DIPARTIMENTO DI MATEMATICA E INFORMATICA

UNIVERSITÀ degli STUDI di CATANIA

Recommender Systems

Alessandro Ortis
ortis@dmi.unict.it

IMAGE PROCESSING LABORATORY
http//iplab.dmi.unict.it

Introduction

Recommender System: a system able to predict ***user*** *ratings* or *preferences* to ***items***. Recommender systems have become increasingly popular in recent years, and are used in a variety of areas including movies, music, news, books, research articles and products in general.

Examples:

- Offering news articles to on-line newspaper readers, based on a prediction of reader interests (Google News)
- 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 (Amazon)

Introduction

amazon.com

NETFLIX

PANDORA

movielens
helping you find the *right* movies

StumbleUpon

last.fm
the social music revolution

Google News

del.icio.us

YouTube

XBOX LIVE

Introduction

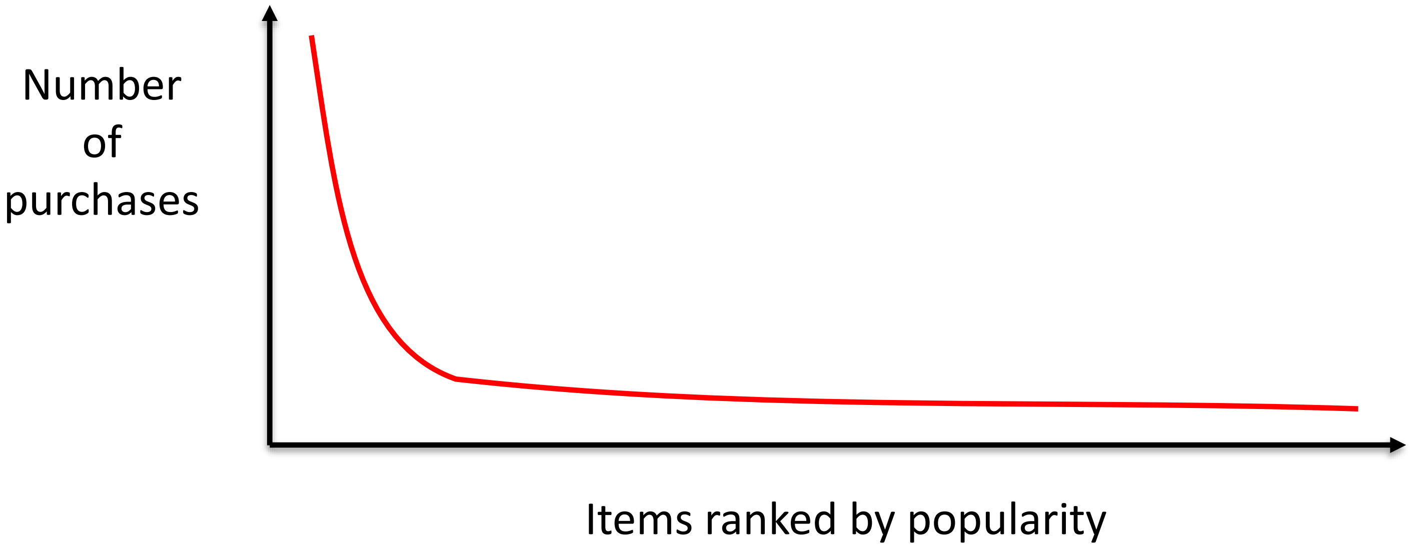
Introduction

Traditional Retailers **VS** **Online Market**

- Shelf space is a scarce commodity for traditional retailers
- Web enables near-zero-cost dissemination of information about products
- More choice necessitates better filters

