Data Mining & Machine Learning

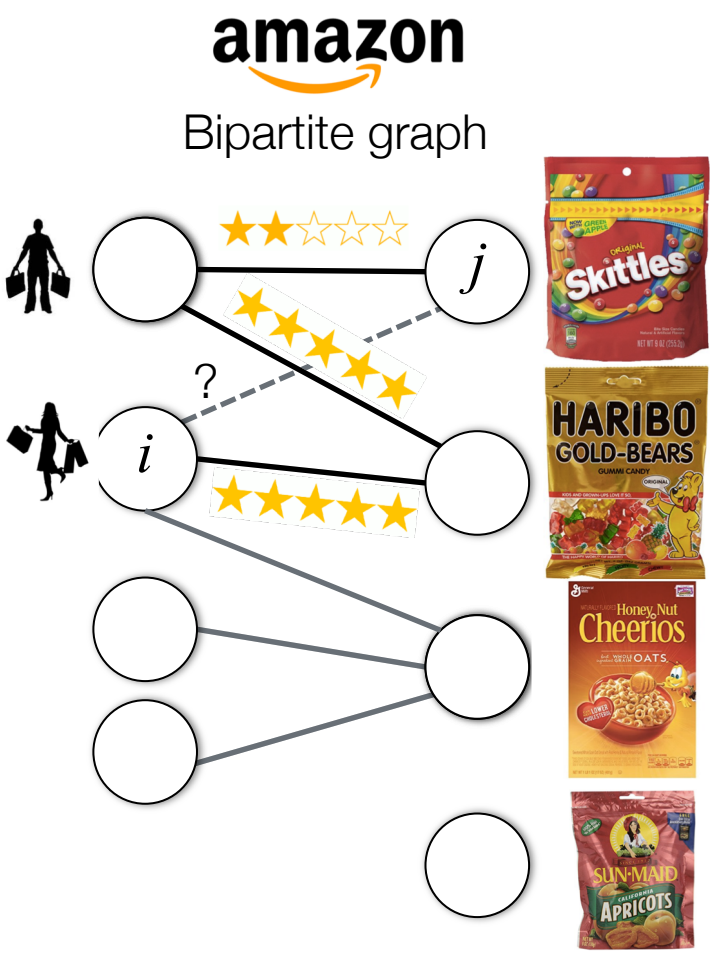
CS57300
Purdue University

April 3, 2018

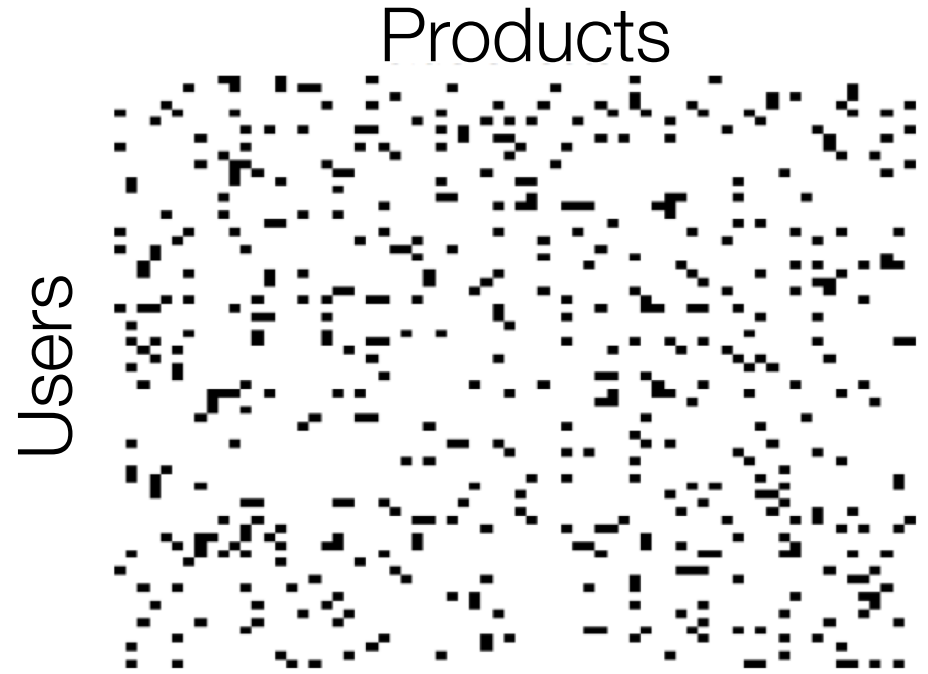
Collaborative Filtering

Product Recommendation

Example of collaborative filtering application.
Product recommendation based on user ratings



X is an $n \times m$ matrix.



$X_{i,j}$ - Rating user i gives to product j

Rating Matrix

X is an $n \times m$ matrix.

	b_1	b_2	b_3	b_4	b_5	b_6
a_1	x_{11}	x_{12}	?	?	x_{15}	x_{16}
a_2	?	?	?	x_{24}	?	?
a_3	?	x_{32}	?	?	?	?
a_4	?	?	x_{43}	?	?	x_{46}
a_5	?	?	?	?	x_{55}	?

