If you like…

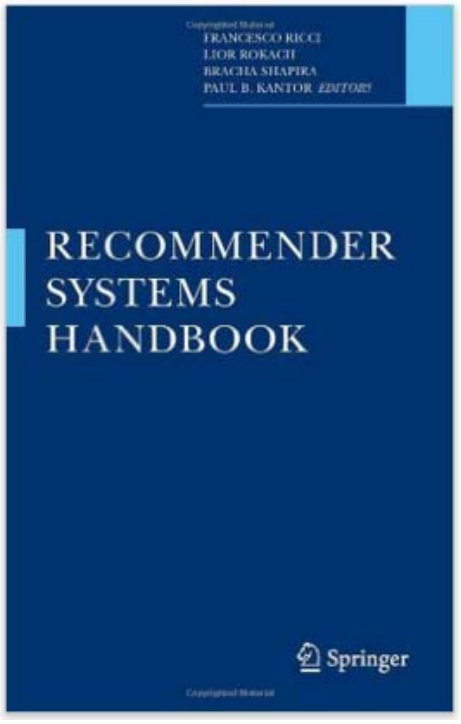
Recommender Systems

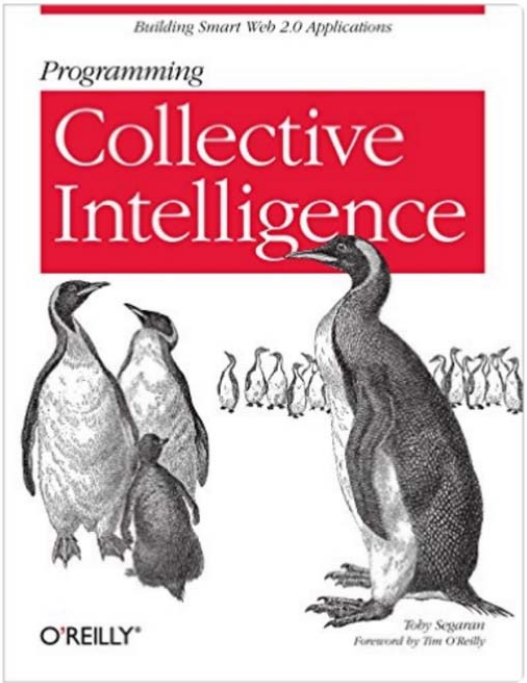
Dr. Barry Shepherd
Institute of Systems Science
National University of Singapore
Email: barryshepherd@nus.edu.sg

© 2018 NUS. The contents contained in this document may not be reproduced in any form or by any means, without the written permission of ISS, NUS, other than for the purpose for which it has been supplied.

© 2018 NUS. All rights reserved.

Reference Books





© 2018 NUS. All rights reserved.

Day Agenda

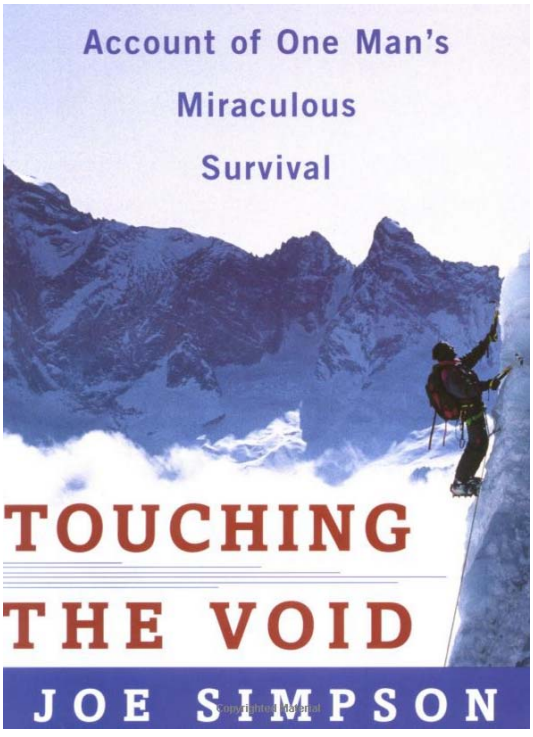
- Why Recommender Systems?
- Types of Recommender System
- Issues & Challenges
- Workshop (Build and test various recommender systems using R)

© 2018 NUS. All rights reserved.

Touching The Void

In 1988, a British mountain climber named Joe Simpson wrote a book called **Touching the Void**, a harrowing account of near death in the Peruvian Andes. It got good reviews but, only a modest success, it was soon forgotten. Then, a decade later, a strange thing happened. Jon Krakauer wrote **Into Thin Air**, another book about a mountain-climbing tragedy, which became a publishing sensation. Suddenly Touching the Void started to sell again.

WHAT HAPPENED?



© 2018 NUS. All rights reserved.

Touching The Void

What happened? In short, Amazon.com recommendations.

The online bookseller's software noted patterns in buying behavior and suggested that readers who liked "Into Thin Air" ***would also like*** "Touching the Void".

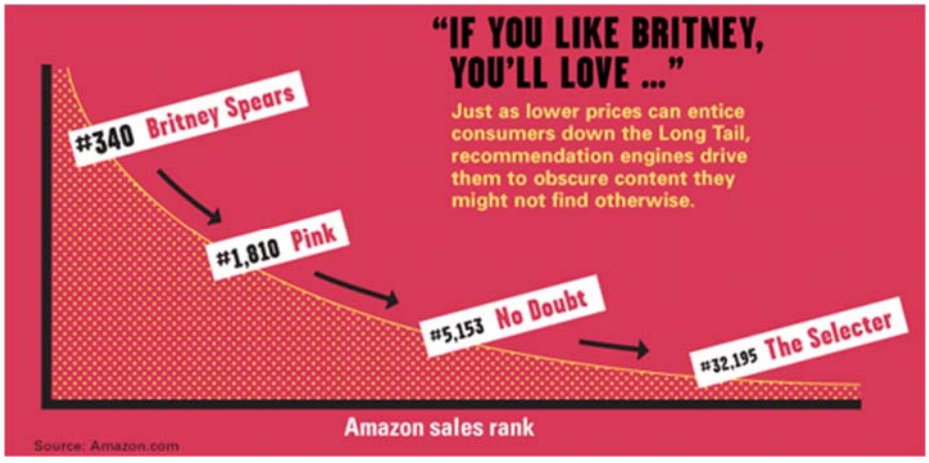
People took the suggestion, agreed wholeheartedly, wrote rhapsodic reviews. More sales, more algorithm-fueled recommendations, and the positive feedback loop kicked in.

