COMP6237 Data Mining
Lecture 1: Recommendation Systems

Zhiwu Huang

Zhiwu.Huang@soton.ac.uk

Lecturer (Assistant Professor) @ VLC of ECS

University of Southampton

Lecture slides available here:

http://comp6237.ecs.soton.ac.uk/zh.html

(Thanks to Prof. Jonathon Hare and Dr. Jo Grundy for providing the lecture materials used to develop the slides.)

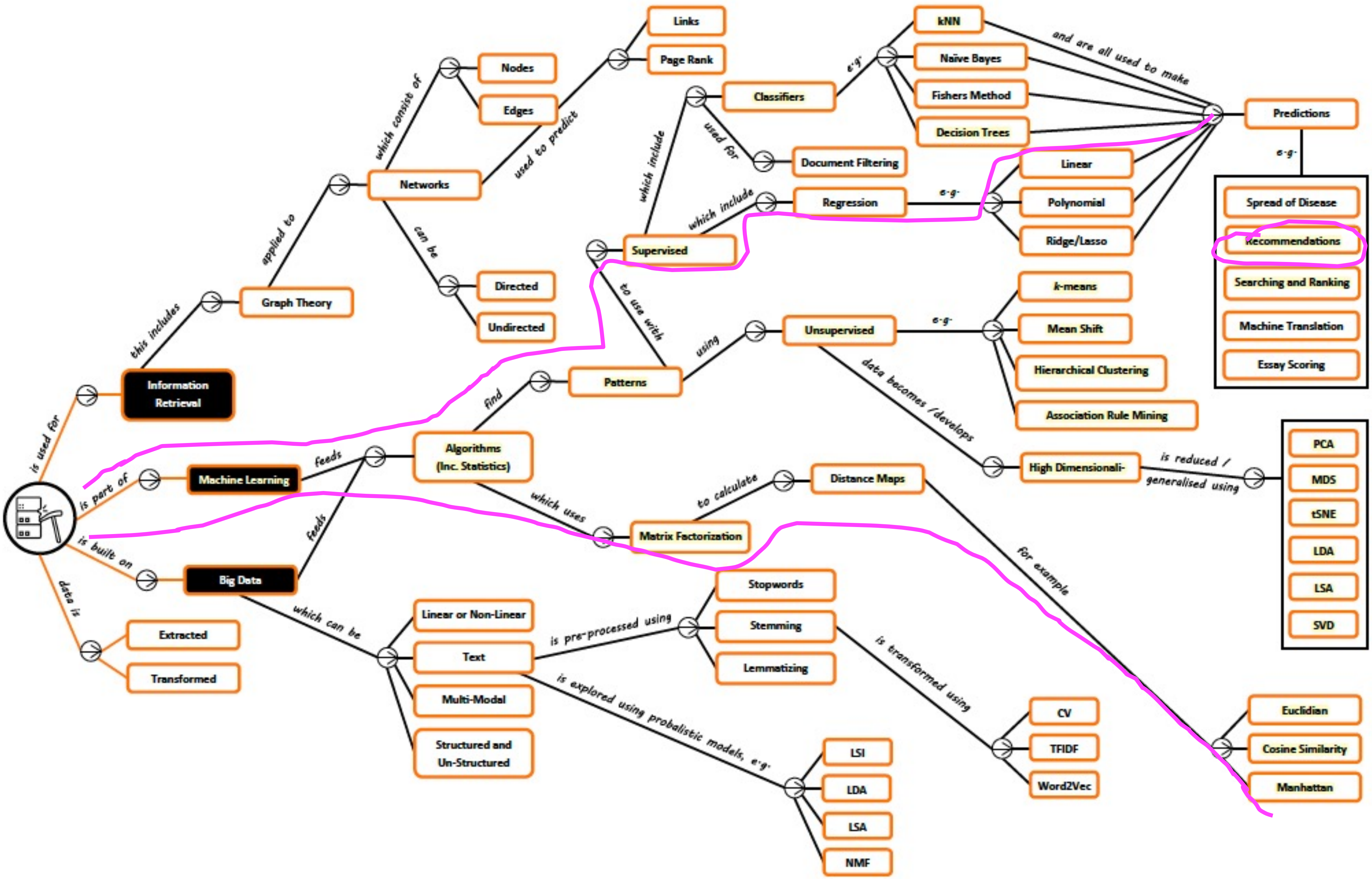
My Lectures

26-Feb	5	Zhiwu	Making Recommendations
27-Feb		Zhiwu	Finding Groups
29-Feb		Zhiwu	Covariance
04-Mar	6	Zhiwu	Embedding Data
05-Mar		Zhiwu	Search
07-Mar		Zhiwu	Document filtering
11-Mar	7	Zhiwu	Modelling with decision trees
12-Mar		Zhiwu	Modelling Prices & Nearest Neighbours
14-Mar		Zhiwu	Market Basket Analysis
18-Mar	8	Zhiwu & Shoaib & Markus	*Group coursework presentations*
19-Mar		Zhiwu & Shoaib & Markus	*Group coursework presentations*
21-Mar		Zhiwu & Shoaib & Markus	*Group coursework presentations*
22-Mar		Zhiwu & Shoaib & Markus	*Group coursework presentations*
Easter			
22-Apr	9	Zhiwu	Semantic Spaces & Latent Semantics
23-Apr		Zhiwu	Topic Modelling
25-Apr		Zhiwu	Outlier Detection

Four Key Slides for each lecture:

Roadmap + Textbook + Overview + Learning Outcomes (Exercise/Exam)

