Building a Recommendation system for e-commerce

AI Ukraine 2017
About me

Alex Konduforov

Data Science Group Leader @

Co-organizer @ Kharkiv AI club
“31% of ecommerce revenues were generated from personalized product recommendations” - Barilliance.com, 2014

“Already, 35% of what consumers purchase on Amazon and 75% of what they watch on Netflix come from product recommendations based on such algorithms” - McKinsey
Evolution

Amazon

Recommended for You

Amazon.com has new recommendations for you based on items you purchased or told us you own.

- The Little Big Things: 163 Ways to Pursue Excellence
- Fascinate: Your 7 Triggers to Persuasion and Captivation
- Sherlock Holmes (Blu-ray)
- Alice in Wonderland (Blu-ray)

Customers Who Bought This Item Also Bought

- Oliver Twist (Dover Thrift Editions) by Charles Dickens
  - Paperback: $3.50
- David Copperfield (Dover Thrift Editions) by Charles Dickens
  - Paperback: $5.00
- JANE EYRE
  - Paperback: $2.99

Netflix

Top 10 for Angela

- WAITING FOR FOREVER
- THE FATBOY CHRONICLES
- howstuffworks

Ukraine
Why recommendations so important

Traditional Retail can serve only most popular products.

Online can serve much more products, but it’s overwhelming for customers.
How to apply

Website recommendations
• Main goals: cross-sale, save customer time

Personalized marketing emails
• Main goals: return customer on the website, upsale
Recommendation systems
Formulation of the problem

Goal of recommendation system is to predict blanks in the utility matrix

<table>
<thead>
<tr>
<th></th>
<th>LOTR</th>
<th>Star Wars</th>
<th>GoT</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Bob</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Gathering Data

**Explicit**
- Ask people to rate items
- Cons: doesn’t scale, only a small fractions of users leave ratings and reviews

**Implicit**
- Inferences from user actions
- Cons: only one value, no difference between dislike and unknown
Main approaches

- Non-personalized Summary Statistics
- Content-based Filtering
- Collaborative Filtering (nearest neighbors)
  - User-User
  - Item-Item
  - Matrix Factorization
- Hybrid
- Probability models
- etc.
Ecommerce specifics

- Implicit customer feedback (views, purchases, other actions)
- Utility matrix with only 1’s
  - Possible to calculate some score but more complicated
- Collaborative Filtering + Matrix Factorization
- Not every similarity/distance works
Collaborative Filtering

Method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating)
Types of Collaborative Filtering

**User-to-user**
1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for)
2. Use the ratings from those like-minded users to calculate a prediction for the active user

**Item-to-item**
1. Build an item-item matrix determining relationships between pairs of items
2. Infer the tastes of the current user by examining the matrix and matching that user's data
User-to-user

- Let $r_x$ be the vector of user $x$’s ratings
- Let $N$ be the set of $k$ users most similar to $x$ who have also rated item $i$
- Prediction for user $x$ and item $i$

- Option 1: $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
- Option 2: $r_{xi} = \frac{\sum_{y \in N} s_{xy} r_{yi}}{\sum_{y \in N} s_{xy}}$
  where $s_{xy} = \text{sim}(x,y)$
### Similarity

#### Jaccard similarity
- \( \text{sim}(A,B) = \frac{|r_A \cap r_B|}{|r_A \cup r_B|} \)
- \( \text{sim}(A,B) = 1/5; \text{sim}(A,C) = 2/4 \)
  - \( \text{sim}(A,B) < \text{sim}(A,C) \)

Ignores rating values

#### Pearson similarity (~cosine)

\[
\text{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}}
\]

Contrary to cosine treats missing values not as negatives, but as zeros

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Item-to-item

- 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 $x$ on item $j$
$N(i;x)$... set items rated by $x$ similar to $i$
User-based vs. item-based

In practice, item-based CF outperforms user-based CF in many cases.

