

COMP3308/3608, Lecture 13a

ARTIFICIAL INTELLIGENCE

Applications of Artificial Intelligence

- Recommender Systems

Outline

- **Introduction**
- **Collaborative filtering approaches**
- **Content-based approaches**

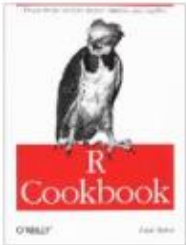
Recommender Systems

- Recommender Systems (RS) recommend *items* to people
 - Which *digital camera (mobile phone, washing machine, etc)* should I buy?
 - What is the best *holiday* for me?
 - Which *movie* should I rent?
 - Which *web sites* will I find interesting?
 - Which *book* should I buy for my next *holiday*?
 - Which *degree* and *university* are the best for my future?

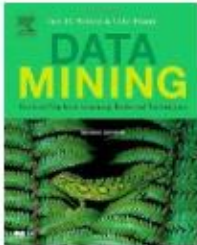
Your Amazon.com

[Featured Recommendations](#)[Books](#)[Video Games](#)[See All Recommendations](#)


Books



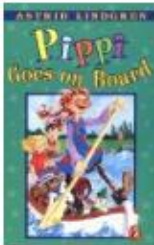
R Cookbook
Paul Teator
★★★★☆ (19)
Paperback
\$39.99 **\$28.04**
[Why recommended?](#)



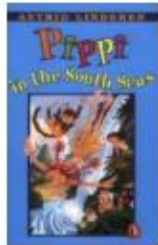
Data Mining
Ian H. Witten
★★★★☆ (36)
Paperback
[Why recommended?](#)




Data Mining with Rattle...
Graham J. Williams
★★★★☆ (6)
Paperback
\$64.95 **\$51.10**
[Why recommended?](#)



Pippi Goes on Board
Astrid Lindgren
★★★★☆ (15)
Paperback
\$5.99
[Why recommended?](#)



Pippi in the South Seas
Astrid Lindgren
★★★★☆ (16)
Paperback
\$5.99
[Why recommended?](#)

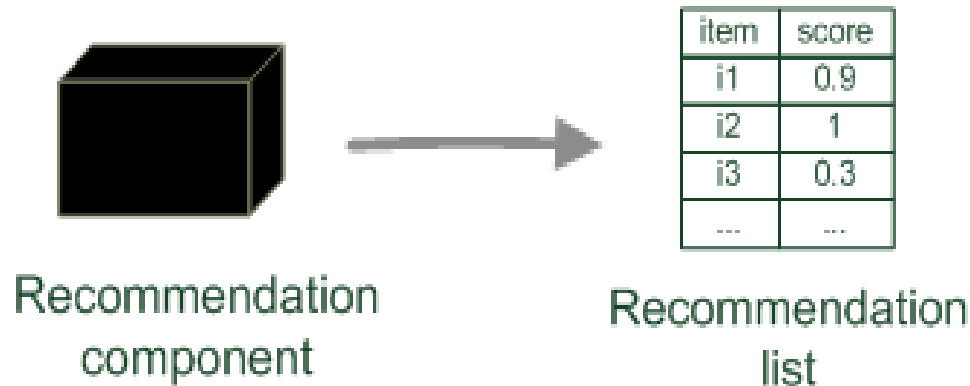


Machine Learning
Stephen Marsland
★★★★☆ (22)
Hardcover
\$72.95 **\$61.21**
[Why recommended?](#)

[See all recommendations in Books](#)

Motivation

- WWW has become the primary source of information
- It has a huge content -> need *to reduce the information overload*
- RS help users to find products, services and information, by predicting their relevance



Three Main Approaches

1) Collaborative Filtering

- *“Show me books that are popular among my peers”*
- Uses the preferences of the community and not the item’s features

2) Content-Based

- *“Show me more of the type of books I like”*
- Uses only the item’s features and not the preferences of the community

3) Hybrid – combinations of collaborative filtering and content-based

Collaborative Filtering

Collaborative Filtering (CF)

