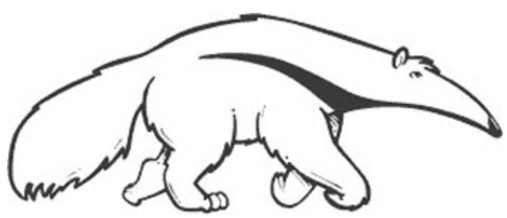
+

Machine Learning and Data Mining

Collaborative Filtering & Recommender Systems

Prof. Alexander Ihler



BREN:ICS
INFORMATION AND COMPUTER SCIENCES

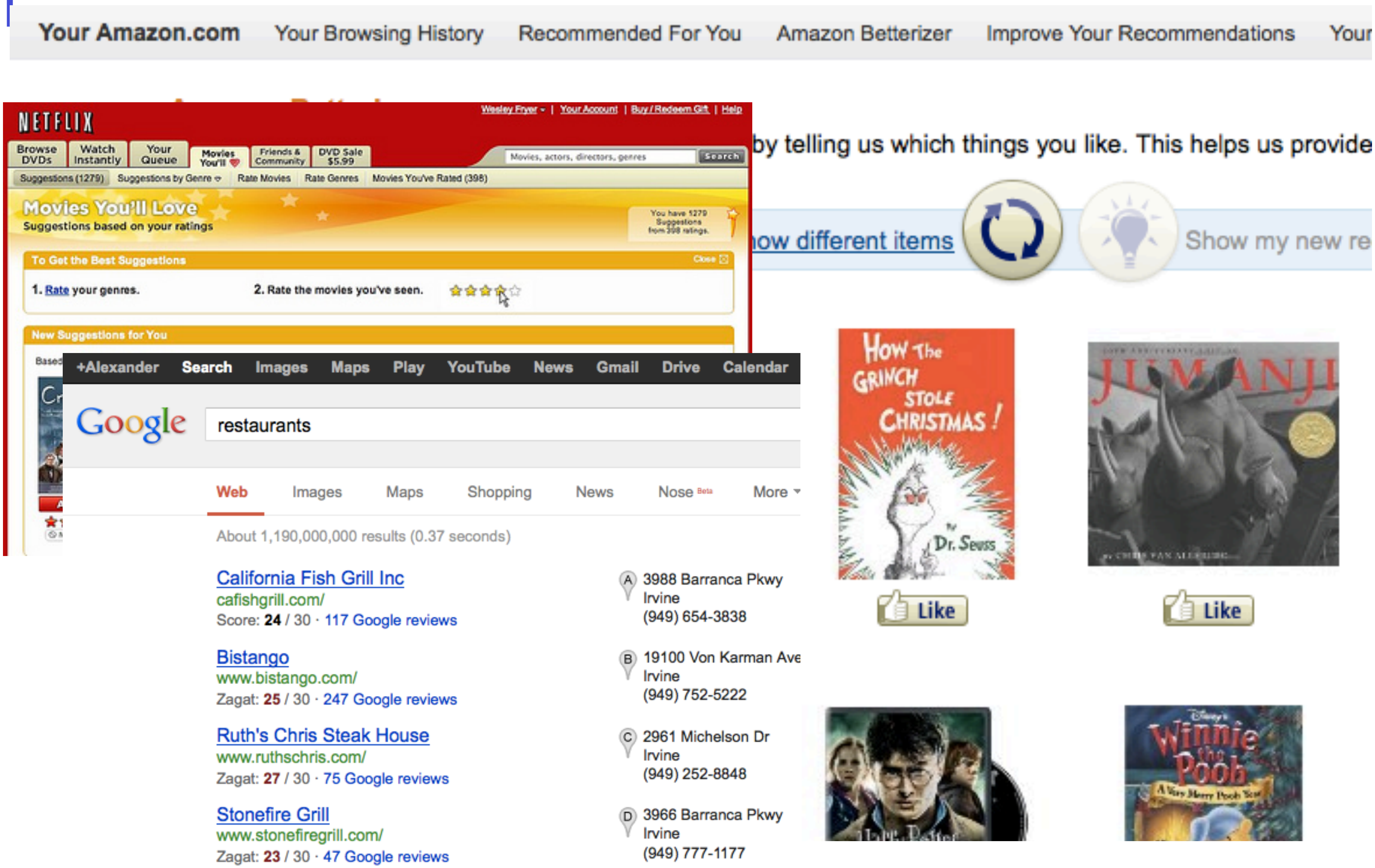
UNIVERSITY of CALIFORNIA IRVINE

Recommender systems

- Automated recommendations

- Inputs
  - User information
    - Situation context, demographics, preferences, past ratings
  - Items
    - Item characteristics, or nothing at all
- Output
  - Relevance score, predicted rating, or ranking

Recommender systems: examples



Paradigms of recommender systems

Recommender systems reduce information overload by estimating relevance

Recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

Recommendations

Paradigms of recommender systems

Personalized recommendations

User profile / context

Recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

Recommendations

Paradigms of recommender systems

Content-based:
"Show me more of the same things that I've liked"

User profile / context

Title	Genre	Actors	...