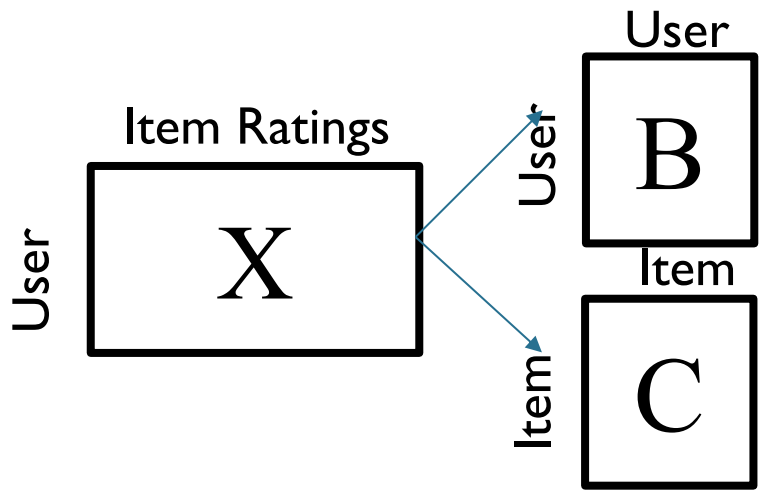
- Description of rating matrix should be invariant to permutation
  - Order of users (a) and products (b) in the matrix carries no information
- Absence / presence of rating is a signal
  - But means two things:
    (a) user has never seen the product
    (b) user does not want to rate the product
- Can be extended to graphs with vertex attributes

Simplest Idea: Recommendation via Similarity

- (user,user) similarity to recommend items
  - $B = XX^T$/<normalization factor>
  - good if item base is smaller than user base
  - good if item base changes rapidly

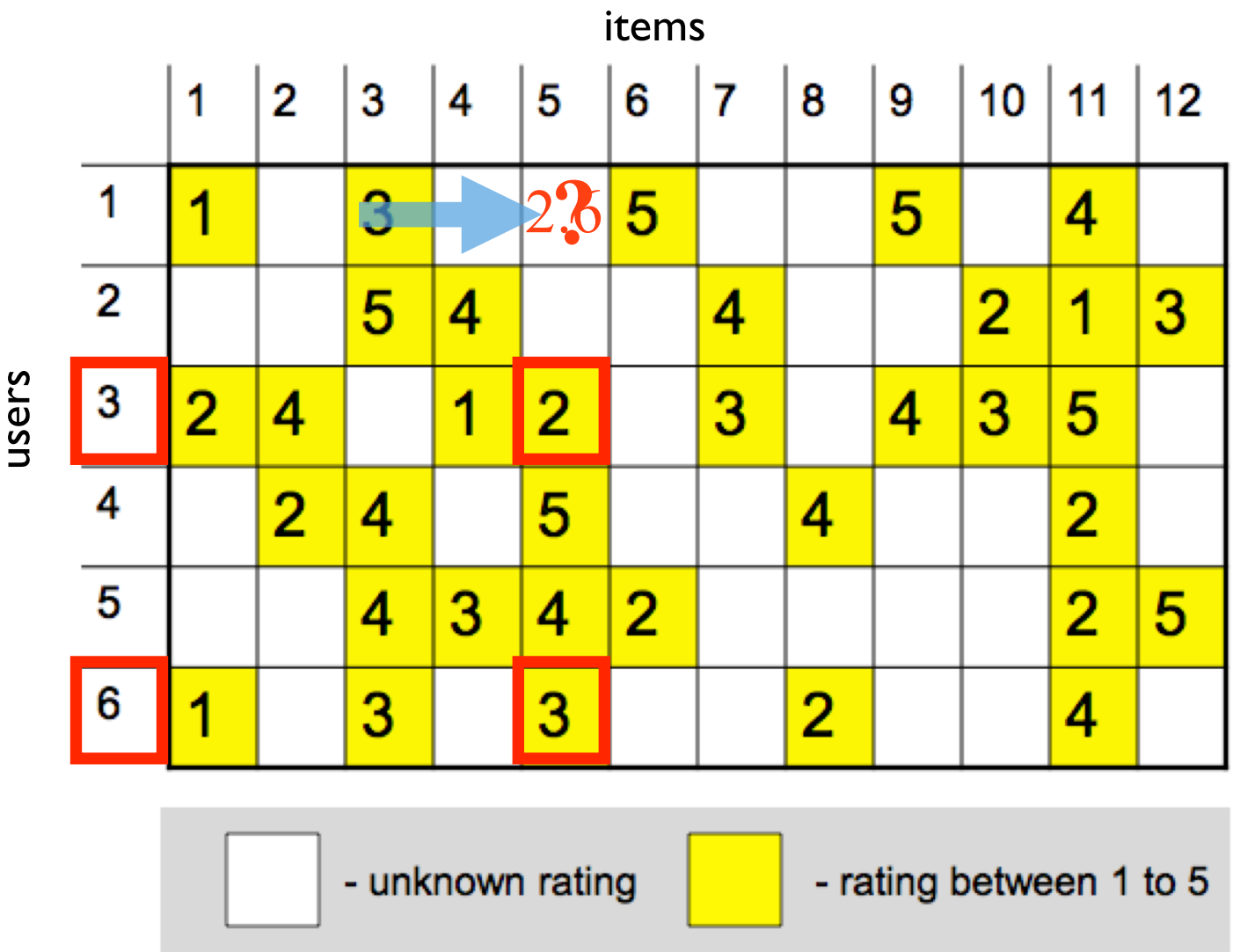
- (item,item) similarity to recommend new items that were also liked by the same users
  - $C = X^TX$ /<normalization factor>
  - good if the user base is small

- Oldest known collaborative filtering method



Recommendations Can be Based on Similarity

Heuristic: $X_{i,j} \approx \frac{\sum_{i,b} B_{i,b} X_{b,j}}{\sum_{i,b} B_{i,b}}$



