© 2024 г. К.А. НАЙДЕНОВА, канд. техн. наук (ksennaid@gmail.com)
(Военно-медицинская академия имени С.М. Кирова, Санкт-Петербург),
В.А. ПАРХОМЕНКО (vladimir.parkhomenko@spbstu.ru)
(Санкт-Петербургский политехнический университет Петра Великого),
Т.А. МАРТИРОВА (martta462@yandex.ru)
(Военно-медицинская академия имени С.М. Кирова, Санкт-Петербург),
А.В. ЩУКИН, канд. техн. наук (alexander.schukin@spbstu.ru)
(Санкт-Петербургский политехнический университет Петра Великого)

ПРАВДОПОДОБНЫЕ РАССУЖДЕНИЯ В АЛГОРИТМЕ ГЕНЕРАЦИИ ХОРОШИХ КЛАССИФИКАЦИОННЫХ ТЕСТОВ

Статья посвящена применению принципов (правил) правдоподобных рассуждений к символьному машинному обучению (МО). Эти применения существенны и необходимы для увеличения эффективности алгоритмов МО. Множество таких алгоритмов порождают и используют правила в форме импликаций. Обсуждается генерация этих правил по отношению к классам объектов. Эти классификационные правила специфичны. Их посылки, называемые хорошими замкнутыми тестами (ХЗТ), покрывают максимально возможное множество объектов. Представлен один из алгоритмов генерации ХЗТ, называемый NIAGARA. Алгоритм пересмотрен и предложены новые процедуры на основе правдоподобных рассуждений. Их корректность доказывается. Используются следующие правила: импликации, запреты, индуктивные правила расширения текущих множеств целевых объектов, правила сокращения области поиска решений. Они позволяют увеличить эффективность алгоритма.

Ключевые слова: правдоподобные рассуждения, замкнутые множества, хорошие диагностические тесты, анализ хороших тестов, символьное машинное обучение.

DOI: 10.31857/S0005231024030066, **EDN:** TRSWBL

1. Введение

В статье предлагается новая, более эффективная версия алгоритма NIAGARA, представленная в [1] для генерации максимально избыточных хороших тестов (ХМИТ, ЗХТ), введенных в [2]. Увеличение эффективности алгоритма основано на введении нескольких новых импликативных правил правдоподобного вывода, которые делают возможным использовать непосредственно свойства целевых объектов, генерируемых в алгоритме. Правила правдоподобных рассуждений играют большую роль в конструировании алгоритмов извлечения знаний из данных. В свою очередь, эти правила порождаются при использовании методов машинного обучения.

Правила правдоподобного вывода продуктивны в решении следующих проблем: формирование контекстов для решаемых проблем, сокращение области поиска решений, формирование описаний объектов, выделение существенных элементов в обработке данных, выделение отношений между эле-

ментами в области поиска решений, интерпретация полученных результатов и многие другие. С этой точки зрения разумно рассматривать правила правдоподобных рассуждений как системный элемент в задачах конструирования алгоритмов обработки данных.

Статья организована следующим образом. Раздел 2 посвящен короткому введению в правдоподобный вывод. Рассматриваются только такие правила, которые применяются в новой версии NIAGARA. Раздел 3 дает определения ХМИТ. В разделе 4 обсуждается применение правил правдоподобного рассуждения в алгоритме NIAGARA-2 для генерации ХМИТ и дается пример работы алгоритма. В конце статьи обсуждаются некоторые работы, относящиеся к рассматриваемым проблемам.

2. Правила правдоподобных рассуждений

В разделе рассматриваются правила правдоподобного вывода как «если..., то...» или логические утверждения. Эти правила подразделяются на следующие категории:

— ПРИМЕРЫ, или отношения между реально наблюдаемыми объектами и фактами;

— ПРАВИЛА ПЕРВОГО ТИПА, или логические утверждения, основанные на регулярных отношениях между объектами и (или) их свойствами;

— ПРАВИЛА ВТОРОГО ТИПА, или правила правдоподобного вывода, с помощью которых правила первого типа применяются, модифицируются и извлекаются из данных.

Описание этих правил и информация об их применении дается в [1].

2.1. Правила первого типа в алгоритме NIAGARA-2

Импликации. Правило импликации есть классическое логическое правило. Оно имеет следующую форму: $x, y, z \to w$. Левая и правая части этого выражения называются антецедентом (посылкой) и следствием (заключением) соответственно. Если все значения посылки истинны, то заключение также истинно.

Запрет, или запрещающее правило. Запрет есть импликация специального вида. Это правило может быть рассмотрено как следующее выражение: $x, y, z \to$ false (никогда). Можно представить запрет как несколько импликаций. Например, $x, y \to$ не z; $x, z \to$ не y; $y, z \to$ не x.