Product / item features

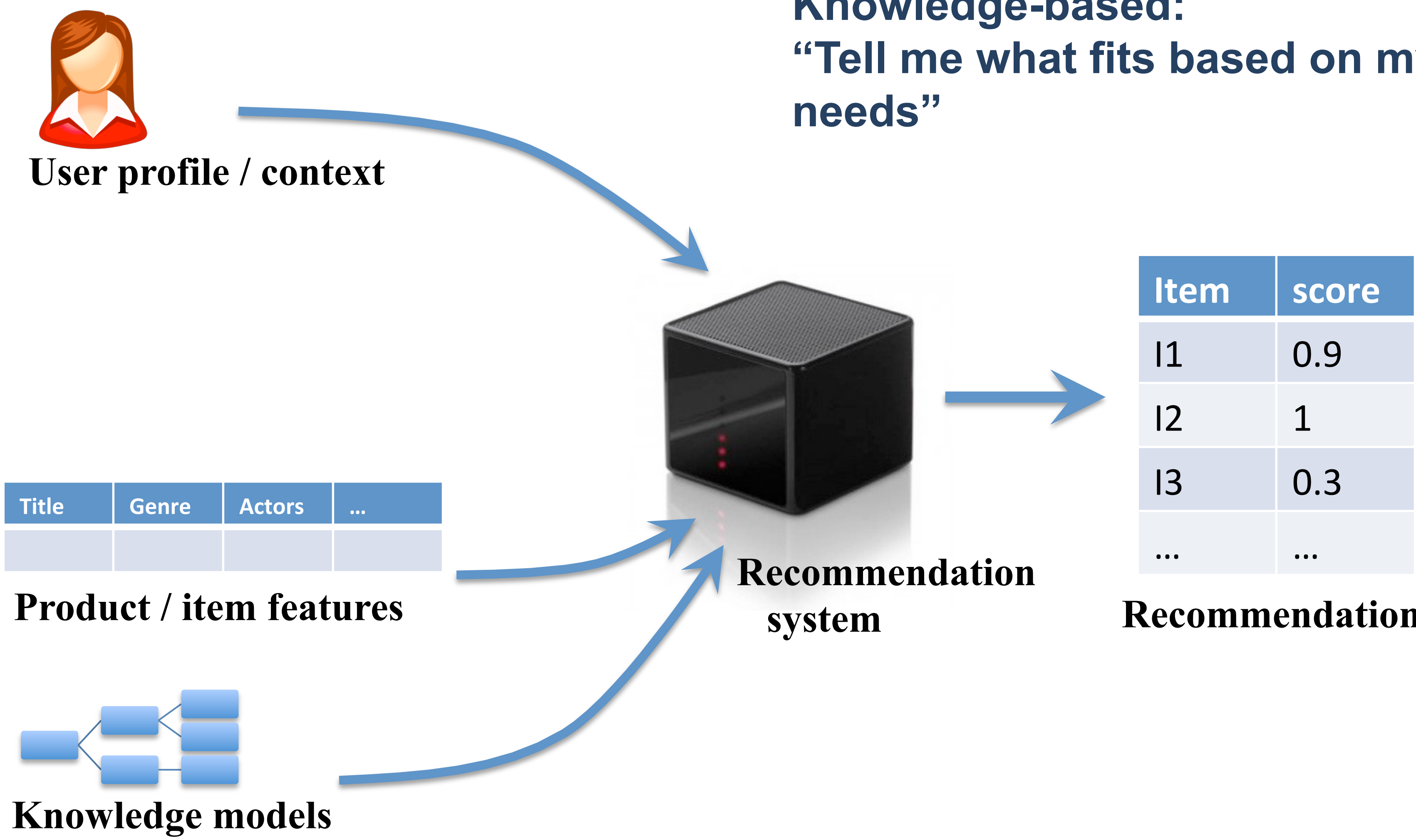
Recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
...	...

