# 相关技术与理论基础

## 推荐算法理论研究

### 常用向量相似度计算方法

在上文步骤三中提到了计算用户偏好向量和物品向量的相似程度，下面列举出几种常用的相似度计算方式，并对其含义进行说明。

（1）**余弦相似度**：如（式2-1）所示，通过度量两个向量夹角的值来表示向量之间的相似程度。余弦相似度计算简单迅速，并且在高维的情况下依旧适用。

（2）**皮尔逊相关系数**：如（式2-2）所示，在利用余弦相似度进行计算的时候，若用户没有评价的物品分数会被直接判0，这么做的后果会导致原本高分和低分的物品在计算完相似度后特别靠近。而皮尔逊相关系数将用户还未评分的物品填充样本均值，然后再计算余弦相似度。

### 协同过滤的优化—矩阵分解

随着用户和物品的不断增多，由用户-物品所构成的矩阵会以O(N2)的速度不断增大，用户的历史行为数据也随着被稀释。矩阵分解作为协同过滤算法的优化版本，其算法原理如下图2-3所示：

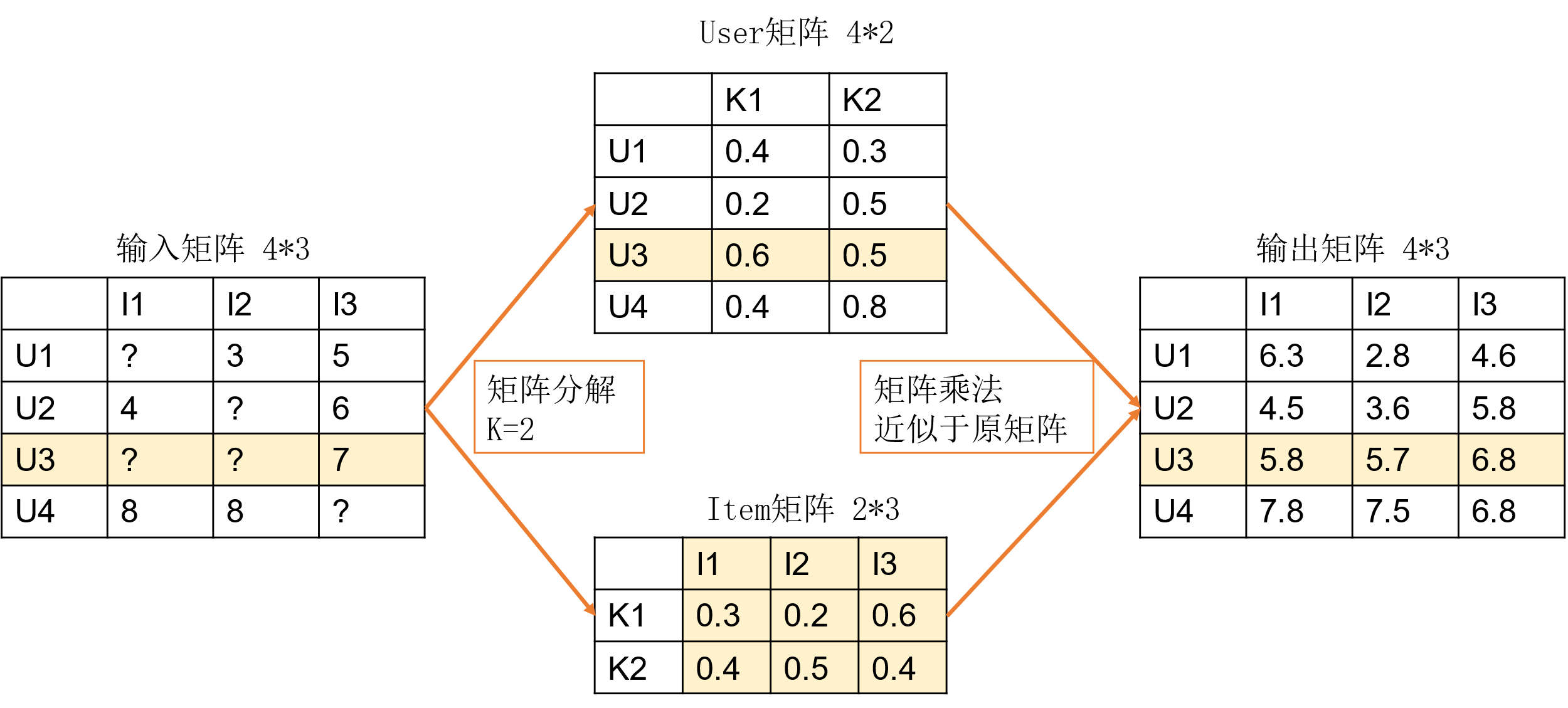


图 2‑3 矩阵分解原理

矩阵分解首先将一个M\*N维的用户-物品矩阵分解成为一个N\*K维的用户矩阵和一个K\*M维的物品矩阵，分解所得到的K1、K2向量被称之为隐向量或隐含特征（Latent Factor），它们被用来将User兴趣和Item特征联系起来。而输出矩阵通过两个分解矩阵相乘后每个单元格都有值，而这些缺失值的填充值就是对用户还未评价物品的预测值。在相乘之后，输出矩阵不一定和原矩阵一致，而我们可以通过多轮训练，利用梯度下降的方法不断调整K1、K2的参数，最终让这个差值降到最小。

