ALTRec: Adversarial Learning for Autoencoder-based Tail Recommendation

Jixiong Liu\textsuperscript{1, 2}, Dugang Liu\textsuperscript{1, 2}, WeiKe Pan\textsuperscript{1, 2*} and Zhong Ming\textsuperscript{1, 2*}

liujixiong@email.szu.edu.cn, dugang.ldg@gmail.com, panweike@szu.edu.cn, mingz@szu.edu.cn

\textsuperscript{1}College of Computer Science and Software Engineering
Shenzhen University, Shenzhen, China

\textsuperscript{2}Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China
Long-Tail Items Recommendation

- Tail items: items with less popularity, usually the least popular part of an item list w.r.t. the popularity.
- Input: Observations in the form of (user, item) pairs.
- Goal: Generate a personalized ranked list of items from long-tail items $I_t$ for each user $u$ from the set of items that user $u$ has not observed before, i.e., $i \in I_t \setminus I_u$, where $I_t$ and $I_u$ denote the set of long-tail items and user $u$’s interacted ones, respectively.
Motivation (1/2)

- Previous auto-encoder models often emphasize the importance of the head items in training, resulting in insufficient training on the tail ones.

**Figure**: Average predicted preferences of AutoRec on decile items with different popularity, i.e., from the most popular segment (s1) to the least popular one (s10), of MovieLens 1M.
Motivation (2/2)

The preference relationship between any two users should keep unchanged. However, a small number of inaccurate predicted preferences on tail items can not influence the similarity a lot.

Figure: An example of the input and prediction vectors on three different users, where values with red and black correspond to the head and tail items, respectively. Obviously, user 2 and user 3 have high similarity in both stages even if there are large prediction difference on tail items.
Illustration of our ALTRec, where the generator (G) aims to reconstruct the input and minimize the similarity difference (of two users) between the input stage and the output stage, and the discriminator (D) is a neural network that maps the inputs from two different stages to a same semantic space and maximizes the similarity difference.
Advantages of Our Solution

1. We introduce a similarity constraint to retain the two-user similarity at the input stage and output stage, in order to keep users’ interaction relationships.

2. We introduce an adversarial strategy to help a small number of inaccurate predicted preferences on tail items influence the similarity a lot.
## Notations

**Table**: Some notations and explanations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{U}$</td>
<td>user set, $u \in \mathcal{U}$</td>
</tr>
<tr>
<td>$\mathcal{I}$</td>
<td>item set, $i \in \mathcal{I}$</td>
</tr>
<tr>
<td>$n$</td>
<td>number of users</td>
</tr>
<tr>
<td>$m$</td>
<td>number of items</td>
</tr>
<tr>
<td>$\mathcal{U}_a$</td>
<td>active users set, $u \in \mathcal{U}$</td>
</tr>
<tr>
<td>$\mathcal{I}_p$</td>
<td>popular/head items set, $i \in \mathcal{I}_p$</td>
</tr>
<tr>
<td>$\mathcal{I}_t$</td>
<td>tail items set, $\mathcal{I}_p \cup \mathcal{I}_t = \mathcal{I}$</td>
</tr>
<tr>
<td>$X_u$</td>
<td>user $u$’s interacted history vector</td>
</tr>
<tr>
<td>$\hat{X}_u$</td>
<td>user $u$’s predicted preferences on all items</td>
</tr>
<tr>
<td>$\alpha$, $\lambda_g$, $\lambda_d$</td>
<td>some hyper-parameters</td>
</tr>
<tr>
<td>$\theta$</td>
<td>generator’s parameters</td>
</tr>
<tr>
<td>$\phi$</td>
<td>discriminator’s parameters</td>
</tr>
</tbody>
</table>
Overall Objective

In our ALTRec, the generator and discriminator play a min-max game as follows,

$$\min_{\theta} \max_{\phi} \mathbb{E}_{u,v \in U} f(Z_t, Z_f),$$

(1)

where $\theta$ and $\phi$ denote the parameters of the generator and the discriminator, respectively. And $f(Z_t, Z_f)$ denotes the similarity difference of any two users, in which $Z_t$ and $Z_f$ represent users’ similarity at input stage and prediction stage.
Generative Training (1/3)

The generator aims to minimize the reconstruction loss and keep users’ interaction relationships unchanged,

\[
\theta^* = \arg\min_{\theta} - \mathbb{E}_{u \in U} [X_u \log(\hat{X}_u) + (1 - X_u) \log(1 - \hat{X}_u)] + \lambda_g R(\theta) + \alpha \mathbb{E}_{u, v \in U} f(Z_t, Z_f),
\]