When Krakauer's book hit the shelves, Simpson's was nearly out of print. Now Touching the Void outsells Into Thin Air more than two to one.

© 2018 NUS. All rights reserved.

The Long Tail in E-Retail

- E-retailers can hold huge inventories without needing big (expensive) shops to display them. The result is a large number of products that only sell a small amount each – the long tail!
- How to advertise and market the log tail?
- Answer (one) = Recommendation Engines



Read the full article at: http://www.wired.com/2004/10/tail

© 2018 NUS. All rights reserved.

The Netflix Challenge

- Launched in 2006 with a $1 million prize
- Netflix was an online DVD rental company - now media streaming
- Competition
  - Data = 100 million movie ratings from 500,000 anonymous users on more than 17,000 movies. Ratings were on a scale 1 to 5
  - Data format = (user, movie, rating, time)
  - Goal = predict user ratings on a test set of 3 million ratings
  - Winner = first team that can beat their existing recommendation system performance by 10% (as measure by RMSE, root mean square error)

NETFLIX

HOW WAS THE COMPETITION WON?

© 2018 NUS. All rights reserved.

Vocal Exercise

- Shout out some well-know examples of recommendation systems (other than Amazon's!)
- What are your experiences with recommender systems: are their recommendations good or bad?

© 2018 NUS. All rights reserved.

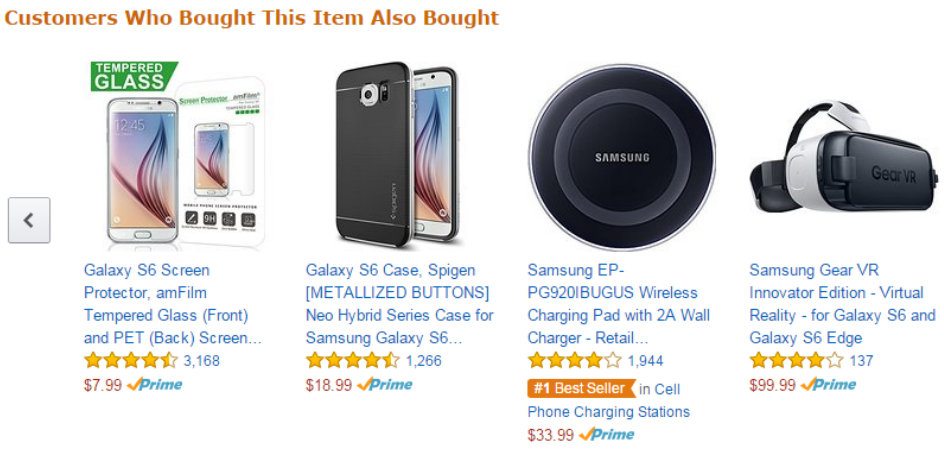
Data Sources For Recommender Systems

- **Items** ~ the objects to be recommended and their properties
- **Users** ~ their demographics, likes, preferences, goals (direct or inferred)
  - E.g. Inferences can be made from examining what the user searched for, viewed, downloaded, listened to, added to wish list or shopping cart, purchased, …
- **Ratings** ~ interactions between the user and the RS
  - ***Explicit Ratings*** – the user provides an opinion on an item using a rating scale, e.g. 1->5 (numerical) or "agree", "neutral", "disagree" etc. (ordinal)
  - ***Implicit ratings*** – derived by analysing the items selected by the user
  - ***Text comments*** - made by the user about the products
  - ***Tags*** – user assigns a tag to the items, e.g. movie is "too long", "bad acting", etc.
- **Context** ~ used to filter the recommendations
  - E.g. Is the user a beginner or advanced camera user?
  - E.g. Is the user only interested in restaurants near to their current location?

© 2018 NUS. All rights reserved.

Recommender System Approaches

- **Non-Personalised** ~ recommend the top 10 best selling products!
- **Personalised** ~ e.g. people who bought X also bought Y*

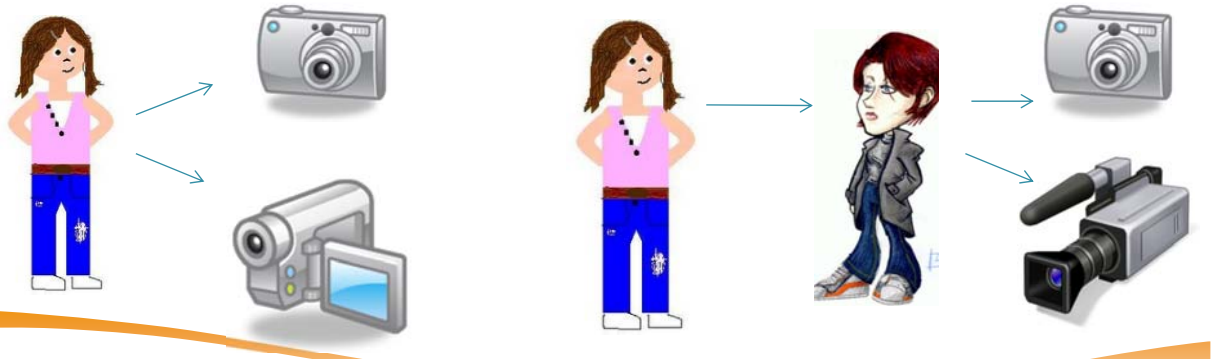


(Note: due to the long tail effect this list could be extremely long, so usually only the most frequent co-purchases are shown)

© 2018 NUS. All rights reserved.

