УДК 519.712

ОПТИМИЗАЦИЯ РЕШАЮЩИХ ПРАВИЛ, ОСНОВАННАЯ НА МЕТОДАХ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Б. Зелоско [1]***, М.Ю. Мошков*** [2]***, И.В. Чикалов*** [2]

[1] Университет Силезии, Польша
[2] Научно-технологический университет им. короля Абдуллы, Саудовская Аравия

mikhail.moshkov@kaust.edu.sa

Поступила в редакцию 07.07.2010

Рассматривается новый подход к минимизации длины приближенных решающих правил, основанный на методах динамического программирования. Обсуждается модификация этого подхода, использующая верхнюю оценку минимальной длины решающих правил, полученную с помощью градиентного алгоритма.

Ключевые слова: тестовая таблица, решающее правило, динамическое программирование, градиентный алгоритм.

Введение

Решающие правила широко используются для представления знаний, извлеченных из больших массивов статистических или экспериментальных данных, и для построения классификаторов, способных предсказывать значения параметров новых объектов на основе информации об имеющихся [1].

Точные решающие правила могут быть «переобучены», т.е. могут зависеть от «шума», присутствующего в исходных данных. Поэтому последние годы особое внимание уделяется исследованию приближенных решающих правил [1, 2].

Широко распространено мнение, что среди моделей, имеющих сходную точность, более адекватными являются модели с более короткими описаниями (minimum description length principle [3]). Следуя этому принципу, мы заинтересованы в построении кратчайших решающих правил с заданной степенью точности. К сожалению, как для точных, так и для приближенных правил задача минимизации длины правил является NP-трудной [2]. Однако для конкретных данных мы можем так подобрать степень приближенности решающих правил, что задача минимизации длины правил будет решаться за приемлемое время.

В настоящей работе рассматривается подход к оптимизации решающих правил, основанный на методах динамического программирования. При этом исходная задача (тестовая таблица) разбивается на подзадачи (подтаблицы). Процесс разбиения заканчивается на подтаблицах, для которых существуют решающие правила заданной точности, имеющие нулевую длину левой части. Начиная с этих подтаблиц, мы строим оптимальные правила для все более и более сложных подтаблиц, включая исходную тестовую таблицу.

Изменяя порог точности решающих правил, мы можем регулировать число рассматриваемых подтаблиц и, следовательно, сложность алгоритма оптимизации. Сходный подход использовался в [4, 5] для оптимизации точных и приближенных деревьев решений.

Один из путей уменьшения сложности рассматриваемого алгоритма заключается в ограничении глубины поиска. Мы можем заранее построить решающие правила для каждой строки тестовой таблицы с помощью градиентного алгоритма [2], а затем использовать максимальную длину построенного правила в качестве верхней оценки глубины поиска.

Тестовые таблицы и α-решающие правила

Тестовая таблица – прямоугольная таблица T с n столбцами, заполненная натуральными числами. Столбцам таблицы приписаны проверки (имена проверок) $f_1, \ldots, f_n$. Строки таблицы попарно различны, и каждой строке r приписано натуральное число $d(r)$ – решение. Строки таблицы интерпретируются как наборы значений проверок $f_1, \ldots, f_n$.

Подтаблица таблицы T – таблица, полученная из T удалением некоторых строк с приписанными им решениями. Пусть $j(1), \ldots, j(k) \in$

$\in \{1, \ldots, n\}$ и $b_1, \ldots, b_k$ – натуральные числа. Обозначим $T(f_{j(1)}, b_1) \ldots (f_{j(k)}, b_k)$ подтаблицу таблицы T, содержащую те и только те строки T, которые на пересечении со столбцами $f_{j(1)}, \ldots, f_{j(k)}$ имеют числа $b_1, \ldots, b_k$ соответственно. Подобные подтаблицы, включая таблицу T, будем называть отделимыми подтаблицами таблицы T.

