Tomáš Horváth

RECOMMENDER SYSTEMS

Tutorial at the conference

Znalosti 2012

October 14-16, 2012, Mikulov, Czech Republic

Institute of Computer Science, Faculty of Science
Pavol Jozef Šafárik University in Košice, Slovak Republic

Information Systems and Machine Learning Lab
University of Hildesheim, Germany
Contents

- Introduction
- Basic concepts
- Knowledge-based techniques
- Content-based techniques
- Collaborative-filtering
- Matrix factorization
- Issues worth to mention
- The MyMedialite library
- Summary
  ... and, if still alive,
- Questions & Answers
Introduction
What is a RS?
Why do we need RS?

A company wants to

- sell more (diverse) items
- increase users’ satisfaction and fidelity
- better understand users’ needs

A user would like to

- find some (or all, in case of critical domains such as medicine) good items with a relatively small effort
- express herself by providing ratings or opinions
- help others by contribute with information to the community
The Big Bang

- Contest begun on October 2, 2006
  - 100M ratings (1-5 stars) from 480K users on 18K movies
  - decrease RMSE of Cinematch (0.9525) at least with 10% ($\leq 0.8572$)
- Grand Prize $1,000,000$, Annual Progress Prizes $50,000$

<table>
<thead>
<tr>
<th>Rank</th>
<th>Team Name</th>
<th>Best Test Score</th>
<th>%6 Improvement</th>
<th>Best Submit Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BellKor's Pragmatic Chaos</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:18:28</td>
</tr>
<tr>
<td>2</td>
<td>The Ensemble</td>
<td>0.8567</td>
<td>10.06</td>
<td>2009-07-26 18:38:22</td>
</tr>
<tr>
<td>3</td>
<td>Grand Prize Team</td>
<td>0.8582</td>
<td>9.90</td>
<td>2009-07-10 21:24:40</td>
</tr>
<tr>
<td>4</td>
<td>Opera Solutions and Vandelay United</td>
<td>0.8588</td>
<td>9.84</td>
<td>2009-07-10 01:12:31</td>
</tr>
<tr>
<td>5</td>
<td>Vandelay Industries !</td>
<td>0.8591</td>
<td>9.81</td>
<td>2009-07-10 00:32:20</td>
</tr>
<tr>
<td>6</td>
<td>PragmaticTheory</td>
<td>0.8594</td>
<td>9.77</td>
<td>2009-06-24 12:06:56</td>
</tr>
<tr>
<td>7</td>
<td>BellKor in BigChaos</td>
<td>0.8601</td>
<td>9.70</td>
<td>2009-05-13 08:14:09</td>
</tr>
<tr>
<td>8</td>
<td>Dace</td>
<td>0.8612</td>
<td>9.59</td>
<td>2009-07-24 17:18:43</td>
</tr>
<tr>
<td>9</td>
<td>Feeds2</td>
<td>0.8622</td>
<td>9.48</td>
<td>2009-07-12 13:11:51</td>
</tr>
<tr>
<td>10</td>
<td>BigChaos</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-04-07 12:33:59</td>
</tr>
<tr>
<td>11</td>
<td>Opera Solutions</td>
<td>0.8623</td>
<td>9.47</td>
<td>2009-07-24 00:34:07</td>
</tr>
<tr>
<td>12</td>
<td>BellKor</td>
<td>0.8624</td>
<td>9.46</td>
<td>2009-07-26 17:19:11</td>
</tr>
</tbody>
</table>
### Netflix and Movielens data (1/2)

#### Netflix

<table>
<thead>
<tr>
<th>File: train.txt</th>
<th>Line 1 Col 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>157</td>
</tr>
<tr>
<td>1</td>
<td>173</td>
</tr>
<tr>
<td>1</td>
<td>175</td>
</tr>
<tr>
<td>1</td>
<td>191</td>
</tr>
<tr>
<td>1</td>
<td>197</td>
</tr>
<tr>
<td>1</td>
<td>241</td>
</tr>
<tr>
<td>1</td>
<td>295</td>
</tr>
<tr>
<td>1</td>
<td>299</td>
</tr>
<tr>
<td>1</td>
<td>329</td>
</tr>
<tr>
<td>1</td>
<td>361</td>
</tr>
<tr>
<td>1</td>
<td>445</td>
</tr>
<tr>
<td>1</td>
<td>457</td>
</tr>
<tr>
<td>1</td>
<td>468</td>
</tr>
<tr>
<td>1</td>
<td>494</td>
</tr>
<tr>
<td>1</td>
<td>528</td>
</tr>
<tr>
<td>1</td>
<td>564</td>
</tr>
<tr>
<td>1</td>
<td>580</td>
</tr>
<tr>
<td>1</td>
<td>705</td>
</tr>
<tr>
<td>1</td>
<td>706</td>
</tr>
<tr>
<td>1</td>
<td>723</td>
</tr>
<tr>
<td>1</td>
<td>788</td>
</tr>
<tr>
<td>1</td>
<td>872</td>
</tr>
<tr>
<td>1</td>
<td>886</td>
</tr>
</tbody>
</table>

#### Movielens (100K, 1M)

<table>
<thead>
<tr>
<th>File: ratings.dat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1193:5:978300760</td>
</tr>
<tr>
<td>1:661:3:978302109</td>
</tr>
<tr>
<td>1:914:3:978301968</td>
</tr>
<tr>
<td>1:3408:4:978300275</td>
</tr>
<tr>
<td>1:2355:5:978824291</td>
</tr>
<tr>
<td>1:1197:3:978302268</td>
</tr>
<tr>
<td>1:1287:5:978302039</td>
</tr>
<tr>
<td>1:2804:5:978300719</td>
</tr>
<tr>
<td>1:594:4:978302268</td>
</tr>
<tr>
<td>1:919:4:978301368</td>
</tr>
<tr>
<td>1:595:5:978824268</td>
</tr>
<tr>
<td>1:938:4:978301752</td>
</tr>
<tr>
<td>1:2390:4:978302281</td>
</tr>
<tr>
<td>1:2918:4:978302124</td>
</tr>
<tr>
<td>1:1035:5:978301753</td>
</tr>
<tr>
<td>1:2791:4:978302188</td>
</tr>
<tr>
<td>1:2687:3:978824268</td>
</tr>
<tr>
<td>1:2018:4:978301777</td>
</tr>
<tr>
<td>1:3105:5:978301713</td>
</tr>
<tr>
<td>1:2797:4:978302039</td>
</tr>
<tr>
<td>1:2321:3:978302205</td>
</tr>
<tr>
<td>1:720:3:978300760</td>
</tr>
<tr>
<td>1:1270:5:978300055</td>
</tr>
<tr>
<td>1:527:5:978824105</td>
</tr>
</tbody>
</table>
Netflix and Movielens data (2/2)

RecSys 2010 - 2012

WWW 2010 - 2012

UMAP 2010 - 2012

WSDM 2010 - 2012
Closely related fields

Information Retrieval

• unstructured data, various topics (IR) vs. repositories focused on a single topic (RS)
• relevant content for the query (IR) vs. relevant content for the user (RS)

Data mining & Machine Learning

• hardly measurable, subjective evaluation criteria (RS) besides some classic, objective evaluation measures (ML)

Human-Computer Interaction

• RS should convince the user to try the recommended items
• clear, transparent and trustworthy system logic
• provide details about recommended items and opportunity to refine recommendations
Related conferences

