协同过滤 & 矩阵分解

先声明一点，仅仅基于**用户行为**(浏览、加购物车、购买)数据设计的推荐算法一般称为**协同过滤算法。**凡是那些考虑了商品本身特性的算法不叫协同过滤。

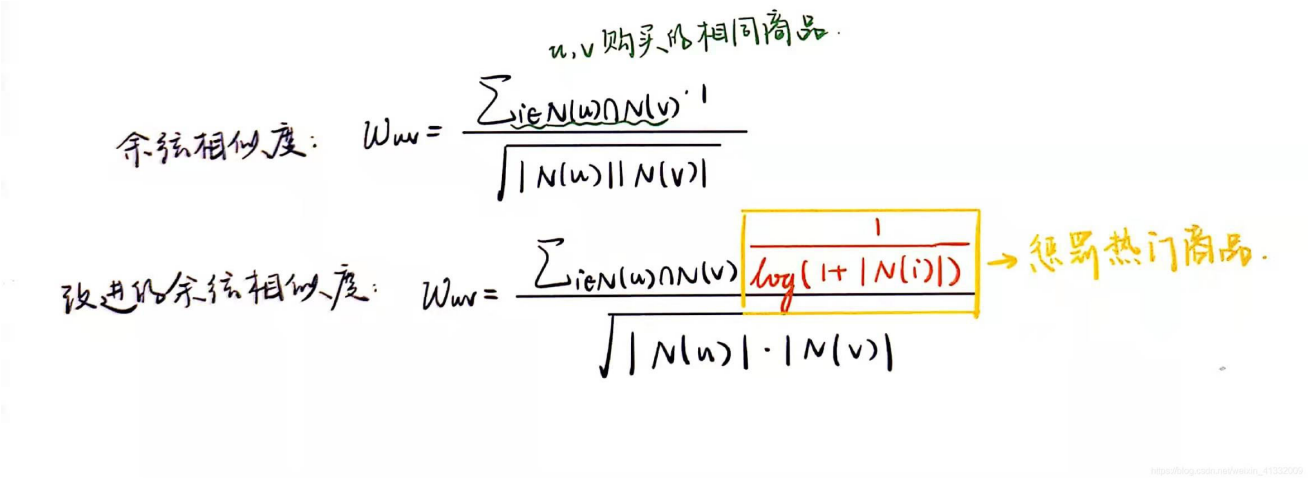
1. 基于用户的协同过滤算法

定义：在一个在线个性化推荐系统中，当一个用户A需要个性化推荐时，可以先找到和他**有相似兴趣的整个用户群**体，然后把这个用户群体喜欢的、而用户A没有听说过的物品推荐给A。

主要包含两个步骤： a. 计算**用户之间的相似度**：找到和用户A兴趣相似的用户群体。 b. 用户群体对物品的喜好程度、以及用户A和用户群体的相似度为每个物品打分：找到这个集合中的用户喜欢的，且目标用户没有听说过的物品推荐给目标用户。

那么首先，怎么计算用户的相似度呢？给定用户u和用户v, 令$N(u)$和$N(v)$分别表示用户u和用户v曾经有过正反馈的物品集合，然后求其Jaccard相似度/余弦相似度。

一种改进的余弦相似度思想是：余弦相似度计算用户兴趣过于粗糙，因为可能有一些非常热门的商品，很多人都会去买，但是这并不代表用户是相似的。所以，**两个用户对冷门物品采取过同样的行为更能说明他们兴趣的相似度。**



计算出若干相似的用户之后，可以召回一批这些用户都喜欢的物品。

2. 基于物品的协同过滤

定义：给用户推荐那些和他们之前喜欢的物品“相似的”物品。（“买过这个商品的人也买了...”）

步骤：

- 计算物品之间的相似度。
- 根据物品的相似度和用户的历史行为给用户生成推荐列表。

step 1. 计算物品的相似度 注意，这里的“物品相似度”指的不是物品本身内容上的相似，而是利用用户的行为数据，即比较对两个物品有过正反馈的**用户集合的相似性。**令N(i)为喜欢物品i的用户集合，改进的余弦相似度：

$$w_{ij} = \frac{\sum_{u \in N(i) \cap N(j)} \frac{1}{log1+|N(u)|}}{\sqrt{|N(i)||N(j)|}}$$