Пусть $r = (a_1, \ldots, a_n)$ – строка таблицы T, которой приписано решение $d(r)$. Обозначим $U(T, r)$ множество строк таблицы T, которым приписаны решения, отличные от $d(r)$. Пусть $r' \in U(T, r)$. Будем говорить, что проверка f_i отделяет строку r от строки r', если на пересечении со столбцом f_i строки r и r' содержат различные числа.

Пусть α – действительное число и $0 \le \alpha < 1$. Выражение

$$f_{i(1)} = a_{i(1)} \wedge \ldots \wedge f_{i(m)} = a_{i(m)} \to d(r) \qquad (1)$$

будем называть α-решающим правилом для r и T, если проверки $f_{i(1)}, \ldots, f_{i(m)}$ отделяют от r не менее $(1 - \alpha)|U(T, r)|$ строк из $U(T, r)$ (или, что то же самое, не могут отделить от r не более $\alpha|U(T, r)|$ строк из $U(T, r)$). Число m называется длиной рассматриваемого решающего правила.

Пусть Θ – подтаблица таблицы T и r – строка Θ. Будем говорить, что выражение (1) является (α, T)-решающим правилом для r и Θ, если проверки $f_{i(1)}, \ldots, f_{i(m)}$ не могут отделить от r не более $\alpha|U(T, r)|$ строк из $U(\Theta, r)$. Решающее правило

$$\to d(r) \qquad (2)$$

длины 0 является (α, T)-решающим правилом для r и Θ тогда и только тогда, когда $|U(\Theta, r)| \le \alpha|U(T, r)|$. Отметим, что понятие (α, T)-решающего правила для r и T совпадает с понятием α-решающего правила для r и T.

Граф $G_\alpha(T)$

В этом разделе описан алгоритм построения ориентированного ациклического графа $G_\alpha(T)$, используемого в процедуре оптимизации правил.

Пусть Θ – подтаблица таблицы T. Обозначим $E(\Theta)$ множество проверок f_i, для которых в столбце f_i таблицы Θ содержатся различные числа. Для $f_i \in E(\Theta)$ обозначим $E(\Theta, f_i)$ множество чисел, содержащихся в столбце f_i таблицы Θ.

Обозначим $S(\Theta)$ множество строк r таблицы Θ, для которых $|U(\Theta, r)| \le \alpha|U(T, r)|$. Мы будем писать $S(\Theta) = \Theta$ (соответственно $S(\Theta) \ne \Theta$), чтобы обозначить тот факт, что $S(\Theta)$ совпадает (соответственно не совпадает) с множеством строк таблицы Θ.

Рассмотрим алгоритм, строящий ориентированный ациклический граф $G_\alpha(T)$. Вершинами графа являются отделимые подтаблицы таблицы T. На каждом шаге мы обрабатываем одну вершину и помечаем ее символом *. В течение первого шага алгоритм строит граф, содержащий единственную вершину T.

Пусть алгоритм уже выполнил p шагов. Опишем шаг $(p + 1)$. Если все вершины графа помечены знаком * как обработанные, то алгоритм заканчивает свою работу и выдает полученный граф в качестве $G_\alpha(T)$. В противном случае выбираем вершину (таблицу) Θ, которая еще не была обработана. Если $S(\Theta) = \Theta$, то пометим Θ символом * и перейдем к шагу $(p + 2)$. Пусть $S(\Theta) \ne \Theta$. Для каждого $f_i \in E(\Theta)$ проведем пучок дуг из вершины Θ. Пусть $E(\Theta, f_i) = \{b_1, \ldots, b_t\}$. Тогда проведем t дуг из Θ и пометим их парами $(f_i, b_1), \ldots, (f_i, b_t)$ соответственно. Эти дуги входят в вершины $\Theta(f_i, b_1), \ldots, \Theta(f_i, b_t)$. Если некоторые из вершин $\Theta(f_i, b_1), \ldots, \Theta(f_i, b_t)$ отсутствуют в конструируемом графе, то добавим их в этот граф. Пометим вершину Θ символом * и перейдем к шагу $(p + 2)$.

