MU5IN075
Network Analysis and Mining
11. Recommendation Algorithms

Esteban Bautista-Ruiz, Lionel Tabourier

LIP6 – CNRS and Sorbonne Université

first_name.last_name@lip6.fr

December 14, 2021

Outline

An information filtering problem

Main source: Mining Massive Datasets - *J.Leskovec, A.Rajaraman, J.D.Ullman*

From "scarcity" to "abundance"

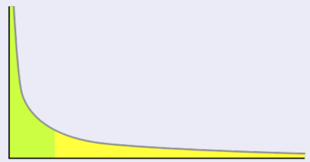
- Physical retailer (Leclerc, Lidl, Walmart, . . .): limited shelf space ⇒ limited number of products
- Web era (Amazon, Google news, Netflix, . . .): ⇒ commodities at dissemination cost $\simeq 0$

⇒ paradigm shift

An information filtering problem

Reaching the "long-tail"

Ordering items by preference:



- limited shelf space ⇒ cut-off in the distribution
- unlimited shelf space ⇒ access to items in the long-tail

example: "Touching the Void" (La mort suspendue) phenomenon
see https://www.wired.com/2004/10/tail/

An information filtering problem

Also new challenges and questions to solve:
how to guide users browsing large catalogs?

Functions

- primary: information filtering, bring more relevant information for less research time
- secondary: bring serendipity to users "happy discoveries"

Difference with search engines

- searching (with a query) is active
- being recommended is passive

but the frontier can be thin

Introduction
Implementing recommendation systems
Conclusion and perspectives
The recommendation problem
Recommendation approaches

Some historical elements

- First appearance of the term associated with Gerry Salton (80s) Salton and McGill - *Introduction to modern Information Retrieval System. 1980*
- First implementations (in today's sense) in the 90s:
  - spam filter Tapestry (@Xerox, Palo Alto): uses annotations from other users to evaluate relevance Goldberg et al. - *1992*
  - document search by GroupLens (@University Minnesota): uses comments for news selection on UseNet Resnick et al. - *1994*
  - musical album search Ringo (@MIT): thresholding based on social similarity Shardanand and Maes - *1995*
- More recent interesting examples:
  - Amazon shopping collaborative filtering system
  - Pandora vs Last.fm (webradios in the 2000's)

Introduction
Implementing recommendation systems
Conclusion and perspectives
The recommendation problem
Recommendation approaches

Some historical elements

- First appearance of the term associated with Gerry Salton (80s) Salton and McGill - Introduction to modern Information Retrieval System. 1980
- First implementations (in today's sense) in the 90s:
  - spam filter Tapestry (@Xerox, Palo Alto): uses annotations from other users to evaluate relevance Goldberg et al. - 1992
  - document search by GroupLens (@University Minnesota): uses comments for news selection on UseNet Resnick et al. - 1994
  - musical album search Ringo (@MIT): thresholding based on social similarity Shardanand and Maes - 1995
- More recent interesting examples:
  - Amazon shopping collaborative filtering system
  - Pandora vs Last.fm (webradios in the 2000's)

4/28

Introduction
Implementing recommendation systems
Conclusion and perspectives
The recommendation problem
Recommendation approaches

Some historical elements

- First appearance of the term associated with Gerry Salton (80s) Salton and McGill - Introduction to modern Information Retrieval System. 1980
- First implementations (in today's sense) in the 90s:
  - spam filter Tapestry (@Xerox, Palo Alto): uses annotations from other users to evaluate relevance Goldberg et al. - 1992
  - document search by GroupLens (@University Minnesota): uses comments for news selection on UseNet Resnick et al. - 1994
  - musical album search Ringo (@MIT): thresholding based on social similarity Shardanand and Maes - 1995
- More recent interesting examples:
  - Amazon shopping collaborative filtering system
  - Pandora vs Last.fm (webradios in the 2000's)

4/28

Recommendation systems in machine learning

Recommendation systems are now deeply related to the machine learning field

Reformulating the recommendation task

- either to predict a score (eg., user rating)
- or to predict if a user clicks or buys, ...