- ACM Recommender Systems (RecSys)
- User Modeling, Adaptation, and Personalization (UMAP)
- ACM Conference on Human Factors in Computing Systems (CHI)
- International World Wide Web Conference (WWW)
- ACM International Conference on Web Search and Data mining (WSDM)
- International Conference on Research and Development in Information Retrieval (SIGIR)
- ACM Conference on Information and Knowledge Management (CIKM)
- ...
Textbook (2011)

RECOMMENDER SYSTEMS HANDBOOK

FRANCESCO RICCI
LIOR ROKACH
BRACHA SHAPIRA
PAUL B. KANTOR EDITORS

Springer
Basic concepts
Users

- set of users $\mathcal{U}$
- user attributes $\mathcal{A}^{user} \subset \mathbb{R}^k$
  - age, income, marital status, education, profession, nationality, ...
  - preferred sport, hobbies, favourite movies, ...
- user characteristics $\chi^{user} : \mathcal{U} \rightarrow \mathcal{A}^{user}$
  - sensitive information, hard to obtain

Items

- set of items $\mathcal{I}$
- item attributes $\mathcal{A}^{item} \subset \mathbb{R}^l$
  - movies: title, genre, year, director, actors, budget, nominations, ...
- item characteristics $\chi^{item} : \mathcal{I} \rightarrow \mathcal{A}^{item}$
  - quite costly to obtain
User feedback

\[ \phi : \mathcal{D} \rightarrow \mathcal{F} \]

- feedback values \( \mathcal{F} \subseteq \mathbb{R} \) observed on \( \mathcal{D} \subseteq \mathcal{U} \times \mathcal{I} \)

Implicit feedback

- information obtained about users by watching their natural interaction with the system
  - view, listen, scroll, bookmark, save, purchase, link, copy & paste, …
  - no burden on the user

Explicit feedback

- rating items on a rating scale (Likert’s scale)
- scoring items
- ranking a collection of items
- pairwise ranking of two presented items
- provide a list of preferred items
The recommendation task

Given

- \( \mathcal{U}, \mathcal{I} \) and \( \phi \)
- \( \chi^{user}, \chi^{item} \)
- some background knowledge \( \kappa \)

To learn

- model \( \hat{\phi} : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R} \) such that \( acc(\hat{\phi}, \phi, \mathcal{T}) \) is maximal
  - a set of “unseen” (or future) user-item pairs \( \mathcal{T} \subseteq (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{D} \)
  - \( acc \) is the accuracy of \( \hat{\phi} \) w.r.t. \( \phi \) measured on \( \mathcal{T} \)

It looks as a simple prediction task, however

- \( \chi^{user}, \chi^{item} \) and \( \kappa \) are often unknown
- usually, \( \mathcal{F} = \{1\} \) in case of implicit feedback
Two distinguished tasks

Rating prediction from explicit feedback

• How would Steve rate the movie Titanic more likely?

<table>
<thead>
<tr>
<th></th>
<th>Titanic</th>
<th>Pulp Fiction</th>
<th>Iron Man</th>
<th>Forrest Gump</th>
<th>The Mummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Ann</td>
<td>5</td>
<td>1</td>
<td></td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Mary</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Steve</td>
<td>?</td>
<td>3</td>
<td>4</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

• $\hat{\phi}(u, i)$ – predicted rating of the user $u$ for an item $i$

Item recommendation from implicit feedback

• Which movie(s) would does Steve see/buy more likely?

<table>
<thead>
<tr>
<th></th>
<th>Titanic</th>
<th>Pulp Fiction</th>
<th>Iron Man</th>
<th>Forrest Gump</th>
<th>The Mummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Ann</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Steve</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

• $\hat{\phi}(u, i)$ – predicted likelihood of a “positive” implicit feedback (ranking score) of the user $u$ for an item $i$
Types of RS

Knowledge-based

- recommendations are based on knowledge about users’ needs and preferences
  - $\chi^{item}$, $\kappa$, $\chi^{user}$

Content-based

- learn user’s interests based on the features of items previously rated by the user, using supervised machine learning techniques
  - $\chi^{item}$, $\phi$

Collaborative-filtering

- recognize similarities between users according to their feedbacks and recommend objects preferred by the like-minded users
  - $\phi$ (also $\chi^{item}$ and/or $\chi^{user}$ can be utilized)

Hybrid
Knowledge

user requirements

- value ranges
  - “the maximal accepted price should be lower than 8K EUR”
- functionality
  - “the car should be safe and suited for a family”
- interactive recommendation process needed
  - conversational systems

dependencies

- between user requirements and product properties
  - “a family car should have big trunk size”
- between different user requirements
  - “if a safe family car is required the maximal accepted price must be higher than 2000 EUR”
Representation

possible user requirements $V_{user}$

- max-price (0, ..., 10K), usage (family, ...), safety (small, medium, big)

possible item characteristics $V_{item}$

- price (0, ..., 100K), doors (3, 4, 5), terrain (yes, no), airbags (1, ..., 12)

compatibility constraints $\kappa_C$

- allowed instantiations of user properties
  - safety = big $\rightarrow$ max-price $\geq$ 2000

filter conditions $\kappa_F$

- item-specific selection criteriae
  - safety = big $\rightarrow$ airbags $>$ 4

item characteristics $\chi_{item}$

- “item constraints”
  - (id=1 $\land$ price=4K $\land$ doors=3 $\land$ terrain=no $\land$ airbags=2) $\lor$ ...
  - $\ldots \lor$ (id=100 $\land$ price=6K $\land$ doors=5 $\land$ terrain=no $\land$ airbags=6)
Recommendation

identifying products matching user’s requirements $REQ$

- can be viewed as a kind of $\chi^user$
- $REQ = \text{max-price}=7000 \land \text{usage}=\text{family} \land \text{safety}=\text{big}$

Constraint-based

- $RES = CSP(V_{user} \cup V_{item}, D, \kappa_C \cup \kappa_F \cup \chi^{item} \cup REQ)$
  - a set $D$ of finite domains for $V_{user}$ and $V_{item}$
  - $RES = \{\text{max-price}=7000, \text{usage}=\text{family}, \text{safety}=\text{big}, \text{id}=100,$
    $\text{price}=6K, \text{doors}=5, \text{terrain}=\text{no}, \text{airbags}=6\}$

 Conjunctive queries

- $\sigma_{\text{airbags} \geq 4 \land \text{price} \leq 8000}(\chi^{item})$

 Case-based

- $similarity(i, REQ) = \sum_{r \in REQ} w_r \cdot sim(i, r) / \sum_{r \in REQ} w_r$
  - weight $w_r$ for requirements $r$
  - similarity $sim(i, r)$ of items $i \in \chi^{item}$ to requirements $r \in REQ$
    - different types of $sim(i, r)$
    - user might maximize (e.g. safety) or minimize (e.g. price)
Interaction – default requirement values

**static** defaults for each user property
- \( \text{default}(\text{usage}) = \text{family} \)

**dependent** defaults on combinations of user requirements
- \( \text{default}(\text{usage} = \text{family}, \text{max-price} = 6000) \)

**derived** defaults from user requirements log
- the known requirement of the current user is \( \text{REQ} = \{ \text{price} = 6000 \} \)
- nearest-neighbor
  - 1-NN: \( \text{REQ} = \{ \text{price} = 6000, \text{doors} = 5, \text{terrain} = \text{no}, \text{airbags} = 6 \} \)
  - 3-NN: \( \text{REQ} = \{ \text{price} = 6000, \text{doors} = 4, \text{terrain} = \text{no}, \text{airbags} = 4 \} \)

<table>
<thead>
<tr>
<th>user id</th>
<th>price</th>
<th>doors</th>
<th>terrain</th>
<th>airbags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6000</td>
<td>5</td>
<td>no</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
<td>3</td>
<td>yes</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5500</td>
<td>4</td>
<td>yes</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6500</td>
<td>4</td>
<td>no</td>
<td>4</td>
</tr>
</tbody>
</table>
Which of the requirements should be changed?