- The most prominent and widely used approach
- Well understood, many variations
- Main idea: use the *"wisdom of the crowd"*
- Input:
 - a matrix of user–item ratings, i.e. assumes that the target user U had rated some of the items
- Output:
 - a predicted rating for a new (unseen) item indicating how much U will like it
 - a list of predicted items (top- N list) sorted in decreasing order based on their scores



Example - Movielens

- User U rates some movies:

actual
ratings

Prediction or Rating ↕	Your Rating	Movie Information	V I
★★★★★	5.0 stars ▾	Spanish Apartment, The (L'auberge espagnole) (2002) DVD info imdb flag Movie Tuner Comedy, Drama, Romance - French, Spanish, English, Catalan, Danish, German, Italian [add tag] Popular tags: Spain college ensemble cast	
★★★★★	5.0 stars ▾	Sea Inside, The (Mar adentro) (2004) DVD VHS info imdb flag Movie Tuner Drama - Spanish [add tag] Popular tags: true story emotional Oscar (Best Foreign Language Film)	
★★★★★	5.0 stars ▾	Lives of Others, The (Das leben der Anderen) (2006) DVD info imdb flag Movie Tuner Drama, Romance, Thriller - German [add tag] Popular tags: spying disturbing Oscar (Best Foreign Language Film)	
★★★★★	5.0 stars ▾	Good bye, Lenin! (2003) DVD VHS info imdb flag Movie Tuner Comedy, Drama - German [add tag] Popular tags: comedy historical romance	
★★★★☆	4.5 stars ▾	Volver (2006) DVD info imdb flag Movie Tuner Comedy, Drama - Spanish [add tag] Popular tags: Spain ghosts visually appealing	
★★★★☆	4.5 stars ▾	Tunnel, The (Tunnel, Der) (2001) DVD info imdb flag Movie Tuner Action, Drama, Thriller - German	

- The system uses U's ratings, together with the ratings of other users, to give recommendations to U:

predicted
ratings

★★★★★	Not seen ▾	Secret in Their Eyes, The (El secreto de sus ojos) (2009) DVD info imdb flag Movie Tuner Crime, Drama, Mystery, Romance, Thriller - Spanish [add tag] Popular tags: revenge based on a book romance
★★★★★	Not seen ▾	Talk to Her (Hable con Ella) (2002) DVD VHS info imdb flag Movie Tuner Drama, Romance - Spanish

Main CF Methods

- **Nearest neighbour**
 - **user-to-user** - uses the similarity between **users**
 - **item-to-item** - uses the similarity between **items**
- **Advanced**
 - **Matrix factorization** – maps users and items to a joint factor space
 - **Probabilistic**
 - **Clustering-based**
 - **Using association rules**

User-to-User CF

- **Makes predictions using the similarity between users**
- **Finds like-minded users who can complement each other's ratings**

Given: a target user Alice and an item i not yet seen by Alice

- **Find (neighbourhood) N - a set of users (nearest neighbors), who liked the same items as Alice in the past and who have rated item i**
- **Combine the ratings of the users in N (e.g. take the average) to predict if Alice will like item i**
- **Do this for all items Alice has not seen and recommend the best-rated**

Example

- **Given:** a matrix of ratings of Alice and other users
- **Goal:** Find if Alice will like or dislike *Item5*, which Alice has not rated (seen) yet

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

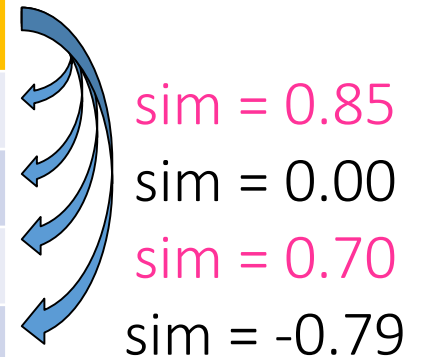
Questions:

- 1) How do we measure similarity between users?
- 2) How many neighbors should we consider?
- 3) How do we generate a prediction from the neighbors' ratings?

