CS 273A: Machine Learning

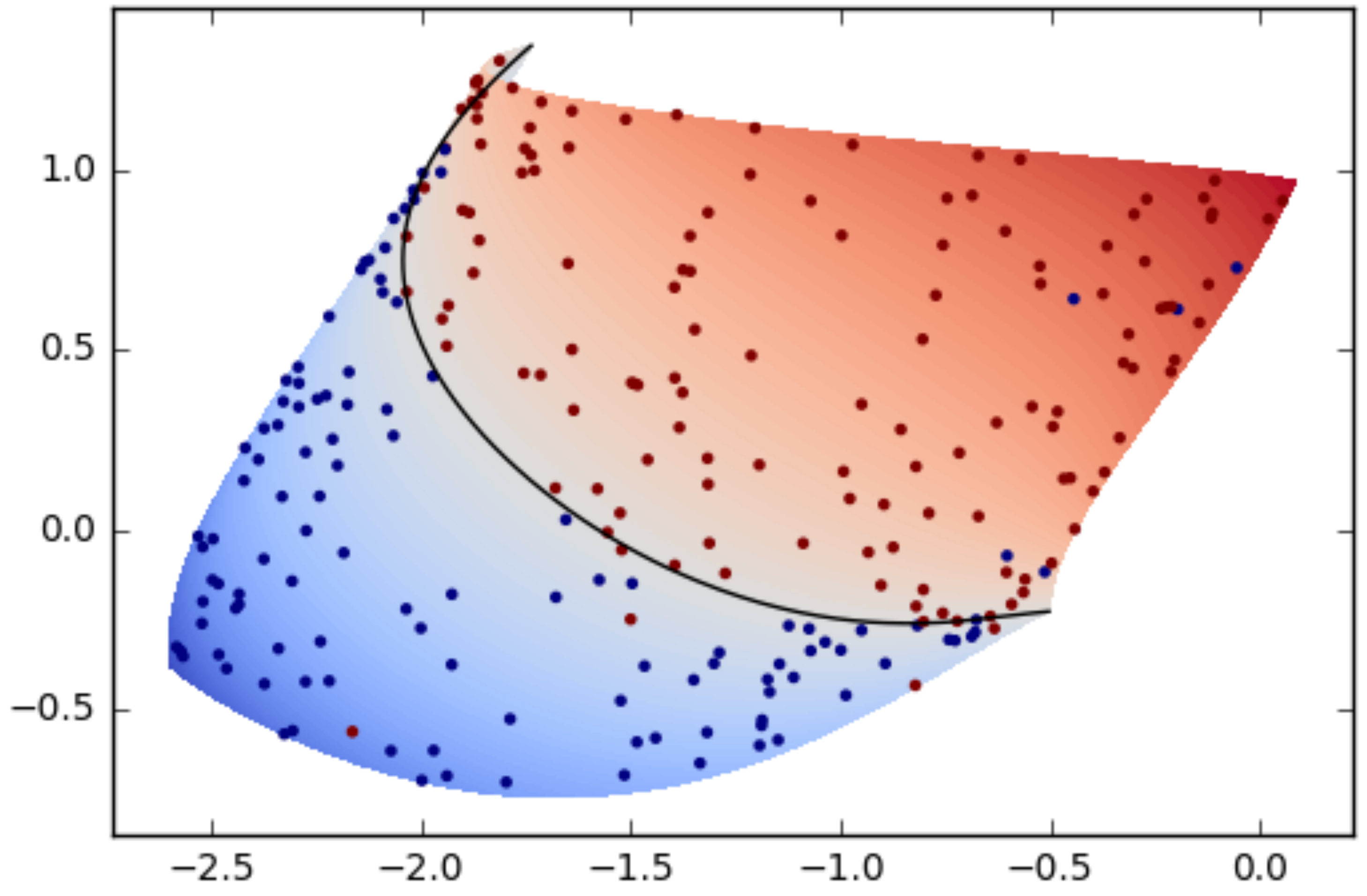
Winter 2021

Lecture 16: Latent-Space Models

(cont.)

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

All slides in this course adapted from Alex Ihler & Sameer Singh



Logistics

assignments	• Assignment 5 due Thursday
project	• Final report due next Thursday
evaluations	• Evaluations due end of next week
final exam	• Review: next Thursday • Final: Thursday, March 18, 1:30–3:30pm

Today's lecture

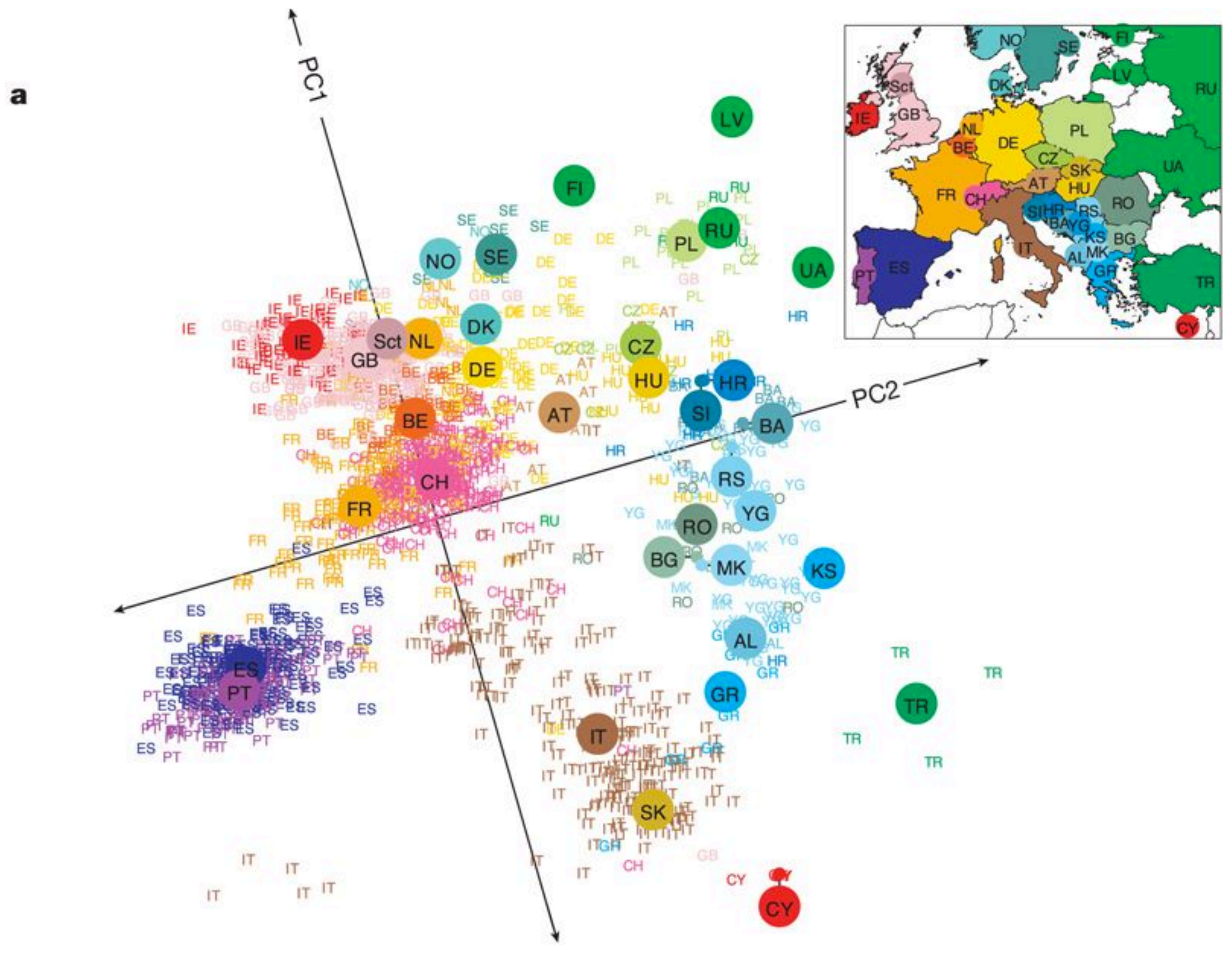
Eigen-faces

Latent Semantic Analysis

Collaborative Filtering

Latent-space representations: uses

- Remove unneeded features
  - Features that add very little information (e.g. low variability, high noise)
  - Features that are similar to others (e.g. almost linearly dependent)
  - Reduce dimensionality for downstream application
    - Supervised learning: fewer parameters, need less data
    - Compression: less bandwidth



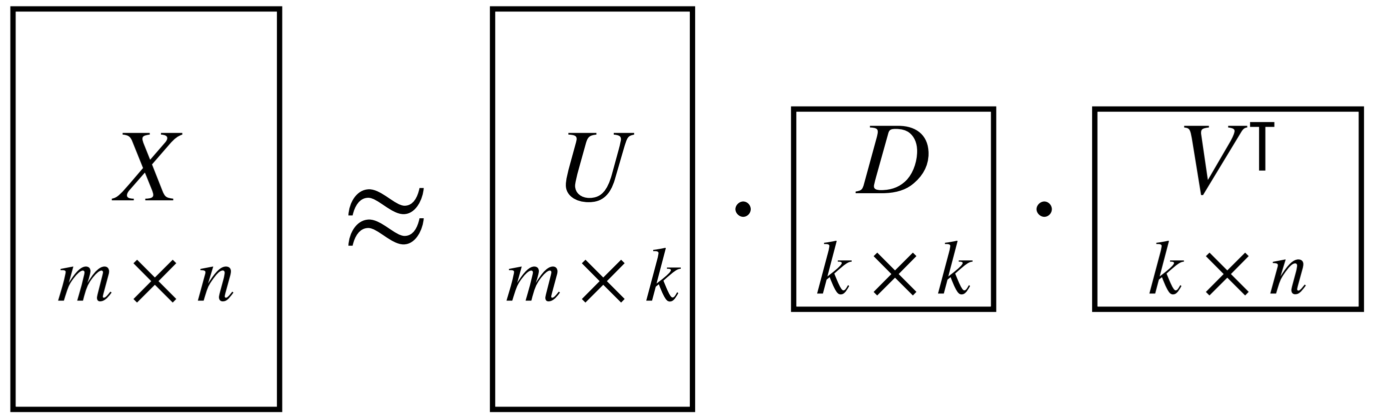
- Can also add features
  - Summarize multiple features into few cleaner / higher-level ones

PCA: applications

- Eigen-faces
  - Represent image data (e.g. faces) using PCA
- Latent-Space Analysis (topic models)
  - Represent text data (e.g. bag of words) using PCA
- Collaborative Filtering for Recommendation Systems
  - Represent sentiment data (e.g. ratings) using PCA

Singular Value Decomposition (SVD)

- Alternative method for finding covariance eigenvectors
  - Has many other uses
- Singular Value Decomposition (SVD): $X = UDV^\intercal$
  - U and V (left- and right singular vectors) are orthogonal: $U^\intercal U = I$, $V^\intercal V = I$
  - D (singular values) is rectangular-diagonal
  - $\Sigma = X^\intercal X = VD^\intercal U^\intercal UDV^\intercal = V(D^\intercal D)V^\intercal$



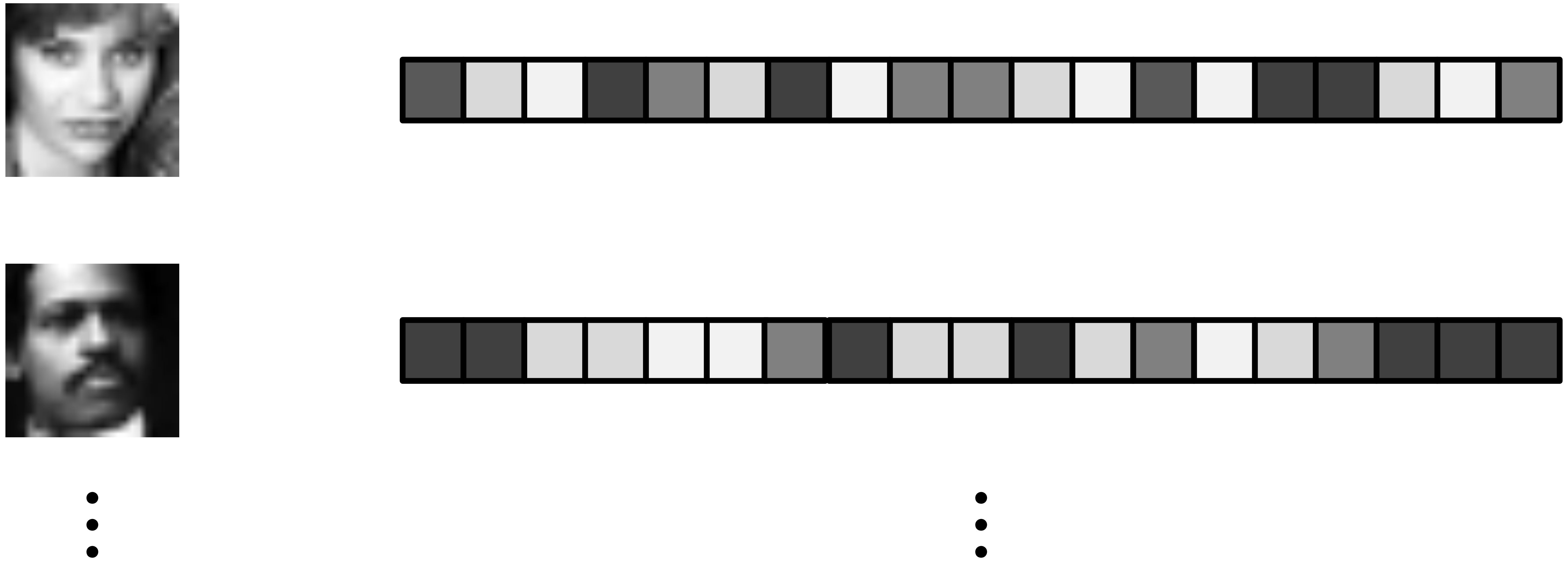
$$\underset{m\times n}{X} \approx \underset{m\times k}{U} \cdot \underset{k\times k}{D} \cdot \underset{k\times n}{V^\intercal}$$