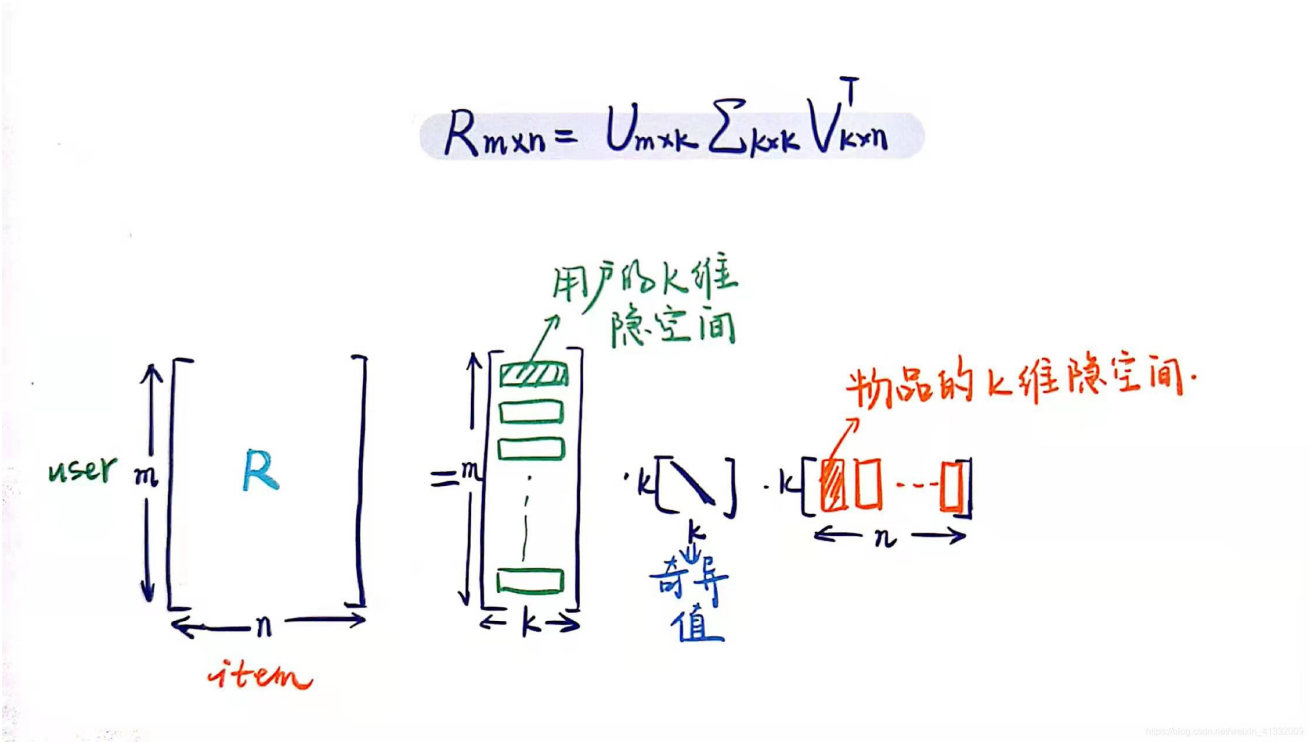
这里也对热门用户进行了惩罚。即：**活跃用户对物品相似度的贡献应该小于不活跃的用户。**

Step2: 为物品打分

在统计完物品之间的相似度之后，就可以利用这种物品相似度以及用户对历史物品的感兴趣程度为未见物品打分。

1. Traditional SVD

对于一个评分矩阵User-Item Matrix，第i行第j列表示第i个用户对于第j个物品的喜爱程度（这个喜爱程度可以通过用户的行为，包括搜索、浏览、加购物车、购买等来表征）。这个矩阵是很**稀疏**的，因为用户对于商品的行为是很不充分的，一个用户对大部分商品的行为根本没有记录。我们的任务是要通过分析已有的数据（观测数据）来对未知数据进行预测（预测某个用户u对于他根本没见过的商品i会有多少的喜爱程度），即这是一个矩阵补全（填充）任务。矩阵填充任务可以通过矩阵分解技术来实现。核心思想是将用户和商品映射到一个共同的k维隐空间，使得在这个隐空间中（获得**稠密向量**），用户对商品的喜爱程度可以使用向量内积来计算。这一隐空间借助从评分矩阵自动推断得到的隐因子，来刻画用户和商品，以解释用户对商品的评分行为。



