BPR: Bayesian Personalized Ranking from Implicit Feedback

/* { for beginners } */

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme

{srendle, freudenthaler, gantner, schmidt-thieme}@ismll.de Machine Learning Lab, University of Hildesheim Marienburger Platz 22, 31141 Hildesheim, Germany

2009 UAI

Matrix Factorization

	i_1	i_2	i ₃	i ₄	i ₅	i ₆	i,	i ₈
u_1	5	2		3		4		
u_2	4	3			5			
u_3	4		2				2	4
u_4								
u_5	5	1	2		4	3		
u_6	4	3		2	4		3	5

$$R = U^T V$$

- 致力于单个项目的得分 $\hat{r_{ui}}$
- 没有对排名进行直接的优化

$$V = \begin{bmatrix} 1.55 & 1.22 & 0.37 & 0.81 & 0.62 & -0.01 \\ 0.36 & 0.91 & 1.21 & 0.39 & 1.10 & 0.25 \\ 0.59 & 0.20 & 0.14 & 0.83 & 0.27 & 1.51 \\ 0.39 & 1.33 & -0.43 & 0.70 & -0.90 & 0.68 \\ 1.05 & 0.11 & 0.17 & 1.18 & 1.81 & 0.40 \end{bmatrix} V = \begin{bmatrix} 1.00 & -0.05 & -0.24 & 0.26 & 1.28 & 0.54 & -0.31 & 0.52 \\ 0.19 & -0.86 & -0.72 & 0.05 & 0.68 & 0.02 & -0.61 & 0.70 \\ 0.49 & 0.09 & -0.05 & -0.62 & 0.12 & 0.08 & 0.02 & 1.60 \\ 0.70 & 0.27 & -0.27 & 0.99 & 0.44 & 0.39 & 0.74 \\ 1.49 & -1.00 & 0.06 & 0.05 & 0.23 & 0.01 & -0.36 & 0.80 \end{bmatrix}$$

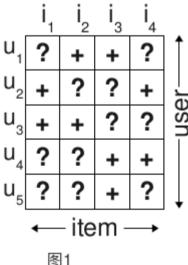
$$1.55$$
 0.36
 0.59
 0.39
 1.05

$$1.00$$
 0.19
 0.49
 -0.40
 1.49

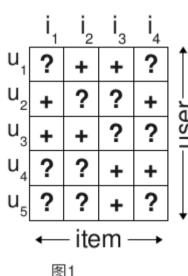
$$= 1.55*1 + 0.36*0.19 + 0.59*0.49 - 0.39*0.4 + 1.05*1.49 = 3.316$$

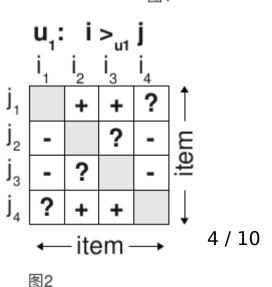
- 这篇论文,从贝叶斯分析入手,提出了一个通用的优 化标准 BPR-Opt ,用于个性化排序
- 隐反馈(在线商城的购买记录、音乐视频网站的播放 记录等) $S \subseteq U \times I$ 如图 1 所示
- 仅 positive observation(+) 可用
- non-observed user-item pairs(?)
 - negative feedback (the user is not interested in buying the item)
 - missing values

(the user might want to buy the item in the future)



- 同时包含 positive pairs 、 negative pairs 、 missing values pairs
 - (u,i_2,j_1) 为 positive pair(+)
 - (u,i_1,j_2) 为 negative pair(-)
 - (u,i_4,j_1) 为 missing value pair(?)
- Item Recommendation $>u \subset I^2$
 - totality $\forall i, j \in I: i \neq j \Rightarrow i > uj \lor j > ui$
 - antisymmetry $\forall i,j \in I: i > uj \land j > ui \Rightarrow i = j$
 - transitivity $\forall i,j,k \in I: i > uj \land j > uk \Rightarrow i > uk$
- 常规方法





- 在这篇论文中,为了更好反映问题的本身(项目排序),作者使用 item pairs(项目对)作为训练数据并且致力于优化正确的 item pairs 的排名,而不是单个项目的得分
- 即,利用隐反馈S,作者致力于为每个用户重构序列 >u ,如果用户u和项目i有交互 $(u,i) \in S$,那么假设该用户更喜欢这个项目相对于其他的 non-observed items j ,有 i > u j ,表示为三元组 (u,i,j)

$$D_S := \{(u, i, j) | i \in I_u^+ \land j \in I \setminus I_u^+ \}$$

where $I_u^+ := \{i \in I : (u, i) \in S\}$

- 具体地,贝叶斯个性化排序就是要最大化下面的后验 概率 $p(\Theta|>_u) \propto p(>_u|\Theta) p(\Theta)$
- 假设用户间行为独立,且用户对每个项目对的排序也是独立的,因此似然函数p (>u| Θ) 可以被重写为

$$\prod_{u \in U} p(>_u |\Theta) = \prod_{(u,i,j) \in D_S} p(i>_u j|\Theta)$$

• 而用户相对项目j更喜欢项目i的个体概率可以定义为 $p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$