Prediction for Alice

- 1) Correlation as a similarity (distance measure)
- 2) 2-Nearest-Neighbour – closest neighbours are User1 and User3
- 3) Average of neighbors' ratings:
 $\Rightarrow P(\text{Alice, item5}) = (3+4)/2=3.5$

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1



Similarity Measure - Correlation

- Pearson correlation coefficient
- Most popular in item-based collaborative filtering

$$sim(a, b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}}$$

a, b - users

P - set of items, rated by both a and b

$r_{a,p}$ - rating of user a for item p

\bar{r}_a - average rating of user a

Combining the Ratings – Various Ways

- **How many neighbors?**
 - use a similarity threshold or fixed number of neighbors
- **How to combine the ratings of the neighbors?**
 - Simple average
 - Weighed average (by the similarity between the neighbour and target user)
 - Weighed normalised adjusted average – commonly used:

Prediction for user a for item p

weighing by the similarity between a & b

Is b 's rating of p higher or lower than b 's average rating?

$$pred(a, p) = \underbrace{\bar{r}_a}_{\text{a's average rating}} + \underbrace{\frac{\sum_{b \in N} sim(a, b) * (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} sim(a, b)}}_{\text{normalization}}$$

- **Adjusted = considers rating differences between users**

Improving the Prediction - Variations

- **Not all neighbor ratings are equally valuable**
 - E.g. agreement on commonly liked items is not so informative as agreement on controversial items
 - Solution: Give more weight to items that have a higher variance
- **Number of co-rated items**
 - Reduce the weight when the number of co-rated items is low
- **Case amplification**
 - Higher weight to very similar neighbors, i.e. similarity close to 1

Item-to-Item CF

- Makes predictions using the similarity between **items**
- Main idea:
 - Find items that are similar to the new item (item5) based on the rankings
 - Use Alice's ratings for these items to predict her rating for the new item

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

e.g. $P(\text{Alice}, \text{item5})$
 $= (5+4)/2 = 4.5$

Recommendation for Alice: Alice will like item 5, as she bought and liked items 1 and 4, and item 5 is similar to items 1 and 4 based other people's ratings

Similarity Measure - Cosine

- **Most popular in item-based collaborative filtering**
 - Ratings are seen as vector in n-dimensional space
 - Cosine similarity is the angle between the vectors

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

- **Adjusted cosine similarity**
 - Considers also the average user ratings to transform the original rating
 - U : set of users who have rated both items a and b

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

Combining the Ratings

- Common prediction function – weighted prediction:

$$pred(u, p) = \frac{\sum_{i \in ratedItem(u)} sim(i, p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i, p)}$$

Can we Precompute the Similarities?

- **Rating matrix: a large number of items and a small number of ratings per user**
- **User-to-user collaborative filtering**
 - similarity between users is unstable (changes considerably when only a few ratings are changing (e.g. added) because of the small number of commonly ranked items)
 - => pre-computing the similarities leads to poor performance
- **Item-to-item collaborative filtering**
 - similarity between items is more stable
 - we can pre-compute the item-to-item similarity and the nearest neighbours
 - prediction involves lookup for these values and computing the weighed sum (Amazon does this)

Explicit vs Implicit Ratings

Explicit Ratings

- **Explicitly provided by the user**
 - **Most commonly used scales: 1 to 5 or 1 to 7 on Likert scale**
- **Explicit ratings are reliable but it is often difficult to obtain a sufficient number of ratings**
 - **Small number of available ratings => sparse rating matrices => poor recommendation quality**
 - **How to encourage users to rate more items?**

Implicit Ratings

- **Collected automatically by the web shop or application in which the recommender system is embedded**
 - **When a customer buys an item, many recommender systems interpret this behavior as a positive rating**
 - **Clicks, page views, time spent on some page, demo downloads, etc.**
- **Main problem – might not be reliable**
 - **One cannot be sure whether the user behavior is correctly interpreted**
 - **E.g. a user might not like all the books he bought or might have bought a book for someone else**
- **Typically used in conjunction with the explicit ratings**