User-user similarity

$B_{13} = 0.2$

$B_{16} = 0.3$

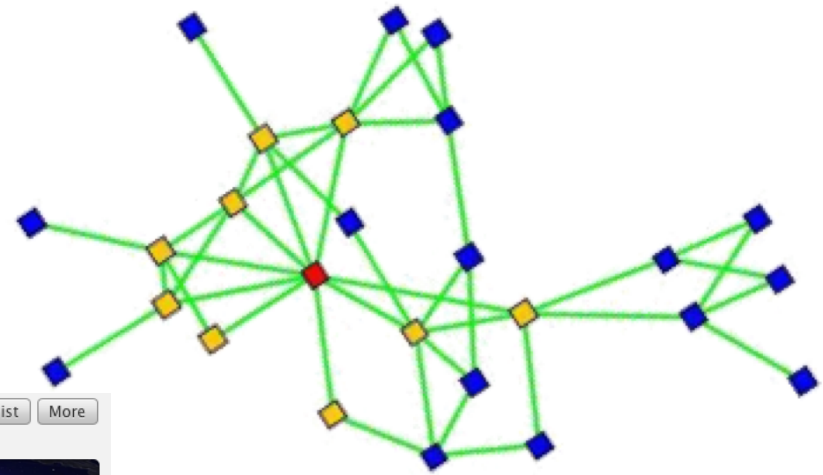
weighted average

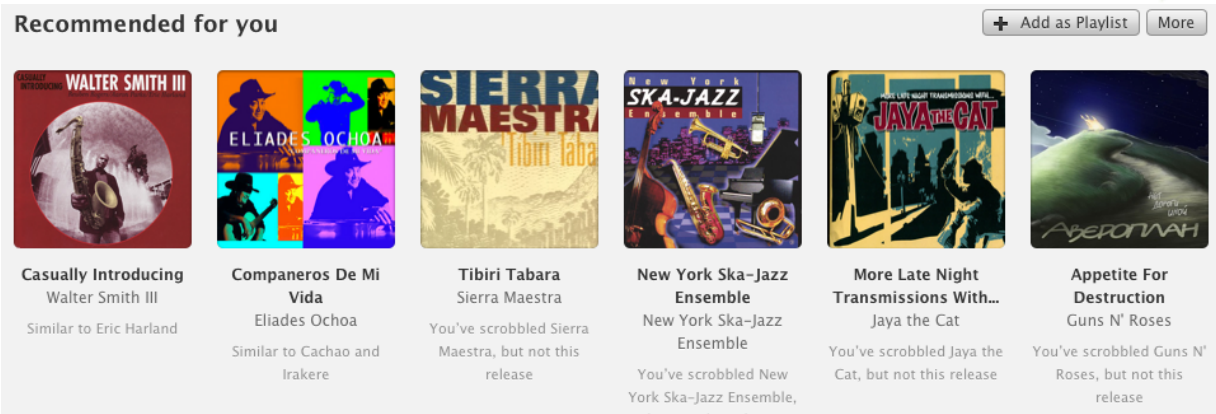
$$\frac{0.2 \cdot 2 + 0.3 \cdot 3}{0.2 + 0.3} = 2.6$$

Ack: A. Smola 2012

Properties

- Intuitive
- No (substantial) training
- Handles new users / items
- Easy to explain to user





- Accuracy & scalability questionable

We will now propose something closer to the state-of-the-art
(it is secretly a probabilistic model)