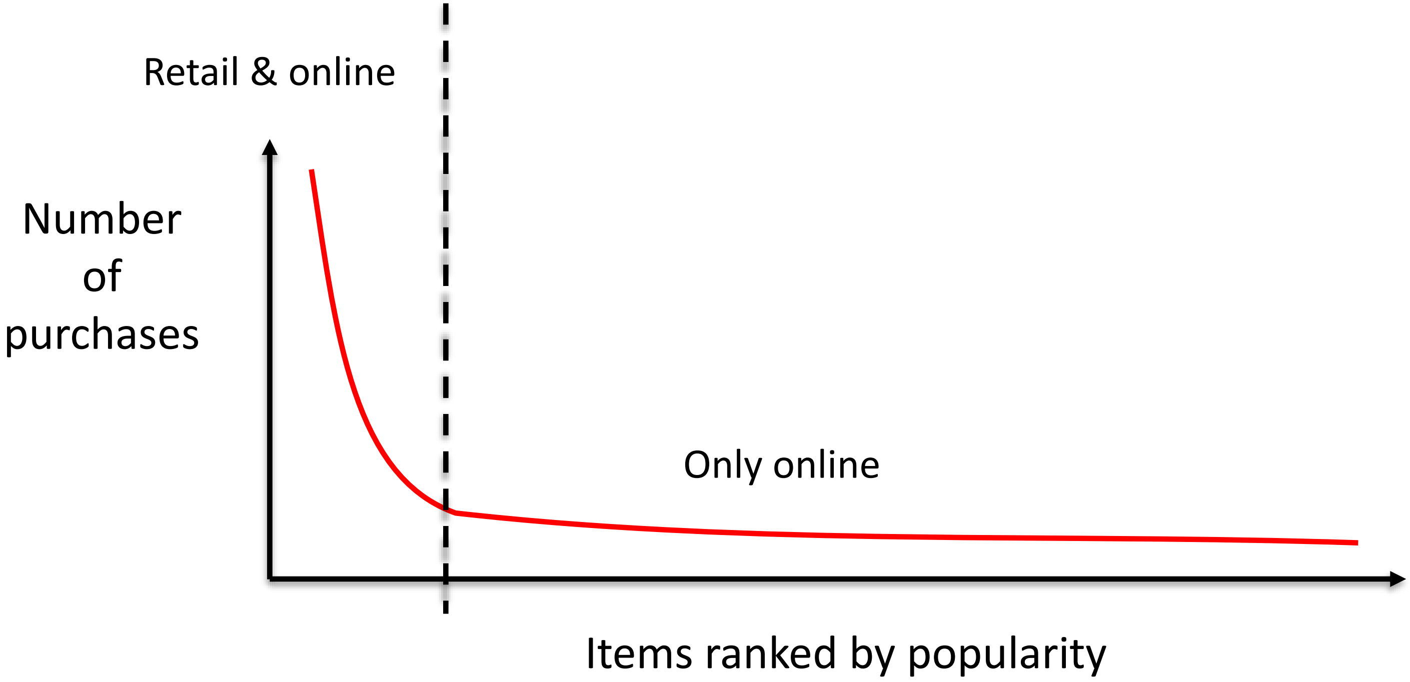
Introduction

The Long Tail Phenomenom



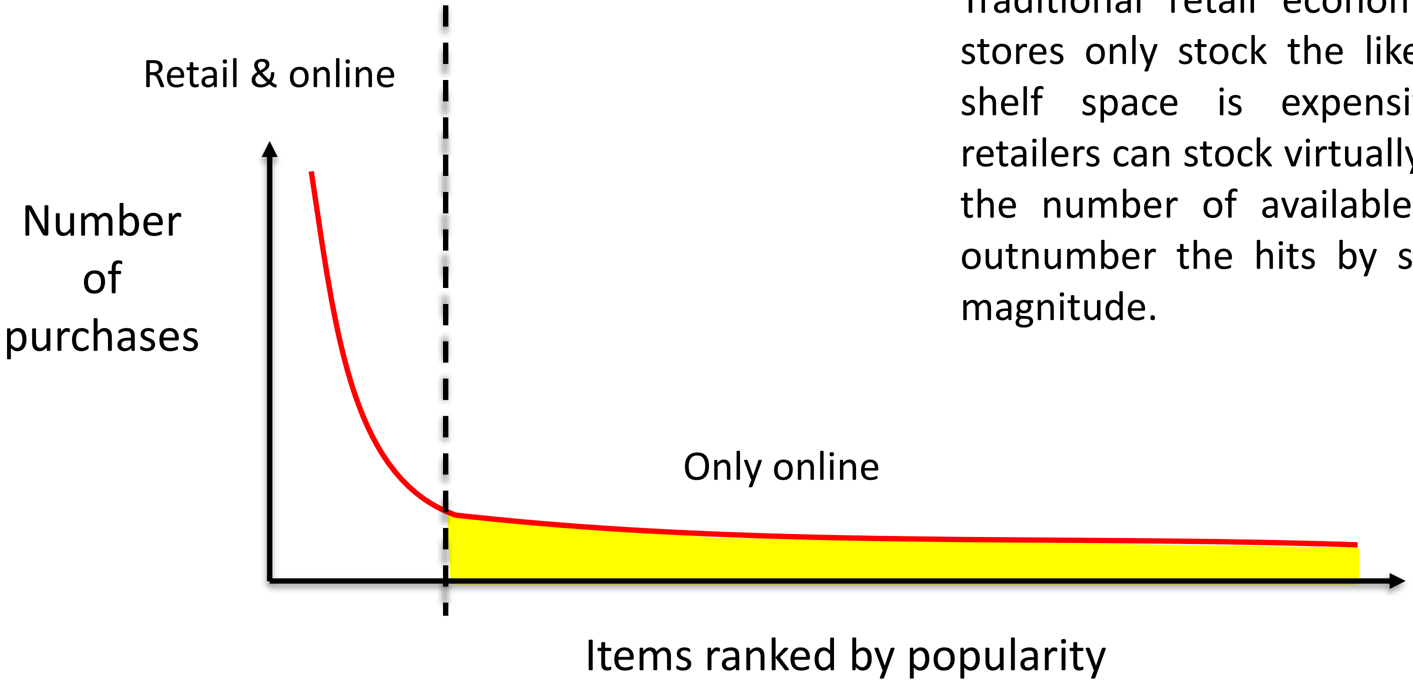
Introduction

The Long Tail Phenomenom



Introduction

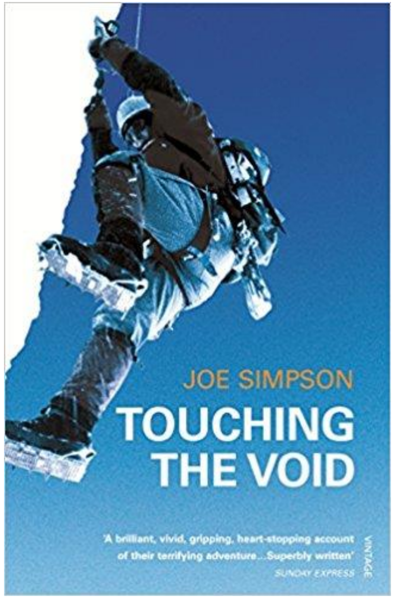
The Long Tail Phenomenom

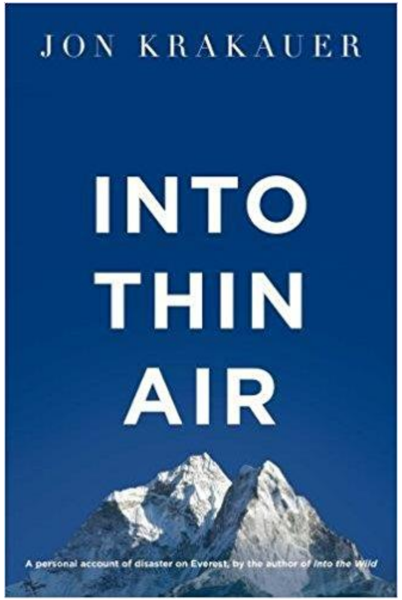


Traditional retail economics dictate that stores only stock the likely hits, because shelf space is expensive. But online retailers can stock virtually everything, and the number of available niche products outnumber the hits by several orders of magnitude.

