 75% of what people choose come from recommendations.*

*Xavier Amatriain, former Director of Research @ Netflix*



---

*A recommender system aims at providing a personalized list of items ranked according to the preferences of the user, as such **ranking methods are at the core of many recommendation algorithms.***

# Problem: Recommender Systems (RS)

---

- RS seen as a function
- Given:
  - **User model** (e.g. ratings, preferences, demographics, situational context)
  - **Items** (with or without description of item characteristics)
- Find:
  - **Relevance score**. Used for ranking.
- Finally:
  - **Recommend relevant items**
- But:
  - Remember that relevance might be context-dependent
  - Characteristics of the list itself might be important (diversity)

# Formally, recommendation problem

---

- Problem
  - Given: the ratings (implicit / explicit) that users provided for certain items
  - Task: predict the ratings for the rest of the items
- Formally
  - If there are  $n$  users and  $m$  items, we are given an  $n \times m$  matrix  $R$  in which the  $r_{ui}$  is the rating value provided by user  $u$  for an item  $i$ .
  - The goal is to estimate the unobserved (missing) entries in the matrix  $R$

# RS: Applications

---

- **Online Shopping & Advertising**
  - Frequently co-purchased items are recommended
- **Entertainment**
  - Movie you may enjoy or Songs you may like
  - Cell phone games you may want to play
  - News you may want to read
- **Social Web**
  - Jobs you may be interested in
  - Groups you may like
  - Experts whose opinion might be relevant
  - Friends you may connect to
- **Finance & Accounting**
  - Stocks that may be in trouble
- **Analytic Workflows**
  - Personal assistance
  - What application feature to look at

# RS: Business Intelligence

---

- **Value for the customer**

- Find things that are interesting
- Narrow down the set of choices
- Help explore the space of options
- Discover new things
- Entertainment
- ...

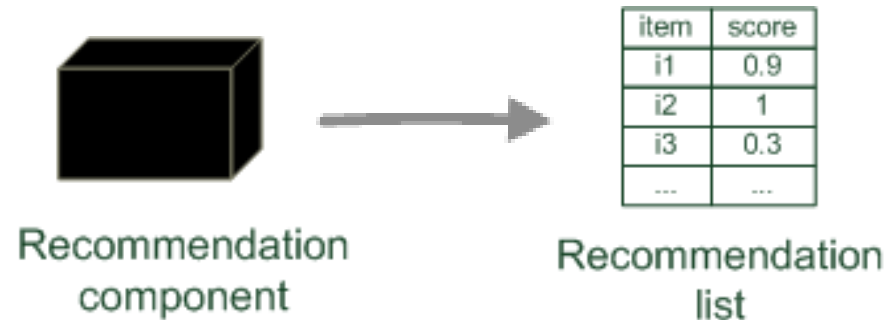
- **Value for the provider**

- Personalized service for the customer
- Increase trust and customer loyalty
- Increase sales, click through rates, conversion etc.
- Opportunities for promotion, persuasion
- Obtain more knowledge about customers
- ...