Отметим, что если $E(\Theta) = \varnothing$, то (в силу того, что строки T попарно различны) Θ содержит единственную строку и, следовательно, $S(\Theta) = \Theta$.

Процедура оптимизации

Опишем процедуру оптимизации, в ходе выполнения которой для каждой вершины Θ графа $G_\alpha(T)$ каждой строке r таблицы Θ будет приписано (α, T)-решающее правило для r и Θ.

Мы будем двигаться от концевых вершин графа $G_\alpha(T)$, из которых не выходят дуги (это вершины Θ с $S(\Theta) = \Theta$), к вершине T.

Пусть Θ – концевая вершина. Припишем каждой строке r таблицы Θ решающее правило (2) длины 0.

Пусть вершина Θ не является концевой и всем строкам подтаблиц, в которые входят дуги, выходящие из Θ, уже приписаны решающие правила. Припишем решающее правило каждой строке r таблицы Θ. Если $r \in S(\Theta)$, то припишем r решающее правило (2) длины 0. Пусть $r \notin S(\Theta)$ и $r = (a_1, \ldots, a_n)$. Рассмотрим все дуги, выходящие из Θ, которым приписаны пары вида (f_i, a_i), $i \in \{1, \ldots, n\}$, и подтаблицы, в которые эти дуги входят. Выберем среди рассматриваемых подтаблиц такую подтаблицу $\Theta(f_i, a_i)$, у которой строке r приписано самое короткое решающее правило $\beta \to d(r)$. Строке r в таблице Θ припишем правило $\beta \wedge f_i = a_i \to d(r)$.

Теорема 1. Для любой вершины Θ графа $G_\alpha(T)$ и любой строки r таблицы Θ решающее правило, приписанное r после окончания процесса оптимизации, является (α, T)-решающим правилом для r и Θ, имеющим минимальную длину.

Доказательство. Утверждение теоремы докажем индукцией по вершинам графа $G_\alpha(T)$. Пусть Θ – концевая вершина графа и r – строка таблицы Θ. Поскольку $r \in S(\Theta)$, r приписано решающее правило (2) длины 0. Учитывая, что $|U(\Theta, r)| \le \alpha|U(T, r)|$, получаем, что (2) – (α, T)-решающее правило для r и Θ, имеющее минимальную длину.

Пусть Θ – вершина, не являющаяся концевой. Предположим, что для всех вершин, в которые входят дуги, выходящие из Θ, утверждение теоремы выполняется. Покажем, что утверждение теоремы выполняется и для Θ. Если $r \in S(\Theta)$, то r приписано решающее правило (2) длины 0, которое является (α, T)-решающим правилом для r и Θ, имеющим минимальную длину.

Пусть $r \notin S(\Theta)$ и $r = (a_1, \ldots, a_n)$. Тогда для некоторого $f_i \in E(\Theta)$ строке r приписано решающее правило $\beta \wedge f_i = a_i \to d(r)$, где $\beta \to d(r)$ – решающее правило, приписанное строке r в таблице $\Theta(f_i, a_i)$. Согласно предположению индукции, проверки из левой части правила $\beta \to d(r)$ не могут отделить от r не более $\alpha|U(T, r)|$ строк из $U(\Theta(f_i, a_i), r)$. Поэтому проверки из левой части правила $\beta \wedge f_i = a_i \to d(r)$ не могут отделить от r не более $\alpha|U(T, r)|$ строк из $U(\Theta, r)$. Следовательно, правило $\beta \wedge f_i = a_i \to d(r)$ является (α, T)-решающим правилом для r и Θ.

