Deep Learning Based Recommendation Systems

Prof. Srijan Kumar
http://cc.gatech.edu/~srijan
Today’s Lecture

- Introduction
- Neural Collaborative Filtering
- RRN
- LatentCross
- JODIE

Deep Recommender Systems

• How can deep learning advance recommendation systems?
• Simple way for content-based models: Use CNNs, LSTMs for generate image and text features of items
Deep Recommender Systems

- But how can DL be used for tasks and methods at the core of recommendation systems?
  - For collaborative filtering?
  - For latent factor models?
  - For temporal dynamics?
  - Some new techniques?
Why Deep Learning Techniques

Pros:
- Capture non-linearity well
- Non-manual representation learning
- Efficient sequence modeling
- Somewhat flexible and easy to retrain

Cons:
- Lack of interpretability
- Large data requirements
- Extensive hyper-parameter tuning
Applicable DL Techniques

Deep Learning methods:
- MLPs and AutoEncoders
- CNNs
- RNNs
- Adversarial Networks
- Attention models
- Deep reinforcement learning

How to use these methods to improve recommender systems?
Today’s Lecture

• Introduction
• Neural Collaborative Filtering
• Recurrent Recommender Networks
• LatentCross
• JODIE

Matrix Factorization

• MF uses an **inner product as the interaction function**
  – Latent factors are **independent** with each other

• **Limitations:** The simple choice of **inner product** function can limit the expressiveness of a MF model.

• **Potential solution:** increase the number of factors. However,
  – This increases the complexity of the model
  – Leads to overfitting
Improving Matrix Factorization

• **Key question:** How can we improve matrix factorization?

• **Answer:** Learn the relation between factors from the data, rather than fixing it to be the simple, fixed inner product
  – Does not increase the complexity
  – Does not lead to overfitting

• **One solution:** Neural Collaborative Filtering
Neural Collaborative Filtering

- **Neural Collaborative Filtering (NCF)** is a deep learning version of the traditional recommender system
- **Learns the interaction function** with a deep neural network
  - Non-linear functions, e.g., multi-layer perceptrons, to learn the interaction function
  - Models well when **latent factors are not independent** with each other, especially true in large real datasets
Neural Collaborative Filtering

- Neural extensions of traditional recommender system

- **Input**: rating matrix, user profile and item features (optional)
  - If user/item features are unavailable, we can use one-hot vectors

- **Output**: User and item embeddings, prediction scores

- Traditional matrix factorization is a special case of NCF
NCF Setup

- User feature vector: $s^\text{user}_u$
- Item feature vector: $s^\text{item}_i$
- User embedding matrix: $U$
- Item embedding matrix: $I$
- Neural network: $f$
- Neural network parameters: $\Theta$
- Predicted rating:

$$\hat{r}_{ui} = f(U^T \cdot s^\text{user}_u, V^T \cdot s^\text{item}_i | U, V, \Theta)$$
NCF Model Architecture

- **Multiple layers of fully connected layers** form the Neural CF layer.
- **Output is a rating score** $\hat{r}_{ui}$
- **Real rating score** is $r_{ui}$
1-Layer NCF

- **Layer 1** an element-wise product
- **Output Layer** as a fully connected layer without bias
Multi-Layer NCF

• Each layer is a **multi-layer perceptron**, with non-linearity on the top
• Final score is used to **calculate the loss and train the layers**
NCF model: Loss function

- **Train** on the difference between predicted rating and the real rating
- **Use negative sampling** to reduce the negative data points
- **Loss = cross-entropy loss**

\[
\mathcal{L} = - \sum_{(u,i) \in \mathcal{O} \cup \mathcal{O}^-} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log (1 - \hat{r}_{ui})
\]
Experimental Setup

• **Two public datasets:** MovieLens, Pinterest
  – Transform MovieLens ratings to 0/1 implicit case

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Interaction#</th>
<th>Item#</th>
<th>User#</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>1,000,209</td>
<td>3,706</td>
<td>6,040</td>
<td>95.53%</td>
</tr>
<tr>
<td>Pinterest</td>
<td>1,500,809</td>
<td>9,916</td>
<td>55,187</td>
<td>99.73%</td>
</tr>
</tbody>
</table>

• **Evaluation protocols:**
  – **Leave-one-out setting:** hold-out the latest rating of each user as the test
  – **Top-k evaluation:** create a ranked list of items
  – **Evaluation metrics:**
    • **Hit Ratio:** does the correct item appear in top 10
Baselines