From a machine learning perspective:

- a **regression** task (predicting a score)
- a **classification** task (predicting if an interaction happens)

Both are **supervised learning** tasks

Recommendation approaches

From basic

- *top-5 more popular products*, ...
  → typically on website frontpages
- but does not help reaching the long-tail, no personalization

To personalization: useful information

1. Knowledge of the user's tastes
2. User relatively to other users
   Very niche tastes more informative than very usual tastes
3. Knowledge of the items to recommend
   ex of a movie: director, actors, genre, year ...
4. Item relatively to other items
   Very niche genre more informative than very popular genre

Recommendation approaches

From basic

- *top-5 more popular products*, ...
  → typically on website frontpages
- but does not help reaching the long-tail, no personalization

To personalization: useful information

1. Knowledge of the user's tastes
2. User relatively to other users
   Very niche tastes more informative than very usual tastes
3. Knowledge of the items to recommend
   ex of a movie: director, actors, genre, year ...
4. Item relatively to other items
   Very niche genre more informative than very popular genre

Two main recommendation families

Content-based filtering

- identify the features in an item that a user likes
- uses factors 1, 3 and 4 but not 2
- *example: Pandora and the Music Genome Project*

Collaborative filtering

- identify users who have similar tastes
- uses factors 1, 2 but not 3 and 4
- a lot of them...
  examples: Tapestry, Ringo, Amazon, Last.fm

Two main recommendation families

Content-based filtering

- identify the features in an item that a user likes
- uses factors 1, 3 and 4 but not 2
- *example: Pandora and the Music Genome Project*

Collaborative filtering

- identify users who have similar tastes
- uses factors 1, 2 but not 3 and 4
- a lot of them...
  examples: Tapestry, Ringo, Amazon, Last.fm

Outline

A baseline recommendation

Illustration on a rating problem, we want to predict $r(u, i)$

A standard baseline score:

$$r_B(u, i) = \overline{r} + (\overline{r(u)} - \overline{r}) + (\overline{r(i)} - \overline{r})$$

where

- $\overline{r}$ is the average rating of the dataset
- $\overline{r(u)}$ is the average rating of user u in the dataset
- $\overline{r(i)}$ is the average rating of item i in the dataset

minimal level of personalization, how can we improve that?

Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

A baseline recommendation

Illustration on a rating problem, we want to predict $r(u, i)$

A standard baseline score:

$$r_B(u, i) = \overline{r} + (\overline{r(u)} - \overline{r}) + (\overline{r(i)} - \overline{r})$$

where

- $\overline{r}$ is the average rating of the dataset
- $\overline{r(u)}$ is the average rating of user u in the dataset
- $\overline{r(i)}$ is the average rating of item i in the dataset

minimal level of personalization, how can we improve that?

9/28

A baseline recommendation

Illustration on a rating problem, we want to predict $r(u, i)$

A standard baseline score:

$$r_B(u,i) = \overline{r} + (\overline{r(u)} - \overline{r}) + (\overline{r(i)} - \overline{r})$$

where

- $\overline{r}$ is the average rating of the dataset
- $\overline{r(u)}$ is the average rating of user u in the dataset
- $\overline{r(i)}$ is the average rating of item i in the dataset

minimal level of personalization, how can we improve that?

Vectorial approach to content-based filtering (1)

First step: item as a vector of features

Listing relevant features → associate score to each item

Examples:

- movie → genre scores (given by expert)
  Back to the Future: 2 Sci-Fi, 3 Action, 2 Comedy, 0 Romance, 0 Drama
- document → set of words with a score of importance (tf-idf)

Limitations

- Assumes an expertise of the field
- Loss of information
  ex: set of words → no idea of context, of order

Vectorial approach to content-based filtering (1)

First step: item as a vector of features

Listing relevant features → associate score to each item

Examples:

- movie → genre scores (given by expert)
  Back to the Future: 2 Sci-Fi, 3 Action, 2 Comedy, 0 Romance, 0 Drama
- document → set of words with a score of importance (tf-idf)

Limitations

- Assumes an expertise of the field
- Loss of information
  ex: set of words → no idea of context, of order