- UD matrix gives coefficients to reconstruct data: $x_i = U_{i,1}D_{1,1}v_1 + U_{i,2}D_{2,2}v_2 + \cdots$
  - We can truncate this after top k singular values (square root of eigenvalues)

Eigen-faces

$$\boxed{\underset{m \times n}{X}} \approx \boxed{\underset{m \times k}{U}} \cdot \boxed{\underset{k \times k}{D}} \cdot \boxed{\underset{k \times n}{V^\intercal}}$$

- “Eigen-X” = represent X using its principal components
- Viola Jones dataset: 24×24 images $\in \mathbb{R}^{576}$
    - Can represent vector as image

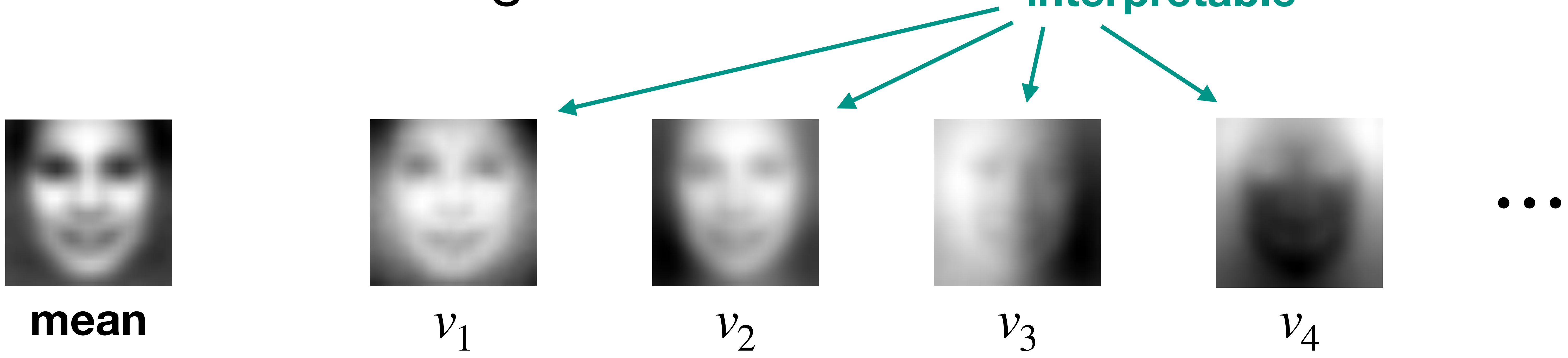


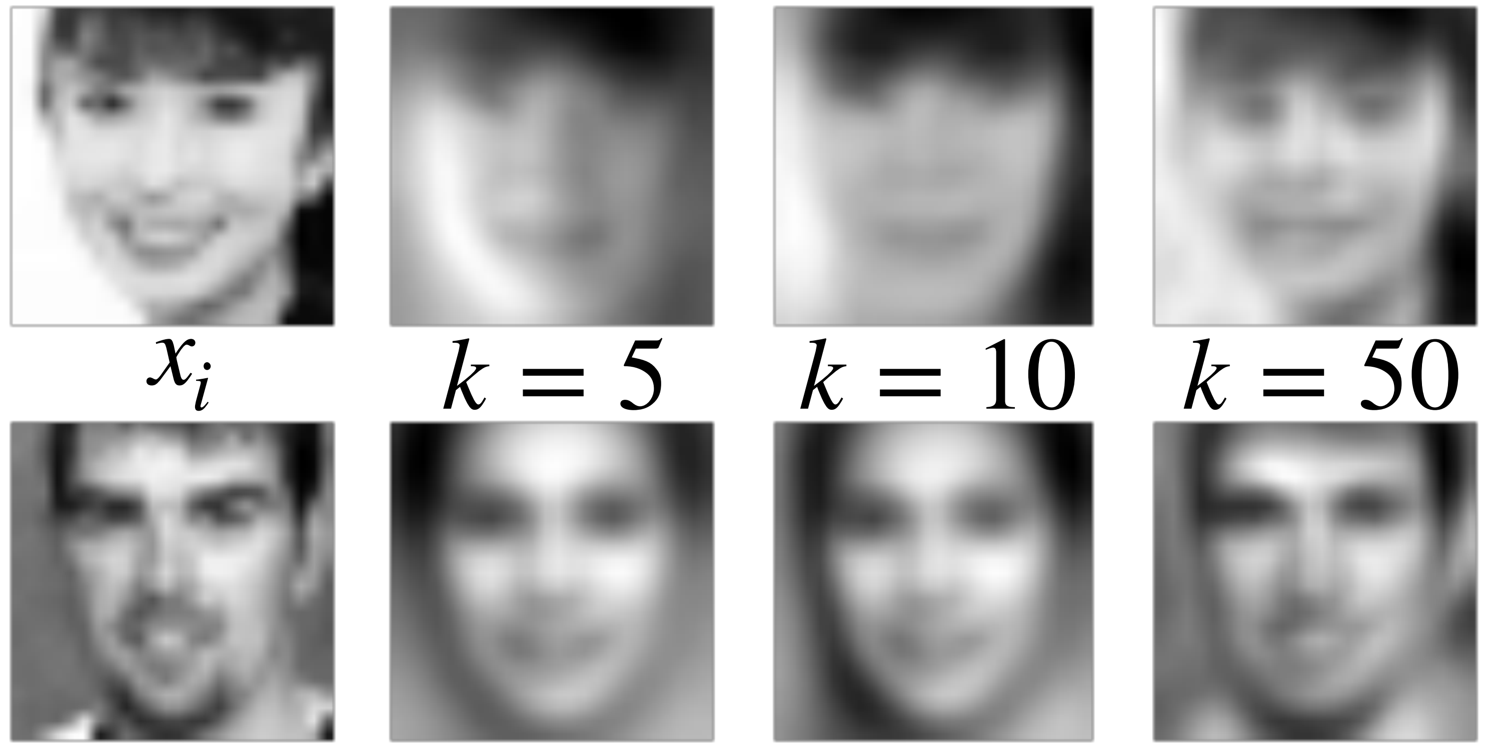
Eigen-faces

$$\underset{m \times n}{X} \approx \underset{m \times k}{U} \cdot \underset{k \times k}{D} \cdot \underset{k \times n}{V^\intercal}$$

- “Eigen-X” = represent X using its principal components
- Viola Jones dataset: 24×24 images $\in \mathbb{R}^{576}$
  - Can represent vector as image
  - Project data on k principal components

somewhat interpretable



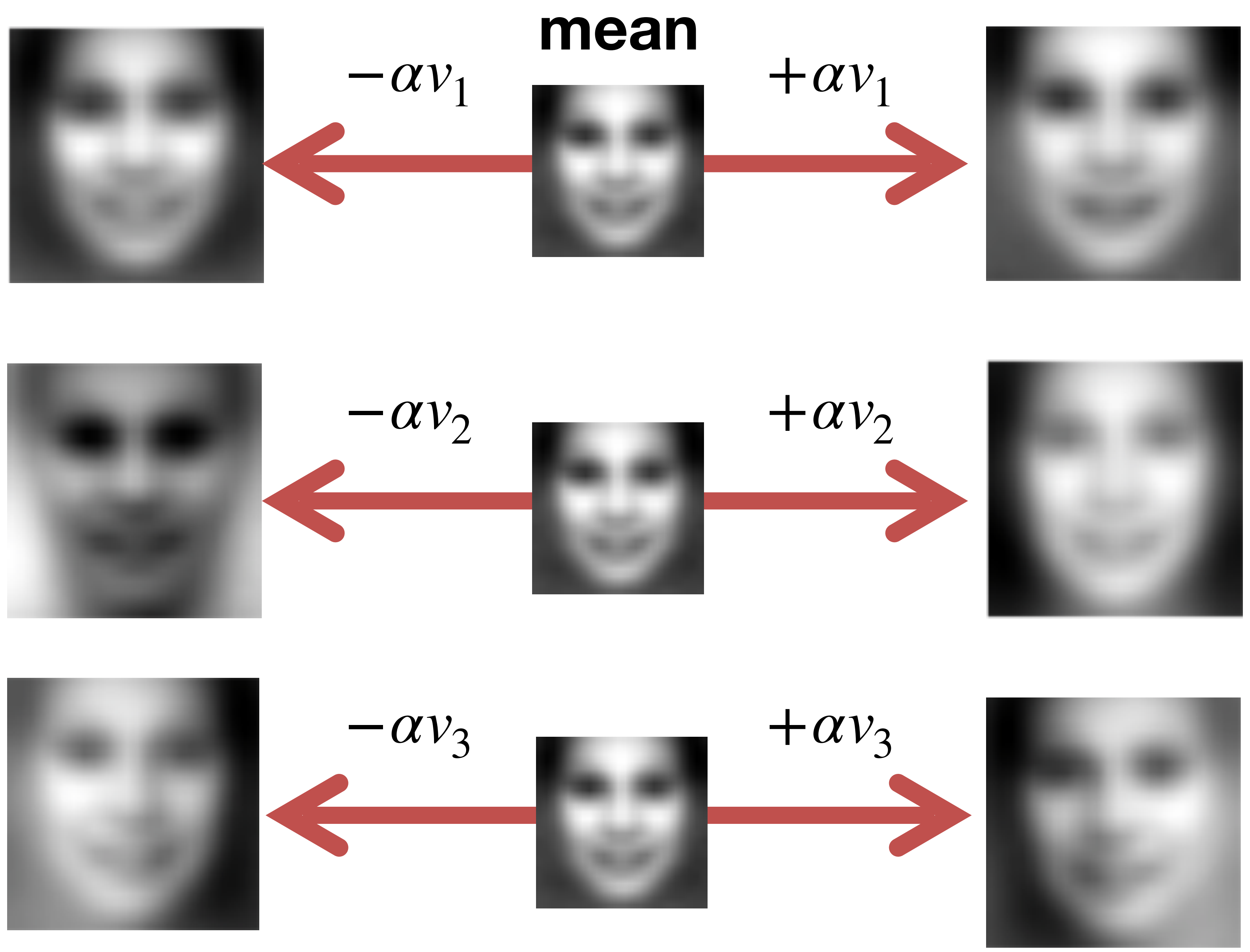


Eigen-faces

- “Eigen-X” = represent X using its principal components

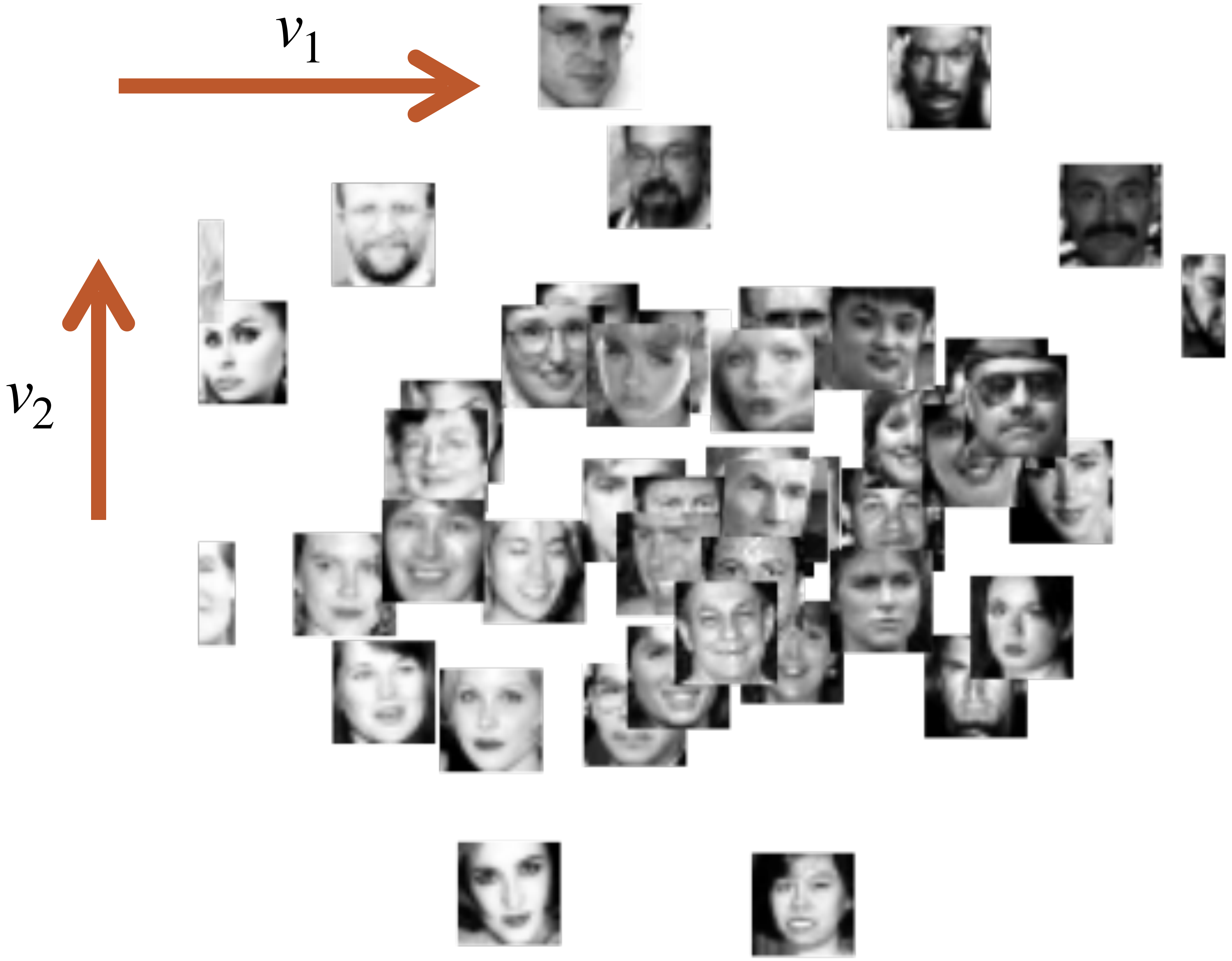
$$\boxed{\underset{m \times n}{X}} \approx \boxed{\underset{m \times k}{U}} \cdot \boxed{\underset{k \times k}{D}} \cdot \boxed{\underset{k \times n}{V^\intercal}}$$