Recommendation Systems – **Roadmap**



Recommendation Systems – **Textbook**

Chapter 9

Recommendation Systems

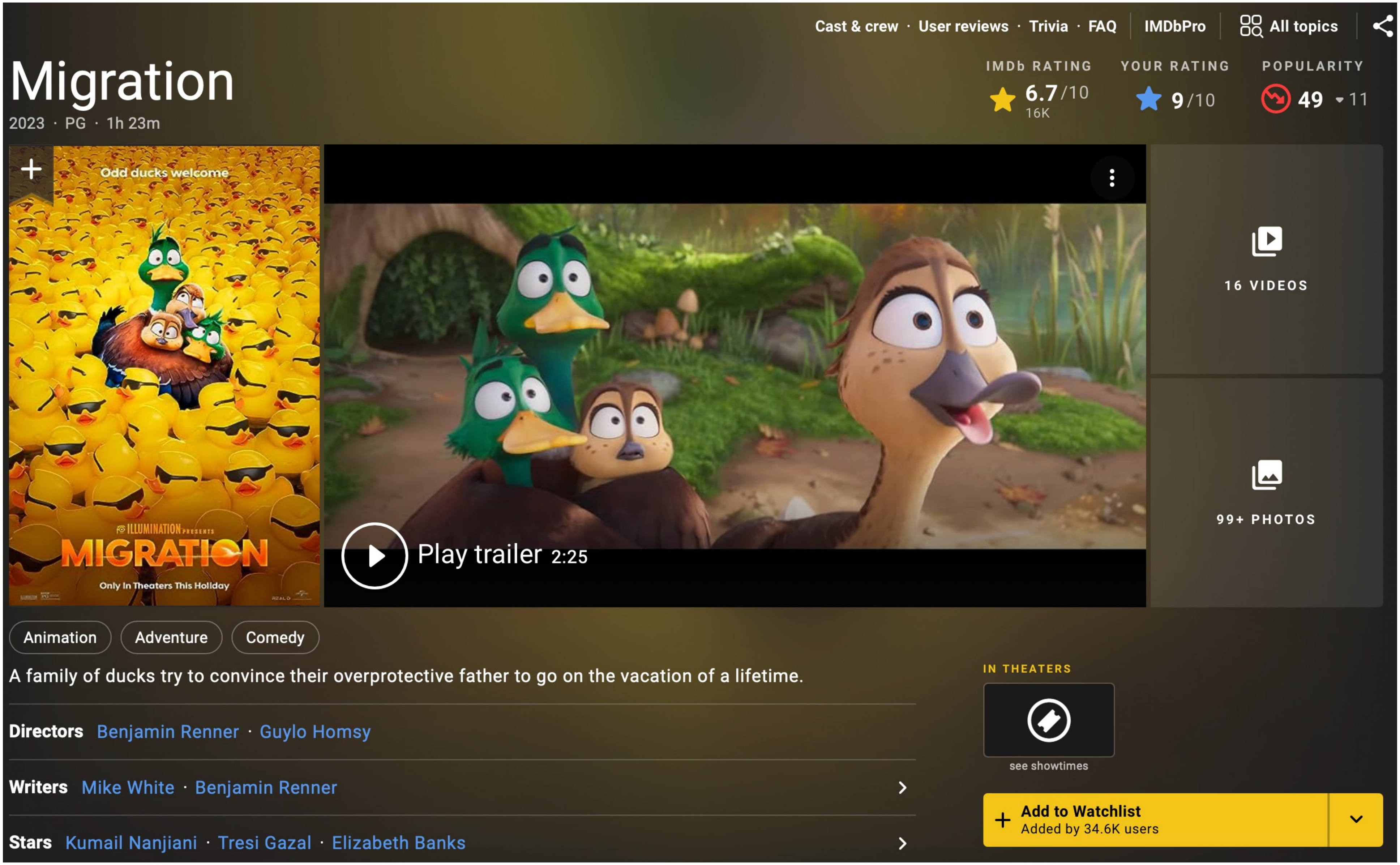
There is an extensive class of Web applications that involve predicting user responses to options. Such a facility is called a *recommendation system*. We shall begin this chapter with a survey of the most important examples of these systems. However, to bring the problem into focus, two good examples of recommendation systems are:

1. Offering news articles to on-line newspaper readers, based on a prediction of reader interests.

2. Offering customers of an on-line retailer suggestions about what they might like to buy, based on their past history of purchases and/or product searches.

- Mining of Massive Datasets J. Leskovec *et al*

Recommendation Systems – **Overview (1/4)**



https://www.imdb.com/title/tt6495056/, Release date: 2 February 2024 (UK)

Recommendation Systems – **Overview (2/4)**

10/10

Great for adults and kids - clean and entertaining family film
m_al_saeed 16 December 2023

The animation is colorful, action starts relatively early and is well spaced-out. The jokes were appropriate. My kids are aged 10 and 14, they both enjoyed it. There were kids of all ages in the theatre and I could hear laughs all the time. Perfect for the holidays and also has a great soundtrack.

The storyline is obvious in the title, but the journey was full of surprises enough to keep you on the edge. There were useful lessons to be learned and a morale to the story.

The last film I had seen at the cinema was Super Mario and "Migration" has delivered the same clean family entertainment that we have been missing.

Highly recommended.

40 out of 53 found this helpful. Was this review helpful? Yes No

Report this | Permalink

2/10

Unoriginal and boring
sjo-15 5 February 2024

There was little here to keep my interest.

David Mitchell was funny and well cast. He really was like a tonic when he came along.

