知乎

想要了解推荐系统？看这里！（1）——协同过滤与奇异值分解

第四范式...
已认证的官方帐号

关注他

10 人赞同了该文章

> 本账号为第四范式智能推荐产品先荐的官方账号。账号立足于计算机领域，特别是人工智能相关的前沿研究，旨在把更多与人工智能相关的知识分享给公众，从专业的角度促进公众对人工智能的理解；同时也希望为人工智能相关人员提供一个讨论、交流、学习的开放平台，从而早日让每个人都享受到人工智能创造的价值。
> 以下内容由第四范式先荐团队编译，原文发布于**Medium**，作者**Steeve Huang**。转载内容仅用于学习交流，版权归原作者所有。

1.背景

推荐系统能够预测用户未来的兴趣偏好，并向用户推荐排序最靠前的item。在现代社会中，推荐系统之所以被人们需要，关键是由于互联网的普及，人们面临着太多的选择。过去，人们习惯在实体店购物，实体店的商品有限，人们的选择也有限。相比之下，如今互联网允许人们在线访问任何资源。尽管可用信息量增加，但人们很难选出他们真正想要的东西，问题来了——这就是推荐系统的用武之地。

知乎

传统上，构建推荐系统有两种方法：

- 基于内容的推荐
- 协同过滤

第一种方法需要分析每一项的属性。例如，对每位诗人的诗歌作品进行自然语言处理，然后向用户推荐诗人。但是，协同过滤却不需要有关项或用户本身的任何信息，而是根据用户的过去行为进行推荐。以下我们将重点讲解协同过滤。

3.协同过滤

如上所述，协同过滤（CF）是基于用户过去行为的推荐平均值。协同过滤可分两类：

- 基于用户：测量目标用户与其他用户之间的相似性
- 基于项目：测量目标用户评价/交互过的项目与其他项目之间的相似性

同过滤背后的关键思想是相似的用户共享相同的兴趣，并且喜欢相似的项目。

假设现有m个用户和n个项目，我们用m * n的矩阵来表示用户过去的行为。矩阵中的每个单元格表示用户持有的相关反馈，例如，M {i，j}表示用户i对项目 j的喜欢程度，这种矩阵称为效用矩阵（utility matrix）。

基于用户的协同过滤

基于用户的协同过滤首先需要计算用户之间的相似性。计算相似性有两种办法，皮尔逊相关系数或余弦相似度。设u {i，k}表示用户i和用户k之间的相似度，v {i，j}表示用户i对项目j的评级，如果用户没有评定该项目，则v_ {i，j} =？。

这两种方法可以表示如下：

1. 皮尔逊相关系数 (Pearson Correlation)

知乎

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}}$$

2. 余弦相似度（Cosine Similarity）

$$\cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

这两种方法很常用，但区别在于用皮尔逊相关系数计算的话，向所有元素添加常量不变。

现在，我们可以使用以下等式预测用户对未评级项目的反馈意见：

$$v_{ij}^{*} = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj}$$

未评级项目预测

在以下矩阵中，每行代表一个用户，而列对应于不同的电影，每个单元格代表用户为该电影提供的

知乎

A	2		2	4	5		NA
B	5		4			1	
C			5		2		
D		1		5		4	
E			4			2	1
F	4	5		1			NA

由于用户A和F不共享与用户E共同的任何电影评级，因此他们与用户E的相似性未在中皮尔逊相关系数中定义。因此只需要考虑用户B、C和D。基于皮尔逊相关系数，计算如下：

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E			4			2	1
F	4	5		1			NA

从上表可以看出，用户D与用户E差异很大，二者之间的皮尔逊相关性为负。用户D对电影*Me Before You*的评价高于他的评分平均值，而用户E的表现则相反。

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E	3.51*	3.81*	4	2.42*	2.48*	2	1
F	4	5		1			NA

基于用户的协同过滤虽然很简单，但它存在几个问题。一个主要问题是用户的偏好会随着时间的推移而改变，这意味着基于其相邻用户预先计算矩阵这种方法可能导致推荐效果不准确。为了解决这