• **Item Popularity**
  – Items are ranked by their popularity

• **ItemKNN** [Sarwar et al, WWW’01]
  – The standard item-based CF method

• **BPR** [Rendle et al, UAI’09]
  – Bayesian Personalized Ranking optimizes MF model with a pairwise ranking loss

• **eALS** [He et al, SIGIR’16]
Performance vs. Embedding Size

- NeuMF > eALS and BPR (5% improvement)
- NeuMF > MLP (MLP has lower training loss but higher test loss)
Convergence Behavior

- Most effective updates in the **first 10 iterations**
- More iterations make **NeuMF overfit**
- **Trade-off** between representation ability and generalization ability of a model.
Is Deeper Helpful?

Table 4: NDCG@10 of MLP with different layers.

<table>
<thead>
<tr>
<th>Factors</th>
<th>MLP-0</th>
<th>MLP-1</th>
<th>MLP-2</th>
<th>MLP-3</th>
<th>MLP-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.253</td>
<td>0.359</td>
<td>0.383</td>
<td>0.399</td>
<td>0.406</td>
</tr>
<tr>
<td>16</td>
<td>0.252</td>
<td>0.391</td>
<td>0.402</td>
<td>0.410</td>
<td>0.415</td>
</tr>
<tr>
<td>32</td>
<td>0.252</td>
<td>0.406</td>
<td>0.410</td>
<td>0.425</td>
<td>0.423</td>
</tr>
<tr>
<td>64</td>
<td>0.251</td>
<td>0.409</td>
<td>0.417</td>
<td>0.426</td>
<td>0.432</td>
</tr>
<tr>
<td>Pinterest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.141</td>
<td>0.526</td>
<td>0.534</td>
<td>0.536</td>
<td>0.539</td>
</tr>
<tr>
<td>16</td>
<td>0.141</td>
<td>0.532</td>
<td>0.536</td>
<td>0.538</td>
<td>0.544</td>
</tr>
<tr>
<td>32</td>
<td>0.142</td>
<td>0.537</td>
<td>0.538</td>
<td>0.542</td>
<td>0.546</td>
</tr>
<tr>
<td>64</td>
<td>0.141</td>
<td>0.538</td>
<td>0.542</td>
<td>0.545</td>
<td>0.550</td>
</tr>
</tbody>
</table>

- Same number of factors, but more nonlinear layers improves the performance.
- Linear layers degrades the performance.
- Improvement diminishes for more layers.
NCF: Shortcomings

• Architecture is **limited**

• NCF **does not model the temporal behavior** of users or items
  – Recall: users and items exhibit temporal bias
  – NCF has the same input for user

• **Non-inductive**: new users and new items, on which training was not done, can not be processed
Today’s Lecture

- Introduction
- Neural Collaborative Filtering
- RRN
- LatentCross
- JODIE
**RRN**

- **RRN = Recurrent Recommender Networks**
- One of the first methods to model the temporal evolution of user and item behavior

- **Reference paper:** Recurrent Recommender Networks. CY Wu, A Ahmed, A Beutel, A Smola, H Jing. WSDM 2017
Traditional Methods

- **Existing models assume user and item states are stationary**
  - States = embeddings, hidden factors, representations
- **However, user preferences and item states change over time**
- **How to model this?**
- **Key idea:** use of RNNs to learn evolution of user embeddings
User Preferences

• User preference changes over time

10 years ago

now
Item States

- Movie reception changes over time

So bad that it’s great to watch

Bad movie
Exogenous Effects

“La La Land” won big at Golden Globes
Seasonal Effects

Only watch during Christmas
Traditional Methods

- Traditional matrix factorization, including NCF, assumes **user state** $u_i$ and **item state** $m_j$ are **fixed and independent** of each other.
- Use both to make predictions about the rating score $r_{ij}$.
- Right figure: **latent variable block diagram of traditional MF**.
RRN Framework

• RRN innovates by **modeling temporal dynamics** within each user state $u_i$ and movie state $m_j$

• $u_{it}$ depends on $u_{it-}$ and influences $u_{it+}$
  – Same for movies

• User and item states are independent of each other
Model Learning Setting

• Actions are happening over time
• How to split training and testing data to respect the time dependency?
Traditional Random Split: N/A

- Random train/test split violates the temporal dependency
  - Future actions can be in train, while past actions can be in test
Realistic Learning Setting

- Train on first $K\%$ data and test in the last data points
• Train two RNNs: one for all users and other for all movies
  – User RNN parameters are shared across all users; same for movies
RRN Process