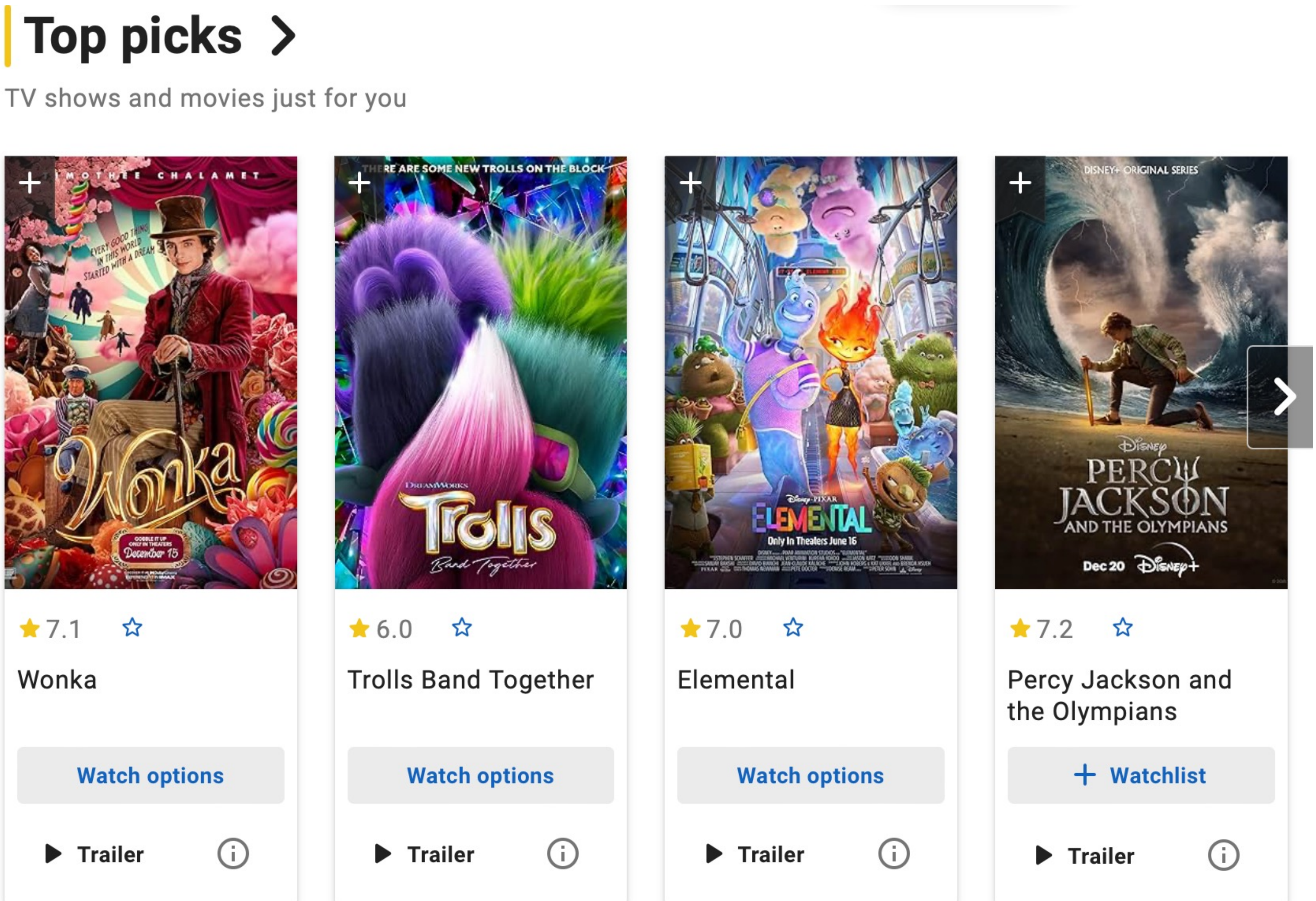
Otherwise, seen it all done before and better.

The story was formulaic, short on laughs and predictable.

Felt long at 90 minutes. Judging by the amount of phones I saw being looked at I wasn't the only adult wondering why they spent their money on this.

https://www.imdb.com/title/tt6495056/

Recommendation Systems – **Overview (3/4)**



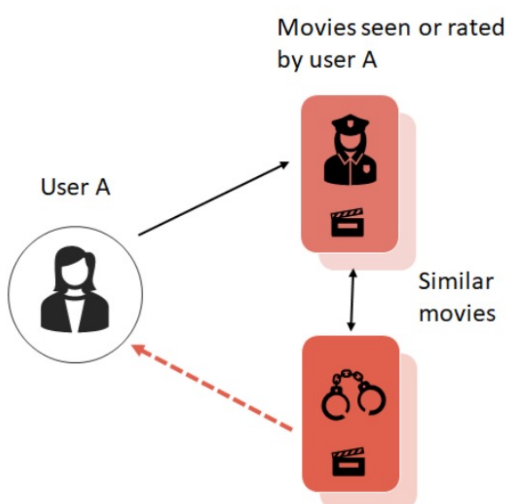
https://www.imdb.com

Recommendation Systems – **Overview (4/4)**

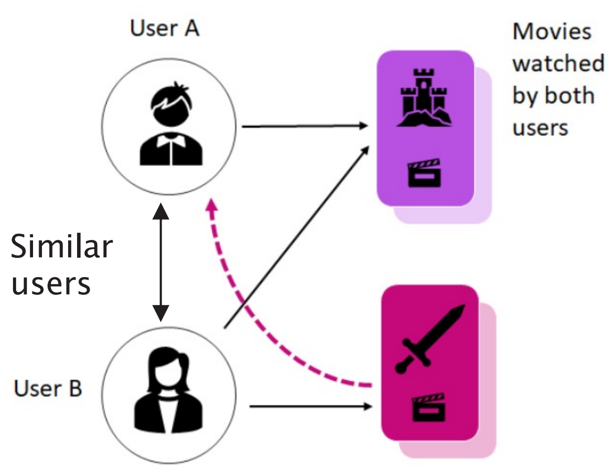
Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Battles	1	5			0.1	1
Dark Knight	5			?	0.1	1

https://en.wikipedia.org/wiki/The_Dark_Knight

Content-based Filtering
Movie recommendations to user A



Collaborative Filtering
Movies watched by user A are recommended to user B



Remaining problems:
- How to represent the item/user data?
- How to save the data?
- How to compare the data (user/item)?

Source: https://www.relataly.com/

Recommendation Systems – **Learning Outcomes**

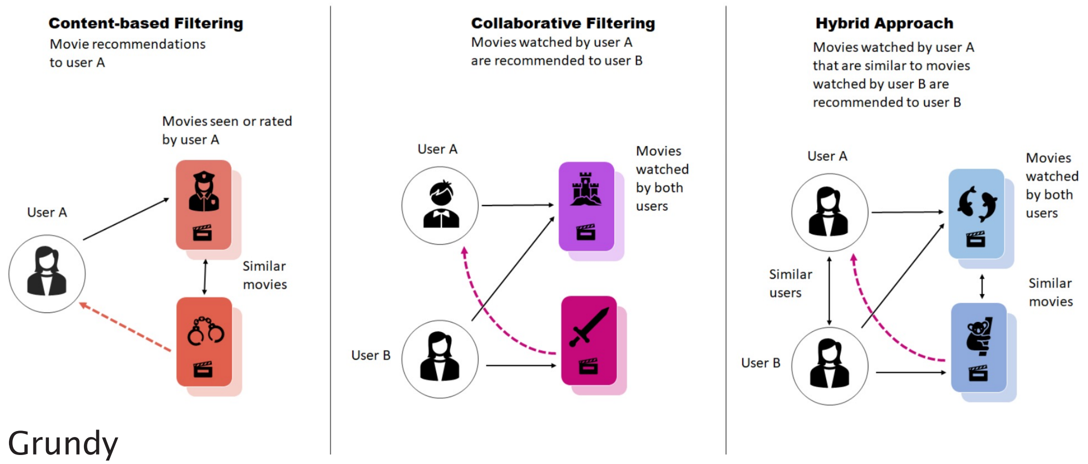
- **LO1:** Mastering fundamental concepts and mathematical calculations of content-based and user-based/item-based collaborative filtering approaches, such as (exam)
  - Measuring distances/similarities between users or items
  - Computing the resulting matrix using Compressed Row/Column Storage
  - Predicting the missing rating using content-based approach
  - Calculating the predicted rating with user-based collaborative filtering approach
- **LO2**: Implement basic algorithms using Python (coursework)