(2)

where, the first and second term denote the reconstruction loss and regularization term inheriting from AutoRec [Sedhain et al., 2015]. And the last term is the adversarial loss to minimize any two users’ similarity difference in order to keep user interaction relationship unchanged. Notice that \(\lambda_g\) and \(\alpha\) are hyper-parameters used to control the importance.
Generative Training (2/3)

Particularly, the adversarial loss for the generator can be formalized as follows,

\[
\mathbb{E}_{u,v \in U} f(Z_t, Z_f) \\
\approx \frac{1}{n} \sum_{u \in U, v \in U} f(D(X_u \oplus X_v), D(\hat{X}_u^M \oplus \hat{X}_v^M)) \\
= \frac{1}{n} \sum_{u \in U, v \in U} (D(X_u \oplus X_v) - D(\hat{X}_u^M \oplus \hat{X}_v^M))^2,
\]

where we take the concatenation vector \(X_u \oplus X_v\) (or \(\hat{X}_u^M \oplus \hat{X}_v^M\)) as the discriminator’s input, and then obtain a mapped vector \(Z_t\) (or \(Z_f\)), representing the similarity of user \(u\) and user \(v\). Notice that we set \(\hat{X}_u^M = \hat{X}_u \cdot X_u\) and \(\hat{X}_v^M = \hat{X}_v \cdot X_v\) to eliminate the impact from non-ground truth items.
Generative Training (3/3)

We use a variant of Euclidean distance to measure the similarity difference,

\[ E_{u,v \in U} f(Z_t, Z_f) \approx \frac{1}{n} \sum_{u \in U_b, v \in U} f(D(X_u \oplus X_v), D(\hat{X}_u^M \oplus \hat{X}_v^M)) \]

\[ = \frac{1}{n} \sum_{u \in U_b, v \in U} (D(X_u \oplus X_v) - D(\hat{X}_u^M \oplus \hat{X}_v^M))^2, \quad (4) \]

To avoid the influence from repeated training on active users set, we use a stop gradient strategy\(^1\) that stops gradient from adversarial loss on \( u \in U_b \) from being back-propagated to the generator so as to treat all the users fairly.

\(^1\)https://www.tensorflow.org/api_docs/python/tf/stop_gradient
Method

Discriminative Training (1/2)

As an adversary, the discriminator aims to maximize the similarity difference between any two users,

\[
\phi^* = \arg \min \phi - \mathbb{E}_{u,v \in U} f(Z_t, Z_f) + \lambda_d \mathbb{E}_{\tilde{X} \in \mathbb{P}_{\tilde{X}} \left[ (\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1)^2 \right]}
\]

Gradient penalty

\[
\approx \arg \min \phi - \frac{1}{n} \sum_{u \in U_b, v \in U} (D(X_u \oplus X_v) - D(\hat{X}_u^M \oplus \hat{X}_v^M))^2
\]

\[
+ \lambda_d \mathbb{E}_{\tilde{X} \in \mathbb{P}_{\tilde{X}}} \left[ (\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1)^2 \right],
\]

where the discriminator takes the concatenation vectors \(X_u \oplus X_v\) and \(\hat{X}_u^M \oplus \hat{X}_v^M\) as input, and maps them to a same semantic space to represent users’ similarities.
Discriminative Training (2/2)

Particularly, the adversarial loss for the discriminator can be formalized as follows,

\[
\phi^* = \arg\min_{\phi} - \mathbb{E}_{u,v \in U} f(Z_t, Z_f) + \lambda_d \mathbb{E}_{\tilde{X} \in \mathcal{P}_{\tilde{X}}} [(\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1)^2]
\]

where the first term is to maximize the similarity difference between the two stages. And the second term is to enforce the Lipschitz constraint [Gulrajani et al., 2017] to generate more stable gradients.

\[
\approx \arg\min_{\phi} - \frac{1}{n} \sum_{u \in U_b, v \in U} (D(X_u \oplus X_v) - D(\hat{X}_u^M \oplus \hat{X}_v^M))^2
\]

\[
+ \lambda_d \mathbb{E}_{\tilde{X} \in \mathcal{P}_{\tilde{X}}} [(\|\nabla_{\tilde{X}} D(\tilde{X})\|_2 - 1)^2],
\]

(6)
The algorithm of our ALTRec.

1: Pre-train the generator $G$ and the discriminator $D$.
2: for $t = 1, \ldots, T$ do
3:     for d-steps do
4:         Discriminative training.
5:     end for
6:     for g-steps do
7:         Generative training.
8:     end for
9: end for
AlGORITHM (2/3)

The algorithm of the discriminative training.