Advanced CF Methods

SVD-based CF

- **Idea: Generate predictions faster by reducing the dimensionality of the rating matrix**
- **Based on Singular Value Decomposition (SVD) for dimensionality reduction of the rating matrix**
- **SVD is widely used for dimensionality reduction – e.g. feature selection in ML, image compression, efficient indexing and retrieval in IR (Latent Semantic Analysis)**

SVD

- Any $n \times m$ matrix X ($n \geq m$) can be written as the product of 3 matrices:

$$X = U \times \Sigma \times V^T$$

U - $n \times m$ orthogonal matrix

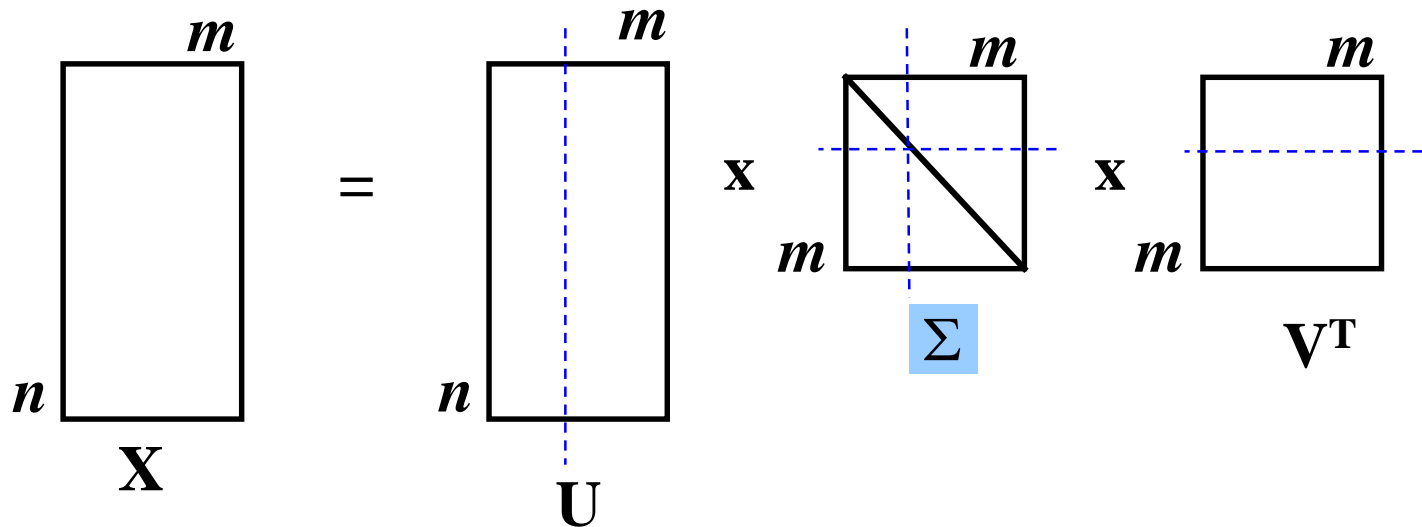
V^t - the transpose of matrix V , where V is a $m \times m$ orthogonal matrix

Σ - $m \times m$ diagonal matrix containing the singular values along the main diagonal, sorted in increasing order

