

基于近似个体协同的进化子结构发现

常新功¹, 李 宏²

(1. 山西财经大学 信息管理学院, 山西 太原 030031;

2. 山西省电子产品检验所, 山西 太原 030024)

摘 要: SUBDUE 是一个主流的图数据挖掘算法。为克服其贪婪式查找易陷入局部极值的问题, 将进化算法与爬山算法相结合并引入图数据挖掘, 较好地权衡了算法的探查和利用能力。另外, 针对图数据挖掘中普遍存在的实例易丢失的问题, 采用了个体协同的查找方法, 该方法与常见的种群间协同进化算法不同, 可以使同一种群中的个体进行协同查找, 重新找回丢失的实例。同时, 还给出了一种具有多项式时间复杂度的近似图匹配算法以改善个体间协同的性能。实验结果表明, 以上措施增强了算法的执行效率及寻优能力, 能够获得更优的解。

关键词: 进化算法; 协同; 图数据挖掘; 子结构发现; 近似图匹配

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2010)09-0106-05

Evolutionary Algorithm for Substructure Discovery Based on Approximate Individual Cooperation

CHANG Xin-gong¹, LI Hong²

(1. Faculty of Information & Management, Shanxi University of Finance

& Economics, Taiyuan 030031, China;

2. Shanxi Province Electronic Products Inspection Institute, Taiyuan 030024, China)

Abstract: SUBDUE is a representative graph-based data mining algorithm. To overcome the limit that the greedy search adopted by SUBDUE may often give sub-optimal solutions, a hybrid evolutionary algorithm, which balances the exploration and exploitation of search by combining the hill-climbing and EA, is developed to perform data mining on graphical databases. In addition, during the searching process, losing instances is common and vital to the algorithm performance. To address this issue, adopt the individual cooperation strategy which is greatly different from the common cooperatively evolutionary approach based on population cooperation. The new strategy enables individuals in the same population to search in a cooperative way and gets back the lost instances. At the same time, an approximate graph matching algorithm with polynomial time complexity is also proposed to improve the performance of the process of individual cooperation. Experimental results show that these measures successfully improve the efficiency and the searching capability of the algorithm and can get better results.

Key words: evolutionary algorithms; cooperation; graph-based data mining; substructure discovery; approximate graph matching

0 引 言

作为通用的数据结构, 图以结点表示问题领域中的对象, 以边表示对象之间的关系, 可以对非常宽泛领域中的复杂事物和复杂行为进行建模。近年来, 从图数据中发现新颖的、潜在有用的和最终可理解的模式的过程, 即图数据挖掘, 得到了广泛的关注和研究, 成

为机器学习和数据挖掘领域的又一热点分支。图数据挖掘算法在生物工程、化学信息学、药学、WWW、交通运输、国家安全、社会网络分析等领域有着广泛的应用。

这些算法一般可分为两类: 第一类算法以基于图论的 AGM^[1]、FSG^[2]、gSpan^[3] 和 FFSM^[4] 为代表, 目的是找出满足最小支持度约束的所有子结构, 其出发点是频繁模式发现。第二类算法则更关注模式的新颖性和潜在有用性, 旨在找出更典型、更有意义的子结构。基于最小描述长度原理^[5] (Minimum Description Length, MDL) 和柱状搜索 (Beam Search, BS) 的 SUBDUE^[6] 和基于数据压缩和信息收益的 GBI^[7] 是这类算

收稿日期: 2010-01-19; 修回日期: 2010-04-10

基金项目: 山西省自然科学基金项目(2010011022-1); 山西省高校科技研究开发项目(20081023)