Recommendations

Paradigms of recommender systems

Knowledge-based:
“Tell me what fits based on my needs”



Paradigms of recommender systems

Collaborative:
"Tell me what's popular among my peers"

User profile / context

Community data

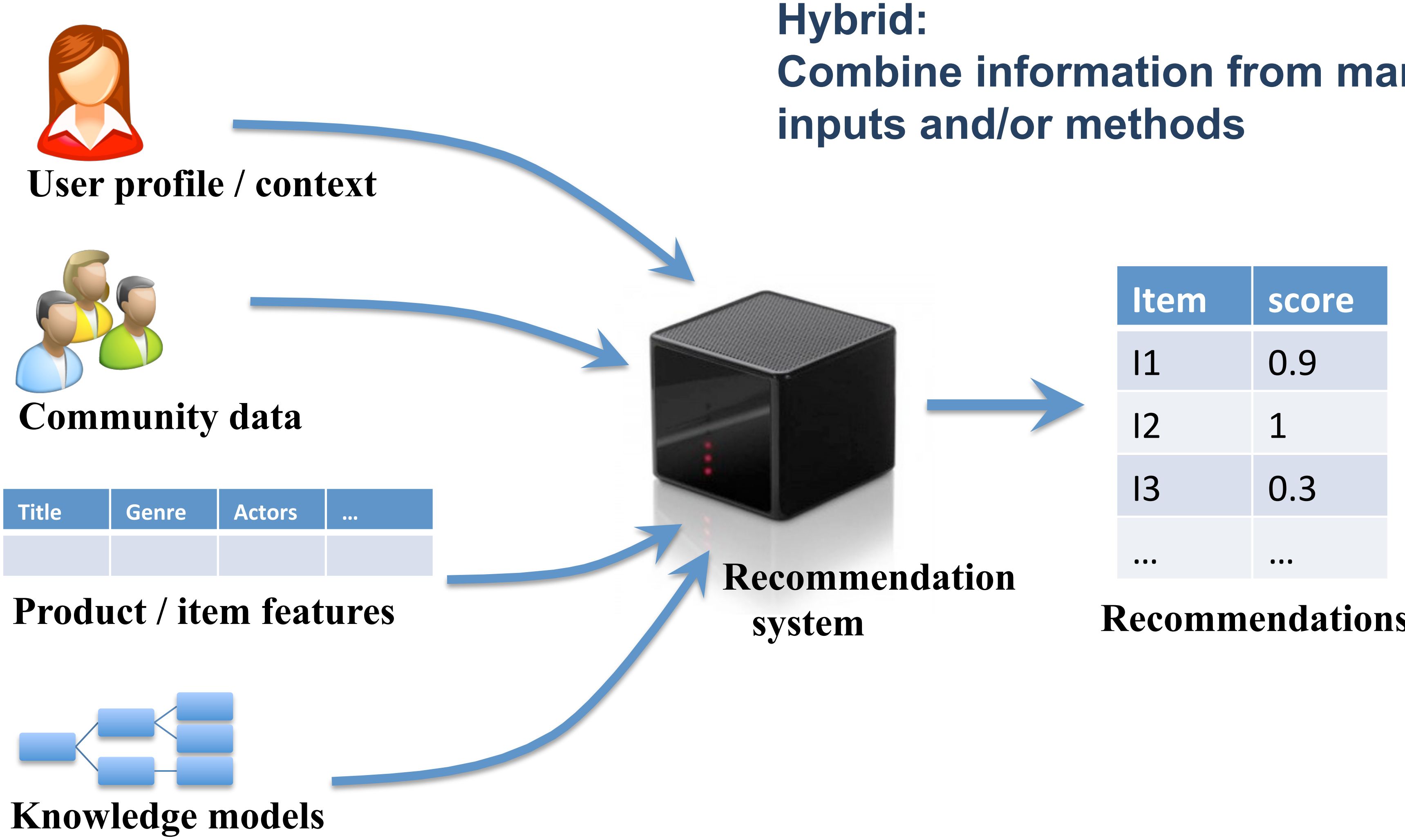
Recommendation system

Item	score
I1	0.9
I2	1
I3	0.3
…	…

Recommendations

Paradigms of recommender systems

Hybrid:
Combine information from many inputs and/or methods



Measuring success

- Prediction perspective
  - Predict to what degree users like the item
  - Most common evaluation for research
  - Regression vs. “top-K” ranking, etc.
- Interaction perspective
  - Promote positive “feeling” in users (“satisfaction”)
  - Educate about the products
  - Persuade users, provide explanations
- “Conversion” perspective
  - Commercial success
  - Increase “hit”, “click-through” rates
  - Optimize sales and profits

Why are recommenders important?