Assessment hints: Multi-choice Questions (single answer: concepts, calculation etc)

- *Textbook Exercises: textbooks (Programming + Mining)*
- *Other Exercises: https://www-users.cse.umn.edu/~kumar001/dmbook/sol.pdf*
- *ChatGPT or other AI-based techs*

Recommendation Systems – Algorithms

- **Content-based systems** (e.g., Netfix) examine properties of the items recommended. For instance, if a Netflix user has watched many cowboy movies, then recommend a movie classified in the database as having the “cowboy” genre.
- **Collaborative filtering systems** (e.g., Facebook Amazon) recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users.
- **Hybrid recommender systems** (e.g., Netflix) Uses combinations of different approaches



We will learn Content-based and collaborative filtering systems in this lecture

Credit: Jo Grundy

Recommendation Systems– Content based approach

Can use a vector of features for each film, eg romance, action

Content-based Features/data

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Each film can be described by the vector $X = [1 \quad x_1 \quad x_2]$ (1 is for the bias term)

Learn 2D parameter vector θ, where $\theta^T X$ gives the number of stars for each user.
$\theta = [0 \quad 5 \quad 0]$ for someone who really likes romance films

Credit: Jo Grundy

Recommendation Systems– Content based approach

Use Linear Regression to find user parameter vector θ where m is the number of films rated by that user

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2 \tag{1}$$

Can also use Bayesian classifiers, MLPs, etc.

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Credit: Jo Grundy

Recommendation Systems– Content based approach

Linear Regression

Can use a vector of features for each film, eg romance, action

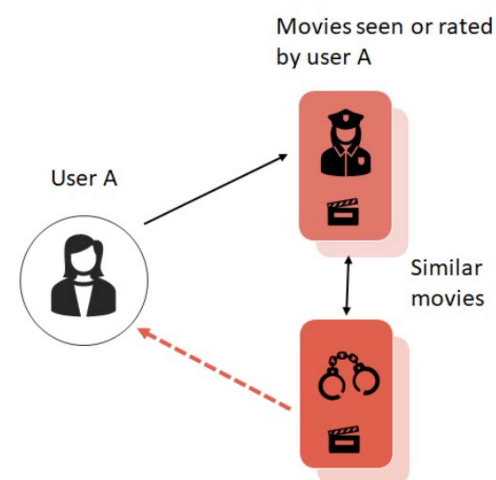
Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Take the user 'Alice' as an example

$y_1 = 4, y_3 = 5, y_4 = 1$

$X_1 = [1, x_1, x_2] = [1, 1, 0.1], X_3 = [1, 0.2, 0.9], X_4 = [1, 0.1, 1],$

$\theta = [b, w_1, w_2]$, what about y_2 for 'Alice'?



Credit: Jo Grundy

Recommendation Systems– Content based approach

Linear Regression $$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2 \quad (1)$$

Can use a vector of features for each film, eg romance, action

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Take the user 'Alice' as an example

$y_1 = 4, y_3 = 5, y_4 = 1$

$X_1 = [1, x_1, x_2] = [1, 1, 0.1], X_3 = [1, 0.2, 0.9], X_4 = [1, 0.1, 1],$

$\theta = [b, w_1, w_2]$, what about y_2 for 'Alice'?

Least Squares $\frac{1}{3}\ [(\theta^T X_1 - y_1) + (\theta^T X_3 - y_3) + (\theta^T X_4 - y_4)] = \min_\theta \frac{1}{3} \sum_i (b + w^T X_i - y_i)$

Credit: Jo Grundy

Recommendation Systems– Content based approach

Linear Regression $$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2 \qquad (1)$$

Can use a vector of features for each film, eg romance, action

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Take the user 'Alice' as an example

$y_1 = 4, y_3 = 5, y_4 = 1$

$X_1 = [1, x_1, x_2] = [1, 1, 0.1], X_3 = [1, 0.2, 0.9], X_4 = [1, 0.1, 1]$,

$\theta = [b, w_1, w_2]$, what about y_2 for 'Alice'?

Least Squares$\frac{1}{3}$ $[(\theta^T X_1 - y_1) + (\theta^T X_3 - y_3) + (\theta^T X_4 - y_4)] = \min_\theta \frac{1}{3} \sum_i (b + w^T X_i - y_i)$

- Data Mining and Machine Learning: Fundamental Concepts and Algorithms M. L. Zaki and W. Meira *Chapter 23, Page 589-593*

Recommendation Systems– Content based approach

Linear Regression $\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2$ (1)

Can use a vector of features for each film, eg romance, action

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Take the user 'Alice' as an example

$y_1 = 4, y_3 = 5, y_4 = 1$

$X_1 = [1, x_1, x_2] = [1, 1, 0.1], X_3 = [1, 0.2, 0.9], X_4 = [1, 0.1, 1],$

$\theta = [b, w_1, w_2]$, what about y_2 for 'Alice'?