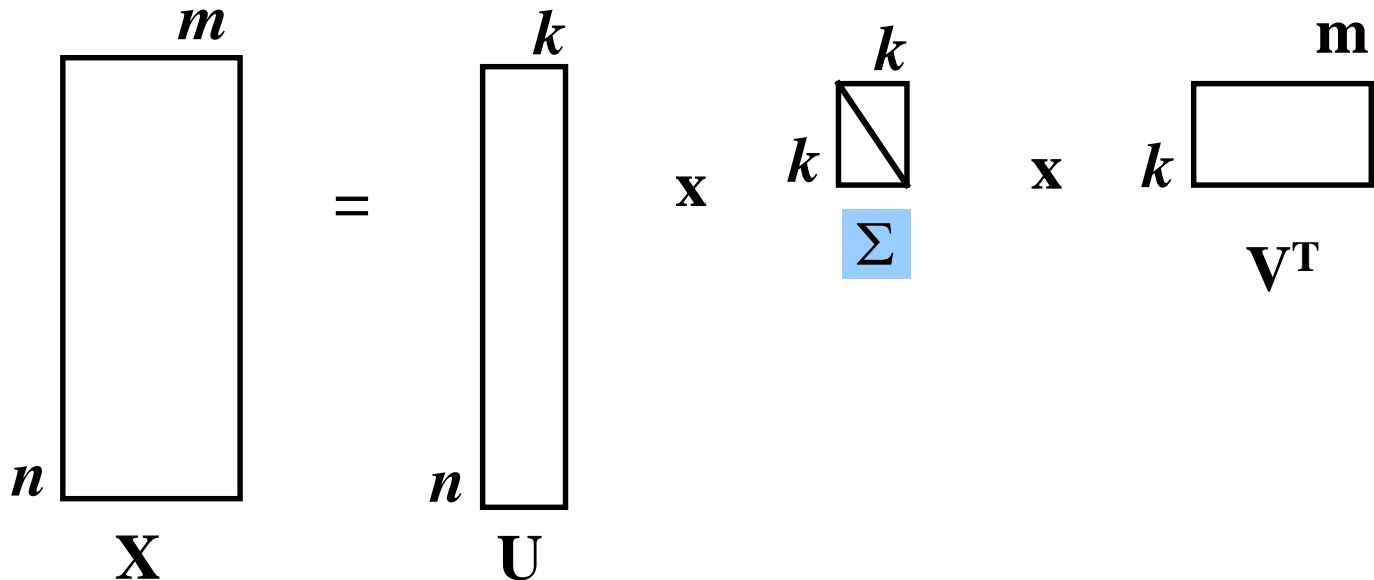
- We can approximate X by retaining only the most important features, those with the largest singular values
- In our example (slide 28), we calculate U , V , and Σ but retain only the two most important features by taking only the first two columns of U and the first two rows of V^t

SVD – Graphical Representation

Without
data
reduction



With data
reduction



Compression Ratio

- Space needed before SVD: $n \times m$
- Space needed after SVD: $n \times k + k + k \times m$
 - The first k columns of U (n dimensional vectors), the k singular vectors and the first k columns of V (m dimensional vectors) \Rightarrow total = $k(n+1+m)$
- Compression ratio

$$r = \frac{n \times k + k + k \times m}{n \times m}$$

- For $n \gg m > k$, this ratio is approximately k/m

SVD-Based Recommendation

- SVD:** $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2	new features (factors)
Alice	0.47	-0.30	
Bob	-0.44	0.23	
Mary	0.70	-0.06	
Sue	0.31	0.93	

	Terminator	Die Hard	Tomorrow Never Dies	Eat Pray Love	Pretty Woman
V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

- Calculating the prediction for user u and item $i=EPL$:**

$$\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$$

$$= 3 + 0.84 = 3.84$$

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- Predictions are generated faster on the low dimensional data**

Matrix Factorization

- **An extension of SVD-based recommendation**
- **The rating matrix is sparse and the standard SVD cannot be directly applied**
- **Earlier systems filled in the missing data and then applied SVD – data distortion, inaccurate predictions**
- **Current approach**
 - **Use only the available ratings**
 - **Formulate the problem of finding U and V as an optimisation problem (minimizing the regularized squared error on the set of known ratings)**
 - **Use stochastic gradient descent to solve it**
- **Matrix factorization is the state-of-the-art approach for CF**

Y. Koren, R. Bell and C. Volinsky (2009). Matrix Factorization Techniques for Recommender Systems. IEEE Computer, Volume 42, Issue 8, p.30-37.