- The "long tail" of product appeal
  - A few items are very popular
  - Most items are popular only with a few people

- Goal: recommend not-widely known items that the user might like!

Collaborative filtering

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	

Collaborative filtering

- Simple approach: standard regression
  - Use “user features” ϕ_u, “item features” ϕ_i
  - Train $f(\phi_u, \phi_i) \approx r_{ui}$
  - Learn “users with my features like items with these features”

- Extreme case: per-user model / per-item model
- Issues: needs lots of side information!

Features:

1 0 1 0 0 …

0 0 1 0 0 …

…

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	

Collaborative filtering

- Example: nearest neighbor methods
  - Which data are “similar”?

- Nearby items? (based on…)

Features:

1 0 1 0 0 …

0 0 1 0 0 …

…

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	

Collaborative filtering

- Example: nearest neighbor methods
  - Which data are “similar”?

- Nearby items? (based on…)

Based on ratings alone?

Find other items that are rated similarly…

Good match on observed ratings

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	

Collaborative filtering

- Which data are "similar"?

- Nearby items?
- Nearby users?
  - Based on user features?
  - Based on ratings?

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	

Collaborative filtering

- Some **very simple** examples
  - All users similar, items not similar?
  - All items similar, users not similar?
  - All users and items are equally similar?

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	

Measuring similarity

- Nearest neighbors depends significantly on distance function
  - "Default": Euclidean distance
- Collaborative filtering:
  - Cosine similarity: $$\frac{x^{(i)} \cdot x^{(j)}}{\|x^{(i)}\| \|x^{(j)}\|}$$ (measures angle between x^i, x^j)
  - Pearson correlation: measure correlation coefficient between x^i, x^j
  - Often perform better in recommender tasks
- Variant: weighted nearest neighbors
  - Average over neighbors is weighted by their similarity
- Note: with ratings, need to deal with missing data!

Nearest-Neighbor methods

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	

Neighbor selection:
Identify movies similar to 1, rated by user 5

Nearest-Neighbor methods

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	

Compute similarity weights:

$s_{13}=0.2$, $s_{16}=0.3$

Nearest-Neighbor methods

users (columns) / **movies** (rows)

movies	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		**2.6**	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	

Predict by taking weighted average:

(0.2*2+0.3*3)/(0.2+0.3)=2.6

Latent space methods

From Y. Koren of BellKor team

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	

$$\underset{N \times D}{X} \approx \underset{N \times K}{U} \; \underset{K \times K}{S} \; \underset{K \times D}{V^T}$$

