

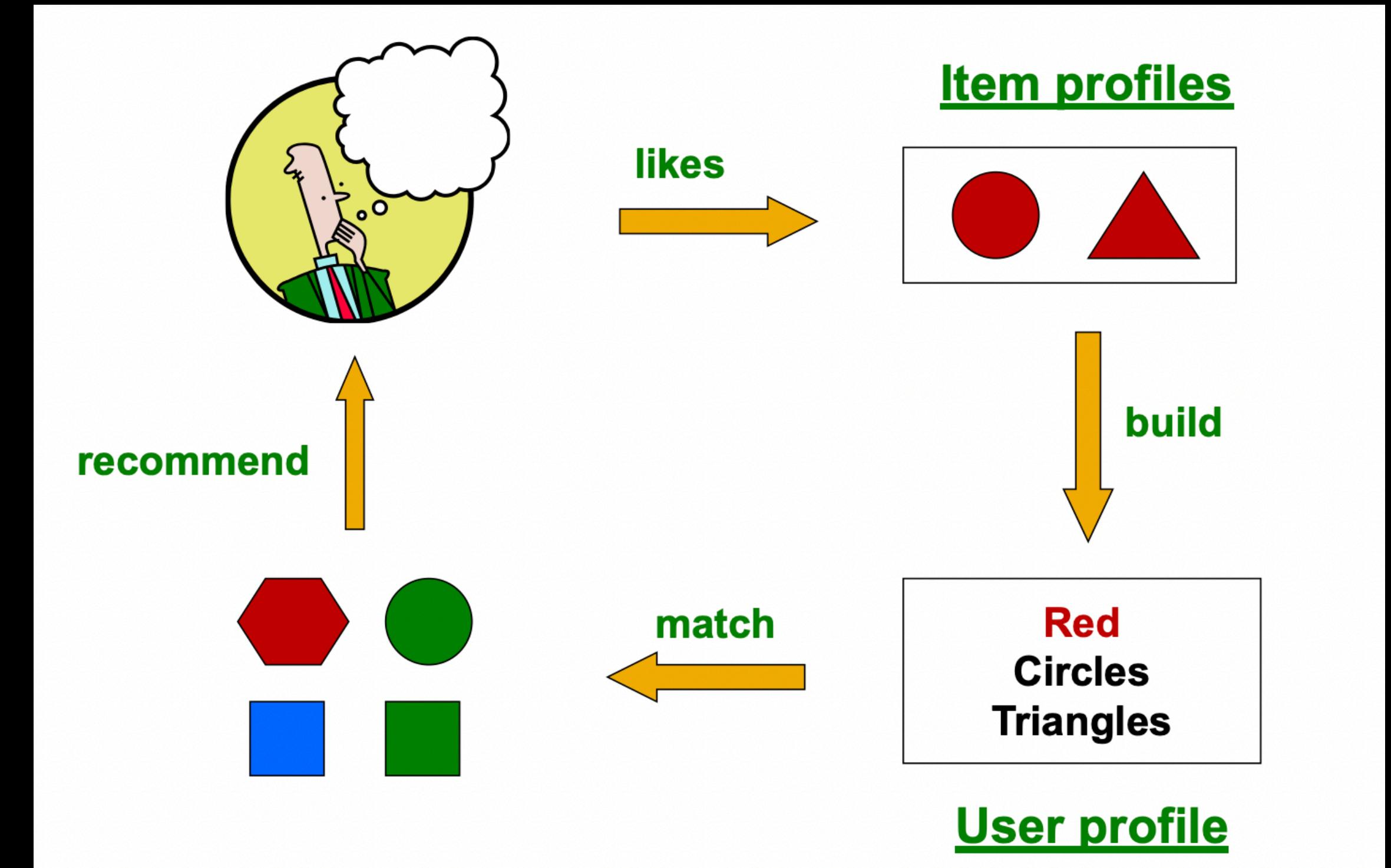
Recommender Systems

An abridging of Jure Leskovec's research

Pablo Rodriguez Bertorello

Sketch of Plan

- How much will user x like item y?
- Recommend top k items user x may like
- Update user from further action of all users



Approaches

BellKor Recommendation Engine

- Compare to content-based: eg TF-IDF
- Collaborative: user-based, or item-based
- Latent factor

- **The winner of the Netflix Challenge**

- **Multi-scale modeling of the data:**

Combine top level, “regional” modeling of the data, with a refined, local view:

- **Global:**

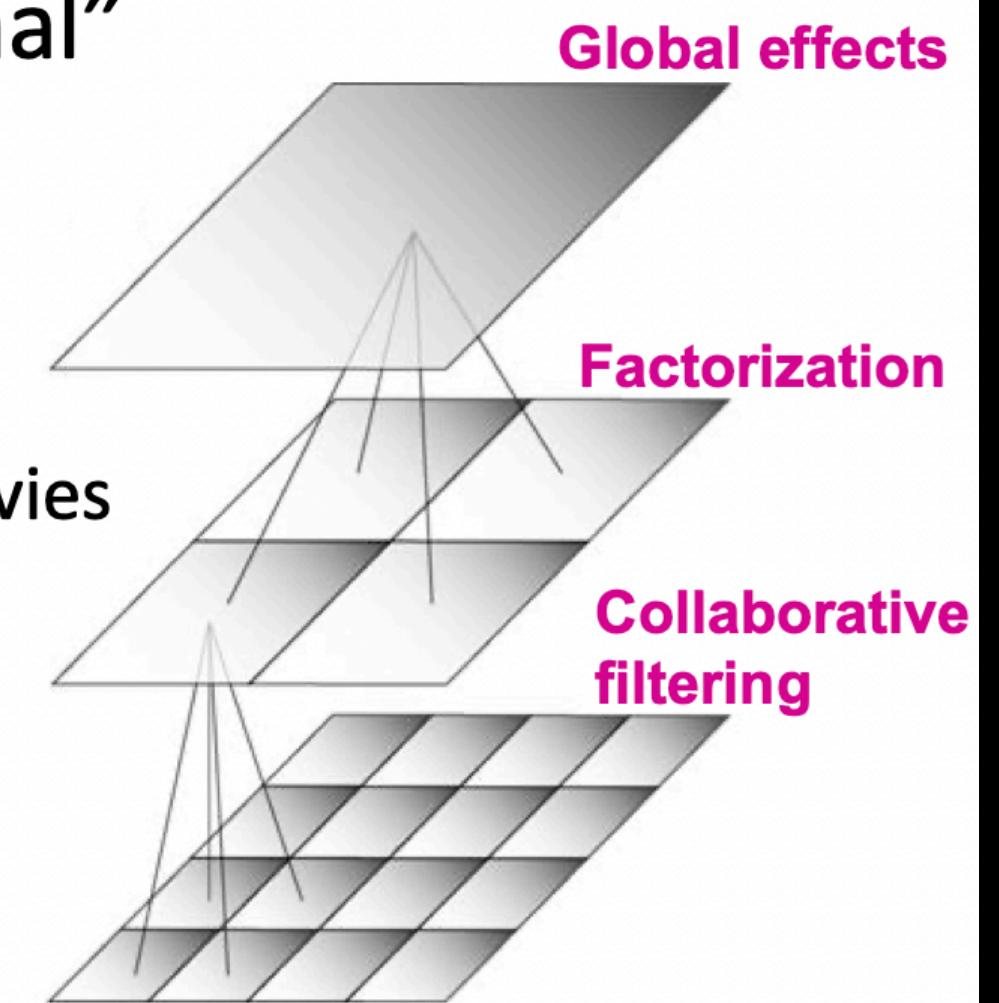
- Overall deviations of users/movies

- **Factorization:**

- Addressing “regional” effects

- **Collaborative filtering:**

- Extract local patterns



Collaborative Filtering

User Collaborative Filtering

- Consider user x
- Find set N of other users whose ratings are “similar” to x’s ratings
- Estimate x’s ratings based on ratings of users in N
- Alternatively: Item Collaborative Filtering

	Avatar	LOTR	Matrix	Pirates
Alice	1	0.2		
Bob		0.5	0.3	
Carol	0.2		1	
David				0.4

Which Similarity?

- Let r_x be the vector of user x's ratings
 - Jaccard similarity measure
 - Cosine similarity measure
 - Pearson correlation coefficient

Prediction Via Cosine Similarity

- Let r_x be the vector of user x's ratings
- Let N be the set of k users most similar to x who have rated item i
- Prediction for item i of user x:

- $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
- Or even better: $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Similarity to Nearest Neighbors

- Define **similarity** s_{ij} of items i and j
- Select k nearest neighbors $N(i; x)$
 - Items most similar to i , that were rated by x
- Estimate rating r_{xi} as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i; x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i; x)} s_{ij}}$$

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

- μ = overall mean movie rating
- b_x = rating deviation of user x
 $= (\text{avg. rating of user } x) - \mu$
- b_i = rating deviation of movie i

