Recommendation Systems

Dr. Johan Hagelbäck

johan.hagelback@lnu.se

http://aiguy.org

Recommendation Systems

- The task:
  - Find new items (movies, books, music, …) you may like based on what you have liked before
- Usages:
  - Shopping sites: find items you didn't know existed but that you may like
  - Streaming sites like Netflix/Spotify: find movies/songs you may like but have never heard of
  - Sites like Reddit: suggest new topics that you may like

The most simple approach

- You want to see a new movie, but you don’t know which movies are good
- You ask your friends about what movies they liked that you haven’t seen yet
- You can listen more to friends that you know have good ”taste”, meaning that they usually like the same movies as you do
- Drawbacks:
  - Time consuming
  - Very limited amount of data, meaning that you miss movies none of your friends have seen

Collaborative Filtering

- Collaborative Filtering is a set of techniques for making automatic recommendations for a user
- The term was first used by David Goldberg at Xerox PARC in 1992
- He developed a system for automatic recommendation of documents based on what a user previously has labeled as *interesting* or *uninteresting*

User Preferences

- The first step in developing a Collaborative Filtering system is to store the data, i.e. user preferences
- Preferences must be numeric, for example a scale between 1 and 5 for how good a movie is
- Non-numeric preferences can be translated:
  - Buy: 1, Not buy: 0
  - Buy: 2, Browsed: 1, Not buy: 0
  - Liked: 1, Disliked: -1, No vote: 0
- We will use a small dataset consisting of seven users and six movies:

The dataset

Movie	Lisa	Gene	Michael	Claudia	Mick	Jack	Toby
Lady in the Water	2.5	3.0	2.5		3.0	3.0	
Snakes on a Plane	3.5	3.5	3.0	3.5	4.0	4.0	4.5
Just My Luck	3.0	1.5		3.0	2.0		
Superman Returns	3.5	5.0	3.5	4.0	3.0	5.0	4.0
You, Me and Dupree	2.5	3.5		2.5	2.0	3.5	1.0
The Night Listener	3.0	3.0	4.0	4.5	3.0	3.0	

How to find a new movie?

- The most simple approach is to average the score on movies you haven't seen:

Movie	Lisa	Gene	Mike	Claudia	Mick	Jack	Toby
Lady in the Water	2.5	3.0	2.5		3.0	3.0	**2.8**
Snakes on a Plane	3.5	3.5	3.0	3.5	4.0	4.0	4.5
Just My Luck	3.0	1.5		3.0	2.0		**2.38**
Superman Returns	3.5	5.0	3.5	4.0	3.0	5.0	4.0
You, Me and Dupree	2.5	3.5		2.5	2.0	3.5	1.0
The Night Listener	3.0	3.0	4.0	4.5	3.0	3.0	**3.42**

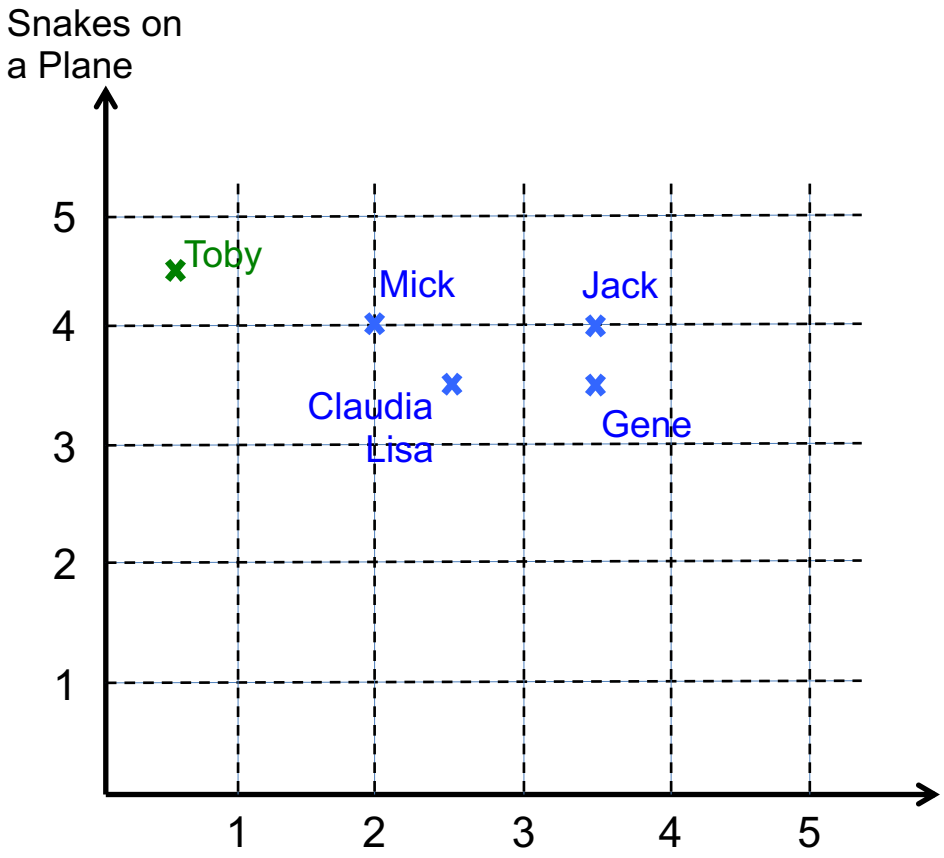
But that does not mean that the other users that have given high ratings for Night Listener has the same taste as Toby...

