# BPR算法设计文档

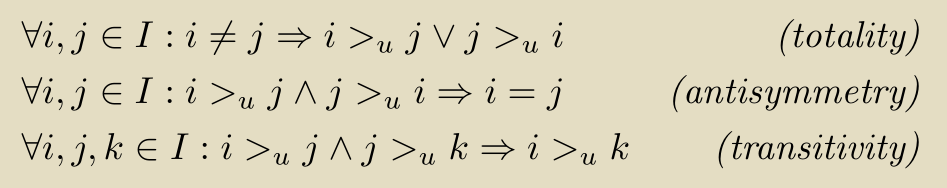
## 1. BPR算法介绍

### 1.1 背景

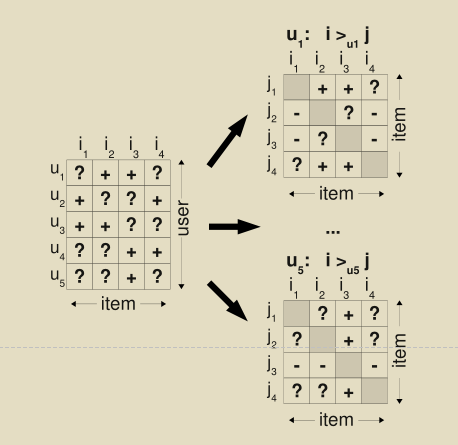
最早的第一类排序算法类别是点对方法(Pointwise Approach)，这类算法将排序问题被转化为分类、回归之类的问题，并使用现有分类、回归等方法进行实现。第二类排序算法是成对方法(Pairwise Approach)，在序列方法中，排序被转化为对序列分类或对序列回归。所谓的pair就是成对的排序，比如(a,b)一组表明a比b排的靠前。我们要讲到的BPR就属于这一类。第三类排序算法是列表方法(Listwise Approach)，它采用更加直接的方法对排序问题进行了处理。它在学习和预测过程中都将排序列表作为一个样本。排序的组结构被保持。

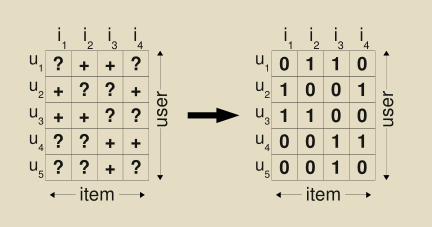
### 1.2 BPR建模思路

贝叶斯个性化排序(Bayesian Personalized Ranking, 以下简称BPR)，为用户进行个性化推荐需要实现一种排序的机制，使得用户对物品的偏好可以量化比较。用户u的排序规则可以用表示，那么表示用户u对物品i的偏好大于其对物品j的偏好.这样的排序规则应该满足如下的性质：



传统的推荐方法将用户—物品的稀疏矩阵U-I中的所有缺失值设为0，即统一认为是用户不感兴趣的.而事实上这些缺失值代表的是用户不感兴趣的物品和感兴趣而未发生关联的物品的集合.因此用简单的0代替全部缺失值后，再经过机器学习拟合出来的模型理论上不能推荐任何东西，只能通过正则化的方法来避免过拟合.





因此，将原有的U-I的矩阵拆分成每个用户的物品I-J的偏好矩阵，避免全部替换为0造成的混淆，再经过算法计算得到其余物品之间未知的偏好关系，从而实现物品间的排序.

#### 基本假设

（a）用户间的行为相互独立。

（b）对于某一用户而言，该用户对某一物品对(i, j)的排序，与该用户对其他物品对的排序之间是相互独立的。

（c）定义用户u对i比j的偏好值：

⑴

（d）定义用户u对物品i相对于j的偏好概率：

⑵

⑶

1. 的概率分布服从均值为0，协方差矩阵为的正态分布.

#### 1.2.2 建模

由前一章节了解到BPR的主要目标是求取用户u对i比j的偏好值，根据论文给出的定义该问题转化为求取，这可以使用矩阵分解的方法，对于用户集U和物品集I的对应的U×I的预测排序矩阵X，我们期望得到两个分解后的用户矩阵W:|U|×k和物品矩阵H:|I|×k，满足

⑷

其中W中的每一行表示一个用户u，H中的每一行表示一个物品i，上述公式可以表示为：

⑸

.