Recommender System Approaches

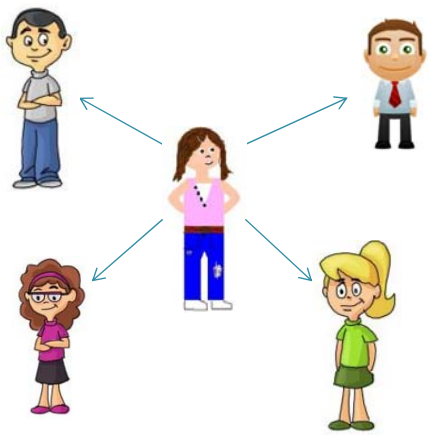
- Content-Based Recommendation
  - Match users directly to products and content
  - Recommend based on what you have bought or viewed in the past
  - Commonly used for document recommendation: webpages, news articles, blogs etc.
- Collaborative Filtering
  - Match users to people with similar tastes – recommend what they like
  - Commonly used in e-retail
  - Avoids the issue of users only being recommended more of what they already like (allows serendipity)



© 2018 NUS. All rights reserved.

Recommender System Approaches

- Community-Based/Social
  - Who are your friends and what do they like?
  - Users trust their friends opinions
- Knowledge-Based
  - Often more appropriate for recommending products where many **user specific requirements and constraints** must be taken into account
  - E.g. cars, houses, computers, financial services
  - Main types:
    - Constraint-based
    - Case-Based

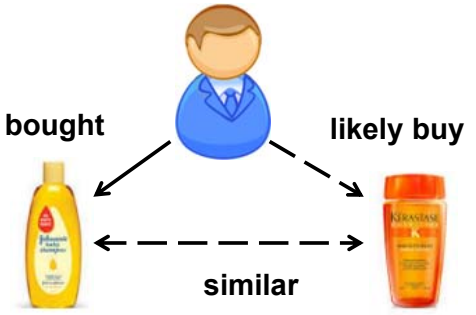


© 2018 NUS. All rights reserved.

Content-Based Recommendations

Match the attributes of a user with those of an item. Main components are:

- **Content Analyser** ~ extracts structured product descriptions from a diversity of products
  - E.g. for movie: director, length, year, genre etc.
  - Data may already be structured -provided by the manufacturer
  - **OR** extract using text mining, e.g. product descriptions are parsed and features extracted using keyword extraction
- **Profile Learner** ~ collects data about the user preferences and generalises into a user profile
- **Filtering** ~ matches the user profile against the product descriptions. Returns the products with the closest match



© 2018 NUS. All rights reserved.

Content-Based Example

	Length	studio	Director	Lead Actor1	Genre	Date
movie1	90	Disney	Spielberg	Tom Hanks	Sci-fi	2014
movie2	110	MGM	Petersen	Brad Pitt	action	2012
movie3	120	Disney	Nolan	C. Bale	action	2008
…..						
movieN						

Which Movie(s) might our user like?

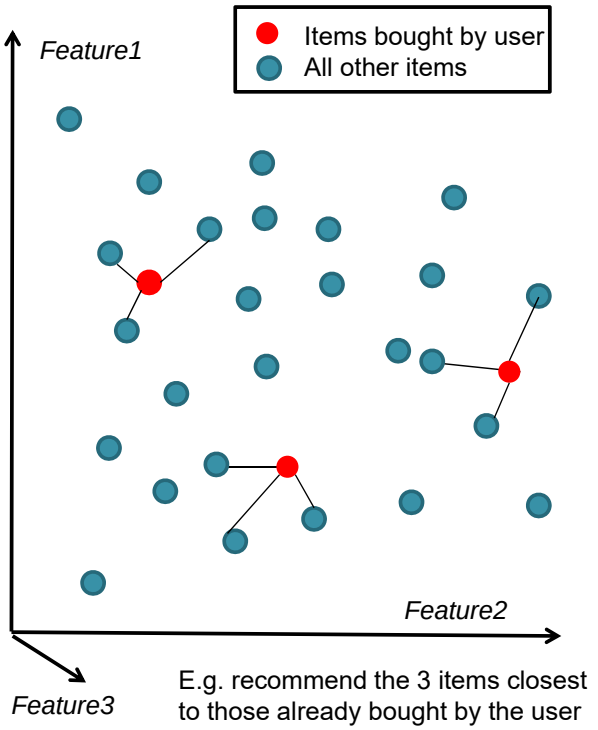
Method1: The user selects preferences for the various features using pull-down menus etc. We match against the movies using Vector-Space methods (described next)

Method2: The user rates a sample of the movies (explicitly or implicitly) as like/dislike – we then build a user profile model for that user using machine learning

© 2018 NUS. All rights reserved.

Method1: The Vector-Space Method

- Each item is represented by a set of features (a ***feature vector***)
- The user is represented by the same feature set, often obtained by:
  - The user selects preferences for the various features using pull-down menus
  - Computing an "average" vector from items already bought/liked by the user
- Then use a distance (or similarity) measure to find the nearest items to the user
- The nearest items are then recommended – ranked by their distance



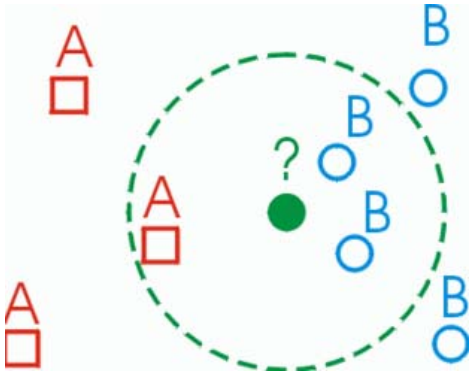
© 2018 NUS. All rights reserved.