Probabilistic CF

$$P(H | E) = \frac{P(E | H)P(H)}{P(E)}$$

- We can use Naïve Bayes
- What is the probability that Alice will give rating 1 to item 5, given her previous ratings?
- Evidence E (Alice's previous ratings) - 5 pieces:
 $E = \text{item1}=1, \text{item2}=3, \text{item3}=3, \text{item4}=2$
- 5 hypotheses H (possible ratings for item5):
 $H1: \text{item5}=1, H2: \text{item5}=2, \dots, H5: \text{item5}=5$
- Assumption: the previous ratings (5 pieces of E) are independent of each other given a hypothesis:

	Item1	Item2	Item3	Item4	Item5
Alice	1	3	3	2	?
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1

$$P(H | E) = \frac{\prod_{i=1}^d P(E_i | H)P(H)}{P(E)}$$

Probabilistic CF

- Estimate the right hand side from the data for the other users:

$$\begin{aligned} &P(\text{item5} = 1|E) \\ &= P(\text{Item1} = 1|\text{Item5} = 1) \times P(\text{Item2} = 3|\text{Item5} = 1) \\ &\times P(\text{Item3} = 3|\text{Item5} = 1) \times P(\text{Item4} = 2|\text{Item5} = 1) = \frac{2}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \approx 0.125 \end{aligned}$$

$$\begin{aligned} &P(\text{item5} = 2|E) \\ &= P(\text{Item1} = 1|\text{Item5} = 2) \times P(\text{Item2} = 3|\text{Item5} = 2) \\ &\times P(\text{Item3} = 3|\text{Item5} = 2) \times P(\text{Item4} = 2|\text{Item5} = 2) = \frac{0}{0} \times \dots \times \dots \times \dots = 0 \end{aligned}$$

$$P(\text{item5} = 3|E)=\dots$$

$$P(\text{item5} = 4|E)=\dots$$

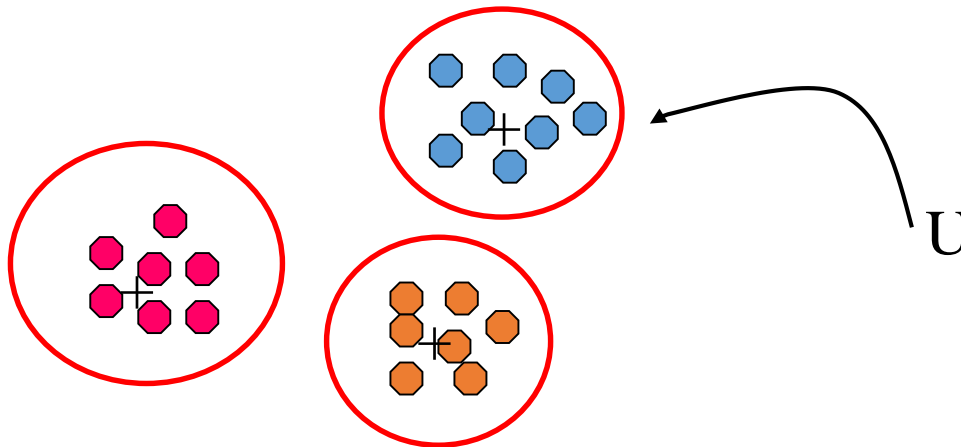
$$P(\text{item5} = 5|E)=\dots$$

	Item1	Item2	Item3	Item4	Item5
Alice	1	3	3	2	?
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1

- Select the hypothesis (rating for item 5) with highest probability

Clustering-Based CF

- **Cluster the users based on their ratings**
 - **Various distance measures and clustering algorithms**
- **Each cluster is represented with its centroid, a vector whose i -th component is the average rating of all users in the cluster for item i**
- **To make a recommendation for a new user U , find the closest centroid to U 's ratings and use the predictions of the users in this cluster**