Least Squares: SSE= $[(\theta^T X_1 - y_1) + (\theta^T X_3 - y_3) + (\theta^T X_4 - y_4)] = \min_{\theta} \frac{1}{3} \sum_i (b + w^T X_i - y_i)$

Gradient Decent:

Differentiate it with respect to b and set the result to 0

$$\frac{\partial}{\partial b} SSE = -2 \sum_{i=1}^{n} (y_i - b - w \cdot x_i) = 0$$

$$\Longrightarrow \sum_{i=1}^{n} b = \sum_{i=1}^{n} y_i - w \sum_{i=1}^{n} x_i$$

$$\Longrightarrow b = \frac{1}{n} \sum_{i=1}^{n} y_i - w \cdot \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\frac{\partial}{\partial w} SSE = -2 \sum_{i=1}^{n} x_i (y_i - b - w \cdot x_i) = 0$$

$$\Longrightarrow \sum_{i=1}^{n} x_i \cdot y_i - b \sum_{i=1}^{n} x_i - w \sum_{i=1}^{n} x_i^2 = 0$$

Recommendation Systems– Content based approach

Linear Regression $\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2$ (1)

Can use a vector of features for each film, eg romance, action

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Take the user 'Alice' as an example

$y_1 = 4, y_3 = 5, y_4 = 1$

$X_1 = [1, x_1, x_2] = [1, 1, 0.1], X_3 = [1, 0.2, 0.9], X_4 = [1, 0.1, 1],$

$\theta = [b, w_1, w_2]$, what about y_2 for 'Alice'?

Least Squares SSE= $[(\theta^T X_1 - y_1) + (\theta^T X_3 - y_3) + (\theta^T X_4 - y_4)] = \min_\theta \frac{1}{3} \sum_i (b + w^T X_i - y_i)$

Gradient Decent:

$$b = \mu_Y - w \cdot \mu_X$$

$$w = \frac{\sum_{i=1}^{n} (x_i - \mu_X)(y_i - \mu_Y)}{\sum_{i=1}^{n} (x_i - \mu_X)^2} = \frac{\sigma_{XY}}{\sigma_X^2} = \frac{\mathrm{cov}(X, Y)}{\mathrm{var}(X)}$$

Recommendation Systems– Content based approach

Use Linear Regression to find user parameter vector θ where m is the number of films rated by that user

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} (\theta^T X_i - y)^2 \tag{1}$$

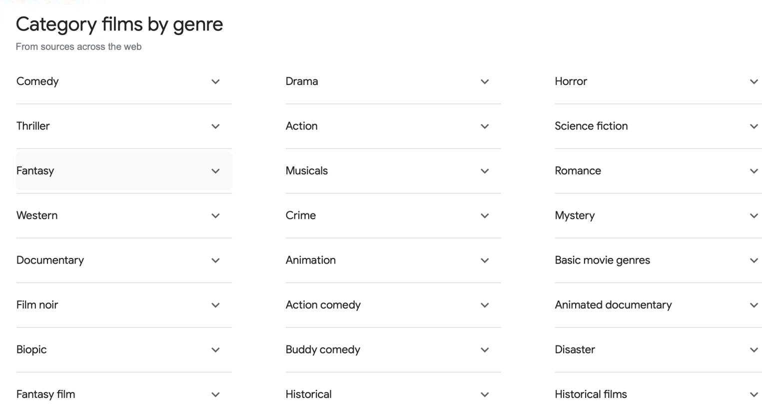
Can also use Bayesian classifiers, MLPs, etc.

Problems?

requires hand coded knowledge of film

not easy to scale up

user may not have rated many films



Can use a vector of features for each film, eg romance, action

Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Fight	1	5			0.1	1

Credit: Jo Grundy

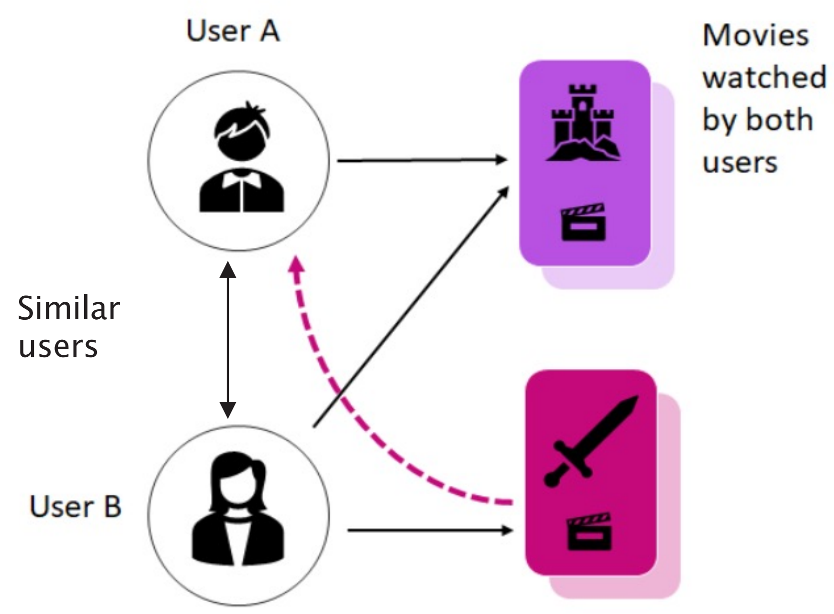
Recommendation Systems – Collaborative Filtering

Collaborative Filtering example:
Alice likes Dr Who, Star Wars and Star Trek
Bob likes Dr Who and Star Trek
A recommender system would correlate the likes, and suggest that Bob might like Star Wars too.
Personal preferences can be correlated.

Task: Discover patterns in observed behaviour across a community of users

- Purchase history
- Item ratings
- Click counts

Predict new preferences based on those patterns



Credit: Jo Grundy

Recommendation Systems – Collaborative Filtering

Collaborative filtering uses a range of approaches to accomplish this task

- Neigbourhood based approach
- Model based approach
- Hybrid (Neighbourhood and model) based approach

This lecture will cover the Neighborhood based approach

Recommendation Systems – Collaborative Filtering

Measure user preferences. Eg. Film recommendation
Users rate films between 0 and 5 stars

	Life is Beautiful	Seven Samurai	Joker	Schindler's List	The Pianist	City of God
Lee	2.5	3.5	3.0	3.5	3.0	2.5
Sofia	3.0	3.5	1.5	5.0	3.0	3.5
Miley	2.5	3.0		3.5	4.0	
Justina		3.5	3.0	4.0	4.5	2.5
Donald	3.0	4.0	2.0	3.0	3.0	2.0
Mikey	3.0	4.0		5.0	3.0	3.5
Tristan		4.5		4.0		1.0

The data is sparse, there are missing values

Credit: Jo Grundy

Collaborative Filtering – Sparsity

Sparsity can be taken advantage of to speed up computations
Most libraries that do matrix algebra are based on LAPACK, written in Fortan90
Computation is done by calls to the Basic Linear Algebra Subprograms (BLAS).