Nearest Neighbour Classifiers (k-NN)

- A generic classification/prediction approach
- The nearest (most similar) neighbours to the 'target' record are found
- The attributes of these neighbors are used to make a decision or prediction. Often a labeled 'decision' field is used (supervised learning).
- A number K is often selected to specify how many neighbours to consider

E.g. predicting a user's credit risk as low, medium or high given a database of past patients who are labelled as low, medium or high credit risk.

Use majority vote to make the decision (decision is the most commonly occurring target category amongst the neighbours)



Using K = 3

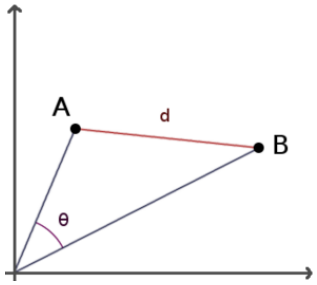
© 2018 NUS. All rights reserved.

Similarity & Distance Measures

- Content-Based Recommender Systems typically use the cosine distance to measure the similarity between user and items
- In information retrieval systems* the similarity between two documents (A and B) is given by:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

**Content-based recommender systems have their roots in Information Retrieval (IR). The goal of IR is to identify relevant documents in response to a user query*

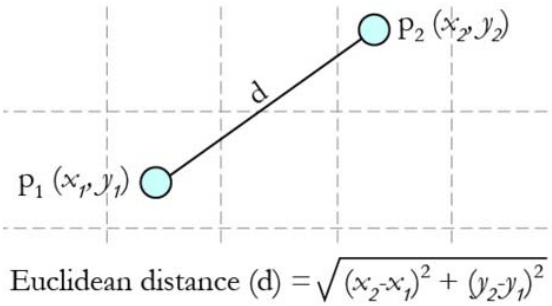


When the documents are similar the angle between their vectors is small and the cosine similarity approaches +1

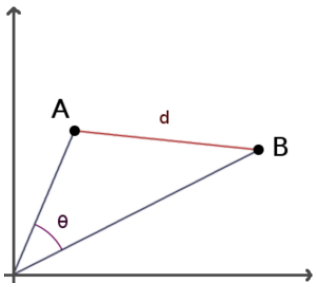
© 2018 NUS. All rights reserved.

Why Cosine Similarity?

- For general classification problems, the distance metric used in k-NN systems is commonly the Euclidean distance



- If the vectors are documents then Cosine Similarity is often better because it ignores the magnitude of the vectors (only considers the angle)



E.g. if one doc has word "disease" 10 times and another has it 100 times then using cosine they will still be similar but they will be different using euclidean distance

	w1	w2	w3	w4	w5
Document1	0	10	20	100	200
Document2	0	100	200	1000	2000

In the above, Cosine Similarity = 1; Euclidean Distance = 2022.5

© 2018 NUS. All rights reserved.

Vector-Space/k-NN: General Pros and Cons

Advantages (Pros)	Disadvantages (Cons)
The nearest neighbours can be examined and this can help 'explain' the decision made* Giving an explanation/justification for a decision is important in many domains, e.g. medical diagnosis	Does not scale well when the database gets very large. The distance between the target record and all other records must be computed *at the time* a decision is made *(k-NN is often called **memory-based** reasoning - no "model" is actually built)*

**Case-Based Reasoning Systems became popular for this reason + experts often find it easier to express their knowledge as cases (examples) rather than rules*

© 2018 NUS. All rights reserved.

Method1 – Other Cons/Issues

	Length	studio	Director	Lead Actor1	Genre	Date
movie1	90	Disney	Spielberg	Tom Hanks	action	2009
movie2	110	MGM	Petersen	Brad Pitt	action	2004
movie3	120	Disney	Nolan	C. Bale	Sci-fi	2008
movie4						
User Preference	130	-	Spielberg	-	action	-

- In practice there are many attributes, the user may not have patience to specify all of their preferences
- To apply cosine distance we need to do "one-hot" encoding to convert the categories to (0/1) numbers – this can result in huge feature vectors
- Other similarity & distance metrics can be considered
  e.g. for for binary attributes: Jaccard Coefficient = a/(a+b+c)

		rec1 1	rec1 0
rec2	1	a	b
	0	c	d

a ~ #attributes with value = 1 for both records

© 2018 NUS. All rights reserved.

Method2: Machine Learning Approach

	Length	Studio	Director	Lead Actor1	Genre	Date	Likes
movie1	90	Disney	Spielberg	Tom Hanks	action	2009	T
movie2	110	MGM	Petersen	Brad Pitt	action	2004	F
movie3	120	Disney	Nolan	C. Bale	Sci-fi	2008	F
movie4							T
movie5							F

If we have sufficient labelled items then we can use supervised learning to build a predictive model for that user

E.g. a decision tree model:

```
If Length < 100
    If Genre == Action
        If Date < 2006
            Then Likes = True
        Else….
    Else …
Else…
```

Apply the model to all movies not seen by the user. Recommend the movies with the highest score (prediction confidence)

Requires one model per user

© 2018 NUS. All rights reserved.