Vectorial approach to content-based filtering (2)

Second step: user profiling

- Using formerly selected items

  item 1: [0 0 1 0 2 0]
  item 2: [1 3 0 0 0 0]
  item 3: [1 1 1 0 0 0]

  $\longrightarrow$ profile: $[\frac{2}{3} \quad 2 \quad \frac{2}{3} \quad 0 \quad \frac{2}{3} \quad 0]$

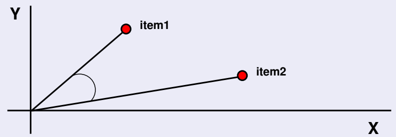
- Option: give weights according to feedbacks
  ex: item1 disliked (weight = -1); i2 liked (+1); i3 neutral (0)
  $\longrightarrow$ profile: $[\frac{1}{2} \quad \frac{3}{2} \quad -\frac{1}{2} \quad 0 \quad -1 \quad 0]$

Vectorial approach to content-based filtering (3)

one example of similarity measurement
but many others available . . .

Cosine similarity

$$cos(\vec{u},\vec{i}) = \frac{\vec{u}\cdot\vec{i}}{||u||\cdot||i||} = \frac{\sum_k u_k.i_k}{\sqrt{\sum_k u_k^2}\sqrt{\sum_k i_k^2}}$$



Recommend items most similar to the user

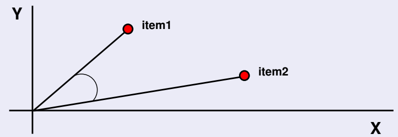
Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

Vectorial approach to content-based filtering (3)

one example of similarity measurement
but many others available . . .

Cosine similarity

$$cos(\vec{u},\vec{i}) = \frac{\vec{u}\cdot\vec{i}}{||u||\cdot||i||} = \frac{\sum_k u_k.i_k}{\sqrt{\sum_k u_k^2}\sqrt{\sum_k i_k^2}}$$



Recommend items most similar to the user

Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

Further analysis of content-based filtering

Advantages

- Personalized
- Explanatory: we know why an item is recommended
- Independent of other users tastes
  $\rightarrow$ allows to recommend new or unpopular items

Drawbacks

- Expert knowledge needed $\rightarrow$ manual feature definition and processing (*ex: how to define a film plot?*)
- Do not use feedback from other users
- Overspecialization: tends to recommend specific items similar to those already selected by a user

Content-based filtering tends to disappear

Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

Further analysis of content-based filtering

Advantages

- Personalized
- Explanatory: we know why an item is recommended
- Independent of other users tastes
  $\rightarrow$ allows to recommend new or unpopular items

Drawbacks

- Expert knowledge needed $\rightarrow$ manual feature definition and processing (*ex: how to define a film plot?*)
- Do not use feedback from other users
- Overspecialization: tends to recommend specific items similar to those already selected by a user

Content-based filtering tends to disappear

Further analysis of content-based filtering

Advantages

- Personalized
- Explanatory: we know why an item is recommended
- Independent of other users tastes
  → allows to recommend new or unpopular items

Drawbacks

- Expert knowledge needed → manual feature definition and processing (*ex: how to define a film plot?*)
- Do not use feedback from other users
- Overspecialization: tends to recommend specific items similar to those already selected by a user

Content-based filtering tends to disappear

Neighborhood approach to collaborative filtering (1)

Case-study: users give explicit feedback on items via rating

	A	B	C	D	E
Blade Runner	5	3	4	1	2
Back to the Future	4	3	5	1	-
Pride & Prejudice	1	3	2	4	-
Inception	-	4	2	5	4
Shrek	-	4	-	-	-

First step: find neighborhood

- neighborhood = group of users with similar tastes

How to find neighborhood?

Introduction | An approach to content-based filtering
Implementing recommendation systems | An approach to collaborative filtering
Conclusion and perspectives | Evaluate recommendations

Neighborhood approach to collaborative filtering (1)

Case-study: users give explicit feedback on items via rating

	A	B	C	D	E
Blade Runner	5	3	4	1	2
Back to the Future	4	3	5	1	-
Pride & Prejudice	1	3	2	4	-
Inception	-	4	2	5	4
Shrek	-	4	-	-	-

First step: find neighborhood

- neighborhood = group of users with similar tastes

How to find neighborhood?