- the MinRelax\(^1\) algorithm

\[\text{PQRS} = \text{compute the partial query results for all atoms} \quad a_i \quad \text{of} \quad Q \quad \text{for the product catalog} \quad P\]

MinRS = ∅

\[
\text{forall} \quad p_i \quad \in \quad P \quad \text{do}
\]

PSX = Compute the product-specific relaxation \(PSX(Q, p_i)\) by using PQRS

% Check relaxations that were already found

SUB = \(\{r \in \text{MinRS} \mid r \text{ is subquery of } PSX\}\)

if SUB ≠ ∅

% Current relaxation is superset of existing

continue with next \(p_i\)

endif

SUPER = \(\{r \in \text{MinRS} \mid \text{PSX is subquery of } r\}\)

if SUPER ≠ ∅

% Remove supersets

MinRS = MinRS \ SUPER

endif

% Store the new relaxation

MinRS = MinRS \cup PSX

endfor

return MinRS

\[\text{REQ} = \{r_1: \text{price} \leq 6000, \; r_2: \text{doors} = 5, \; r_3: \text{terrain} = \text{no}, \; r_4: \text{airbags} \geq 6\}\]

- \(\sigma[r_1 \land r_2 \land r_3 \land r_4](\chi_{\text{item}}) = \emptyset\)

- partial query results PQRS

<table>
<thead>
<tr>
<th>req</th>
<th>(i_1)</th>
<th>(i_2)</th>
<th>(i_3)</th>
<th>(i_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- product-specific relaxation

\[\text{PSX(REQ,} i_1) = \{r_2, r_3\}\]

---

Interaction – repairs for unsatisfiable requirements

**How should the unsatisfiable requirements be changed?**

- **derive repair actions** using $MinRS$
  
  for each $r \in MinRS$ derive $\pi_{\text{attributes}(r)}[\sigma_{\text{REQ} - r}](\chi_{\text{item}})$

\[
\begin{align*}
\text{REQ} &= \{r_1: \text{price} \leq 3000, \; r_2: \text{doors} = 5, \; r_3: \text{terrain} = \text{yes}, \; r_4: \text{airbags} \geq 6\} \\
\text{MinRS} &= \{\{r_2, r_4\}, \{r_2, r_3\}\} \\
\pi_{\text{doors, airbags}}[\sigma_{r_1, r_3}](\chi_{\text{item}}) &= \{(\text{doors} = 3, \text{airbags} = 4), (\text{doors} = 4, \text{airbags} = 2)\} \\
\pi_{\text{doors, terrain}}[\sigma_{r_1, r_4}](\chi_{\text{item}}) &= \{(\text{doors} = 4, \text{terrain} = \text{no})\}
\end{align*}
\]

- **repair alternatives**
  
  - $\text{REQ} = \{r_1: \text{price} \leq 3000, \; r_2: \text{doors} = 3, \; r_3: \text{terrain} = \text{yes}, \; r_4: \text{airbags} = 4\}$
  - $\text{REQ} = \{r_1: \text{price} \leq 3000, \; r_2: \text{doors} = 4, \; r_3: \text{terrain} = \text{yes}, \; r_4: \text{airbags} = 2\}$
  - $\text{REQ} = \{r_1: \text{price} \leq 3000, \; r_2: \text{doors} = 4, \; r_3: \text{terrain} = \text{no}, \; r_4: \text{airbags} = 6\}$
Interaction – ranking the retrieved items (1/2)

Contributions

• pre-defined set of dimensions

<table>
<thead>
<tr>
<th></th>
<th>value</th>
<th>quality</th>
<th>economy</th>
<th>safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>(0, 3000)</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(3000, 7000)</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>≥ 7000</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>terrain</td>
<td>yes</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>airbags</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>doors</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• contribution(item, dimension)

• $i = (\text{price}=4000 \land \text{terrain}=\text{no} \land \text{airbags}=2 \land \text{doors}=3)$
• $\text{contribution}(i, \text{quality}) = 3+2+2+3 = 10, \ldots$
Interaction – ranking the retrieved items (2/2)

Interest of the user in pre-defined dimensions

- user-defined
  - interest(quality) = 0.3
  - interest(economy) = 0.6
  - interest(safety) = 0.1

- derived from requirements
  - REQ = \{ price=4000 \land airbags=2 \}
    - contribution(req,quality) = 3+2 = 5
    - contribution(req,economy) = 2+4 = 6
    - contribution(req,safety) = 4+2 = 6
  - interest(quality) = 5/(5+6+6) = 5/17 = 0.3
  - interest(economy) = interest(safety) = 6/17 = 0.35

- other approaches

\[ utility(i) = \sum_{d \in \text{dimensions}} interest(d).contribution(i, d) \]
Interaction – Critiquing

a browsing-based approach used in case-based systems
- requirements refined w.r.t. the recommended item
  - “Show me cheaper cars” … “cars with more airbags” …
- unit vs. compound critiques
  - static (user wants more airbags but there are no such cars)

Dynamic critiquing
- suggests **critique patterns** according to the candidate items
  - association rules ($>_{\text{price}} \rightarrow <_{\text{doors}}$)
  - compound critique patterns ($>_{\text{price}} \land <_{\text{doors}}$)

<table>
<thead>
<tr>
<th>entry item</th>
<th>price</th>
<th>doors</th>
<th>terrain</th>
<th>airbags</th>
</tr>
</thead>
<tbody>
<tr>
<td>candidate item 1</td>
<td>4500</td>
<td>3</td>
<td>no</td>
<td>4</td>
</tr>
<tr>
<td>candidate item 2</td>
<td>5600</td>
<td>4</td>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>critique pattern 1</td>
<td>&gt;</td>
<td>&lt;</td>
<td>≠</td>
<td>=</td>
</tr>
<tr>
<td>critique pattern 2</td>
<td>&gt;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Content-based techniques

Real Men Change Diapers
Item features/characteristics ($\chi^{item}$)

- **explicitly defined**
  - attributes (price, airbags, doors, ...)
- **implicitly computed** from the document $d \in \mathcal{D}$
  - keywords $w$, boolean vector space model ...

$$TF - IDF(w, d) = TF(w, d) \cdot IDF(w, \mathcal{D})$$

- term frequency
  $$TF(w, d) = \frac{\text{freq}(w, d)}{\max\{\text{freq}(w', d) | w' \neq w\}}$$

- inverse document frequency
  $$IDF(w, \mathcal{D}) = \log\frac{|\mathcal{D}|}{|\{d \in \mathcal{D} | w \in d\}|}$$

- $\chi^{item} = (TF - IDF(w_1, item), \ldots, TF - IDF(w_k, item))$
Similarity-based recommendation

How to check if a user would like an item?