作者简介: 常新功(1968-), 男, 山西太原人, CCF 会员, 教授, 博士, 研究方向为进化计算、数据挖掘。

法中的典型代表。第一类算法返回的结果全面,但重点不突出,不便于实际应用。第二类算法返回结果数量相对较少,但却更典型,有实际意义。

文中的算法属于第二类算法。鉴于柱状查找也是一种贪婪式查找,因而 SUBDUE 常常易陷入局部极值问题,文中在前期工作^[8,9]中在 SUBDUE 的基础上将进化算法和爬山算法相融合引入到子结构发现问题之中,在一定程度上提高了解的质量。进一步的研究发现在图数据挖掘中广泛存在着一种称之为“实例丢失”的现象,它会导致算法对一个子结构的不正确评价,从而造成解质量的下降。为此文中提出了一种新的遗传算子—个体协同算子,使得种群中彼此不同但却代表同一子结构的个体能够以协同的方式进行查找,互通有无,这样可以提高查全率并最终提高解的质量。另外,由于个体协同算子广泛使用时间复杂度较高的图同构,同时也考虑到进化算法本身的随机性,故文中还提出了一种近似图匹配算法,以提高协同算子的运行效率。

1 基本概念

文中研究的图为带标签的图(labeled graph),它可用一个四元组 $G = (V, E, L, \varphi)$ 来表示,其中, V, E 分别为结点集和边集,分别代表图 G 所表示的系统中的对象及对象间的关系。 L 是标签集,结点和边均可以带标签(label),不同的标签代表了不同的对象类型或对象间关系类型。 $\varphi: V \cup E \rightarrow L$ 称为标签函数,它赋予每个结点或边以不同的标签。图 1 给出了一个带标签的图的示例。其中,结点用椭圆表示,边用线段来表示,图中的数字 1~12 表示各结点的编号,字符 u, v 表示各结点的标签, $a \sim f$ 表示各边的标签。

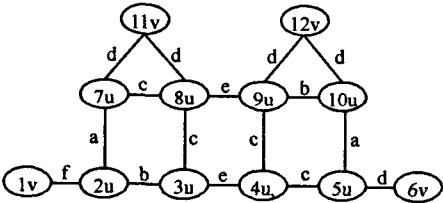


图 1 一个带标签的输入图

1.1 子结构和实例

子结构(substructure)是图数据中所有彼此同构的连通子图的公共结构。实例(instances)是与某一子结构同构的连通子图。一个子结构可以有多个实例,而一个实例只能属于一个子结构。例如表 1 中 $g(2,7), g(5,10)$ 为 S_1 的实例, $g(2,3,7,8,11), g(4,5,9,10,12)$ 为 S_4 的实例。以上 $g(\text{结点编号}, \dots)$ 表示由相应结点的编号构成的子图。子结构可理解为其实例的公共

模板,而每个实例是输入图中与其子结构同构的一个子图片段。为了便于叙述,文中以下还将以 $S(\text{边标签} \dots)$ 的方式表示一个子结构。例如,子结构 $S(a), S(ab), S(ac), S(acdd)$ 分别就代表表 1 中的 S_1, S_2, S_3, S_4 。

表 1 图 1 中若干子结构及其实例集

No.	结构(图定义)	实例集	No.	结构(图定义)	实例集
S_1		$g(2,7), g(5,10)$	S_3		$g(2,7,8), g(4,5,10)$
S_2		$g(2,3,7), g(5,9,10)$	S_4		$g(2,3,7,8,11), g(4,5,9,10,12)$