14/28

Neighborhood approach to collaborative filtering (1)

Cosine similarity

$$cos(\vec{A}, \vec{B}) = \frac{\vec{A}.\vec{B}}{||A||\cdot||B||} = \frac{\sum_i A_i.B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}$$

Here, $\vec{X}$ is the vector of ratings of user X

But two major limitations in this context:

- unrated items seen as 0
- humans have different types of rating behaviors:

Neighborhood approach to collaborative filtering (1)

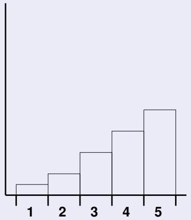
Cosine similarity

$$cos(\vec{A},\vec{B}) = \frac{\vec{A}.\vec{B}}{||A||\cdot||B||} = \frac{\sum_i A_i.B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}$$

Here, $\vec{X}$ is the vector of ratings of user X

But two major limitations in this context:

- unrated items seen as 0
- humans have different types of rating behaviors:



Neighborhood approach to collaborative filtering (1)

Centered cosine similarity

- compute *rating - average rating* (centering)

	A	B	C	D	E
Blade Runner	5	3	4	1	2
Back to the Future	4	3	5	1	-
Pride & Prejudice	1	3	2	4	-
Inception	-	4	2	5	4
Shrek	-	4	-	-	-
average	10/3	17/5	13/4	11/4	6/2

- missing rating → average rating
- $sim(A,B) = cos(\vec{A'},\vec{B'}) = \frac{\sum_i (A_i - \overline{A}).(B_i - \overline{B})}{\sqrt{\sum_i (A_i - \overline{A})^2}\sqrt{\sum_i (B_i - \overline{B})^2}}$

  = Pearson coefficient (A,B)

Neighborhood approach to collaborative filtering (1)

Centered cosine similarity

- compute *rating - average rating* (centering)

	A	B	C	D	E
Blade Runner	+5/3	-2/5	+3/4	-7/4	-2/2
Back to the Future	+2/3	-2/5	+7/4	-7/4	-
Pride & Prejudice	-7/3	-2/5	-5/4	+5/4	-
Inception	-	+3/5	-5/4	+9/4	+2/2
Shrek	-	+3/5	-	-	-
cent. average	0	0	0	0	0

- missing rating → average rating
- $sim(A,B) = cos(\vec{A'},\vec{B'}) = \frac{\sum_i (A_i - \overline{A}).(B_i - \overline{B})}{\sqrt{\sum_i (A_i - \overline{A})^2}\sqrt{\sum_i (B_i - \overline{B})^2}}$

  = Pearson coefficient (A,B)

Neighborhood approach to collaborative filtering (1)

Centered cosine similarity

- compute *rating - average rating* (centering)

	A	B	C	D	E
Blade Runner	+5/3	-2/5	+3/4	-7/4	-2/2
Back to the Future	+2/3	-2/5	+7/4	-7/4	-
Pride & Prejudice	-7/3	-2/5	-5/4	+5/4	-
Inception	-	+3/5	-5/4	+9/4	+2/2
Shrek	-	+3/5	-	-	-
cent. average	0	0	0	0	0

- missing rating → average rating
- $sim(A,B) = cos(\vec{A'},\vec{B'}) = \frac{\sum_i (A_i - \overline{A}).(B_i - \overline{B})}{\sqrt{\sum_i (A_i - \overline{A})^2}\sqrt{\sum_i (B_i - \overline{B})^2}}$

  = Pearson coefficient (A,B)

Neighborhood approach to collaborative filtering (2)

Score prediction

Predict score between user u and item i:

- select N_u: k users most similar to u who have selected i
- prediction $r^*(u,i)$ is the average score over N_u:

$$r^*(u,i) = \frac{1}{k}\sum_{x \in N_u} r(x,i)$$

- (more elaborate) weighted average score over N_u:

$$r^*(u,i) = \frac{\sum_{x \in N_u} sim(u,x).r(x,i)}{\sum_{x \in N_u} sim(u,x)}$$