Pros & Cons of Content-Based Filtering

Pros

- Can recommend new items even if there are no ratings provided by users
- Possible to explain why recommendations are made

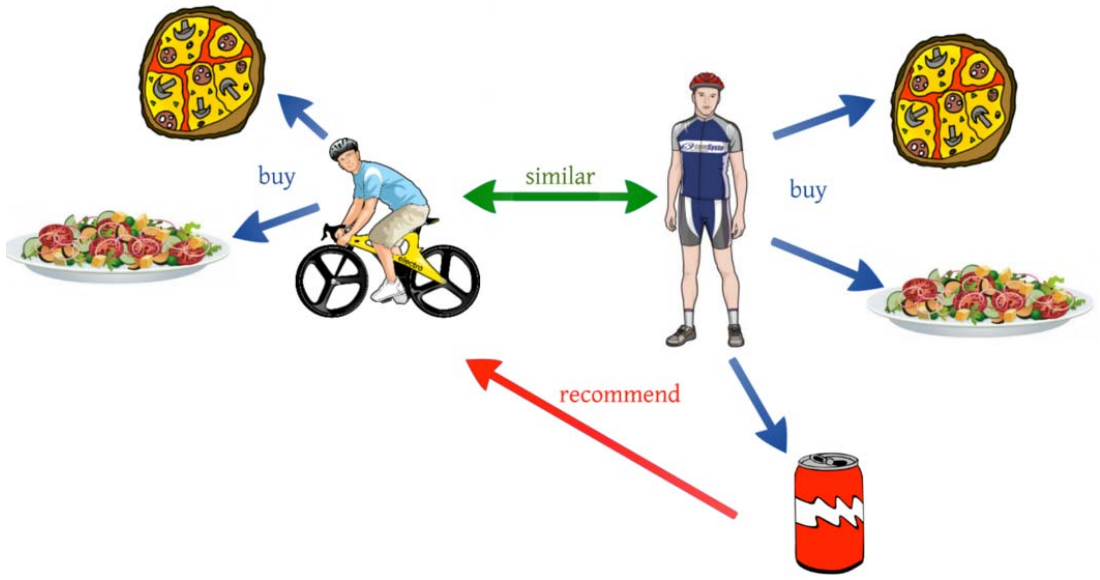
Cons

- Hard to recommend items not similar to those already seen by the user
- Getting an adequate set of product descriptor features.
- Can be hard to know the user preferences against the product features

© 2018 NUS. All rights reserved.

Collaborative Filtering (CF)

Match the user to similar users and recommend what they like



© 2018 NUS. All rights reserved.

Measuring User Similarity

- How do we find similar users? What user data can we use?
- Most common data used is the user utility matrix – a matrix of the user likes or interests in the various products/items on offer
- E.g. the "user-ratings" matrix:

User	movie1	movie2	movie3	movie4	movie5	movie6	etc....
1	2	5	4		3	1	
2		3		5	3	1	
3			5	3			

Entries are numbers, e.g. 1->5, where 1 means do not like, 5 means like a lot

- How do we get this matrix?
  - Explicitly – ask the user for their rating of various items when they visit the website (this typically results in a vey sparse matrix)
  - Implicitly – infer interests by looking at the pages and products they view and buy

© 2018 NUS. All rights reserved.

Finding Similar Users

- How to compute distances/similarities between entries in the utility matrix?
- Sparsity (many missing values) is a problem
- E.g. what is the similarity between user1 and user3?

User	movie1	movie2	movie3	movie4	movie5	movie6	etc....
1	2	5	4		3	1	
2		3		5	3	1	
3			5	3			

- Common Metrics
  - Euclidean Distance
  - Cosine Similarity
  - Pearson Correlation Coefficient (most common)

© 2018 NUS. All rights reserved.

Finding Similar Users

- Some users may use different rating values to quantify the same level of appreciation for an item. E.g. two users may both like a movie equally well but one user may rate it as 5 and the other 4. This can create a bias.
- **Pearson Correlation Coefficient** is similar to cosine similarity but eliminates this bias by subtracting each users mean rating from their individual ratings. Value ranges from -1 to +1

$$Sim(\mathbf{u},\mathbf{v}) = \frac{\sum_{i \in C} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in C} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in C} (r_{v,i} - \bar{r}_v)^2}}$$

$r_{u,i}$ = rating of user u on item i
$r_{v,i}$ = rating of user v on item i

$\bar{r}_u$, $\bar{r}_v$ = mean ratings of user's u, v

C is the set of all co-rated items

User	movie1	movie2	movie3	movie4	movie5	movie6	Mean
1	2	5	4		3	1	3
2		3		5	4	2	3.5
3			5	3		4	4

1	-1	2	1		0	-2
2		-0.5		1.5	0.5	-1.5
3			1	-1		0

© 2018 NUS. All rights reserved.

Making a Recommendation

- Once we have the nearest neighbours to a user then we compute a recommendation based on the preferences of these neighbours. One method is to sum the preferences of the K neighbours weighted by their similarity – the weighted average

$$\text{Predicted rating } (\mathbf{u},i) = \bar{r}_u + \frac{\sum_{v \in \boldsymbol{V}} \text{sim}(\mathbf{u},\mathbf{v}) * (r_{\mathbf{v},i} - \bar{r}_v)}{\sum_{v \in \boldsymbol{V}} |\text{sim}(\mathbf{u},\mathbf{v})|}$$

V is the set of nearest neighbours

		User Ratings (after normalisation)					
User	**similarity**	**movie1**	**movie2**	**movie3**	**movie4**	**movie5**	**movie6**
target	1	1	**?**	**?**	3	2	**?**
user1	-0.9	-1	2	1		0	-2
user2	0.3		-0.5		1.5	0.5	-1.5
user3	-1			1	-1		0
weighted average		-	2*-0.9-0.5*0.3 / 0.9+0.3 = -1.63	1*-0.9+1*-1 / 0.9+1 = -1	-	-	-2*-0.9-1.5*0.3+0*-1 / 0.9+0.3+1 = 0.61
predicted rating			2–1.63= 1.37	2-1 =1			2 + 0.61= 2.61

Assume these have been computed elsewhere

**similarities have been rounded for ease of illustration*

© 2018 NUS. All rights reserved.