Using Association Rules

- **Association rules are used to analyse purchasing behaviour of customers**
 - Customers who buy **X** and **Y**, also buy **Z**
 - **X,Y -> Z**
- **Rule quality measures:**

$$\text{support}(X \rightarrow Y) = \frac{\#(X \cup Y)}{n}$$

$$\text{confidence}(X \rightarrow Y) = \frac{\#(X \cup Y)}{\# X}$$

Association Rules-Based CF

- Transform 5-point ratings into binary
 - e.g. rating = 1, if it is above the user's average score and 0 otherwise
- Mine rules such as Item1 \rightarrow Item5
 - support=2/4
 - confidence=2/2 (without Alice)
- To make recommendations for Alice:
 - Left-hand side - find "relevant" rules based on Alice's transactions (the above rule will be relevant as Alice bought Item1)
 - Right-hand side - find items not already bought by Alice and sort them based on the rules' confidence values
- Different variations possible
 - dislike statements, user associations, etc.

	Item1	Item2	Item3	Item4	Item5
Alice	1	0	0	0	?
User1	1	0	1	0	1
User2	1	0	1	0	1
User3	0	0	0	1	1
User4	0	1	1	0	0

CF – Strengths and Weaknesses

- **Strengths**
 - **Minimum knowledge engineering - requires only a rating matrix and no knowledge about the features of the products**
- **Weaknesses**
 - **Requires user community**
 - **Requires sufficient number of co-rated items**
 - **Sparse rating matrix**
 - **Cold start problem – users need to rate items before a recommendation can be made**
 - **Does not provide explanation for the recommendation**

Content-Based Recommenders

Content-Based Recommender

- Uses information about the items and not about the user community
 - e.g. recommend fantasy novels to people who liked fantasy novels in the past
- Has its roots in Information Retrieval
- What we need:
 - Information about the *content* of the items (e.g. for movies: genre, leading actors, director, awards, etc.)
 - Information about what the user likes (*user preferences*, also called *user profile*) – explicit (e.g. movie rankings by the user) or implicit
- Task: recommend items that match the user preferences

Classification Task

- Content-based recommendation can be formulated as a classification/regression task
- Using the rated data as a training set, build a classifier, for each user, that predicts the rating of new items

Item's features

class: like/dislike

Title	Genre	Country	Leading actors	awards	Keywords or Movie review	rating
The lives of others	Drama, thriller	German	Sebastian Koch, Martina Gedeck, Ulrich Mühe	Oscar and 66 other awards	In 1984 East Berlin, an agent of the secret police, conducting surveillance on a writer and his lover, ...	5
The Spanish apartment	Comedy, drama, romance	France, Spain	Romain Duris, Jidith Godreche,...	8 awards	A French student moves into an apartment in Barcelona with six other European students. Together, they speak the international language of love and friendship. ...	5
The sea inside	Drama, biography	Spain	Javier Bardem, Belén Rueda, Lola Dueñas	Oscar and 61 other awards	The real-life story of Ramon Sampedro, who ...	5
Austin Powers	comedy	USA	Mike Myers, Elizabeth Hurley	3 awards	A 1960s hipster secret agent is brought out of cryofreeze ...	1
...						

Representing Text Documents

- Content-based approaches are mainly used for recommending text-based products, e.g. web pages, news, etc.
- The task becomes *text classification*
- Standard representation of a document: a vector of *tf-idf* values for each selected word
 - Each document is represented as a vector of words, each word is represented with its *tf-idf* value (weight)
 - *tf* (term frequency) part: measures how often a term (word) appears a document, assumes that important terms appear more often
 - *idf* (inverse document frequency) part: aims to reduce the weight of terms that appear in all documents
- Instead of single words, bigrams and n-grams can be used
 - An example of bigram: “machine learning”

Example of tf-idf Representation

- tf** part - term frequency:

- Each document is a **count vector** in $\mathbb{N}^{|v|}$

“Antony” appears 73 times in the document Julius Ceasar

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Each document is represented as a vector v with dimension $|v| = 7$