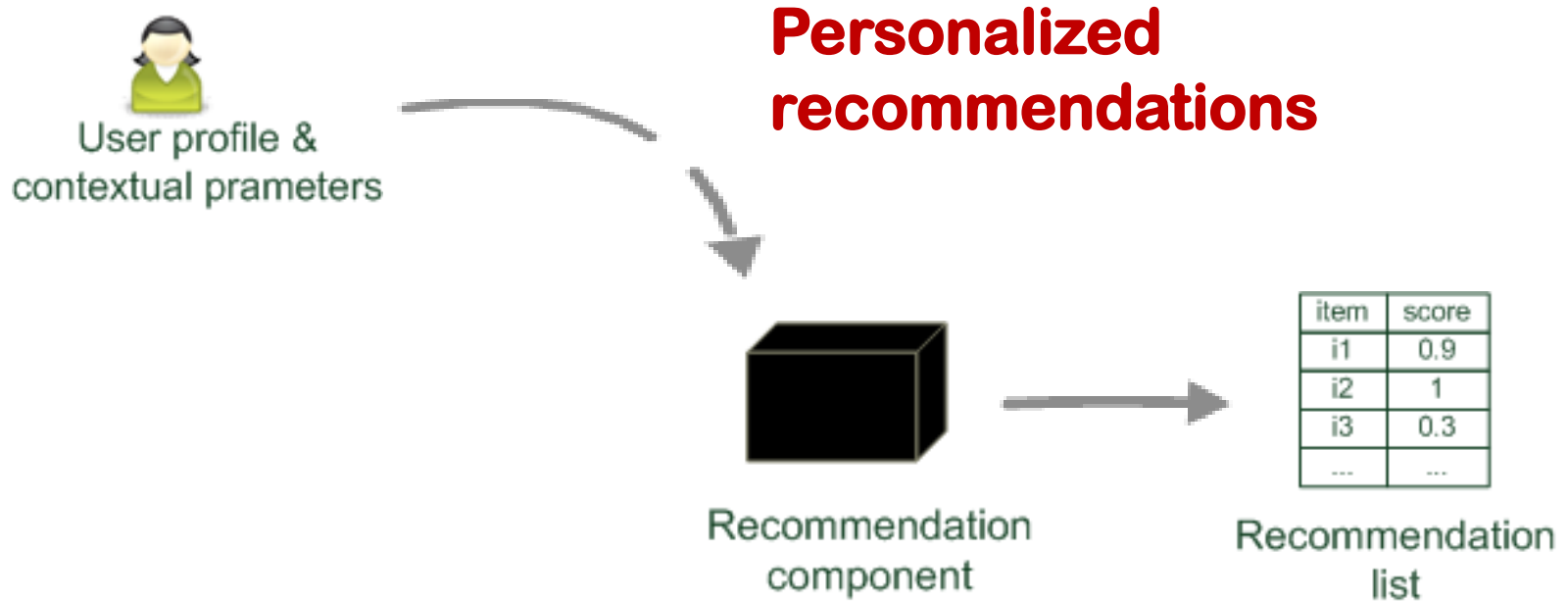
# Paradigms of Recommender Systems

---

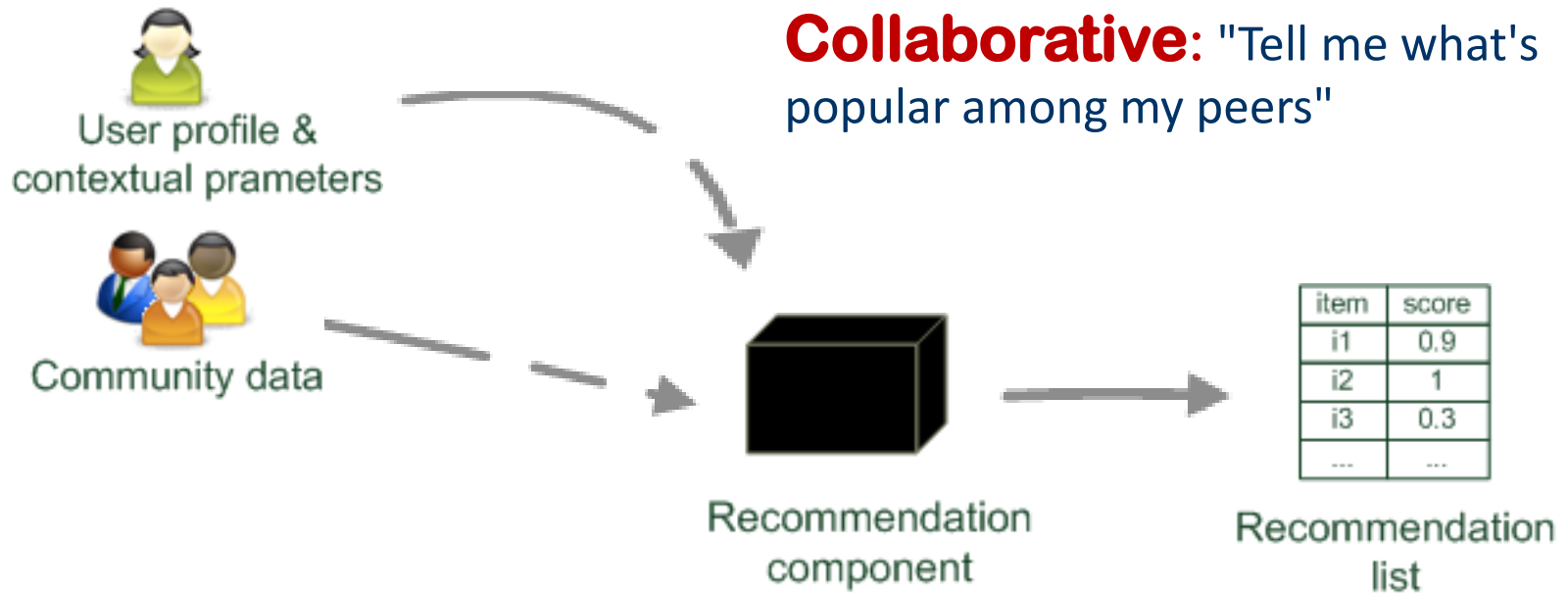
RS **reduce information overload**  
by estimating relevance