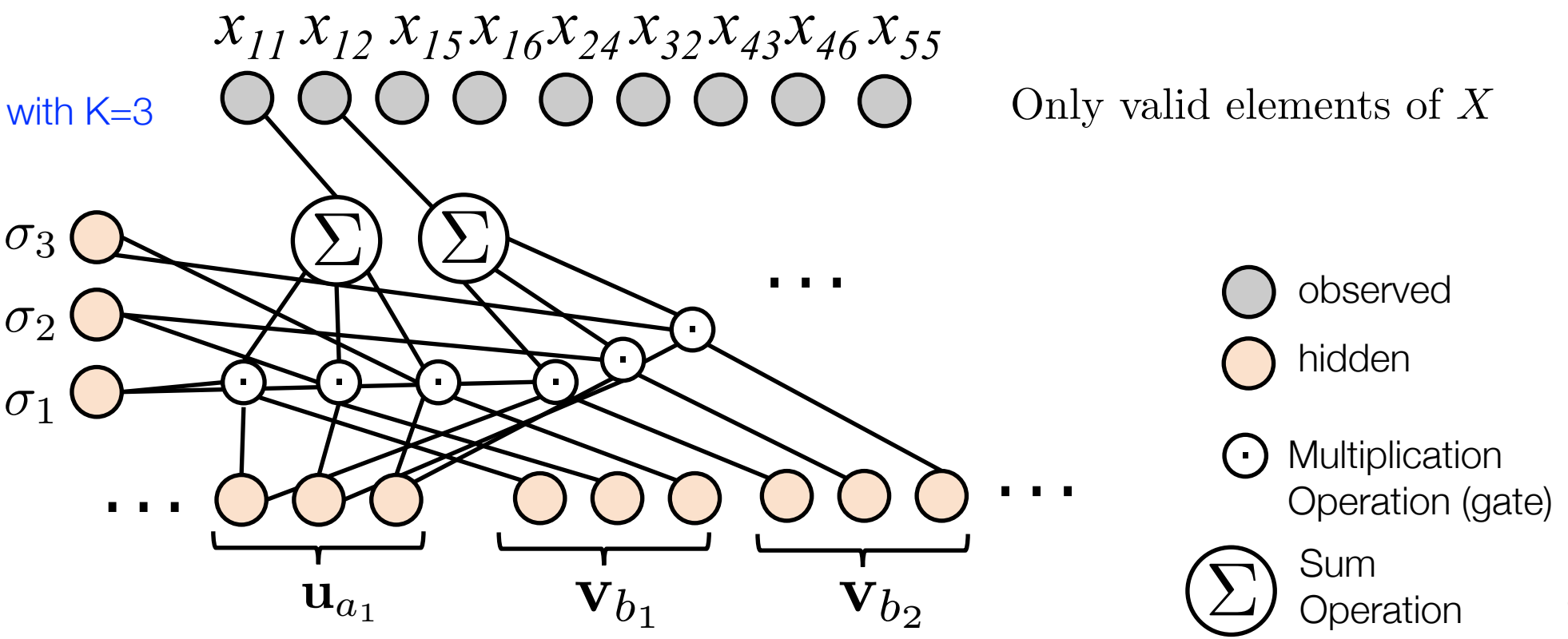
A Rating Matrix Model

- Assume X has is a set of random variables organized in a matrix (observable)
  - The model has the following parameters

$\mathbf{v_j}$ is a $1 \times K$ vector of latent factors of product j

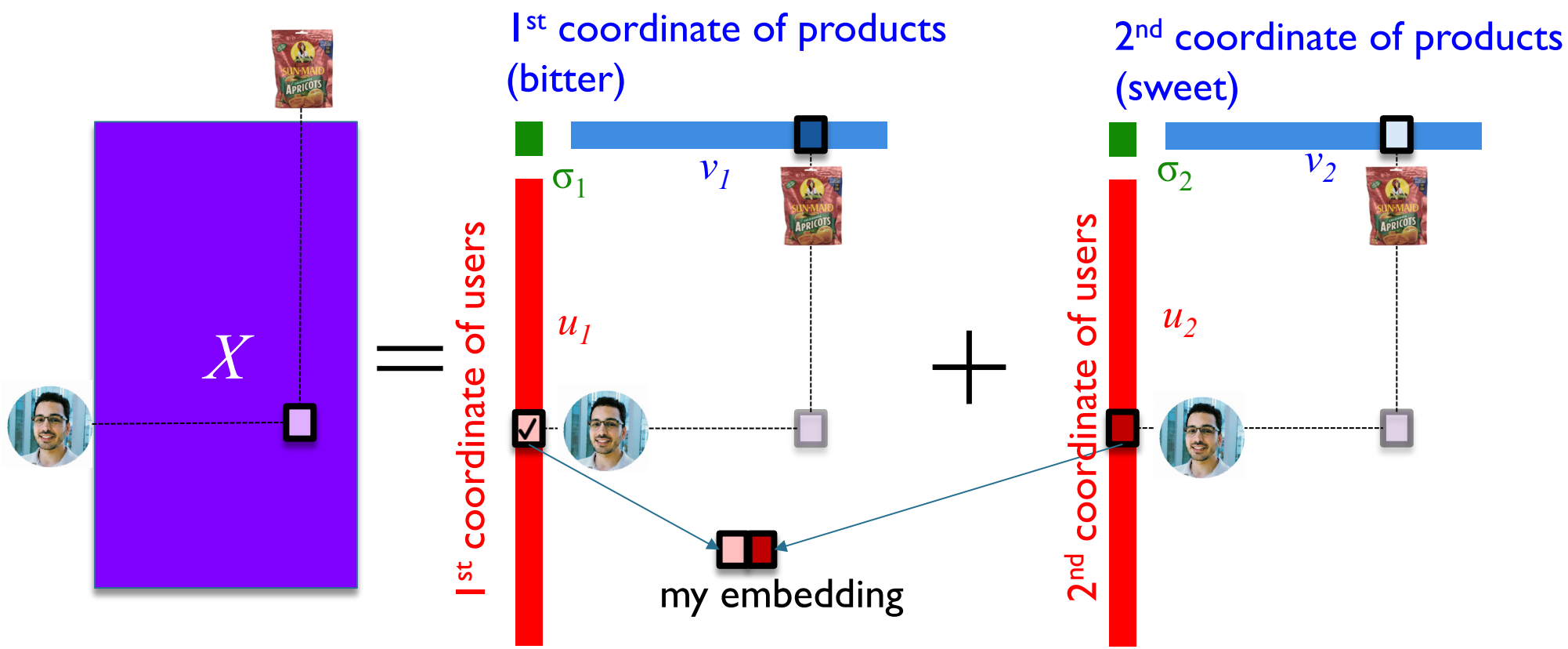
$\mathbf{u_i}$ is a $1 \times K$ vector of latent factors of user i