Example from <http://informationretrieval.org>

Example of tf-idf Representation (2)

- **tf-idf** weights
 - The **idf** part penalizes terms that appear in many documents

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth			
Antony	157	73	0	0	0	0			
Brutus	4		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	
Caesar	232		Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	
Calpurnia	0	Antony	5.25	3.18	0	0	0	0.35	
Cleopatra	57	Brutus	1.21	6.1	0	1	0	0	
mercy	1.51	Caesar	8.59	2.54	0	1.51	0.25	0	
worser	1.37	Calpurnia	0	1.54	0	0	0	0	
		Cleopatra	2.85	0	0	0	0	0	
		mercy	1.51	0	1.9	0.12	5.25	0.88	
		worser	1.37	0	0.11	4.15	0.25	1.95	

Example from <http://informationretrieval.org>

Improvements

- **Vectors are usually long and sparse**
- **Remove the frequent words (stop words)**
 - **They will appear in all documents, e.g. "a", "the", "on", ...**
- **Use stemming**
 - **replaces variants of words by their common stem, e.g. went->go, playing -> play**
- **Select top n words to represent the document**
 - **only use top n most representative words, e.g. top 100**

Classifiers

- **Most popular**
 - **Naive Bayes**
 - **Support vector machines**
- **Other**
 - **Decision trees**
 - **Nearest neighbor**
 - **Linear regression**
 - **Rule-based**

Content-Based Recommenders – Strengths and Weaknesses

- **Strengths**
 - **Does not require user community**
 - **No cold-start problem if the user profile is available (e.g. the user has specified his preferences explicitly)**
 - **Easier to explain the recommendation to the user than when using CF (depends on the classifier)**
- **Weaknesses**
 - **Feature extraction and representation is difficult for some domains, e.g. music; CF is more appropriate**
 - **Over-specialization – the user is recommended items that are similar to those he already rated (no serendipity)**
- **Pure content-based methods are rarely found in commercial applications**

Challenges in RS

- **Scalability of algorithms with large and real-world data**
- **Short-term and long-term preferences**
- **Diversity of the generated recommendations**
- **Recommending a sequence of items (e.g. a playlist)**
- **Recommendations for mobile users**
- **Recommendations for a group of users**
- **Context-aware recommendations – different recommendations for:**
 - **Travel destinations in winter and summer**
 - **Restaurant recommendations on Saturday and Monday**
 - **Song recommendation for the car if alone or with children**
- **Reciprocal recommenders (2-way recommenders)**

Useful Links

- **Books**

- **Recommender Systems: An Introduction** (2010), D. Jannach, M. Zanker, A. Felfernig and G. Friedrich, Cambridge University Press.



- **Recommender Systems Handbook** (2011), F. Ricci, L. Rokach and B. Shapira (Eds), Springer, 2nd Edition
<http://www.springer.com/gp/book/9781489976369>



- **Top conferences**

- ACM Conference on Recommender Systems <http://recsys.acm.org/>
2020: <https://recsys.acm.org/recsys20/>
- UMAP Conference – User Modeling, Personalization and Adaptation, <http://www.um.org/>

Useful Links (2)

- **Systems**
 - **Movies: Movielens, <http://movielens.umn.edu/login>**
 - **Music – Pandora internet radio, <http://www.pandora.com/>**
 - **Movies (DVD rentals): Netflix, <http://netflix.com>**
 - **Netflix prize: <http://www.netflixprize.com/>**
- **Video on TAD - Filter bubble by Eli Pariser**
 - **http://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles.html**

Acknowledgements

Slides based on:

- **Francesco Ricci's course on Information Search and Retrieval**
<http://www.inf.unibz.it/~ricci/ISR/index.html>
- **IJCAI 2011 and IJCAI2013 tutorial on Recommender Systems by Dietmar Jannach, Markus Zanker and Gerhard Friedrich**
<http://ijcai-13.ijcai.org/program/tutorial/TD3>