Latent Space Models

From Y. Koren of BellKor team

Model ratings matrix as "user" and "movie" positions

Infer values from known ratings

Extrapolate to unranked

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

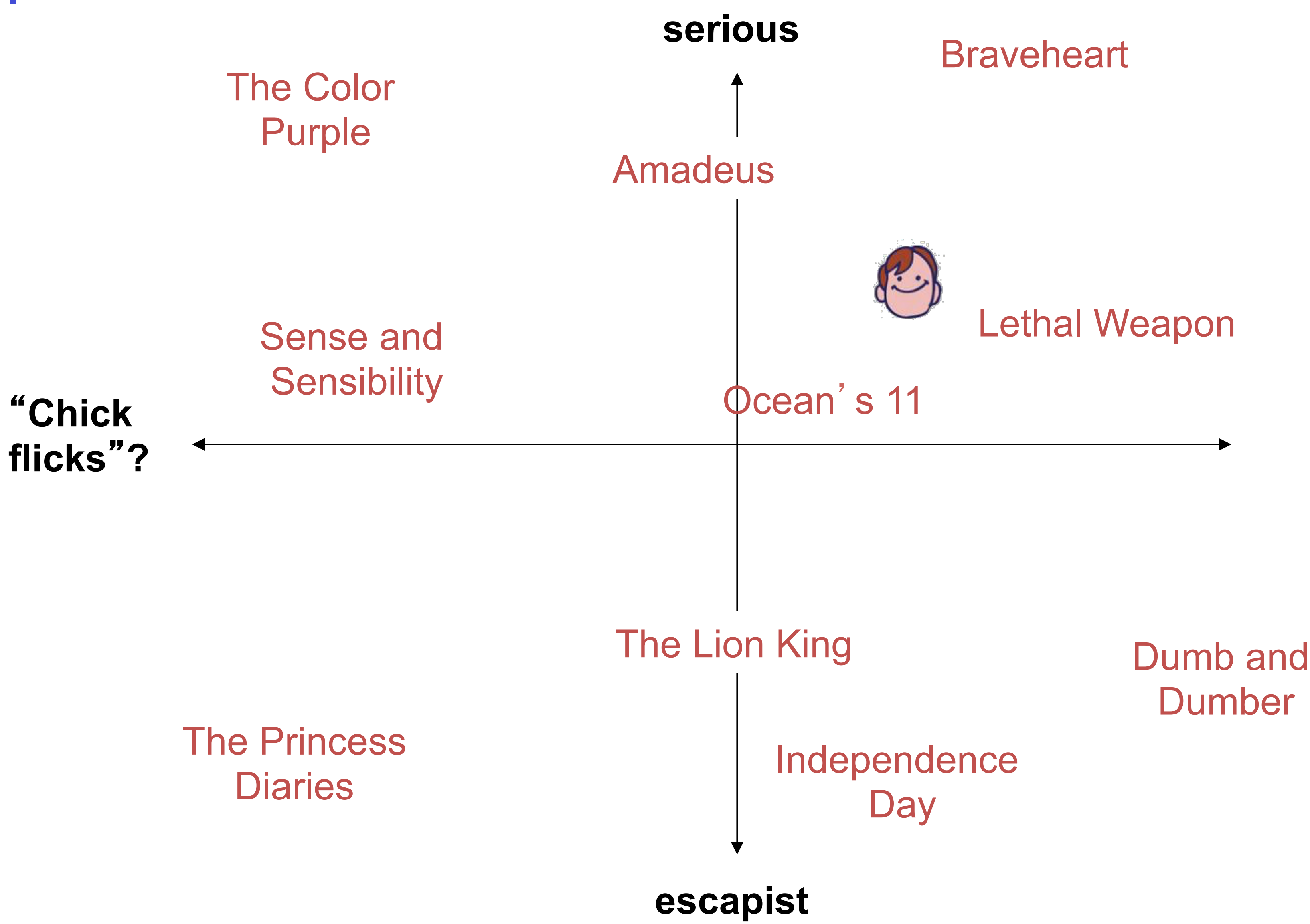
●

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

From Y. Koren of BellKor team



Some SVD dimensions

See timelydevelopment.com

Dimension 1	
Offbeat / Dark-Comedy	Mass-Market / 'Beniffer' Movies
Lost in Translation	Pearl Harbor
The Royal Tenenbaums	Armageddon
Dogville	The Wedding Planner
Eternal Sunshine of the Spotless Mind	Coyote Ugly
Punch-Drunk Love	Miss Congeniality

Dimension 2	
Good	Twisted
VeggieTales: Bible Heroes: Lions	The Saddest Music in the World
The Best of Friends: Season 3	Wake Up
Felicity: Season 2	I Heart Huckabees
Friends: Season 4	Freddy Got Fingered
Friends: Season 5	House of 1

Dimension 3	
What a 10 year old boy would watch	What a liberal woman would watch
Dragon Ball Z: Vol. 17: Super Saiyan	Fahrenheit 9/11
Battle Athletes Victory: Vol. 4: Spaceward Ho!	The Hours
Battle Athletes Victory: Vol. 5: No Looking Back	Going Upriver: The Long War of John Kerry
Battle Athletes Victory: Vol. 7: The Last Dance	Sex and the City: Season 2
Battle Athletes Victory: Vol. 2: Doubt and Conflic	Bowling for Columbine

Latent space models

- Latent representation encodes some "meaning"
- What kind of movie is this? What movies is it similar to?

- Matrix is full of missing data
  - Hard to take SVD directly
  - Typically solve using gradient descent
  - Easy algorithm (see Netflix challenge forum)

```
# for user u, movie m, find the kth 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

- Can be a bit more sophisticated:

  $$r_{iu} \approx \mu + b_u + b_i + \sum_k W_{ik} V_{ku}$$

  - "Overall average rating"
  - "User effect" + "Item effect"
  - Latent space effects (k indexes latent representation)
  - (Saturating non-linearity?)

- Then, just train some loss, e.g. MSE, with SGD
  - Each (user, item, rating) is one data point

Ensembles for recommenders

- Given that we have 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" : train a weighted combination of model predictions