Other Normalisation Methods

Z-score normalization

- Divide the user *mean-centered* rating by the user's rating standard deviation. Hence the normalised rating for user *u* on movie *i* is...

$$\frac{r_{ui} - \bar{r}_u}{\sigma_u}$$

- The predicted rating for an unrated movie *i* for user *u* is then

$$\hat{r}_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} (r_{vi} - \bar{r}_v) / \sigma_v}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}$$

- Z-score normalisation is useful if the rating scale has a wide range of discrete values or is continuous. But, because the ratings are divided and multiplied by possibly very different standard deviation values, *Z*-score can be more sensitive than mean-centering and can often predict ratings that are outside the rating scale.

© 2018 NUS. All rights reserved.

Item-Based Collaborative Filtering

- User-Based Collaborative Filtering does not scale well
  - User likes/interests may change often hence similarities between users are best computed at the instant of the recommendation – this becomes computationally prohibitive for large numbers of users
- Item-Based Collaborative Filtering
  - Transpose the user-rating matrix, compute distances between items not users
  - Faster than User-Based – there are usually far fewer items than users AND the similarities between items change much less frequently than between users hence all item-item distances can be pre-computed
  - A well known application of this method is Amazon's recommendation engine

item	user1	user2	user3	user4	….	userN
movie1	2	5	4			
movie2		3			1	
…..						

Transposition of the user rating matrix

© 2018 NUS. All rights reserved.

Example Item-Item Similarity Matrix

	item1	item2	item3	item4	item5	item6	…
item1	1	0.90	0.61	0.13	0.68	0.75	…
item2		1	0.58	0.78	0.12	0.29	…
item3			1	0.55	0.32	0.69	…
item4				1	0.11	0.98	…
item5					1	0.75	…
item6						1	…
….							…

The matrix is symmetrical. Similarity is measured from 0 to 1 or -1 to 1, diagonals hence take the value 1

Typically precomputed, the frequency of updating depends on type of item (e.g. fast or slow seller, speed of new releases) – weekly or monthly updates may suffice.

© 2018 NUS. All rights reserved.

Item-Based Collaborative Filtering

- Compute the distance between items ***i*** and ***j*** using the adjusted cosine similarity:

$$\text{sim}(\mathbf{i}, \mathbf{j}) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$$

$r_{u,i}$ = rating of user u on item j
$r_{u,j}$ = rating of user u on item j
$\bar{r}_u$ = mean rating of user u
U = set of all users

- Predict user ***u***'s rating for an item ***i*** using:

$$\text{Predicted rating } (\mathbf{u},i) = \frac{\sum_{j \in J} \text{sim}(i,j) * r_{u,j}}{\sum_{j \in J} |\text{sim}(i,j)|}$$

J is the set of K similar items to i (among items that u has rated)

**an alternative to taking the absolute value (mod) in the denominator is to ignore items with negative similarity to the target (i)*

© 2018 NUS. All rights reserved.

Please read....

Industry Report

Amazon.com Recommendations

Item-to-Item Collaborative Filtering

Greg Linden, Brent Smith, and Jeremy York • *Amazon.com*

http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

Also uploaded to IVLE

© 2018 NUS. All rights reserved.

A Simple Example*: CF using Users vs Items

The raw ratings….

User	Lady In Water	Snakes On A Plane	Just My Luck	Superman	You Me And Dupree	The Night Listener
Rose	2.5	3.5	3	3.5	2.5	3
Seymour	3	3.5	1.5	5	3.5	3
Philips	2.5	3		3.5		4
Puig		3.5	3	4	2.5	4.5
LaSalle	3	4	2	3	2	3
Matthews	3	4		5	3.5	3
Toby		4.5		4	1	

The goal is to make movie recommendations for Toby.

Do this by predicting his likely rating for his unseen movies:
***Lady** in Water, Just My **Luck**, The **Night** Listener*

**Taken from "Programming Competitive Intelligence", O'Reilly*

© 2018 NUS. All rights reserved.

User-Based 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

The similarity of the other users to Toby

The weighted average of the other users ratings for each unseen movie

© 2018 NUS. All rights reserved.

Item-Item Recommendations for Toby

↓ Movies seen by Toby → Movies not seen by Toby

Movie	Rating	Night	R.xNight	Lady	R.xLady	Luck	R.xLuck
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
Total		0.433	1.378	0.713	1.764	0.352	0.914
Normalized			3.183		2.473		2.598

(= 1.378/0.433)

Toby's ratings for the movies he has seen

The similarities between ***Night Listener*** and the movies seen by Toby – these are derived from the pre-computed item-item similarity matrix

Toby's estimated rating for ***Night Listener*** is obtained by computing his average rating for all of his seen movies – weighted by their distance to ***Night Listener***

© 2018 NUS. All rights reserved.

Issues With Collaborative Filtering

- Sparsity of Ratings
  - In practice there will be thousands of items but most users will have made few ratings, this makes similarity measurement hard

User	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10
1	1	4	2							
2					5	2	3	1		

  - What is the similarity (or distance) between user 1 and user2?
  - Distance and correlation calculations do not work with missing values
- Absence of a rating does not mean the user doesn't like the item

© 2018 NUS. All rights reserved.

Matrix Factorisation

- While user-based or item-based collaborative filtering methods are simple and intuitive, Matrix Factorization techniques are usually more effective because they allow us to discover the **latent features** underlying the interactions between users and items.
- Of course, matrix factorization is simply a mathematical tool for playing around with matrices, and is therefore applicable in many scenarios where one would like to find out something hidden under the data.
- We decompose a big matrix (e.g. items * features) into the product of two smaller matrices
  - Items * Concepts
  - Concepts * Features

© 2018 NUS. All rights reserved.

