--文·博客虫

关于推荐系统的基础整理，是对于部门内部交流培训学习“推荐系统基础”的一个整理，比较基础。And，这基本是技术知识~~

1 个性化推荐概述

1.1 推荐系统概述

首先，需要申明一点的就是推荐系统！=推荐算法。推荐系统是一套完善的推荐机制，包括前期数据的准备、具体推荐的过程(这个过程可能是一套复杂的算法模型，也可能是一个简单的规则，也可能是多种模型的混合结果等等)、后期数据的预测、AB测试效果评估等等。

1.2 推荐算法模型概述

在算法模型上大体可以分基于内容的推荐、基于协同过滤的推荐。

基于内容推荐，即通过内容本身的属性，然后计算内容的相似性，找到与某物品属性相似的物品。协同过滤，所谓协同过滤，即不依赖于物品本身的物品属性，而是通过其他相关特征，例如人参与的e行为数据，来达到推荐物品的目的。

关于协同过滤，又分为以下几个类别：基于物品的协同，即ItemCF；基于用于的协同，即UserCF；基于模型的协同，即ModelCF。

其中，基于模型的协同又可以分为以下几种类型：基于距离的协同过滤；基于矩阵分解的协同过滤，即Latent Factor Model(SVD)；基于图模型协同，即Graph，也叫社会网络图模型。

2 基于内容的推荐

其实但从字面上也好理解，其推荐的依据为物品的内容，即物品的具体相关属性。换言之，即我们希望找到的是跟当前物品相似的物品。

那么，我们的目标就明确了，计算当前物品与其他物品的相似度，然后生成一个相似TopN列表，然后想要多少个推荐多少个。

通常我们会有以下两种方式来计算相似度：通过物品间的距离去度量相似；通过直接计算相似度。

2.1 计算物品距离的几种方法

(1) 欧几里得距离(Euclidean Distance)

最常见的距离度量方式，衡量多维空间中两点之间的绝对距离，要求维度的统一。

(2) 明可夫斯基距离(Minkowski Distance)

明氏距离是欧氏距离的扩展，是对多个距离度量公式的概括性的表述(可以看到，当p=2时，其实就是欧式距离)。

(3) 曼哈顿距离(Manhattan Distance)

曼哈顿距离来源于城市区块距离，是将多个维度上的距离进行求和后的结果，即当上面的明氏距离中p=1时得到的距离度量。

//还有其他的一些距离度量，但是都不太常用，最常用的依然是欧式距离度量。

2.2 计算相似度量的几种方法

(1) 向量空间余弦相似度(Cosine Similarity)

余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。相比距离度量，余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。

(2) 皮尔森相关系数(Pearson Correlation Coefficient)

即相关分析中的相关系数r，分别对X和Y基于自身总体标准化后计算空间向量的余弦夹角。

基于内容的推荐，还有一点需要注意的就是，对于物品自身属性，如果属性值过少，我们需要适当进行扩大维度，如果维度过多，则需要进行降维。

关于降维和升维，都是一个很大的研究方向，大体上可以说一下几种常见的方式。例如降维，我们可以进行维度聚类、主题抽取，进一步把相关维度进行合并，进一步减少维度；而对于升维，我们可以把维度进行矩阵化，例如假设物品X有A和B两个维度属性，那么我们通过生成A*B矩阵的方式，把维度扩充到A*B个维度。

3 基于协同过滤的推荐

3.1 基于用户的协同(UserCF)

基于用户的协同过滤，即我们希望通过用户之间的关系来达到推荐物品的目的，于是，给某用户推荐物品，即转换为寻找为这个用户寻找他的相似用户，然后相

似用户喜欢的物品，那么也可能是这个用户喜欢的物品(当然会去重)。

来看一个表格：

用户/物品	物品A	物品B	物品C	物品D
用户A	Y	?	Y	?
用户B	-	Y	-	-
用户C	Y	-	Y	Y

//其中Y表示对应用户喜欢对应物品，-表示无交集，？表示需不需要推荐。

这是一个最简单的例子，其实目的很简单，我们需要给用户A推荐物品，而且可以看到，用户已经喜欢了物品A和物品C，其实剩下也就B和D了，要么是B，要么是D。

那么根据UserCF算法，我们先计算用户A与用户BC之间的相似度，计算相似，我们前文说了，要么距离，要么余弦夹角。

假如我们选择计算夹角(四维)：cosAB=0(90度的夹角)，cosAC=0.8199(角度自己算吧)。所以相比来说，我们会发现用户A与用户C的相似度是稍微大一些的。于是，我们观察用户C都喜欢了哪些物品，然后与用户的去重，然后会发现该给用户A推荐物品D。