- Viola Jones dataset: 24×24 images $\in \mathbb{R}^{576}$
  - Can represent vector as image
  - Visualize basis vectors v_i as $\mu \pm \alpha v_i$



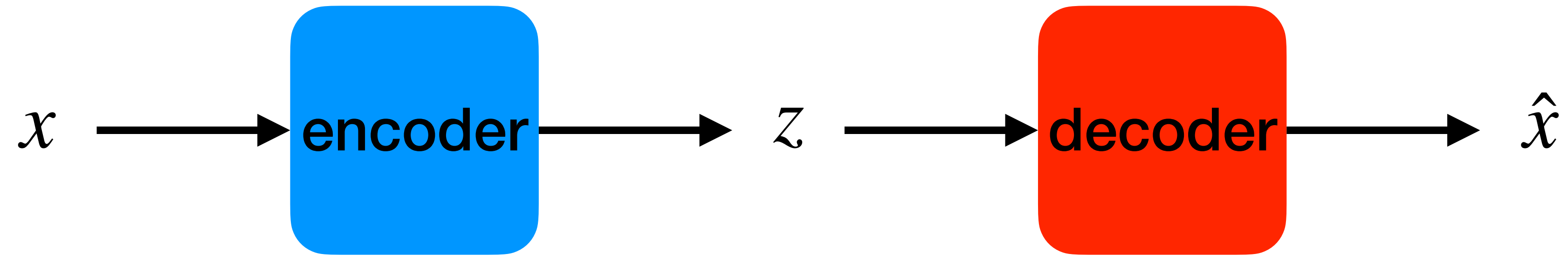
Eigen-faces

- “Eigen-X” = represent X using its principal components
- Viola Jones dataset: 24×24 images $\in \mathbb{R}^{576}$
  - Can represent vector as image
  - Visualize data by projecting onto 2 principal components

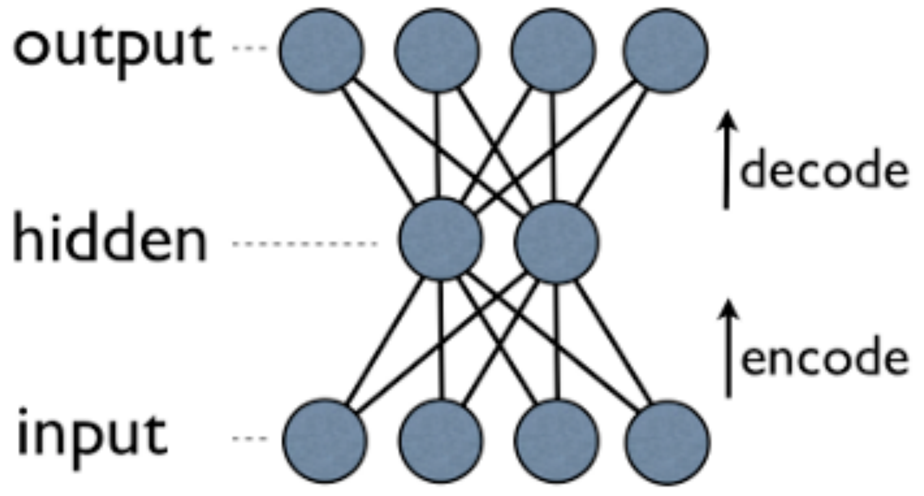


Nonlinear latent spaces

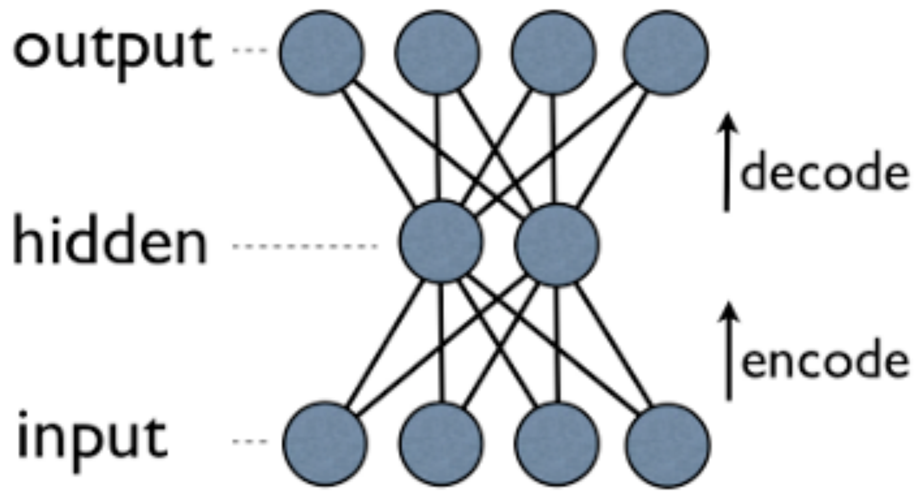
- Latent-space representation = represent x_i as z_i
    - Usually more succinct, less noisy
    - Preserves most (interesting) information on $x_i \Longrightarrow$ can reconstruct $\hat{x}_i \approx x_i$
    - Auto-encoder = encode $x \rightarrow z$, decode $z \rightarrow \hat{x}$



- Linear latent-space representation:
    - Encode: $Z = XV_{\leq k} = (UDV^\intercal V)_{\leq k} = U_{\leq k}D_{\leq k}$; Decode: $X \approx ZV^\intercal_{\leq k}$
- Nonlinear: e.g., encoder + decoder are neural networks
    - Restrict z to be shorter than $x \Longrightarrow$ requires succinctness



Variational Auto-Encoders (VAE)



- Probabilistic model:
  - Simple prior over latent space $p(z)$ (e.g. Gaussian)
  - Decoder = generator $p_\theta(x|z)$, tries to match data distribution $p_\theta(x) \approx \mathscr{D}$
  - Encoder = inference $q_\phi(z|x)$, tries to match posterior $q_\phi(z|x) \approx \dfrac{p(z)p_\theta(x|z)}{p_\theta(x)}$

Today's lecture

Eigen-faces

Latent Semantic Analysis

Collaborative Filtering

Textual features

- How to extract features from text for model inputs?
  - Idea: bag of words = indicate word count, not order

Rain and chilly weather didn't keep thousands of paradegoers from camping out Friday night for the 111th Tournament of Roses.

Spirits were high among the street party crowd as they set up for curbside seats for today's parade.

“I want to party all night,” said Tyne Gaudielle, 15, of Glendale, who spent the last night of the year along Colorado Boulevard with a group of friends.

Whether they came for the partying or the parade, campers were in for a long night. Rain continued into the evening and temperatures were expected to dip down into the low 40s.

Observed Data (text docs):

DOC #	WORD #	COUNT
1	29	1
1	56	1
1	127	1
1	166	1
1	176	1
1	187	1
1	192	1
1	198	2
1	356	1
1	374	1
1	381	2
	...	

VOCABULARY:
0001 ability
0002 able
0003 accept
0004 accepted
0005 according
0006 account
0007 accounts
0008 accused
0009 act
0010 acting
0011 action
0012 active
....

Topic models

- Word distribution is different for different document topics
  - Represent the bag-of-words feature vector in a latent space
  - Such that representation keep topic information

c1: Human machine interface for ABC computer applications
c2: A survey of user opinion of computer system response time
c3: The EPS user interface management system
c4: System and human system engineering testing of EPS
c5: Relation of user perceived response time to error measurement

Human–Computer Interfaces

m1: The generation of random, binary, ordered trees
m2: The intersection graph of paths in trees
m3: Graph minors IV: Widths of trees and well-quasi-ordering
m4: Graph minors: A survey

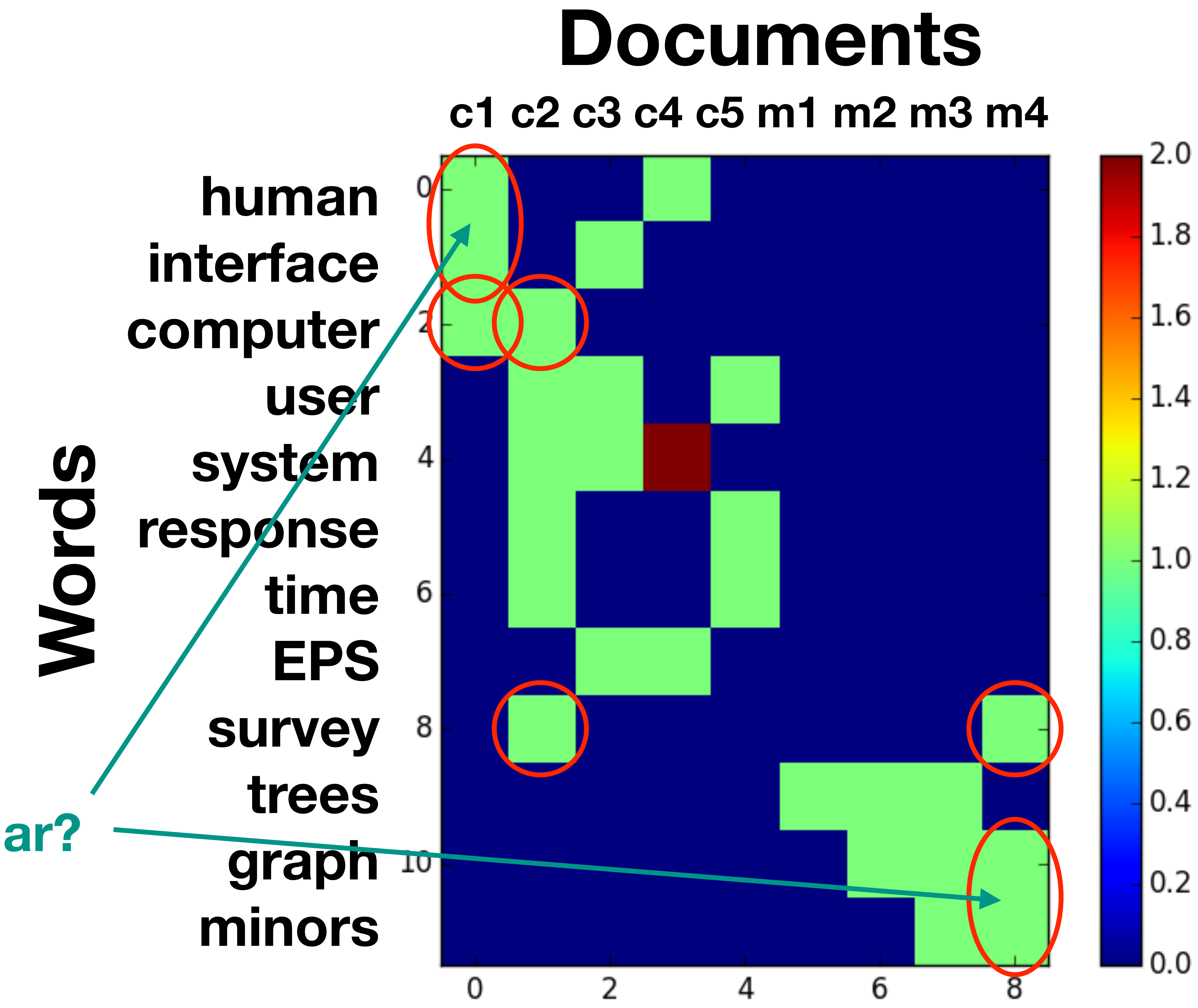
Graph Theory

Latent Semantic Analysis (LSA)

- Represent dataset as matrix of word counts: X_{ij} = # of word j in document i
  - Sparse matrix = mostly 0s
  - Typically normalize rows: X_{ij} = probability that random word in doc j is word i
    - Make documents of different length comparable
  - Typically don't shift or scale columns to have mean 0, var 1
- Perform PCA on X to get latent representation = top k components
  - Allows fuzzy search = documents of given topic, rather than word matching

Word–doc matrix: example

- Observation: many words have little overlap between the topics
- Typical sizes:
  - #docs = $D \sim 10^6$
  - #words in vocabulary = $W \sim 10^5$
  - Matrix size $\sim 10^{11}$, but only $\sim 10^8$ in sparse representation