A better approach

Finding Similar Users

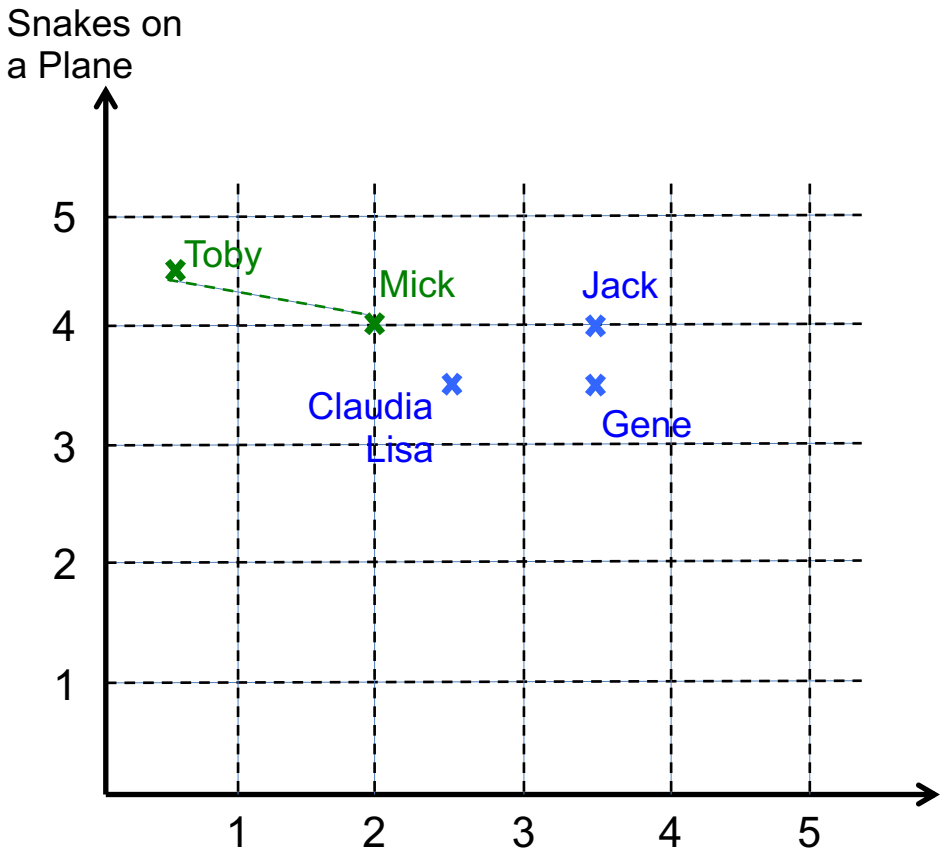
- A better approach is to find users similar to yourself
- This is done by comparing every user with every other user and calculate a *similarity score*
- There are many ways to calculate similarity
- Here we will take a look at two of them:
  - *Euclidean Distance*
  - *Pearson Correlation*