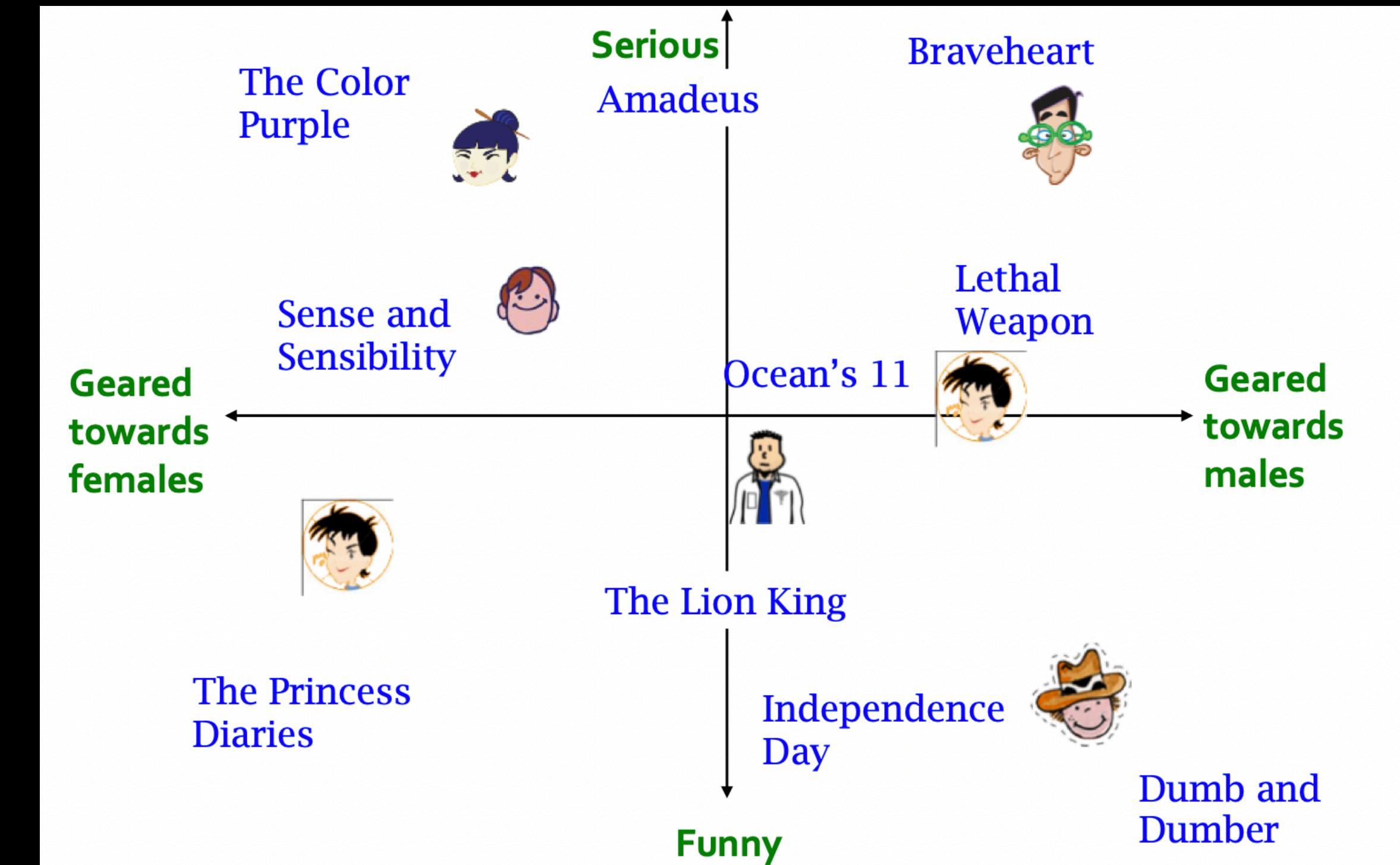
Pro/Cons of Collaborative Filtering

- Works for any kind of item
- But relies only on local knowledge
- Cold start, need enough users in the system to find a match
- Sparsity, hard to find users that have rated the same items
- Popularity bias, cannot recommend items to someone with unique taste

