# ALS（Spark）算法在英雄推荐中的简单应用

## 概述

本文拿《王者荣耀》游戏中玩家所拥有的英雄信息（该信息主要保存在tbHeroInfo表中），用Spark中的交替最小二乘算法（ALS：Alternating Least Square），做一个简易的英雄推荐模型，在测试集上的推荐效果用AUC和MSE两种方式来衡量。

## ALS(Alternating Least Square)算法简介

假设有二维矩阵A，横轴表示玩家id（player\_id），纵轴表示英雄（hero\_id），矩阵中的元素表示玩家是否拥有该英雄，或玩家对该英雄的评分值，不管元素的具体含义是什么，这里抽象为玩家对英雄的偏好，用rating来表示，值越大表示玩家对该英雄的偏好越大。矩阵A的维度一般较高，但是其中的元素确是比较稀疏的，甚至是非常稀疏的。如果根据玩家的已知偏好，要为玩家推荐其可能感兴趣的英雄，这时候就要设法补全二维矩阵A。其中一种补全的办法是把矩阵用两个低维矩阵的乘积表示，即：

A

其中k << min(m, n)，k值可理解为英雄隐含的种类，X矩阵的含义为玩家对不同种类英雄的偏好矩阵，Y矩阵的含义为英雄与不同种类的英雄的关系矩阵。接下来的问题是要如何寻找X、Y矩阵。ALS采用的方法是最小化重构误差，即：

min L(x, y) =

根据上面的公式，结合一些正则化的方法，控制x向量和y向量的大小，进而控制X矩阵和Y矩阵的大小，使模型尽量保持稳定和简单，得到实际的优化目标如下：

min L(x, y) = +

对上面的公式中的x和y分别求导，得到偏导数为0时x和y的表达式，然后再用最优化方法来迭代求解。得到的x的表达式中含有y变量，得到的y表达式中含有x变量，而在两组变量都未知的情况下，在迭代更新其中一个变量是要固定另一个变量。比如在计算x时把y值固定为上一次迭代得到的y值，在计算y时把x的值固定为上一次迭代中求得的x的值。这种方法即为交替最小二乘法（ALS）。

在得到了两个矩阵X和Y之后，就可以代入测试数据进行预测。但需要注意的是，因为得到的X、Y矩阵，分别表示用户、物品与类别的关系，如果在训练数据中缺失某些user\_id或item\_id，则得到的X、Y矩阵中就不会存在这些user\_id、item\_id与类别的关系，而如果测试数据中又含有这些user\_id或item\_id，则无法对该测试数据进行预测。所以在使用ALS的时候，要注意训练集中的用户集合和物品集合的覆盖度要尽量全。如果一开始无法做到这一步的话，那么也应该设置某种机制让新发现的用户或物品不断加入到模型的更新中来。或者换一种做推荐的思路，把推荐的问题转化成点击率预估的问题。

## 数据概况

玩家数目： 254w

英雄数目： 52

打分方法： 如果玩家i拥有某英雄j，则玩家i对英雄j的打分为1.0；

训练和测试数据：对所有的玩家-英雄对进行随机划分，80%为训练数据，20%为测试数据；

## 模型评估方法

### AUC

AUC的物理含义是，任取一对（正、负）样本，正样本的score大于负样本score的概率。

在当前的推荐场景下，由于测试数据中均为正样本，即测试数据中的玩家-英雄对中，玩家确实拥有该英雄。为了验证模型的效果，必须对测试数据的每个玩家加入负样本，即加入玩家没有的英雄，然后模型对正样本和负样本统一打分，好的模型对正样本的打分会高于对负样本的打分。

在模型对正样本和负样本都打过分后，就可以用AUC指标对模型进行评估。AUC越大则模型的效果越好，最大值为1。

### MSE

MSE（Mean Square Error）指的是均方差，其等于预测值与实际值之差的平方和。在本例中，实际值为1.0，预测值为模型计算结果，范围在[0, 1.0]之间。该值越小表示模型的效果越好。

## 测试集评估结果

令lambda参数分别为1.0，0.5，1.5，其余参数相同，得到如下三组评估结果。

表

|  |  |  |  |
| --- | --- | --- | --- |
| rank | lambda正则化因子 | alpha | 迭代次数 |
| 10 | 1.0 | 1.0 | 20 |
| 训练集规模 | | 9857439对 | |
| 正样本测试集规模 | | 2466566对 | |
| 构造的负样本测试集规模 | | 2466494对 | |
| 正样本测试集成功预测的规模 | | 2357398对 | |
| 负样本测试集成功预测的规模 | | 2357326对 | |
| AUC | | 0.884 | |
| MSE | | 0.637 | |

表

|  |  |  |  |
| --- | --- | --- | --- |
| rank | lambda正则化因子 | alpha | 迭代次数 |
| 10 | 0.5 | 1.0 | 20 |
| 训练集规模 | | 9858768对 | |
| 正样本测试集规模 | | 2465237对 | |
| 构造的负样本测试集规模 | | 2465148对 | |
| 正样本测试集成功预测的规模 | | 2356617对 | |
| 负样本测试集成功预测的规模 | | 2356528对 | |
| AUC | | 0.879 | |
| MSE | | 0.450 | |

表

|  |  |  |  |
| --- | --- | --- | --- |
| rank | lambda正则化因子 | alpha | 迭代次数 |
| 10 | 1.5 | 1.0 | 20 |
| 训练集规模 | | 9860402对 | |
| 正样本测试集规模 | | 2463603对 | |
| 构造的负样本测试集规模 | | 2463527对 | |
| 正样本测试集成功预测的规模 | | 2354940对 | |
| 负样本测试集成功预测的规模 | | 2354886对 | |
| AUC | | 0.893 | |
| MSE | | 0.877 | |

通过上面三个表的对比可以看出，在减小lambda的情况下MSE也减小，这是因为提高了在训练数据上的计算精度，但AUC却降低了，从一定程度上说明模型整体的泛化能力也降低了。

## 总结