- If she **liked similar** items in the past...
  - feedback and similarity measures needed

**cosine vector similarity**

\[
sim_{cv}(\chi^i, \chi^j) = \frac{\chi^i \cdot \chi^j}{\|\chi^i\| \cdot \|\chi^j\|} = \frac{\sum_{k=1}^{n} \chi^i_k \chi^j_k}{\sqrt{\sum_{k=1}^{n} \chi^i_k^2} \sqrt{\sum_{k=1}^{n} \chi^j_k^2}}
\]

**k-nearest-neighbors**

- k most similar items user has got feedback on
  - recommend an item according to majority vote/average/etc.
- reflects on short-term preferences
  - considering only recent feedbacks
- simple to implement, small number of feedbacks is enough
Relevance feedback

Rocchio’s method

- find a **prototype** of “user’s ideal item”
- user-defined queries refined **iteratively**
  - good results already after the first iteration
- vector space model and similarity measure

input for \( i + 1 \)-th iteration

- \( D^- \) – documents with negative user feedback
- \( D^+ \) – documents with positive user feedback
- \( Q_i \) – actual query (vector) in the iteration \( i \)
- \( \alpha, \beta, \gamma \) – parameters

\[
Q_{i+1} = \alpha Q_i + \beta \left( \frac{1}{|D^+|} \sum_{d^+ \in D^+} d^+ \right) + \gamma \left( \frac{1}{|D^-|} \sum_{d^- \in D^-} d^- \right)
\]
Machine learning

learn a mapping $\hat{\phi} : A^{item} \rightarrow \mathbb{R}$ from

- item features/characteristics $\chi^{item}$
- user’s feedback $\phi$

with appropriate classification/regression techniques

- nearest-neighbor
- probabilistic methods
- decision trees, SVM
- ...

<table>
<thead>
<tr>
<th>item</th>
<th>$A^i$</th>
<th>$\phi(u, item)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_1$</td>
<td>$\chi^{item}(i_1)$</td>
<td>$\phi(u, i_1)$</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$\chi^{item}(i_2)$</td>
<td>$\phi(u, i_2)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$i_n$</td>
<td>$\chi^{item}(i_n)$</td>
<td>$\phi(u, i_n)$</td>
</tr>
</tbody>
</table>
A little commercial ;)
A fuzzy recommender system

First prototype developed during the NAZOU\textsuperscript{1} project (2006 – 2008)

• 2009 – 2012, developed without funding (BSc, MSc theses)
• 2012 – now, development within the CEZIS project

Main characteristics of the UPRE recommender module\textsuperscript{2}

• fuzzy preference models
  • on attributes (local)
  • aggregated (global)
  • top-k item retrieval
• explicit user feedback
• conversational

A hybrid of content-based and knowledge-based techniques...

• collaborative-filtering is planned

\textsuperscript{1}http://nazou.fiit.stuba.sk/
Preferences on attributes

defined by the user

computed from explicit feedback
Aggregated preferences

computed\(^1\) with **monotone prediction** techniques

\[ \mathcal{D} \xrightarrow{\leq^u_D = p^u_G} [0, 1] \]

\[ \text{data}_\mathbb{R} \xrightarrow{\#1} \text{monotone dataset} \mathbb{R} \]

\[ \Pi(\mathcal{D}_i, \leq^u_{\mathcal{D}_i}) \xrightarrow{p^u_{L_1}, \ldots, p^u_{L_n} \approx \leq^u_{\mathcal{D}_1}, \ldots, \leq^u_{\mathcal{D}_n}} [0, 1]^n \]

preference rules integrated\(^2\) with **top-k search**

- fast computation of pareto-optimal values
- implicit ranking of items in the resulting list


Iterative recommendation
Collaborative filtering
Neighborhood-based CF

Recommendation $\hat{\phi}(u, i)$ for user $u$ on item $i$ using $\phi$

- user-based
  - $\hat{\phi}(u, i)$ computed using feedback given by $k$ most similar users
    \[
    \mathcal{N}^{u,k}_i = \arg \max_{\mathcal{U}'} \sum_{v \in \mathcal{U}', v \neq u} \text{sim}(u, v)
    \]
    
    - $\mathcal{U}_i = \{v \in \mathcal{U} \mid \phi(v, i) \text{ is defined on } \mathcal{D}\}$

- item-based
  - $\hat{\phi}(u, i)$ computed using feedback given by $k$ most similar items
    \[
    \mathcal{N}^{i,k}_u = \arg \max_{\mathcal{I}'} \sum_{j \in \mathcal{I}', j \neq i} \text{sim}(i, j)
    \]
    
    - $\mathcal{I}_u = \{j \in I \mid \phi(u, j) \text{ is defined on } \mathcal{D}\}$
Item recommendation

What is the likelihood of an item \(i\) being liked by the user \(u\)?

- a simple \textbf{\textit{k-nearest-neighbor}} approach\(^1\)
  - user-based
    - an average similarity of most similar users which liked the item \(i\)
    
    \[
    \hat{\phi}_{ui} = \frac{\sum_{v \in \mathcal{N}_{i}^{u,k}} \text{sim}(u, v)}{k}
    \]

  - item-based
    - an average similarity of most similar items liked by the user \(u\)
    
    \[
    \hat{\phi}_{ui} = \frac{\sum_{j \in \mathcal{N}_{u}^{i,k}} \text{sim}(i, j)}{k}
    \]

assume that only (implicit) feedback \(\phi\) is available

- users and items represented by \textbf{sparse vectors}
  - cosine-vector similarity \(\text{sim}_{cv}\)

\(^1\)Simplified notation: \(\phi(u, i) \sim \phi_{ui}, \mathcal{I}_u \cap \mathcal{I}_v \sim \mathcal{I}_{uv}, \mathcal{U}_i \cap \mathcal{U}_j \sim \mathcal{U}_{ij}\)
Item recommendation – example

<table>
<thead>
<tr>
<th>$sim_{cv}(i,j)$</th>
<th>Titanic</th>
<th>Pulp Fiction</th>
<th>Iron Man</th>
<th>Forrest Gump</th>
<th>The Mummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titanic</td>
<td>1.0</td>
<td>0.87</td>
<td>0.67</td>
<td>0.82</td>
<td>0.67</td>
</tr>
<tr>
<td>Pulp Fiction</td>
<td>–</td>
<td>1.0</td>
<td>0.87</td>
<td>0.71</td>
<td>0.87</td>
</tr>
<tr>
<td>Iron Man</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
<td>0.41</td>
<td>0.67</td>
</tr>
<tr>
<td>Forrest Gump</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
<td>0.41</td>
</tr>
<tr>
<td>The Mummy</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$sim_{cv}(u,v)$</th>
<th>Joe</th>
<th>Ann</th>
<th>Mary</th>
<th>Steve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>1.0</td>
<td>0.75</td>
<td>0.75</td>
<td>0.87</td>
</tr>
<tr>
<td>Ann</td>
<td>–</td>
<td>1.0</td>
<td>0.75</td>
<td>0.58</td>
</tr>
<tr>
<td>Mary</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
<td>0.58</td>
</tr>
<tr>
<td>Steve</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**user-based**\(^1\)

- $N_{Titanic, Steve}^2 = \{Joe, Ann\}$, $\hat{\phi}_{ST} = \frac{scv(S,J)+scv(S,M)}{2} = \frac{0.87+0.58}{2} = 0.725$
- $N_{ForrestGump, Steve}^2 = \{Ann, Mary\}$, $\hat{\phi}_{ST} = \frac{scv(S,A)+scv(S,M)}{2} = \frac{0.58+0.58}{2} = 0.58$

**item-based**

- $N_{Steve}^{Titanic, 2} = \{PulpFiction, IronMan\}$, $\hat{\phi}_{ST} = \frac{scv(T,P)+scv(T,I)}{2} = \frac{0.87+0.67}{2} = 0.77$
- $N_{Steve}^{ForrestGump, 2} = \{PulpFiction, IronMan\}$, $\hat{\phi}_{ST} = \frac{scv(F,P)+scv(F,I)}{2} = \frac{0.71+0.41}{2} = 0.56$

\(^1\) $scv$ – cosine–vector similarity
Rating prediction

How would the user rate an item?