简单来讲，UserCF就是如上过程，但在实际的过程中，数据量肯定不止这么点，于是我们需要做的是为用户计算出相似用户列表，然后在相似用户中经过去重之后，计算一个推荐的物品列表(在计算推荐物品的时候，可以叠加用户的相似程度进一步叠加物品的权重)。

然后在喜欢物品的表达形式上，可以是如上的这种二值分类，即Yes Or No，也可以是带有评分的程度描述，比如对于某个物品打多少分的这种表现形式。这样的话，针对于后一种情况，我们就需要在求在计算相似度时，加入程度的权重考量。

3.2 基于物品的协同(ItemCF)

不同于基于用户的协同，这里，我们计算的是物品之间的相似度，但是，请注意，我们计算物品相似度的时候，与直接基于物品相似度推荐不同是，我们所用的特征并不是物品的自身属性，而依然是用户行为。

用户/物品	物品A	物品B	物品C

用户**A**	Y	-	Y
用户**B**	Y	Y	Y
用户**C**	Y	?	?

//其中Y表示对应用户喜欢对应物品，-表示无交集，？表示需不需要推荐。

同样，这是一个简单实例。目的也明确，我们在知道用户AB喜欢某些物品情况，以及在用户C已经喜欢物品C的前提下，为用户C推荐一个物品。

看表格很简单嘛。只有两个选项，要么物品B，要么物品C。那么到底是物品B还是物品C呢？

我们来计算物品A与其他两种物品的相似度，计算向量夹角。对于用户A，物品A与物品B，则对于AB向量为(1,0),(1,1)，对于AC向量为(1,1),(1,1)，分别计算夹角cosAB=0.7,cosAC=1。或者用类似关联规则的方法，计算两者之间的共现，例如AB共现1次，AC共现2次。通过类似这种方式，我们就知道物品A与物品C在某种程度上是更相似的。

我要说的就是类似共现类做计算的这种方式，在大规模数据的情况下是很有效的一种方式，基于统计的方法在数据量足够的时候，更能体现问题的本质。

3.3 基于模型的协同(ModelCF)

除了我们熟悉的基于用户以及基于物品的协同，还有一类，基于模型的协同过滤。基于模型的协同过滤推荐，基于样本的用户偏好信息，训练一个模型，然后根据实时的用户喜好信息进行预测推荐。

常见的基于模型推荐又有三种：最近邻模型，典型如K最近邻；SVD模型，即矩阵分解；图模型，又称为社会网络图模型。

(1) 最近邻模型

最近邻模型，即使用用户的偏好信息，我们计算当前被推荐用户与其他用户的距离，然后根据近邻进行当前用户对于物品的评分预测。

典型如K最近邻模型，假如我们使用皮尔森相关系数，计算当前用户与其他所有用户的相似度sim，然后在K个近邻中，通过这些相似用户，预测当前用户对于每一个物品的评分，然后重新排序，最终推出M个评分最高的物品推荐出去。

需要注意的是，基于近邻的协同推荐，较依赖当前被推荐用户的历史数据，这样计算出来的相关度才更准确。

(2) SVD矩阵分解

我们把用户和物品的对应关系可以看做是一个矩阵X，然后矩阵X可以分解为X=A*B。而满足这种分解，并且每个用户对应于物品都有评分，必定存在与某组隐含的因子，使得用户对于物品的评分逼近真实值，而我们的目标就是通过分解矩阵得到这些隐性因子，并且通过这些因子来预测还未评分的物品。

有两种方式来学习隐性因子，一为交叉最小二乘法，即ALS；而为随机梯度下降法。

首先对于ALS来说，首先随机化矩阵A，然后通过目标函数求得B，然后对B进行归一化处理，反过来求A，不断迭代，直到A*B满足一定的收敛条件即停止。

对于随机梯度下降法来说，首先我们的目标函数是凹函数或者是凸函数，我们通过调整因子矩阵使得我们的目标沿着凹函数的最小值，或者凸函数的最大值移动，最终到达移动阈值或者两个函数变化绝对值小于阈值时，停止因子矩阵的变化，得到的函数即为隐性因子。