1: All users set $\mathcal{U}$ in reverse active order.
2: Select the most active users $\mathcal{U}_a$ and shuffle them.
3: Repeat $\mathcal{U}_a$ to obtain $\mathcal{U}_b$ until $|\mathcal{U}_b| = |\mathcal{U}|$
4: for $u \in \mathcal{U}_b$, $v \in \mathcal{U}$ do
5: Obtain true preference vectors $X_u$ and $X_v$.
6: Obtain predicted preferences $\hat{X}_u^M$ and $\hat{X}_v^M$.
7: Sample $\epsilon \in U(0, 1)$.
8: Obtain $\tilde{X} = \epsilon(X_u \oplus X_v) + (1-\epsilon)(\hat{X}_u^M \oplus \hat{X}_v^M)$.
9: Optimize $\phi$ by minimizing Eq.(6).
10: end for
Algorithm (3/3)

The algorithm of the generative training.

1: All users set $\mathcal{U}$ in reverse active order.
2: Shuffle $\mathcal{U}_a$.
3: Repeat $\mathcal{U}_a$ to obtain $\mathcal{U}_b$ until $|\mathcal{U}_b| = |\mathcal{U}|$
4: for $u \in \mathcal{U}_b$, $v \in \mathcal{U}$ do
5: Obtain true preference vectors $X_u$ and $X_v$.
6: Obtain $\hat{X}_u$, $\hat{X}_v$, $\hat{X}_u^M$ and $\hat{X}_v^M$.
7: Optimize $\theta$ by minimizing Eq.(2).
8: end for
Research Questions:

- **RQ1**: Does our ALTRec outperform the state-of-the-art recommendation methods for recommendation of tail items?
- **RQ2**: How effective are the adversarial strategies we propose in ALTRec?
- **RQ3**: How does our ALTRec perform on users with different levels of activity?
- ...
## Datasets

**Table: Statistics of the datasets used in the experiments.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User #</th>
<th>Item #</th>
<th>Interaction #</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens 100K</td>
<td>943</td>
<td>1,605</td>
<td>99,894</td>
<td>93.40%</td>
</tr>
<tr>
<td>MovieLens 1M</td>
<td>6,040</td>
<td>3,648</td>
<td>1,000,117</td>
<td>95.46%</td>
</tr>
<tr>
<td>Anime</td>
<td>69,600</td>
<td>9,475</td>
<td>6,336,619</td>
<td>99.04%</td>
</tr>
</tbody>
</table>
Baselines

- BPR-MF (factorization-based method) [Rendle et al., 2009]
- FISM (factorization-based method) [Kabbur et al., 2013]
- AutoRec (autoencoder-based method) [Sedhain et al., 2015]
- Mult-VAE (autoencoder-based method) [Liang et al., 2018]
- CFGAN (GAN-based recommendation method) [Chae et al., 2018]
- LongTailGAN (GAN-based recommendation method) [Krishnan et al., 2018]
For BPR and FISM, we use the SGD optimizer and tune the regularization coefficient and learning rate from \{1e-4, 1e-3, 1e-2, 1e-1\}, and the dimension of latent feature vectors from \{50, 150, 250, 350\}, independently.

For autoencoder-based methods and the generator of adversarial methods, we fix the dimension of the hidden layer as 350.

For AutoRec and Mult-VAE, we select the regularization coefficient from \{1e-4, 1e-3, 1e-2, 1e-1\}. 
Parameter Configurations (2/2)

- For LongTailGAN, we select the dropout rate for the discriminator and the generator from \{0.1, 0.2, 0.3, 0.4, 0.5\} independently, the coefficient of the adversarial term in the generator from \{1e-2, 1e-1, 1, 10, 50, 100\}.

- For CFGAN, we use the ZP strategy and tune the sample ratio of ZR from \{10\%, 30\%, 50\%\}, the sample ratio of PM from \{50\%, 70\%, 90\%\}, the coefficient of zero-reconstruction loss from \{1e-2, 1e-1, 1\}, and the regularization coefficient for the generator and the discriminator from \{1e-4, 1e-3, 1e-2, 1e-1\} independently.