其中 σ 是logistic sigmoid function:

$$\sigma(x) := \frac{1}{1 + e^{-x}}$$

这里的 $\hat{x}_{uij}(\Theta)$ 可以是任意一个关于模型参数 Θ ,可以捕

获用户u和项目i,j之间的关系的实值函数。比如我们可以

定义为:

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj}$$

- 关于先验分布,作者引入了一个比较通用的均值为 0 ,方差为协方差矩阵 $\Sigma\Theta$ 的正态分布 $p(\Theta) \sim N(0, \Sigma_{\Theta})$
- 这个时候,我们的贝叶斯个性化排序框架 BPR-OPT 就已经完成了

BPR-OPT :=
$$\ln p(\Theta|>_u)$$

= $\ln p(>_u|\Theta) p(\Theta)$
= $\ln \prod_{(u,i,j)\in D_S} \sigma(\hat{x}_{uij}) p(\Theta)$
= $\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta)$
= $\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_{\Theta} ||\Theta||^2$

可以看出,当 Θ 的先验分布是上面说的正态分布的时候,有着类似正则化项的效果 $\ln p(\Theta) \propto -\lambda_{\Theta} \|\Theta\|^2$

$$\ln p(\Theta) := \ln N(\Theta|0, \lambda_{\Theta}I)$$

$$= \ln \frac{1}{\sqrt{2\pi\lambda_{\Theta}I}} \exp(-\frac{\|\Theta\|^2}{2\lambda_{\Theta}I})$$

$$= -\frac{1}{2}\ln(2\pi\lambda_{\Theta}I) - \frac{\|\Theta\|^2}{2\lambda_{\Theta}I}$$

$$= -\frac{1}{2}\ln(\lambda_{\Theta}I) - \frac{\|\Theta\|^2}{2\lambda_{\Theta}I} + C$$

$$\propto -\lambda_{\Theta}\|\Theta\|^2$$

BPR-Learn

贝叶斯个性化排序需要最大化后验概率,为此我们可以用梯度上升的方法求解

 $(u,i,j) \in D_S$

$$\frac{\partial \text{BPR} - \text{OPT}}{\partial \Theta} = \sum_{(u,i,j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma \left(\hat{x}_{uij} \right) - \lambda_{\Theta} \frac{\partial}{\partial \Theta} \|\Theta\|^2$$

$$\propto \sum_{(u,i,j) \in D_S} \frac{1}{\sigma(\hat{x}_{uij})} \cdot \frac{0 - (-e^{-\hat{x}_{uij}})}{(1 + e^{-\hat{x}_{uij}})^2} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_{\Theta} \Theta$$

$$\propto \sum_{(u,i,j)\in D_S} \frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_{\Theta} \Theta$$

 $\propto \sum_{i} (1 - \sigma(\hat{x}_{uij})) \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_{\Theta} \Theta$

具体优化流程为:

1: **procedure** LearnBPR(
$$D_S, \Theta$$
)

- 2: initialize Θ
- 3: repeat
- 4: draw (u, i, j) from D_S
- 5: $\Theta \leftarrow \Theta + \alpha((1 \sigma(\hat{x}_{uij})) \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} \lambda_{\Theta}\Theta)$
- 6: until convergence
- 7: return Θ
- 8: end procedure

BPR-MF

我们知道在矩阵分解中,我们想把评分矩阵R分解用户隐特征因子矩阵P和项目隐特征因子矩阵Q相乘的形式:

$$R = P^T Q$$

即用户u对项目i的评分为 $\hat{x}_{ui} = \sum_{f=1}^k p_{uf} \cdot q_{if}$

所以这里 \hat{x}_{uij} 可以定义为:

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj} = \sum_{f=1}^{k} p_{uf} \cdot q_{if} - \sum_{f=1}^{k} p_{uf} \cdot q_{jf}$$

fighthal

https://github.com/Coder-Yu/RecQ/blob/master/algorithm/ranking/BPR.py

$$\frac{\partial}{\partial \theta} \hat{x}_{uij} = \begin{cases} \left(q_{if} - q_{jf} \right) & \text{if } \theta = p_{uf} \\ p_{uf} & \text{if } \theta = q_{if} \\ -p_{uf} & \text{if } \theta = q_{jf} \end{cases}$$

$$\frac{\partial \mathrm{BPR} - \mathrm{OPT}}{\partial \Theta}$$

$$\propto \sum_{(u,i,j)\in D_S} (1 - \sigma(\hat{x}_{uij})) \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_{\Theta} \Theta$$

```
def optimization(self, u, i, j):
    s = sigmoid(self.P[u].dot(self.Q[i]) - self.P[u].dot(self.Q[j]))
    # update latent features of user u
    self.P[u] += self.lRate * (_(1 - s) * (self.Q[i] - self.Q[j]) - self.regU * self.P[u]_)
    # update latent features of item i
    self.Q[i] += self.lRate * (_(1 - s) * self.P[u] - self.regI * self.Q[i]_)
    # update latent features of item j
    self.Q[j] -= self.lRate * (_(1 - s) * self.P[u] - self.regI * self.Q[j]_)
    self.loss += -log(s)
```

The End

Thank You