知乎

基于项目的协同过滤并不计算用户之间的相似性，而是基于用户与目标用户评定的项目之间的相似性来推荐项目。同样，可以使用皮尔逊相关系数或余弦相似度计算相似度。主要区别在于，使用基于项目的协同过滤，我们垂直填补空白，与基于用户的协同过滤的水平方式相反。下表展示了如何为电影*Me Before You*执行此操作。

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Similarity	-1	-1	0.86	1	1	

这种方法成功地避免了动态的用户偏好带来的问题，因为基于项目的CF更加静态。但是，这种方法仍然存在一些问题。第一个问题是可扩展性。数据计算量随用户和项目的增长而增长。最坏的情况复杂度是O（mn），即m个用户和n个项目。稀疏性是另一个问题。在上表中，虽然只有一个用户对*Matrix*和*Titanic*都进行了评级，但它们之间的相似性为1。在极端情况下，我们可能拥有数百万用户，两部截然不同的电影之间的相似性也可能非常高，因为唯一一个对这两部电影评分的用户给了它们相似的评分。

4. 奇异值分解（Singular Value Decomposition）

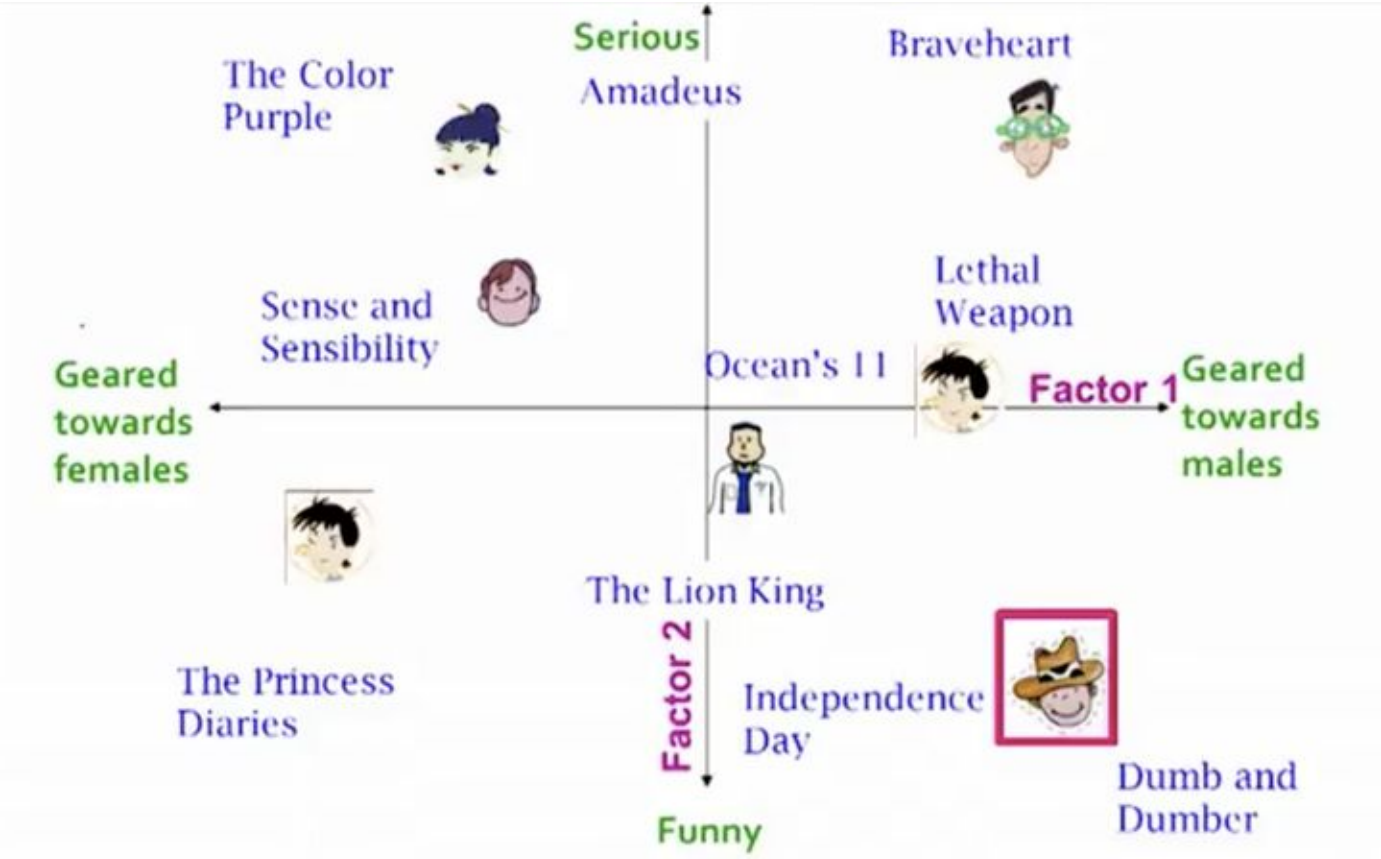
处理CF中可扩展性和稀疏性问题的方法之一是利用潜在因子模型来捕获用户和项之间的相似性。从本质上讲，我们希望将推荐问题转化为优化问题，把它看做系统预测用户给定项目的评级的性能。

