Fast Matrix Factorization for Online Recommendation with Implicit Feedback

Xiangnan He, Hanwang Zhang, Min-Yen Kan, Tat-Seng Chua

National University of Singapore (NUS)
SIGIR 2016, July 20, Pisa, Italy
Value of Recommender System (RS)

- Netflix: 60+% of the movies watched are recommended.
- Google News: RS generates 38% more click-through.
- Amazon: 35% sales from recommendations.

Statistics come from Xavier Amatriain.
Collaborative Filtering (CF)

- **Explicit Feedback**
  - Rating prediction problem
  - Popularized by the Netflix Challenge
  - Only observed ratings are considered.
  - But, it is sub-optimal (missing-at-random assumption) for Top-K Recom. \((\text{Cremonesi and Koren}, \text{RecSys 2010})\)

- **Implicit Feedback**
  - Ranking/Classification problem
  - Aims at recommending (unconsumed) items to users.
  - Unobserved missing data (0 entries) is important!

### Real-valued Rating matrix

<table>
<thead>
<tr>
<th>users</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>

### 0/1 Interaction matrix

<table>
<thead>
<tr>
<th>users</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Technical Background & Motivation
• Popularity-aware Implicit Method
• Experiments (offline setting)
• Experiments (online setting)
• Conclusion
Matrix Factorization (MF)

- MF is a linear latent factor model:

```
  1  0  0  1
  0  1  0  0
  1  1  0  0
  1  0  0  1
```

0/1 Interaction matrix

User 'u' interacted with item 'i'

Learn latent vector for each user, item:

\[ 1 \times K \]

\[ v_u^U \]

\[ v_i^I \]

Affinity between user 'u' and item 'i':

\[ \hat{y}_{ui} = \langle v_u, v_i \rangle \]
## Previous Implicit MF Solutions

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sampling negative instances:</strong></td>
<td><strong>Treating all missing data as negative:</strong></td>
</tr>
<tr>
<td><strong>LIKELIHOOD:</strong></td>
<td><strong>LOSS:</strong></td>
</tr>
<tr>
<td>$p(.</td>
<td>\Theta) = \prod_u \prod_{i \in R_u} \prod_{j \notin R_u} \sigma(y_{ui} - \hat{y}_{ui})$</td>
</tr>
<tr>
<td>All Items bought by $u$</td>
<td>Weight for Missing data</td>
</tr>
<tr>
<td>Items not bought by $u$</td>
<td></td>
</tr>
<tr>
<td><strong>Pros:</strong></td>
<td></td>
</tr>
<tr>
<td>+ Efficient</td>
<td></td>
</tr>
<tr>
<td>+ Optimized for ranking (good precision)</td>
<td></td>
</tr>
<tr>
<td><strong>Cons:</strong></td>
<td></td>
</tr>
<tr>
<td>- Only model partial data (low recall)</td>
<td></td>
</tr>
</tbody>
</table>

**Sigmoid:**

Address the effectiveness and efficiency issue of regression method.
Drawbacks of Existing Methods

(whole-data based)
Uniform Weighting

- Limits model’s fidelity and flexibility

• Uniform weighting on missing data assumes that
  “all missing entries are equally likely to be a negative assessment.”
  – The design choice is for the optimization efficiency --- an efficient ALS algorithm
    (Hu, ICDM 2008) can be derived with uniform weighting.

• However, such an assumption is unrealistic.
  – Item popularity is typically non-uniformly distributed.
  – Popular items are more likely to be known by users.

Figures adopt from Rendle, WSDM 2014.
Low Efficiency
- Difficult to support online learning

• An analytical solution known as *ridge regression*  
  – Vector-wise ALS  
  – Time complexity: $O((M+N)K^3 + MNK^2)$  
    M: # of items, N: # of users, K: # of latent factors

• With the uniform weighting, Hu can reduce the complexity to  
  $O((M+N)K^3 + |R|K^2)$  
  |R| denotes the number of observed entries.

• However, the complexity is too high for large dataset:  
  – $K$ can be thousands for sufficient model expressiveness  
    e.g. YouTube RS, which has over billions of users and videos.