Latent Factor

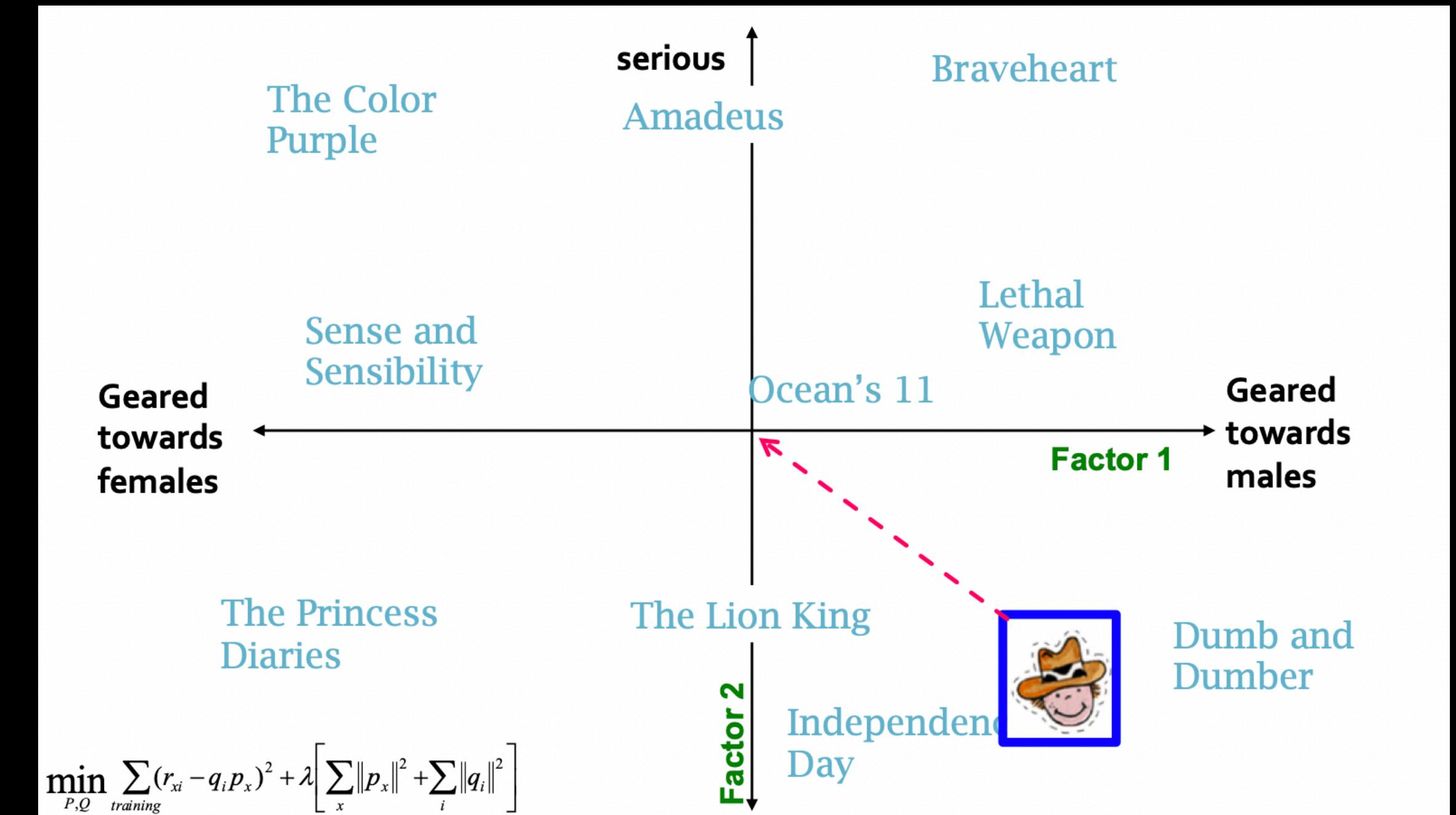
The Plan (I)

Future Space



Enhanced Plan (I)

Principal concept-space



Predict

How much will user like item?

- How to estimate the missing rating of user x for item i ?

		users					
		1	3	5	5	5	4
items	1		3	5	?	4	
	2	4		1	2	3	4
items		2	4	5		4	2
	1		4	3	4	2	
		1	3	3		2	4

≈

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

		users											
		1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
• factors	1	-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
	2	2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1
	3												
	4												
	5												
	6												
	7												

Q

$$\begin{aligned}\hat{r}_{xi} &= q_i \cdot p_x \\ &= \sum_f q_{if} \cdot p_{xf} \\ q_i &= \text{row } i \text{ of } Q \\ p_x &= \text{column } x \text{ of } P^T\end{aligned}$$

