G-HASH[1]

贾新禹

2015 年 4 月 28 日

1 相关工作

1.1 子图搜索

1.2 模糊子图搜索

1.3 图相似性搜索

通常，我们有很多方法去测量图之间的相似度。第一种方法是编辑距离 *(edit distance)*. 编辑距离就是我们将图 G 通过一系列操作 (增删点边，重新标号等) 变换为另一个图 G' 所需的操作数。我们可以通过给不同操作分配不同的费用，然后用费用总和当做距离来调整这个方法的准确度。虽然编辑距离是一种非常直观的图相似性测度方法，但是实际上我们难以计算它 (是个 NP-hard 问题).*C-Tree*[2] 是一种被广泛使用的图索引模型。它没有使用图的片段信息作为特征值，而是把数据图组织在一种内部节点是图闭包 *(graph closures)*, 叶节点是数据图的树形结构中。相比于前两种方法 *GraphGrep* 和 *gIndex*,*C-Tree* 的最大优势在于其支持相似性搜索，而前两个并不支持。

还有一种名为 *GString* 的子图相似性查询方法也和 *GraphGrep* 一样是用图片段作为特征值的。当然，这种方法与前面两种基于特征值的子图查询还是不同的。在这个方法中，首先我们分解复杂图为节点数较少的连通图，得到的这些连通图每个都是一个特定的图片段。随后，我们用一种标准的编号方式把数据库中的所有图都转化为一个个字符串。并用这些字符串构建一颗后缀树来支持相似性搜索。这种方法融合了子图的数据表达能力 (信息完整) 和用字符串匹配来查询图的速度 (速度快)。

另外, 最大公共子图 *(maximal common subgraph)*[3] 和图配对 *(graph alignment)*[4, 5] 这两种方法也常被用来定义图相似度。虽然有这么多方法，但是不幸的是迄今为止我们仍没有一种简单的方法来索引或者度量大图数据库。

1.4 图核函数 Graph Kernel Functions

目前业界有很多图核函数，而开创性的一个图核函数是 Haussler 在他对 *R 卷积核 (R-convolution kernel)* 的研究中提出的。现在大多数图核函数都遵循它提出的这种框架。R 卷积核是基于把离散的结构 (如图) 分解成一系列的组成元素 (子图) 这个思想的。我们可以定义许多这样的分解, 就像组成成分中的核心一样。R 卷积核保证无论选择怎样的分解方式或者成分核心，都能得要一个对称的，半正定的正函数，或者一个成分间的核函数

1.5 Hash 函数实现快速图相似性搜索

根据将图中的每个节点利用函数映射到一个特征空间上的这一思想，我们可以设计出一种核函数来进行快速相似性搜索。具体的说，我们有了以下两条发现。

- 当图 G 中节点 u 向量和图 G' 中的节点 v 完全不同，节点 u 和 v 的核值很小基本不影响图核值。所以我们如果仅计算相似点对的核值，得到的核矩阵将和 WA 方法中的相似性度量类似，但耗时很少。
- 如果基于节点向量来构造哈希表的话，相似的对象的位置上将也很靠近。所以我们可以很快速得利用哈希表来寻找相似点对。另外，如果数据库中所有图都存入哈希表中的话，一个位置可能对应不同图中的多个相似节点。因为这些节点都是相似的，所以只要利用其中两个节点进行一次 RBF 核运算就可以代表所有的结果了。节点覆盖给了我们另一个省时间的契机。

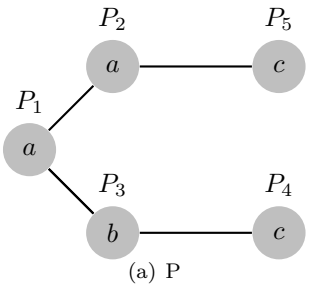
基于以上两条发现，我们引入我们的方法:k-NNs 查询 (G-hash).G-Hash 是一种利用 WA 方法进行精确相似度度量，并利用哈希来降低时间复杂度。具体算法见下文。

1.6 索引构建

像 WA 方法一样，我们首先需要利用小波变换将数据库中的所有图拆解成点向量的形式。因为 WA 方法对距离参数的选取并不敏感，用不同的小波算法造成的差异也并不大 [6] ，所以我们设定 h 为 2，并才用 *Haar* 小波算法来简化算法

当图完全拆解成节点向量后，我们就可以利用这些向量构建哈希表。这时，我们需要一个哈希函数来把相似的节点哈希到相同的位置。这就意味着我们要基于节点向量构建哈希键。我们可以用和构造节点向量同样的思想构建哈希键，即利用节点标号和邻接信息来构造。我们将每一个节点向量中的特征信息离散成一个整数，并将每个节点标签直接编码为一个 n 位的字符串（除去一个代表本身标号的 1，其他全 0)。其他特征就近似到最近的整数上。当节点向量变成了一个整数列表后，我们再把节点向量变为一个字符串 (用二进制表示每个整数，并用下划线连接)。这样的一个字符串就是对于节点的哈希键。我们以图表 1 的图 P 来举例说明。