矩阵分解模型最终的输出是在分解后最终产生的用户特征向量和物品特征向量，它们都是K个维度，代表着K个不同的隐含兴趣点。得到用户和物品的隐向量之后，我们就可以通过相似度计算公式，轻易的计算出用户与用户之间、用户与物品之间、物品与物品之间的相似程度，形成推荐列表。

若将用户U对物品I的预估评分表示为矩阵相乘的形式，则如（式2-7）所示：其中Qi表示物品i在Item矩阵中所对应的列向量，Pu表示用户u在User矩阵中对应的行向量。

我们求解矩阵分解的目标函数如下（式2-8）所示：其表示了对于K维的用户矩阵p\*和和物品矩阵q\*，让用户已评分和通过隐向量乘积生成的“预估评分”差值最小，将预估的值尽可能的精准。

为了避免部分噪声点的影响，减少过拟合现状，对（式2-8）加入L2正则化项后如（式2-9）所示： 为正则化系数，其值越大表示正则化的能力越强。加入正则化的模型权重变小，使得整个模型变得更加稳定。

在确定损失函数（式2-9）之后，根据梯度下降的流程，对 求偏导的结果如（式2-10）所示：

对求偏导的结果如（式2-11）所示：

利用上式的偏导结果，沿着梯度的反方向更新参数，新的参数等于旧的参数减去学习率\*变化率：称为学习率，也叫做学习步长，刚刚求得的偏导便是参数在这个方向的变化率。更新公式如（式2-12、2-13）所示：

在不断进行梯度下降的过程中，满足两种情况即可终止训练。一是训练的轮数超过了我们预定义的次数；二是损失值（Loss）已经低于一个限度。即可结束训练，否则不断重复上述的梯度反向传播的过程。

下表2-3展示了矩阵分解算法的优缺点

表 2-3 矩阵分解的优缺点分析

|  |  |
| --- | --- |
| **优点** | **缺点** |
| 将高维的矩阵映射成多个低维矩阵的乘积，解决了数据稀疏性的问题 | 推荐结果无法直观的进行解释，其隐空间的维度无法与现实中的概念结合起来 |
| 最终的预测精度比较高 | 模型训练比较耗时，通常以天为粒度更新 |

### Embedding在推荐系统中的应用

（1）Embedding技术是什么

Embedding技术是近几年推荐系统研究的前沿，大量的论文都在攻克这一技术方向。最难能可贵的是，其在真实的应用中收获了非常好的效果。

Embedding从直观上来讲就是一个向量，里面的元素都是小数，如[0.8, 0.9]就是一个二维的Embedding向量，若有100个小数，就是一个一百维的Embedding向量。

（2）Embedding技术怎么用

从实际意义上来讲，每个维度上的小数表示了一个强度或是概率。如数组第一维表示数据结构、第二位表示图论。那么用户Embedding[0.8, 0.9]表示这个人擅长0.8强度的数据结构，0.9强度的图论。物品Embedding[0.4，0.6]表示这道题目有0.4的强度是数据结构、0.6的强度是图论。通过余弦相似度([0.8, 0.9],[0.4, 0.6])就可以计算出该用户做出这道题目的概率。同理，我们也可以轻松的通过运算得出用户之间的相似程度和物品之间的相似程度。但一般来说Embedding的维度都是成百上千维，很难用实际的含义去对应每一维数字代表的含义。虽然无法解释，但是这样的隐含兴趣向量就和2.1.5节中解释的矩阵分解时生成的隐向量一致，可以在推荐领域发挥出巨大的价值。

下表2-4展示了Embedding的应用范围：

表 2-4 Embedding向量的用途

|  |  |
| --- | --- |
| **序号** | **Embedding向量的用途** |
| **用途一：** | 物-物推荐，相关推荐、看了又看、做了又做 |
| **用途二：** | 人-物推荐，猜你喜欢、猜你想看、猜你想练 |
| **用途三：** | 人-人推荐，猜你认识、推荐队友 |

（3）如何生成Embedding向量

Word2Vec

Item2Vec

矩阵分解

（4）如何快速搜索Embedding向量

局部敏感哈希

1. 局部敏感哈希的基本原理局部敏感哈希的基本思想是希望让相邻的点落入同一个“桶”，这样在进行最近邻搜索时，我们仅需要在一个桶内，或相邻几个桶内的元素中进行搜索即可。如果保持每个桶中的元素个数在一个常数附近，我们就可以把最近邻搜索的时间复杂度降低到常数级别。

图 2 中的输入层是由用户 ID 和物品 ID 生成的 One-hot 向量，Embedding 层是把 One-hot 向量转化成稠密的 Embedding 向量表达，这部分就是矩阵分解中的用户隐向量和物品隐向量。输出层使用了用户隐向量和物品隐向量的内积作为最终预测得分，之后通过跟目标得分对比，进行反向梯度传播，更新整个网络。

NeuralCF 用一个多层的神经网络替代掉了原来简单的点积操作。这样就可以让用户和物品隐向量之间进行充分的交叉，提高模型整体的拟合能力。

### 深度学习在推荐系统中的应用

Embedding + MLP

NeuralCF

双塔模型

HDFS的主要组件包括一个namenode和其管理的多个datanode，以及Secondarynamenode检测节点。