一种基于GridGIS的增量式协同过滤算法

邸佳奇[1,2] 王霓虹[2]
(哈尔滨师范大学计算机科学与信息工程学院 哈尔滨 150025)[1]
(东北林业大学信息与计算机工程学院 哈尔滨 150040)[2]

摘 要 空间数据的广泛应用需要高效的推荐系统来管理,以增加空间数据的可用性。用户协同过滤(Collaborative Filtering)是推荐系统中发展最为迅速的方法之一,也是在电子商务领域应用最广泛的方法。在研究传统协同过滤算法的基础上提出了一种减轻数据稀疏性对推荐效果产生的负面影响的方法。提出了一种基于项目相似度的数据填充方法,其目的在于当原始数据集比较稀疏时为算法提供足够的数据支持。经实验证明,改进算法在空间数据集上比传统方法有更好的预测性能和运行效率。

关键词 GIS,网格计算,GridGIS,协同过滤,增量算法
中图法分类号 TP311.13 **文献标识码** A

Incremental Collaborative Filtering Algorithm Based on GridGIS

DI Jia-qi[1,2] WANG Ni-hong[2]
(College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China)[1]
(College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, China)[2]

Abstract Wide application of spatial data requires an efficient system to manage the recommendation in order to increase the availability of spatial data. Extensive application of spatial data requires an efficient framework to manage, in order to increase the availability of spatial data. Grid geographic information system (GridGIS) supports rapid spatial data retrieval, allowing users to transparently access data at any time in any place. Traditional similarity algorithm is mathematically very rigorous, has somewhat less usefulness and lacks data support. The experiment proves that algorithm in spatial data sets than the traditional method has better prediction performance and operating efficiency.

Keywords GIS, Grid computing, GridGIS, Collaborative filtering, Incremental algorithm

随着信息技术尤其是互联网的高速发展,接入互联网的服务器数量越来越多,网页的数量也因此呈现指数级的增长态势,每时每刻都有大量的信息呈现在我们的面前,让人目不暇接。为了更有效地过滤信息,人们研发了搜索引擎。然而传统的搜索算法只根据关键词来搜索网页,对于同一个词只能呈现一模一样的结果给所有用户,无法提供个性化的服务给特定的用户,如像 Google, Baidu 等一些传统的搜索引擎。尽管这种大众化的服务在很大程度上方便了我们的生活,但在有的情况下我们更需要一种能主动为我们过滤信息的工具,而不是我们来掌握主动权。

这种工具,就是在今天大行其道的推荐系统。根据推荐算法的不同,一般把推荐系统划分为3类:基于内容的系统、协同过滤系统以及混合式系统。基于内容的推荐系统主要采用一种内容分析的技术,而有关这方面的研究在推荐系统出来之前就已经有了比较多的成果。

协同过滤的出现虽然比内容分析晚一点,但是其发展却是相当迅速的。协同过滤的思想比内容分析还要直接、简单。其依靠的是一种群体效应,简单说就是利用与目标用户意趣相投的人的观点来为目标用户产生推荐,由此可见,协同过滤的关键点在于怎么定位这些与自己有共同喜好的群体。基于协同过滤的系统克服了内容过滤不能分析非文本化物品的缺点,并且在推荐物品的种类上也更多样。而缺点在于对用户行为数据比较敏感。

由此也可以看出这两类算法是可以互补的,因此实际的推荐系统大多把不同的推荐算法结合起来,也就是混合式的推荐算法,而这些混合式的推荐算法往往比普通算法具有更好的性能,不过也因此具有更高的算法复杂度。

本文将协同过滤的经典算法与GridGIS的相关技术进行结合,提出了一种在分布式环境下实现的增量式协同过滤算法。

1 相关研究概述

1.1 GridGIS

网格技术给GIS的发展带来了机遇和挑战,因此,人们提出了网格GIS的概念。例如"GridGIS是指基于网格计算的地理信息系统,可以认为网格GIS是GIS与网格技术的有

到稿日期:2013-03-05 返修日期:2013-06-12 本文受黑龙江省教育厅科学技术研究项目:基于Android的智能教育资源信息平台构建研究(12531215)资助。