- user’s/item’s ratings are **biased**
  - optimistic, pessimistic users
  - items rated above or below average

**mean-centered** rating prediction

- user-based

\[ \hat{\phi}_{ui} = \bar{\phi}_u + \frac{\sum_{v \in \mathcal{N}^u_k} \text{sim}(u, v) \cdot (\phi_{vi} - \bar{\phi}_v)}{\sum_{v \in \mathcal{N}^u_k} |\text{sim}(u, v)|} \]

\[ \bar{\phi}_u = \frac{\sum_{i \in \mathcal{I}_u} \phi(u, i)}{|\mathcal{I}_u|} \]

- item-based

\[ \hat{\phi}_{ui} = \bar{\phi}_i + \frac{\sum_{j \in \mathcal{N}^i_u} \text{sim}(i, j) \cdot (\phi_{uj} - \bar{\phi}_j)}{\sum_{v \in \mathcal{N}^i_u} |\text{sim}(i, j)|} \]

\[ \bar{\phi}_i = \frac{\sum_{u \in \mathcal{U}_i} \phi(u, i)}{|\mathcal{U}_i|} \]
Pearson-correlation similarity

What similarity measure to use?

- $sim_{cv}$ doesn’t take into account the mean and variances of ratings

**pearson-correlation** similarity

$$sim_{pc}(u, v) = \frac{\sum_{i \in I_{uv}} (\phi_{ui} - \bar{\phi}_u)(\phi_{vi} - \bar{\phi}_v)}{\sqrt{\sum_{i \in I_{uv}} (\phi_{ui} - \bar{\phi}_u)^2 \sum_{i \in I_{uv}} (\phi_{vi} - \bar{\phi}_v)^2}}$$

$$sim_{pc}(i, j) = \frac{\sum_{u \in U_{ij}} (\phi_{ui} - \bar{\phi}_i)(\phi_{uj} - \bar{\phi}_j)}{\sqrt{\sum_{u \in U_{ij}} (\phi_{ui} - \bar{\phi}_i)^2 \sum_{i \in U_{ij}} (\phi_{uj} - \bar{\phi}_j)^2}}$$
Rating prediction – example

<table>
<thead>
<tr>
<th>( \text{sim}_{pc}(i, j) )</th>
<th>Titanic</th>
<th>Pulp Fiction</th>
<th>Iron Man</th>
<th>Forrest Gump</th>
<th>The Mummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titanic</td>
<td>1.0</td>
<td>-0.956</td>
<td>-0.815</td>
<td>NaN</td>
<td>-0.581</td>
</tr>
<tr>
<td>Pulp Fiction</td>
<td>–</td>
<td>1.0</td>
<td>0.948</td>
<td>NaN</td>
<td>0.621</td>
</tr>
<tr>
<td>Iron Man</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
<td>NaN</td>
<td>0.243</td>
</tr>
<tr>
<td>Forrest Gump</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
<td>NaN</td>
</tr>
<tr>
<td>The Mummy</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
</tr>
</tbody>
</table>

NaN values are usually converted to zero (rare in case of enough data)

<table>
<thead>
<tr>
<th>( \text{sim}_{pc}(u, v) )</th>
<th>Joe</th>
<th>Ann</th>
<th>Mary</th>
<th>Steve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>1.0</td>
<td>-0.716</td>
<td>-0.762</td>
<td>-0.005</td>
</tr>
<tr>
<td>Ann</td>
<td>–</td>
<td>1.0</td>
<td>0.972</td>
<td>0.565</td>
</tr>
<tr>
<td>Mary</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
<td>0.6</td>
</tr>
<tr>
<td>Steve</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**user-based**

- \( \mathcal{U}_{Titanic} = \{ Joe, Ann, Mary \} \), \( \mathcal{N}_{Titanic}^{Steve} = \{ Mary, Ann \} \)
- \( \bar{\phi}_{Steve} = \frac{11}{3} = 3.67 \), \( \bar{\phi}_{Mary} = \frac{12}{4} = 3 \), \( \bar{\phi}_{Ann} = \frac{13}{4} = 3.25 \)
- \( \phi_{ST} = \phi_S + \frac{s_{pc}(S,M) \cdot (\phi_{MT} - \bar{\phi}_M) + s_{pc}(S,A) \cdot (\phi_{AT} - \bar{\phi}_A)}{|s_{pc}(S,M)| + |s_{pc}(S,A)|} = 3.67 + \frac{0.6 \cdot (4 - 3) + 0.565 \cdot (5 - 3.25)}{0.6 + 0.565} = 1.36 \)

**item-based**

- \( \mathcal{I}_{Steve} = \{ Pulp Fiction, Iron Man, The Mummy \} \), \( \mathcal{N}_{Steve}^{Titanic} = \{ Iron Man, The Mummy \} \)
- \( \bar{\phi}_T = \frac{10}{3} = 3.34 \), \( \bar{\phi}_I = \frac{11}{3} = 3.67 \), \( \bar{\phi}_M = \frac{9}{3} = 3 \)
- \( \phi_{ST} = \phi_T + \frac{s_{pc}(T,I) \cdot (\phi_{SI} - \bar{\phi}_I) + s_{pc}(T,M) \cdot (\phi_{SM} - \bar{\phi}_M)}{|s_{pc}(T,I)| + |s_{pc}(T,M)|} = 3.34 + \frac{-0.815 \cdot (4 - 3.34) - 0.581 \cdot (4 - 3)}{0.815 + 0.581} = 2.73 \)
Matrix factorization
A latent space representation

Map users and items to a common latent space

- where dimensions or factors represent
  - items’ implicit properties
  - users’ interest in items’ hidden properties

---

Known factorization models (1/2)

\( \phi \) represented as a user-item matrix \( \Phi^{n \times m} \)

- \( n \) users, \( m \) items

Principal Component Analysis (PCA)

- transform data to a new coordinate system
- variances by any projection of the data lies on coordinates in decreasing order

\[ \text{The picture is taken from wikipedia.} \]
Known factorization models (2/2)

Singular Value Decomposition (SVD)

\[ \Phi = W^{n \times k} \Sigma^{k \times k} H^{n \times k^T} \]

- \( W^T W = I, \ H^T H = I \)
- column vectors of \( W \) are orthonormal eigenvectors of \( \Phi \Phi^T \)
- column vectors of \( H \) are orthonormal eigenvectors of \( \Phi^T \Phi \)
- \( \Sigma \) contains eigenvalues of \( W \) in descending order

PCA, SVD computed algebraically

- \( \Phi \) is a **big** and **sparse** matrix
  - approximations of PCA\(^1\), SVD\(^2\)