Предположим, что существует более короткое (α, T)-решающее правило для r и Θ. Поскольку $r \notin S(\Theta)$, в левой части этого правила должно быть хотя бы одно равенство вида $f_j = a_j$, где $f_j \in E(\Theta)$. Поэтому более короткое правило можно представить в виде $\gamma \wedge f_j = a_j \to d(r)$. Так как рассматриваемое правило является (α, T)-решающим правилом для r и Θ, то правило $\gamma \to d(r)$ является (α, T)-решающим правилом для r и $\Theta(f_j, a_j)$. Согласно предположению индукции, строке r в таблице $\Theta(f_j, a_j)$ приписано правило $\delta \to d(r)$, длина которого не превосходит длины правила $\gamma \to d(r)$. В соответствии с описанием процедуры оптимизации получаем, что строке r в таблице Θ приписано правило, длина которого не превосходит длины правила $\gamma \wedge f_j = a_j \to d(r)$. Полученное противоречие показывает, что правило, приписанное строке r в таблице Θ, имеет минимальную длину. Таким образом, теорема 1 доказана.

Следствие 1. После окончания процесса оптимизации каждой строке r таблицы T в графе $G_\alpha(T)$ приписано α-решающее правило для r и T, имеющее минимальную длину.

Результаты первых экспериментов

Цель экспериментов состояла в построении графов $G_0(T)$ для некоторых тестовых таблиц T из [6] и подсчете в этих графах числа вершин, не являющихся концевыми. Ясно, что в рассматриваемом случае (α = 0) некоторая отделимая подтаблица Θ таблицы T может быть концевой вершиной графа $G_0(T)$ только в том случае, когда всем строкам Θ приписано одно и то же решение. Если в рассматриваемой таблице T присутствовали одинаковые строки, то из каждой группы одинаковых строк мы оставляли только одного представителя.

- Для таблицы T «mushroom» (22 столбца и 8124 строки) число вершин в графе $G_0(T)$, не являющихся концевыми, равно 58800.
- Для таблицы «poker-hand-training-true» (10 столбцов и 25010 строк) число вершин, не являющихся концевыми, в построенном графе равно 1426236.
- Для таблицы «spect_all» (22 столбца и 267 строк) число вершин, не являющихся концевыми, в построенном графе равно 1089352.
- Для таблицы «car» (6 столбцов и 1728 строк) число вершин, не являющихся концевыми, в построенном графе равно 3008.
- Для таблицы «nursery» (8 столбцов и 12960 строк) число вершин, не являющихся концевыми, в построенном графе равно 53716.
- Для таблицы «tic-tac-toe» (9 столбцов и 958 строк) число вершин, не являющихся концевыми, в построенном графе равно 26678.

Полученные результаты показывают, что для небольших тестовых таблиц построение графа $G_\alpha(T)$ возможно иногда даже в случае α = 0. Для больших тестовых таблиц рассматриваемый метод будет применим только для значений α, близких к 1.

Градиентный алгоритм построения α-решающих правил

Определенный интерес представляют модификации процедуры оптимизации, позволяющие ограничить глубину поиска в графе $G_\alpha(T)$. Одна из возможных модификаций основана на использовании градиентного алгоритма построения α-решающих правил, позволяющего получить верхнюю оценку на длину кратчайших α-решающих правил.

Градиентный (жадный) алгоритм для построения α-решающих правил изучался в работе [2]. Пусть $r = (a_1, \ldots, a_n)$ – строка таблицы T. На каждом шаге градиентный алгоритм выбирает проверку f_i с минимальным номером i, которая отделяет от строки r максимальное число еще не отделенных строк из $U(T, r)$, и добавляет в левую часть строящегося решающего правила равенство $f_i = a_i$. Процесс заканчивается, как только решающее правило становится α-решающим правилом для r и T. Обозначим $L_{\text{greedy}}(\alpha, T, r)$ длину построенного правила и $L_{\min}(\alpha, T, r)$ – минимальную длину α-решающего правила для r и T. В работе [2] показано, что

$$L_{\text{greedy}}(\alpha, T, r) < L_{\min}(\alpha, T, r)\,(\ln \lceil (1-\alpha)|U(T, r)| \rceil - \ln \ln \lceil (1-\alpha)|U(T, r)| \rceil + 0.78)$$

и эта оценка практически не улучшаема в худшем случае.