邸佳奇(1981-),男,博士生,讲师,主要研究方向为GIS、网格计算、云计算,E-mail:di_jacky@163.com;王霓虹 教授,主要研究方向为数字林业,E-mail:wnh@mail.nefu.edu.cn(通信作者)。

机结合，是 GIS 在网格环境下的一种应用”[1]。网格 GIS 是网格技术背景下 GIS 理论与技术相融合创新的阶段。分布式空间数据网格和网格计算在解决 GIS 存储效率和计算效率的矛盾时非常有效，为解决空间数据的全面共享问题提供了有利的环境与工具。看一下网格 GIS 的本质，网格 GIS 的基本构成，如图 1 所示[2]。

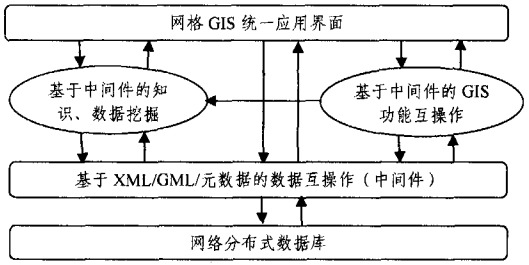

图 1　网格 GIS 的基本构成

1.2　协同过滤

GridGIS 能够有效地集成和管理空间数据，但由于高效率的集成度和大量的空间数据，使用户获得自己想要的数据成为一个急待解决的问题。协同过滤可以非常直接地解决这个问题，简单来说就是利用与目标用户兴趣比较相似的群体的喜好来为其产生推荐。

与以往广泛采用的基于内容的推荐系统相比，协同过滤具有以下优点：

(1)协同过滤可以很好地规避掉那些不容易文本结构化的对象。

(2)可以推荐新事物。

(3)协同过滤的一大特点是可以不断自我学习，随着用户数据的增加，推荐准确度也会越来越高。

目前有两类协同过滤推荐算法：一类是基于使用者的协同过滤推荐算法[3,4]，另一类是基于项目的协同过滤推荐算法[5-8]。基于使用者的协同过滤推荐算法基于这样一个假设，即如果使用者对一些项目的评分比较相似，则使用者对其它项目的评分也比较相似。算法根据目标使用者的最邻近者(最相似的若干使用者)对某一个项目的评分逼近目标使用者对该项目的评分[3,4,9]。基于项目的协同过滤推荐算法认为，使用者对不同项目的评分存在相似性，当需要估计使用者对某个项目的评分时，可以就使用者对该项目的若干相似项目的评分进行估计[5-9]。

2　基于 GridGIS 的增量式协同过滤算法

2.1　一种解决数据稀疏性的方法

通常情况下一个用户评价的项目数与项目总数是不成正比的，项目越多不代表用户的评价也会越多。实际上一个用户的评分数量是远远低于项目总数的，从用户-项目矩阵的角度去看会发现有效数据非常稀少。那么，从概率论的角度讲，任意两个用户之间的共同评分项会更加的稀少，而传统的相似度计算方法是在用户之间的共同评分项上计算相似度的。于是造成的最直接的影响是，有可能本来兴趣非常相似的两个用户因为共同评分项太少而不会得到太高的相似度，而本来兴趣不太相似的两个用户因为在其仅有的 1,2 个共同评分项中的评分类似而被系统判定为邻居，这样一来推荐质量就难以保证了。为了减轻数据稀疏性的影响，人们尝试了多种策略，其中把用户未评分的项目统一填写成该用户的平均评分被实践证明是一种行之有效的方法，通过这种方法协同过滤系统的推荐精度也得到了一定提高。

然而上述方法毕竟只是一种折中的策略，并不能从根本上解决数据极端稀疏这个问题。因为实际情况中用户对未评分项目的评分是不可能完全相同的，这样做难免会给推荐带来一些误差。为此本文提出一种改进的方法，其核心思想是首先计算项目相似度，然后找出用户之间评价过的项目的并集来组成共同评分项并利用相似项目的评分来预测这个共同评分项中未评分项目的评分，最后再在这个共同评分项上计算用户的相似度。这样一来我们就可以有效地降低数据的稀疏度。具体采用的方法是：

