2019 春数据科学导论

Influence Maximization 大作业

李梓童 2017202121

○、OS & Python

Windows

Python 3.7.1

一、IM 函数的代码实现思路

最终采取算法：DegreeDiscount。

1. 读取图数据：首先读取第一行的边数，按照 u->v 的模式添加到 networkx 生成的有向图 DiGraph 中。

2. 初始化 d[u]、p[u]、dd[u]：d[u]赋值为 u 出度之和加 1，p[u]均赋值为 1.0，dd[u]赋值为 d[u]和 p[u]的乘积。

3. 进行 k-1 轮循环，根据 dd[u]选择顶点。每轮循环中，都对上一轮找出的 max_discountdegree_node 有关的点进行修改。以 max_discountdegree_node 为出发点的边，其终止点的 p[u]自乘(1-该边的概率)；以 max_discountdegree_node 为终止点的边，其出发点的 d[u]自减该边的概率。做了如上更新后，选择 dd[u]最大的顶点加入 S。

4. 当 S 个数达到 k 个，退出循环，返回 S。

二、实验细节与反思

1. 起初在读取文件构建有向图时，光是构建图就用时超过 5 分钟。最初代码中有一个 dataset，每读一条边就判断边的起点终点是否在 dataset 中，如果没有就把该顶点 append 到 dataset 里。实际上这一步并不需要，G.nodes()可以直接返回图的顶点。于是删除了该部分 append 代码，运行速度大大提高，我想太多的 append 操作拖慢了速度。

2. 上课的 ppt 里以无向图为例，边对两边都有效果，因此凡是与边有关联的顶点 u 的 d[u]、p[u]都直接自乘自减。但在用 graph 处理有向图时，要考虑是边的出度自乘而边的入度不用，边的入度自减概率而边的出度不用，要再考虑这一层。

三、其他尝试

老师给的 ppt 最后一页的参考文献链接打不开，所以主要根据下面这篇论文进行尝试分析：《Efficient Influence Maximization in Social Networks》

链接：http://snap.stanford.edu/class/cs224w-readings/chen09influence.pdf

Attempt 1：Representative Nodes Algorithm

这个算法在参考文献里没有提到。它主要思路是搜索 k 个节点的集合，首先考虑集合中每对顶点之间所有距离的累加和（累加和越小，则顶点影响范围重复概率越小），其次是集合中每对顶点之间的最小距离（最小距离越大，则顶点影响范围重复概率越小）。在 small 数据集上，跑出来的结果是 3,8,4，在小数据集上 influencespread 效果已经不好，所以没有采用。这个算法的短板其实也很明显：虽然算法简单，但没有考虑到不同边影响力不同。它适合在所有边影响概率相差不大的图里跑，这样一来忽略边的不同权重也不会对结果产生重大影响；它的提出背景也是 ppt 里一开始出现的等概率无向图。

Attempt 2：General Greedy Algorithm

这个算法的主要思路在上课已经提到过，主要是迭代计算。每一步迭代，都选取使当前状态增益最大的点。考虑到浮点数相乘的特殊性，我选择用（1-传播概率）来代替原本的传播概率，然后以（1-传播概率）为每边权重，根据迪杰斯特拉算法寻找最小路径（实际上是最大概率传播路径）。大概是我代码没有写好，在大数据集上，并不能在 5min 之内跑出结果；在 small 数据集上，跑出来的结果也依然不是最优，所以没有采用。个人认为贪心算法如果正确运行，结果应该比 DiscountDegree 会好些，但是在代码实现的过程中，可能是计算最短路径等算法不够优化，用去太多时间，运行效率不行。

四、应对大规模图数据的设想

从存储方式上看，大规模图数据可以用稀疏矩阵、分布式数据库进行存储。稀疏矩阵存储能节省的空间毕竟是有限的，分布式存储则从根本上改变了存储方式。

从计算处理上看，大规模图数据可以通过运用 MapReduce 模型、BSP 计算框架及其分解框架 GAS 模型等手段提高计算效率。不过怎么在分布式环境下计算 influence maximization 呢？它一个图是整体，图本身的数据难道分开来存吗？如果分开来存储，是不是先计算子图的较优解，然后根据子图较优联合起来求全图较优？但是这样计算出来的种子符合整张图的情况吗？个人认为 IM 问题放到分布式环境里不会太容易。一旦计算规模变小，结果的准确性也随之受影响。

五、实验小结

本次实验以开放的形式，研究 influence maximization 问题，没有固定解法和标准答案，在限定时空条件下求较优解。实验过程中，“和计算机打交道是一个 balance 的过程”这句话得到充分的体现。简单的算法、低时空消耗、结果的有效性，这几个因素不能全部都拿到满分，程序员只能在其中取舍，在限制条件下找到平衡点。