一个常见的指标是均方根误差（RMSE，Root Mean Square Error）。 RMSE越低，性能越好。由于我们不知道不可见项目的评级，因此会暂时忽略它们。也就是说，我们只在效用矩阵中的已知条目上最小化RMSE。为了实现最小RMSE，采用奇异值分解（SVD）如下面的公式所示。

$$\underset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \cdot & \cdot & \cdot & \end{pmatrix}} \approx \underset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \end{pmatrix}} \underset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \end{pmatrix}} \underset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \end{pmatrix}}$$

知乎

X表示效用矩阵，U是左奇异矩阵，表示用户和潜在因子之间的关系。 S是描述每个潜在因子强度的对角矩阵，而V转置是右奇异矩阵，表示项和潜在因子之间的相似性。什么是潜在因子呢？这是一个广义的概念，用来描述用户或项目具有的属性。以音乐为例，潜在因子可以指音乐所属的类型。 SVD通过提取其潜在因子来减小效用矩阵的维数。基本上，我们将每个用户和每个项目映射到维度为r的潜在空间。潜在空间有助于我们更好地理解用户和项目之间的关系，如下所示：



SVD将用户和项目映射到潜在空间

SVD具有最小的重建平方误差和（SSE，Sum of Square Error）;因此，它也常用于降维。在下面的公式中，用A代替X，用Σ代替S。

$$\min_{U,V,\Sigma} \sum_{ij \in A} \left(A_{ij} - [U\Sigma V^T]_{ij}\right)^2$$

平方误差总和

这与开头提到的RMSE有什么关系呢？事实证明，RMSE和SSE呈单调相关。这意味着SSE越低，

知乎

Python Scipy为稀疏矩阵提供了很好的SVD实现。

```
>>> from scipy.sparse import csc_matrix
>>> from scipy.sparse.linalg import svds
>>> A = csc_matrix([[1, 0, 0], [5, 0, 2], [0, -1, 0], [0, 0, 3]],
dtype=float)
>>> u, s, vt = svds(A, k=2) # k is the number of factors
>>> s
array([ 2.75193379,  5.6059665 ])
```

SVD成功解决了CF带来的可扩展性和稀疏性问题，然而SVD并非没有缺陷。 SVD的主要缺点是向用户推荐项目的原因几乎没有解释。如果用户想知道为什么要向他们推荐特定项目，这可能是一个很大的问题。这个问题将会在下一篇文章中提到。

5.结论

本文讨论了构建推荐系统的两种典型方法，协同过滤和奇异值分解。 在下一篇文章中，将继续讨论一些用于构建推荐系统的更高级算法。

相关阅读：

AutoML如何实现智能推荐系统的自动上线和运维？

如何实现AI赋能新媒体的技术落地？

AutoML在推荐系统中的应用

入门推荐系统，你不应该错过的知识清单

如欲了解更多，欢迎搜索并关注官方微博@先荐、微信公众号（**ID：dsfsxj**）。

编辑于 2019-06-17