Euclidean Distance



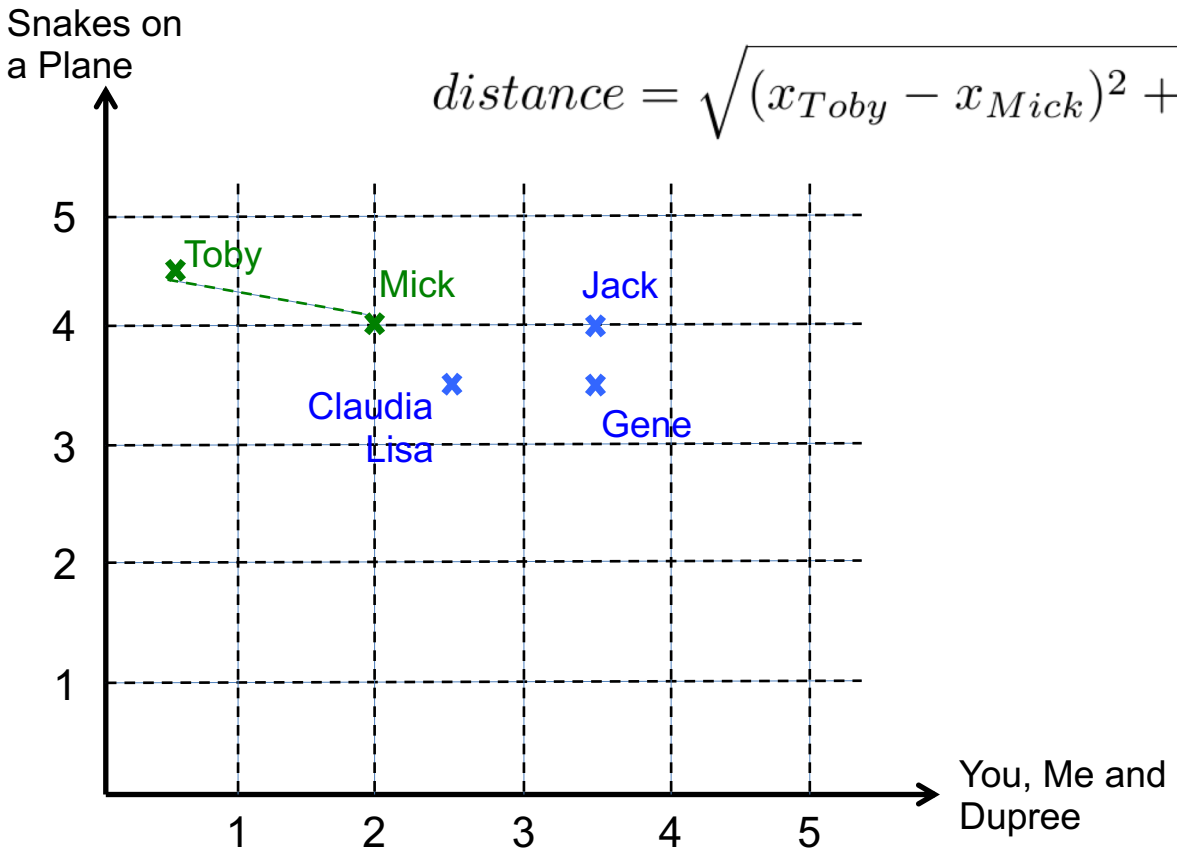
Question: which user is the best match for Toby?

Euclidean Distance



For user Toby,
Mick is the closest
match!

Euclidean Distance



Euclidean Distance

- The distance is however smaller for people who are more similar, but we want the opposite!
- Therefore we have to invert it, and add 1 to avoid division by zero:

$$similarity = \frac{1}{1 + \sqrt{(x_{Toby} - x_{Mick})^2 + (y_{Toby} - y_{Mick})^2}}$$