将任意用户u对应的物品进行标记，如果用户u在同时有物品i和j的时候点击了i，那么我们就得到了一个三元组<u,i,j>，它表示对用户u来说，i的排序要比j靠前。如果对于用户u来说我们有m组这样的反馈，那么我们就可以得到m组用户u对应的训练样本。

#### 1.2.3 优化

由先前的假设，我们可以推出：

⑹

对于：

其中，

⑻

化简得到,

⑼

将式(1)代入式(9)得到，

对于，根据假设v有：

⑾

将式(10)和(11)代入式(6)，有

这个式子可以用梯度上升法来优化求解模型参数。对θ求导，我们有

由于

可以求得：

## 2 编程实现

数据集：[MovieLens 100K](http://link.zhihu.com/?target=http%3A//files.grouplens.org/datasets/movielens/ml-100k.zip)数据集。包含943个用户对1682部电影的打分。由于BPR是排序算法，因此数据集里的打分会被我们忽略，主要是假设用户看过的电影会比用户没看的电影的排序评分高。

### 2.1算法流程、伪码：

输入：训练集D三元组，梯度步长α，正则化参数λ，分解矩阵维度k.

输出：模型参数，矩阵W，H.

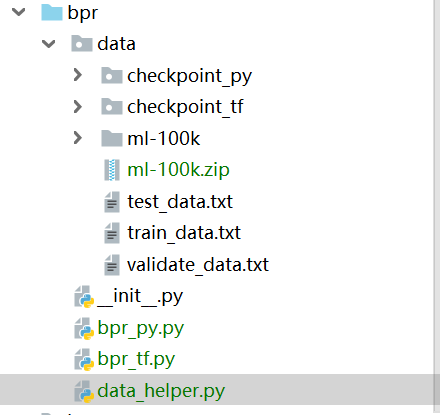
1). 随机初始化矩阵W，H

2). 迭代更新模型参数：

3). 如果W，H收敛，则算法结束，输出W，H，否则回到步骤2.

