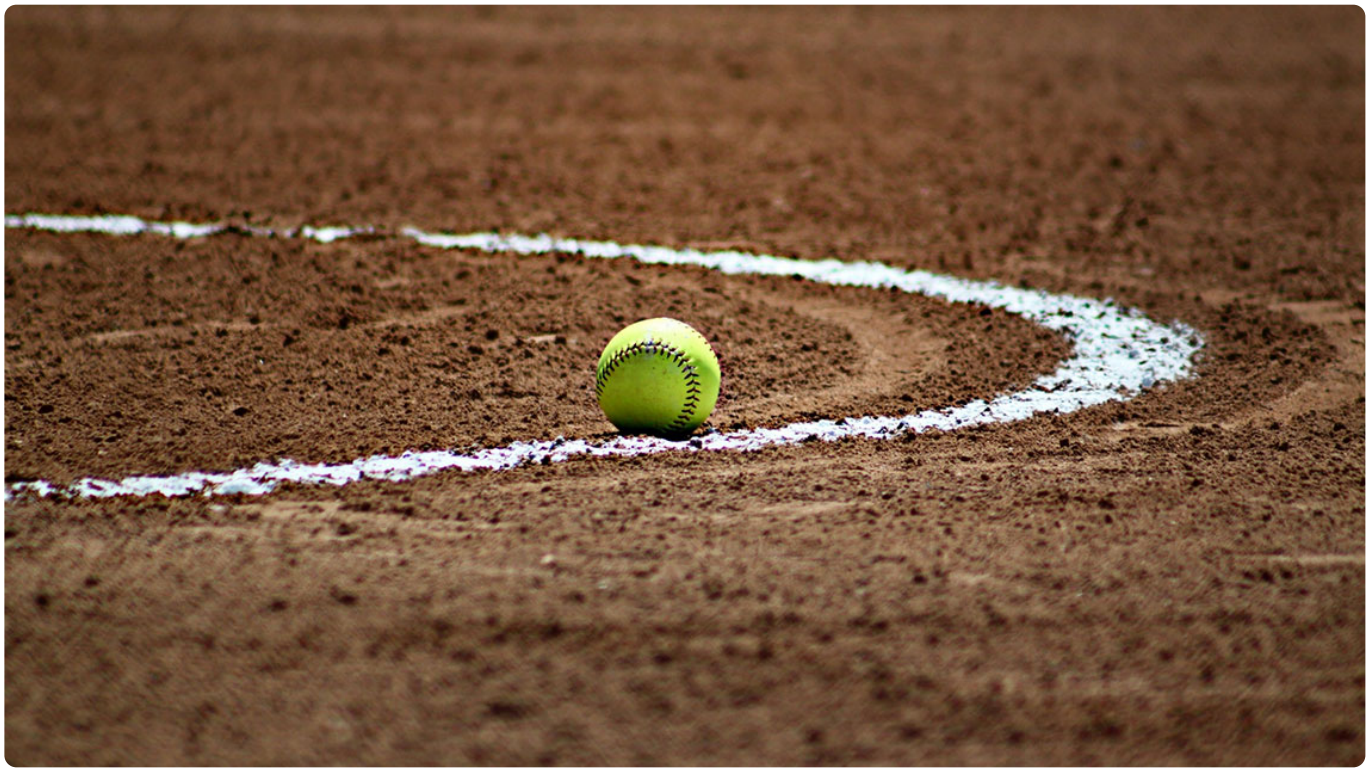
066 | 基于隐变量的模型之一：矩阵分解

2018-03-05 洪亮劼

AI技术内参 进入课程 >



讲述：初明明
时长 07:06 大小 3.25M

上周我们聊了三个简单的推荐模型，分别是基于流行度的推荐模型，基于相似信息的推荐模型和基于内容特征的推荐模型。

这周，我们来看一类非常重要的推荐模型：**基于隐变量的推荐模型**。这类模型的优势是对用户和物品信息中的隐含结构进行建模，从而能够挖掘更加深层次的用户和物品关系。

什么是隐变量

在解释如何用隐变量来生成推荐结果之前，我们先来说一下什么是隐变量。

隐变量（Latent Variable），顾名思义，就是“隐藏的变量”或者叫“隐藏的参数”，这里主要是指我们假定实际的数据是由一系列的隐含变量产生的。我们通过模型的假设，知道

隐变量之间的关系，但暂时并不知道隐变量的取值。因此需要通过“推断”（Inference）过程来确定隐变量的实际取值。当我们知道了这些隐变量的取值之后，就可以根据这些取值来对未来的数据进行预测和分析。

隐变量往往还带有“统计分布”（Distribution）的假设。什么意思呢？就是隐变量之间，或者隐变量和显式变量之间的关系，我们往往认为是由某种分布产生的。

举一个最简单的隐变量模型的例子，那就是“**高斯混合模型**”（Mixture of Gaussian）。

高斯混合模型假设数据是由多个不同的高斯分布产生的，每一个高斯分布有自己的均值和方差。在最简单的两个高斯的情况下，每一个数据点，都有可能是由这两个高斯分布中的一个产生的，但是，究竟是哪一个我们并不知道。于是，对于每一个数据点，我们就有一个隐含的变量，来表达当前这个数据点究竟来自哪一个高斯分布。

很明显，这个隐含变量的取值事先并不知道。除了这个隐含变量我们不知道以外，两个高斯分布的均值和方法其实也不知道。于是，对于高斯混合模型来说，整个学习的过程就需要估计每个数据点的来源以及多个高斯分布的均值和方差。**高斯混合模型，几乎是最简单的隐变量模型，但也给我们了展示了使用隐变量模型对数据建模的灵活性以及训练的复杂性。**