Euclidean Distance

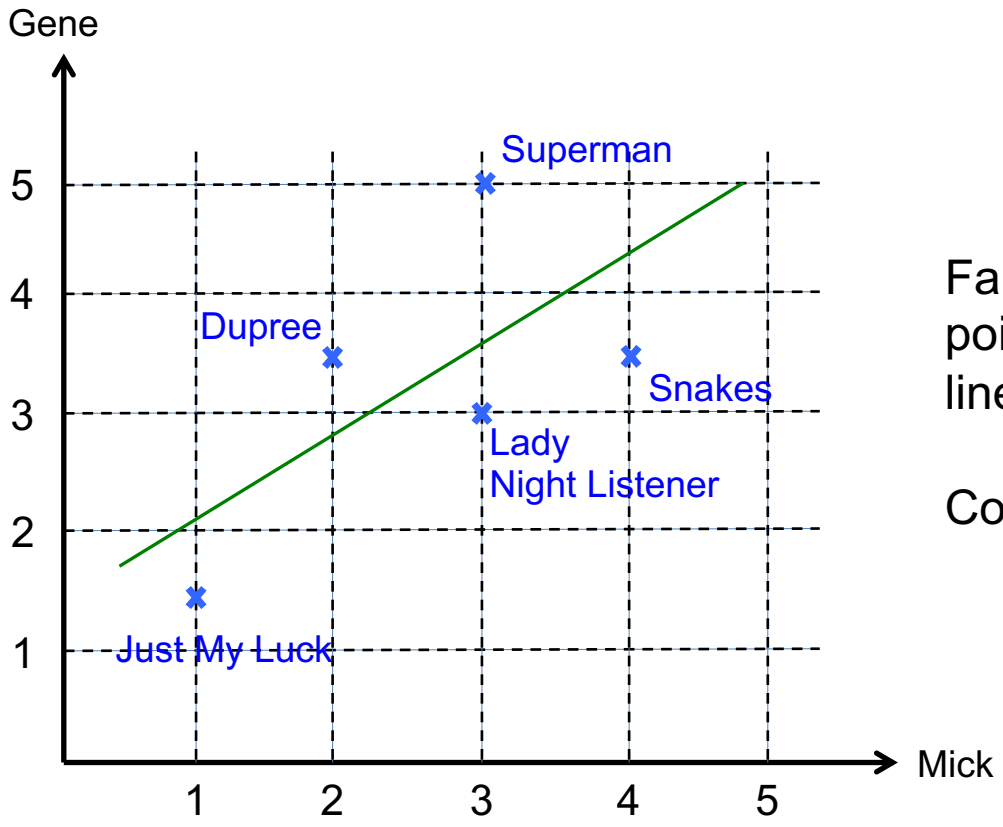
```
float euclidean(User A, User B)
  //Init variables
  sim=0
  //Counter for number of matching products
  n = 0
  //Iterate over all rating combinations
  for (Rating rA : A.rating)
    for (Rating rB : B.rating)
      if (rA == rB)
        sim += (rA.score – rB.score)**2 //a*a
        n += 1
  //No ratings in common – return 0
  if (n == 0)
    return 0
  //Calculate inverted score
  inv = 1 / (1 + sim)
  return inv
```

Note! For performance reasons we skip the square root calculation. This will not affect the relative distance between users!

Pearson Correlation Score

- PCS is a more sophisticated way to calculate similarity
- The correlation coefficient is a measure of how well two sets of data fit on a straight line
- If all data points fit on the straight line, we have a perfect match resulting on correlation score 1
- PCS tends to give better score for data that isn't well normalized, for example if a harsh user routinely give lower scores than the other users
- It is also more robust to *grade inflation*, where one user consistently gives higher (or lower) scores than the other users