Introduction

How **Into Thin Air** made **Touching the Void** a bestseller



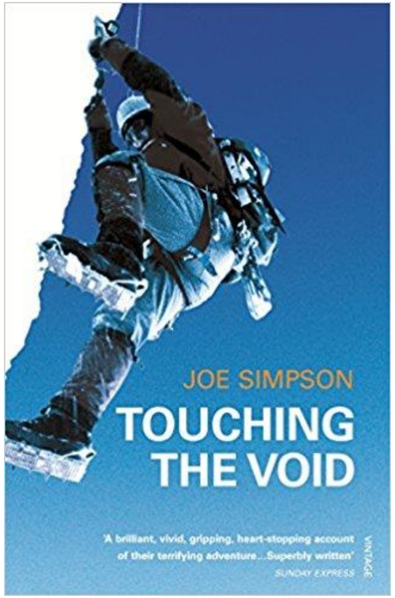


1988

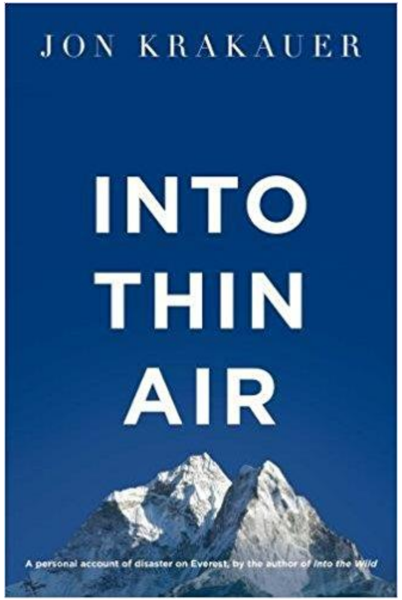
1997

Introduction

How **Into Thin Air** made **Touching the Void** a bestseller



1988



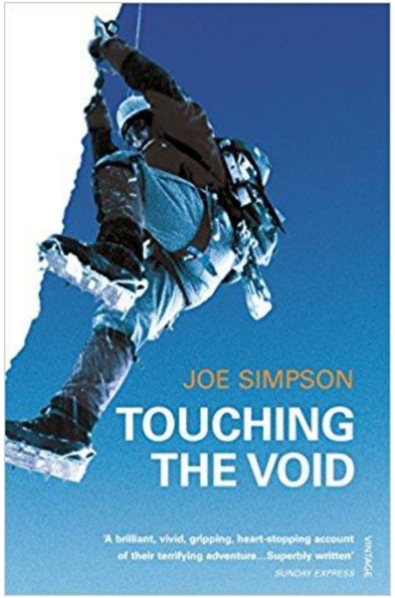
1997

Amazon recommendation 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 enthusiastically, wrote positive reviews.

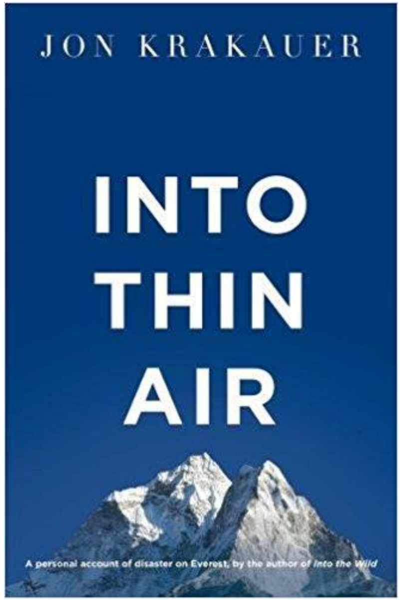
More sales, more feedback, more the algorithm fueled recommendations…

Introduction

How **Into Thin Air** made **Touching the Void** a bestseller



1988



1997

Amazon recommendation 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 enthusiastically, wrote positive reviews.

More sales, more feedback, more the algorithm fueled recommendations…

…***Touching the void*** became a movie!

Introduction

Types of recommendations:

1. **Editorial and hand curated**
   List of favorites
   Lists of “essential” items

2. **Simple aggregates**
   Top 10, Most Popular, Recent Uploads

3. **Tailored to individual users**
   Amazon, Netflix, …

Introduction

Formal Model

X = set of **Customers/Users**

S = set of **Items/Products**

Utility function *u*: $\boldsymbol{X} \times \boldsymbol{S} \rightarrow \boldsymbol{R}$

Utility Matrix: **U =** $[u(x,i)]$

$x \in \boldsymbol{X},\ i \in \boldsymbol{S}$

R = set of ratings

R is a totally ordered set

e.g., **0-5** stars, real number in **[0,1]**

Introduction

Utility Matrix

Items

Users		Avatar	LOTR	Matrix	Pirates
	Alice	1		0.2	
	Bob		0.5		0.3
	Carol	0.2		1	
	David				0.4

Introduction

Utility Matrix

	Avatar	LOTR	Matrix	Pirates
Alice	5		1	
Bob		2		1
Carol	1		5	
David				2

The utility matrix is sparse, meaning that most entries are unknown. That is, we have no explicit information about the user’s preference for the item.

Introduction

The goal of recommendation systems:

would user *A* like *Star Wars 2*?

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Introduction

Key problems

1. **Gathering "known" ratings for matrix**
   How to collect the data in the utility matrix

2. **Extrapolate unknown ratings from the known ones**
   Mainly interested in high unknown ratings.
   We are not interested in knowing what you don't like but what you like.

3. **Evaluating extrapolation methods**
   How to measure success/performance of recommendation methods

Introduction

Gathering ratings

Explicit

- Ask people to rate items
- Doesn’t work well in practice – people can’t be bothered

Implicit

- Learn ratings from user actions
  - E.g., purchase implies high rating
- What about low ratings?