设用户 u_a 评过分的项目的集合是 I_{u_a}，用户 u_b 评过分的项目的集合是 I_{u_b}，U_{u_a,u_b} 是两个用户评分项目的并集，也就是说：$U_{u_a,u_b}=I_{u_a}I_{u_b}$，用户 u_a 和用户 u_b 在项目集合 U_{u_a,u_b} 中未评分的项目通过用户对相似项目的评分预测出来。相似项目的计算公式如下所示：

$$\mathrm{sim}(i_j,i_k)=\frac{\sum_{u\in U}(r_{u,ij}-\overline{r_{ij}})(r_{u,ik}-\overline{r_{ik}})}{\sqrt{\sum_{u\in U}(r_{u,ij}-\overline{r_{ij}})^2\sum_{i\in I}(r_{u,ik}-\overline{r_{ik}})^2}} \tag{1}$$

式中，U 代表同时对项目有评分的用户集合，$\overline{r_{ij}}$ 代表对 ij 的平均评分，$\overline{r_{ik}}$ 代表对项目 ik 的平均评分。然后利用以下公式(假设用户 u_a 没有对项目 i 评分，显然用户 u_b 对项目 i 有评分)：

$$p_{u_a,i}=\overline{r_{u_a}}+\frac{\sum_{j\in I}\mathrm{sim}(i,j)(r_{u_a,j}-\overline{r_{u_a}})}{\sum_{j\in I}|\mathrm{sim}(i,j)|} \tag{2}$$

$$\mathrm{sim}(u_a,u_b)=\frac{\sum_{i\in I}(r_{u_a,i}-\overline{r_{u_a}})(r_{u_b,i}-\overline{r_{u_b}})}{\sqrt{\sum_{u\in U}(ru_{a,i}-\overline{r_{u_a}})^2\sum_{i\in I}(r_{u_b,i}-\overline{r_{u_b}})^2}} \tag{3}$$

得到用户 u_a 对项目 i 的预测评分(此数据只用于辅助运算，不会保存到数据库中)。其中 I 代表项目 i 的最近邻集合，$\overline{r_{u_a}}$ 代表用户 u_a 的平均评分。最后再在项目集合 $U_{u_a'u_b}$ 上计算用户 u_a 和用户 u_b 之间的相似度。

为了计算用户之间的相似度，需要采用式(3)所描述的计算方法。在传统的协同过滤算法中，用户每次访问时，系统都需要重新计算用户之间的相似度以产生推荐，可想而知，在大型商务系统中这将会是多么庞大的一个计算量，同时也可能会给用户带来极为不好的体验(系统反应速度必然会受影响)。为此研究者也提出了多种解决方案，有的提出结合贝叶斯网络和聚类技术，有的提出利用奇异值分解(SVD)来减小用户-商品矩阵的维度，有的提出实例选择，更贪婪一些的算法如随机用户采样以及直接丢弃一些评分很少的用户或者很受欢迎/很不受欢迎的项目等等。

不幸的是，所有这些方法尽管能提高系统性能，但在某种程度上都降低了推荐的质量。这也是很好理解的，因为上述方法都是对原始数据集的一种简化，以达到减少计算量的目的，或多或少丢失了一部分信息。如果想要在计算量上有所减少，会造成推荐质量的降低，反之如果想保证推荐质量，就需要在完整数据集上进行计算，但是会造成运算量过大的问题。

2.2 增量式协同过滤算法

针对运算量过大，也就是算法可扩展性的问题，我们提出了一种增量式的协同过滤算法。算法的核心思想是：在系统初始时利用式(1)在原始用户-项目矩阵上完整地运行一次(这一步比较耗费时间，可离线进行)，以后每次数据有更新时都采用增量的方式重新计算目标用户与其他用户之间的相似度，这样可以在不损失信息的情况下非常有效地降低运算量。为了详细分析变化过程，我们把式(1)改写为：

$$A=\frac{B}{\sqrt{C}\sqrt{D}} \tag{4}$$

A 代表 $\mathrm{sim}(u_a,u_b)$，B 代表 $\sum_{i\in I}(r_{u_a,i}-\overline{r_{u_a}})(r_{u_b,i}-\overline{r_{u_b}})$，$C$ 代表 $\sum_{i\in I}(r_{u_a,i}-\overline{r_{u_a}})^2$，$D$ 代表 $\sum_{i\in I}(r_{u_b,i}-\overline{r_{u_b}})^2$。