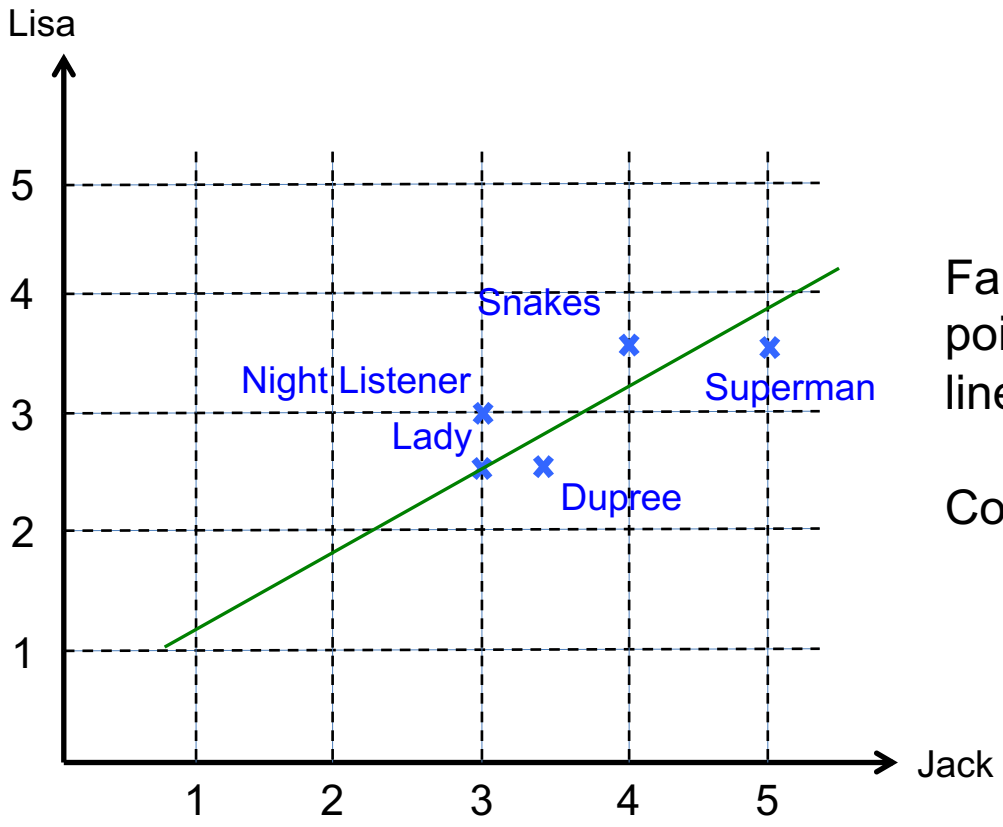
Pearson Correlation Score



Fairly bad fit: the data points are far from the line.

Correlation: 0.4

Pearson Correlation Score



Fairly good fit: the data points are close to the line.

Correlation: 0.75

Pearson Correlation Score

- The Pearson Correlation score *r* is calculated as:

$$num = pSum - (\frac{sum1 \cdot sum2}{n})$$

$$den = \sqrt{(sum1sq - \frac{sum1^2}{n}) \cdot (sum2sq - \frac{sum2^2}{n})}$$

$$r = \frac{num}{den}$$

n	Number of ratings in common
sum1, sum2	Sum of ratings for user 1 and user 2
sum1sq, sum2sq	Sum of squared ratings for user 1 and user 2
pSum	Product of ratings of user 1 and user 2

Pearson Correlation Score

```
float pearson(User A, User B)
  //Init variables
  sum1=0, sum2=0, sum1sq=0, sum2sq=0, pSum=0
  //Counter for number of matching products
  n = 0
  //Iterate over all rating combinations
  for (Rating rA : A.rating)
    for (Rating rB : B.rating)
      if (rA == rB)
        sum1 += rA.score //sum of ratings for user A
        sum2 += rB.score //sum of ratings for user B
        sum1sq += rA.score**2 //sum of squared ratings for A
        sum2sq += rB.score**2 //sum of squared ratings for B
        pSum += rA.score * rB.score //product of ratings from A and B
        n += 1 //number of ratings in common

  //No ratings in common – return 0
  if (n == 0)
    return 0
  //Calculate Pearson
  num = pSum – (sum1 * sum2 / n)
  den = sqrt((sum1sq – sum1**2 / n) * (sum2sq – sum2**2 / n))
  return num/den
```

Find top three matches for a user

- This can be done as follows:
  - Calculate the similarity score between the user and all other users
  - Store the scores in a list
  - Sort the list in descending order (highest scores first)
  - Return the first 3 entries in the list

Find top three matches for a user

User: 7: Toby Similarity: Euclidean Results: 3

Find top matching users

Name	ID	Score
Mick	5	0.3077
Mike	3	0.2857
Claudia	4	0.2353

User: 7: Toby Similarity: Pearson Results: 3

Find top matching users

Name	ID	Score
Lisa	1	0.9912
Mick	5	0.9245
Claudia	4	0.8934

Which similarity metric to use?

- There are many other metrics than the two mentioned:
  - *Manhattan Distance*
  - *Jaccard Coefficient*
  - …
- There is no universal answer to which one is the best to use
- It depends on the application
- Try at least *Euclidean* and *Pearson* to see which one works best in your case!

Finding recommended movies

Recommending Items

- Finding similar users is just the first step
- What we really want to know is a movie recommendation
- To do this, we need to calculate a weighted score for each user and movie

- Task: - find a movie recommendation for Toby

- We create a table with user similarities and weighted rating scores:

Weighted Scores

- Calculate the weighted scores as the similarity between Toby and the other users multiplied by the other users ratings (using Pearson):

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Gene	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Mike	-1.00	4		2.5			
Claudia	0.89	4.5	4.02			3.0	2.68
Mick	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Jack	0.66	3.0	1.99	3.0	1.99		

Note that we don't include Mike in the calculations since he has a similarity of 0 or below for Toby

Weighted Scores

- Calculate the sum of weighted scores:

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Gene	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Mike	-1.00	4		2.5			
Claudia	0.89	4.5	4.02			3.0	2.68
Mick	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Jack	0.66	3.0	1.99	3.0	1.99		
Σws			12.89		8.38		8.07

Weighted Scores

- Calculate the sum of similarity for all users who has rated each movie:

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Gene	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Mike	-1.00	4		2.5			
Claudia	0.89	4.5	4.02			3.0	2.68
Mick	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Jack	0.66	3.0	1.99	3.0	1.99		
Σws			12.89		8.38		8.07
Σsim			3.84		2.95		3.18

Weighted Scores

- Divide the sum of weighted scores with sum of similarity:

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Gene	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Mike	-1.00	4		2.5			
Claudia	0.89	4.5	4.02			3.0	2.68
Mick	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Jack	0.66	3.0	1.99	3.0	1.99		
Σws			12.89		8.38		8.07
Σsim			3.84		2.95		3.18
Σws / Σsim			3.35		2.83		2.53

Which movie shall Toby see?