---


MF – rating prediction (1/2)

recommendation task

- to find $\hat{\phi} : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ such that $\text{acc}(\hat{\phi}, \phi, \mathcal{T})$ is maximal
  - $\text{acc}$ is the expected accuracy on $\mathcal{T}$
  - training $\hat{\phi}$ on $\mathcal{D}$ such that the empirical loss $\text{err}(\hat{\phi}, \phi, \mathcal{D})$ is minimal

a simple, approximative MF model

- only $W^{n \times k}$ and $H^{m \times k}$
- $k$ – the number of factors

$$
\Phi^{n \times m} \approx \hat{\Phi}^{n \times m} = WH^T
$$

- predicted rating $\hat{\phi}_{ui}$ of the user $u$ for the item $i$

$$
\hat{\phi}_{ui} = w_u h_i^T
$$
MF – rating prediction (2/2)

the **loss** function $err(\hat{\phi}, \phi, D)$

- squared loss

$$err(\hat{\phi}, \phi, D) = \sum_{(u,i) \in D} e_{ui}^2 = \sum_{(u,i) \in D} (\phi_{ui} - \hat{\phi}_{ui})^2 = \sum_{(u,i) \in D} (\phi_{ui} - w_u h_i^T)^2$$

the **objective function**

- **regularization** term $\lambda \geq 0$ to prevent overfitting
  - penalizing the magnitudes of parameters

$$f(\hat{\phi}, \phi, D) = \sum_{(u,i) \in D} (\phi_{ui} - w_u h_i^T)^2 + \lambda(\|W\|^2 + \|H\|^2)$$

The task is to find parameters $W$ and $H$ such that, given $\lambda$, the objective function $f(\hat{\phi}, \phi, D)$ is minimal.
Gradient descent

How to find a minimum of an “objective” function $f(\Theta)$?

- in case of MF, $\Theta = W \cup H$, and
- $f(\Theta)$ refers to the error of approximation of $\Phi$ by $WH^T$

Gradient descent

**input:** $f, \alpha, \Sigma^2$, stopping criteria
initialize $\Theta \sim \mathcal{N}(0, \Sigma^2)$
repeat
\[
\Theta \leftarrow \Theta - \alpha \frac{\partial f}{\partial \Theta}(\Theta)
\]
until approximate minimum is reached
return $\Theta$

stopping criteria

- $|\Theta^{old} - \Theta| < \epsilon$
- maximum number of iterations reached
- a combination of both
Stochastic gradient descent

if $f$ can be written as

$$f(\Theta) = \sum_{i=1}^{n} f_i(\Theta)$$

Stochastic gradient descent (SGD)

**input:** $f_i, \alpha, \Sigma^2$, stopping criteria

`initialize` $\Theta \sim \mathcal{N}(0, \Sigma^2)$

`repeat`

`for all` $i$ `in random order` `do`

`\Theta \leftarrow \Theta - \alpha \frac{\partial f_i}{\partial \Theta}(\Theta)`

`end for`

`until` approximate minimum is reached

`return` $\Theta$
MF with SGD

**Updating** parameters iteratively for each data point $\phi_{ui}$ in the opposite direction of the gradient of the objective function at the given point until a convergence criterion is fulfilled.

- updating the vectors $w_u$ and $h_i$ for the data point $(u, i) \in D$

\[
\frac{\partial f}{\partial w_u}(u, i) = -2(e_{ui} h_i - \lambda w_u)
\]

\[
\frac{\partial f}{\partial h_i}(u, i) = -2(e_{ui} w_u - 2\lambda h_i)
\]

\[
w_u(u, i) \leftarrow w_u - \alpha \frac{\partial f}{\partial w_u}(u, i) = w_u + \alpha(e_{ui} h_i - \lambda w_u)
\]

\[
h_i(u, i) \leftarrow h_i - \alpha \frac{\partial f}{\partial h_i}(u, i) = h_i + \alpha(e_{ui} w_u - \lambda h_i)
\]

where $\alpha > 0$ is a learning rate.
MF with SGD – Algorithm

Hyper-parameters: $k$, $\text{iters}$ (the max number of iteration), $\alpha$, $\lambda$, $\Sigma^2$

$W \leftarrow \mathcal{N}(0, \Sigma^2)$

$H \leftarrow \mathcal{N}(0, \Sigma^2)$

for $\text{iter} \leftarrow 1, \ldots, \text{iters} \cdot |\mathcal{D}|$ do

\[ \hat{\phi}_{ui} \leftarrow 0 \]

for $j \leftarrow 1, \ldots, k$ do

\[ \hat{\phi}_{ui} \leftarrow \hat{\phi}_{ui} + W[u][j] \cdot H[i][j] \]

end for

$e_{ui} = \phi_{ui} - \hat{\phi}_{ui}$

for $j \leftarrow 1, \ldots, k$ do

\[ W[u][j] \leftarrow W[u][j] + \alpha \cdot (e_{ui} \cdot H[i][j] - \lambda \cdot W[u][j]) \]

\[ H[i][j] \leftarrow H[i][j] + \alpha \cdot (e_{ui} \cdot W[u][j] - \lambda \cdot H[i][j]) \]

end for

end for

return $\{W, H\}$
MF with SGD – Example

Let’s have the following hyper-parameters:

\( K = 2, \alpha = 0.1, \lambda = 0.15, \text{iter} = 150, \sigma^2 = 0.01 \)

\[
\Phi = \begin{bmatrix}
1 & 4 & 5 & 3 \\
5 & 1 & 5 & 2 \\
4 & 1 & 2 & 5 \\
3 & 4 & 2 & 4
\end{bmatrix}
\]

Results are:

\[
W = \begin{bmatrix}
1.1995242 & 1.1637173 \\
1.8714619 & -0.02266505 \\
2.3267753 & 0.27602595 \\
2.033842 & 0.539499
\end{bmatrix}
\]

\[
H^T = \begin{bmatrix}
1.6261001 & 1.1259034 & 2.131041 & 2.2285593 & 1.6074764 \\
-0.40649664 & 0.7055319 & 1.0405376 & 0.39400166 & 0.49699315
\end{bmatrix}
\]

Results\(^1\) are:

\[
\hat{\Phi} = \begin{bmatrix}
1.477499 & 2.171588 & 3.767126 & 3.131717 & 2.506566 \\
3.052397 & 2.091094 & 3.964578 & 4.161733 & 2.997066 \\
3.671365 & 2.814469 & 5.245668 & 5.294111 & 3.877419 \\
3.087926 & 2.670543 & 4.895569 & 4.745101 & 3.537480
\end{bmatrix}
\]

\(^1\)Note, that these hyper-parameters are just picked up in an ad-hoc manner. One should search for the “best” hyper-parameter combinations using e.g. grid-search (a brute-force approach).

\(^2\)Thanks to my colleague Thai-Nghe Nguyen for computing an example.
**Biased MF**

**baseline estimate**
- user-item bias

\[ b_{ui} = \mu + b'_u + b''_i \]
- \( \mu \) – average rating across the whole \( \mathcal{D} \)
- \( b', b'' \) – vectors of user and item biases, respectively

**prediction**

\[ \hat{\phi}_{ui} = \mu + b'_u + b''_i + w_u h_i \]

**objective function** to minimize

\[
f(\phi, \hat{\phi}, \mathcal{D}) = \sum_{(u,i) \in \mathcal{D}} (\phi_{ui} - \mu - b'_u - b''_i - w_u h_i)^2 + \lambda(\|W\|^2 + \|H\|^2 + b'^2 + b''^2)
\]
Biased MF with SGD

similar to unbiased MF

- initialize average and biases

\[ \mu = \frac{\sum_{(u,i) \in D} (u,i)}{|D|} \]

\[ b' \leftarrow (\phi_{u1}, \ldots, \phi_{un}) \]

\[ b'' \leftarrow (\phi_{i1}, \ldots, \phi_{im}) \]