User-based collaborative filtering

Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

Neighborhood approach to collaborative filtering (2)

Score prediction

Predict score between user u and item i:

- select N_u: k users most similar to u who have selected i
- prediction $r^*(u,i)$ is the average score over N_u:

$$r^*(u,i) = \frac{1}{k}\sum_{x \in N_u} r(x,i)$$

- (more elaborate) weighted average score over N_u:

$$r^*(u,i) = \frac{\sum_{x \in N_u} sim(u,x).r(x,i)}{\sum_{x \in N_u} sim(u,x)}$$

User-based collaborative filtering

16/28

Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

Neighborhood approach to collaborative filtering (2)

Score prediction

Predict score between user u and item i:

- select N_u: k users most similar to u who have selected i
- prediction $r^*(u,i)$ is the average score over N_u:

$$r^*(u,i) = \frac{1}{k}\sum_{x \in N_u} r(x,i)$$

- (more elaborate) weighted average score over N_u:

$$r^*(u,i) = \frac{\sum_{x \in N_u} sim(u,x).r(x,i)}{\sum_{x \in N_u} sim(u,x)}$$

User-based collaborative filtering

16/28

Introduction
Implementing recommendation systems
Conclusion and perspectives
An approach to content-based filtering
An approach to collaborative filtering
Evaluate recommendations

Neighborhood approach to collaborative filtering: Item-based CF (1)

Similar principle, but based on item similarity

Centered cosine similarity

- User-item rating matrix

	A	B	C	D	E	avg
B. R.	5	3	4	1	2	15/5
B. to the F.	4	3	5	1	-	13/4
P. & P.	1	3	2	4	-	10/4
Inc.	-	4	2	5	4	15/4
Shrek	-	4	-	-	-	4/1

- similarity score

$$sim(I,J) = \frac{\sum_x (I_x - \bar{I}).(J_x - \bar{J})}{\sqrt{\sum_x (I_x - \bar{I})^2}\sqrt{\sum_x (J_x - \bar{J})^2}}$$

Neighborhood approach to collaborative filtering: Item-based CF (1)

Similar principle, but based on item similarity

Centered cosine similarity

- User-item rating matrix

	A	B	C	D	E	c. av.
B. R.	+10/5	0	+5/5	-10/5	-5/5	0
B. to the F.	+3/4	-1/4	+7/4	-9/4	-	0
P. & P.	-6/4	+2/4	-2/4	+6/4	-	0
Inc.	-	+1/4	-7/4	+5/4	+5/4	0
Shrek	-	0	-	-	-	0

- similarity score

$$sim(I,J) = \frac{\sum_x (I_x - \bar{I}).(J_x - \bar{J})}{\sqrt{\sum_x (I_x - \bar{I})^2}\sqrt{\sum_x (J_x - \bar{J})^2}}$$

Neighborhood approach to collaborative filtering: Item-based CF (2)

Score prediction

Predict score between u and i:

- select N_i: k items most similar to i which have been selected by u
- average score:

$$r^*(u,i) = \frac{1}{k}\sum_{j \in N_i} r(u,j)$$

- or weighted average score:

$$r^*(u,i) = \frac{\sum_{j \in N_i} sim(i,j).r(u,j)}{\sum_{j \in N_i} sim(i,j)}$$

Item-based CF is more efficient than User-based CF
in many practical applications

Linden et al. - *Amazon.com recommendations, 2003.*

Neighborhood approach to collaborative filtering: Item-based CF (2)

Score prediction

Predict score between u and i:

- select N_i: k items most similar to i which have been selected by u
- average score:

$$r^*(u,i) = \frac{1}{k}\sum_{j \in N_i} r(u,j)$$

- or weighted average score:

$$r^*(u,i) = \frac{\sum_{j \in N_i} sim(i,j).r(u,j)}{\sum_{j \in N_i} sim(i,j)}$$

Item-based CF is more efficient than User-based CF
in many practical applications