σ_k is weight of dimension $k = 1, \ldots, K$



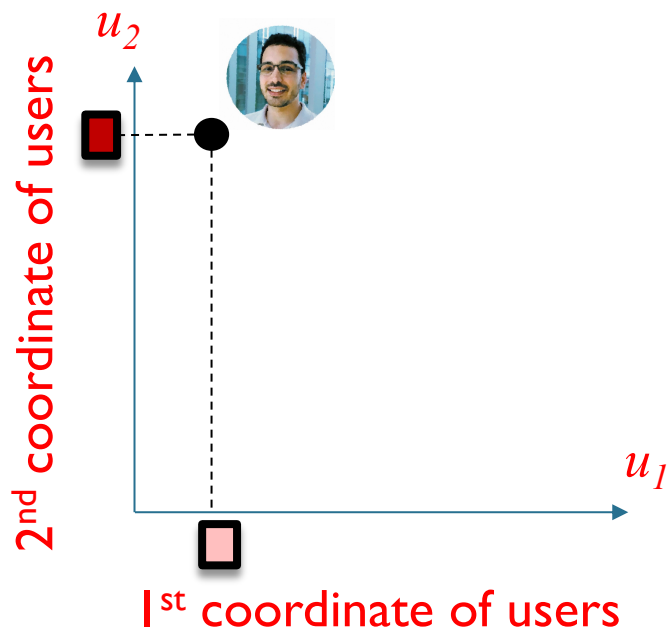
Alternative View: Matrix Decomposition

Illustration of a rank-K matrix approximation (assuming rank($\mathbf{X}$) < K) :



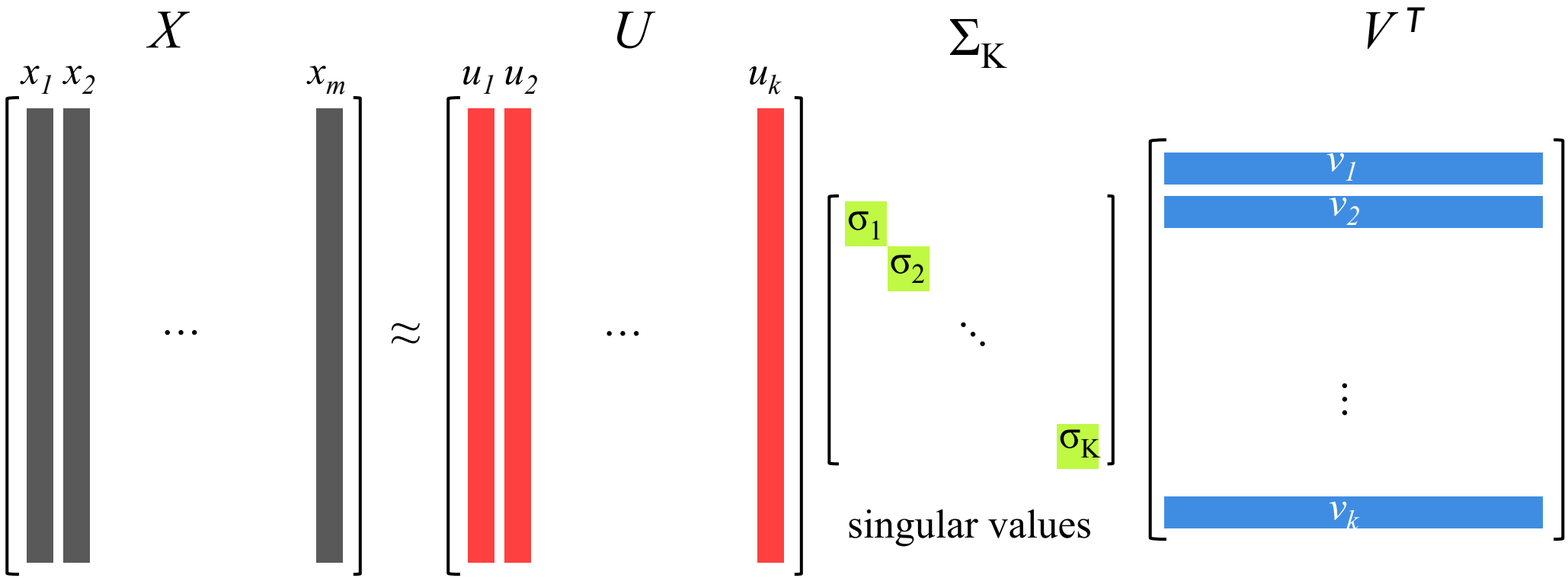
$$X \approx U\Sigma V^T = \sum_{k=1}^{K} \sigma_k u_k v_k^T$$

My Embedding (Representation as Latent Variables)



Singular Value Decomposition

$$\mathbf{X} \approx \mathbf{U}\Sigma_K\mathbf{V}^T$$



Data

Left singular vectors
(Columns are Orthogonal and Normalized)

Right singular vectors
(Columns are Orthogonal and Normalized)