# Paradigms of Recommender Systems

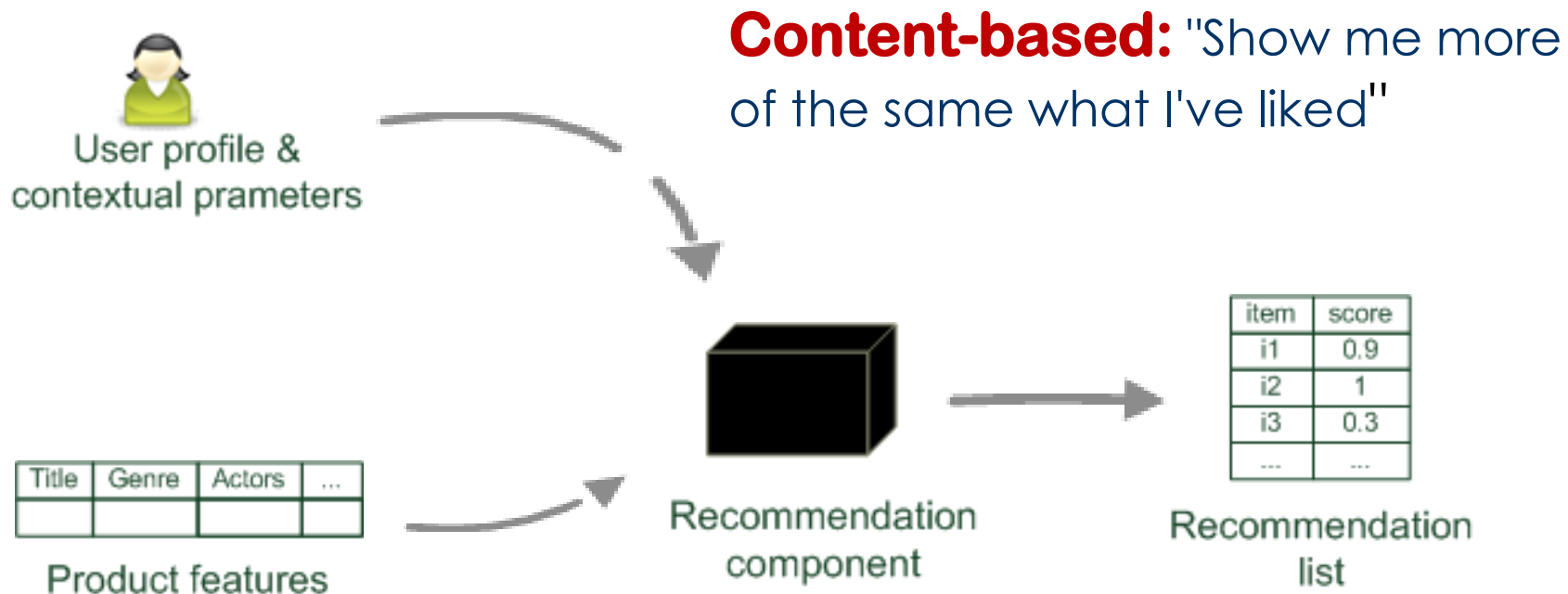


# Paradigms of recommender systems

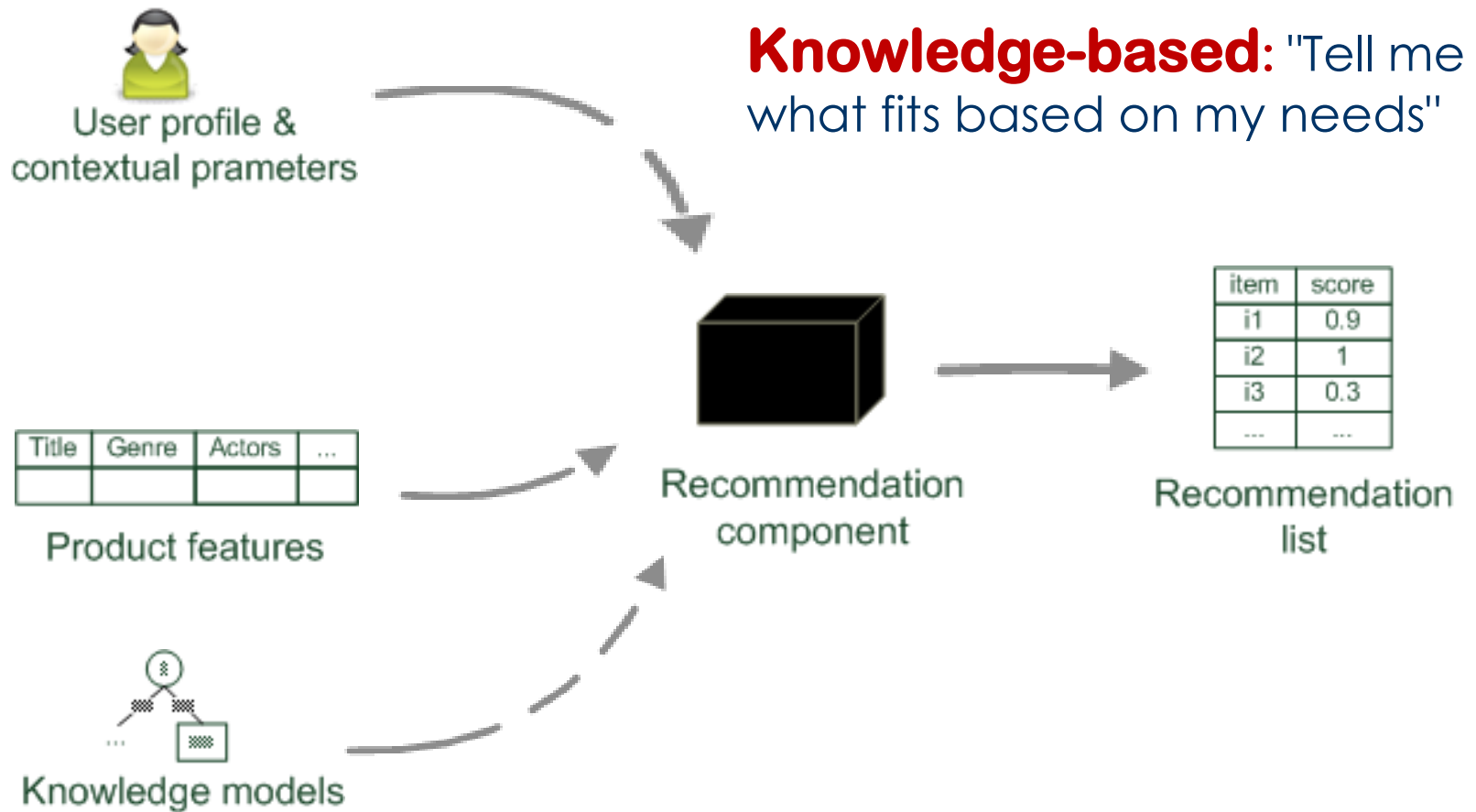




# Paradigms of Recommender Systems

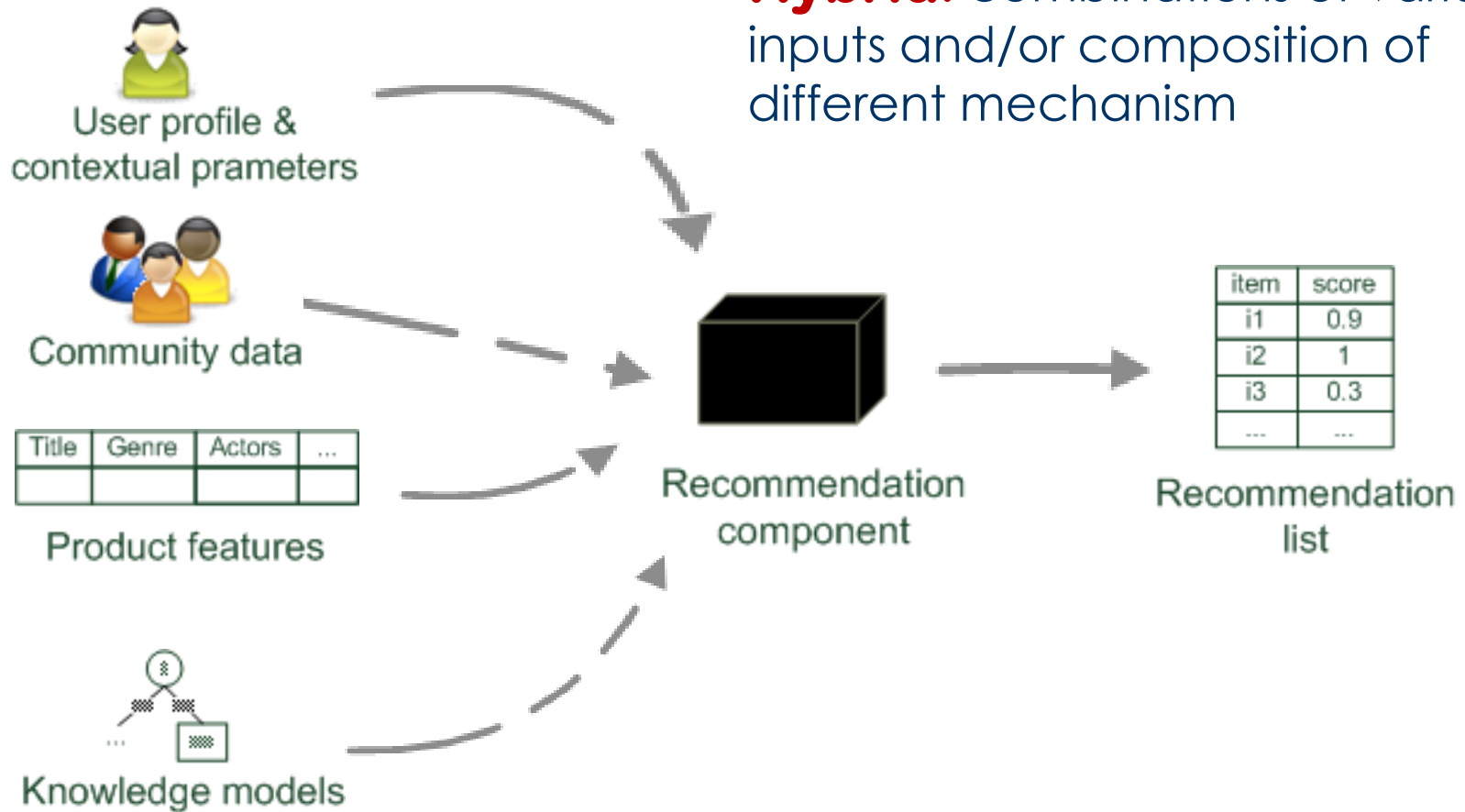


# Paradigms of Recommender Systems





# Paradigms of Recommender Systems

**Hybrid:** combinations of various inputs and/or composition of different mechanism



# Recommender Systems: Basic Techniques

---

	Pros 	Cons 
<b>Collaborative</b>	No knowledge-engineering effort, serendipity of results, learns market segments	Requires some form of rating feedback, cold start for new users and new items
<b>Content-based</b>	No community required, comparison between items possible	Content descriptions necessary, cold start for new users, no surprises
<b>Knowledge-based</b>	Deterministic recommendations, assured quality, no cold-start, can resemble sales dialogue	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

