Deep Recommender System for Personalized Search & Discovery

T. Poutanen & M. Volkovs
Outline

• Challenges of traditional systems
• Our approach
  – Input
  – Model architecture
  – Generalization to multiple inputs
• Experimental Results
• Q&A
Challenges with Traditional Recommender Systems

1. Incorporating user context & session-based data in real-time

2. Leveraging all user and item information to improve accuracy and handle cold start
Challenge 1: New Interactions

- Latent models are typically trained with gradient descent:
  \[
  \sum_{i,j} w_{ij} (R_{ij} - U_i V_j)^2 \quad \sum_{R_{ij} > R_{ik}} \log(1 + e^{-U_i (V_j - V_k)})
  \]

- When new information is available in \( R \) it is difficult to propagate it to \( U \) and \( V \):
  - Gradient updates are expensive and can destabilize the model
  - Blending with user/item vectors is not principled and requires extensive tuning
  - Many systems opt not to update in real time and conduct regular retraining instead
Challenge 2: Cold Start

- **Approach 1**: add additional objective terms to model content
  - Resulting objective is complex, difficult to tune and optimize
  - Diverts model capacity from interaction data
  - Very difficult to incorporate new interactions in real-time

- **Approach 2**: make $U$ and $V$ functions of content features
  - Model is only as good as the underlying content features
  - Doesn't work well if many items have noisy/incomplete content
  - Can't handle new interaction data without further training
Time for a different approach?

- Is there a generalized deep learning framework that can replace collaborative filtering?
- Can we use nets to learn compact embeddings?
- Can we input all available knowledge?
- Can the net leverage recent advances in deep learning?
The Non-Intuitive Enabling Idea

- Rows/columns of $\mathbf{R}$ satisfy all conditions of “valid” feature input (numeric fixed length vectors); can we use them directly as input?
  
- Turns out we can!
Our Approach: Input

- users
- items
- Interaction matrix
Our Approach: Input

- Select a subset of users
Our Approach: Input

- Select a subset of users
- Items are represented by all the users who interacted with them

Diagram:
- Users
- Items
- Item feature
- Item feature input
Our Approach: Input

- Select a subset of users
- Items are represented by all the users who interacted with them
- Users are represented by all the items they interacted with
### Illustrative Example

$$R$$

<table>
<thead>
<tr>
<th>Users</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:** The matrix $$R$$ represents a user-item interaction matrix.
Illustrative Example

- Select a subset of users

<table>
<thead>
<tr>
<th>Users</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Illustrative Example

- Select a subset of users
- $\text{Item}_2 = [1, 1, 0, 0, 0, 0, 1, 0, 0, 0]$
Illustrative Example

- Select a subset of users
- \( \text{Item}_2 = [1,1,0,0,0,0,1,0,0,0] \)
- \( \text{User}_2 = \text{Item}_2 + \text{Item}_7 \)
Illustrative Example

- Select a subset of users
- $\text{Item}_2 = [1,1,0,0,0,0,1,0,0,0]$ 
- $\text{User}_2 = \text{Item}_2 + \text{Item}_7 = [1,2,1,0,0,0,2,0,0,1]$
Our Approach: Model

- Parametrized Siamese mapping from feature inputs to latent representations [1]:

\[ S_{ij} = U_i \times V_j \]

[1] Learning a Similarity Metric Discriminatively, with Application to Face Verification, S. Chopra, R. Hadsell and Y. LeCunn, CVPR'05
Our Approach: Model

- Parametrized Siamese mapping from feature inputs to latent representations [1]:
  
  \[ S_{ij} = U_i \times V_j \]

- \( f \) is a deep neural net

[1] Learning a Similarity Metric Discriminatively, with Application to Face Verification, S. Chopra, R. Hadsell and Y. LeCunn, CVPR'05
Our Approach: Model

- Parametrized Siamese mapping from feature inputs to latent representations [1]:

\[ S_{ij} = U_i \times V_j \]

- \( f \) is a deep neural net
- Can be trained with any existing objective

[1] Learning a Similarity Metric Discriminatively, with Application to Face Verification, S. Chopra, R. Hadsell and Y. LeCunn, CVPR'05
Our Approach: Model

• Parametrized Siamese mapping from feature inputs to latent representations [1]:

\[ S_{ij} = U_i \times V_j \]

- \( f \) is a deep neural net
- Can be trained with any existing objective
- Number of learned parameters is reduced

