Recommendation with Implicit Feedbacks: Bayesian Personalized Ranking

Weike Pan

College of Computer Science and Software Engineering
Shenzhen University

1502030001: Semantic Web (Spring 2015)
Focus on Intelligent Recommendation Technology
Outline

1. Introduction
2. Method
3. Experiments
4. Conclusion
5. References
We may represent users’ implicit feedbacks in a *matrix* form:

If we can estimate the missing values (denoted as “?”) in the matrix or rank the items directly, we can make recommendations for each user.
Typical Steps in Recommendation with Implicit Feedbacks

For each user $u$:

1. Predict the preference of user $u$ on item $j$, i.e., $\hat{r}_{uj}$, where $j \in \mathcal{I} \setminus \mathcal{I}_u$. We can use different methods, e.g.,
   - PopRank
   - User-based OCCF, item-based OCCF, hybrid OCCF
   - BPR
   - FISM
   - ...

2. Rank the items in $\mathcal{I} \setminus \mathcal{I}_u$ and use the top-$k$ items with highest preference values to construct the recommendation list.
### Notations (1/2)

**Table:** Some notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>user number</td>
</tr>
<tr>
<td>$m$</td>
<td>item number</td>
</tr>
<tr>
<td>$u \in {1, 2, \ldots, n}$</td>
<td>user ID</td>
</tr>
<tr>
<td>$i, j \in {1, 2, \ldots, m}$</td>
<td>item ID</td>
</tr>
<tr>
<td>$\mathcal{R} = {(u, i)}$</td>
<td>(user, item) pairs in training data</td>
</tr>
<tr>
<td>$y_{ui} \in {1, 0}$</td>
<td>indicator variable, $y_{ui} = 1$ if $(u, i) \in \mathcal{R}$</td>
</tr>
<tr>
<td>$\mathcal{I}_u$</td>
<td>preferred items by user $u$ in training data</td>
</tr>
<tr>
<td>$\mathcal{I}$</td>
<td>the whole item set</td>
</tr>
<tr>
<td>$\mathcal{U}$</td>
<td>the whole user set</td>
</tr>
</tbody>
</table>
### Notations (2/2)

**Table:** Some notations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_i \in \mathbb{R}$</td>
<td>item bias</td>
</tr>
<tr>
<td>$d \in \mathbb{R}$</td>
<td>number of latent dimensions</td>
</tr>
<tr>
<td>$U_u. \in \mathbb{R}^{1 \times d}$</td>
<td>user-specific latent feature vector</td>
</tr>
<tr>
<td>$V_i. \in \mathbb{R}^{1 \times d}$</td>
<td>item-specific latent feature vector</td>
</tr>
<tr>
<td>$\hat{r}_{ui}$</td>
<td>predicted rating of user $u$ on item $i$</td>
</tr>
<tr>
<td>$T$</td>
<td>iteration number in the algorithm</td>
</tr>
</tbody>
</table>
Pointwise Preference Assumption

The assumption of pointwise preference on an item [Hu et al., 2008, Pan et al., 2008] can be represented as follows,

\[
\hat{r}_{ui} = 1, \hat{r}_{uj} = 0, \quad i \in I_u, j \in I \setminus I_u,
\]

where 1 and 0 are used to denote “like” and “dislike” for an observed (user, item) pair and an unobserved (user, item) pair, respectively.

Notes:

- Treating all observed feedbacks as “likes” and unobserved feedbacks as “dislikes” may mislead the learning process.
Pairwise Preference Assumption

The assumption of pairwise preferences over two items [Rendle et al., 2009] relaxes the assumption of pointwise preferences, which can be represented as follows,

\[ \hat{r}_{ui} > \hat{r}_{uj}, \ i \in I_u, j \in I \setminus I_u \]  

(2)

where the relationship \( \hat{r}_{ui} > \hat{r}_{uj} \) means that a user \( u \) is likely to prefer an item \( i \in I_u \) to an item \( j \in I \setminus I_u \).