矩阵分解作为隐变量模型

了解了隐变量模型的基本的概念以后，我们还是回到推荐的场景。

在推荐系统中，有一种最普遍的数据表达，那就是用户和物品的交互，比如评分、点击等等。这里我们用评分作为一般性的讨论。对于每一个用户，如果我们用一个向量来表达其对所有可能物品的评分，那么把所有用户的向量堆积起来，就可以得到一个矩阵。这个矩阵的每一行代表一个用户，每一列代表一个物品，每一个交叉的元素代表某一个用户对于某一个商品的评分。

那么这个矩阵有什么特点呢？最大的特点就是，这个矩阵的数据其实非常稀少。因为在一个现实的系统中，一个用户不可能对所有的物品都进行评分。实际上，一个用户仅仅对非常少的物品进行过评分，而对绝大多数物品并不会评分，甚至连评分的打算都没有。因此，这个矩阵的绝大多数元素其实是 0。当然，实际上这些元素也不是 0，而是未知，因为用户并不是觉得这些物品的评分是 0，而是压根没有对这些物品进行打分，也就是说评分信息不完备。

对于这样一个矩阵，我们的任务其实就是根据有评分的用户物品交互，对那些还没有评分信息的物品进行预测。如果我们能够“补全”（Complete）整个矩阵里的其他元素，那么就可以根据预测的评分从大到小给用户进行推荐。

怎么来补全这个矩阵的信息呢?

这里，我们假设，矩阵的每一个元素都是经过这样一个过程产生的。

首先，我们假设每一个用户和每一个物品都对应一个隐向量（Latent Factor）。比如，我们用 100 维的向量来表达一个用户，用另外一个 100 维的向量来表达一个物品。如果我们有 1 百万个用户和 1 万个物品，那么我们就需要 1 百万个 100 维的用户向量，和 1 万个 100 维的物品向量。

然后，我们假设矩阵的每一个元素是由所对应的用户和物品的隐向量点积得到的。也就是说，矩阵里的每一个元素，都是一个 100 维的用户向量和一个 100 维的物品向量对应元素相乘相加而得。

在这样的一个假设下，一个原本 1 百万乘以 1 万的矩阵就可以被分解为 1 百万乘以 100 的用户矩阵和 100 乘以 1 万的物品矩阵的乘积。这也就是为什么在这样的模型下，我们会称这个方法为“**矩阵分解**”（Matrix Factorization）的原因。

在矩阵分解这个假设下，我们可以看到，原本是需要对整个矩阵，也就是 1 百万乘以 1 万个数据进行建模，而现在缩减到了两个小一些的矩阵 1 百万乘以 100 和 100 乘以 1 万。对比来看，之前对于每一个用户，我们需要一万个元素来表达用户对于这一万个物品的评分喜好；现在，对每个用于仅仅需要保存 100 个元素。

只不过，这一百个元素的数值并不代表任何意义。从这个层面上理解矩阵分解，也就能够帮助我们知道，这个方法其实是一种“**降维**”（Dimension Reduction），也就是把一个比较大的矩阵分解成两个小矩阵来表达。

我们可以看到，矩阵分解的核心其实就是刚才的假设，用隐向量来表达用户和物品，他们的乘积关系就成为了原始的元素。不同的假设可以得到不同的矩阵分解模型。

学习矩阵分解模型

很明显，矩阵分解仅仅告诉了我们这个模型的表达，但并没有告诉我们怎么去获得整个模型的最核心内容，那就是用户矩阵和物品矩阵里每一个元素的数值。也就是我们所说的，用户隐向量和物品隐向量都是事先不知道的。

那有什么办法能够得到这些向量的取值呢？

这里介绍一种常用的方法，就是利用**最小二乘法**的原理（Least Square）来拟合求解这些隐向量。

前面讲到，矩阵里的每一个元素来自于两个隐向量的点积，我们就可以利用这一点来构造一个目标函数。这个目标函数其实就是说，这两个隐向量的点积一定要与我们观测到的矩阵数值相近。这里的“相近”是用这两个数值的误差，在这里也就是**平方差**（Square Error）来衡量的。误差越小，说明数据拟合得越好，隐向量也就更能表达用户和物品。

构造好了这个目标函数以后，我们就可以利用**优化算法**来求解这个目标函数。因为两个未知数点积的原因，这个目标函数没有一个全局的最优解，但是我们可以通过“**梯度下降**”（Gradient Descent）的方法找到一个不错的局部解（Local Minima）。

小结

今天我为你讲了推荐系统的一个重要分支，隐变量模型。我们讲了其中最重要的一个基本模型，矩阵分解。

一起来回顾下要点：第一，我们简要介绍了隐变量模型的基本原理；第二，我们详细介绍了矩阵分解作为隐变量模型的假设和原理；第三，我们简要地讨论了如何求解矩阵分解里的隐变量。

最后，给你留一个思考题，矩阵分解模型最大的问题是什么？

欢迎你给我留言，和我一起讨论。

上一篇　065 | 简单推荐模型之三：基于内容信息的推荐模型

下一篇　067 | 基于隐变量的模型之二：基于回归的矩阵分解

精选留言 (2)

写留言

林彦
2018-03-05

3

基于隐变量的矩阵分解有如下缺点：
推荐结果不具有很好的可解释性，分解出来的用户和物品矩阵的每个维度无法和现实生活中的概念来解释，无法用现实概念给每个维度命名，只能理解为潜在语义空间。

除了上面一点，现实的评分矩阵特别稀疏。为了使得数据更加稠密，可以加入了历史的...
展开

极客星星
2018-03-06

矩阵分解方法应用时 经常遇到的问题是 只有正样本 缺少负样本 针对这个问题有一些策略

如BPR 等