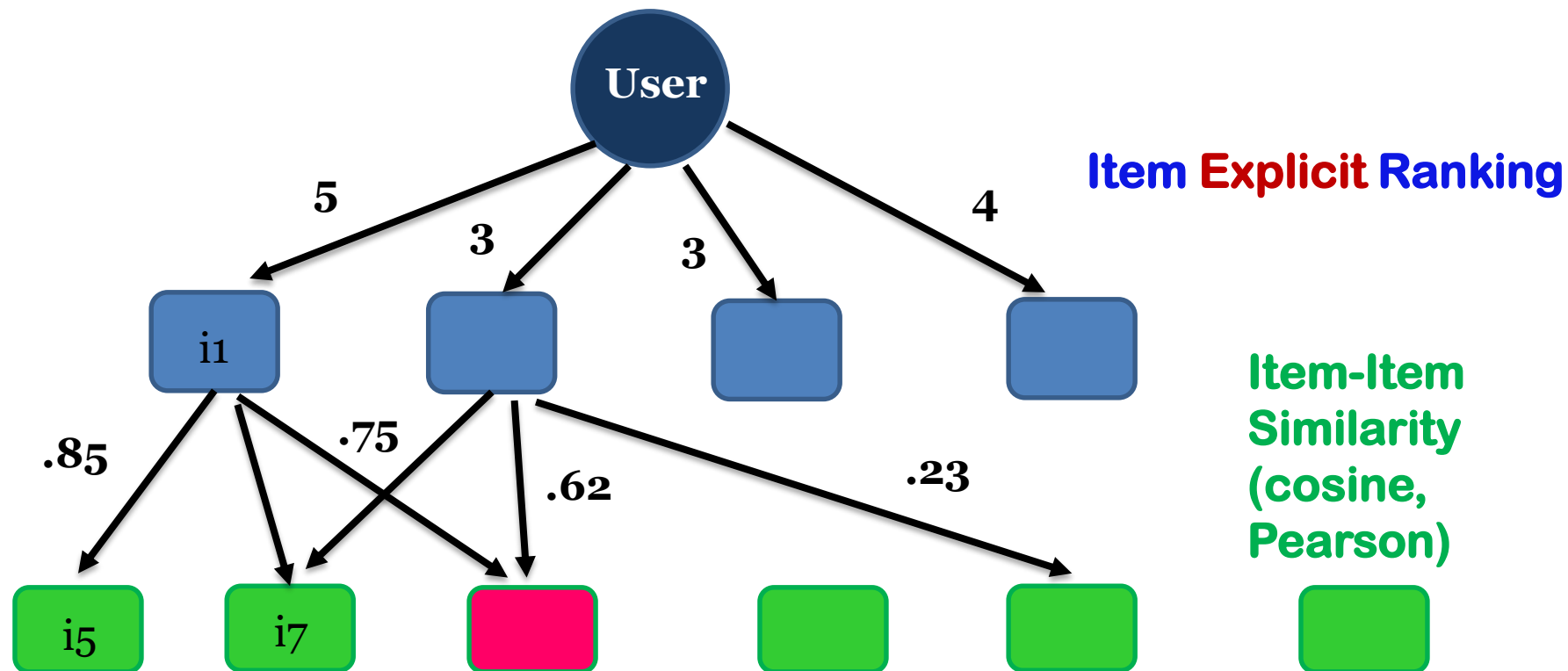
# Explicit vs. Implicit Ranking

---

- **Explicit Feedback**
  - **Feedback that users directly report on their interest in products**
    - e.g. star ratings for movies
    - e.g. thumbs-up/down for TV shows
- **Implicit Feedback**
  - **Feedback that indirectly reflects opinion through observing user behavior**
    - e.g. purchase history, browsing history, or search patterns

# Item-Based Collaborative Filtering (IBCF)

## Recommender System Algorithm with **Explicit** Rankings



**Item\_Rank = Average Weighted Sum**

# Various Forms of Rankings

---

- The **goal** of a ranking system is to find the best possible ordering of a set of items for a user, within a specific context, in real-time.
- **To optimize consumption:**
  - Rank based on *item popularity*: on average, a user is most likely to like what most others like
  - But, *popularity* is the opposite of *personalization*
- **To optimize personalization**
  - Goal: to find a personalized ranking function that is better than item popularity to better satisfy users with varying tastes.
  - Goal: to recommend items that each user is most likely to enjoy.

# Personalization Approach

---

- First, ask users to rate a few titles they have read in the past in order to build a rating prediction component.
- Then, use the user's predicted rating of each item as an adjunct to item popularity.
- Using predicted ratings on their own as a ranking function can lead to items that are too niche or unfamiliar, and can exclude items that the user would want to watch even though they may not rate them highly.



# Combining Popularity w/ Personalization

---

- To compensate for this, rather than using either popularity or predicted rating on their own, produce rankings that balance both of these aspects: build a ranking prediction model using these two features.

# What is the Goal behind Recommendation

- **What User response is a good response to a recommendation?**
  - **What are you trying to optimize when you build a RS?**
    - Not only to click but get engaged for a long time
    - The amount of time that the User engages with the service: this correlates with the business objective of retaining users over time