Scary complexity and unrealistic for practical usage
Importance of Online Learning for RS

- Scenario of Recommender System:
  - New data continuously streams in:
    - New users;
    - Old users have new interactions;
  - It is extremely useful to provide *instant personalization* for new users, and *refresh recommendation* for old users, but retraining the full model is expensive

=> Online Incremental Learning
Key Features

Our proposal

- Non-uniform weighting on Missing data
- An efficient learning algorithm (K times faster than Hu’s ALS, the same magnitude with BPR-SGD learner)
- Seamlessly support online learning.
#1. Item-Oriented Weighting on Missing Data

**Old Design:**

\[ L(\Theta) = \sum_{(u,i) \in R} (y_{ui} - \hat{y}_{ui})^2 + w_0 \sum_{(u,i) \notin R} (0 - \hat{y}_{ui})^2 \]

**Our Proposal:**

\[ L(\Theta) = \sum_{(u,i) \in R} (y_{ui} - \hat{y}_{ui})^2 + \sum_u \sum_{i \notin R_u} c_i (0 - \hat{y}_{ui})^2 \]

The confidence that item \( i \) missed by users is a true negative assessment.

**Popularity-aware Weighting Scheme:**

- **Intuition:** A popular item is more likely to be known by users, thus a missing on it is more probably due to the user not being interested in it.

**Similar to frequency-aware negative sampling in word2vec.**

**Smoothness:** 0.5 works well
#2. Optimization (Coordinate Descent)

- Existing algorithms do not work:
  - SGD: needs to scan all training instance $O(MN)$.
  - ALS: requires a uniform weight on missing data.

- We develop a Coordinate Descent learner to optimize the whole-data based MF:
  - Element-wise Alternating Least Squares Learner (eALS)
  - Optimize one latent factor with others fixed (greedy exact optimization)

<table>
<thead>
<tr>
<th>Property</th>
<th>eALS (ours)</th>
<th>ALS (traditional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization Unit</td>
<td>Latent factor</td>
<td>Latent vector</td>
</tr>
<tr>
<td>Matrix Inversion</td>
<td>No</td>
<td>Yes (ridge regression)</td>
</tr>
<tr>
<td>Time Complexity</td>
<td>$O(MNK)$</td>
<td>$O((M+N)K^3 + MNK^2)$</td>
</tr>
</tbody>
</table>
#2.1 Efficient eALS Learner

- An efficient learner by using memoization.
- Key idea: memoizing the computation for missing data part:

\[
L(\Theta) = \sum_{(u,i) \in \mathcal{R}} (y_{ui} - \hat{y}_{ui})^2 + \sum_u \sum_{i \notin \mathcal{R}_u} c_i (0 - \hat{y}_{ui})^2
\]

- Reformulating the loss function:

\[
L(\Theta) = \sum_{(u,i) \in \mathcal{R}} [(y_{ui} - \hat{y}_{ui})^2 - c_i \hat{y}_{ui}^2] + \sum_u \sum_i c_i \hat{y}_{ui}^2
\]

Sum over all user-item pairs, can be seen as a prior over all interactions!

This term can be computed efficiently \( O(|R| + MK^2) \), rather than \( O(MNK) \). Algorithm details see our paper.
#2.2 Time Complexity

\[ O((M+N)K^2 + |R|K) \]

- # of users
- # of items
- # of observed ratings

Linear to data size!
Given a new \((u, i)\) interaction, how to refresh model parameters without retraining the full model?

Our solution: only perform updates for \(v_u\) and \(v_i\).
- We think the new interaction should change the local features for \(u\) and \(i\) significantly, while the global picture remains largely unchanged.

Pros:
+ Localized complexity: \(O(K^2 + (|R_u| + |R_i|)K)\)
Outline

• Introduction
• Technical Background & Motivation
• Popularity-aware Implicit Method
• Experiments (offline setting)
• Experiments (online setting)
• Conclusion
Dataset & Baselines

- Two public datasets (filtered at threshold 10):
  - Yelp Challenge (Dec 2015, ~1.6 Million reviews)
  - Amazon Movies (SNAP.Stanford)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Interaction#</th>
<th>Item#</th>
<th>User#</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yelp</td>
<td>731,671</td>
<td>25.8K</td>
<td>25.7K</td>
<td>99.89%</td>
</tr>
<tr>
<td>Amazon</td>
<td>5,020,705</td>
<td>75.3K</td>
<td>117.2K</td>
<td>99.94%</td>
</tr>
</tbody>
</table>

- Baselines:
  - ALS (*Hu et al, ICDM’08*)
  - RCD (*Devooght et al, KDD’15*)
    Randomized Coordinate Descent, state-of-the-art implicit MF solution.
  - BPR (*Rendle et al, UAI’09*)
    SGD learner, Pair-wise ranking with sampled missing data.
Offline Protocol (Static data)

• Leave-one-out evaluation \((\text{Rendle et al, UAI'}09)\)
  – Hold out the latest interaction for each user as test (ground-truth).