Notes:

- Empirically, this assumption generates better recommendation results than the pointwise assumption.
The predicted rating of user $u$ on item $i$, 

$$\hat{r}_{ui} = U_u \cdot V_i^T + b_i$$

(3)

Question:

- why not include $b_u$ and $\mu$
The Bernoulli distribution of binary random variable $\delta((u, i) \succ (u, j))$ is defined as follows [Rendle et al., 2009],

$$LPP_u = \prod_{i,j \in \mathcal{I}} Pr(\hat{r}_{ui} > \hat{r}_{uj}) \delta((u,i) \succ (u,j)) [1 - Pr(\hat{r}_{ui} > \hat{r}_{uj})][1 - \delta((u,i) \succ (u,j))]$$

$$= \prod_{(u,i) \succ (u,j)} Pr(\hat{r}_{ui} > \hat{r}_{uj}) \times \prod_{(u,i) \preceq (u,j)} [1 - Pr(\hat{r}_{ui} > \hat{r}_{uj})]$$

where $(u, i) \succ (u, j)$ means that user $u$ prefers item $i$ to item $j$. 
Likelihood of Pairwise Preferences (2/2)

We use \( \sigma(\hat{r}_{uij}) \) to approximate the probability \( Pr(\hat{r}_{ui} > \hat{r}_{uj}) \) [Rendle et al., 2009], where \( \hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj} \), and have

\[
\ln LPP_u = \ln \prod_{(u,i) > (u,j)} \sigma(\hat{r}_{uij}) + \ln \prod_{(u,i) \leq (u,j)} [1 - \sigma(\hat{r}_{uij})] \\
\approx \ln \prod_{(u,i) > (u,j)} \sigma(\hat{r}_{uij}) + \ln \prod_{(u,i) > (u,j)} [1 - \sigma(-\hat{r}_{uij})] \\
= \ln \prod_{(u,i) > (u,j)} \sigma(\hat{r}_{uij}) + \ln \prod_{(u,i) > (u,j)} \sigma(\hat{r}_{uij}) \\
= 2 \sum_{(u,i) > (u,j)} \ln \sigma(\hat{r}_{uij}) \\
= 2 \sum_{i \in I_u} \sum_{j \in I \setminus I_u} \ln \sigma(\hat{r}_{uij})
\]  

(4)

where \( \sigma(z) = 1/(1 + e^{-z}) \) is the sigmoid function.
Objective Function

Objective function,

$$\min_{\Theta} \sum_{u \in U} \sum_{i \in I_u} \sum_{j \in I \setminus I_u} f_{uij}$$

where

$$f_{uij} = -\ln \sigma (\hat{r}_{uij}) + \frac{\alpha_u}{2} \| U_u \|^2 + \frac{\alpha_v}{2} \| V_i \|^2 + \frac{\alpha_v}{2} \| V_j \|^2 + \frac{\beta_v}{2} \| b_i \|^2 + \frac{\beta_v}{2} \| b_j \|^2$$

and \( \Theta = \{ U_u, u = 1, 2, \ldots, n; V_i, b_i, i = 1, 2, \ldots, m \} \) denotes the set of parameters to be learned.
Gradients

For a randomly sampled triple \((u, i, j)\), we have the gradients,

\[
\nabla U_u. = \frac{\partial f_{uij}}{\partial U_u.} = -\sigma(-\hat{r}_{uij})(V_i. - V_j.) + \alpha_u U_u.,
\]

\[
\nabla V_i. = \frac{\partial f_{uij}}{\partial V_i.} = -\sigma(-\hat{r}_{uij}) U_u. + \alpha_v V_i.,
\]

\[
\nabla V_j. = \frac{\partial f_{uij}}{\partial V_j.} = -\sigma(-\hat{r}_{uij})(-U_u.) + \alpha_v V_j.,
\]

\[
\nabla b_i = \frac{\partial f_{uij}}{\partial b_i} = -\sigma(-\hat{r}_{uij}) + \beta_v b_i,
\]

\[
\nabla b_j = \frac{\partial f_{uij}}{\partial b_j} = -\sigma(-\hat{r}_{uij})(-1) + \beta_v b_j,
\]

where \(\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}\).
For a randomly sampled triple \((u, i, j)\), we have the update rules,

\[
U_u = U_u - \gamma \nabla U_u, \\
V_i = V_i - \gamma \nabla V_i, \\
V_j = V_j - \gamma \nabla V_j, \\
b_i = b_i - \gamma \nabla b_i, \\
b_j = b_j - \gamma \nabla b_j,
\]