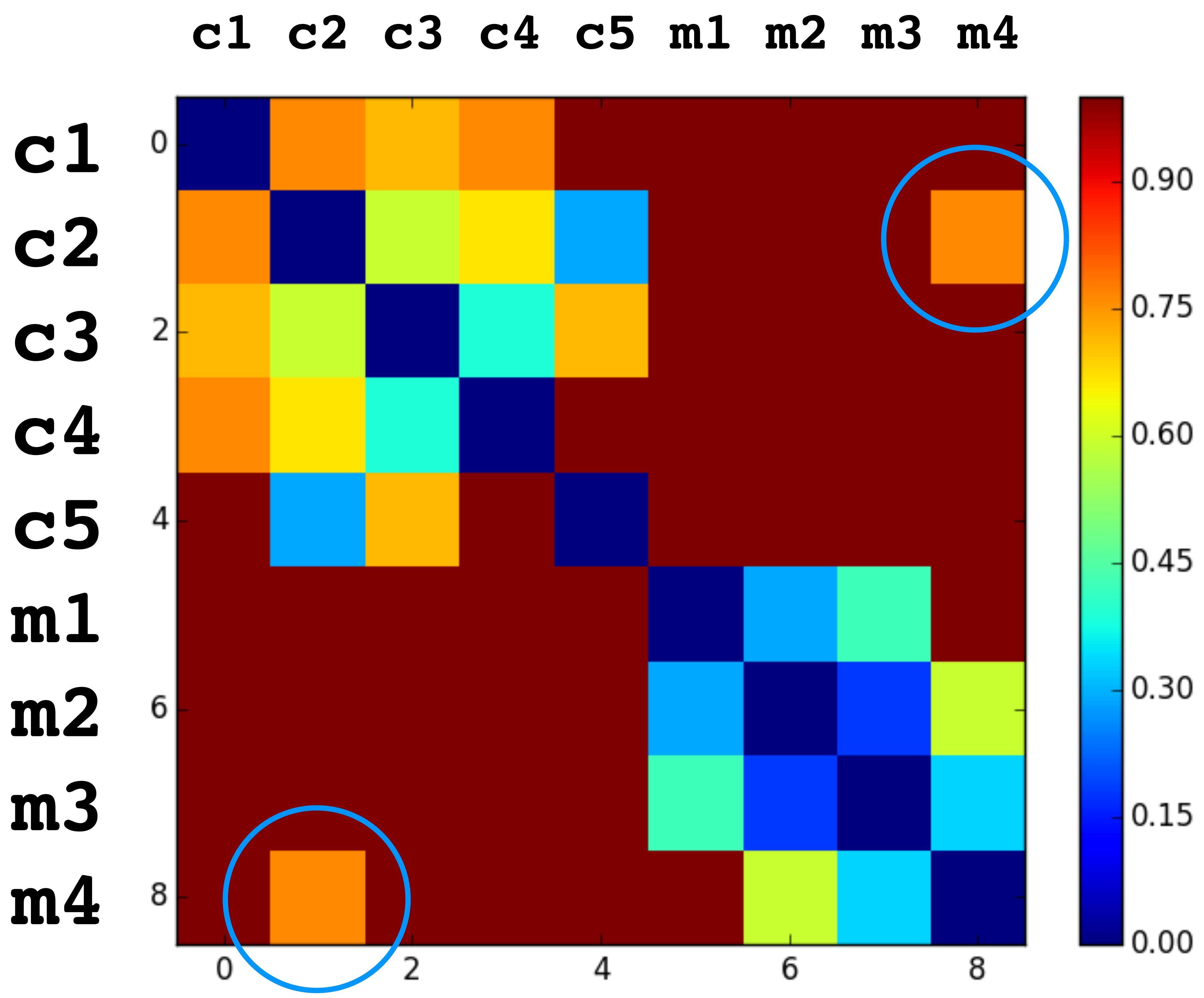


which is more similar?

Doc dissimilarity matrix: example

- Define a distance / dissimilarity measure between docs $D_{ij} = d(x_{i,\cdot}, x_{j,\cdot})$

- Clustering can be tricky
  - High dissimilarity within topic
  - Some cross-topic similarity



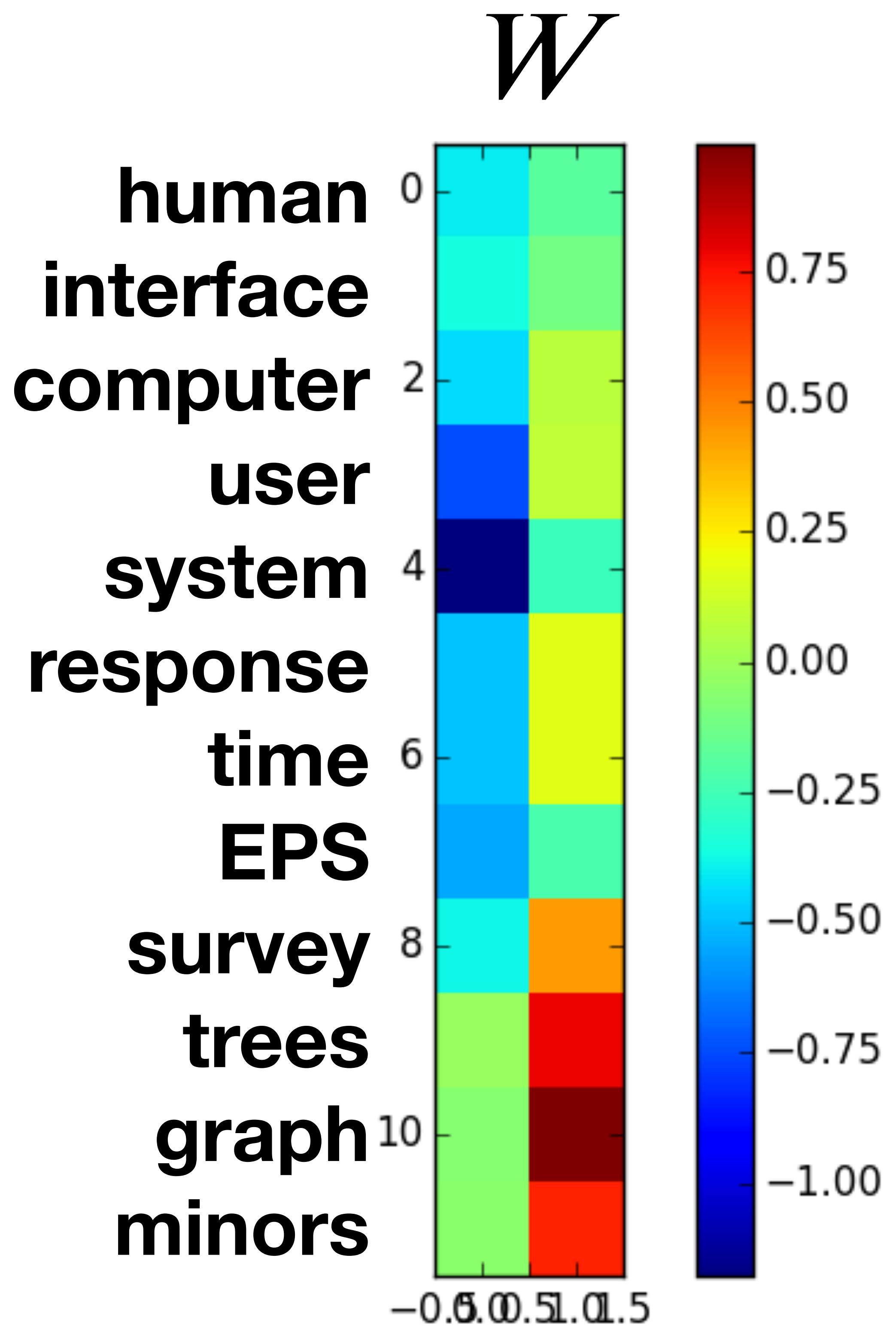
Singular Value Decomposition (SVD)

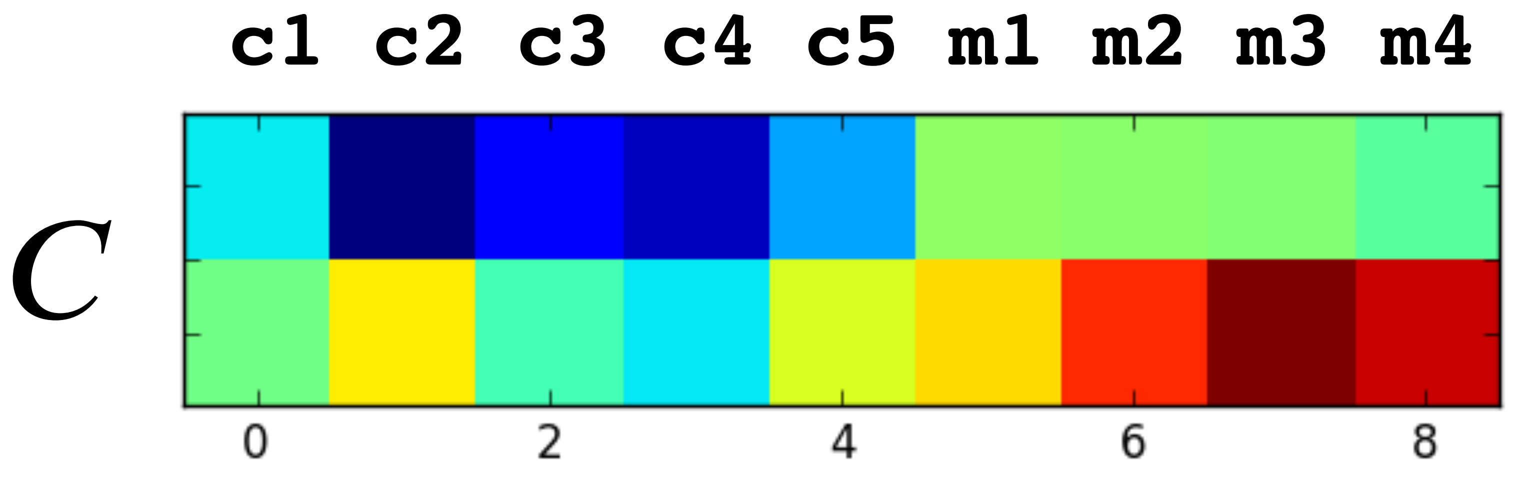
- Useful to preserve feature scale:

$$\boxed{\underset{m\times n}{X}} \approx \boxed{\underset{m\times k}{U}} \cdot \boxed{\underset{k\times k}{D}} \cdot \boxed{\underset{k\times n}{V^\intercal}}$$

$$= \boxed{\underset{m\times k}{W}} \cdot \boxed{\underset{k\times n}{C}} \qquad W = UD^{\frac{1}{2}} \quad C = D^{\frac{1}{2}}V^\intercal$$

- With $k = 2$:

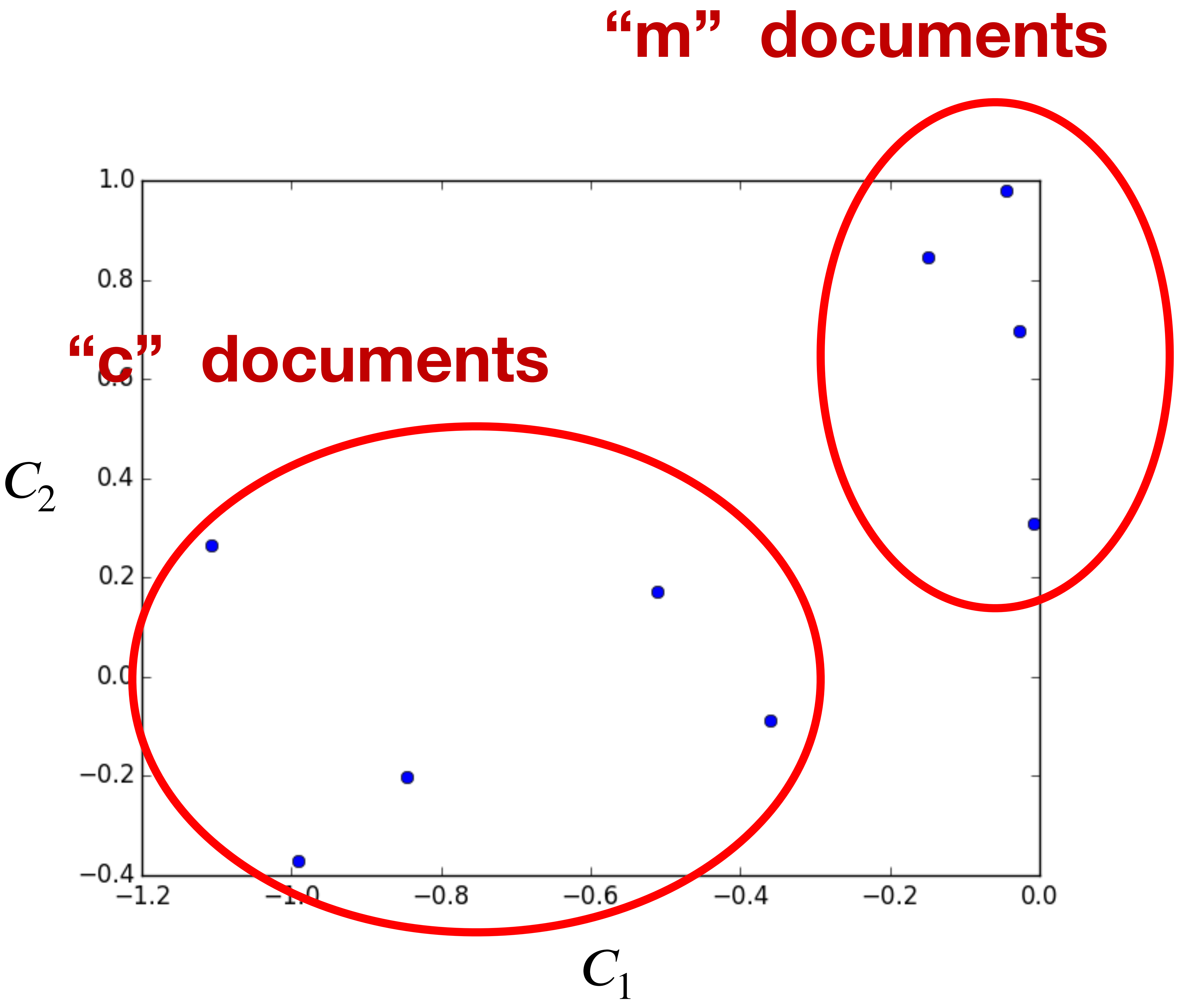




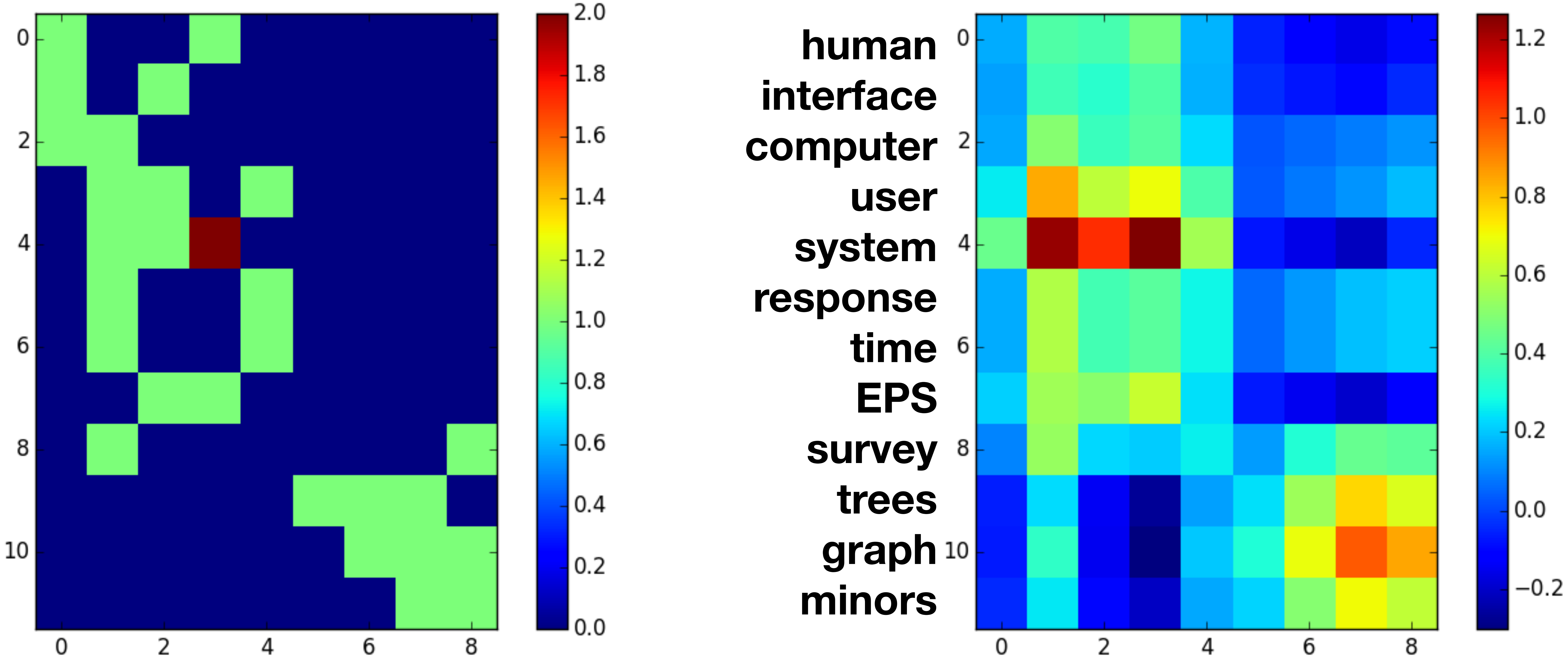
documents in topic c

documents in topic m

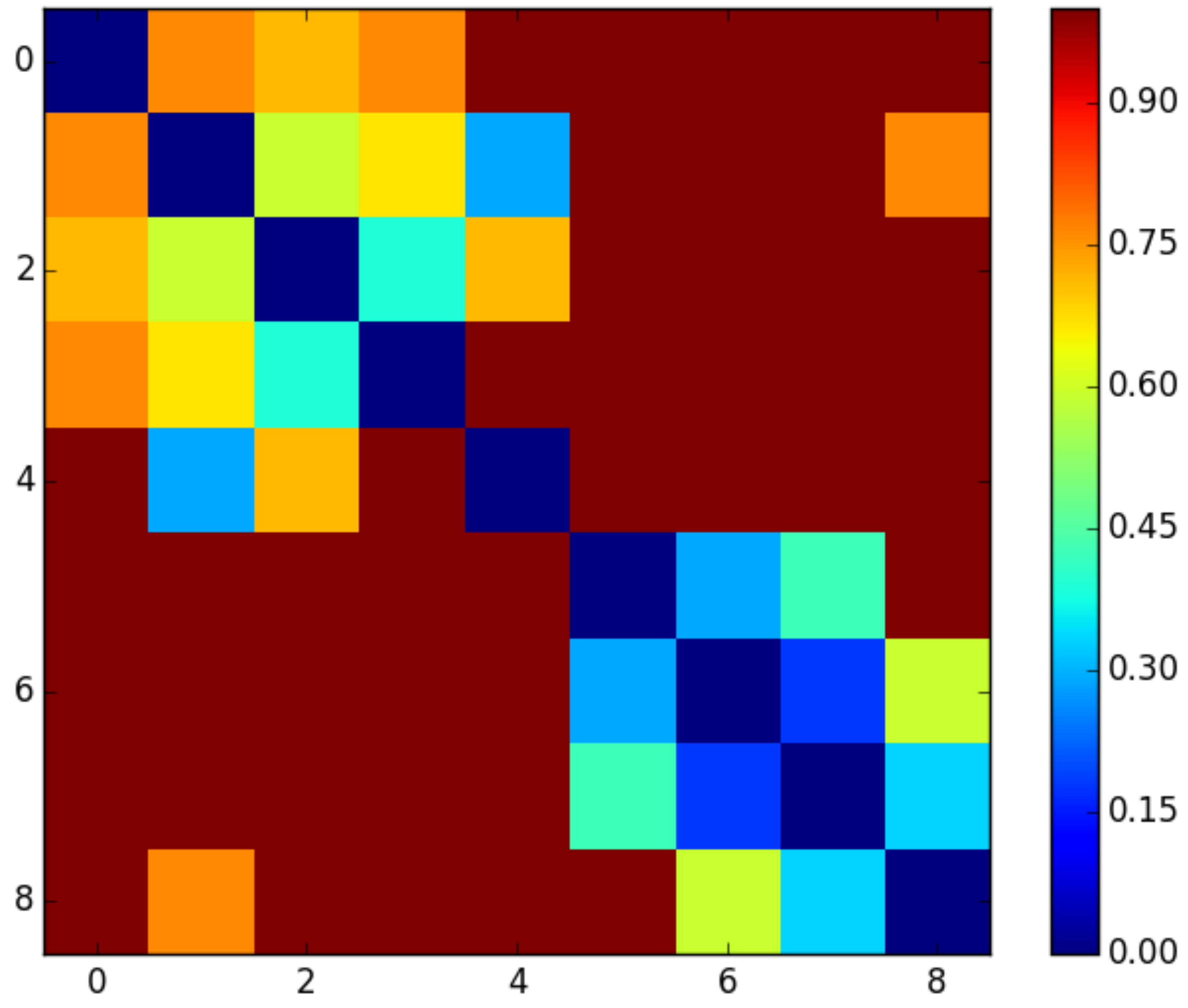
Latent representation: example



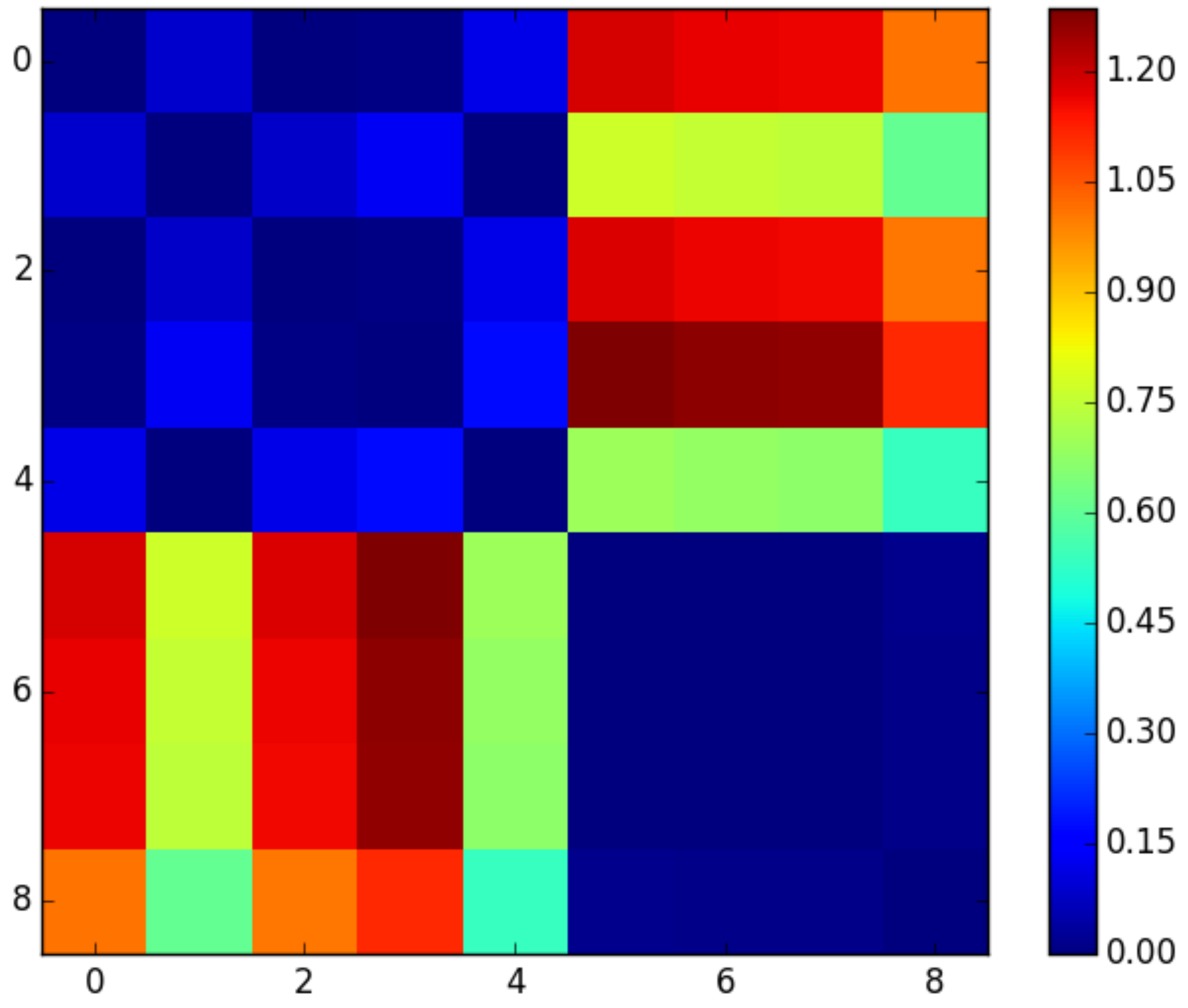
Reconstruction: example



Distance matrix: example



original feature space



latent feature space

- LSA results in more similarity within topics
  - Still some similarity across topics, but easily separable

Today's lecture

Eigen-faces

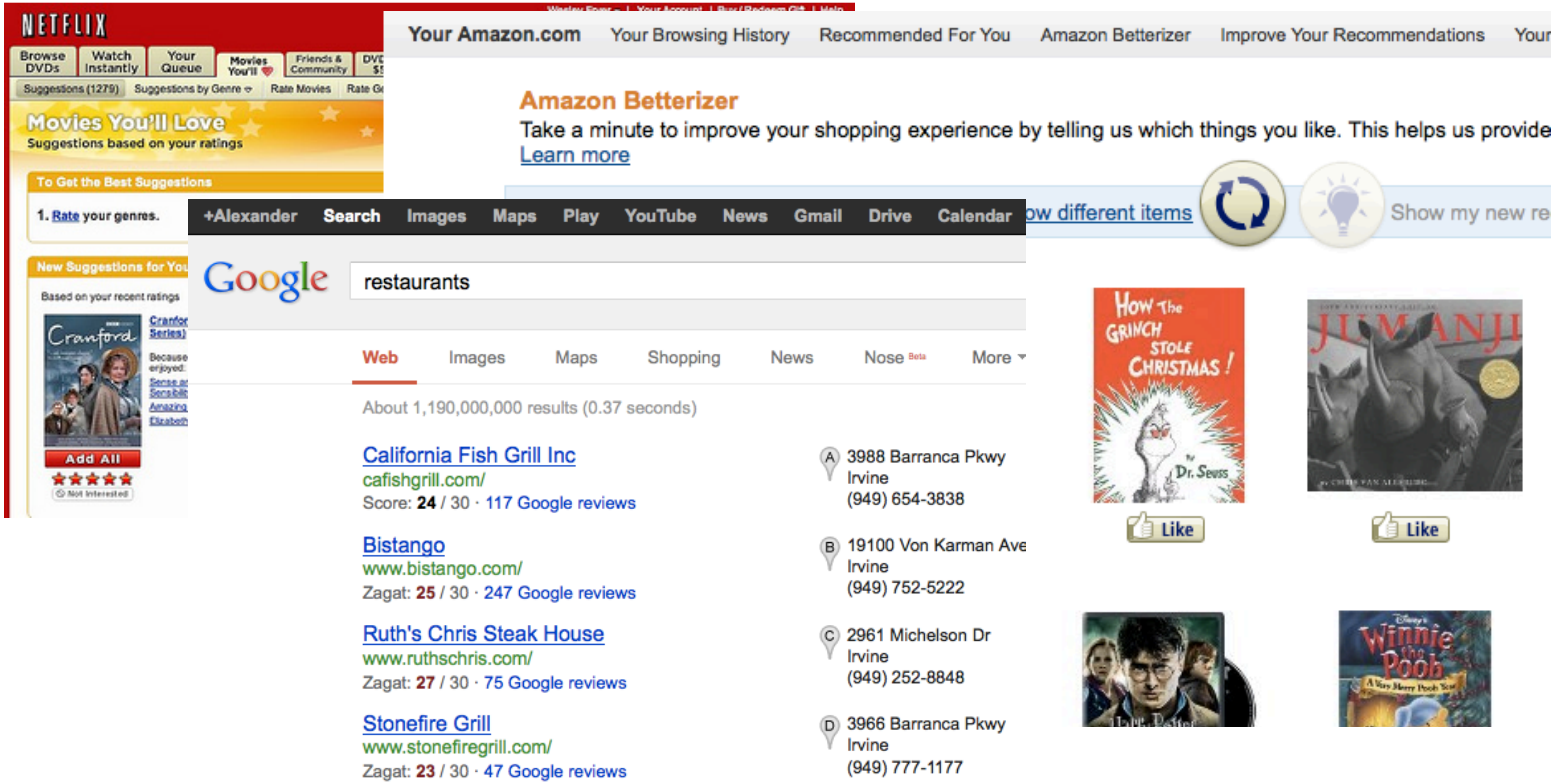
Latent Semantic Analysis

Collaborative Filtering

Recommendation Systems

- Recommend decisions / items the user may like
  - Need to understand both the items and the users
  - E.g. movies, books, groceries, restaurants (remember those?)
- Training data
  - User information $\mapsto$ features: demographics, social network, past ratings
  - Item metadata $\mapsto$ features: creator, genre, ingredients
- Learning output = decision function:
  - Relevance score, predicted rating / ranking

