Bayesian Personalized Ranking

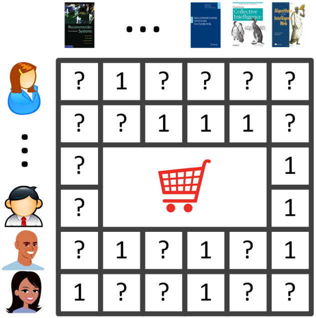
Weike Pan

College of Computer Science and Software Engineering
Shenzhen University

Outline

Recommendation with Implicit Feedback

- We may represent users' implicit feedback 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 Feedback

For each user u:

- Step 1: Predict the preference of user u on item j, i.e., $\hat{r}_{uj}$, where $j \in \mathcal{I} \backslash \mathcal{I}_u$. We can use different methods, e.g.,
  - PopRank
  - User-based OCCF, item-based OCCF, hybrid OCCF
  - BPR
  - FISM
  - ...
- Step 2: Rank the items in $\mathcal{I} \backslash \mathcal{I}_u$ and use the top-k items with highest preference values to construct the recommendation list

Notations (1/2)

Table: Some notations.

n	user number
m	item number
$u \in \{1, 2, \ldots, n\}$	user ID
$i, j \in \{1, 2, \ldots, m\}$	item ID
$\mathcal{R} = \{(u, i)\}$	(user, item) pairs in training data
$y_{ui} \in \{1, 0\}$	indicator variable, $y_{ui} = 1$ if $(u, i) \in \mathcal{R}$
$\mathcal{I}_u$	preferred items by user u in training data
$\mathcal{I}$	the whole item set
$\mathcal{U}$	the whole user set

Notations (2/2)

Table: Some notations.

$b_i \in \mathbb{R}$	item bias
$d \in \mathbb{R}$	number of latent dimensions
$U_{u\cdot} \in \mathbb{R}^{1\times d}$	user-specific latent feature vector
$V_{i\cdot} \in \mathbb{R}^{1\times d}$	item-specific latent feature vector
$\hat{r}_{ui}$	predicted rating of user u on item i
T	iteration number in the algorithm

Pointwise Preference Assumption

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

$$\hat{r}_{ui} = 1, \hat{r}_{uj} = 0, \; i \in \mathcal{I}_u, j \in \mathcal{I} \backslash \mathcal{I}_u, \tag{1}$$

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 feedback as "likes" and unobserved feedback 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,

$$\hat{r}_{ui} > \hat{r}_{uj}, \ i \in \mathcal{I}_u, j \in \mathcal{I} \backslash \mathcal{I}_u \tag{2}$$

where the relationship $\hat{r}_{ui} > \hat{r}_{uj}$ means that a user u is likely to prefer an item $i \in \mathcal{I}_u$ to an item $j \in \mathcal{I} \backslash \mathcal{I}_u$.

Notes:

- Empirically, this assumption generates better recommendation results than the pointwise assumption.

Prediction Rule

The predicted rating of user u on item i,

$$\hat{r}_{ui} = U_{u\cdot} V_{i\cdot}^T + b_i \tag{3}$$

Question:

- why not include b_u and μ

Likelihood of Pairwise Preferences (1/2)

The Bernoulli distribution of binary random variable $\delta((u,i) \succ (u,j))$ is defined as follows [Rendle et al., 2009],

$$\begin{aligned} \text{LPP}_u &= \prod_{i,j \in \mathcal{I}} Pr(\hat{r}_{ui} > \hat{r}_{uj})^{\delta((u,i)\succ(u,j))}[1 - Pr(\hat{r}_{ui} > \hat{r}_{uj})]^{[1-\delta((u,i)\succ(u,j))]} \\ &= \prod_{(u,i)\succ(u,j)} Pr(\hat{r}_{ui} > \hat{r}_{uj}) \times \prod_{(u,i)\preceq(u,j)} [1 - Pr(\hat{r}_{ui} > \hat{r}_{uj})] \end{aligned}$$

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

$$
\begin{aligned}
\ln \mathrm{LPP}_u &= \ln \prod_{(u,i)\succ(u,j)} \sigma(\hat{r}_{uij}) + \ln \prod_{(u,i)\preceq(u,j)} [1 - \sigma(\hat{r}_{uij})] \\
&\approx \ln \prod_{(u,i)\succ(u,j)} \sigma(\hat{r}_{uij}) + \ln \prod_{(u,i)\succ(u,j)} [1 - \sigma(-\hat{r}_{uij})] \\
&= \ln \prod_{(u,i)\succ(u,j)} \sigma(\hat{r}_{uij}) + \ln \prod_{(u,i)\succ(u,j)} \sigma(\hat{r}_{uij}) \\
&= 2 \sum_{(u,i)\succ(u,j)} \ln \sigma(\hat{r}_{uij}) \\
&= 2 \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I} \setminus \mathcal{I}_u} \ln \sigma(\hat{r}_{uij}) \qquad (4)
\end{aligned}
$$