1.2 最小描述长度与子结构的评价

文中、SUBDUE 和 EPSD^[10]均用最小描述长度原理来评价一个子结构的优劣。最小描述长度原理用数据中的规则性来压缩数据,压缩程度越大表明规则性越强,反之亦然。由于子结构是图数据中众多实例的公共结构,它反映了一种规则性,可用其实例来压缩图数据。压缩过程为:设 S 是图 G 的一个子结构,将 G 中 S 的每个实例分别换为一个新结点;该结点的标签为子结构 S 的编号;原来和实例中任一结点相连的边现在和该新结点相连。压缩后所得的图称为 $G|S$ 。 G 在用 S 压缩后,描述长度为 $DL(S) + DL(G|S)$ 。和 SUBDUE 与 EPSD 一样,文中定义 $DL(S)$ 为 S 的结点数和边数之和, $DL(G|S)$ 为将 G 中 S 的实例换为新结点后所得图的结点数和边数之和。这样,子结构 S 对 G 的压缩率可定义为 $(DL(S) + DL(G|S))/DL(G)$ 。例如设 G 为图 1 所示的输入图, S_2 为表 1 中子结构,易见 $DL(S_2) = 3 + 2 = 5, DL(G|S_2) = 8 + 12 = 20, DL(G) = 12 + 16 = 28$,子结构 S_2 的压缩率 $= (DL(S_2) + DL(G|S_2))/DL(G) = (5 + 20)/28 = 0.893$ 。根据最小描述长度原理,一个子结构的压缩率越小,表示其对输入图的压缩程度越大,同时也说明该子结构具有较强的规则性。文中就是要利用进化算法寻找输入图中压缩率最小的子结构。

容易看到,同一子结构的任意两个实例不能相交,即这两个实例之间不能有公共的结点和边,否则在用 一个子结构的实例集压缩输入图时,就会发生如下现象:当一个实例换为一个新结点后,与该实例有公共结点或边的其它实例即变得不再完整,无法进一步压缩。

正是这一事实导致了“实例丢失”,从而增加了寻找最优解的难度。将在 2.4 节中再次谈到这个问题。

1.3 子结构的扩展

子结构扩展是由旧子结构加一边以生成新子结构的过程。其具体步骤为:设 S 为一个子结构,对 S 的每个实例以所有可能的方式加一边以得到新的子图;按同构关系对这些子图进行归类,属于同一类的子图抽象出一个新子结构,而这些子图即为该子结构的实例;返回这些新子结构。例如对 S_1 进行扩展, S_1 有二个实例: $g(2,7)$ 和 $g(5,10)$ 。 $g(2,7)$ 加一边可得 $g(1,2,7)$ 、 $g(2,3,7)$ 、 $g(2,7,8)$ 和 $g(2,7,11)$ 。 $g(5,10)$ 加一边可得 $g(4,5,10)$ 、 $g(5,6,10)$ 、 $g(5,9,10)$ 和 $g(5,10,12)$ 。按同构关系对新得的子图进行归类,可得 $\{g(1,2,7)\}$ 、 $\{g(2,3,7), g(5,9,10)\}$ 、 $\{g(2,7,8), g(4,5,10)\}$ 、 $\{g(2,7,11), g(5,6,10), g(5,10,12)\}$, 其对应的新子结构 $S(\text{af}), S(\text{ab}), S(\text{ac}), S(\text{ad})$ 作为结果返回。

2 近似个体协同进化子结构发现算法

2.1 个体的表示

种群中个体代表候选解,在本算法中一个个体就是一个子结构。文中用 C 语言中的结构体来表示一个子结构:

```
typedef struct
{
    Graph * graphDef; /* 图定义 */
    unsigned int numIns; /* 实例数 */
    InstanceList * ins; /* 实例集 */
    double rate; /* 压缩率 */
} Individual;
```

这种表示在进化算法中称为直接编码,即不做任何变换,用最自然的方式来表示一个个体。其优点是直观、自然,缺点是不能使用二进制、排列或其它编码情形下已成熟的交叉、变异等遗传算子,必须专门设计相应的遗传算子。下文中的混合变异和个体协同算子就是针对子结构发现问题专门设计的。

2.2 种群初始化、适应值函数和选择

首先扫描一遍图数据集,生成所有的单边子结构,然后从中随机选择若干子结构构成初始种群。一个子结构 S 的适应值 $S.\text{rate} = (\text{DL}(S) + \text{DL}(G|S)) \cdot \text{DL}(G)$ 。文中采用基于精英保留的联赛选择,联赛规模为 2。这种选择方式可以较好地保持种群的多样性。

