УДК 510.5/.6

АЛГОРИТМ АВТОМАТИЧЕСКОГО НАХОЖДЕНИЯ РЕДАКТА В НЕЧЕТКИХ МНОЖЕСТВАХ З. ПАВЛАКА

к.т.н. С.А. Марьин
(представил д.т.н., проф. Е.В. Бодянский)

Эта статья посвящена проблеме автоматического нахождения минимального набора эквивалентностей, разбивающих исходное множество на подмножества в том же количестве, что и исходный набор эквивалентностей. Для решения этой задачи исследованы: алгоритм построения полного дерева конкатенаций эквивалентностей и улучшенный алгоритм построения дерева, основанный на знаниях о построении редактов.

Введение. Математической основой теории нечетких множеств З. Павлака является отношение неразличимости [1, 2]. С каждым объектом в этой теории может быть связана некоторая информация, знания, данные. Если эта информация одинакова для нескольких объектов множества, тогда эти объекты в рамках данного множества различить невозможно. Использование этой теории позволяет решать ряд проблем, связанных с описанием объектов в терминах значений атрибутов (признаков), с определением значимости атрибутов (признаков), с автоматической генерацией правил вывода, с сокращением числа атрибутов (признаков). Для оптимального описания объектов универсума с помощью имеющихся признаков в этой теории используются редакты [1 – 3]. Под редактом понимается минимальный набор эквивалентностей, разбивающий множество объектов на подмножества в таком количестве, в каком разбивает их исходный набор эквивалентностей. Использование теории З. Павлака в экспертных системах представляется достаточно интересным, однако требует нахождения алгоритмического решения, необходимого для автоматического выполнения функций данной теории. Одной из основных функций теории нечетких множеств З. Павлака является нахождение редакта.

Постановка задачи. К настоящему моменту уже существует алгоритм автоматического получения редакта, который находит все возможные редакты, существующие для исходного множества [3]. Создание этого алгоритма базировалось на построении полного дерева конкатенации признаков, переборе всех вершин этого дерева, поиске редактов. В данной статье предлагается алгоритм построения дерева меньшего раз-

мера, чем у полного дерева. Это усеченное дерево, которое строится по тем же правилам, что и полное, однако, от него отсекаются тупиковые ветви. Отсечение этих ветвей производится согласно знаниям о правилах построения редактов. В статье проведен краткий сравнительный анализ качества работы этих двух алгоритмов.

Описание алгоритмов. На вход исходного алгоритма поступает исходное множество элементов и логическое уравнение, являющееся объединением всех эквивалентностей, заданных на этом множестве. Уравнение имеет следующий вид:

$$x^{a_1}x_1^{A_1}..x_i^{B_1}..x_k^{Z_1} \vee ... \vee x^{a_k}x_1^{A_1}..x_i^{B_m}..x_k^{Z_n} = 1. \qquad (1)$$

Алгоритм автоматического нахождения редакта включает в себя следующие шаги [3]:

1. Ищем признаки, которые можно будет по одному удалять из начального уравнения без изменения количества разбиений исходного множества на подмножества. Результатом этого этапа будет множество признаков.

2. На основе этого множества строим дерево перебора, в которое включаем все возможные конкатенации этих признаков. Результат этого этапа – полностью построенное дерево конкатенаций.

3. Просматриваем дерево и для каждого узла определяем количество подмножеств, на которые разбивается исходное множество после удаления списка признаков, приписанных этому узлу. Результатом этого этапа будет перечень узлов, при удалении которых подмножества разбиения остаются теми же, что и при начальном наборе эквивалентностей.

4. Просматриваем по порядку выписанные узлы, из полного множества признаков вычитаем признаки, приписанные каждому узлу, и получаем редакты. Результат данного этапа – список редактов.

На примере рассмотрим как работает стандартный алгоритм нахождения редакта, затем в него внесем изменения, и сравним длительность работы полученного алгоритма с длительностью исходного.

Алгоритм построения дерева, включающего все конкатенации признаков, выполняется следующим образом:

1. Берем множество редактов M, полученное на предыдущем этапе алгоритма автоматического нахождения редактов.

2. Вводим дополнительно множества списков признаков – L^1, … , L^n.

3. Во вспомогательное множество L^1 заносим все признаки, удаление которых образует простые редакты.

4. n = 1.

5. Берем множество L^n. К каждому из списков признаков этого

множества добавляем по порядку все признаки из множества L^1, таким образом, чтобы получающиеся списки признаков не совпадали.

6. Результат заносим во множество L^{n+1}.

7. Каждый элемент множества L^{n+1} проверяем, насколько подмножеств он разбивает исходное множество.

8. Те элементы, которые разбивают множество на меньшее количество подмножеств, из множества удаляем.

9. $n = n + 1$.

10. Если количество элементов множества больше одного, осуществляем переход на пункт 5.

11. $m = 1$.

12. От элементов множества L^m строим ветви дерева к соответствующим элементам множества L^{m+1}.

13. $m = m + 1$.

14. Если $m < n$, тогда идем на шаг 12, иначе шаг 15.

15. Список признаков, который находится в вершине дерева, вычитаем из общего списка признаков и снова записываем в вершину.

16. Каждую вершину проверяем, на сколько подмножеств разбивают множество признаки, записанные в данной вершине.

17. Если количество подмножеств меньше, чем при разбиении всеми эквивалентностями, тогда эту вершину удаляем из дерева.

18. Оставшиеся вершины этого дерева и есть редакты.

Работу алгоритма продемонстрируем на множестве с заданными на нем эквивалентностями (рис. 1).