Из рассматриваемой оценки следует что разница между $L_{\text{greedy}}(\alpha, T, r)$ и $L_{\min}(\alpha, T, r)$ может быть весьма существенной. К сожалению, при некоторых естественных предположениях о классе NP градиентный алгоритм близок по точности к наилучшим приближенным полиномиальным алгоритмам для минимизации длины α-решающих правил [2] (мы рассматриваем здесь точность алгоритмов в худшем случае). Экспериментальные и некоторые теоретические результаты [2] показывают, что «в среднем» ситуация выглядит гораздо лучше: градиентный алгоритм часто строит минимальные или весьма близкие к минимальным решающие правила.

В [2] была высказана следующая неформальная «0.5-гипотеза» относительно градиентного алгоритма: для большинства тестовых таблиц для каждой строки r в ходе построения α-решающего правила на каждом шаге градиентный алгоритм выбирает проверку, отделяющую от r не менее половины ранее не отделенных строк, которые отличны от r и которым приписаны решения, отличные от $d(r)$ (мы рассматриваем здесь общий случай, когда в тестовой таблице могут присутствовать одинаковые строки).

Нетрудно показать, что для случая, когда 0.5-гипотеза справедлива,

$$L_{\text{greedy}}(\alpha, T, r) \le \lceil \log_2 (1 / \alpha) \rceil$$

для $\alpha > 0$. В частности, $L_{\text{greedy}}(0.1, T, r) \le 4$, $L_{\text{greedy}}(0.01, T, r) \le 7$ и $L_{\text{greedy}}(0.001, T, r) \le 10$.

В работе [2] получено теоретическое подтверждение 0.5-гипотезы для случайных бинарных (заполненных 0 и 1) тестовых таблиц с m строками и n столбцами при условии $n \ge m + \log_2 m$. Экспериментальные результаты для случайно сгенерированных бинарных тестовых таблиц, приведенные в [2], подтверждают 0.5-гипотезу и для случаев, когда $n < m + \log_2 m$.

Таблица

Максимальная длина α-решающего правила, построенного градиентным алгоритмом

Тестовая таблица	Число строк	Число столбцов	α					
			0.0	0.001	0.01	0.1	0.2	0.5
balance-scale	625	4	4	4	3	2	2	1
balloons(adult+stretch)	20	4	2	2	2	2	2	1
Car	1728	6	6	6	4	2	2	1
Flags	194	26	4	4	3	2	1	1
hayes-roth.test	28	4	3	3	3	2	2	1
Krkopt	28056	6	6	4	3	2	1	1
kr-vs-kp	3196	36	11	10	6	3	2	1
Lenses	24	4	4	4	4	3	2	1
letter-recognition	20000	16	6	4	3	2	1	1
Lymphography	148	18	4	4	4	2	2	1
monk-1.test	432	6	3	3	3	2	2	1
monk-1.train	124	6	4	4	4	3	2	1
monk-2.test	432	6	6	6	5	2	2	1
monk-2.train	169	6	6	6	6	3	2	1
monk-3.test	432	6	2	2	2	2	1	1
monk-3.train	122	6	4	4	4	2	2	1
Nursery	12960	8	8	6	4	2	2	1
poker-hand-training-true	25010	10	5	3	2	1	1	1
shuttle-landing-control	15	6	2	2	2	2	2	1
soybean-small	47	35	1	1	1	1	1	1
spect_all	267	22	10	10	10	7	5	2
tic-tac-toe	958	9	5	5	4	3	2	1
Zoo	101	16	4	4	4	2	2	1

Наибольший интерес представляют результаты экспериментов с реальными данными из [6]. В [2] показано, что из 23 тестовых таблиц из [6] (их названия и размеры приведены в таблице) для 20 тестовых таблиц 0.5-гипотеза справедлива. Исключение составляют таблицы «kr-vs-kp», «spect_all» и «nursery».

В таблице указана максимальная длина α-решающих правил, построенных градиентным алгоритмом для рассматриваемых 23 тестовых таблиц и различных значений α. Основная часть данных взята из [2] (для α, равного 0.0, 0.001, 0.01 и 0.1). К этим данным мы добавили результаты экспериментов для $\alpha = 0.2$ и $\alpha = 0.5$.