2.3 混合变异算子

本算子将个体扩展和爬山算法融为一体,因而称为混合变异算子。在对子结构 S 进行变异时,首先对其做子结构扩展;然后在扩展所得的结果中随机选择

δ (一个正整数,可由用户指定)个子结构,其中适应值最小的子结构作为结果返回。例如对 S_1 进行混合变异,扩展 S_1 可得 $S(\text{ab}), S(\text{ac}), S(\text{ad}), S(\text{af})$ 四个子结构。设 $\delta = 2$, 随机选择的两个子结构为 $S(\text{ab}), S(\text{af})$, 则其中适应值较小的 $S(\text{ab})$ 即 S_2 作为变异结果返回。

以上过程融入了爬山算法的机制。该算法首先将用户指定的初始可行解作为当前解,然后持续地在当前解的邻域中寻找最优解,并以该最优解作为下一代的当前解,如此不断反复直到解不能更新,该解即为最终结果。作为一种贪婪式的搜索策略,爬山算法长于局部搜索但却易陷入局部极值,而进化算法则长于全局搜索但却拙于细粒度的局部查找。理论和实践证明,将二者结合可大大提高算法的性能和解的质量。上文中的 δ 定义了爬山算法中搜索邻域的大小。同时它也起到了对变异操作的随机性和贪婪性进行权衡调节的作用,随机性强时算法探查的范围广,贪婪性强时算法则更重视利用局部信息,探查能力相对较弱。易见,当 $\delta = 1$ 时,混合变异变成了随机选择; δ 越大混合变异的随机性越弱而贪婪性则越强,当 $\delta = S$ 扩展后的新子结构个数时,变异就成了贪婪性最强的爬山式查找。恰当地权衡探查 (Exploration) 和利用 (Exploitation) 正是进化设计的关键所在。

2.4 个体协同算子

在 1.2 节中谈到同一子结构的两个不同实例之间不能有公共结点,否则会造成实例丢失。以下来看一个例子:在图 1 中, $g(3,8)$ 、 $g(4,5)$ 、 $g(4,9)$ 和 $g(7,8)$ 都与子结构 $S(c)$ 同构,但由于 $g(3,8)$ 和 $g(7,8)$ 、 $g(4,5)$ 和 $g(4,9)$ 相交, $S(c)$ 的实例只能从以上 4 个子图中做无交选择,假设选择了 $g(7,8)$ 和 $g(4,9)$ 。以下扩展 $S(c)$ 可生成多个子结构。看 $S(\text{ce})$ 是如何生成的:扩展 $g(7,8)$ 可得 $g(7,8,9)$, 扩展 $g(4,9)$ 可得 $g(3,4,9)$ 和 $g(4,8,9)$, 它们都是 $S(\text{ce})$ 的实例但彼此相交。只能从中选一个,设为 $g(7,8,9)$ 作为新子结构 $S(\text{ce})$ 的实例。仔细观察图 1, 会发现 $g(3,4,5)$ 和 $S(\text{ce})$ 同构且与 $g(7,8,9)$ 无交,显然它也是 $S(\text{ce})$ 的实例,但它在扩展过程丢失了。实例丢失会导致一个子结构对图数据的压缩程度降低,从而不能正确地评价该子结构。

针对实例丢失问题, SUBDUE 和 EPSD 并未给出相应的解决方案。文中为此设计了个体协同算子,其伪码描述见图 2。该算子以整个种群 Pop 为操作对象,在新一代种群生成之后,扫描整个种群并以个体协同概率 pic 进行个体协同。协同过程为:如两个个体同构,则彼此交换与对方实例集中实例不相交的实例。这样可找回进化过程中丢失的实例,降低实例丢失带来的负面影响。

```

Cooperation(Pop, pic)
{
    for(i = 1; i < popsize; i++)
        随机生成一个随机数  $r \in [0, 1]$ 
        if( $r < pic$ )
            for(j = i + 1; j <= popsize; j++)
                if(Pop[i]和Pop[j]同构) /* Pop[i]为第i个个体 */
                    将Pop[i]中与Pop[j]中实例均不相交的实例插入Pop[j]
                    将Pop[j]中与Pop[i]中实例均不相交的实例插入Pop[i]
}