Recommendation Systems: examples



Amazon Betterizer

Take a minute to improve your shopping experience by telling us which things you like. This helps us provide

Learn more

Google: restaurants

About 1,190,000,000 results (0.37 seconds)

California Fish Grill Inc
cafishgrill.com/
Score: 24 / 30 · 117 Google reviews
A 3988 Barranca Pkwy, Irvine, (949) 654-3838

Bistango
www.bistango.com/
Zagat: 25 / 30 · 247 Google reviews
B 19100 Von Karman Ave, Irvine, (949) 752-5222

Ruth's Chris Steak House
www.ruthschris.com/
Zagat: 27 / 30 · 75 Google reviews
C 2961 Michelson Dr, Irvine, (949) 252-8848

Stonefire Grill
www.stonefiregrill.com/
Zagat: 23 / 30 · 47 Google reviews
D 3966 Barranca Pkwy, Irvine, (949) 777-1177

Paradigms

- Recommendation Systems help reduce information overload
  - Identify most relevant items to attend to

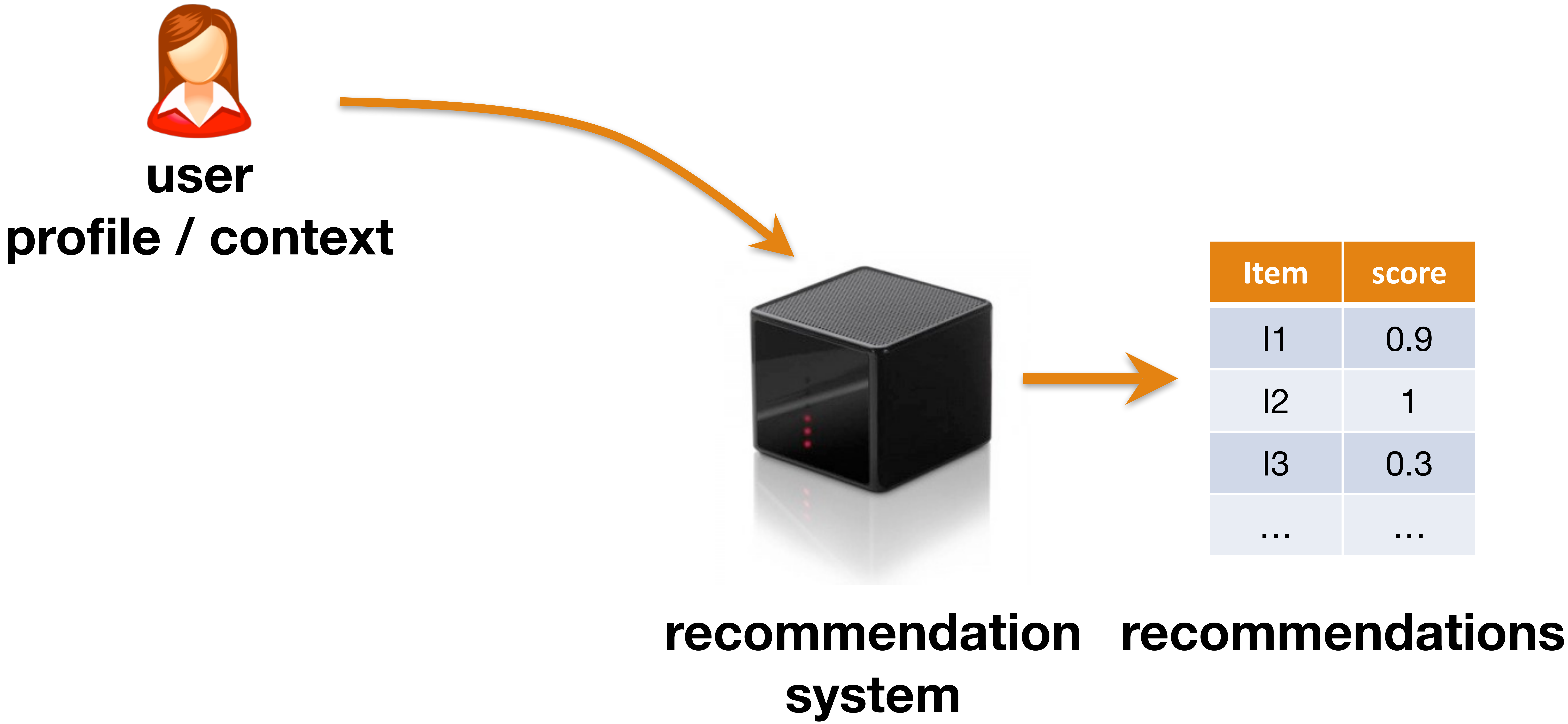
Item	score
I1	0.9
I2	1
I3	0.3
…	…

recommendation system

recommendations

Paradigms

- Personalized recommendations



user
profile / context

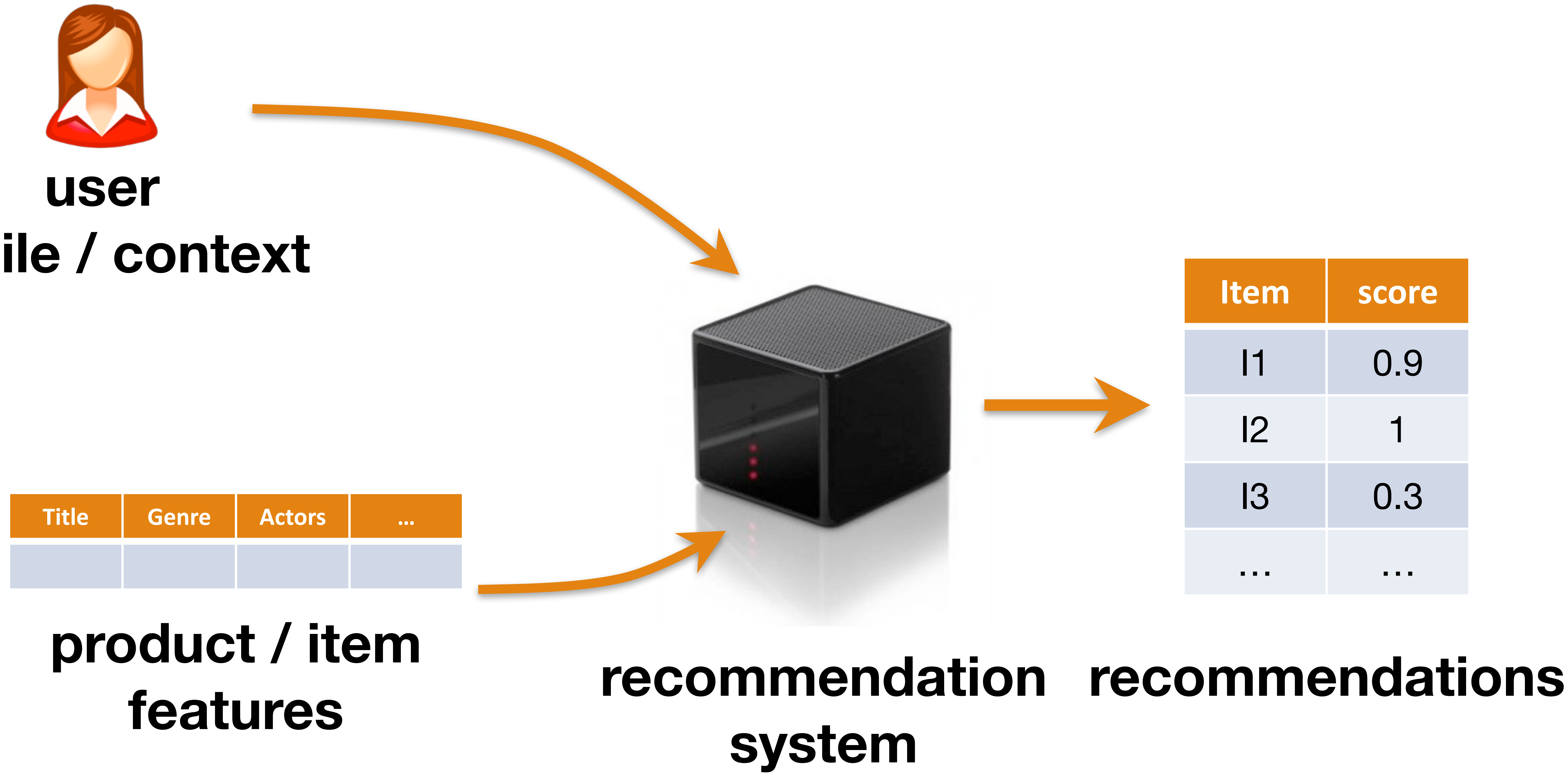
recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

recommendations

Paradigms

- Content-based: “show me more of the things I liked before”



user profile / context

Title	Genre	Actors	...

product / item features

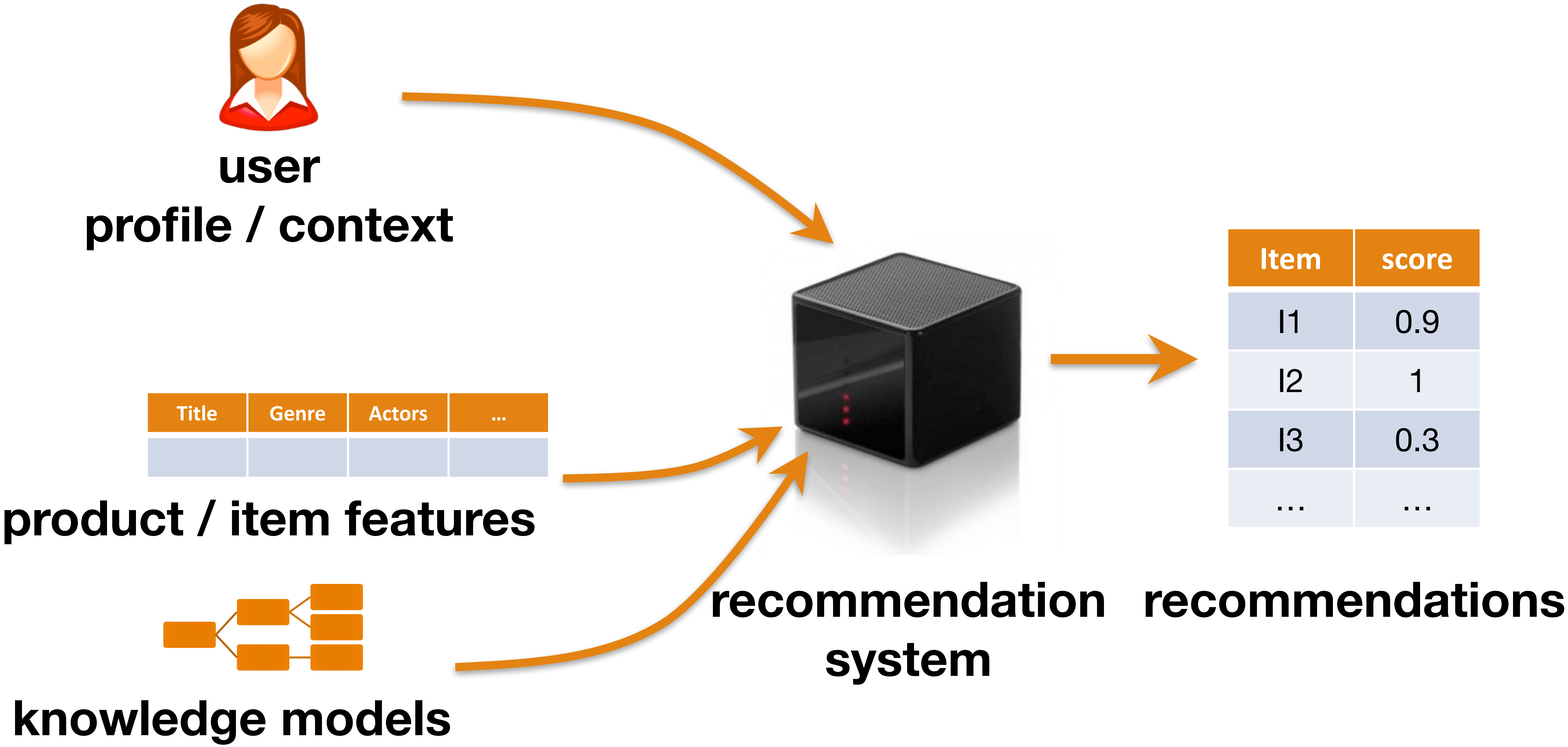
recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