- *The Night Listener* has the highest score and is the top recommendation:

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Gene	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Mike	-1.00	4		2.5			
Claudia	0.89	4.5	4.02			3.0	2.68
Mick	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Jack	0.66	3.0	1.99	3.0	1.99		
Σws			12.89		8.38		8.07
Σsim			3.84		2.95		3.18
Σws / Σsim			3.35		2.83		2.53

Euclidean Distance

- Same table using Euclidean Distance as similarity score:

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.22	3.0	0.66	2.5	0.55	3.0	0.66
Gene	0.11	3.0	0.33	3.0	0.33	1.5	0.17
Mike	0.29	4	1.16	2.5	0.73		
Claudia	0.24	4.5	1.08			3.0	0.72
Mick	0.31	3.0	0.93	3.0	0.93	2.0	0.62
Jack	0.12	3.0	0.36	3.0	0.36		
Σws			4.52		2.90		2.17
Σsim			1.29		1.05		0.88
Σws / Σsim			3.50		2.76		2.46

Now Mike is included since similarity is not 0 or below when Euclidean is used

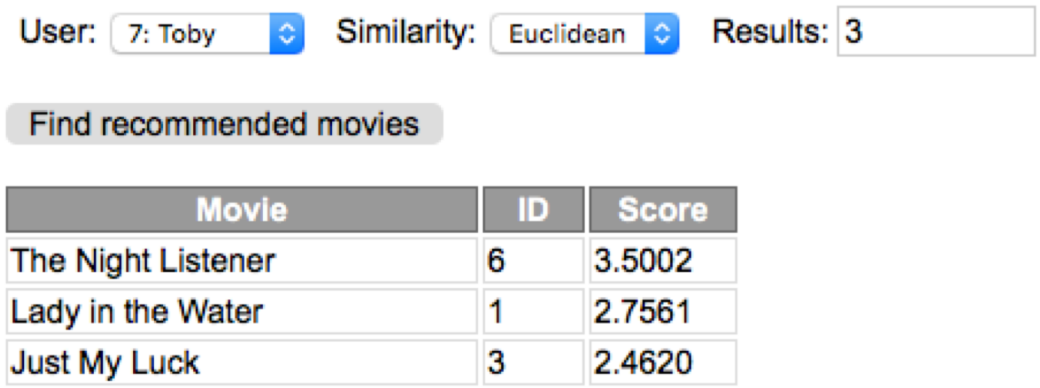
Which movie shall Toby see?

- *The Night Listener* is still the top recommendation for Toby:

User	Similarity Toby	Night	W Score, Night	Lady	W Score, Lady	Luck	W Score, Luck
Lisa	0.22	3.0	0.66	2.5	0.55	3.0	0.66
Gene	0.11	3.0	0.33	3.0	0.33	1.5	0.17
Mike	0.29	4	1.16	2.5	0.73		
Claudia	0.24	4.5	1.08			3.0	0.72
Mick	0.31	3.0	0.93	3.0	0.93	2.0	0.62
Jack	0.12	3.0	0.36	3.0	0.36		
Σws			4.52		2.90		2.17
Σsim			1.29		1.05		0.88
Σws / Σsim			3.50		2.76		2.46

Web application

- This is an example of how the recommendation system can look like when implemented as a web application:



Find matching movies

Finding matching movies

- It is also possible to find top matching movies for a movie
- To do this we need to transpose the dataset so movies replaces users:

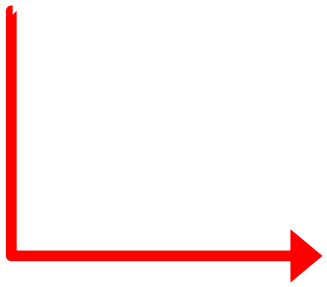
From:

Lisa: [(Lady in the Water : 2.5), (Snakes on a Plane : 3.5)]
Gene: [(Lady in the Water : 3.0), (Snakes on a Plane : 3.5)]

To:

Lady in the Water: [(Lisa : 2.5), (Gene : 3.0)]
Snakes on a Plane: [(Lisa : 3.5), (Gene : 3.5)]

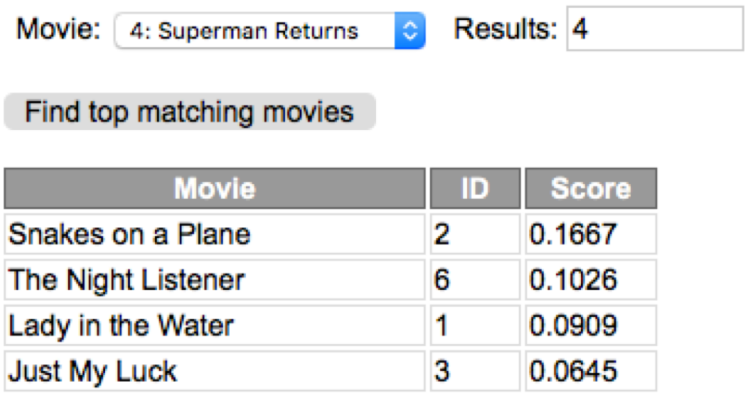
Movie	Lisa	Gene	Michael	Claudia	Mick	Jack	Toby
Lady in the Water	2.5	3.0	2.5		3.0	3.0	
Snakes on a Plane	3.5	3.5	3.0	3.5	4.0	4.0	4.5
Just My Luck	3.0	1.5		3.0	2.0		
Superman Returns	3.5	5.0	3.5	4.0	3.0	5.0	4.0
You, Me and Dupree	2.5	3.5		2.5	2.0	3.5	1.0
The Night Listener	3.0	3.0	4.0	4.5	3.0	3.0	



User	Lady	Snakes	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
Mick	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 matching products

- We can then use the same method as we previously used to find similar users to find matching movies for a movie:



Movie	ID	Score
Snakes on a Plane	2	0.1667
The Night Listener	6	0.1026
Lady in the Water	1	0.0909
Just My Luck	3	0.0645

- Transposing the data set is however a rather slow operation if the data set is large

User-based collaborative filtering

- To find recommendations for a user, the approach we have used so far requires calculating similarity between a user and all other users
- This approach is called *user-based collaborative filtering*
- This works for small data sets, but will be very ineffective for large data sets
- Also, if there are many users there is most likely very little overlap between most users
- There is another approach we can use:

Item-based collaborative filtering

- The other approach is called *item-based collaborative filtering*
- We can make an assumption that the comparison between movies will not change as much as comparisons between users
- We can therefore pre-calculate and store the top N matching movies for each movie in a new data set
- This requires that we transpose the dataset so we can find top matching movies for each movie

Recommending movies again

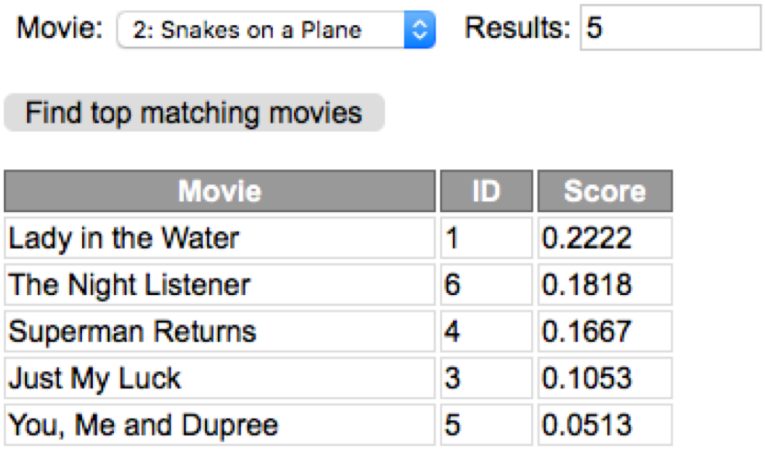
- We can then use the pre-generated dataset to find recommended movies
- Now, similarity between users are not involved at all
- Instead we use the pre-calculated similarity score between movies and generate a similar table:

Recommending movies, IB

- First, fill in similarity between movies using the pre-generated matching movies table:

Movie	Rating (Toby)	Sim Night	WR Night	Sim Lady	WR Lady	Sim Luck	WR Luck
Snakes	4.5	0.182		0.222		0.105	
Superman	4.0	0.103		0.091		0.065	
Dupree	1.0	0.148		0.4		0.182	