```

图2 个体协同算子伪码描述

上述算法中参数 pic 起到了调节个体协同算子的执行频率的作用,这样可在解质量和算法效率之间进行权衡。毕竟子图同构是比较花费时间的。以下来看一个个体协同的例子:在上例中如 $S(c)$ 的实例取为 $g(3, 8)$ 和 $g(4, 5)$,则在生成 $S(ce)$ 时,扩展 $g(3, 8)$ 和 $g(4, 5)$ 可得 $g(3, 4, 5)$, $g(3, 8, 9)$, $g(3, 4, 8)$ 。由于彼此相交,只能从中选一个,假设选择了 $g(3, 4, 5)$,这样就得到了一个带有一个实例 $g(3, 4, 5)$ 的子结构 $S(ce)$ 。用协同算子将这个个体 $S(ce)$ 和上文中带有实例 $g(7, 8, 9)$ 的个体 $S(ce)$ 进行协同可以得到带全实例 $g(3, 4, 5)$ 和 $g(7, 8, 9)$ 的子结构 $S(ce)$ 。

事实上,实例丢失是图数据挖掘中非常常见的现象,能否恰当地处理该问题直接关系到图数据挖掘算法的性能和效率。另外一个子结构是一个连通子图,它可以通过将其任一边设为起始边,以任意可能的加边的顺序通过子结构扩展以多种方式生成。而不同的生成顺序会导致种群中拥有这样的个体,它们彼此同构但其丢失的实例却不同,这些子结构之间的协同便是可能的,同时也是必须的。

2.5 近似图匹配算法

一个不容忽视的问题是图2协同算子中频繁地用了图同构,这会导致协同算子的时间复杂度过高。为了提高该算子的运行效率,文中提出了基于公共实例的近似图匹配算法。其伪码描述见图3。

```

CommonInsMatching( $S_1, S_2, thres$ )
{
    if( $S_1$  和  $S_2$  的结点数或边数对应不相等) return false
    if( $S_1$  的结点数 <  $thres$ ) return SubdueMatching( $S_1, S_2$ )
    else if( $S_1$  与  $S_2$  有公共实例) return true
    else return false
}

```

图3 基于公共实例的近似图匹配算法

其中 S_1, S_2 为两个个体, $thres$ 为用户指定的阈值。在图的规模小于 $thres$ 时,用 SUBDUE 提供的图同构算法;当图的规模较大时,则用基于公共实例的近似图同构算法。该算法的底层思想是:只有当两个个体的实例集有公共实例时才认为它们是同构的。这种近似的判断方法是有效的,因为既然两个个体是同构的,那么它们带有公共实例的概率就是很大的。这种方法也是合理的,因为进化算法本身是随机算法,而协同算子的报告也是受 pic 控制随机进行的。

以下定理说明了基于公共实例的近似图匹配算法具有较高的效率,同时在其证明过程中也给出了寻找两个子结构公共实例的过程。

定理1 CommonInsMatching 算法具有多项式时间复杂度。

证明:设 N 为输入图 G 的结点数, S_1 和 S_2 为两个个体,实例数分别为 n_1 和 n_2 。

CommonInsMatching 算法的第一条语句只需 $O(1)$ 时间,因为个体的结点数和边数信息可从其图定义中直接取得。第二条语句也只需多项式时间,因为当图的规模小于某一阈值时, SUBDUE 图匹配算法是多项式时间级别的^[6]。接下来证明在两个个体的结点数和边数相同时,判断二者是否有公共实例也只需多项式时间,这样便证明了整个定理。