recommendations

Paradigms

- Knowledge-based: e.g., “tell me what fits my needs”



user profile / context

Title	Genre	Actors	…

product / item features

knowledge models

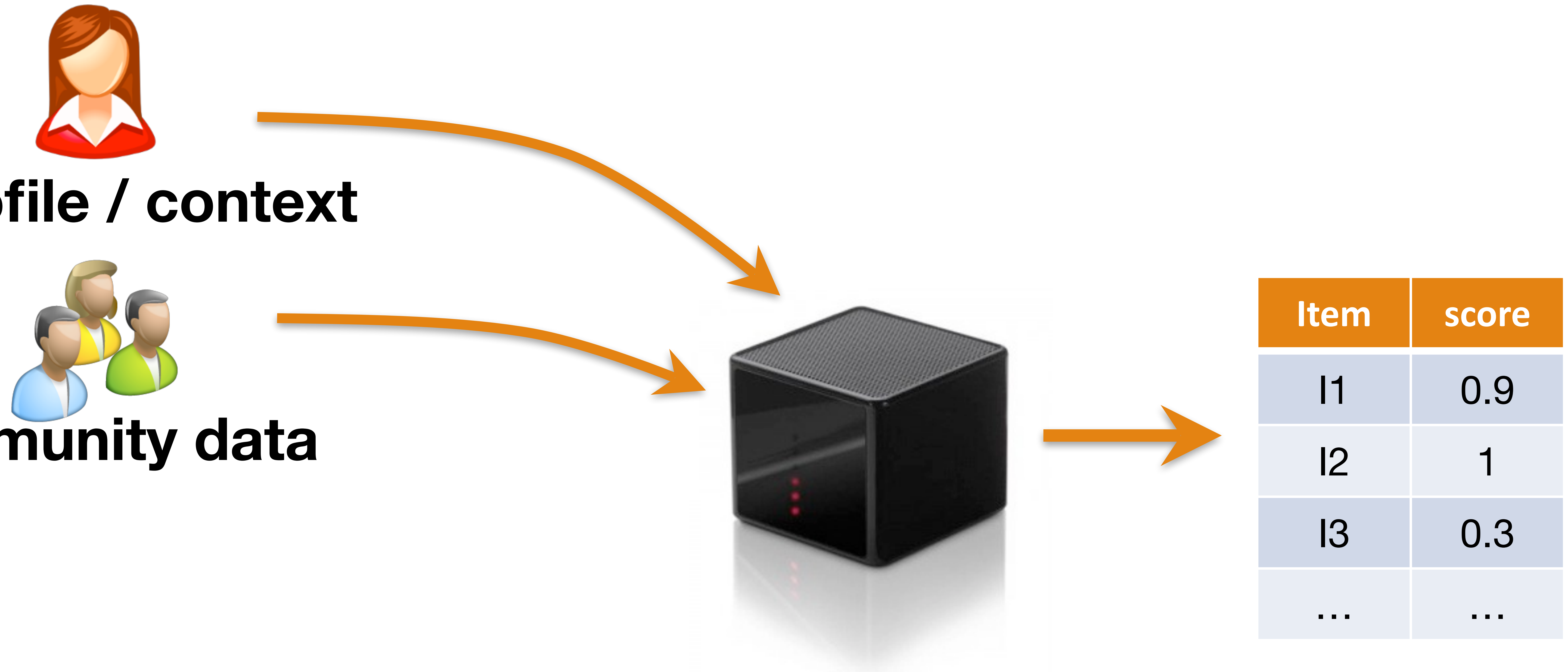
recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

recommendations

Paradigms

- Collaborative Filtering: “tell me what’s popular among my peers”



user profile / context

community data

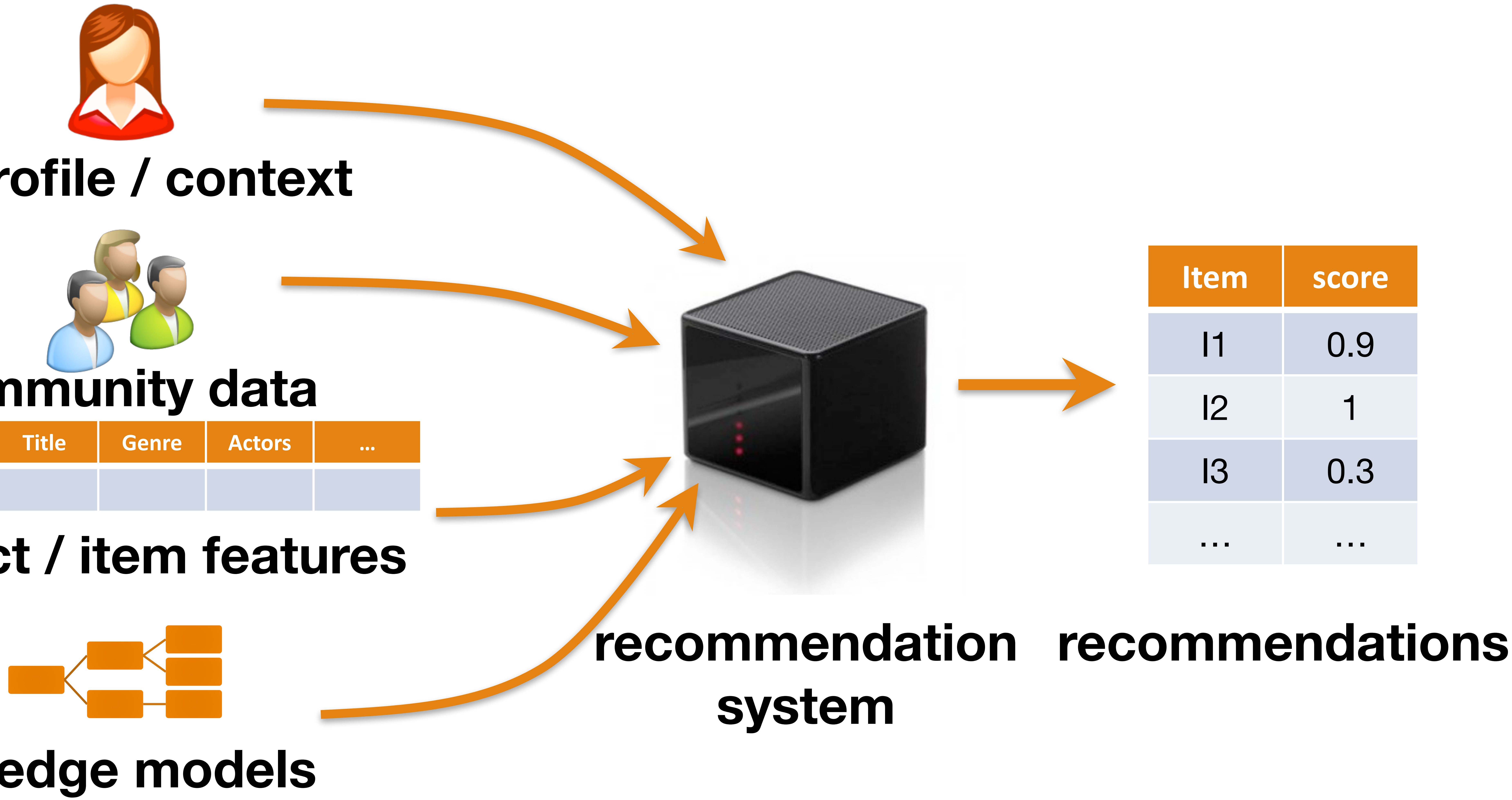
Item	score
I1	0.9
I2	1
I3	0.3
…	…

recommendation system

recommendations

Paradigms

- Hybrid: combine information from many inputs and/or methods



user profile / context

community data

Title	Genre	Actors	...

product / item features

knowledge models

recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

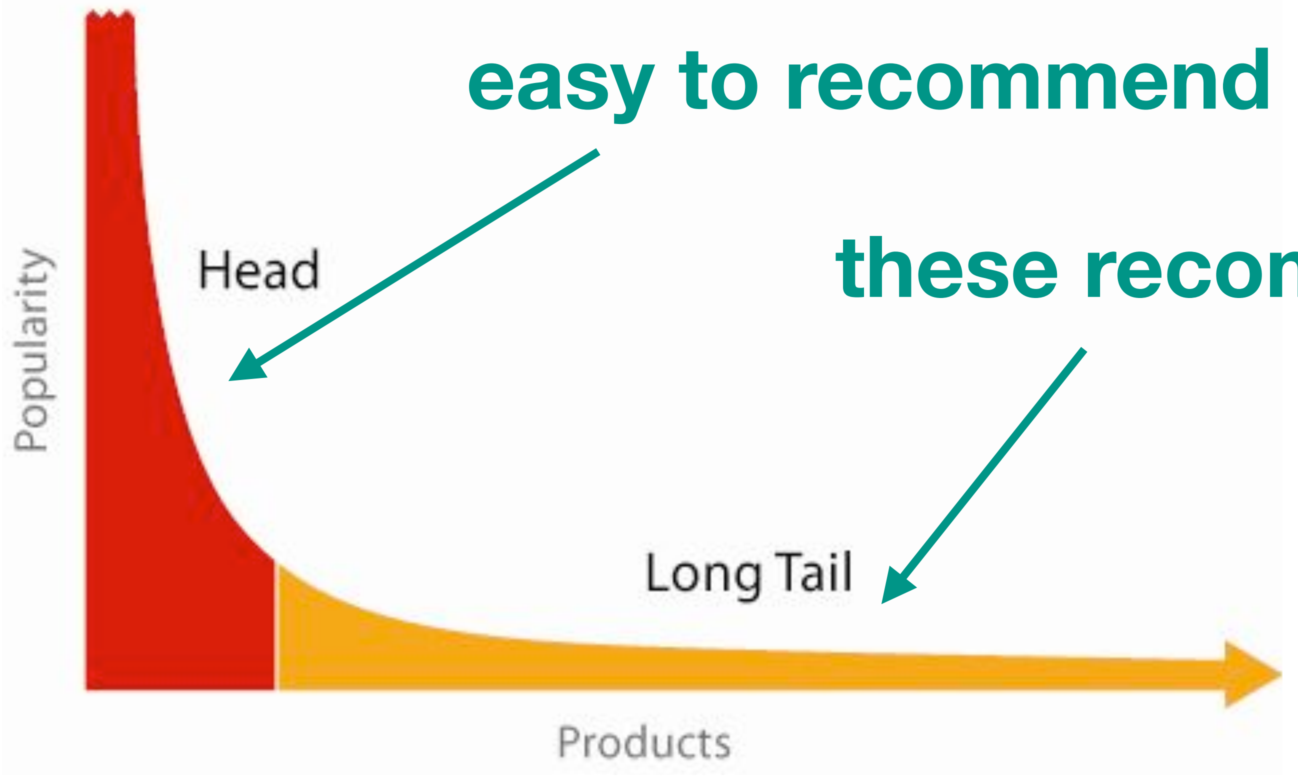
recommendations

Measuring success

- Prediction perspective
  - Predict to what degree users like the item
  - Most common evaluation for research
  - Regression vs. “top-K” ranking
- Interaction perspective
  - Promote positive “feeling” in users (“satisfaction”, engagement)
  - Educate vs. persuade
- Conversion perspective
  - Measure commercial success (“hits”, click-through rates)
  - Optimize sales and profits

Why are recommenders important?