Item-based CF pros:
- better when user size is large
- better for new users
- no need to recalculate so often as user-based (caching)
- more likely to converge => better accuracy
Matrix Factorization

Approximates the utility matrix as product of low-rank matrices

Identifies latent features
Matrix Factorization algorithm

- Initialize P and Q with small random numbers
- Teach P and Q
  - Alternating Least Squares
  - Stochastic Gradient Descent

\[ r_{ui} = p_u^T q_i \]

\[ L = \sum_{u,i \in S} (r_{ui} - x_u^T y_i)^2 + \lambda_x \sum_u ||x_u||^2 + \lambda_y \sum_u ||y_i||^2 \]
**MF example**

Latent features are calculated via MF:

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating Matrix</th>
<th>User Matrix</th>
<th>Item Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>W</td>
<td>4.5</td>
<td>1.2</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>2.0</td>
<td>0.8</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>4.0</td>
<td>1.4</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>5.0</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>B</td>
<td>W</td>
<td>4.0</td>
<td>1.4</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>3.5</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>2.0</td>
<td>1.7</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>2.0</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>C</td>
<td>W</td>
<td>5.0</td>
<td>1.5</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>4.0</td>
<td>1.0</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>2.0</td>
<td>1.1</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>4.0</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>D</td>
<td>W</td>
<td>3.5</td>
<td>1.2</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>4.0</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>1.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Evaluation

**Academic metrics:**
- RMSE
- MAE
- Precision/Recall

(all may have low correlation with actual user satisfaction)

**Business metrics:**
- CTR/CVR
- ROI
- CLV (Customer Lifetime Value)

**Customer metrics:**
- Coverage – covering more items for recommendations
- Diversity – higher variety of items (rich-get-richer effect)
- Novelty – recommending new items
Sparsity problem

There is an approximate threshold of 99.5% sparsity for CF to work

- Add product views, shopping cart and other activities
  - Decreases sparsity
- Matrix Factorization, SVD
  - No zeros
- Content description
  - Hybrid content-based + collaborative filtering
Cold start problem

User cold start: new users
- Non-personalized recommendations: most popular, highly rated
- Use user profile (age, gender, etc.) and segment

Item cold start: new items
- Don’t recommend (what about news?)
- Use item content if available
Scalability problem

Amazon had 30+ mln of customers and several million catalog items.

Solution:

• Reduce number of customers by randomly sampling them or discarding customers with few purchases
• Reduce number of items by discarding very popular or unpopular items
• Dimensionality reduction techniques such as clustering
Other challenges

- Gray sheep
- Diversity and the long tail (rich-get-richer effect)
- Shilling attacks
- Privacy
  - EU has quite strict rules and culture of data privacy
  - Netflix was sued for dataset publication => cancellation of a second Netflix Prize competition in 2010
Implementation questions

- For CF+MF - automatic model updates? how frequently?
- How and where to store MF model?
- Emails - track recommended items and don’t duplicate
## Tools

<table>
<thead>
<tr>
<th>Language / Stack</th>
<th>Tools / Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>recommenderlab, recosystem</td>
</tr>
<tr>
<td>Python</td>
<td>Scikit-learn crab, implicit, python-recsys, Surprise GraphLab Create ($$$)</td>
</tr>
<tr>
<td>Java</td>
<td>LensKit, Cofi Apache Mahout</td>
</tr>
<tr>
<td>C++</td>
<td>SVDFeature, Waffles, Graphchi, LIBMF GraphLab Create ($$$)</td>
</tr>
<tr>
<td>C#</td>
<td>Nreco</td>
</tr>
<tr>
<td>Node.JS</td>
<td>raccoon</td>
</tr>
<tr>
<td>SaaS</td>
<td>Google Cloud Prediction API Amazon Machine Learning PredictionIO SuggestGrid</td>
</tr>
</tbody>
</table>

[https://github.com/grahamjenson/list_of_recommender_systems](https://github.com/grahamjenson/list_of_recommender_systems)
Materials

- A Gentle Introduction to Recommender Systems with Implicit Feedback
- Matrix Factorization: A Simple Tutorial and Implementation on Python
- Matrix Factorization Model in Collaborating Filtering
- Finding similar music using Matrix Factorization
- Mining of Massive Databases (Stanford), Chapter 9
- AI Ukraine 2014 - Сергей Николенко - Рекомендательные системы
- Recommender Systems specialization (Coursera)
Thank you!

Skype: alex_konduforov
Email: alex.konduforov@altexsoft.com