Matrix Factorisation Example

- Assume the items to be recommended have a set of properties.
  E.g. Movies could have properties: type, actors, directors, film studios, location, length etc. Properties are numerical strengths, e.g. values 0 (low) to 10 (high)

	Length	Disney	Universal	Action	Comedy	SciFi	T. Hanks	B. Pitt
Movie1	4	10	0	2	8	0	8	2
Movie2	8	0	10	9	1	7	0	9

- Assume users express their preferences using the same properties

	Length	Disney	Universal	Action	Comedy	SciFi	T. Hanks	B. Pitt
User1	2	7	5	4	8	5	5	9
User2	9	3	7	9	1	7	2	5

© 2018 NUS. All rights reserved.

Matrix Factorisation Approaches

- The users rating for any movie can now be computed as a simple product of user preferences * movie properties
- E.g. User1's rating score for movie 2 is

  = 2*8 + 7*0 + 5*10 + … (to normalise divide by the max score)

	Length	Disney	Universal	Action	Comedy	Sci.Fi	T. Hanks	B. Pitt
M1	4	10	0	2	8	0	8	2
M2	8	0	10	9	1	7	0	9

	Length	Disney	Universal	Action	Comedy	Sci.Fi	T. Hanks	B. Pitt
U1	2	7	5	4	8	5	5	9
U2	9	3	7	9	1	7	2	5

© 2018 NUS. All rights reserved.

Matrix Factorisation Approaches

- BUT we don't have the movie properties or user preferences, we don't even know what the properties should be. We only have a set of ratings from the users (the ratings matrix **R**). E.g.

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	…….	MovieN
U1	-	3	-	-	8	0	…….	3
U2	8	0	10	9	1	7	…….	-
U3							…….	6

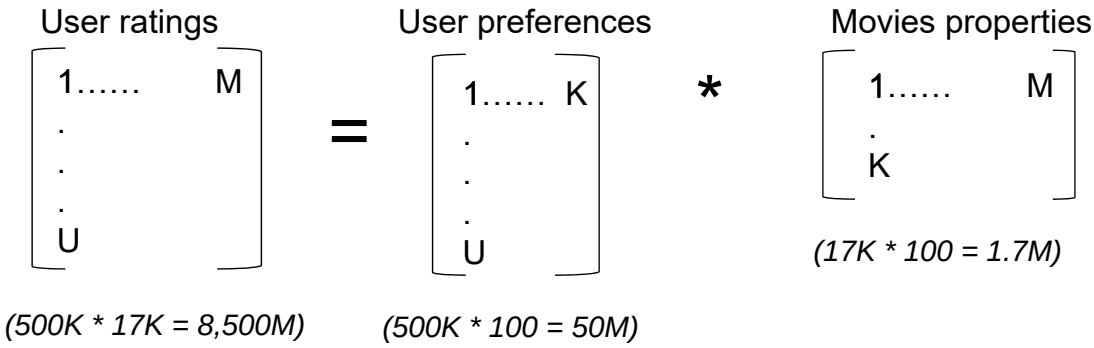
- Solution ~ Assume the movies have K properties (e.g. say 100) and use matrix factorisation on **R** to derive the user preference matrix (U) and the movie properties matrix (M):

$$\mathbf{R} = \mathbf{U}^{\mathbf{T}}\mathbf{M}$$

- We don't need to know what the K properties are, we just assume they exist, they are called ***latent*** (hidden) variables

© 2018 NUS. All rights reserved.

Performing the Factorisation



- SVD (Singular Value Decomposition) is common method for matrix factorisation
- Issues ~ much of the ratings matrix is empty (very sparse matrix)
- Straight SVD doesn't work. Hard to find exact factors
- Instead Netflix winner used incremental learning solution based on gradient descent by taking derivatives of the approximation error

© 2018 NUS. All rights reserved.

Factorising using Gradient Descent

- Use stochastic gradient descent to minimise the squared error between the predicted rating and the actual rating
- The update rule for the user preference matrix **U** and the movie properties matrix **M**:

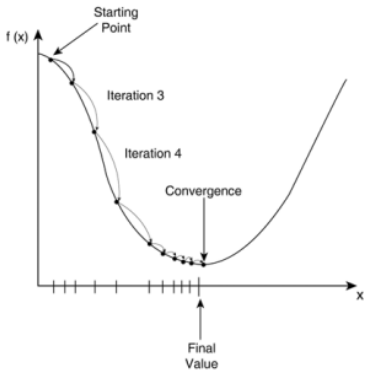
$$u_{ki}^{t+1} = u_{ki}^{t} + 2\gamma\,(r_{ij} - p_{ij})\; m_{kj}^{t}$$

$$m_{kj}^{t+1} = m_{kj}^{t} + 2\gamma\,(r_{ij} - p_{ij})\; u_{ki}^{t}$$

where
u_{ki} = value of kth preference for user i
m_{kj} = value of kth property for movie j
r_{ij} = rating of user i on movie j
p_{ij} = predicted rating of user i on movie j
γ is the learning rate

Procedure:
Present the training examples in turn and update the weights after each (similar to NN learning). Continue until convergence (error stops reducing)

Training examples are triplets: (user, movie, rating)



© 2018 NUS. All rights reserved.