    - Hence, optimize over a long period of time
- **Rating prediction, or preference for each item:**
  - How much does it rely on explicit user feedback & user feedback quality? Recent trends: less explicit feedback and of the worse quality
  - Instant streaming: if you do not like smth, you will not stop to give negative feedback, you will just quickly switch to smth else
  - Assume that your actions and what you do are informing the system
  - An account that represents the whole household (not just a single user): mix of feedbacks from the household
  - Even worse: adult intentionally rate kids' content poorly so that it goes away; plus kids tend to be bi-modal in their ratings: too high or too low
  - But ton of implicit feedback

# Goal of RS

---

- **Optimizing for next rating prediction: what is the next things you will be rating highly versus:**
  - **Rank highly documentaries but only watch them 10% of the time but 90% of the time you will watch Comedy even though your rating is not very high:**
    - Rating is low but it is entertaining and you spend lots of time on this entertaining

# Ranking Problem in the Abstract Sense

---

- **Keep growth popularity as your baseline; good starting point when you are starting a ranking problem**
- **What can you add to this popularity to make it more personalized?**
  - What do you have at hand to help personalize the ranking system?
  - Take it as an input feature to a good ranking algorithm
  - Two features: popularity ranking and personalized ranking
  - Learning to rank: is how to learn which weights to give to each feature (what is relative merit of each of these two features: popularity vs personalized rating)
- **How to learn these weights?**
  - Treat it a classification problem: point-wise approach to learning to rank and use logistic regression (where by using a logit problem you convert a linear regression problem into a classification problem)
    - Cons: you are optimizing a log-likelihood, which is not a ranking metric
    - What you really want to do is to optimize some ranking metric that really represents your problem

---

Matrix Factorization

**ALTERNATE LEAST SQUARES**