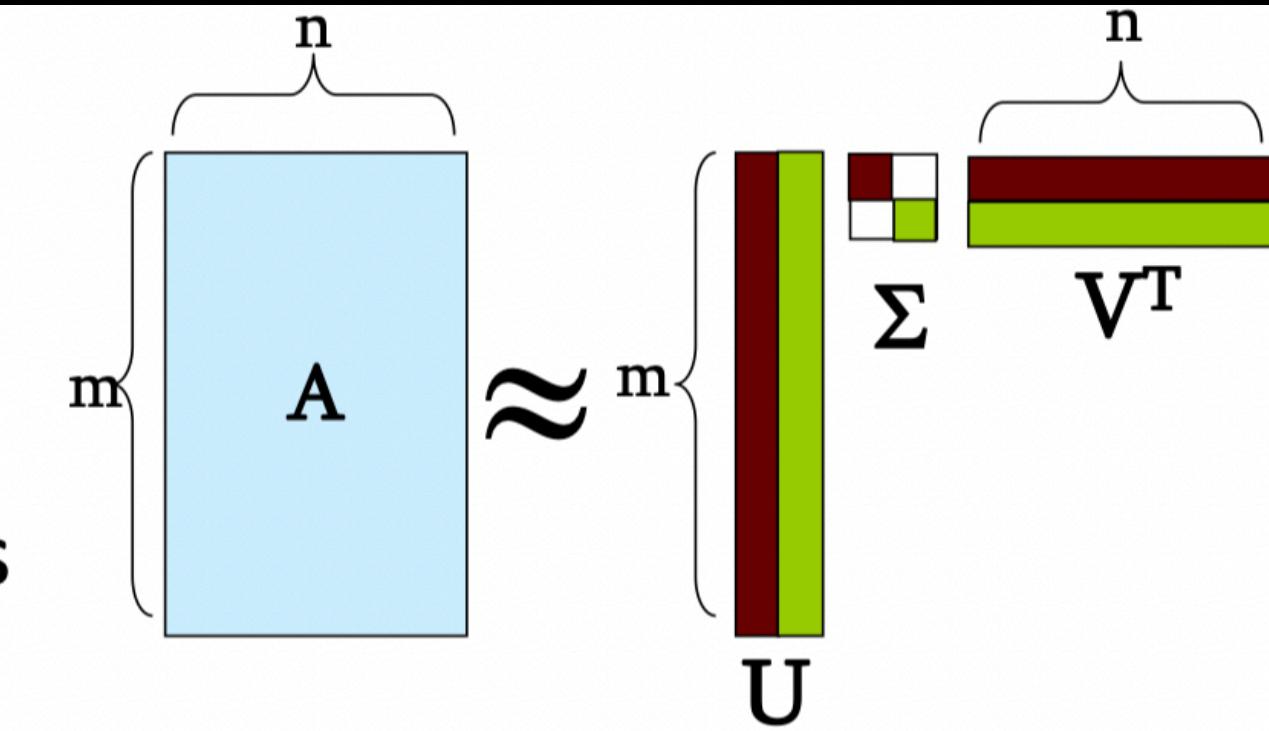
SVD

Factor the User-concept,
and Item-concept eigen-vectors

- **Remember SVD:**
 - A : Input data matrix
 - U : Left singular vecs
 - V : Right singular vecs
 - Σ : Singular values

- **So in our case:**
“SVD” on Netflix data: $R \approx Q \cdot P^T$
 $A = R, Q = U, P^T = \Sigma V^T$

$$\hat{r}_{xi} = q_i \cdot p_x$$



Latent Factor Model

- Machine-learn the factorization
- Minimize rating matrix reconstruction error
- Loss excludes un-rated cells (empty/null). No assumptions:
 - zero
 - mean

- We already know that SVD gives minimum reconstruction error (Sum of Squared Errors):

$$\min_{U,V,\Sigma} \sum_{ij \in A} (A_{ij} - [U\Sigma V^T]_{ij})^2$$

- Note two things:
 - SSE and RMSE are monotonically related:
 - $RMSE = \frac{1}{c}\sqrt{SSE}$ Great news: SVD is minimizing RMSE!
 - Complication: The sum in SVD error term is over all entries (no-rating is interpreted as zero-rating). But our R has missing entries!

Backup

Term Frequency - Inverse Document Frequency

f_{ij} = frequency of term (feature) i in doc (item) j

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

Note: we normalize TF
to discount for “longer”
documents

n_i = number of docs that mention term i

N = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

TF-IDF score: $w_{ij} = TF_{ij} \times IDF_i$

Doc profile = set of words with highest TF-IDF scores, together with their scores