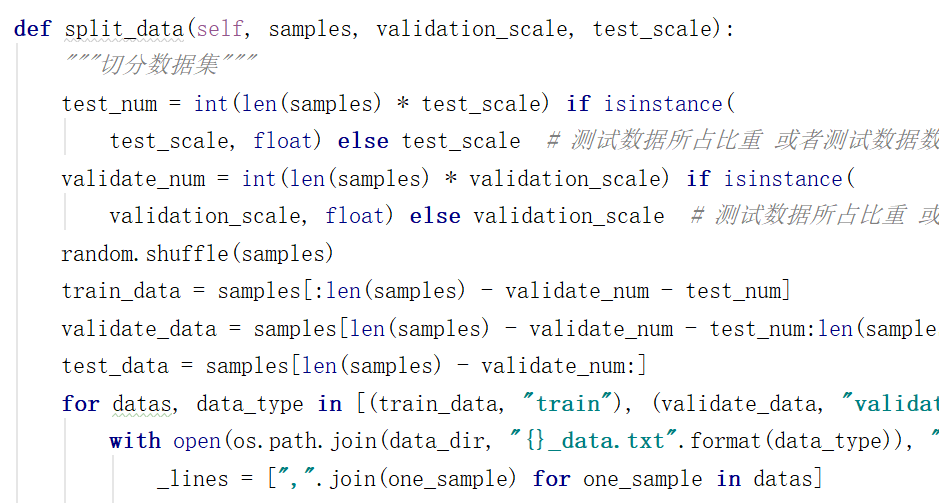
### 2.2 代码详解

代码分为tensorflow版和纯python numpy版两个部分，其中数据载入代码两个版本公用。

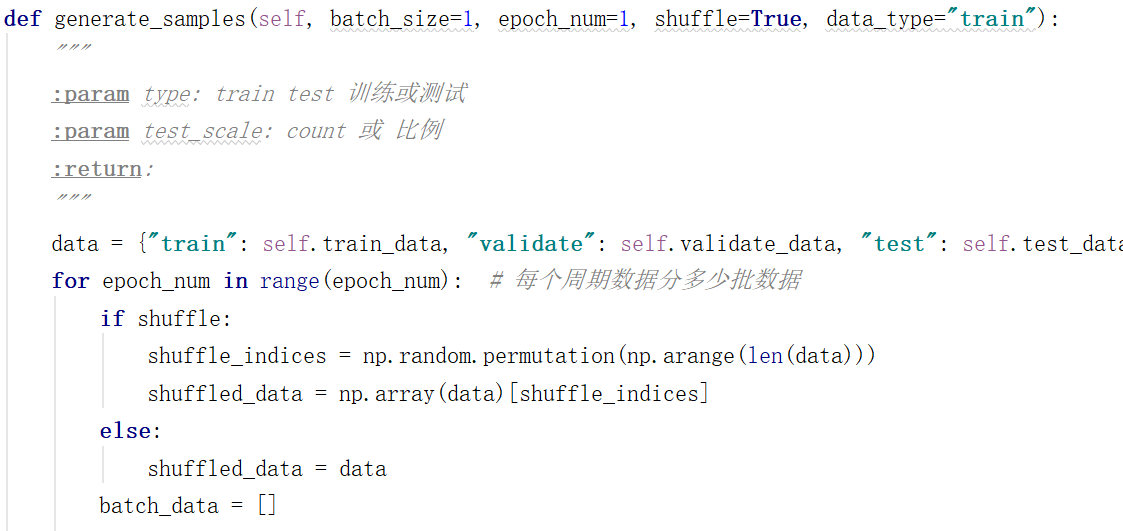


#### 2.2.1 数据载入（data\_helper.py）

**1）数据集切分**

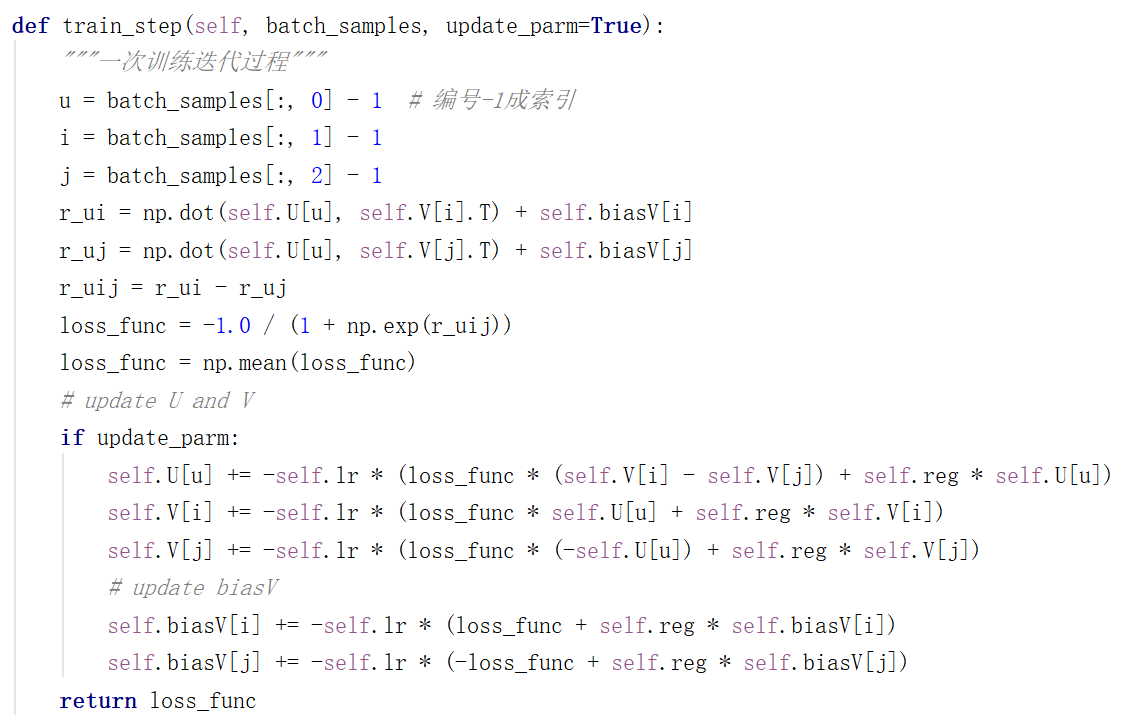


