Collaborative Filtering: basic ideas (slides based on chapter 2 of Programming Collective Intelligence book by Toby Segaran)

Fernando Lobo

Data mining

Recommendation Systems

- Use the preferences of a group of people to make recommendations to other people.
- Applications:
  - product recommendation for online shopping (like Amazon)
  - suggesting interesting websites
  - helping people find music and movies

Low-tech solution

- Ask friends for suggestions.
- You want to ask friends that have good taste (they should usually like the same things as you do)
- It's a good approach, but it's limited.
  - Shall we ask all of them?
  - Even if we do so, we don't have that many friends ...
  - But even with lots of friends, how to integrate the results?

Collaborative Filtering

- Searches in a large group of people and finds a smaller set with tastes similar to yours.
- Looks at other things they like and combines them to create a ranked list of suggestions.

Example: rows=People, columns=Movies

	Lady	Snake	Luck	Superman	Dupree	Night
Lisa	2.5	3.5	3.0	3.5	2.5	3.0
Gene	3.0	3.5	1.5	5.0	3.5	3.0
Michael	2.5	3.0		3.5		4.0
Claudia		3.5	3.0	4.0	2.5	4.5
Mike	3.0	4.0	2.0	3.0	2.0	3.0
Jack	3.0	4.0		5.0	3.5	3.0
Toby		4.5		4.0	1.0	

Finding Similar Users

- Need a way to determine how similar people are in their tastes.
- We need a similarity measure (just like in clustering or nearest neighbor algorithms)
- Various similarity measures (distance functions) can be used.

Finding Similar Users

- Similarity measure is usually applied to items (movies) rated in common.
- Example based on Euclidean Distance (gives value between 0 and 1):

$$Similarity(X, Y) = \frac{1}{1 + EuclideanDistance(X, Y)}$$

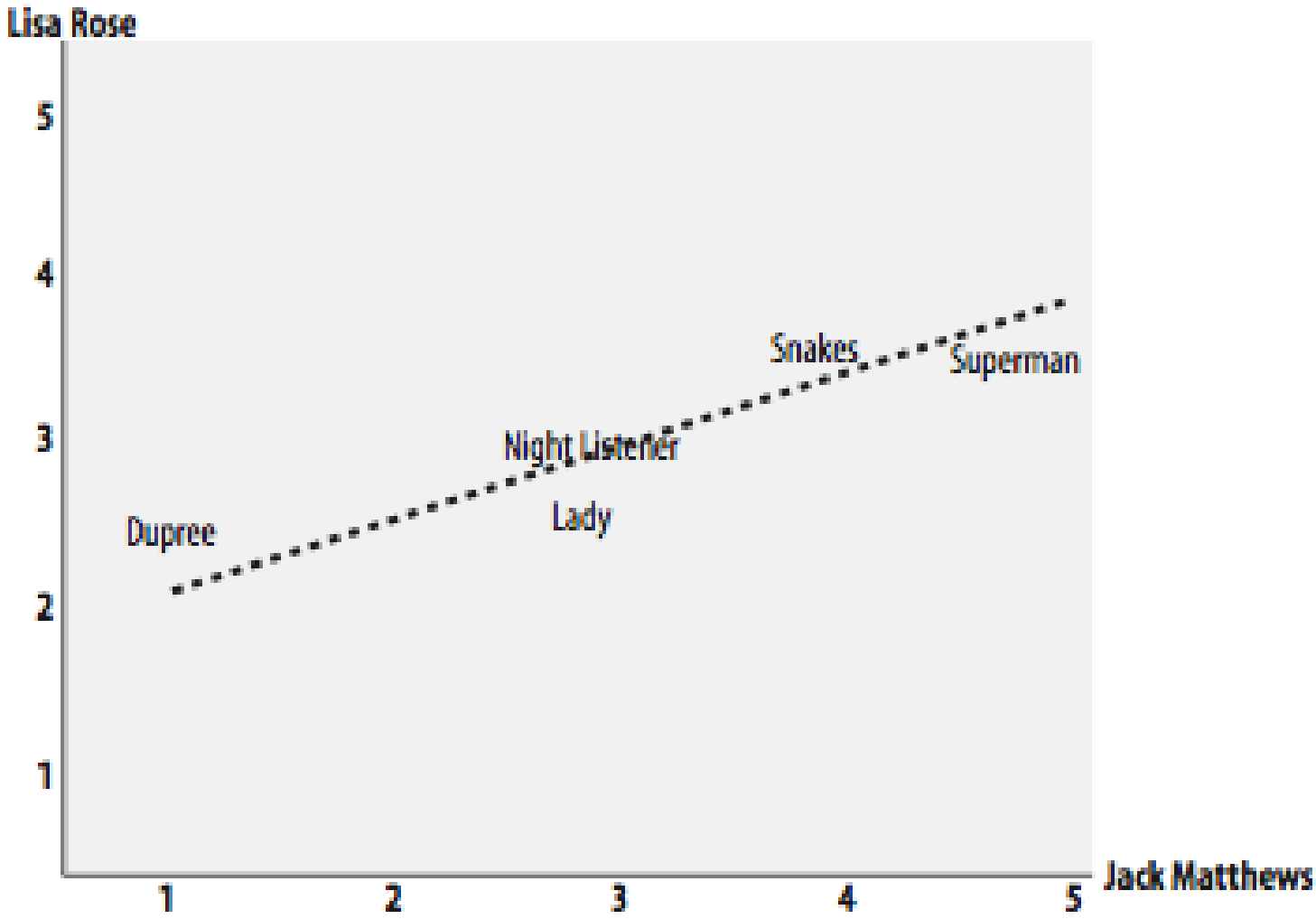
$$Similarity('Michael', 'Claudia') = \frac{1}{1 + \sqrt{(3.0 - 3.5)^2 + (3.5 - 4.0)^2 + (4.0 - 4.5)^2}} = 0.536$$