Introduction

Extrapolating ratings

Key problem: Utility matrix ***U*** is **sparse**

Most people have not rated most items

Cold start:

New items have no ratings

New users have no history

Three approaches to recommender systems:

1) Content-based

2) Collaborative

3) Latent factor based

DIPARTIMENTO DI MATEMATICA E INFORMATICA
UNIVERSITÀ degli STUDI di CATANIA

Contet Based Recommendation Systems

IMAGE PROCESSING LABORATORY
http//iplab.dmi.unict.it

Contet Based Recommendation Systems

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***

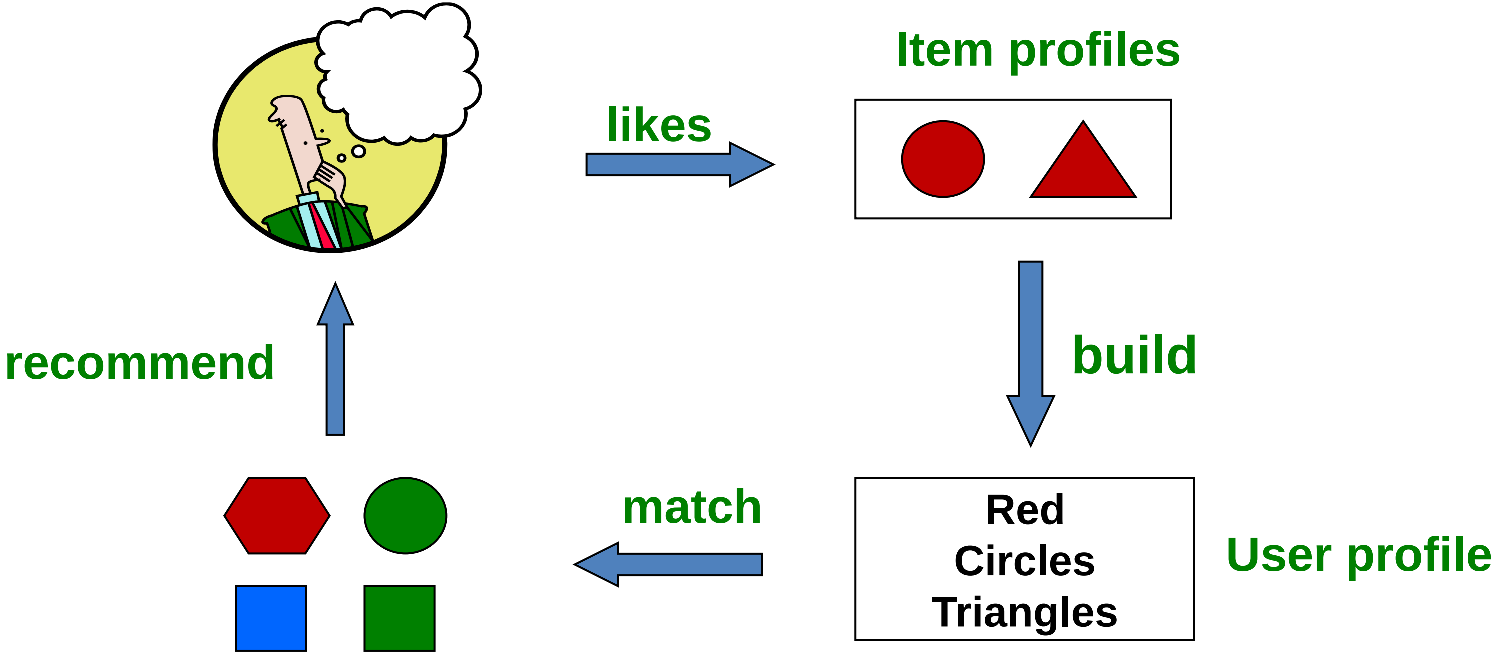
Examples:

- **Movie recommendations**
  Recommend movies with same actor(s), director, genre, …

- **Websites, blogs, news**
  Recommend other sites with "similar" content

Contet Based Recommendation Systems

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***



Contet Based Recommendation Systems

1. For each item, create an **item profile,** that is a **set (vector) of features**

   - **Movies:** author, title, actor, director,…
   - **Books:** author, genre, …
   - **Text:** set of “important” words in document (e.g., TF-IDF)

Contet Based Recommendation Systems

1. For each item, create an **item profile,** that is a **set (vector) of features**

2. For each user, create a **user profile**

   Examples:

   - Average of rate item profiles
   - Weighted average of rated item profiles with the ratings
   - Weight by difference from average rating for item

Contet Based Recommendation Systems

1. For each item, create an **item profile,** that is a **set (vector) of features**

2. For each user, create a **user profile**

3. Prediction heuristic: given user profile ***x*** and item profile ***i***, estimate

$$u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$

Contet Based Recommendation Systems

Pros	Cons
+: No need for data on other users No cold-start or sparsity problems **+: Able to recommend to users with unique tastes** **+: Able to recommend new & unpopular items** No first-rater problem **+: Able to provide explanations** Can provide explanations of recommended items by listing content-features that caused an item to be recommended	**–: Finding the appropriate features is hard** E.g., images, movies, music **–: Recommendations for new users** **How to build a user profile?** **–: Overspecialization** Never recommends items outside user's content profile People might have multiple interests **Unable to exploit quality judgments of other users**

DIPARTIMENTO DI MATEMATICA E INFORMATICA

UNIVERSITÀ degli STUDI di CATANIA

Collaborative Filtering

IMAGE PROCESSING LABORATORY

http//iplab.dmi.unict.it

Collaborative Filtering

Main Idea: harnessing quality judgments of other users.