设此时两个体的结点数为 α , 边数为 β 。在图定义中,结点和边是按其编号顺序存放的,因此当判断两实例是否同构时,只需看这两个实例的边的编号序列是否对应相等,而这只需 $O(\beta)$ 时间。因此在最差情况下,要找出 S_1 和 S_2 任二实例之间是否有公共实例,只需 $O(n_1 * n_2 * \beta)$ 时间复杂度,因为只需要将 n_1 和 n_2 个实例两两对应比较即可。另外,由于同一个体的实例彼此无交,所以 $n_1 \leq N/\alpha, n_2 \leq N/\alpha$, 因而 $n_1 * n_2 * \beta \leq \beta N^2 / \alpha^2$, 因此决定 S_1 和 S_2 是否同构只需 $O(\beta N^2 / \alpha^2)$ 时间。说明该算法具有多项式时间复杂度。

这种改进是巨大的,因为如果对结点数为 α 的两个子图用蛮力法来做图同构,其时间复杂度是 $\alpha!$ 。以第4节图 W3 用 HEASDCI 发现的最优子结构为例,此时 $\alpha = 9, \beta = 36, N = 19$, 如用蛮力法则需要 $\alpha! = 9! = 362880$ 的工作量,如用文中提出的近似匹配算法,只需用 $\beta N^2 / \alpha^2 = 36 * 19^2 / 9^2 = 160.4$ 的工作量。

2.6 算法的伪码表示

图4为基于近似个体协同的进化子结构发现算法 HEASDCI 的伪码描述。算法反复地对种群 $P(t)$ 进行选择、混合变异,并对生成的种群 $P(t+1)$ 进行近似个体协同,当达到最大进化代数限制时,返回目前找到的最优解。

```
HEASDCI(G, popsize, limit, pm,  $\delta$ , pic)
/* G:输入图, popsize:种群规模, limit:最大进化代数
*/
/* pm:变异概率,  $\delta$ :邻域大小, pic:个体协同概率 */
{
    t = 0
    种群初始化生成 P(0)
    while ( t < limit ) {
        计算 P(t) 中每个个体的适应值
        按概率 pm 对 P(t) 中个体进行混合变异并生成 P(t +
1)
        Cooperation(P(t + 1), pic)
        t ++
    }
    返回最优个体
}
```

图 4 基于近似个体协同的进化子结构发现算法伪码描述

3 实验结果及实验分析

文中从 <http://ailab.eecs.wsu.edu/SUBDUE/> 的数据集中选择了 10 个图作为本算法进行测试的实验数据集, 其中每个图都代表一个网站, 图中的每一个结点表示网站中的一个网页, 每一条边表示网站中的一个超链接。每个图的基本情况显示在表 2 中。

表 2 实验数据描述

图	结点数	边数	图	结点数	边数
W1	8	30	W6	85	303
W2	31	43	W7	27	492
W3	19	159	W8	25	532
W4	94	106	W9	86	501
W5	22	314	W10	732	16725

实验结果显示在表 3 中, 其中 EPSD^[10] 是基于进化规划 (Evolutionary Programming, EP) 的子结构发现算法, 它只使用了变异算子, 且该算子只做子结构扩展并从中随机选择一个新子结构作为结果返回, 它没有和爬山算法结合, 也没有使用协同算子。文献 [10] 表明 EPSD 在表 2 数据上的实验结果全面超过了 SUBDUE。HEASD^[9] 为没有使用协同算子的进化子结构发现算法, HEASDCI 为用了近似协同算子的进化子结构发现算法。实验参数设置为: popsize = 30, limit = 150 (在输入图 W10 时 limit = 400), pm = 1, δ = 3, pic = 0.3, thres = 8。实验结果显示在表 3 中。由表 3 易见, HEASD 优于 EPSD, HEASDCI 优于 HEASD。可见近似协同算子可以找回丢失的实例, 对解质量的提高有巨大的促进作用。