Understanding SVD singular vectors

- Let K = rank($\mathbf{X}$)
- As U and V have *orthonormal** columns

**orthonormal* = orthogonal with norm 1

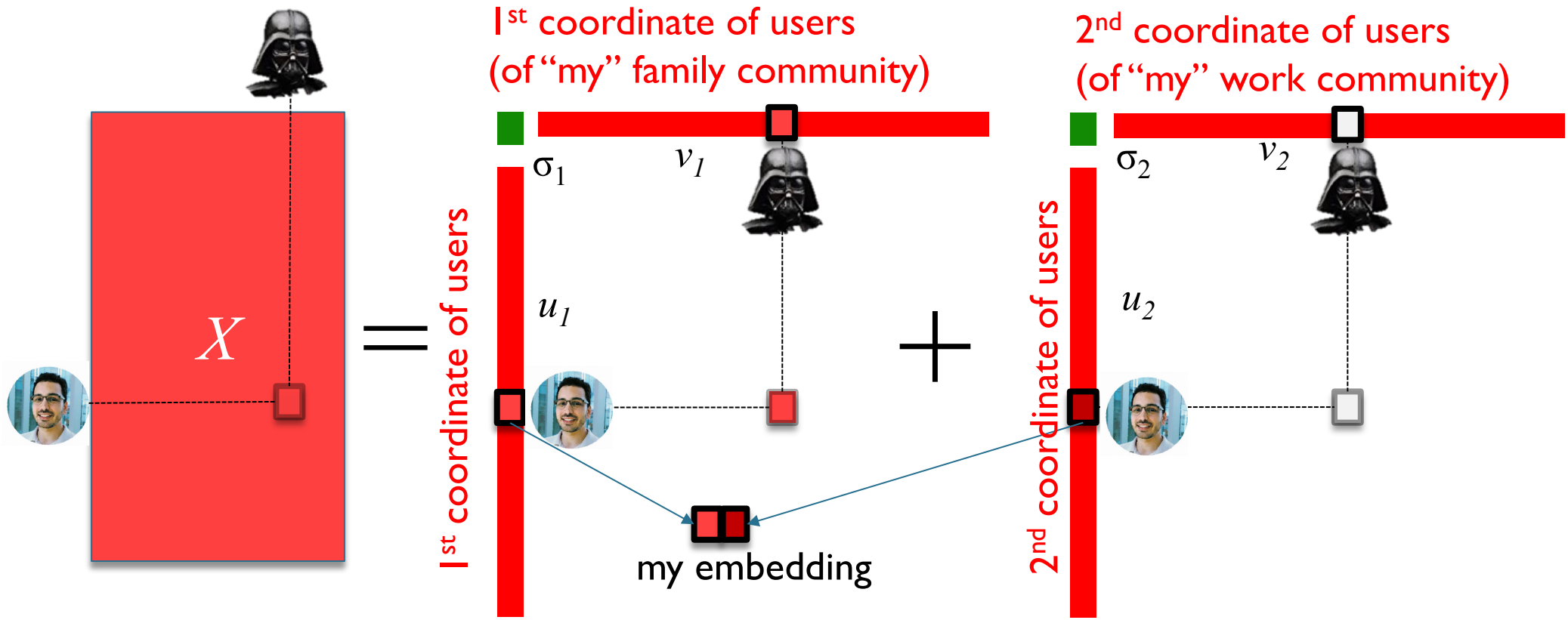
$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^2 V^T$$

$$X X^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma^2 U^T$$

What do V and U represent?
(hint, slide 5)

Undirected Network Embedding

- Illustration of a rank-2 approximation:

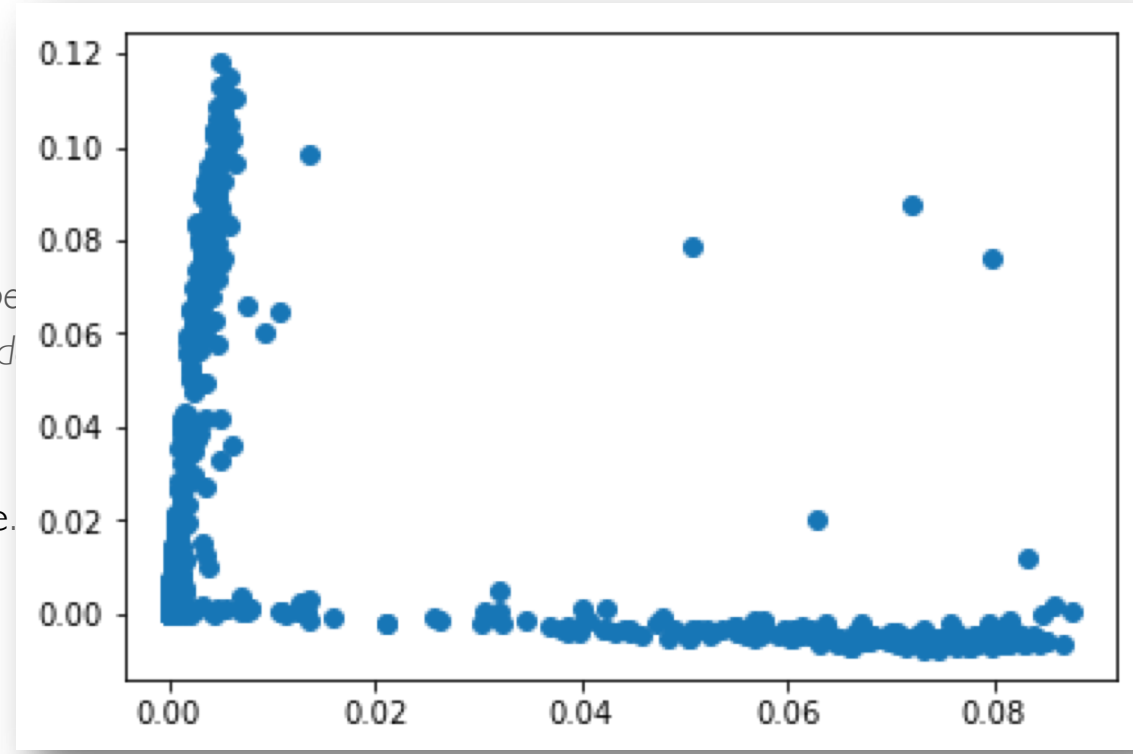


$$X \approx U\Sigma U^T = \sum_{k=1}^{K} \sigma_k u_k u_k^T$$

iPython Notebook

```
from sklearn.decomposition import TruncatedSVD
import matplotlib.pyplot as plt
import numpy as np

maxi = 0
maxj = 0
Xlist = []
# data at https://www.cs.purdue.edu/homes/ribe
# Original data from http://snap.stanford.edu/d
with open("FBmatrix.csv") as f:
        for line in f.readlines():
                i,j,Xij = list(map(int, line.
                maxi = max(i,maxi)
                maxj = max(j,maxj)
                Xlist.append((i,j,Xij))
X = np.zeros((maxi+1, maxj+1))
for i,j,Xij in Xlist:
                X[i,j] = Xij
SVD = TruncatedSVD(n_components=2, n_iter=10)
SVD.fit(X)
# 2D embedding of FB egonet 1912
plt.plot(SVD.components_[0], SVD.components_[1], 'o')
plt.show()
```