This is how the Python `numpy` library does its linear algebra.

Collaborative Filtering – Sparsity

Compresssed Row Storage (CRS) [1]

Matrix specified by three arrays: *val*, *col_ind* and *row_ptr*

val stores the non zero values

col_ind stores column indices of each element in val

row_ptr stores the index of the elements in *val* which start a row

E.g. What matrix would this give?

val = [1, 2, 9, 8, 2, -1, 4, 5, 2, 7]

col_ind = [1, 2, 4, 3, 4, 1, 2, 4, 2, 3]

row_ptr = [1, 4, 6, 9]

[1] Harwell-Boeing sparse matrix format, Duff et al, ACM Trans. Math. Soft., 15 (1989), pp. 1-14.

Credit: Jo Grundy

Collaborative Filtering – Sparsity

Compresssed Row Storage (CRS) [1]

Matrix specified by three arrays: *val*, *col_ind* and *row_ptr*

val stores the non zero values

col_ind stores column indices of each element in val

row_ptr stores the index of the elements in *val* which start a row

E.g. What matrix would this give?

val = [1, 2, 9, 8, 2, -1, 4, 5, 2, 7]

col_ind = [1, 2, 4, 3, 4, 1, 2, 4, 2, 3]

row_ptr = [1, 4, 6, 9]

$$\begin{bmatrix} 1 & 2 & & 9 \\ & & 8 & 2 \\ -1 & 4 & & 5 \\ & 2 & 7 & \end{bmatrix}$$

[1] Harwell-Boeing sparse matrix format, Duff et al, ACM Trans. Math. Soft., 15 (1989), pp. 1-14.

Credit: Jo Grundy

Collaborative Filtering – Sparsity

Analogously, there is also Compresssed Column Storage (CCS)

Matrix specified by three arrays: *val*, *row_ind* and *col_ptr*

val stores the non zero values

row_ind stores row indices of each element in val

col_ptr stores the index of the elements in *val* which start a column

E.g. What matrix would this give?

$val = [2,2,5,3,1,4]$

$row\_ind = [1,4,3,1,2,1]$

$col\_ptr = [1,3,4,6]$

Credit: Jo Grundy

Collaborative Filtering – Sparsity

Analogously, there is also Compresssed Column Storage (CCS)
Matrix specified by three arrays: *val*, *row_ind* and *col_ptr*
val stores the non zero values
row_ind stores row indices of each element in val
col_ptr stores the index of the elements in *val* which start a column
E.g. What matrix would this give?
val = [2,2,5,3,1,4]
row_ind = [1,4,3,1,2,1]
col_ptr = [1,3,4,6]

$$\begin{bmatrix} 2 & & 3 & 4 \\ & & 1 & \\ & 5 & & \\ 2 & & & \end{bmatrix}$$

The CCS is the CRS of A^T

Credit: Jo Grundy

Collaborative Filtering – Sparsity

Also Block Compressed Row Format (BSR) :

val stores the non zero blocks

col_ind stores column indices of each element in val

row_ptr stores the index of the elements in *val* which start a row

$$val = \begin{bmatrix} 2 & \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 4 & 7 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 4 & 5 \\ 3 & \end{bmatrix} \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$$

col_ind = [1,2,3,1]

row_ptr = [1,2,4]

Credit: Jo Grundy

Collaborative Filtering – Sparsity

Also Block Compressed Row Format (BSR) :

val stores the non zero blocks

col_ind stores column indices of each element in val

row_ptr stores the index of the elements in *val* which start a row

$$val = \begin{bmatrix} 2 & \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 4 & 7 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 4 & 5 \\ 3 & \end{bmatrix} \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$$

col_ind = [1,2,3,1]

row_ptr = [1,2,4]

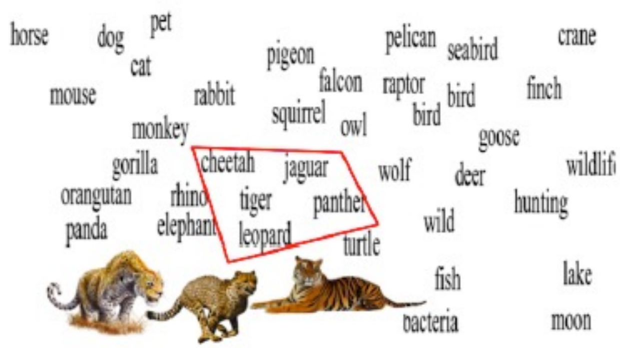
$$\begin{bmatrix} 2 & & & & & \\ 3 & 1 & & & & \\ & & 4 & 7 & 4 & 5 \\ & & 3 & 1 & 3 & \\ 1 & & & & & \\ & 1 & & & & \end{bmatrix}$$

Credit: Jo Grundy

Collaborative Filtering – Feature Extraction

Features are stored in a 'feature vector', a fixed length list of numbers.

- The length of this vector is the number of dimensions
- Each vector represents a point and a direction in the featurespace.
- Each vector must have the same dimensionality



A projection of encoded word vectors shows that similar words are close to each other in feature space.

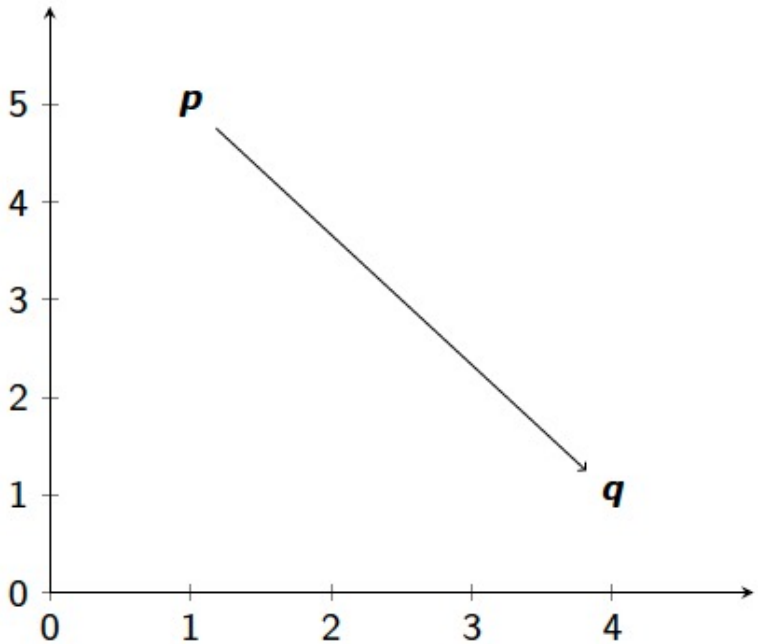
We say two things are *similar* if they have similar feature vectors, i.e. are close to each other in featurespace.