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

Objective Function

Objective function,

$$\min_{\Theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u} \sum_{j \in \mathcal{I} \backslash \mathcal{I}_u} f_{uij} \tag{5}$$

where

$f_{uij} = -\ln \sigma(\hat{r}_{uij}) + \frac{\alpha_u}{2}||U_{u\cdot}||^2 + \frac{\alpha_v}{2}||V_{i\cdot}||^2 + \frac{\alpha_v}{2}||V_{j\cdot}||^2 + \frac{\beta_v}{2}||b_i||^2 + \frac{\beta_v}{2}||b_j||^2$ and $\Theta = \{U_{u\cdot}, u = 1, 2, \ldots, n; V_{i\cdot}, b_i, i = 1, 2, \ldots, m\}$ denotes the set of parameters to be learned.

Gradients

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

$$\nabla U_{u\cdot} = \frac{\partial f_{uij}}{\partial U_{u\cdot}} = -\sigma(-\hat{r}_{uij})(V_{i\cdot} - V_{j\cdot}) + \alpha_u U_{u\cdot}, \quad (6)$$

$$\nabla V_{i\cdot} = \frac{\partial f_{uij}}{\partial V_{i\cdot}} = -\sigma(-\hat{r}_{uij})U_{u\cdot} + \alpha_v V_{i\cdot}, \quad (7)$$

$$\nabla V_{j\cdot} = \frac{\partial f_{uij}}{\partial V_{j\cdot}} = -\sigma(-\hat{r}_{uij})(-U_{u\cdot}) + \alpha_v V_{j\cdot}, \quad (8)$$

$$\nabla b_i = \frac{\partial f_{uij}}{\partial b_i} = -\sigma(-\hat{r}_{uij}) + \beta_v b_i, \quad (9)$$

$$\nabla b_j = \frac{\partial f_{uij}}{\partial b_j} = -\sigma(-\hat{r}_{uij})(-1) + \beta_v b_j, \quad (10)$$

where $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$.

Update Rules

For a randomly sampled triple (u, i, j), we have the update rules,

$$
\begin{aligned}
U_{u\cdot} &= U_{u\cdot} - \gamma \nabla U_{u\cdot}, & (11)\\
V_{i\cdot} &= V_{i\cdot} - \gamma \nabla V_{i\cdot}, & (12)\\
V_{j\cdot} &= V_{j\cdot} - \gamma \nabla V_{j\cdot}, & (13)\\
b_{i} &= b_{i} - \gamma \nabla b_{i}, & (14)\\
b_{j} &= b_{j} - \gamma \nabla b_{j}, & (15)
\end{aligned}
$$

where γ is the learning rate.

Algorithm

```
Initialize the model parameters Θ
for t = 1, ..., T do
    for t_2 = 1, ..., |R| do
        Randomly pick up a pair (u, i) ∈ R
        Randomly pick up an item j from I\I_u
        Calculate the gradients via Eq.(6-10)
        Update the model parameters via Eq.(11-15)
    end for
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 $80000/943/1682 = 5.04\%$.
- 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 $Pre@k = \sum_{u\in\mathcal{U}^{te}} Pre_u@k/|\mathcal{U}^{te}|$.

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

Initialization of Model Parameters

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

$$
\begin{aligned}
b_i &= \left(\frac{1}{n}\sum_{u=1}^{n} y_{ui}\right) - \mu \\
V_{ik} &= (r - 0.5) \times 0.01, k = 1, \ldots, d \\
U_{uk} &= (r - 0.5) \times 0.01, k = 1, \ldots, d
\end{aligned}
$$

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 $d = 20$.

Results

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.

	PopRank	BPR
Pre@5	0.2338	0.3864
Rec@5	0.0571	0.1184

Conclusion

- 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/

Hu, Y., Koren, Y., and Volinsky, C. (2008).
Collaborative filtering for implicit feedback datasets.
In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA. IEEE Computer Society.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008).
One-class collaborative filtering.
In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, Washington, DC, USA. IEEE Computer Society.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009).
Bpr: Bayesian personalized ranking from implicit feedback.
In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.