These systems recommend items based on similarity measures between users and/or items. The items recommended to a user are thoese preferred by similar users.

Collaborative Filtering

- Represent **users** by their rows in the utility matrix (i.e., the set of ratings given by the user)

For each user $\boldsymbol{x}$, we define the user rating vector $\boldsymbol{r_x}$

Collaborative Filtering

- Represent **users** by their rows in the utility matrix (i.e., the set of ratings given by the user)

For each user ***x***, we define the user rating vector $\boldsymbol{r_x}$

		Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
$\boldsymbol{r_A}$	A	4			5	1		
	B	5	5	4				
$\boldsymbol{r_C}$	C				2	4	5	
	D		3					3

Collaborative Filtering

- Represent **users** by their rows in the utility matrix (i.e., the set of ratings given by the user)

  For each user ***x***, we define the user rating vector $\boldsymbol{r_x}$

- Recommendation for a user ***y*** is made by looking at the users whose rating vectors are most similar to $\boldsymbol{r_y}$

Collaborative Filtering

- Represent **users** by their rows in the utility matrix (i.e., the set of ratings given by the user)

  For each user $\boldsymbol{x}$, we define the user rating vector $\boldsymbol{r_x}$

- Recommendation for a user $\boldsymbol{y}$ is made by looking at the users whose rating vectors are most similar to $\boldsymbol{r_y}$

- This process of identifying similar users and recommending what similar users like is called *collaborative filtering*

Collaborative Filtering

How to measure similarity between users rating vectors?

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Collaborative Filtering

How to measure similarity between users rating vectors?

Compare users A and C ratings

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1	**?**	
B	5	5	4				
C				2	4	5	
D		3					3

Collaborative Filtering

How to measure similarity between users rating vectors?

Compare users A and B ratings

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4	?	?	5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Collaborative Filtering

How to measure similarity between users rating vectors?

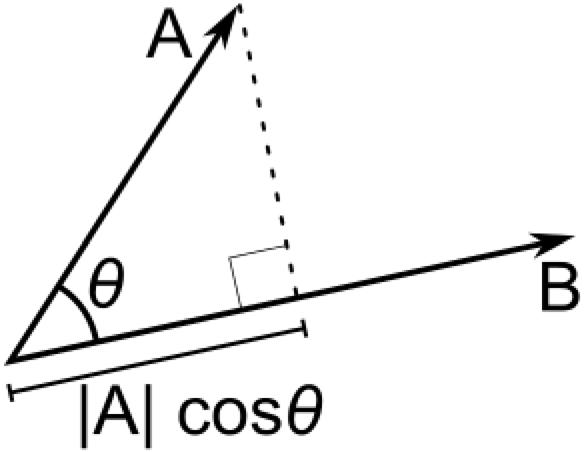
Intuitively we want: $\mathrm{sim}(\boldsymbol{r}_A, \boldsymbol{r}_B) > \mathrm{sim}(\boldsymbol{r}_A, \boldsymbol{r}_C)$

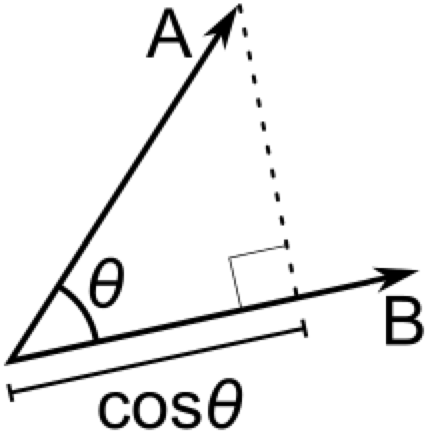
	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4	**?**	**?**	5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Collaborative Filtering

Cosine Similarity:

$$\cos(\boldsymbol{r_A}, \boldsymbol{r_B}) = \frac{\boldsymbol{r_A} \cdot \boldsymbol{r_B}}{||\boldsymbol{r_A}|| \cdot ||\boldsymbol{r_B}||}$$





Collaborative Filtering

Cosine Similarity:
(we can treat blanks as 0 values)

$$\cos(\boldsymbol{r}_A, \boldsymbol{r}_B) = \frac{\boldsymbol{r}_A \cdot \boldsymbol{r}_B}{||\boldsymbol{r}_A|| \cdot ||\boldsymbol{r}_B||}$$