**2）生成训练数据**

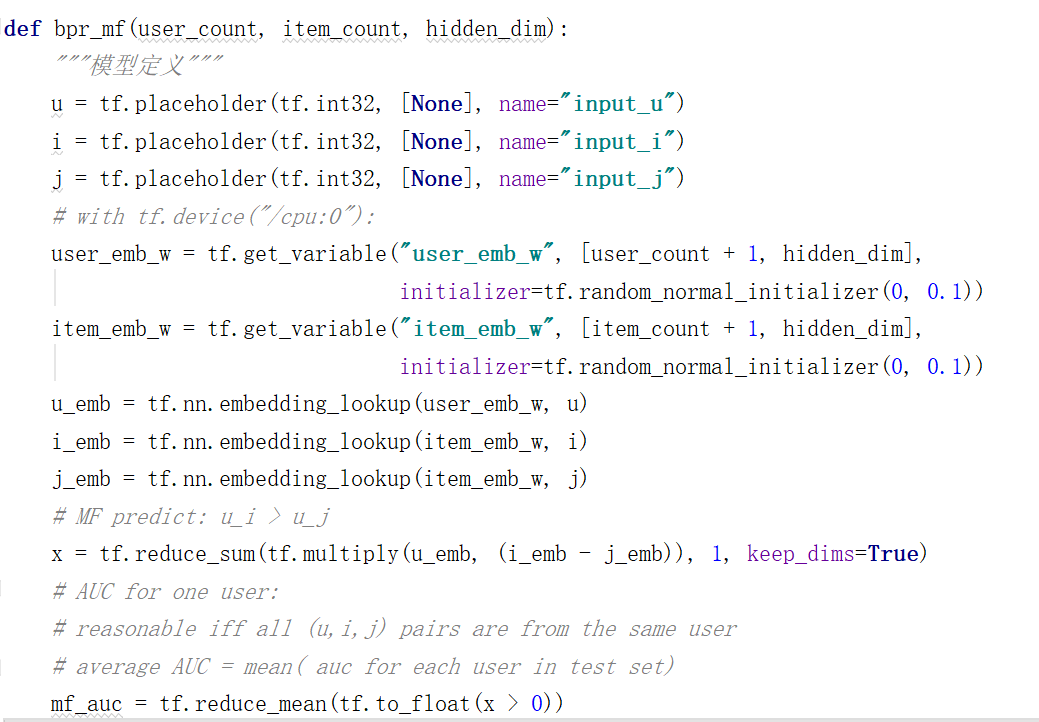


#### 2.2.2 模型构建

**1）python版**

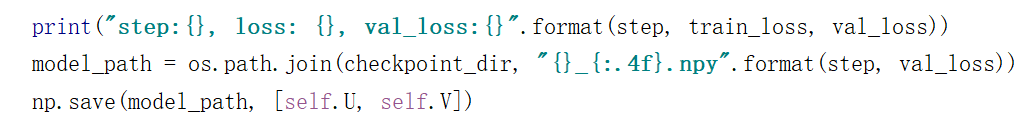


**2）tensorflow版**

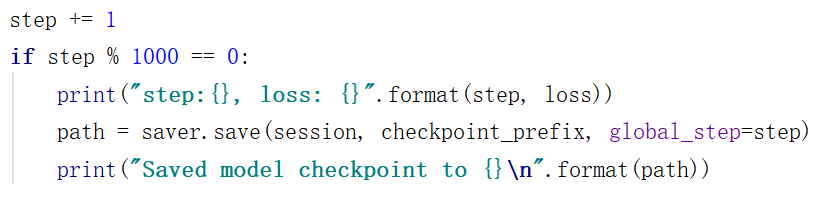


#### 2.2.3 模型保存

1）Python numpy版