2. ALS（Alternating Least Squares）

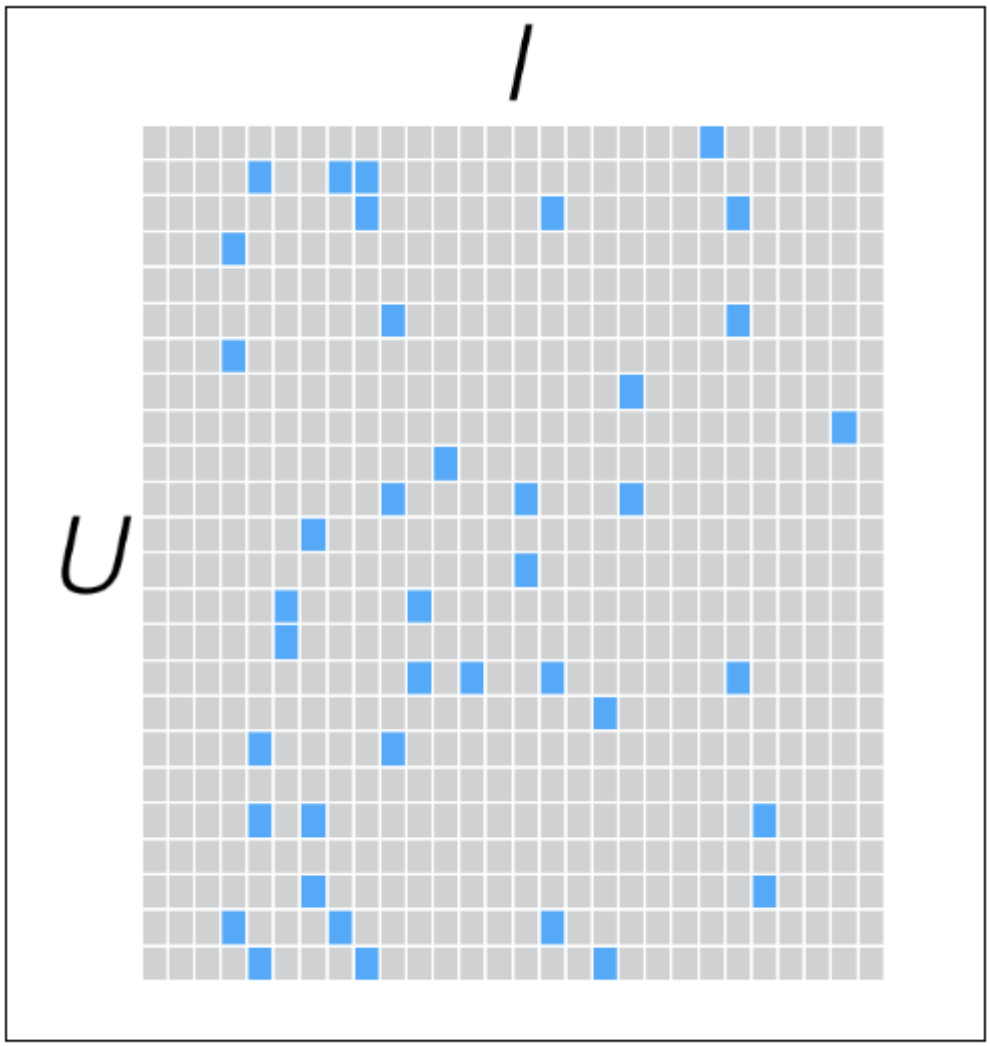
2.1 矩阵分解

ALS算法是2008年以来，用的比较多的协同过滤算法。它已经集成到Spark的Mllib库中。

从协同过滤的分类来说，ALS算法属于 User-Item CF，也叫做混合CF，它同时考虑了User和Item两个方面。用户和商品的关系，可以抽象为如下的三元组：<User,Item,Rating>。其中，Rating是用户对商品的评分，表征用户对该商品的喜好程度。