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4	0	0	5	1	0	0
B	5	5	4	0	0	0	0
C	0	0	0	2	4	5	0
D	0	3	0	0	0	0	3

Collaborative Filtering

$$\text{sim}(\boldsymbol{r}_A, \boldsymbol{r}_B) = \frac{4\times5}{6{,}48\times8{,}12} = 0{,}380$$

$$\text{sim}(\boldsymbol{r}_A, \boldsymbol{r}_C) = \frac{5\times2+1\times4}{6{,}48\times6{,}71} = 0{,}322$$

we wanted $\text{sim}(\boldsymbol{r}_A, \boldsymbol{r}_B) > \text{sim}(\boldsymbol{r}_A, \boldsymbol{r}_C)$

This measure tells us that *A* is slightly closer to *B* than *C.*

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4	0	0	5	1	0	0
B	5	5	4	0	0	0	0
C	0	0	0	2	4	5	0
D	0	3	0	0	0	0	3

Collaborative Filtering

Cosine Similarity on normalized ratings:

Subtract the (row) mean from each value in the utility matrix.

If we normalize ratings, we turn low ratings into negative numbers and high ratings into positive numbers.

Without normalization

sim($\boldsymbol{r}_A$, $\boldsymbol{r}_B$) = $\frac{4\times5}{6{,}48\times8{,}12} = 0{,}380$

sim($\boldsymbol{r}_A$, $\boldsymbol{r}_C$) = $\frac{5\times2+1\times4}{6{,}48\times6{,}71} = 0{,}322$

With normalization

sim($\boldsymbol{r}_A$, $\boldsymbol{r}_B$) = $0{,}092$

sim($\boldsymbol{r}_A$, $\boldsymbol{r}_C$) = $-0{,}599$

Collaborative Filtering

Cosine Similarity on normalized ratings:

This measure captures the intuition better, since *A* and *C* disagree on the two movies they rated in common, while *A* and *B* give similar scores to the one movie they rated in common.

Without norm.
sim($\boldsymbol{r}_A$, $\boldsymbol{r}_B$) = 0,380
sim($\boldsymbol{r}_A$, $\boldsymbol{r}_C$) = 0,322

With normalization
sim($\boldsymbol{r}_A$, $\boldsymbol{r}_B$) = 0,092
sim($\boldsymbol{r}_A$, $\boldsymbol{r}_C$) = −0,599

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Twilight	Star Wars 1	Star Wars 2	Star Wars 3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Collaborative Filtering

Rating Prediction

Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
Let $\boldsymbol{N}$ be the set of $\boldsymbol{k}$ users most similar to $\boldsymbol{x}$ who have rated item $\boldsymbol{i}$

Prediction for item $\boldsymbol{s}$ of *user* $\boldsymbol{x}$:

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi} \qquad or \qquad r_{xi} = \frac{\sum_{y \in N} sim_{xy} \cdot r_{yi}}{\sum_{y \in N} sim_{xy}}$$

Collaborative Filtering

So far: user-user collaborative filtering
Dual approach: Item-item collaborative filtering

- For item ***i***, find other similar items
- Estimate rating for item ***i*** based on ratings for similar items
- Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij} = similarity of items *i* and *j*
r_{xj} = rating of user *u* on item *j*
N(i;x) = set items rated by *x* similar to *i*

Collaborative Filtering

In practice, it has been observed that item-item often works better than user-user

Why? Items are simpler, users have multiple tastes. Intuitively. Items tend to be classifiable in simple terms (e.g., music genre).

UNIVERSITÀ degli STUDI di CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA

Latent Factor Methods

IMAGE PROCESSING LABORATORY
http//iplab.dmi.unict.it

Latent Factor Methods

Features are very important in Machine Learning, the features you choose will have a big effect on the performance of your learning algorithm.

There are algorithms that can try to automatically learn a good set of features for you.

Rather than trying to hand design features, there are cases where you might be able to have an algorithm which just learns what feature to use.

Latent Factor Methods

- Each item is represented by a feature vector ***x***
- Each user is represented by a vector of parameters $\boldsymbol{\theta}$
- Predict the rating of the user *j* to the item *i* as the product

$$(\theta^j)^T x^i$$

That is, we treat the prediction as a linear regression problem.