表 3 EPSD、HEASD 和 HEASDCI 发现的结果

图	算法	发现的子结构			压缩率
		结点数	边数	实例个数	
W1	EPSD	4	5	2	0.815789
	HEASD	4	5	2	0.815789
	HEASDCI	4	5	2	0.815789
W2	EPSD	6	5	3	0.743243
	HEASD	6	5	3	0.743243
	HEASDCI	8	7	3	0.635134
W3	EPSD	3	3	5	0.893253
	HEASD	9	36	2	0.758427
	HEASDCI	9	36	2	0.758427
W4	EPSD	8	7	3	0.865000
	HEASD	20	19	2	0.815000
	HEASDCI	8	7	4	0.795001
W5	EPSD	3	5	5	0.919643
	HEASD	10	41	2	0.854167
	HEASDCI	8	47	3	0.793834
W6	EPSD	3	2	19	0.817010
	HEASD	3	2	19	0.817010
	HEASDCI	5	4	10	0.817010
W7	EPSD	3	9	6	0.895954
	HEASD	6	30	3	0.867052
	HEASDCI	8	49	3	0.786126
W8	EPSD	4	16	5	0.865350
	HEASD	5	25	4	0.845602
	HEASDCI	8	54	3	0.782767
W9	EPSD	4	6	16	0.771721
	HEASD	8	13	8	0.763201
	HEASDCI	4	6	18	0.741056
W10	EPSD	2	1	345	0.960646
	HEASD	2	1	346	0.960532
	HEASDCI	2	1	346	0.960532

由于个体是以加一边的方式进行扩展的, 而变异是以个体扩展为基础的, 因此每次变异都可能会有多种扩展方式, 如果像 EPSD 那样以纯随机的方式进行变异, 会使算法在较长的时间找不到较优的解。引入爬山算法后, 就可以对更有希望的区域进行集中搜索。这是 HEASD 较 EPSD 优秀的根本原因所在。而个体协同算子的引入则使个体以协作的方式进行查找, 在一定程度上克服了图数据挖掘过程中普遍存在的实例丢失现象, 显著地提高了解的质量。这是 HEASDCI 较 HEASD 优秀的主要原因。

4 结束语

针对图数据挖掘问题中广泛存在的“实例丢失”现象, 文中设计了一种个体间进行协同查找的遗传算子, 使得种群中的个体以协同合作的方式进行查找, 以找回丢失的实例, 从而最终提高解的质量。传统的协同进化是基于种群的协同进化, 即基于“分治法”的观点, 使用多个种群, 每个种群针对不同的子任务, 经过若干代进化后, 由各种群的最优解合成整个问题的解。文中提出的协同进化则是个体间的协同, 多个个体共同进化, 互通有无, 以保证解的质量。另外, 文中还提出

好,此次分类神经网络系统将学习率设置为 0.1,训练次数运行 74 次,系统运行误差的动态变化在 Error's dynamics 中给出,图中 74 个节点分别对应了 74 次训练次数,由图 4 窗口的曲线的动态变化可知随着迭代次数的增加系统的误差越来越小。神经网络系统的权值和阈值在运行界面中也相应地给出,系统根据权值和阈值的值来求出 Data 界面中对应的用于分类的四条直线。

5 结束语

分类作为数据挖掘的主要内容之一,主要是通过分析训练数据样本,产生关于类别的精确描述。这种类别通常由分类规则组成,可以用来对未来的对象进行分类预测,有着广泛的应用前景。文中通过构造一种单层感知器的神经网络的分类方法,并进行设计分析和实验仿真,对相关技术的研究有一定的借鉴意义。

参考文献:

- [1] Pandya A S, Macy R B. Pattern Recognition with Neural Networks in C++ [M]. [s. l.]: CRC Press, 1993: 119 - 125.
- [2] 钱卫宁,魏 黎,王 焱,等. 一个面向大规模数据库的数据挖掘系统[J]. 软件学报, 2002, 13: 1540 - 1545.
- [3] Berson A, Smith S, Thearling K. Building Data Mining Application for CRM [M]. [s. l.]: McGraw - Hill, 2001: 180 - 230.
- [4] 王红军,陈庆新,陈 新,等. 基于效用分析的客户聚类方法研究[J]. 计算机集成制造系统, 2003, 9(3): 237 - 241.
- [5] Lee W, Stolfo S J. Data mining approaches for intrusion detection[C]//Proc. of the 7th USENIX Security Symposium. Berkeley, USA: USENIX Assoc, 1998: 79 - 90.
- [6] 宋世杰,胡华平,胡笑蕾,等. 数据挖掘技术在网络型异常入侵检测系统中的应用[J]. 计算机应用, 2003, 23(12): 20 - 23.
- [7] Wang J C, Huang Y, Wu G S, et al. Web Mining: Knowledge Discovery on the Web Systems, Man, and Cybernetics [C]// IEEE SMC '99 Conference Proceedings. Tokyo: IEEE Computer Society, 1999: 116 - 121.
- [8] Schroedl S, Wagstaff K, Rogers S, et al. Mining GPS traces for map refinement[J]. Data Mining and Knowledge Discovery, 2004(9): 59 - 87.
- [9] 魏宏业,吕永波,刘志硕. 基于数据挖掘的智能交通系统的决策方法研究[J]. 交通运输系统工程与信息, 2003, 3(1): 23 - 27.
- [10] Pawlak Z. Rough sets[J]. International Journal of Information and Computer Science, 1982, 11: 256 - 341.
- [11] 袁曾任. 人工神经网络及其应用[M]. 北京:清华大学出版社, 1999: 66 - 117.
- [12] Watson K, Nagel C. C# 入门经典[M]. 北京:清华大学出版社, 2006: 331 - 379.

(上接第 110 页)

了一种子结构发现情形下的近似图匹配算法,该算法具有多项式时间复杂度。实验表明混合变异和个体协同提高了算法的性能,可以获得更优的解;图的近似匹配则在不降低解质量的同时提高了算法的运行效率。不断改进算法的性能以及将文中算法应用于图分类、图聚类等图数据挖掘任务是下一步研究的方向。

参考文献:

- [1] Inokuchi A, Washio T, Motoda H. An Apriori - based Algorithm for Mining Frequent Substructures from Graph Data [C]//Proc. of the 4th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD). Lyon, France: Springer, 2000: 13 - 23.
- [2] Kuramochi M, Karypis G. Finding Frequent Patterns in A Large Sparse Graph [C]//Proc. of the 2004 SIAM Data Mining Conf.. Lake Buena Vista, Florida, USA: Morgan Kaufmann, 2004.
- [3] Yan Xi - feng, Han Jia - wei. gSpan: Graph - based Substructure Pattern Mining [C]//Int. Conf. on Data Mining. Maebashi City, Japan: IEEE Computer Society, 2002: 721 - 724.
- [4] Huan J, Wang W, Prins J. Efficient Mining of Frequent Subgraph in the Presence of Isomorphism [C]//Third IEEE International Conference on Data Mining (ICDM 2003). [s. l.]: [s. n.], 2003: 549 - 552.
- [5] Grunwald P. A Tutorial Introduction to the Minimum Description Length Principle [EB/OL]. 2009 - 12 - 12. <http://www.csee.wvu.edu/natalias/ec568/grunwald04.pdf>.
- [6] Cook D J, Holder L B. Graph - based data mining[J]. IEEE Intelligent Systems, 2000, 15(2): 32 - 41.
- [7] Yoshida K, Motoda H, Indurkha N. Graph - based induction as a unified learning framework[J]. Journal of Applied Intelligence, 1994(4): 297 - 328.
- [8] 常新功,李敏强,寇纪淞. 基于进化算法的图形数据模式发现[J]. 模式识别与人工智能, 2008, 21(1): 116 - 121.
- [9] 常新功,寇纪淞,李敏强. 一种基于混杂 EA 的子结构发现算法[J]. 系统仿真学报, 2008, 20(6): 1626 - 1629.
- [10] Bandyopadhyay S, Maulik U, Cook D J, et al. Enhancing structure discovery for data mining in graphical databases using evolutionary programming [C]//Proceedings of the Florida Artificial Intelligence Research Symposium. Pensacola Beach, Florida, USA: AAAI Press, 2002: 232 - 236.