例子 4.1. 我们选取节点标签和邻接节点各个标号的个数作为度量特征。所以我们一共有四个特征。这个图的特征仅属于取值中的一种:*nominal*。我们首先写出节点特征矩阵如图表 *1* 中的 *(B)*。然后用小波函数提取出实际特征值。举例而言，对于 P_3 在用 $h = 0$ 的小波分析提取后，局部特征是 $[b, 2, 0, 1]$。然后生成的哈希表如图表 *1(C)* 所示。



(a) P

Nodes	Label	#a	#b	#c
P_1	a	1	1	0
P_2	a	1	1	1
P_3	b	2	0	1
P_4	c	0	1	0
P_5	c	1	0	0

(b) 节点特征矩阵

哈希键	哈希值
a,1,1,0	P_1
a,1,1,1	P_2
b,2,0,1	P_3
c,0,1,0	P_4
a,1,1,1	P_5

(c) 图 P 的哈希表

图 1: 一幅简单图

注意: 在这个算法框架中，我们在构建哈希索引时没有对查询图做任何假设。

1.7 *K-NNs* 查询过程

为了获得对于给定图的 *K-NNs*, 我们需要计算它和数据库中其他图的距离。以下是我们定义的利用核函数进行两图距离度量的函数

$$\begin{aligned} d(G, G') &= \sqrt{\|\phi(G) - \phi(G')\|_2^2} \\ &= \sqrt{\langle \phi(G) - \phi(G'), \phi(G) - \phi(G') \rangle} \\ &= \sqrt{\langle \phi(G), \phi(G) \rangle + \langle \phi(G'), \phi(G') \rangle - 2\langle \phi(G), \phi(G') \rangle} \\ &= \sqrt{k_m(G, G) + k_m(G', G') - 2k_m(G, G')} \end{aligned} \tag{1}$$

公式中 $k_m(G, G)$ 代表图 G 和其本身的核函数值, $k_m(G', G')$ 是图 G' 及其本身的值, $k_m(G, G')$ 就是图 G 和 G' 的。在后文中，我们会介绍该如何计算这些值。

尽管在将查询图节点哈希到哈希表时，我们可以得到核函数

$$k_m(G, G') = \sum_{v \in G', u \in simi(v)} K(\Gamma^h(u), \Gamma^h(v)) \tag{2}$$

$simi(v)$ 是一个包含着图 G 和节点 v 哈希到同一个位置的节点集合。我们用以下的解码方式来获取包含这些节点的图号和节点号。

显然，仅利用相似点对而非所有点对来计算两图相似度可以节约很多运算时间。因此为了增加准确度，相似的节点应该被哈希到相邻的位置。在图很大 (如大于 40) 时，我们也要计算相邻位置的节点。

因此在核函数计算时我们只考虑相似点对，在使用 RBF 核的情况下，$K(\Gamma^h(u), \Gamma^h(v)) \approx 1$，所以公式 (2) 可以写成

$$K(G, G') \approx \sum_{v \in G', u \in simi(v)} 1 = \sum_{v \in G'} |simi(v)| \tag{3}$$

$|simi(v)|$ 是在 $simi(v)$ 中的节点数目。这意味着我们只需要统计图 G 中和查询 G' 相似的点个数，其和就是我们要求的核。同理，我们可以这样计算每个图与其自己的核。

在完成上述操作后，我们会得到一个距离向量，其每一个值都对应这一个数据库中的图与查询图的距离，通过对这个距离向量排序，我们就可以得到对于给定的查询的 $K - NNs$。

1.8 动态插入与删除

如果要向数据库中插入一副图，我们只需将新图的节点进行哈希，然后加入节点哈希表。这样插入完成后，只有图中所有的节点的位置会发生变化。而且，因为一幅图的节点是有限的，所以插入操作的时间和构建索引的内存也是有限的。删除操作和插入类似，我们只需找到图中节点在哈希表中的对应哈希值，然后删掉即可。

参考文献

[1] Xiaohong Wang, Aaron Smalter, Jun Huan, and Gerald H. Lushington. G-hash: Towards fast kernel-based similarity search in large graph databases. *EDBT 2009, March 24–26*, pages 472–480, 2009.

[2] H. He and A. K. Singh. Closure-tree: an index structure for graph queries. *Proc. International Conference on Data Engineering'06 (ICDE)*, 2006.

[3] T. Jiang Y. Cao and T. Girke. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics, 24(13)*, 2008.

[4] H. Fro¨hlich, J. K. Wegner, F. Sieker, and A. Zell. Optimal assignment kernels for attributed molecular graphs. *Proceedings of the 22nd international conference on Machine learning*, 2005.

[5] J.-P. Vert. The optimal assignment kernel is not positive definite. *Technical Report HAL-00218278, French Center for Computational Biology*, 2008.

[6] A. Smalter, J. Huan, and G. Lushington. Graph wavelet alignment kernels for drug virtual screening. *Proceedings of the 7th Annual International Conference on Computational Systems Bioinformatics*, 2008.