- update average and biases

\[ \mu \leftarrow \mu - \frac{\partial f}{\partial \mu} (u, i) = \mu + \alpha e_{ui} \]

\[ b' \leftarrow b' - \frac{\partial f}{\partial b'} (u, i) = b' + \alpha (e_{ui} - \lambda b') \]

\[ b'' \leftarrow b'' - \frac{\partial f}{\partial b''} (u, i) = b'' + \alpha (e_{ui} - \lambda b'') \]
MF – item recommendation

to predict a personalized ranking score\(^1\) \(\hat{\phi}_{ui}\)

- how the item \(i\) is preferred to other items for the user \(u\)
- to find \(W\) and \(H\) such that \(\hat{\Phi} = WH^T\)

\[
\hat{\phi}_{ui} = w_u h_i^T
\]

problem: positive feedback only

- **pairwise ranking** data

\[
D_p = \{(u, i, j) \in D | i \in I_u \land j \in I \setminus I_u\}
\]

---

\(^1\) S. Rendle et al. (2009). BPR: Bayesian Personalized Ranking from Implicit Feedback. 25th Conference on Uncertainty in Artificial Intelligence.
Bayesian formulation of the problem

- $\succ$ – the unknown preference structure (ordering)
  - we use the derived pairwise ranking data $\mathcal{D}_p$
- $\Theta$ – parameters of an arbitrary prediction model
  - in case of MF, $\Theta = W \cup H$

\[
p(\Theta | \succ) \propto p(\succ | \Theta)p(\Theta)
\]

Prior probability

- assume independence of parameters
- assume, $\Theta \sim N(0, \frac{1}{\lambda} I)$

\[
p(\Theta) = \prod_{\theta \in \Theta} \sqrt{\frac{\lambda}{2\pi}} e^{-\frac{1}{2} \lambda \theta^2}
\]
MF – Bayesian Personalized Ranking (2/3)

likelihood

• assume users’ feedbacks are independent
• assume, ordering of each pair is independent

\[
p(\succ | \Theta) = \prod_{u \in U} p(\succ_u | \Theta) = \prod_{(u, i, j) \in D_p} p(i \succ_u j | \Theta)
\]

• using the ranking scores \( \hat{\phi} \)

\[
p(i \succ_u j | \Theta) = p(\hat{\phi}_{ui} - \hat{\phi}_{uj} > 0) = \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) = \frac{1}{1 + e^{-(\hat{\phi}_{ui} - \hat{\phi}_{uj})}}
\]
maximum a posteriori estimation of $\Theta$

$$\arg \max_{\Theta} p(\Theta, \succ) =$$

$$\arg \max_{\Theta} p(\succ \mid \Theta)p(\Theta) =$$

$$\arg \max_{\Theta} \ln p(\succ \mid \Theta)p(\Theta) =$$

$$\arg \max_{\Theta} \ln \prod_{(u,i,j) \in D_p} \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) \sqrt{\frac{\lambda}{2\pi}} e^{-\frac{1}{2} \lambda \theta^2}$$

$$\arg \max_{\Theta} \sum_{(u,i,j) \in D_p} \ln \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \|\Theta\|^2$$

$BPR\text{-\textit{OPT}}$
Finding parameters for BPR-OPT

Stochastic gradient ascent

\[
\frac{\partial BPR - OPT}{\partial \Theta} \propto \sum_{(u,i,j) \in D_p} \frac{e^{-(\hat{\phi}_{ui} - \hat{\phi}_{uj})}}{1 + e^{-(\hat{\phi}_{ui} - \hat{\phi}_{uj})}} \frac{\partial}{\partial \Theta}(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \Theta
\]

\[
\frac{\partial}{\partial \theta}(\hat{\phi}_{ui} - \hat{\phi}_{uj}) = \begin{cases} 
(h_i - h_j) & \text{if } \theta = w_u \\
w_u & \text{if } \theta = h_i \\
-w_u & \text{if } \theta = h_j \\
0 & \text{else} 
\end{cases}
\]

LearnBPR

**input:** \( f_i, \alpha, \Sigma^2, \text{stopping criteria} \)

initialize \( \Theta \sim \mathcal{N}(0, \Sigma^2) \)

repeat

\[
\Theta \leftarrow \Theta + \alpha \frac{\partial BPR - OPT}{\partial \Theta}(\Theta)
\]

until approximate maximum is reached

return \( \Theta \)
BPR-OPT vs AUC

Area under the ROC curve (AUC)

- probability that the ranking of a randomly drawn pair is correct

\[
AUC = \sum_{u \in U} AUC(u) = \frac{1}{|U|} \frac{1}{|I_u| \left| \mathcal{I} \setminus \mathcal{I}_u \right|} \sum_{(u,i,j) \in \mathcal{D}_p} \delta(\hat{\phi}_{ui} > \hat{\phi}_{uj})
\]

- \( \delta(\hat{\phi}_{ui} > \hat{\phi}_{uj}) = 1 \) if \( \hat{\phi}_{ui} > \hat{\phi}_{uj} \), and 0, else

Smoothed AUC objective function with regularization of parameters

\[
AUC - OPT = \sum_{(u,i,j) \in \mathcal{D}_p} \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \| \Theta \|^2
\]

\[
BPR - OPT = \sum_{(u,i,j) \in \mathcal{D}_p} \ln \sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda \| \Theta \|^2
\]
More info on ranking with factorization models
Issues worth to mention
The cold-start problem arises when not enough collaborative information is available

- new user or new item

possible solutions

- recommend popular items, “predict” global average, ... 
- utilize item attributes

---

1 Z. Gantner et al. (2010). Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. 10th IEEE International Conference on Data Mining.
Context-aware recommendation

**Context** is any additional information, besides $\chi_{user}$, $\chi_{item}$, $\phi$ and $\kappa$, that is relevant for the recommendation\(^1\)

- time, location, companion (when, where and with whom the user wants to watch some movie)

---

Evaluating RS (1/3)

experiments

• **offline**
  • no interaction with real users, need to simulate user behaviour
  • low cost, short time
  • answers only a few questions, e.g. the predictive power of techniques

• **user studies**
  • observing test subjects’ behaviour in the system
  • questionnaires
  • expensive, small scale,

• **online evaluation**
  • redirect a small part of the traffic to an alternative recommendation engine
  • risky – we can lose some customers
  • good to do after an offline testing of a recommendation engine shows good results
properties of a recommender system

- user preference
  - Which one from different RS users prefer more?
- prediction accuracy
  - How precise recommendations does a RS provide?
- coverage
  - What proportion of all items can a RS ever recommend? To what proportion of users can a system recommend? How rich a user profile should be for making recommendation?
- cold-start as a subproblem ("coldness" of an item)
- confidence
  - How confident the system is with its recommendation? (e.g. depends on amount of data in CF…)
- novelty
  - Does the system recommends items the user did not know about?
- trust
  - What is the users’ trust in recommendation?
Evaluating RS (3/3)

- **serendipity**
  - How surprising the recommendations are? (e.g. a new movie with the user’s favourite actor can be novel but not surprising)
- **diversity**
  - How “colorful” the recommendations are?
- **utility**
  - How useful a RS is for the provider/user? (e.g. generated revenue)
- **robustness**
  - How stable a RS is in presence of fake information?
- **privacy**
  - How users’ privacy is retained in a RS?
- **adaptivity**
  - How does a RS adapt to changes in the item collection?
- **scalability**
  - How scalable a RS is?
The MyMediaLite library
MyMediaLite Recommendation Algorithm Library