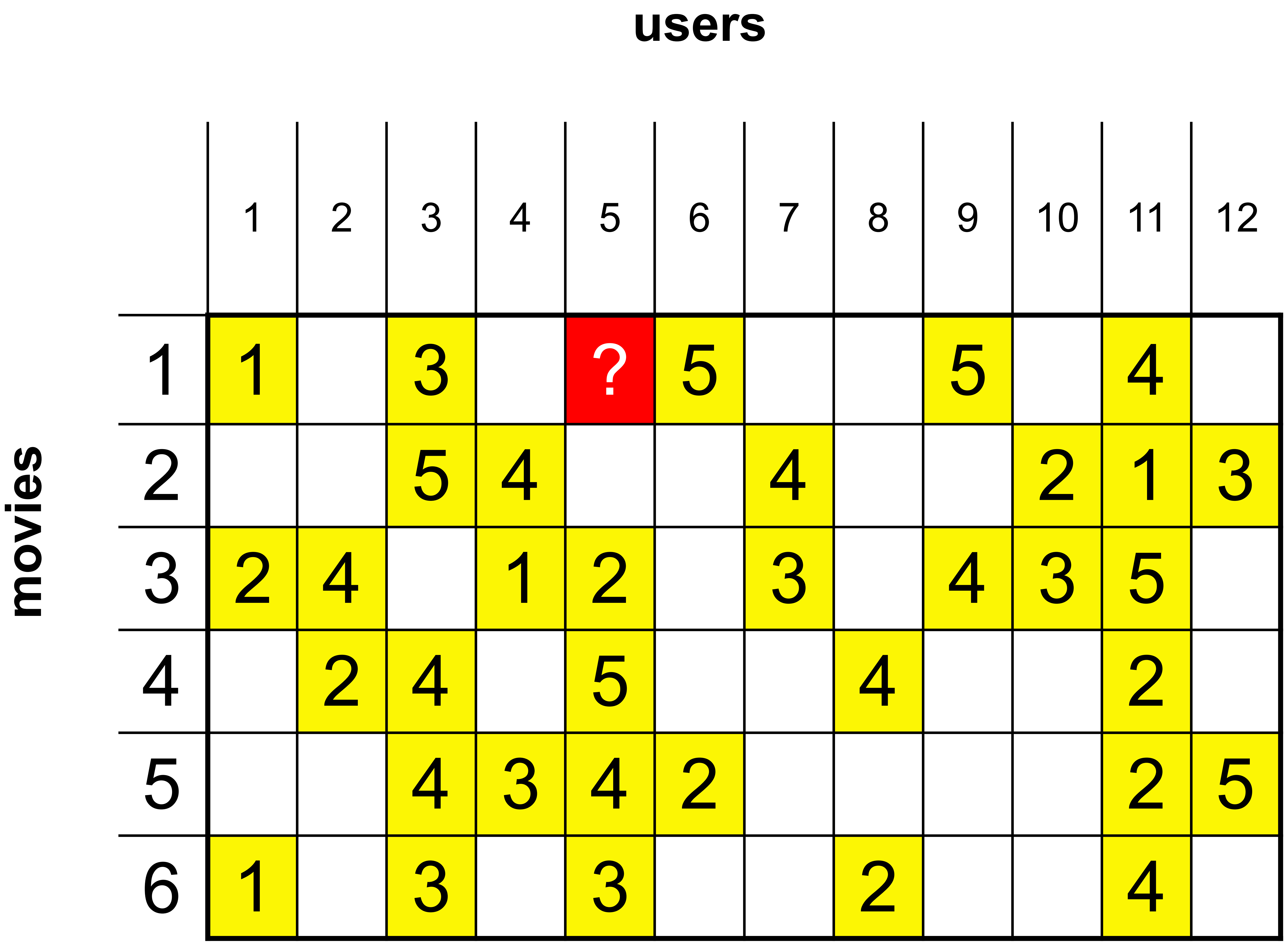
- The long tail of item appeal
  - A few items are very popular
  - Most items are popular only with a few people
    - But everybody is interested in some obscure items
- Goal: recommend scarcely known items that the user might like



easy to recommend best-sellers

these recommendations must be well-targeted

Collaborative Filtering: example

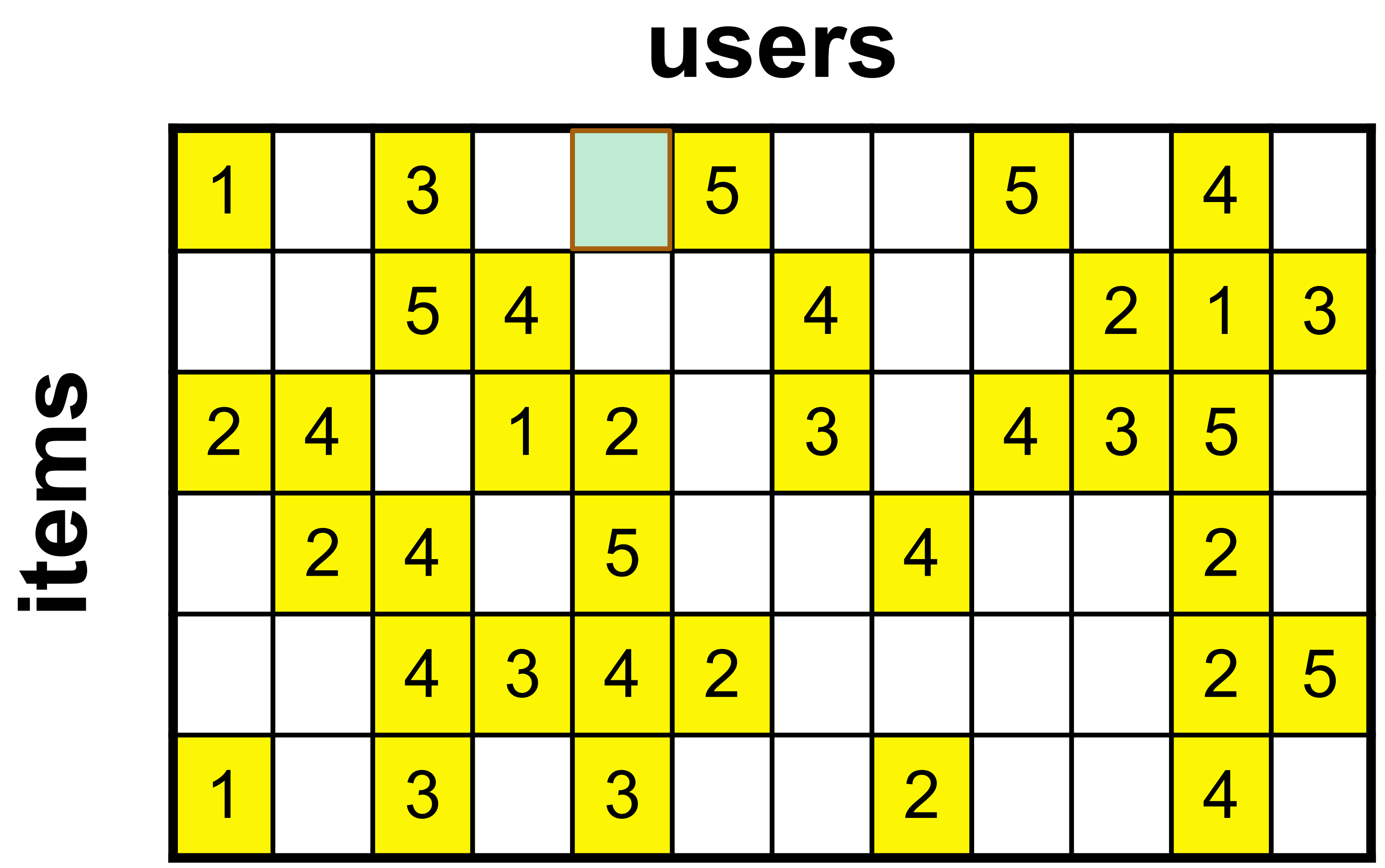


users

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Latent-space models

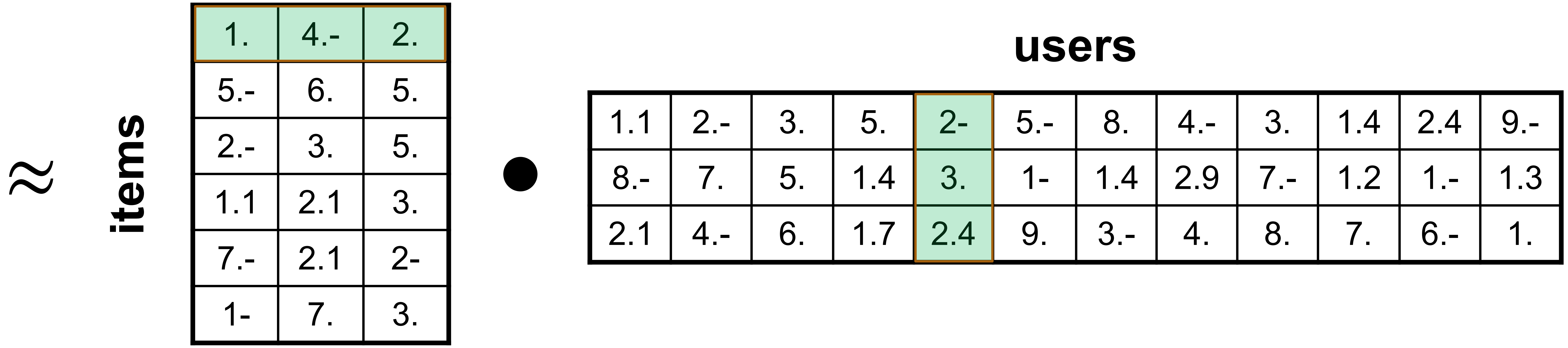
- Model: ratings matrix = user features · movie features
  - Learn values from known ratings
  - Extrapolate to unrated



users

items

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

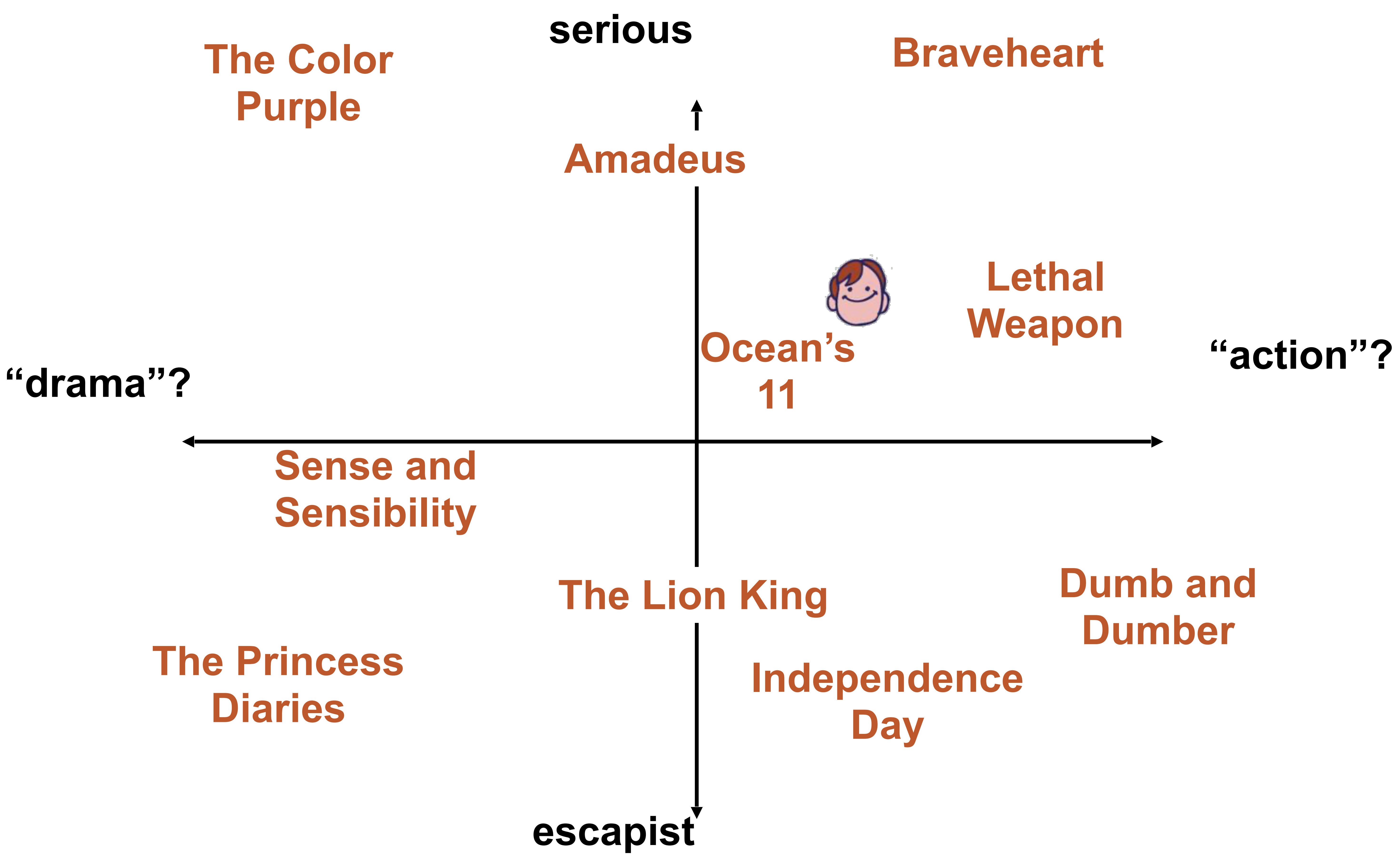
1.	4.-	2.
5.-	6.	5.
2.-	3.	5.
1.1	2.1	3.
7.-	2.1	2-
1-	7.	3.

•

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.

Latent-space models



Latent-space models

- Ideally, latent representation encodes some meaning
  - What kind of movie is this? Which movies is it similar to?

- Most data is missing $\implies$ hard to perform SVD
  - $\implies$ gradient decent on loss $\mathscr{L}(U, V) = \sum_{u,m} \left(X_{mu} - \sum_k U_{mk} V_{ku} \right)^2$

```
# for user u, movie m, find the k'th eigenvector & coefficient by iterating:
predict_um = U[m,:].dot( V[:,u] )   # predict: vector-vector product
err = ( rating[u,m] - predict_um )  # find error residual
V_ku, U_mk = V[k,u], U[m,k]         # make copies for update
U[m,k] += alpha * err * V_ku        # Update our matrices
V[k,u]  += alpha * err * U_mk       # (compare to least-squares gradient)
```

Latent-space models: extensions

- Add lower-order terms: $r_{mu} \approx \mu + b_m + b_u + \sum_k U_{mk} V_{ku}$
  - μ = overall average rating (affected by user interface etc.)
  - $b_m + b_u$ = item and user biases
- Can add non-linearity (saturating?)
- Choose a loss, e.g. MSE or multilogistic
- Train using a gradient-based optimizer

Ensembles for recommendations

- Many possible models:
  - Feature-based regression
  - (Weighted) kNN on items
  - (Weighted) kNN on users
  - Latent space representation
- Perhaps we should combine them?
- Use an ensemble average, or a stacked ensemble
  - Stacked ensemble = train a weighted combination of model predictions

Demo

- http://www.benfrederickson.com/matrix-factorization/

Logistics

assignments

- Assignment 5 due Thursday

project

- Final report due next Thursday

evaluations

- Evaluations due end of next week

final exam

- Review: next Thursday
- Final: Thursday, March 18, 1:30–3:30pm