where \(\gamma\) is the learning rate.
Algorithm

1: Initialize the model parameters $\Theta$
2: $\textbf{for } t = 1, \ldots, T \textbf{ do}$
3: $\quad \textbf{for } t_2 = 1, \ldots, |\mathcal{R}| \textbf{ do}$
4: $\quad$ Randomly pick up a pair $(u, i) \in \mathcal{R}$
5: $\quad$ Randomly pick up an item $j$ from $\mathcal{I}\setminus\mathcal{I}_u$
6: $\quad$ Calculate the gradients via Eq.(6-10)
7: $\quad$ Update the model parameters via Eq.(11-15)
8: $\quad$ $\textbf{end for}$
9: $\textbf{end for}$

Figure: The SGD algorithm for BPR.
Data Set

- We use the files u1.base and u1.test of MovieLens100K\(^1\) as our training data and test data, respectively.

- user number: \( n = 943; \) item number: \( m = 1682. \)

- u1.base (training data): 80000 rating records, and the density (or sparsity) is \( \frac{80000}{943/1682} = 5.04\% . \)

- u1.test (test data): 20000 rating records.

- Pre-processing (for simulation): we only keep the (user, item) pairs with ratings 4 or 5 in u1.base and u1.test as preferred (user, item) pairs, and remove all other records. Finally, we obtain u1.base.OCCF and u1.test.OCCF.

\(^1\)http://grouplens.org/datasets/
Evaluation Metrics

- **Pre@5**: The precision of user $u$ is defined as,

$$ Pre_u@k = \frac{1}{k} \sum_{\ell=1}^{k} \delta(i(\ell) \in \mathcal{I}_u^{te}), $$

where $\delta(x) = 1$ if $x$ is true and $\delta(x) = 0$ otherwise. Then, we have

$$ Pre@k = \sum_{u \in U^{te}} Pre_u@k / |U^{te}|. $$

- **Rec@5**: The recall of user $u$ is defined as,

$$ Rec_u@k = \frac{1}{|\mathcal{I}_u^{te}|} \sum_{\ell=1}^{k} \delta(i(\ell) \in \mathcal{I}_u^{te}), $$

which means how many preferred items are recommended in the top-$k$ list. Then, we have

$$ Rec@k = \sum_{u \in U^{te}} Rec_u@k / |U^{te}|. $$
Experiments

Initialization of Model Parameters

We use the statistics of training data to initialize the model parameters,

\[ b_i = \left( \frac{1}{n} \sum_{u=1}^{n} y_{ui} \right) - \mu \]

\[ V_{ik} = (r - 0.5) \times 0.01, \quad k = 1, \ldots, d \]

\[ U_{uk} = (r - 0.5) \times 0.01, \quad k = 1, \ldots, d \]

where \( r \) (\( 0 \leq r < 1 \)) is a random variable, and \( \mu = \sum_{u=1}^{n} \sum_{i=1}^{m} y_{ui} / n / m \).
Parameter Configurations

We fix $\gamma = 0.01$, and search the best values of the following parameters,

- $\alpha_u = \alpha_v = \beta_v \in \{0.001, 0.01, 0.1\}$
- $T \in \{100, 500, 1000\}$
- $d = 20$

Finally, we use $\gamma = 0.01$, $\alpha_u = \alpha_v = \beta_v = 0.01$, $T = 500$ and $d = 20$. 
Table: Prediction performance of PopRank and BPR on MovieLens100K (u1.base.OCCF, u1.test.OCCF). Note that the time cost using Java in my PC is less than 15 seconds.

<table>
<thead>
<tr>
<th></th>
<th>PopRank</th>
<th>BPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre@5</td>
<td>0.2338</td>
<td>0.3864</td>
</tr>
<tr>
<td>Rec@5</td>
<td>0.0571</td>
<td>0.1184</td>
</tr>
</tbody>
</table>
The pairwise preference assumption is useful.
Homework

- Read the code of BPR implementation in MyMediaLite\(^2\)
  - Pay attention to different sampling strategies

- Implement BPR and conduct empirical studies on u2.base.OCCF, u2.test.OCCF of MovieLens100K with similar pre-processing

- Read the UAI 2009 paper [Rendle et al., 2009]

\(^2\)http://www.mymedialite.net/
<table>
<thead>
<tr>
<th>References</th>
</tr>
</thead>
</table>