Credit: Jo Grundy

Collaborative Filtering – Distance

There are a number of ways to measure distance in feature space:

- Euclidean Distance:



$\boldsymbol{p}$ and $\boldsymbol{q}$ are N-dimensional feature vectors,
$\boldsymbol{p} = [p_1, p_2, ..., p_N]$,
$\boldsymbol{q} = [q_1, q_2, ..., q_N]$

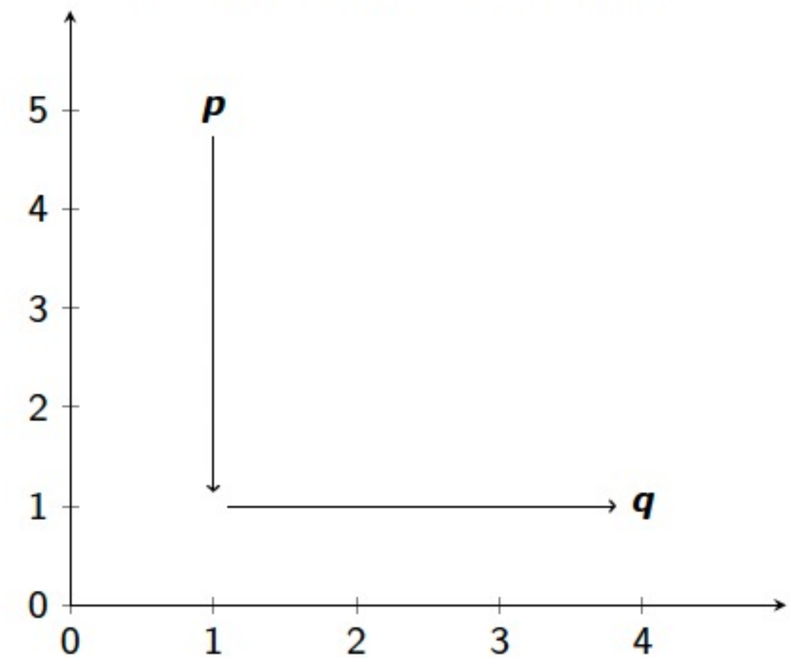
Euclidean distance:

$$||\boldsymbol{p} - \boldsymbol{q}|| = \sqrt{\sum_{i=1}^{N}(q_i - p_i)^2}$$

Credit: Jo Grundy

Collaborative Filtering – Distance

- Manhattan Distance:



$\boldsymbol{p}$ and $\boldsymbol{q}$ are N-dimensional feature vectors,
$\boldsymbol{p} = [p_1, p_2, ..., p_N]$,
$\boldsymbol{q} = [q_1, q_2, ..., q_N]$

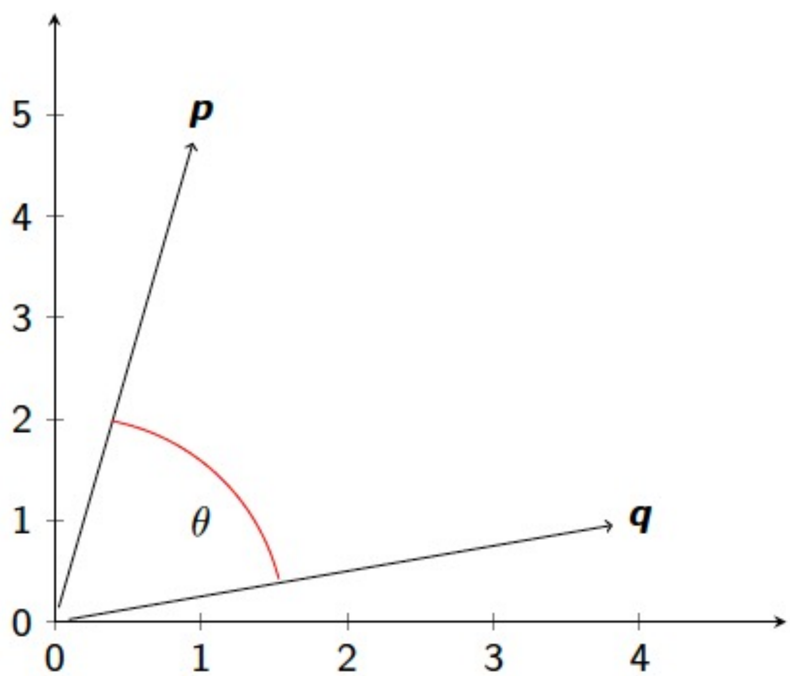
Manhattan distance:

$$\|\boldsymbol{p} - \boldsymbol{q}\|_1 = \sum_{i=1}^{N} |q_i - p_i|$$

Credit: Jo Grundy

Collaborative Filtering – Distance

- Cosine Similarity



Only measures direction, not magnitude of vector.

$\boldsymbol{p}$ and $\boldsymbol{q}$ are N-dimensional feature vectors,
$\boldsymbol{p} = [p_1, p_2, ..., p_N]$,
$\boldsymbol{q} = [q_1, q_2, ..., q_N]$

Cosine Similarity:

$$cos(\theta) = \frac{\boldsymbol{p}.\boldsymbol{q}}{|\boldsymbol{p}||\boldsymbol{q}|}$$

$$= \frac{\sum_{i=1}^{N} p_i q_i}{\sqrt{\sum_{i=1}^{N} p_i^2}\sqrt{\sum_{i=1}^{N} q_i^2}}$$

Credit: Jo Grundy

Collaborative Filtering – User Similarity

Need to define a similarity score, based on the idea that similar users have similar tastes, i.e. like the same movies.)

Needs to take in to account sparsity, not all users have seen all movies.