假设我们有一批用户数据，其中包含m个User和n个Item，则我们定义矩阵R，其中第u行第i列表示第u个User对第i个Item的评分。

在实际使用中，由于n和m的数量都十分巨大，因此R矩阵的规模很容易就会突破1亿项。这时候，传统的矩阵分解方法对于这么大的数据量已经是很难处理了。 另一方面，一个用户也不可能给所有商品评分，因此，R矩阵注定是个**稀疏**矩阵。

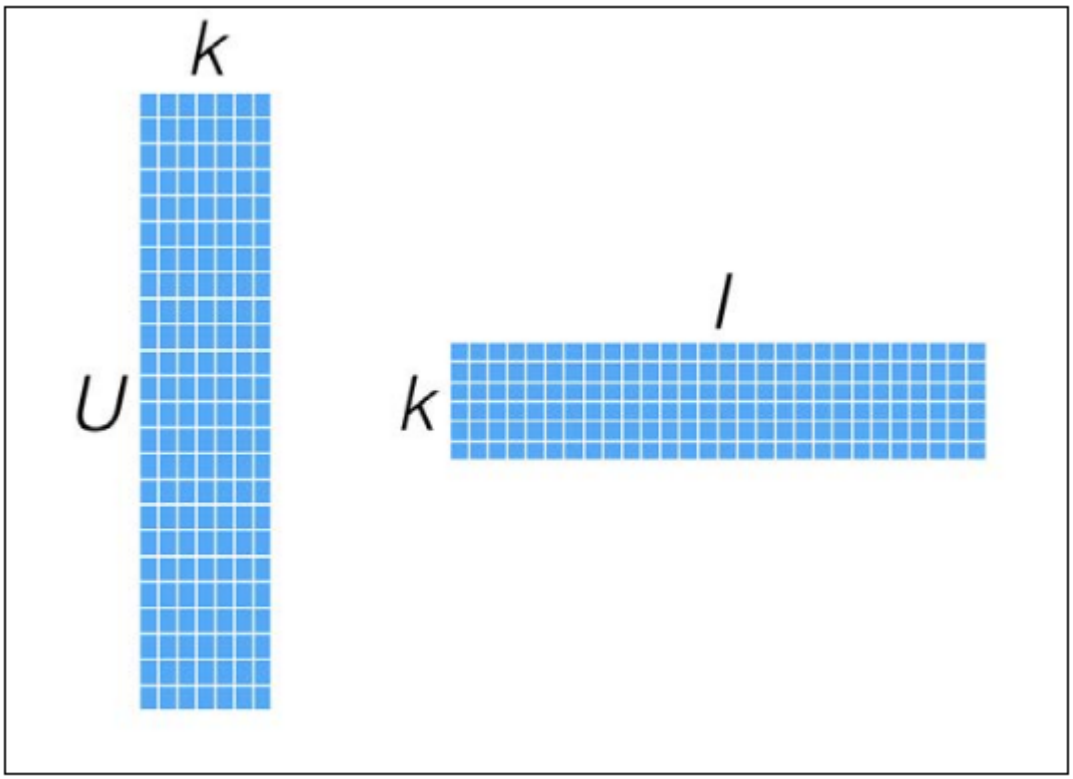


A sparse ratings matrix

FunkSVD/ALS不再将矩阵分解为3个矩阵，而是分解为2个**低秩**的用户和商品矩阵。

$$R_{m\times n} \approx X_{m\times k} Y^T_{n\times k}$$

一般情况下，k的值远小于n和m的值，从而达到了数据降维的目的。这里的表明这个投射只是一个近似的空间变换。



The user- and item-factor matrices

为了使低秩矩阵X和Y尽可能地逼近R，需要最小化下面的平方误差损失函数：

$$\min_{x_*,y_*} \sum_{u,i \text{ is known}} (r_{ui} - x_u^T y_i)^2$$

考虑到矩阵的稳定性问题，使用 L2 正则化项，则上式变为：

$$\min_{x_*,y_*} L(X,Y) = \min_{x_*,y_*} \sum_{u,i \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(|x_u|^2 + |y_i|^2)$$

ALS算法的缺点在于：

1.它是一个离线算法。

2.无法准确评估新加入的用户或商品。这个问题也被称为Cold Start问题。