Модифицированная процедура оптимизации

С помощью градиентного алгоритма построим для каждой строки таблицы T α-решающее правило. Пусть l – максимальная длина построенного правила, где максимум берется по всем строкам таблицы T.

Рассмотрим алгоритм, строящий дерево с корнем $D_{\alpha,l}(T)$, вершинами которого являются отделимые подтаблицы таблицы T. В течение первого шага алгоритм строит дерево, содержащее единственную вершину T, которая является корнем дерева.

Пусть алгоритм уже выполнил p шагов. Опишем шаг $(p + 1)$. Если все вершины дерева обработаны (помечены знаком *), то алгоритм заканчивает свою работу и выдает полученное дерево в качестве $D_{\alpha,l}(T)$. В противном случае выбираем вершину (таблицу) Θ, которая еще не была обработана (которой не приписан символ *). Если $S(\Theta) = \Theta$ или длина пути от T до Θ (число дуг в этом пути) равно l, то пометим вершину Θ символом * и перейдем к шагу $(p + 2)$. В противном случае для каждого $f_i \in E(\Theta)$ проведем пучок дуг из вершины Θ. Пусть $E(\Theta, f_i) = \{b_1, \ldots, b_t\}$. Тогда проведем t дуг из Θ и пометим их парами $(f_i, b_1), \ldots, (f_i, b_t)$ соответственно. Эти дуги входят в новые вершины $\Theta(f_i, b_1), \ldots, \Theta(f_i, b_t)$. Пометим вершину Θ символом * и перейдем к шагу $(p + 2)$.

Опишем процедуру оптимизации, в ходе выполнения которой для каждой вершины Θ дерева $D_{\alpha,l}(T)$ некоторым строкам r таблицы Θ будет приписано (α, T)-решающее правило для r и Θ.

Мы будем обрабатывать вершины дерева $D_{\alpha,l}(T)$ начиная с концевых вершин (из которых не выходят дуги) и вплоть до корня дерева. Пусть Θ – концевая вершина. Каждой строке $r \in S(\Theta)$ припишем решающее правило (2) длины 0.

Пусть вершина Θ не является концевой и все вершины, в которые входят дуги, выходящие из Θ, уже обработаны. Рассмотрим произвольную строку r таблицы Θ. Если $r \in S(\Theta)$, то припишем r решающее правило (2) длины 0. Пусть $r \notin S(\Theta)$ и $r = (a_1, \ldots, a_n)$. Рассмотрим все дуги, выходящие из Θ, которым приписаны пары вида (f_i, a_i), $i \in \{1, \ldots, n\}$, и таблицы (вершины), в которые эти дуги входят. Если ни в одной из рассматриваемых таблиц строке r не приписано решающее правило, то и в таблице Θ строке r не будет приписано правило. В противном случае выберем среди рассматриваемых таблиц такую таблицу $\Theta(f_i, a_i)$, у которой строке r приписано самое короткое решающее правило $\beta \to d(r)$. Строке r в таблице Θ припишем правило $\beta \wedge f_i = a_i \to d(r)$.

Теорема 2. После окончания модифицированного процесса оптимизации каждой строке r таблицы T в дереве $D_{\alpha,l}(T)$ приписано α-решающее правило для r и T, имеющее минимальную длину.

Доказательство. Пусть r – строка таблицы T. Индукцией по вершинам дерева $D_{\alpha,l}(T)$ докажем, что для любой вершины (таблицы) Θ, содержащей строку r, если строке r приписано решающее правило, то это правило является (α, T)-решающим правилом для r и Θ. Пусть Θ – концевая вершина дерева. Если $r \in S(\Theta)$, то r приписано правило (2) длины 0. Учитывая, что $|U(\Theta, r)| \le \alpha|U(T, r)|$, получаем, что (2) – (α, T)-решающее правило для r и Θ. Пусть $r \notin S(\Theta)$. Тогда r не приписано решающее правило. Таким образом, для концевых вершин дерева, содержащих строку r, рассматриваемое утверждение выполняется.