Typically between 0 and 1, where 1 is the same, and 0 is totally different

Can visualise the users in feature space, using two dimensions at a time

Visualisation of users in film space ipynb

Film	Alice	Bob	Carol	Dave
Love Really	4	1		4
Deadly Weapon		1	4	5
Fast and Cross	5		5	4
Star Battles	1	5		

Credit: Jo Grundy

Collaborative Filtering – User Similarity

There are many ways to compute similarity based on Euclidean distance

We *could* chose:

$$sim_{L2}(x, y) = \frac{1}{1 + \sqrt{\sum_{i \in I_{xy}} (r_{x,i} - r_{y,i})^2}}$$

where $r_{x,i}$ is the rating from user x for item i

I_{xy} is set of items rated by both x and y

i.e. when the distance is 0, the similarity is 1, but when the distance is large, similarity $\to 0$

Credit: Jo Grundy

Collaborative Filtering – User Similarity

Can also use cosine similarity

$$sim_{cos}(x, y) = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_{xy}} r_{xi}^2} \sqrt{\sum_{i \in I_{xy}} r_{yi}^2}}$$

Only performed over the items which are rated by both users

Credit: Jo Grundy

Collaborative Filtering – User Similarity

Alternatively, calculate correlation of users, based on ratings they share

Using Pearson's Correlation: standard measure of dependence between two related variables.

$$sim_{Pearson}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Where $\bar{r_x}$ is average rating user x gave for all items in I_{xy}

Correlation between users in ipynb

Credit: Jo Grundy

Collaborative Filtering – User Similarity

Users are inconsistent. Some users always give out 5s to films they like, whereas some are more picky.
For example, look at Lisa and Jill, both rank the films in the same order, but give different ratings.
Pearson correlation corrects for this, but Euclidean similarity doesn't.
Data normalisation and mean centering can overcome this.

Data standardisation

Film	Alice	Bob	Carol	Dave	Lisa	Jill
Love Really	4	1		4	1	2
Deadly Weapon		1	4	5	2	3
Fast and Cross	5		5	4	3	4
Star Battles	1	5			4	5

Credit: Jo Grundy

Collaborative Filtering – User Filtering

We now have a set of measures for computing the similarity between users

Produce a ranked list of best matches to a target user. Typically want the top-N users

May only want to consider a subset of users, i.e. those who rated a particular item.

Ranking users by similarity ipynb

Collaborative Filtering – Recommending

Now we have a list of similar users, how can we recommend items?
Predict rating $r_{u,i}$ of item i by user u as an aggregation of the ratings of item i by users similar to u

$$r_{u,i} = aggr_{\hat{u} \in U}(r_{\hat{u},i})$$

Where U is the set of *top* users most similar to u that rated item i
Multiply the score by the similarity of the user
Normalise by sum of similarities (otherwise items rated more often will dominate)

$$r_{u,i} = \frac{\sum_{\hat{u} \in U} sim(u, \hat{u}) r_{\hat{u},i}}{\sum_{\hat{u} \in U} |sim(u, \hat{u})|}$$

This is *User Based Filtering*
Demo:User based recommendation

Credit: Jo Grundy

Collaborative Filtering – User Based Filtering

Can also aggregate by computing average over similar users

$$r_{u,i} = \frac{1}{N} \sum_{\hat{U} \in U} r_{\hat{U},i}$$

Or by subtracting the average user rating score for all the items they scored, this is to compensate for people that judge generously or meanly.

$$r_{u,i} = \bar{r_u} + \frac{\sum_{\hat{u} \in U} sim(u, \hat{u})(r_{\hat{u},i} - \bar{r_{\hat{u}}})}{\sum_{\hat{u} \in U} |sim(u, \hat{u})|}$$

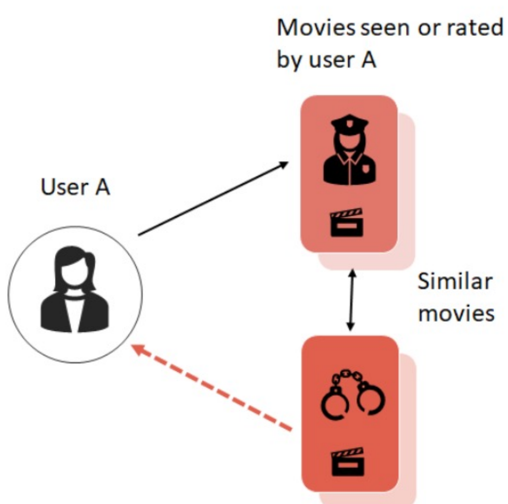
Credit: Jo Grundy

Recommendation Systems – **Summary**

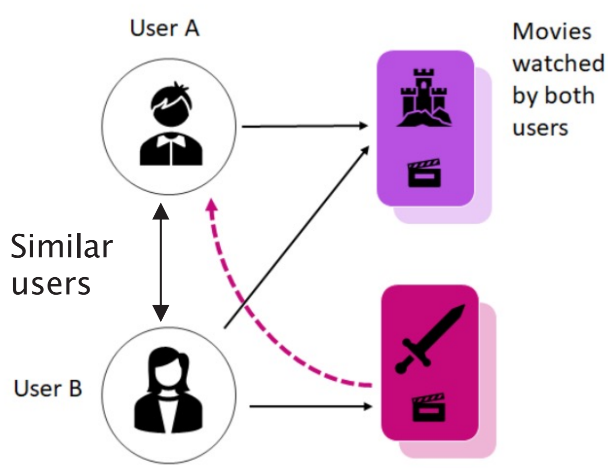
Film	Alice	Bob	Carol	Dave	x_1 romance	x_2 action
Love Really	4	1		4	1	0.1
Deadly Weapon		1	4	5	0.1	1
Fast and Cross	5		5	4	0.2	0.9
Star Battles	1	5			0.1	1
Dark Knight	5			?	0.1	1

https://en.wikipedia.org/wiki/The_Dark_Knight

Content-based Filtering
Movie recommendations to user A



Collaborative Filtering
Movies watched by user A are recommended to user B



Collaborative Filtering:

- Uses peoples behaviour to gather information
- Doesn't need content based features
  User based Neighbourhood approach:
  - Computes similarities between users
  - Predicts unseen item weights using ratings of similar users

Source: https://www.relataly.com/