- For our ALTRec, we tune the coefficient of the gradient penalty $\lambda_d$ from \{1e-1, 1, 1e1, 1e2\}, the regularization coefficient $\lambda_g$ in the generator from \{1e-4, 1e-3, 1e-2, 1e-1\}, the coefficient of the adversarial term $\alpha$ in the generator from \{1e-2, 1e-1, 1, 10, 50, 100\}, and $|U_a|$ from \{100, 300, 500\}. 
We take the top 10% and 20% most popular items as head items and the rest as the tail ones, and show the recommendation performance via the following metrics,

- Precision@K
- Recall@K
- NDCGK
- MRR@K
RQ1: Performance Comparison (1/2)

Table: Recommendation performance on tail items (by removing the top 10% and 20% most popular items) on the test data Anime. Similar results can be found in ML100K and ML1M.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tail items</th>
<th>Method</th>
<th>P@5</th>
<th>P@20</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
<th>M@5</th>
<th>M@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anime</td>
<td>w/o top 10%</td>
<td>BPR-MF</td>
<td>0.0310</td>
<td>0.0248</td>
<td>0.0273</td>
<td>0.0822</td>
<td>0.0374</td>
<td>0.0534</td>
<td>0.0670</td>
<td>0.0841</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FISM</td>
<td>0.0521</td>
<td>0.0351</td>
<td>0.0544</td>
<td>0.1130</td>
<td>0.0710</td>
<td>0.0853</td>
<td>0.1162</td>
<td>0.1325</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CFGAN</td>
<td>0.0024</td>
<td>0.0009</td>
<td>0.0019</td>
<td>0.0023</td>
<td>0.0039</td>
<td>0.0029</td>
<td>0.0090</td>
<td>0.0094</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LongTailGAN</td>
<td>0.0443</td>
<td>0.0318</td>
<td>0.0419</td>
<td>0.0986</td>
<td>0.0562</td>
<td>0.0706</td>
<td>0.0912</td>
<td>0.1074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AutoRec</td>
<td>0.0518</td>
<td>0.0364</td>
<td>0.0721</td>
<td>0.1594</td>
<td>0.0753</td>
<td>0.1033</td>
<td>0.1158</td>
<td>0.1385</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mult-VAE</td>
<td>0.0686</td>
<td>0.0423</td>
<td>0.0769</td>
<td>0.1510</td>
<td>0.0964</td>
<td>0.1134</td>
<td>0.1574</td>
<td>0.1759</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTRRec</td>
<td><strong>0.0901</strong></td>
<td><strong>0.0590</strong></td>
<td><strong>0.0923</strong></td>
<td><strong>0.2052</strong></td>
<td><strong>0.1188</strong></td>
<td><strong>0.1462</strong></td>
<td><strong>0.1920</strong></td>
<td><strong>0.2160</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imp.(%)</td>
<td>31.34%</td>
<td>39.48%</td>
<td>20.03%</td>
<td>35.89%</td>
<td>23.24%</td>
<td>28.92%</td>
<td>21.98%</td>
<td>22.80%</td>
</tr>
<tr>
<td>Anime</td>
<td>w/o top 20%</td>
<td>BPR-MF</td>
<td>0.0078</td>
<td>0.0078</td>
<td>0.0093</td>
<td>0.0370</td>
<td>0.0097</td>
<td>0.0190</td>
<td>0.0160</td>
<td>0.0236</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FISM</td>
<td>0.0177</td>
<td>0.0131</td>
<td>0.0253</td>
<td>0.0590</td>
<td>0.0261</td>
<td>0.0365</td>
<td>0.0387</td>
<td>0.0473</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CFGAN</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LongTailGAN</td>
<td>0.0138</td>
<td>0.0115</td>
<td>0.0168</td>
<td>0.0468</td>
<td>0.0179</td>
<td>0.0275</td>
<td>0.0273</td>
<td>0.0357</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AutoRec</td>
<td>0.0129</td>
<td>0.0113</td>
<td>0.0251</td>
<td>0.0722</td>
<td>0.0208</td>
<td>0.0368</td>
<td>0.0281</td>
<td>0.0386</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mult-VAE</td>
<td>0.0231</td>
<td>0.0162</td>
<td>0.0367</td>
<td>0.0851</td>
<td>0.0361</td>
<td>0.0514</td>
<td>0.0535</td>
<td>0.0651</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALTRRec</td>
<td><strong>0.0389</strong></td>
<td><strong>0.0283</strong></td>
<td><strong>0.0555</strong></td>
<td><strong>0.1420</strong></td>
<td><strong>0.0550</strong></td>
<td><strong>0.0819</strong></td>
<td><strong>0.0818</strong></td>
<td><strong>0.1000</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imp.(%)</td>
<td>68.40%</td>
<td>74.69%</td>
<td>51.23%</td>
<td>66.86%</td>
<td>52.35%</td>
<td>59.34%</td>
<td>52.90%</td>
<td>53.61%</td>
</tr>
</tbody>
</table>
RQ1: Performance Comparison (2/2)

Observations

- The proposed ALTRec algorithm performs significantly best in all cases. Such promising results clearly show the effectiveness of our ALTRec in addressing the long-tail items recommendation problem.