Creates the adjacency matrix

2 dimensions

Finds latent embedding

Plots the embedding

SVD: Optimization View

- Objective Function does not need Σ :

Row vectors of U an V

Regularization

$$\underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \sum_{i,j} (X_{i,j} - (\mathbf{u}_{i,\star})(\mathbf{v}_{j,\star})^T)^2 + \lambda(\|\mathbf{U}\|_2^2 + \|\mathbf{V}\|_2^2)$$

- Can be easily implemented via alternating least squares

- What is the statistical assumption about $\mathbf{X}$?
  What does the L_2 minimization imply?
  - Conditionally independent, Normally distributed ratings

$$X_{i,j} \sim \text{Normal}((\mathbf{u}_{i,\star})(\mathbf{v}_{j,\star})^T)^2, 1)$$

  - But ratings are not Normal!
  - Ratings $1,\ldots,5$ or binary { , }

Poisson Matrix Factorization

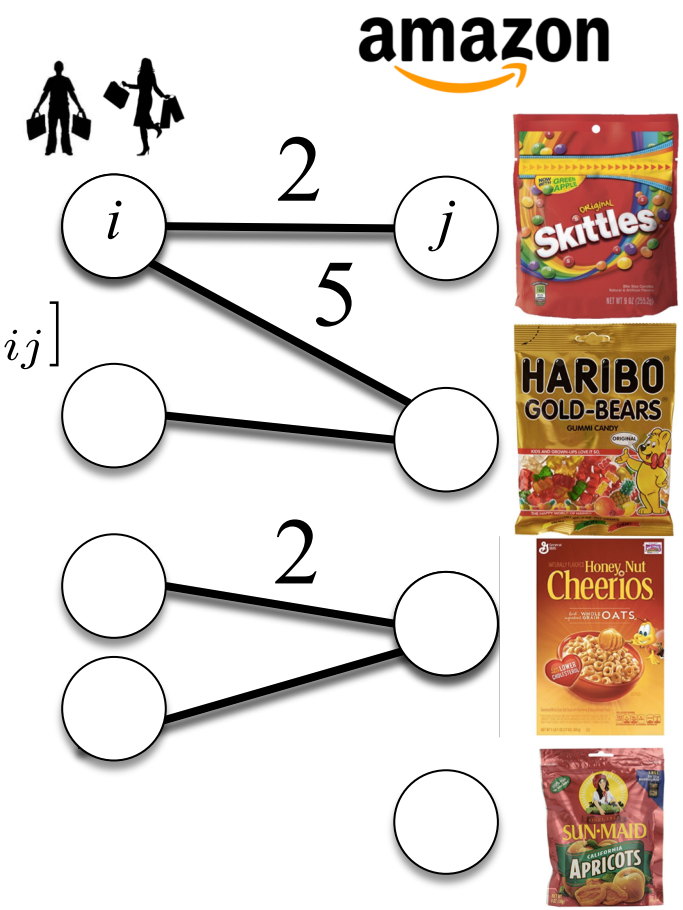
Users have a propensity rate to buy certain category of product ($\mathbf{W}$)

In a category some products have a propensity rate to be bought ($\mathbf{H}$)

$$\mathbf{X}_{ij} \sim \text{Poisson}([\mathbf{WH}]_{ij})$$

$$\text{MLE} \rightarrow \underset{\mathbf{W},\mathbf{H}}{\text{argmax}} \sum_i \sum_j [X_{ij} \log(\mathbf{WH})_{ij} - (\mathbf{WH})_{ij}]$$

(Poisson Matrix Factorization,
Non-negative matrix factorization)