2.2. Правила второго типа в алгоритме NIAGARA-2

Предположим, что y есть множество значений признаков некоторого объекта, наблюдаемых одновременно. Пусть `p`, `antecedent(p)` и `consequent(p)` обозначают импликацию, ее антецедент и заключение соответственно. Тогда применения импликации и запрета имеют следующий вид.

Применение импликации. Если `antecedent(p)`$\subseteq y$, тогда y можно расширить за счет `consequent(p)`: $y \leftarrow y \cup$ `consequent(p)`. Это применение использует modus ponens: если X, то Y; X; следовательно Y.

Применение запрета. Предположим, что p есть $z \to$ не k. Если antecedent(p)$\subseteq y$, тогда k есть запрещенное значение для всех расширений y. Это применение использует modus ponendo tollens: X или Y; X; следовательно не Y; X или Y; Y; следовательно не X. X и Y называются альтернативами.

В дальнейшем используем эти правила для расширения элементов начального множества ХМИТ.

3. Анализ хороших тестов

Напомним основные определения анализа хороших тестов [1–3].

Предоположим, что R и S соответственно многозначная таблица описаний объектов [5] и множество индексов объектов. Тогда $S(k)$ и $R(k)$ называют соответственно множеством индексов и описаний k-объектов, где $k \in K$ есть класс объектов, например «положительных» (+) или «отрицательных» (–), в некотором смысле.

Пусть FM, определяемое как $R \backslash R(k)$, есть множество описаний объектов, отличающихся от k класса. Обозначим через U и T соответственно множества атрибутов и значений атрибутов («значений», для краткости). Каждое значение появляется по крайней мере в описании одного объекта (в «объекте», для краткости) из R. Обозначим через n и $dom(Atr)$ соответственно общее число объектов (индексов объектов) и область значений $Atr \in U$.

Соответствие Галуа [4] от значений атрибутов к индексам объектов задается функцией $s(\cdot)$, которая принимает t, подмножество T попарно различных значений атрибутов, и возвращает подмножество индексов объектов, в описание которых входит t. Полагаем, что для значений атрибутов используются номинальные шкалы [5].

Назовем $t \subseteq T$, $s(t) \neq \emptyset$, *диагностическим тестом* для $R(k)$, если и только если $t \not\subset d$, $\forall d \in FM$. Быть диагностическим тестом t означает, что условие $s(t) \subseteq S(k)$ и импликативное правило p «если t, то k» выполняются. Очевидно, t есть antecedent(p), где p есть правило первого типа.

Обозначим через $DT(k)$ множество всех диагностических тестов для $R(k)$. Для любой пары $t, d \in DT(k)$, одно и только одно из следующих условий выполняется: $s(t) \subset s(d)$, $s(d) \supset s(t)$ и $s(t) \sim s(d)$, где sim означает отношение несравнимости.

Тогда не пустой тест t называется *хорошим тестом* для $R(k)$, если и только если $s(t) \subseteq S(k)$ и одновременно $\forall g$, $g \in S(k) \backslash s(t)$, расширение $\{s(t) \cup g\}$ не является тестом для $R(k)$.

Множество $t \subseteq T$ называется *максимально избыточным* (*замкнутым*), если для любого имликативного правила $Y \to z$ в R имеем $(Y \subseteq t) \to (z \in t)$.

Соответствие Галуа $T \to S$ задается как $s(B) = \{i \mid i \in S, B \subseteq t_i$, где t_i есть описание объекта с индексом $i\}$. Другое соответствие Галуа $S \to T$ задается как $t(s) = \{$пересечение всех $t_i \mid t_i \subseteq T, i \in s\}$.

Существует два оператора замыкания [5] $generalization\_of(t) = t'' =$

$= t(s(t))$ и $generalization\_of(s) = s'' = s(t(s))$. Множество t *замкнуто*, если $t(s(t)) = t$, и s *замкнуто*, если $s(t(s)) = s$.

Взаимосвязь между анализом хороших тестов и анализом формальных понятий была освещена в [3]. Кроме того, в статье показано, что ХМИТ является популярным классификатором. Приведены взаимосвязи с другими символьными классификаторами, например JSM-гипотезами [6, 7].

4. Алгоритм NIAGARA-2 для генерации хороших тестов с использованием правил правдоподобного вывода

4.1. Идея алгоритма

NIAGARA-2 является пакетным алгоритмом для вывода всех ХМИТ для положительных или отрицательных примеров объектов. Это новый вариант NIAGARA алгоритма, описанного в [1]. Для этой цели порождается последовательность $S_0 \subseteq \ldots \subseteq S_q \subseteq S_{q+1} \subseteq \ldots \subseteq S_{q+m}$, где S_q есть множество всех подмножеств $S(+)$ мощностью q. Операция генерализации применяется к каждому элементу $(s_q, t(s_q))$ начиная с двух начальных множеств $R(+) = \{t_1, \ldots, t_i, \ldots, t_{nt}\}$ и $S(+) = \{1, \ldots, i, \ldots, nt\}$, где nt есть число положительных объектов.