当用户 u_a 对项目 i_a 提交了一个新的评分 r_{u_a,i_a} 时，需要更新该用户与其他用户之间的相似度 A'，即：

$$A'=\frac{B'}{\sqrt{C'}\sqrt{D'}} \tag{5}$$

式中，$B'=B+e$，$C'=C+f$，$D'=D+g$。e,f,g 分别代表增量算法中各个模块的增量。根据具体情况的不同，有两组计算公式：

若用户 u_b 对 i_a 没有评分，其计算公式如下：

$$e=(r_{u_a,i_a}-\overline{r'_{u_a}})(r_{u_b,i_a}-\overline{r'_{u_b}})+\sum_{i\in I}\mathrm{d}\overline{r_{u_a}}\,\mathrm{d}\overline{r_{u_b}} \tag{6}$$

$$f=(r_{u_a,i_a}-\overline{r'_{u_a}})^2+\sum_{i\in I}\mathrm{d}\overline{r_{u_a}}^2 \tag{7}$$

$$g=(r_{u_b,i_a}-\overline{r'_{u_b}})^2+\sum_{i\in I}\mathrm{d}\overline{r_{u_b}}^2 \tag{8}$$

否则，第一步解决数据稀疏性时系统会因为用户 u_b 对该项目有评分而预测 r_{u_a,i_a} 的值，此时就相当于更新了 r_{u_a,i_a}，其计算公式如下：

$$e=\mathrm{d}r_{u_a,i_a}(r_{u_b,i_a}-\overline{r_{u_b}}) \tag{9}$$

$$f=\mathrm{d}r_{u_a,i_a}^2+2\mathrm{d}r_{u_a,i_a}(r_{u_a,i_a}-\overline{r'_{u_a}})+\sum_{i\in I}\mathrm{d}\overline{r_{u_a}}^2 \tag{10}$$

$$g=0 \tag{11}$$

为了实现增量计算，需要保存所有用户对的 B、C、D 和各个用户的平均评分值以及它们共同评分项的项目集合，表 1 是公式中出现过的元素的详细描述。

表 1 出现在增量 e,f,g 计算公式中的元素描述

元素	描述
B,C,D	缓存数据(针对所有用户对)
$\sum_{i\in I}r_{u_a,i}$，$\sum_{i\in I}r_{u_b,i}$	缓存数据(针对所有用户对，他们共同评分项的各自算术和)
I	缓存数据(针对所有用户对，他们共同评分项的项目集合)
$\overline{r_{u_a}}$，$\overline{r_{u_b}}$	用户 u_a 和用户 u_b 共同评分项的平均评分，可通过缓存数据计算获得
$\overline{r'_{u_a}}$，$\overline{r'_{u_b}}$	用户 u_a 和用户 u_b 新的共同评分项的平均评分，分两种情况： • 若用户 u_a 对 i_a 没有评分 $\overline{r'_{u_a}}=\frac{r_{u_a,i_a}}{m+1}+\frac{m}{m+1}\overline{r_{u_a}}$ • 否则：$\overline{r'_{u_a}}=\frac{\mathrm{d}r_{u_a,i_a}}{m}+\overline{r_{u_a}}$ m 为用户 u_a 和用户 u_b 共同评分项目的数量，可通过 I 获取
r_{u_a,i_a}	用户 u_a 对项目 i_a 的评分(新提交的评价数据)
r_{u_b,i_a}	用户 u_b 对项目 i_a 的评分： • 若用户 u_b 对 i_a 没有评分，则该数据为预测所得 • 否则为查询数据库所得
$\mathrm{d}\overline{r_{u_a}}$，$\mathrm{d}\overline{r_{u_b}}$	用户新的共同评分项的平均评分和之前的差值： $\mathrm{d}\overline{r_{u_a}}=\overline{r'_{u_a}}-\overline{r_{u_a}}\Leftrightarrow\overline{r'_{u_a}}=\overline{r_{u_a}}+\mathrm{d}\overline{r_{u_a}}$
$\mathrm{d}r_{u_a,i_a}$	用户 u_a 对项目 i_a 的当前的评分和之前的差值： $\mathrm{d}r_{u_a,i_a}=r'_{u_a,i_a}-r_{u_a,i_a}\Leftrightarrow r'_{u_a,i_a}=r_{u_a,i_a}+\mathrm{d}r_{u_a,i_a}$