- From the perspective of modeling techniques, we assume that keeping users’ interaction relationships unchanged can help recommend tail items well. And we adopt an adversarial strategy to make inaccurate prediction on tail items influence a lot to the similarity difference. The effectiveness also verifies the effectiveness of the proposed similarity constraint and adversarial strategy.

- ...
RQ2: Ablation Study

**Figure:** Recommendation performance of AutoRec, SimilarityAE and our ALTRec on tail items (by removing the top 10% most popular items on the test data) for ablation study.

- **SimilarityAE**  >  **AutoRec:** Maintaining the similarity between any two users is helpful on the performance of the tail items.
- **ALTRec**  >  **SimilarityAE:** Showcasing the effectiveness of the adversarial strategy.
RQ3: Performance on Users with Different Levels of Activity

Our ALTRec and other models improve the performance more with the increase of the activity of the user segment. Our ALTRec and SimilarityAE have no improvement on the [0%, 25%] segment of Anime, because users in that segment may only interact only one item.

Figure: Tail items recommendation performance on the most inactive users segment [0%, 25%] to the most active one [75%, 100%].
Bayesian Personalized Ranking (BPR)

In BPR [Rendle et al., 2009], the users’ examination behaviors are modeled based on a so-called pairwise preference assumption, i.e., a user \(u\) prefers an interacted item \(i\) to an uninteracted one \(j\).

\[
\text{Pref}(u, i|\Phi) > \text{Pref}(u, j|\Phi), \ (u, i) \in \mathcal{E}, (u, j) \notin \mathcal{E}. \tag{7}
\]
AutoRec

AutoRec [Sedhain et al., 2015] consists of an encoder and a decoder. The encoder takes user’s past interaction behaviors to a vector as input, and the decoder generates user’s preferences on all items by minimizing the reconstruction loss.

\[ Z_u = \text{enc}(X_u), \]
\[ \hat{X}_u = \text{dec}(Z_u). \]

where \( \text{enc} \) and \( \text{dec} \) represent the encoder function and the decoder function, respectively. And \( Z_u \) is the latent representation of user \( u \), and \( \hat{X}_u \) is the user \( u \)’s predicted preferences on all items.
Mult-VAE

Different from AutoRec, Mult-VAE [Liang et al., 2018] assumes that the representation of a user should conform to a Gaussian distribution, in which the encoder is designed to learn the mean value and standard deviation, and the decoder generates the user’s preferences on all the items based on the corresponding multinomial likelihood,

\[ Z_u = \text{enc}(X_u), \]
\[ \hat{X}_u = \text{dec}(Z_u). \]
In LongTailGAN [Krishnan et al., 2018], the generator aims to reproduce the relationship between head items and tail ones. In LongTailGAN, the generator sample items with high probability with users’ interacted popular ones. And the discriminator is to distinguish the real pairs and the generated ones.

\[
\theta^*, \phi^* = \arg\min_\theta \arg\max_\phi \sum_{u \in U} \mathbb{E}(i^p, i^t) \sim p_{\text{true}}(i^p, i^t) \sigma(f_\phi(i^p, i^t)) \\
+ \mu \mathbb{E}(\tilde{i}^p, \tilde{i}^t) \sim p_\theta(\tilde{i}^p, \tilde{i}^t) \log(1 - \sigma(f_\phi(\tilde{i}^p, \tilde{i}^t)))
\] (12)

where \(\sigma(\cdot)\) represents the reward assigned from the discriminator.
Conclusions

- We have studied a long-tail items recommendation problem, which aims to provide ranking lists about tail items for users.

- We point out that previous autoencoder-based methods perform not well on tail items recommendation, which unconsciously emphasize the fitting on the head items while neglecting the tail ones due to the difference in the amount of training samples.

- We introduce a constraint to retain the two-user similarity at the input stage and the output stage of the generator, and adopt an adversarial strategy to make inaccurate predicted preferences on tail items to influence similarity difference a lot.

- Empirical results on three real-world datasets show the effectiveness of our proposed ALTRec algorithm.
Thank you!

We thank the support of National Natural Science Foundation of China Nos. 62172283 and 61836005.
CFGAN: A generic collaborative filtering framework based on generative adversarial networks.

Improved training of Wasserstein GANs.

FISM: Factored item similarity models for top-N recommender systems.
In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 659–667.

An adversarial approach to improve long-tail performance in neural collaborative filtering.

Variational autoencoders for collaborative filtering.

BPR: Bayesian personalized ranking from implicit feedback.

AutoRec: Autoencoders meet collaborative filtering.