Linden et al. - *Amazon.com recommendations, 2003.*

Further analysis of collaborative filtering

Advantages

- Personalized
- No expert knowledge needed, works on any item
  ⇒ very popular because feature selection is hard
- Use feedback from other users

Drawbacks

- Not explanatory (no other information than users tastes)
- Does not allow to recommend new or unpopular items
- Need a lot of users (the *critical mass*)
- Tendency to popularity bias (*the Harry Potter effect*): popular items are often in a neighborhood

Further analysis of collaborative filtering

Advantages

- Personalized
- No expert knowledge needed, works on any item
  ⇒ very popular because feature selection is hard
- Use feedback from other users

Drawbacks

- Not explanatory (no other information than users tastes)
- Does not allow to recommend new or unpopular items
- Need a lot of users (the *critical mass*)
- Tendency to popularity bias (*the Harry Potter effect*): popular items are often in a neighborhood

How to set the method parameters?

Prediction methods have many parameters:

- Content-based: what are the significant features? ...
- Collaborative: neighborhood size? similarity measure? ...

How to set these parameters in the "best" way?

Need an evaluation methodology:

- **score** to measure efficiency
- **measure** efficiency on known data
- **compare scores** → best parameters for predictions

How to set the method parameters?

Prediction methods have many parameters:

- Content-based: what are the significant features? ...
- Collaborative: neighborhood size? similarity measure? ...

How to set these parameters in the "best" way?

Need an evaluation methodology:

- **score** to measure efficiency
- **measure** efficiency on known data
- **compare scores** → best parameters for predictions

Evaluation scores

Depending on the problem (classification, regression)
a lot of evaluation scores are available

Regression problems (our example)

Comparing the actual rating (r_i) to the predicted rating (r_i^*):

- Root Mean Square Error
  racine de l'erreur quadratique moyenne

$$RMSE = \sqrt{\frac{\sum_{i=1}^{K}(r_i^* - r_i)^2}{K}}$$

Rk: Relevant to give the same importance to all scores?
→ e.g. just focus on top-5,...

Evaluation scores

Depending on the problem (classification, regression)
a lot of evaluation scores are available

Regression problems (our example)

Comparing the actual rating (r_i) to the predicted rating (r_i^*):

- Root Mean Square Error
  racine de l'erreur quadratique moyenne

$$RMSE = \sqrt{\frac{\sum_{i=1}^{K}(r_i^* - r_i)^2}{K}}$$

Rk: Relevant to give the same importance to all scores?
$\rightarrow$ e.g. just focus on top-5,...

Introduction | Implementing recommendation systems | Conclusion and perspectives
An approach to content-based filtering | An approach to collaborative filtering | Evaluate recommendations

Evaluation scores

Depending on the problem (classification, regression)
a lot of evaluation scores are available

Regression problems (our example)

Comparing the actual rating (r_i) to the predicted rating (r_i^*):

- Root Mean Square Error
  racine de l'erreur quadratique moyenne

$$RMSE = \sqrt{\frac{\sum_{i=1}^{K}(r_i^* - r_i)^2}{K}}$$

Rk: Relevant to give the same importance to all scores?
$\rightarrow$ e.g. just focus on top-5,...

Split-Predict

A framework for learning problems:

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
i_1	2	4	-	-	-	4	1	3	2	-
i_2	2	5	5	-	-	-	4	5	2	1
i_3	-	-	2	-	-	-	3	3	-	1
i_4	-	5	5	5	-	5	-	2	-	-
i_5	-	-	1	2	4	4	-	-	1	-
i_6	-	-	-	1	4	5	-	-	-	-
i_7	-	5	4	5	-	-	4	4	3	5
i_8	-	-	1	2	-	-	-	-	1	5

Splitting matrix: learning set vs test set (blue cells)

Introduction | Implementing recommendation systems | Conclusion and perspectives
An approach to content-based filtering | An approach to collaborative filtering | Evaluate recommendations

Split-Predict

A framework for learning problems:

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
i_1	**2**	4	-	-	-	4	1	**3**	2	-
i_2	2	5	**5**	-	-	-	4	**5**	2	1
i_3	-	-	2	-	-	-	3	3	-	1
i_4	-	5	**5**	5	-	5	-	2	-	-
i_5	-	-	1	2	4	**4**	-	-	**1**	-
i_6	-	-	-	**1**	4	5	-	-	-	-
i_7	-	5	4	5	-	-	**4**	4	**3**	**5**
i_8	-	-	1	2	-	-	-	-	**1**	5

Splitting matrix: learning set vs test set (blue cells)

Split-Predict

A framework for learning problems:

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
i_1	1	4	-	-	-	4	1	3	2	-
i_2	2	5	4	-	-	-	4	5	2	1
i_3	-	-	2	-	-	-	3	3	-	1
i_4	-	5	3	5	-	5	-	2	-	-
i_5	-	-	1	2	4	4	-	-	1	-
i_6	-	-	-	2	4	5	-	-	-	-
i_7	-	5	4	5	-	-	3	4	3	4
i_8	-	-	1	2	-	-	-	-	2	5

Prediction: on the test set (orange cells) using the learning set

Collaborative Filtering in practice

A standard implementation of a user-based CF

- Split: select x% of ratings randomly ($\equiv$ test set), remaining $(100 - x\%)$ ratings $\equiv$ learning set
- Predict: for each rating $r(u, i)$ of the test set:
  - find users in the learning set who have rated the item i
  - compute their similarities to user u (in the learning set)
  - select the k most similar
  - compute predicted score and compare to the actual score

For more accurate parameters:
repeat the split X times and compute average predictions

Collaborative Filtering in practice

A standard implementation of a user-based CF

- Split: select x% of ratings randomly ($\equiv$ test set), remaining $(100 - x\%)$ ratings $\equiv$ learning set
- Predict: for each rating $r(u, i)$ of the test set:
  - find users in the learning set who have rated the item i
  - compute their similarities to user u (in the learning set)
  - select the k most similar
  - compute predicted score and compare to the actual score

For more accurate parameters:
repeat the split X times and compute average predictions

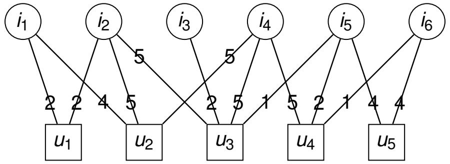
Outline

1. Introduction
   - The recommendation problem
   - Recommendation approaches
2. Implementing recommendation systems
   - An approach to content-based filtering
   - An approach to collaborative filtering
   - Evaluate recommendations
3. Conclusion and perspectives

Recommendation seen as a graph

Recommendation based on user-item matrix
and a matrix can be seen as a bipartite graph. . .

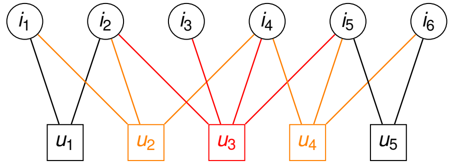
	u_1	u_2	u_3	u_4	u_5
i_1	2	4	-	-	-
i_2	2	5	5	-	-
i_3	-	-	2	-	-
i_4	-	5	5	5	-
i_5	-	-	1	2	4
i_6	-	-	-	1	4



Recommendation seen as a graph

Notion of neighborhood:
community of users around a user

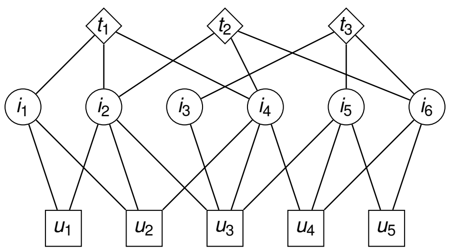
	u_1	u_2	u_3	u_4	u_5
i_1	2	4	-	-	-
i_2	2	5	5	-	-
i_3	-	-	2	-	-
i_4	-	5	5	5	-
i_5	-	-	1	2	4
i_6	-	-	-	1	4