- The first row is similarity between *Snakes on a Plane* and the other three movies:



Movie	ID	Score
Lady in the Water	1	0.2222
The Night Listener	6	0.1818
Superman Returns	4	0.1667
Just My Luck	3	0.1053
You, Me and Dupree	5	0.0513

Recommending movies, IB

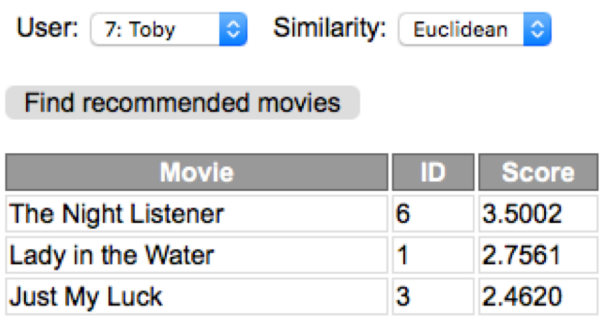
- Calculate sum of weighted ratings and similarities:

Movie	Rating (Toby)	Sim Night	WR Night	Sim Lady	WR Lady	Sim Luck	WR Luck
Snakes	4.5	0.182	0.818	0.222	0.999	0.105	0.474
Superman	4.0	0.103	0.412	0.091	0.363	0.065	0.258
Dupree	1.0	0.148	0.148	0.4	0.4	0.182	0.182
Σwr			1.378		1.764		0.914
Σsim		0.433		0.713		0.352	
Σwr / Σsim			3.183		2.598		2.473

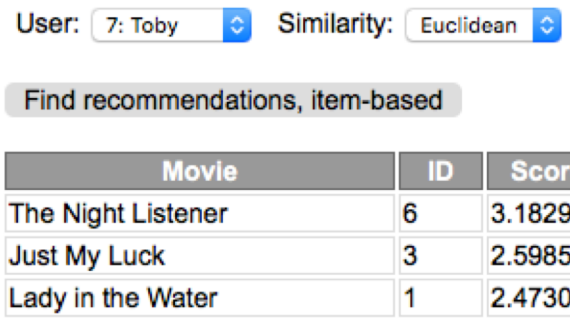
- *The Night Listener* is still the recommended movie for Toby

Comparison

- The results from User-Based and Item-Based Collaborative Filtering differs slightly:



Movie	ID	Score
The Night Listener	6	3.5002
Lady in the Water	1	2.7561
Just My Luck	3	2.4620



Movie	ID	Score
The Night Listener	6	3.1829
Just My Luck	3	2.5985
Lady in the Water	1	2.4730

User-Based or Item-Based?

- Getting a list of recommendations is faster for item-based for large datasets
- The drawback is that the similar items table must be generated and updated regularly, which is a very slow operation
- Item-Based is usually more accurate on sparse datasets, i.e. datasets with little overlap between users
- Our dataset is however dense; every user has rated nearly every movie

Real-world data set

- The *GroupLens* project at University of Minnesota has collected and generated several datasets for public use
- The dataset that is most interesting for us is *MovieLens*:
  - https://grouplens.org/datasets/movielens/
- The dataset is generated from the movie recommendation service movielens.org
- The dataset comes in two sizes, the full dataset with 27 million ratings or a smaller dataset with 100 000 ratings

MovieLens dataset

User: 342 Results: 5 Min no ratings: 25

Find recommended movies

Euclidean			
Movie	**ID**	**Score**	**Ratings**
Wallace & Gromit: The Best of Aardman Animation (1996)	720	4.5882	27
Emma (1996)	838	4.526	30
Gods Must Be Crazy, The (1980)	2150	4.4635	28
Cool Hand Luke (1967)	1276	4.4304	57
Shawshank Redemption, The (1994)	318	4.4212	317

Finding recommendations took 2.702 sec

Pearson			
Movie	**ID**	**Score**	**Ratings**
Shawshank Redemption, The (1994)	318	4.5663	317
Patton (1970)	1272	4.5376	33
Grand Day Out with Wallace and Gromit, A (1989)	1223	4.4427	28
Pulp Fiction (1994)	296	4.388	307
Cool Hand Luke (1967)	1276	4.3807	57

Finding recommendations took 2.551 sec

Recommendation Systems

Dr. Johan Hagelbäck

johan.hagelback@lnu.se

http://aiguy.org