- **Initialization**: User and movie embeddings are initialized
  - Initialization can be one-hot
- **Embedding update**
- **Prediction**: To predict the rating a user gives to a movie, the user’s embedding is multiplied with the movie’s embedding
- **Loss**: User-movie rating score prediction error is used to update the RNN parameters
User RNN

• User RNN takes a user’s (movie, rating) sequence
  – Each input: concatenation of movie embedding and one-hot vector of rating score
  – RNN initialization: special ‘new’ vector to indicate a new user

• For the next user, the process is repeated, starting from initialization
Movie RNN

• Movie RNN takes the movie’s \((\text{user}, \text{rating})\) sequence
  – Each input: concatenation of user embedding and one-hot vector of rating score
  – RNN initialization: special ‘new’ vector to indicate a new movie

• For the next movie, the process is repeated, starting from initialization
Rating Prediction

• What is the rating by a $u_i$ to $m_j$ at time $t$?
• Take the user and movie embedding till time $t$ and output rating $\hat{r}_{ij|t} = f(u_{it}, m_{jt})$
• **Output function**: MLP, Hadamard product, etc.
Model Training

• Learn the model parameters $\theta$ such that the predicted rating is close to the actual rating

• $R(\theta)$ is a regularization term to avoid overfitting

$$\min_{\theta} \sum_{(i,j,t) \in \mathcal{I}_{\text{train}}} (r_{ij|t} - \hat{r}_{ij|t}(\theta))^2 + R(\theta)$$
Experiments

- **Three datasets, several baselines**
  - PMF: Salakhutdinov & Mnih NIPS ’07
  - T-STD: Koren KDD ’09
  - U-AR & I-AR: Sedhain et al. WWW ‘15

- **Metric = RMSE (Root Mean Square Error)**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PMF</th>
<th>I-AR</th>
<th>U-AR</th>
<th>T-SVD</th>
<th>RRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDb</td>
<td>2.3913</td>
<td>2.0521</td>
<td>2.0290</td>
<td>2.0037</td>
<td>1.9703</td>
</tr>
<tr>
<td>Netflix 6 months</td>
<td>0.9584</td>
<td>0.9778</td>
<td>0.9836</td>
<td>0.9589</td>
<td>0.9427</td>
</tr>
<tr>
<td>Netflix full</td>
<td>0.9252</td>
<td>0.9364</td>
<td>0.9647</td>
<td>0.9275</td>
<td>0.9224</td>
</tr>
</tbody>
</table>
Temporal Effects

• How well does the model capture the temporal effects?
Exogenous Effects

• RRN automatically captures the exogenous effects

Oscar & Golden globe
System Effects

- RRN automatically learns the system effects

Netflix changed the Likert scale
Movie Age Effect

- RRN automatically learns effects that we typically capture via hand-crafted features

![Graph showing predicted rating against movie age](image-url)
### RRN Summary

#### Novel model

- \( \{\text{new}\} \rightarrow \ldots \rightarrow \hat{y}_{i,t-2} \rightarrow \hat{y}_{i,t-1} \rightarrow \hat{u}_i \)

- \( \{\text{new}\} \rightarrow \ldots \rightarrow \hat{y}_{j,t-3} \rightarrow \hat{y}_{j,t-2} \rightarrow \hat{y}_{j,t-1} \rightarrow \hat{m}_j \)

#### Future prediction

![Future prediction images](image)

#### Accurate prediction

<table>
<thead>
<tr>
<th></th>
<th>PMF</th>
<th>I-AR</th>
<th>U-AR</th>
<th>T-SVD</th>
<th>RRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDb</td>
<td>2.3913</td>
<td>2.0521</td>
<td>2.0290</td>
<td>2.0037</td>
<td><strong>1.9703</strong></td>
</tr>
<tr>
<td>Netflix 6 months</td>
<td>0.9584</td>
<td>0.9778</td>
<td>0.9836</td>
<td>0.9589</td>
<td><strong>0.9427</strong></td>
</tr>
<tr>
<td>Netflix full</td>
<td>0.9252</td>
<td>0.9364</td>
<td>0.9647</td>
<td>0.9275</td>
<td><strong>0.9224</strong></td>
</tr>
</tbody>
</table>

#### Temporal dynamics

![Temporal dynamics chart](chart)
Next Lecture

- Introduction
- Neural Collaborative Filtering
- RRN
- LatentCross
- JODIE