3 实验对比分析

为了对比改进算法与传统算法(即采用皮尔逊相关系数作为相似度计算公式的算法)的性能差异，我们设计了如下一些实验。

实验所用数据来源于 MovieLens 的数据库。MovieLens 的数据是由用户提交的对 GridGIS 平台上的空间数据信息的需求记录组成，我们选择下载了一个由十万条评分记录组成的数据集，里面包含了一千位用户的评分，评分范围是 1～5 分，并且每个用户的评分记录不少于 20 个。

首先比较两种算法的推荐质量，因此我们暂不考虑增量算法对系统在时间性能上的提升。我们采用了一种最常用的度量方法平均绝对误差 MAE(Mean Absolute Error)，该方法简单且直观，其计算公式如下所示：

$$MAE=\frac{\sum_{i=1}^{N}|p_i-r_i|}{N} \tag{12}$$

式中，p_i 代表用户的预测评分，r_i 代表用户的实际评分，N 代表参与项目数。测试算法前需要首先把原始数据集划分为训练集和测试集，MovieLens 的下载包里提供了一个脚本程序可以替我们完成这个工作。划分标准是这样的：把原始数据集划分为 80%的训练集和 20%的测试集。

实验在一台主频 3.2GHz，2G 内存的机器上进行，一共测试了 7 组，每一组采用不同的最近邻居数。其实验测试结果如表 2 所列。

表 2 两种算法的 MAE 对比

最近邻居数	传统相似度算法	改进的相似度算法
5	0.889	0.895
10	0.900	0.890
20	0.905	0.900
30	0.911	0.891
40	0.908	0.885
50	0.907	0.881
100	0.877	0.858

从表中可以观察到几个现象：

1. 除了最近邻居数为 5 的时候改进的相似度算法的 MAE 略高于传统算法以外，其他情况下改进算法的性能都要好于传统算法。

2. 不论是哪种算法，当最近邻数从 5 个逐渐增加到 20 个时，其 MAE 都呈现一个上升的趋势。这与我们的直觉有点偏差，按常理，参与运算的邻居数越多，越容易抵消偏差，也就是说理想情况下 MAE 应该是一直呈下降趋势的。在理论上这样的情况是不成立的。经过分析，发现造成这种现象的原因应该是我们并没有准确为目标用户找到其最近邻，算法可能误把本来不太相似的用户当成了目标用户的邻居，原因还是和数据稀疏性有关。这也说明改进的算法也并不能彻底解决这个问题。

3. 当邻居数多于 30 个后，MAE 便开始一直下降，这说明在其他条件不变的情况下，单纯增加参与运算的最近邻数对预测性能的提高是有好处的。然而现实系统中邻居越多，计算量和空间需求就越大，不可能维护一个数量太大的相似列表。而实验也证明，当邻居数为 5 个时，其 MAE 与邻居数为

50个时相差并不大(在传统算法中甚至性能更好),所以如果系统不是对预测性能要求特别严格,那么只需要维护一个相对小一点的相似列表就可以了。

最后是增量算法与传统算法的性能对比,我们设计的测试方法如下:

首先假定前述增量算法所需使用的数据已经全部保存到内存中,然后模拟用户提交一个新的评分,分别采用增量算法和传统算法来进行计算,发现速度提升是非常明显的。在继续采用上个实验数据集的前提下,我们通过改变参与运算的空间数据量或用户数量反复做了几次试验,测试结果如表3和表4所列。

表3 在不同空间数据量下的算法时间对比(用户数量不变)

空间数据量	235	471	943
传统算法	0.080s	0.175s	0.340s
增量算法	0.005s	0.005s	0.005s

表4 在不同用户数量下的算法时间对比(空间数据量不变)

用户数量	420	841	1682
传统算法	0.080s	0.165s	0.340s
增量算法	0.001s	0.002s	0.005s

从这两张表中可以发现:

1. 不论空间数据量和用户数量的大小如何,增量算法的用时都要比传统算法少,当然这样的结果也是可以预期的,因为增量算法的计算量相对传统方法大幅度下降。但是这个实验仅仅是一个理论性的,在大型系统中,那些缓存下来的数据量是比较巨大的,换句话说就是对内存消耗较大,这并不太现实。可采取的做法是将其写到磁盘中,而这样的代价是在I/O上会有比较大的开销,因此如果采取了这种策略,实际速度提升不会像上述那么大。

2. 另外一个比较重要的现象是对于传统算法而言,不论是空间数据量还是用户数量的大小都会对算法的运行时间造成重要影响,而对于增量算法而言,空间数据量的大小并不会给算法运行时间造成影响,算法只对用户数量的大小敏感。这是因为传统算法重新计算用户之间的相似度时采用的是式(3)的方法,该方法的计算复杂度与空间数据(项目)数量显然是一个正比关系,而增量算法所采用的方法得益于缓存数据的使用而规避了这个问题。由此可以得出以下结论:系统中项目数量越多,增量算法的优势就越明显。

增量算法由于需要缓存一些数据用于增量计算,因此在内存消耗上必然大于传统算法。为了考量增量算法较传统算法在内存使用上的多余消耗,我们模拟了几个不同规模大小的数据集,测试结果如表5所列。

表5 在不同规模大小数据集下的算法内存消耗对比

数据规模(用户/项目)	1000/2000	2000/5000	4000/10000
传统算法	15.523MB	76.614MB	305.592MB
增量算法	15.716MB	77.097MB	306.555MB

从表中可以看出,即使在数据规模较大的情况下,增量算法较传统算法也并没有过多的内存消耗,而用这点空间换取的时间,却对系统性能提升有非常明显的帮助。

结束语 本文提出了一种当用户-项目矩阵(评分矩阵)有更新需要重新产生推荐列表时减少计算量的增量式算法。该算法主要使用了一种以空间换时间的思想,通过缓存中间数据来达到简化计算的目的,可以在一定程度上提高系统的性能。经实验证明,改进算法在空间数据集上比传统方法有更好的预测性能和运行效率。如果用户没有提供评分,系统就没办法为其产生推荐。又或者一个新的项目一开始也不会有评分,系统也没办法把它推荐出去。因此下一步我们可以考虑结合其他推荐技术,比如基于用户资料或者内容相似度的推荐算法,采用一种混合式的推荐算法并根据实际情况应用到不同的场景中。

参 考 文 献

[1] 王铮,吴兵. GridGIS-基于网格计算的地理信息系统[J]. 计算机工程,2009(3):38-40

[2] 王金鑫. 网格GIS的四重地理空间网格模型[J]. 武汉大学学报:信息科学版,2011(1):73-76

[3] Y Chuan, X Jie-ping. Recommendation algorithm combining the user-based classified regression and the item-based filtering[C]// Processing of the International Conference on Electronic Commerce, Proceedings-the new E-commerce: Innovations for Conquering Current Barriers, Obstacles and Limit at ions to Conducting Success fu l Business on the Internet. 2012:574-578

[4] Breese J, Hecherman D, Kadie C. Empirical analysis of predictive algorithm s f or collaborative filtering [C]// Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI' 98). 2010:43-52

[5] Arwar B, Karypls G, Konstan J, et al. Item-based collaborative filtering recommendation algorithm s [C]// Proceedings of the 10th International World Wide Web Conference. 2009:285-295

[6] Kim B M, Li Q, Park C S, et al. A new approach f or combining content-based and collaborative filters [J]. Journal of Intelligent Information System, 2010, 27(1):79-91

[7] Karypis G. Evaluat ion of item-based top-n recommendation algorithms [C]// Proc. of the Tenth International Conference on Information and Knowledge Management. 2009:247-254

[8] Deng Ai-lin, Zhu Yang-yong, Shi Bai-le. A collaborative filtering recommendation algorithm based on item rating prediction [J]. Journal of Software, 2010, 14(9):1621-1628

[9] Arwar B, Karypis G, Konstan J, et al. Analysis of recommendation algorithms for E-commerce [C]// Processing of 2nd ACM Conference on Electronic Commerce. 2008:158-167