Two cases:

1. Given x^1, x^2,…., x^m we can learn θ^j for each user *j* (i.e., content based)
2. Given θ^1, θ^2,…., θ^n we can learn x^i for each item *i* (i.e., collaborative filtering)

Latent Factor Methods

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

Latent Factor Methods

Each item is represented by a feature vector ***x***

	Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
x^1	Love at last	5	5	0	0
x^2	Romance forever	5	?	?	0
x^3	Cute puppies of love	?	4	0	?
x^4	Nonstop car chases	0	0	5	4
x^5	Swords vs. karate	0	0	5	?

Latent Factor Methods

Each user is represented by a vector of weights $\boldsymbol{\theta}$

Each item is represented by a feature vector $\boldsymbol{x}$

	Movie	Alice (1) θ^1	Bob (2) θ^2	Carol (3) θ^3	Dave (4) θ^4
x^1	Love at last	5	5	0	0
x^2	Romance forever	5	?	?	0
x^3	Cute puppies of love	?	4	0	?
x^4	Nonstop car chases	0	0	5	4
x^5	Swords vs. karate	0	0	5	?

Latent Factor Methods

Each user is represented by a vector of weights $\boldsymbol{\theta}$

Each item is represented by a feature vector $\boldsymbol{x}$

	Movie	Alice (1) θ^1	Bob (2) θ^2	Carol (3) θ^3	Dave (4) θ^4
x^1	Love at last	5	5	0	0
x^2	Romance forever	5	?	$\boxed{?}$	0
x^3	Cute puppies of love	?	4	0	?
x^4	Nonstop car chases	0	0	5	4
x^5	Swords vs. karate	0	0	5	?

For user 3 (Carol) and movie 2 (Romance forever), the predicted rating is: $(\theta^3)^T x^2$

Latent Factor Methods

Linear Regression

to learn the parameter θ^j (feature for user *j*):

$$\min_{\theta^{(j)}} \frac{1}{2} \underline{\sum_{i:r(i,j)=1}} \left((\theta^{(j)})^T x^{(i)} - \underline{y^{(i,j)}} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

to learn θ^1, θ^2,…., θ^n:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$r(i,j) = 1$ if user *j* rated movie i

Latent Factor Methods

How to chose the features x^1, x^2,…., x^m ?

Let's change a bit the problem:

- Say that we know the user vector parameters θ^1, θ^2,…., θ^n
- We can infer the values of x^i for each movie

$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2$$

Latent Factor Methods

- Given x^1, x^2,...., x^m

  we can estimate θ^1, θ^2,...., θ^n

- Given θ^1, θ^2,...., θ^n

  we can estimate x^1, x^2,...., x^m

Latent Factor Methods

- Given $x^1, x^2, \ldots, x^m$

  we can estimate $\theta^1, \theta^2, \ldots, \theta^n$

- Given $\theta^1, \theta^2, \ldots, \theta^n$

  we can estimate $x^1, x^2, \ldots, x^m$

Unified cost function:

$$J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) =$$

$$= \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Latent Factor Methods

Procedure:

1. Initialize $x^1, x^2, \ldots, x^m$ and $\theta^1, \theta^2, \ldots, \theta^n$ to small random values

2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$

Latent Factor Methods

Procedure:

1. Initialize $x^1, x^2,\ldots, x^m$ and $\theta^1, \theta^2,\ldots, \theta^n$ to small random values

2. Minimize $J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)})$

Guess $\boldsymbol{\theta}$

Latent Factor Methods

Procedure:

1. Initialize $x^1, x^2, \ldots, x^m$ and $\theta^1, \theta^2, \ldots, \theta^n$ to small random values

2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$

Guess $\boldsymbol{\theta}$ → infer $\boldsymbol{x}$

Latent Factor Methods

Procedure:

1. Initialize $x^1, x^2, \ldots, x^m$ and $\theta^1, \theta^2, \ldots, \theta^n$ to small random values

2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$

Guess $\boldsymbol{\theta}$ → infer $\boldsymbol{x}$ → infer (better) $\boldsymbol{\theta}$

Latent Factor Methods

Procedure:

1. Initialize $x^1, x^2,\ldots., x^m$ and $\theta^1, \theta^2,\ldots., \theta^n$ to small random values

2. Minimize $J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)})$

Guess $\boldsymbol{\theta}$ → infer $\boldsymbol{x}$ → infer (better) $\boldsymbol{\theta}$ → infer (better) $\boldsymbol{x}$ → etc…

Latent Factor Methods

Procedure:

1. Initialize $x^1, x^2, \ldots, x^m$ and $\theta^1, \theta^2, \ldots, \theta^n$ to small random values

2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$

3. for a user with parameters θ and movie with (learned) features x, predict a rating of

$$(\theta)^T x$$

References

- J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets
  http://www.mmds.org

- *Machine Learning* - Stanford Course by Prof. Andrew Ng (on Coursera)