Recommendation seen as a graph

How to represent content information?
ex: t_1 is SF, t2 is comedy, t_3 is drama

	u_1	u_2	u_3	u_4	u_5
i_1	2	4	-	-	-
i_2	2	5	5	-	-
i_3	-	-	2	-	-
i_4	-	5	5	5	-
i_5	-	-	1	2	4
i_6	-	-	-	1	4

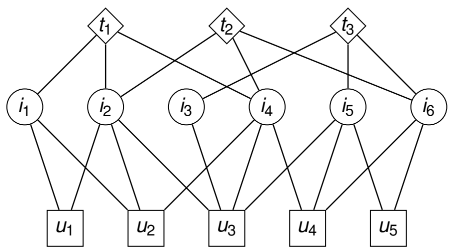


Heterogeneous Information Networks

Recommendation seen as a graph

How to represent content information?
ex: t_1 is SF, t2 is comedy, t_3 is drama

	u_1	u_2	u_3	u_4	u_5
i_1	2	4	-	-	-
i_2	2	5	5	-	-
i_3	-	-	2	-	-
i_4	-	5	5	5	-
i_5	-	-	1	2	4
i_6	-	-	-	1	4



Heterogeneous Information Networks

Reliance on data

Both content-based and collab. methods need data to work

When is recommendation hard?

- Both content-based and collaborative filtering: cold start
  - user is new to the platform (no user profile)
- Content-based filtering:
  - user never selected/rated an item of a given type
  - item profile not expert-evaluated yet
- Collaborative filtering:
  - item is new to the platform (first rater problem)
  - no critical mass of ratings for a user and for an item
    → need enough users rating a product (sparsity problem)

data available is a severely limiting factor
⇒ importance of the data market. . .

Introduction
Implementing recommendation systems
Conclusion and perspectives

Reliance on data

Both content-based and collab. methods need data to work

When is recommendation hard?

- Both content-based and collaborative filtering: cold start
  - user is new to the platform (no user profile)
- Content-based filtering:
  - user never selected/rated an item of a given type
  - item profile not expert-evaluated yet
- Collaborative filtering:
  - item is new to the platform (first rater problem)
  - no critical mass of ratings for a user and for an item
    → need enough users rating a product (sparsity problem)

data available is a severely limiting factor
⇒ importance of the data market. . .

26/28

Reliance on data (2)

How to temper these problems?

- User: look for any information available to get a profile
  ex: self-reported info, language, IP address, browser, OS, incoming website, . . .
- Item: look for content information about new items

Combine content to collaborative information
→ leads to hybrid recommender systems

Introduction
Implementing recommendation systems
Conclusion and perspectives

Reliance on data (2)

How to temper these problems?

- User: look for any information available to get a profile
  ex: self-reported info, language, IP address, browser, OS, incoming website, . . .
- Item: look for content information about new items

Combine content to collaborative information
→ leads to hybrid recommender systems

27/28

Do we answer the actual problem?

What is satisfaction?

Replace *satisfaction* with *a score prediction*

But is it legitimate?

- Depending on the context a recommendation does not have the same value
- User u rated high *Harry Potter 3* and *4* $\Rightarrow$ predicting high score for *Harry Potter 5* is easy but useless
  $\rightarrow$ there is no serendipity here

Do we answer the actual problem?

Humans are unreliable raters

- Depending on the moment, evaluations fluctuate
- Human have biases in their evaluation
  ex: aspiration bias
- In anyway few ratings even for very active users

From active to passive data collection

- More info from activity than from rating
  $\rightarrow$ measure clicks, watch time, time spent on the platform
- ...but harder to get dislike information

$\Rightarrow$ Combine active and passive feedback

Covington et al. - *Deep Neural Networks for YouTube Recommendations, 2016*

Do we answer the actual problem?

Humans are unreliable raters

- Depending on the moment, evaluations fluctuate
- Human have biases in their evaluation
  ex: aspiration bias
- In anyway few ratings even for very active users

From active to passive data collection

- More info from activity than from rating
  $\rightarrow$ measure clicks, watch time, time spent on the platform
- ...but harder to get dislike information

$\Rightarrow$ Combine active and passive feedback

Covington et al. - *Deep Neural Networks for YouTube Recommendations, 2016*