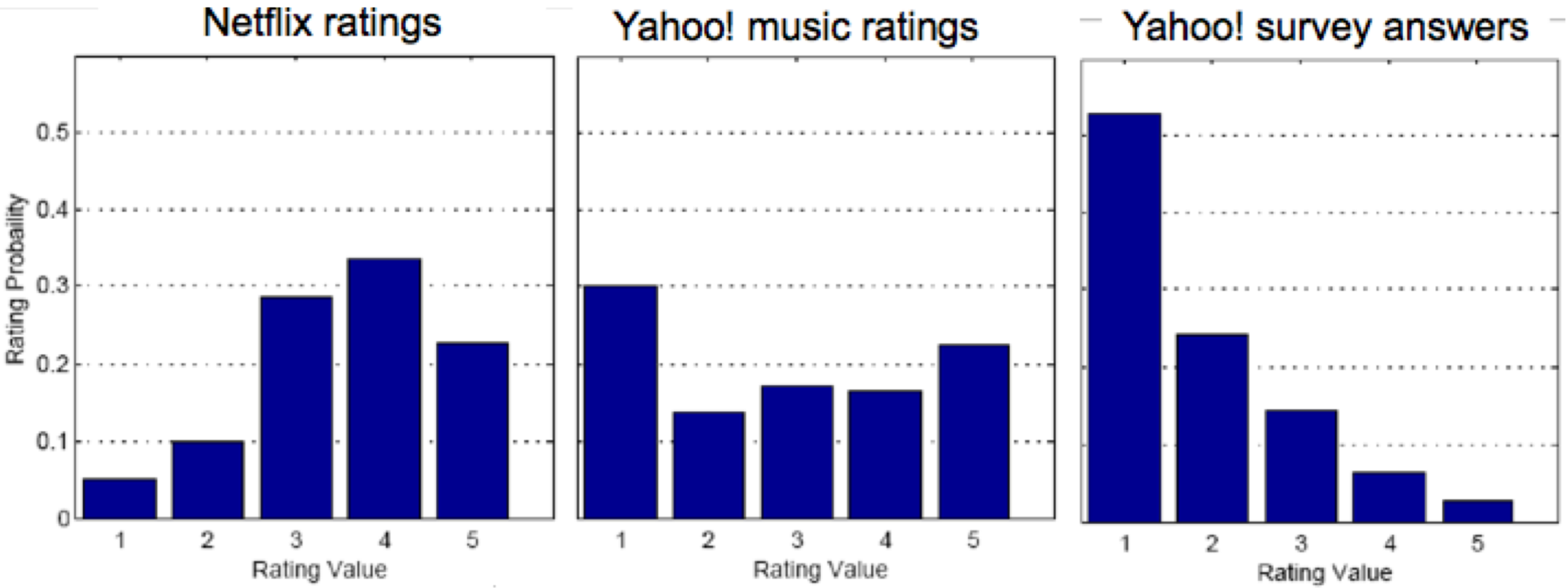


Data Quality Problems

- Amatriain et al. (2009): Asking users to **rate** items **again** reduce prediction error by as much as 14%

- Marlin et al. (2007): most existing algorithms based on the flawed assumption that missing ratings (data) are missing at random (uniformly)

Ratings are not given at random



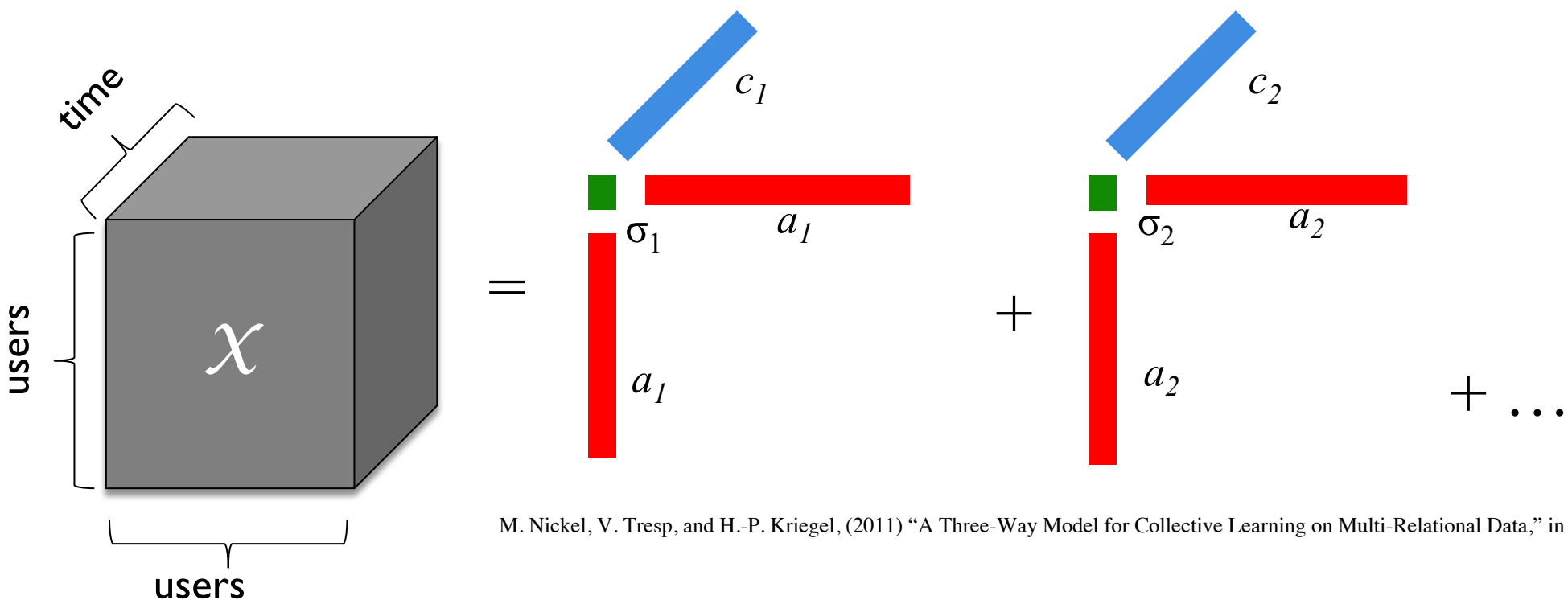
- Marlin et al. "Collaborative Filtering and the Missing at Random Assumption" UAI 2007

Ack: A. Smola 2012

Beyond Matrices

Adding Node/Edge Covariates is Generally Easy

- E.g.: RESCAL tensor factorization (a generalization of matrix factorization) (Nickel et al. 2011)



M. Nickel, V. Tresp, and H.-P. Kriegel, (2011) "A Three-Way Model for Collective Learning on Multi-Relational Data," in *ICML*