MyMediaLite

- is lightweight, multi-purpose library
- is mainly a library, meant to be used by other applications
- is free software (under the terms of the GNU General Public License)
- was developed by Zeno Gantner, Steffen Rendle, and Christoph Freudenthaler at University of Hildesheim

http://ismll.de/mymedialite
MyMediaLite features

major

• scalable implementations of many state-of-the-art recommendation methods
• evaluation framework for reproducible research
• ready to be used: command line tools, not programming necessary

next features

• usable from C#, Python, Ruby, F#
• Java ports available
• written in C#, runs on Mono
• regular releases (ca. 1 every 2 months)

using for

• rating prediction
• item recommendation
• group recommendation
Methods in MyMediaLite

State-of-the-art recommendation methods in MyMediaLite:

- kNN variants
- Online-Updating Regularized Kernel Matrix Factorization [Rendle and Schmidt-Thieme, RecSys 2009]
- SocialMF [Jamali and Ester, RecSys 2010] Freudenthaler at University of Hildesheim
- Asymmetric Factor Models (AFM) [Paterek, KDD Cup 2007]
- SVD++ [Koren, KDD 2008]
- Weighted Regularized Matrix Factorization (WR-MF) [Hu and Koren, ICDM 2008], [Pan et al., ICDM 2008]
- BPR-MF [Rendle et al., UAI 2009]
Data

e.g. MovieLens, Netflix

<table>
<thead>
<tr>
<th>user ID</th>
<th>item ID</th>
<th>rating</th>
<th>timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>196</td>
<td>242</td>
<td>3</td>
<td>881250949</td>
</tr>
<tr>
<td>186</td>
<td>302</td>
<td>3</td>
<td>891717742</td>
</tr>
<tr>
<td>22</td>
<td>377</td>
<td>1</td>
<td>878887116</td>
</tr>
<tr>
<td>244</td>
<td>51</td>
<td>2</td>
<td>880606923</td>
</tr>
</tbody>
</table>

Remarks

• user and item IDs can be (almost) arbitrary strings
• separator: whitespace, tab, comma, ::
• alternative date and time format: yyyy-mm-dd
• rating and date and time fields are optional
• import script; Unix tools, Perl, Python . . .
Usage: Explicit Feedback I

Getting Help

- rating prediction --help

Data sets

- rating prediction --training-file=u1.base
  --test-file=u1.test

Recommender Options

- rating prediction --training-file=u.data
  --test-ratio=0.2

Fixing the Random Seed

- rating prediction ... --random-seed=1

Choosing a Recommender (algorithm)

- rating prediction ... --recommender=UserAverage
- rating prediction ... --recommender=UserItemBaseline
Iterative Recommenders

- rating prediction
  ... --recommender=BiasedMatrixFactorization
  --find-iter=1 --max-iter=30

Recommender Options (Hyperparameters)

- rating prediction
  ... --recommender-options=’’num factors=5’’

- rating prediction ...
  --recommender-options=’’num_factors=5 reg=0.05’’

SVD++

- rating prediction ...
  --recommender=SVDPlusPlus
  --recommender-options=’’num_factors=5 reg=0.1
  learn rate=0.01’’
Example: rating prediction

input data

- user_id item_id rating

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

where user_id and item_id are integers referring to users and items, respectively, and rating is a floating-point number expressing how much a user likes an item

- separator: either spaces, tabs, or commas
- only three columns, all additional columns will be ignored

usage of the rating prediction program

rating_prediction --training-file=TRAINING_FILE --test-file=TEST_FILE --recommender=METHOD [OPTIONS]
Example: simple and advanced recommender

simple recommender

- run: `rating_prediction --training-file=u1.base --test-file=u1.test --recommender=UserAverage`
- output: UserAverage training time 00:00:00.000098 RMSE 1.063 MAE 0.85019 testing time 00:00:00.032326

advanced recommender

- run: `rating_prediction --training-file=u1.base --test-file=u1.test --recommender=BiasedMatrixFactorization`
- output: BiasedMatrixFactorization num_factors=10 regularization=0.015 learn_rate=0.01 num_iter=30 init_mean=0 init_stdev=0.1 training time 00:00:03.3575780 RMSE 0.96108 MAE 0.75124 testing time 00:00:00.0159740
Example: hyperparameter search

- run: rating prediction --training-file=u1.base
  --test-file=u1.test
  --recommender=BiasedMatrixFactorization
  --recommender-options="num_factors=20 num_iter=0"
  --max-iter=25 --num-iter=0

- output:

```
RMSE 1.17083 MAE 0.96918 iteration 0
RMSE 1.01383 MAE 0.8143 iteration 1
RMSE 0.98742 MAE 0.78742 iteration 2
RMSE 0.97672 MAE 0.77668 iteration 3
RMSE 0.9709 MAE 0.77078 iteration 4
RMSE 0.96723 MAE 0.76702 iteration 5
RMSE 0.96466 MAE 0.76442 iteration 6
RMSE 0.96269 MAE 0.76241 iteration 7
RMSE 0.96104 MAE 0.76069 iteration 8
RMSE 0.95958 MAE 0.75917 iteration 9
RMSE 0.95825 MAE 0.75783 iteration 10
RMSE 0.95711 MAE 0.75667 iteration 11
RMSE 0.95626 MAE 0.75569 iteration 12
RMSE 0.95578 MAE 0.75501 iteration 13
RMSE 0.95573 MAE 0.75467 iteration 14
RMSE 0.9561 MAE 0.756 iteration 15
RMSE 0.9569 MAE 0.75499 iteration 16
RMSE 0.95802 MAE 0.75551 iteration 17
RMSE 0.95942 MAE 0.75623 iteration 18
RMSE 0.96102 MAE 0.7571 iteration 19
RMSE 0.96277 MAE 0.75806 iteration 20
RMSE 0.96463 MAE 0.75909 iteration 21
RMSE 0.96656 MAE 0.76017 iteration 22
RMSE 0.96852 MAE 0.7613 iteration 23
RMSE 0.9705 MAE 0.76246 iteration 24
RMSE 0.97247 MAE 0.76364 iteration 25
```
Why use MyMediaLite?

- simple
- free
- scalable
- well-documented
- well-tested

Possibility of using extra features

- Item Recommendation Tool (very similar usage like rating prediction)
- --cross-validation=K
- --chronological-split=2012-01-01
- --online-evaluation
- --save-model=FILE --load-model=FILE
- --measure=RMSE --epsilon=0.001
- ...

Tutorial on Recommender Systems
Summary

Seriously.....

Do I need to say more?
Types of RS (1/2)

Knowledge-based
- pros: no cold-start, deterministic
- cons: knowledge-engineering needed, static

Content-based
- pros: no collaborative information needed
- cons: content is needed, cold-start for new users, no serendipity

Collaborative-filtering
- pros: no user nor item attributes needed, serendipity
- cons: cold-start for new users and items
That's all Folks!
Many thanks go to

- Štefan Pero for his great help
- Zeno Gantner for providing materials and help regarding MyMediaLite
- Artus Krohn-Grimbergh for a picture from his PhD defense presentation
- all my colleagues and friends from ICS, UPJŠ and the ISMLL, UHI as well as other institutes for helping me to understand these things ;)

...also,

- all the people providing their materials (funny pictures, graphs, leaderboards, ...) on the web

...and, last but not least

- YOU for your attention!
Questions?

Tomas.Horvath@upjs.sk
http://www.ics.upjs.sk/~horvath