• Although it is widely used in literatures, it is an artificial split that does not reflect the real scenario.
  – Leak of collaborative information!
  – New users problem is averted.

• Top-K Recommendation \((K=100)\):
  – Rank all items for a user (very time consuming, longer than training!)
  – Measure: Hit Ratio and NDCG.
  – Parameters: \#factors = 128 (others are also fairly tuned, see the paper)
Compare whole-data based MF

Analysis:
1. eALS > ALS: popularity-aware weighting on missing data is useful.
2. ALS > RCD: alternating optimization is more effective than gradient descent for linear MF model.
Compare with Sampled-based BPR

Observation:
1. BPR is a weak performer for Hit Ratio (low recall, as it samples partial missing data only)
2. BPR is a strong performer for NDCG (high precision, as it optimizes a ranking-aware function)
## Efficiency Comparison

### Training time per iteration (Java, single-thread)

<table>
<thead>
<tr>
<th>Factor#</th>
<th>Yelp (0.73M)</th>
<th>Amazon (5M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>eALS</td>
<td>ALS</td>
</tr>
<tr>
<td>32</td>
<td>1 s</td>
<td>10 s</td>
</tr>
<tr>
<td>64</td>
<td>4 s</td>
<td>46 s</td>
</tr>
<tr>
<td>128</td>
<td>13 s</td>
<td>221 s</td>
</tr>
<tr>
<td>256</td>
<td>1 m</td>
<td>23 m</td>
</tr>
<tr>
<td>512</td>
<td>2 m</td>
<td>2.5 h</td>
</tr>
</tbody>
</table>

Analytically:
- eALS: $O((M+N)K^2 + |R|K)$
- ALS: $O((M+N)K^3 + |R|K^2)$

We used a fast matrix inversion algorithm: $O(K^{2.376})$

eALS has the similar running time with RCD (KDD’15), which only supports uniform weighting on missing data.
Online Protocol (dynamic data stream)

- Sort all interactions by time
  - Global split at 90%, testing on the latest 10%.

- In the testing phase:
  - Given a test interaction (i.e., u-i pair), the model recommends a Top-K list to evaluate the performance.
  - Then, the test interaction is fed into the model for an incremental update.

- New users problem is obvious:
  - 57% (Amazon) and 14% (Yelp) test interactions are from new users!
Number of Online Iterations

Impact of online iterations on eALS:

One iteration is enough for eALS to converge!

While BPR (SGD) needs 5-10 iterations.
Compare dynamic MF methods

Performance evolution w.r.t. number of test interactions:

Observations:
1. Our eALS consistently outperforms RCD (Devooght et al, KDD’15) and BPR
2. Performance trend – first decreases (cold-start cases), then increases (usefulness of online learning).
Conclusion

• Matrix Factorization for Implicit Feedback
  – Model the full missing data leads to better prediction recall.
  – Weight the missing data non-uniformly is more effective.
  – Develop an efficient algorithm that supports online incremental learning.

• Explore a new way to evaluate recommendation in a more realistic, better manner.
  – Simulate the dynamic data stream.

• Our efficient eALS technique is a generic solution, which can solve MF with any weighting scheme of missing data.
  – Item-oriented (this work) is just a special case.
Future Work

• Online Recommendation:
  – Balance Short-term (online data) and Long-term preference (offline data).

• Our technique is promising for other applications, e.g., in representation learning of words:
  – GloVe models observed entries only.
  – Word2vec samples negative entries.
  – Recently, Google develops Swivel that accounts for the full missing data, leading to better embedding but very high time complexity.
Thanks!

Codes available: https://github.com/hexiangnan/sigir16-eals