Пусть Θ – вершина (таблица), содержащая r и не являющаяся концевой вершиной. Предположим, что для всех вершин, в которые входят дуги, выходящие из Θ, рассматриваемое утверждение выполняется. Если $r \in S(\Theta)$ или r не приписано решающее правило, то рассматриваемое утверждение выполняется для Θ. Пусть $r \notin S(\Theta)$ и r приписано решающее правило. Тогда это решающее правило имеет вид $\beta \wedge f_i = a_i \to d(r)$, где $f_i \in E(\Theta)$ и $\beta \to d(r)$ – решающее правило, приписанное строке r в таблице (вершине) $\Theta(f_i, a_i)$. По предположению индукции $\beta \to d(r)$ – (α, T)-решающее правило для r и $\Theta(f_i, a_i)$. Поэтому $\beta \wedge f_i = a_i \to d(r)$ – (α, T)-решающее правило для r и Θ.

Из доказанного утверждения следует, что если строке r в вершине T дерева $D_{\alpha,l}(T)$ приписано решающее правило, то это правило является α-решающим правилом для r и T.

Пусть $r = (a_1, \ldots, a_n)$ – строка таблицы T и (1) – α-решающее правило для r и T, имеющее минимальную длину. Нетрудно показать, что $m \le l$, $\Theta = T(f_{i(1)}, a_{i(1)}) \ldots (f_{i(m)}, a_{i(m)})$ – вершина дерева $D_{\alpha,l}(T)$, и строке r таблицы Θ приписано решающее правило (2). Из описания модифицированной процедуры оптимизации следует, что строке r в таблице T приписано решающее правило, длина которого не превосходит m. Следовательно, строке r в таблице T приписано α-решающее правило для r и T, имеющее минимальную длину. Таким образом, теорема 2 доказана.

Нетрудно оценить сверху число вершин $N_{\alpha,l}(T)$ в дереве $D_{\alpha,l}(T)$. Если число значений каждой из проверок $f_1, \ldots, f_n$ не превосходит k, то

$$N_{\alpha,l}(T) \le \sum_{i=0}^{l} (nk)^i = ((nk)^{l+1} - 1)/(nk - 1).$$

Для тех таблиц, для которых 0.5-гипотеза справедлива, $l \le \lceil \log_2 (1 / \alpha) \rceil$. В частности, $l \le 4$ при $\alpha = 0.1$, $l \le 3$ при $\alpha = 0.2$, $l \le 2$ при $\alpha \in \{0.3, 0.4\}$ и $l \le 1$ при $\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$.

Работа выполнена при поддержке Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России», госконтракт 02.740.11.5131.

Список литературы

1. Pawlak Z. Rough sets – theoretical aspects of reasoning about data. Dordrecht: Kluwer Academic Publishers, 1991.

2. Moshkov M., Piliszczuk M., Zielosko B. Partial covers, reducts and decision rules: theory and applications. Heidelberg: Springer, 2008.

3. Rissanen J. Modelling by shortest data description // Automatica. 1978. V. 14. P. 465–471.

4. Chikalov I., Moshkov M., Zelentsova M. On optimization of decision trees // Transactions on Rough Sets IV. LNCS. V. 3700. Springer, 2005. P. 18–36.

5. Alkhalid A., Chikalov I., Moshkov M. On algorithm for building of optimal α-decision trees // RSCTC 2010. LNAI. V. 6086. Springer, 2010. P. 438–445.

6. Frank A., Asuncion A. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2010.

DECISION RULE OPTIMIZATION ON THE BASIS OF DYNAMIC PROGRAMMING METHODS

B. Zielosko, M.Yu. Moshkov, I.V. Chikalov

On the basis of dynamic programming methods, a new approach to minimization of the approximate decision rule length is considered. A modification of this approach is discussed which uses an upper bound on the decision rule minimum length obtained by the greedy algorithm.

Keywords: decision table, decision rule, dynamic programming, greedy algorithm.