Another measure: Pearson Correlation Score

$$Pearson(X, Y) = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum {x_i}^2 - (\sum x_i)^2}\sqrt{n\sum {y_i}^2 - (\sum y_i)^2}}$$

- Measures how well two sets of data fit on a straight line.
- Interesting property: corrects grade inflation.
- Jack tends to give higher scores than Lisa, but the line still fits because they have relatively similar preferences.

Lisa and Jack have a high Pearson Correlation Score



Ranking people

- Now we can rank people according to how their tastes are similar to mine (or those of any other person):
  - just compute the similarity score between myself and every other person.
  - this is just a kind of nearest neighbor algorithm.

Recommending Items

- We can find someone with similar tastes to mine.
- But what we want is a movie recommendation.
- Solution: score the items (movies) by doing a weighted average of the score given by the other people.

Example: recommendations for Toby

Table 2-2. Creating recommendations for Toby

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim. Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

Example: recommendations for Toby

- table shows movies Toby hasn't seen (Night, Lady, Luck).
- columns starting with S.x give similarity multiplied by rating.
- need to divide by the sum of the similarities for people that reviewed that movie (Sim. Sum row, in the table)
- the last row shows the scores (recommendations) for movies that Toby hasn't seen.

Matching products

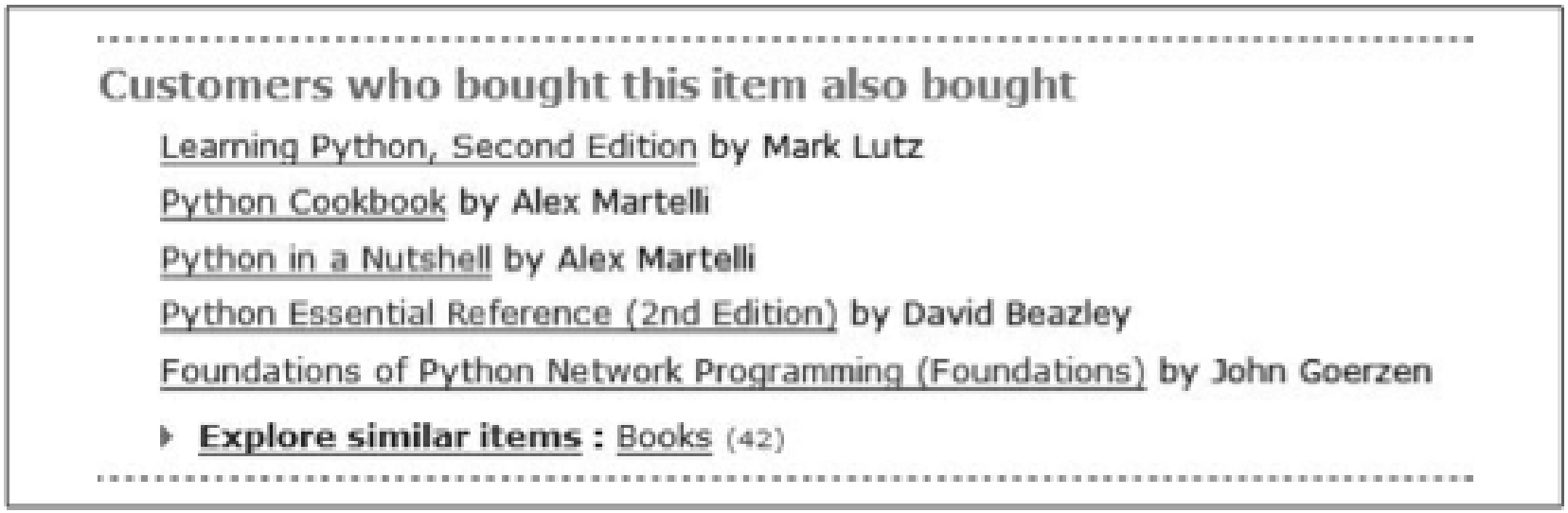


Figure 2-4. Amazon shows products that are similar to Programming Python

Matching products

- We find people with similar taste to ours in order to get movie recommendations.
- We can also find which products (movies) are similar to each other..
- Algorithm is the same (just change the role of people and movies).

More things

- We've just seen a basic method that belong to a class of so-called *memory-based methods*.
- These methods have severe limitations if the data matrix is sparse (the usual case in real applications).
- There are more advanced algorithms to deal with this issue.