Alternating Least Squares Algorithm

Model R as product of user and movie feature matrices A and B of size U×K and M×K

We wish to find values for a_i and b_j that minimises the total squared error:

$$\text{cost} = \sum_{ij} (r_{ij} - a_i * b_j)^2$$

- If we fix **B** and optimize for **A** alone, the problem is simply reduced to the problem of linear regression.
- **R**ecall that in linear regression we solve for **β** by minimizing the squared error $|| (y - \beta X)^2 ||$ given **X** and **y**.
- The solution is given by the Ordinary Least Squares (OLS) formula $\beta = (X^T X)^{-1} X^T y$

Alternating Least Squares (ALS)
- » Start with random A & B
- » Optimize user vectors (A) based on movies
- » Optimize movie vectors (B) based on users
- » Repeat until converged

https://datasciencemadesimpler.wordpress.com/tag/alternating-least-squares/

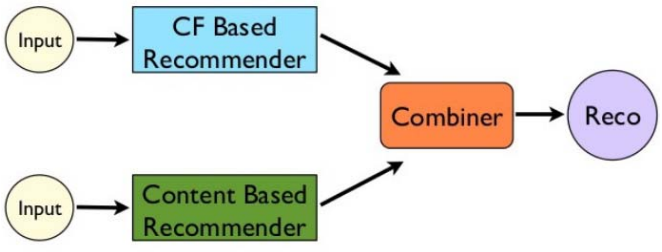
© 2018 NUS. All rights reserved.

Hybrid Recommender Systems

Combining *collaborative filtering* and *content-based filtering* can be more effective in some cases.

- E.g. make *content-based* and *collaborative-based* predictions separately and then combine
- E.g. add *content-based* capabilities to a *collaborative-based* approach (and vice versa); or by unifying the approaches into one model (see link below for a review of approaches).

https://www.cs.ubc.ca/~rap/teaching/504/2008/readings/recommender.pdf



Some techniques *(from Wikipedia):*

- **Weighted:** Numerically combine the scores
- **Switching:** Choose one among different recommendations
- **Mixed:** Present all recommenders together
- **Cascade:** Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
- **Meta-level:** One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

© 2018 NUS. All rights reserved.

Challenges and Issues (1)

- **Scalability** – handling very large numbers of users and items
- **Cold-start** – making recommendations to a new user
- **Sparsity** - recommending items in **the long-tail** proves very hard since there are very few ratings / purchases for them


THE VERGE TECH SCIENCE CULTURE CARS REVIEWS LONGFORM VIDEO MORE

Netflix offers details on its recommendation engine, says it guides 75 percent of viewership

https://www.theverge.com/2012/4/8/2934375/netflix-recommendation-system-explained

ARTICLE INTERNET, MEDIA

Do recommendation systems make the 'tail' longer or shorter?

By Paul Belleflamme 26 April 2012 39 (Updated March 2015)

http://www.ipdigit.eu/2012/04/do-recommendation-systems-make-the-tail-longer-or-shorter/

© 2018 NUS. All rights reserved.

Challenges and Issues (3)

- **Diversity** - users tend to be more satisfied with recommendations when there is a higher diversity, e.g. items from different artists. Don't aspire to provide the perfect recommendation, allow the user to treat the RS as a knowledge discovery tool
- **Serendipity** - "how surprising are the recommendations"
- **Long versus short term** recommendations
- **Avoiding bad recommendations** – assigning a cost to them
- **Repeat Recommendations** - sometimes it may be more effective to re-show recommendations or let users re-rate items, than showing new items
- **Recommending Sequences** of items - e.g. compilation of musical tracks

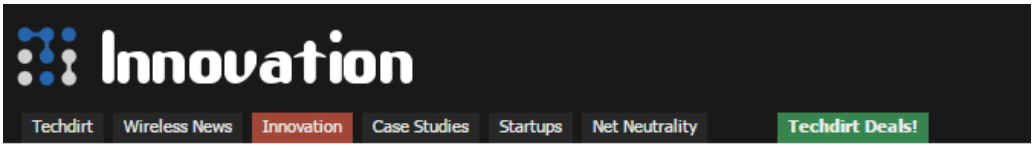
© 2018 NUS. All rights reserved.

Challenges and Issues (4)

- **Privacy** - push-back by users if they feel the RS is collecting too much information about them
- **Trust** - user must trust the system. Trust can be built by explaining how the recommendations are generated and why it recommends an item.
- **Proactive** Recommendations ~ providing recommendations even when they not asked for (push versus pull)
- **Fraud** - fake reviews and fake ratings

© 2018 NUS. All rights reserved.

Challenges and Issues (5)



Innovation
by Mike Masnick
Fri, Apr 13th 2012
12:07am

Why Netflix Never Implemented The Algorithm That Won The Netflix $1 Million Challenge

from the *times-change* dept

- Despite all the plaudits and case studies, Netflix announced this week that despite paying $1 million dollars to a winning team of multinational researchers in 2009, they never bothered to implement their solution.
- Why? Because, according to Netflix the "additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment."

Instead …they gave us the source code. We looked at the two algorithms with the best performance in the ensemble: *Matrix Factorization* (generally called SVD, *Singular Value Decomposition*) and *Restricted Boltzmann Machines* (RBM).To put these to use, we had to overcome some limitations, for instance that they were built to handle 100 million ratings, instead of the 5 billion+ that we have, and they were not built to adapt as members added more ratings. Once we overcame those challenges, we put the two algorithms into production, where they are still used as part of our recommendation engine.
https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

© 2018 NUS. All rights reserved.

Good Reading....

The winning Netflix solution:

The BellKor solution to the Netflix Prize

Robert M. Bell, Yehuda Koren and Chris Volinsky
AT&T Labs – Research
BellKor@research.att.com

A detailed tutorial:

The Recommender Problem *Revisited*

Xavier Amatriain
Research/Engineering Director @ Netflix

Bamshad Mobasher
Professor @ DePaul University

NETFLIX

Xavier Amatriain – August 2014 – KDD

http://www.slideshare.net/xamat/kdd-2014-tutorial-the-recommender-problem-revisited

© 2018 NUS. All rights reserved.