使用分解矩阵的方式进行协同推荐，可解释性较差，但是使用RMSE(均方根误差)作为评判标准，较容易评判。

并且，我们使用这种方法时，需要尽可能的让用户覆盖物品，即用户对于物品的历史评分记录需要足够的多，模型才更准确。

(3) 社会网络图模型

所谓社会网络图模型，即我们认为每个人之间都是有联系的，任何两个用户都可以通过某种或者多个物品的购买行为而联系起来，即如果一端的节点是被推荐用户，而另一端是其他用户，他们之间通过若干个物品，最终能联系到一起。

而我们基于社会网络图模型，即研究用户对于物品的评分行为，获取用户与用户之间的图关系，最终依据图关系的距离，为用户推荐相关的物品。

目前这种协同推荐使用的较少。

4 其他相关知识

4.1 冷启动

所谓冷启动，即在推荐系统初期时，没有任何用户与物品的交集信息，即无用户的行为轨迹，无法通过类似协同的方式进行过滤推荐，这种时候，我们就称推荐系统处于冷启动状态。

这种情况，我们需要尽快的累积起第一批用户行为轨迹。我们可以通过基于内容的推荐，或者做一些其他类似的操作，快速有效的进行物品推荐。
一段时间后，累积到一定的用户行为时，整个系统就能够正常使用协同过滤等方式进行推荐了。
但是，针对于新加入的用户，或者新加入的物品，同样也是出于冷启动状态的，这个时候，我们通过需要对这种物品或者用户做特殊的处理。

4.2 长尾效应/马太效应

所谓长尾效应，在推荐系统中的体现即，部分优质物品，购买的人数较多，即与其相关的的用户行为轨迹会较多。
这样，在协同过滤推荐中，由于我们主要的依据就是我们的历史行为行为数据，所以这种物品得到推荐的机会就越多。
这样，不断循环迭代，得到推荐的物品都集中在少数的一些物品中，而大部分物品是没有被推荐的机会的。
这就造成了造成长尾现象。
而马太效应的意思是，通俗点说就是，强者愈强，弱者愈弱。而长尾的直接体现就是马太效应。
通常来讲(当然也有特殊情况)，一个推荐系统，如果长时间处于长尾之中，就会造成推荐疲劳，推荐的效果就会下降。
所以，很多时候，挖掘长尾是推荐系统不可缺少的部分。即，我们需要把尾巴部分并且是有价值的部分给适当的展示出来。
挖掘长尾的方法很多，其中一种常见的方式就是给热点物品适当的降权。比如物品，我们为热点物品进行权重下降，这样在最终推荐的结果中，非热点物品得到推荐的机会就增大，从而适当的挖掘了长尾。

5.3 AB分流测试

对于推荐系统来说，离线的评测其实并不能很准确的判断一个推荐算法的好坏。最准确的判断应该是线上实际效果观察。
而AB测试是推荐系统评测的标准做法，即我们在线上同时运行两种算法模型，让流量分别走不通的算法，最终通过收集结果数据进行比较算法的优劣。
通过AB测试必须满足以下几个条件：流量必须能够控制，这是为了保证线上效果的稳定性；必须保证流量的随机性，这样的结果才具有说服力。

X 参考资料

《浅谈矩阵分解在推荐系统中的应用》
http://blog.csdn.net/sun_168/article/details/20637833
《基于ALS算法的简易在线推荐系统》
http://blog.csdn.net/zhangyuming010/article/details/38958419
《Databricks孟祥瑞：ALS 在 Spark MLlib 中的实现》
http://www.csdn.net/article/2015-05-07/2824641
《基于Spark MLlib平台的协同过滤算法---电影推荐系统》
http://snglw.blog.51cto.com/5832405/1662153
《SVD奇异值分解》
http://blog.csdn.net/wangran51/article/details/7408414
《基于距离的计算方法》
http://blog.sina.com.cn/s/blog_52510b1d01015nrg.html
《相似度算法》
http://blog.sina.com.cn/s/blog_62b83291010127bf.html
《movielens数据集下载》
http://grouplens.org/datasets/movielens/
《四种网络模型》
http://www.cnblogs.com/forstudy/archive/2012/03/20/2407954.html
《Spark官网ALS实例页面》
http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html

PS：欢迎转载，But，不管是对你的尊重还是对我的尊重，Please保证全文的完整性，保留作者、出处以及公众号，谢谢！//比较官方的说法就是，我保留追究盗版的权利，哈哈~~