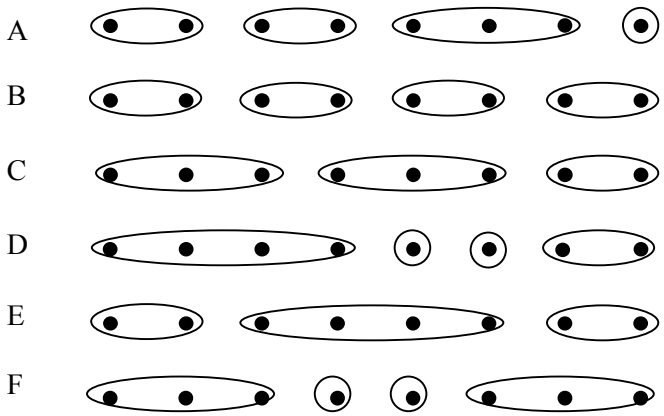

Рис. 1. Пример множества с заданными на нем эквивалентностями

Шаги 1 – 9. Находим множества:

1) L^1={B, C, E, F};
2) L^2={BC, BE, BF, CE,CF, EF};
3) L^3={BCE, BCF, BEF, CEF};
4) L^4={BCEF}.

Шаги 10 – 13. Строим дерево (рис. 2).

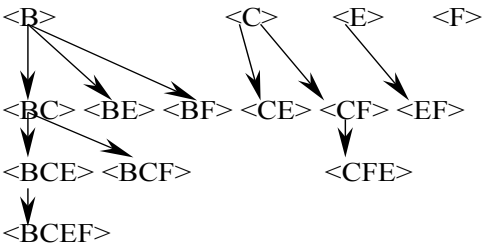

Рис. 2. Дерево конкатенаций признаков

Шаг 14. Преобразовываем дерево конкатенации признаков в дерево предполагаемых редактов (рис. 3).

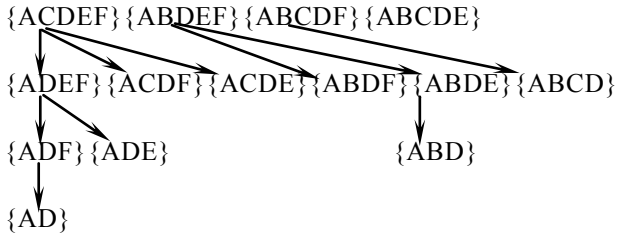

Рис. 3. Дерево предполагаемых редактов

Шаг 15 – 17. Осуществляем проверку вершин дерева на количество подмножеств и удаляем те списки признаков, которые разбивают множество на меньшее количество подмножеств. Получаем редакты: {ACDEF, ABDEF, ABCDF, ABCDE, ADEF, ACDF, ACDE, ABDE, ABCD}.

Параметры данного алгоритма (скорость выполнения, занимаемый объем памяти, длительность вычислений) можно значительно улучшить, если проверку редактов осуществлять сразу же при добавлении признака. Рассмотрим улучшенный алгоритм подробнее.

1. Берем множество редактов M, полученное на предыдущем этапе алгоритма автоматического нахождения редактов.
2. Вводим дополнительно множества списков признаков – L^1, … , L^n.
3. Во вспомогательное множество L^1 заносим все признаки, удале-

ние которых образует простые редакты.

4. n = 1.

5. Берем множество L^n. К каждому из элементов этого множества добавляем признаки из множества L^1.

6. Проверяем, на сколько подмножеств разбивает исходное множество редакт, сформированный удалением текущего списка признаков из всего множества признаков.

7. Если количество подмножеств такое же, как и исходное, тогда включаем этот элемент в множество и строим ветвь дерева.

8. Если количество подмножеств меньше, тогда ветвь дерева отсекаем, и этот элемент из множества удаляем.

9. n = n + 1.

10. Если количество элементов множества больше одного, осуществляем переход на пункт 5, иначе останавливаем алгоритм.

Результаты работы этого алгоритма будут такими же точно, что и у первого алгоритма.

Заключение. Полученный алгоритм будет выполняться быстрее, поскольку в нем строится дерево с объемом меньшим, чем у исходного дерева. В нашем примере от дерева отсекли ветви, которые шли от вершин {CE}, {BCE}. Очевидно, что точная оценка качества приведенных алгоритмов достаточно затруднена, так как она зависит от конфигурации эквивалентностей, разбивающих исходное множество на подмножества.

ЛИТЕРАТУРА

1. *Pawlak Z. Rough sets / Z. Pawlak // International Journal of Computer and Information Sciences. – 1982. – 11. – P. 341 – 356.*
2. *Pawlak Z. Rough set approach to knowledge-based decision support / Z. Pawlak // European Journal of Operational Research. – 1997. – 99. – P. 420 – 432.*
3. *Ситников Д.Э. Логический подход к нечетким множествам З. Павлака / Д.Э. Ситников, С.А. Марьин, Н.С. Кравец // Вестник Национального технического университета «ХПИ». – 2002. – № 6. – С. 30 – 39.*
4. *Amir A. A new and versatile method for association generation / A. Amir, R. Fieldman, R. Kashi // Information systems. – 1997. – Vol. 22, № 6/7. – P. 333 – 347.*

Поступила 20.04.2004

***МАРЬИН Сергей Александрович**, канд. техн. наук, доцент кафедры информационно-документных систем Харьковской государственной академии культуры. В 1994 году окончил Харьковский национальный университет радиоэлектроники. Область научных интересов – алгоритмический подход в задачах алгебры конечных предикатов, нейросети, нейро-fuzzy системы.*