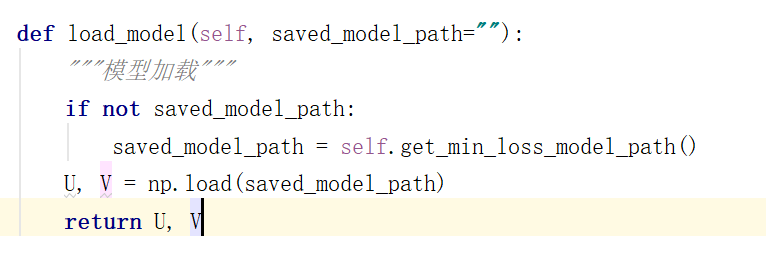


2）tensorflow版

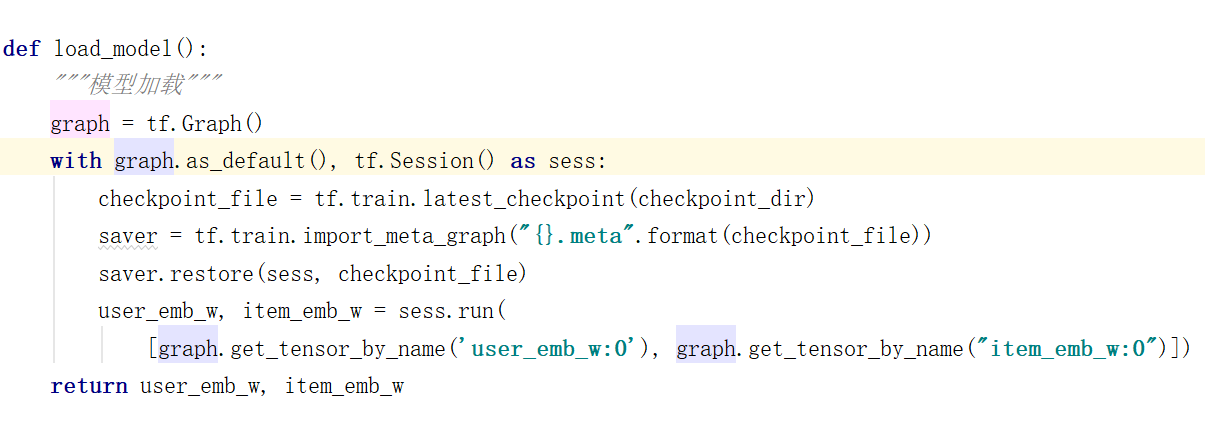


#### 2.2.4 模型加载

**1）Python版**

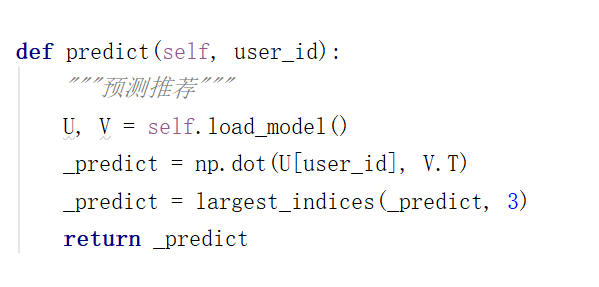


**2）tensorflow版**

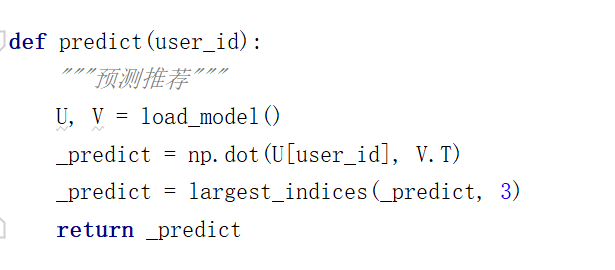