[1] Learning a Similarity Metric Discriminatively, with Application to Face Verification, S. Chopra, R. Hadsell and Y. LeCunn, CVPR'05
Our Approach: Model

- Parametrized Siamese mapping from feature inputs to latent representations [1]:

\[ f(\cdot, \theta) \]

\[ S_{ij} = U_i \times V_j \]

- \( f \) is a deep neural net
- Can be trained with any existing objective
- Number of learned parameters is reduced
- After inference only \( U \) and \( V \) are used for retrieval

[1] Learning a Similarity Metric Discriminatively, with Application to Face Verification, S. Chopra, R. Hadsell and Y. LeCunn, CVPR'05
Challenges with Traditional Recommender Systems

1. Incorporating user context & session-based data in real-time

2. Leveraging all user and item information to improve accuracy and handle cold start
New Interactions

- Reminder: User input is the sum of columns
New Interactions

- Session context should influence recommendations
New Interactions

- Session context should influence recommendations
- Incrementally update user input
New Interactions

- Session context should influence recommendations
- Incrementally update user input
- Recalculate $\mathbf{U}$ with one forward pass through network
Challenges with Traditional Recommender Systems

1. Incorporating user context & session-based data in real-time

2. Leveraging all user and item information to improve accuracy and handle cold start

So far we've only been talking about interaction data!
user_i

interactions $\rightarrow f_R(\cdot, \theta)$

$\vdots$

content_{1} $\rightarrow f_1(\cdot, \theta)$

$\vdots$

content_{n} $\rightarrow f_n(\cdot, \theta)$

$\rightarrow f(\cdot, \theta)$

$S_{ij} = U_i \times V_j$

item_j

interactions $\rightarrow f_R(\cdot, \theta)$

$\vdots$

content_{1} $\rightarrow f_1(\cdot, \theta)$

$\vdots$

content_{n} $\rightarrow f_n(\cdot, \theta)$

$\rightarrow f(\cdot, \theta)$
### Illustrative Example

<table>
<thead>
<tr>
<th>Users</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Interactions:
- $\text{Item}_{10} = [0,0,0,0,0,0,0,0,0,0]$
Illustrative Example

Item_{10}
- Interactions: [0,0,0,0,0,0,0,0,0,0]
- Content: [0.5, 0, 0.1, 0.9, 0, 0.1]
Illustrative Example

Item_{10}
- Interactions: [0,0,0,0,0,0,0,0,0,0]
- Content: [0.5, 0, 0.1, 0.9, 0, 0.1]
**Illustrative Example**

![Matrix Diagram](image)

<table>
<thead>
<tr>
<th>Users</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="User Matrix" /></td>
<td><img src="image" alt="Items Matrix" /></td>
</tr>
</tbody>
</table>

**Item_{10}**
- Interactions: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- Content: [0.5, 0, 0.1, 0.9, 0, 0.1]

**User_{2}**
- Interactions: [1, 2, 1, 0, 0, 0, 2, 0, 0, 1]
- Content: [0.2, 0, 0.7, 0.5, 0, 1.0]
Our Approach: Cold Start

• Using latest advances in deep learning we can apply this framework to virtually any content type:
  – Text: doc2vec, recurrent nets
  – Images: convolutional nets
  – Acoustic: recurrent nets, acoustic deep nets
Our Approach: Cold Start

- Using latest advances in deep learning we can apply this framework to virtually any content type:
  - Text: doc2vec, recurrent nets
  - Images: convolutional nets
  - Acoustic: recurrent nets, acoustic deep nets

- Training is done with stochastic dropout on inputs to condition the model for missing data
Our Approach: Cold Start

- Using latest advances in deep learning we can apply this framework to virtually any content type:
  - Text: doc2vec, recurrent nets
  - Images: convolutional nets
  - Acoustic: recurrent nets, acoustic deep nets
- Training is done with stochastic dropout on inputs to condition the model for missing data
- For complex structured input (images, audio etc.) we can use pre-trained models and optimize only the last few layers
Experiments

- MovieLens
- CiteULike
- Job Listings
Experiments: MovieLens

- Largest of the MovieLens data sets
- Data statistics:
  - 71,567 users
  - 10,681 items
  - 10M user-item interactions
- Partition the data to use 10%, 30%, 50% and 70% of ratings for training and the rest for test
Experiments: MovieLens

- Model architecture: 3 layers to compress 70K input to 20
Experiments: MovieLens

- Model architecture: 3 layers to compress 70K input to 20
Experiments: MovieLens