#### 2.2.5 推荐预测

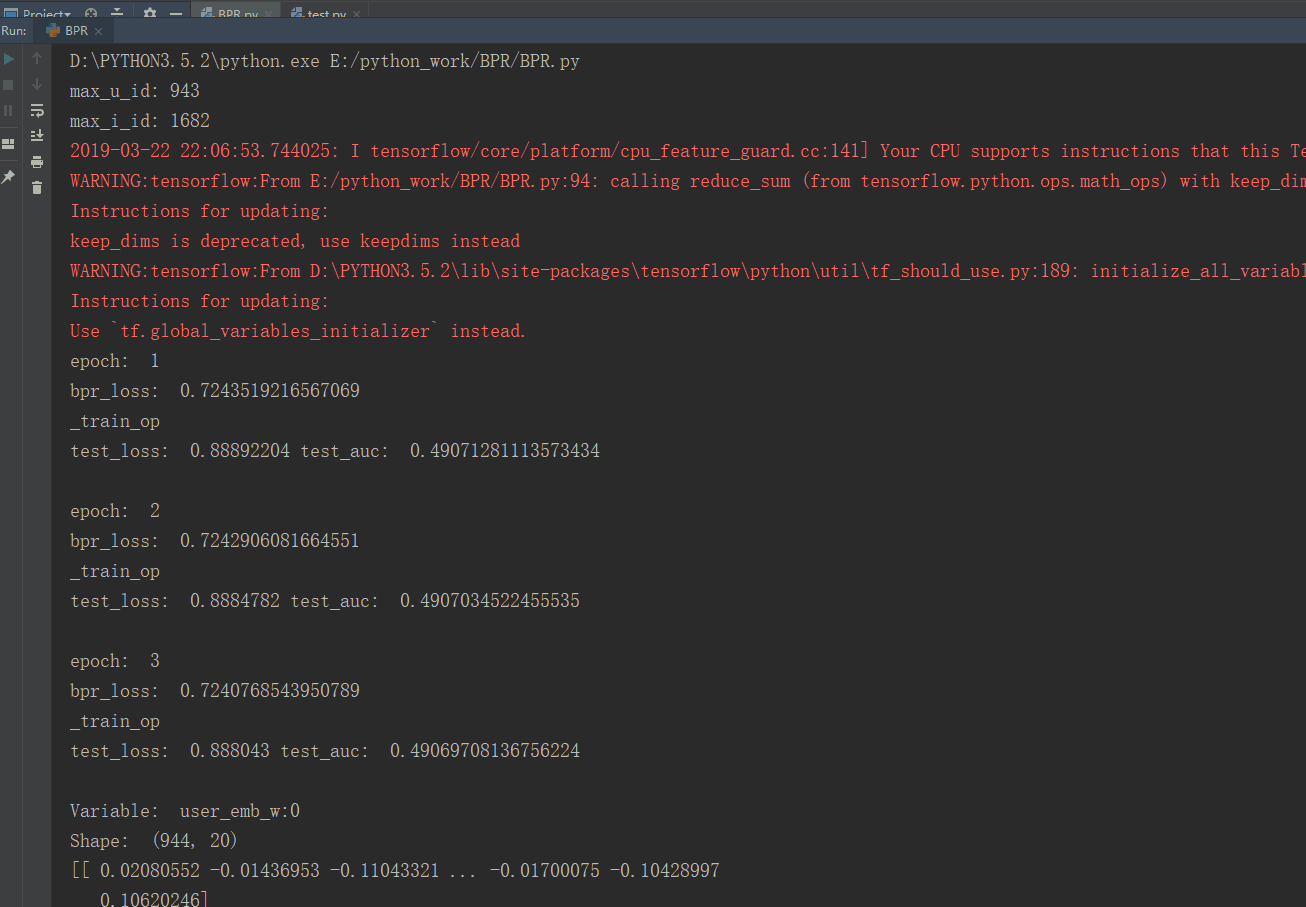
**1）Python版**

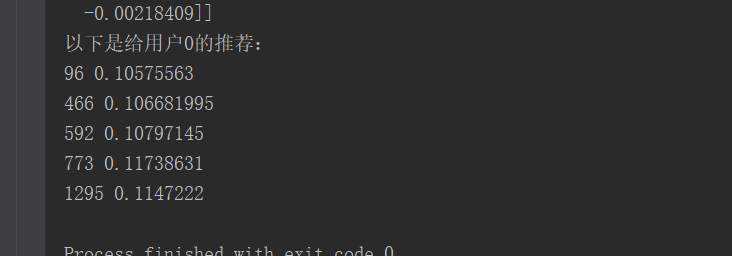


1. **tensorflow版**



### 2.3 运行截图





## 3 总结

训练中发现得bach\_size设定对训练的效果有很大影响。后续应用中还可以优化BPR的损失函数。比如我们可以对这个式子做改进，加上一个基于评分时间的衰减系数，这样我们的排序推荐还可以考虑时间等其他因素。