- Compare with item-based neighbor approach and PMF (same objective)
Experiments: MovieLens

- Continuous inference: use 25%, 50%, 75% and 100% of training ratings to infer $U$ for a held out subset of 1,000 users
Experiments: MovieLens

- Continuous inference: use 25%, 50%, 75% and 100% of training ratings to infer $\mathbf{U}$ for a held out subset of 1,000 users

<table>
<thead>
<tr>
<th>training set size</th>
<th>30%</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>0.7169 (-2.6%)</td>
<td>0.7773 (-0.7%)</td>
</tr>
<tr>
<td>50%</td>
<td>0.7332 (-1.3%)</td>
<td>0.8017 (-0.5%)</td>
</tr>
<tr>
<td>75%</td>
<td>0.7416 (-0.7%)</td>
<td>0.8105 (-0.4%)</td>
</tr>
<tr>
<td>100%</td>
<td><strong>0.7478 (-0.3%)</strong></td>
<td><strong>0.8150 (-0.3%)</strong></td>
</tr>
</tbody>
</table>
Experiments: MovieLens

- Continuous inference: use 25%, 50%, 75% and 100% of training ratings to infer $\mathbf{U}$ for a held out subset of 1,000 users.
Experiments: MovieLens

- Continuous inference: use 25%, 50%, 75% and 100% of training ratings to infer $\mathbf{U}$ for a held out subset of 1,000 users

<table>
<thead>
<tr>
<th>training set size</th>
<th>30%</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>0.7169 (-2.6%)</td>
<td>0.7773 (-0.7%)</td>
</tr>
<tr>
<td>50%</td>
<td>0.7332 (-1.3%)</td>
<td>0.8017 (-0.5%)</td>
</tr>
<tr>
<td>75%</td>
<td>0.7416 (-0.7%)</td>
<td>0.8105 (-0.4%)</td>
</tr>
<tr>
<td>100%</td>
<td><strong>0.7478 (-0.3%)</strong></td>
<td><strong>0.8150 (-0.3%)</strong></td>
</tr>
</tbody>
</table>

- Forward pass takes fraction of a second so we can use it to continuously refine user and item representations

represent. improve with more data
Experiments:CiteULike

- Users create article libraries for future reference
- Goal is to use these libraries + article content to suggest new articles
- Data statistics:
  - 5,551 users
  - 16,980 items
  - 204,986 binary user-item interactions
  - 8,000-dimensional bag of words representations for each article (from Wang & Blei [1])

Experiments: CiteULike

- Model architecture: 4 layers to compress [5K, 8K] input to 200
- Interactions are “dropped out” with probability 0.5 for cold start
Experiments:CiteULike

- Primarily compare with CTR - combination of LDA and WMF approaches [1]
- CTR was recently adopted by the NY Times as their production model for article recommendation [2]

Experiments: CiteULike

- Weak generalization and cold start results

![Graph showing recall@200 for weak generalization and cold start results with different algorithms: WMF, LDA, CTR, DNN-c, and DNN.]
Experiments: CiteULike

- Weak generalization and cold start results

![Graph showing comparable performance on weak generalization](image)
Experiments: CiteULike

- Weak generalization and cold start results

![Bar chart showing recall at 200 for weak generalization and cold start.]

Over 8% gain on cold start
Experiments: Job Recommendations

- Commercial site specializing in connecting users with relevant jobs
- Dataset statistics: 4 months of data from one region (city)
  - 728K users
  - 780K jobs
  - 2.3M user-job applications, 10.5M user-job views
- Over 500x sparser than the Netflix dataset!
- Jobs have lots of content info: category, career level, company stats, description, industry etc.
Experiments: Job Recommendations

- Model architecture: 3 layers to compress ~1M input with 11 feature categories to 200-2,000 dimensions
Experiments: Job Recommendations

- Compare against client's internal baseline used in production
- Evaluate on job views and applications forward in time
Experiments: Job Recommendations

- Compare against client's internal baseline used in production
- Evaluate on job views and applications forward in time
Experiments: Job Recommendations

- Compare against client's internal baseline used in production
- Evaluate on job views and applications forward in time

Over 30% gain on both views and applications
Experiments: Job Recommendations

- Compare against client's internal baseline used in production
- Evaluate on job views and applications forward in time

Increasing rank makes the model more accurate.
Thank You!

- **Clients**: on demand deep learning service for personalization and search
- **Jobs**: hiring data scientists in Toronto
- **Contact**: tomi@layer6.ai