Псевдокод обсуждаемых далее процедур размещен в следующем подразделе.

Процедура DEBUT выполняет расширения элементов множества $S(+) = \{1, \ldots, i, j, \ldots, nt\}$ и конструирует множество $\{s_{12}, s_{13}, \ldots, s_{ij}, \ldots\}$, где $s_{ij} = \{i, j\}$, $1 < i < j < nt$.

Каждый элемент $s_{ij} = i, j$ такой, что $(s_{ij}, t(s_{ij}))$ не является тестом для $R(+)$, хранится в множестве Q запрещенных пар объектов. А каждый набор $s_{ij} = i, j$ такой, что $(s_{ij}, t(s_{ij}))$ есть тест для $R(+)$, обобщается, а результат $s = generalization\_of(s_{ij})$ помещается в $S(test)$.

Когда DEBUT заканчивает работу, проверяется, не соответствует ли набор s из $S(test)$ набору ХМИТ для положительных объектов. Для этой цели служит правило: если некоторый объект j, где $j = 1, \ldots, nt$, принадлежит одному и только одному s из $S(test)$, то s не может быть расширен, следовательно, s является ХМИТ, который переносится из $S(test)$ в STGOOD.

$S(test)$ частично упорядочено и содержит все $s = \{i_1, \ldots, i_q\}$, $q = 1, \ldots, nt$, удовлетворяющие условию, что $(s, t(s))$ является максимально избыточным тестом для $R(+)$, но не хорошим. STGOOD есть частично упорядоченное множество, содержащее все $s = \{i_1, \ldots, i_q\}$, $q = 1, \ldots, nt$ и удовлетворяющее условию, что $(s, t(s))$ есть ХМИТ для положительных объектов. Для каждого s в $S(test)$, $ext(s)$ есть множество всех возможных его расширений, которые соответствуют тестам для положительных примеров. Алгоритм реализует направленный выбор объектов для расширения s с использованием правила генерализации.

Процедура $SELECT(s)$ возвращает множество $select(s)$ допустимых объектов для расширения s.

$Context(s)$ определяется как множество индексов объектов, которые в текущий момент могут быть использованы для расширения $s \subseteq S(+)$: $context(s) = \{\{\cup s*\} \mid Prefix(s*) = Prefix(s), s \prec s*\}$, где $Prefix(s)$ есть первый индекс s и $\prec$ есть символ лексикографического порядка.

Введение функции $context(s)$ позволяет использовать декомпозицию алгоритма на независимые подпроцессы. Множество $V(s)$ определяется как множество индексов объектов, которые должны быть удалены из $context(s)$, чтобы избежать повторной генерации тестов. $CAND(s) = context(s) \setminus V(s)$, где $V(s) = \{\cup s*, s \subseteq s*, s* \in \{\{S(test) \setminus s\} \cup STGOOD\}\}$.

Множество $V(s)$ есть объединение всех множеств в $\{\{S(test) \setminus s\} \cup \cup STGOOD\}$, содержащих s, следовательно, s в пересечении этих множеств. Если хотим, чтобы расширение s не было включено ни в один из элементов множества $\{\{S(test) \setminus s\} \cup STGOOD\}$, надо использовать для расширения s индексы объектов, не появляющиеся одновременно с s во множестве $V(s)$.

$Select(s)$ определяется как множество индексов объектов, разрешенных для расширения s (для этой цели используется множество запрещенных пар индексов Q): $select(s) = \{i, i \in CAND(s) : (\forall j)(j \in s), i, j \notin \{STGOOD \vee Q\}\}$.

Отметим следующие правила правдоподобных рассуждений, применяющиеся в NIAGARA-2:

— правила запрета;

— правило расширения множеств с элиминированием поисковых пространств, не содержащих решений;

— импликативные правила, основанные на известных свойствах хороших тестов как формальных понятий.

4.2. Псевдокод алгоритма NIAGARA-2

В этом разделе представлен псевдокод основных процедур алгоритма NIAGARA-2. При формировании STGOOD множество s сохраняется в STGOOD, если и только если оно не вложено ни в одно множество из STGOOD. В псевдокоде используется обозначение $gnrOf$ вместо $generalization\_of$ из соображений краткости. Представим основную процедуру алгоритма.

Главный алгоритм NIAGARA-2

Вход: $R(+), R(-), nt, S(+) = \{1, \ldots, nt\}$.
Выход: TGOOD
1. DEBUT;
2. **до тех пор, пока** *S(test)*$\neq \emptyset$ **выполнять**
3. SELECT(s);
4. EXTENSION(s);
5. ANALYSIS_OF_EXTENSION(s);
6. **до тех пор, пока** STGOOD **выполнять**
7. TGOOD $\leftarrow \{t(s) \mid s \in$ STGOOD$\}$;
8. STGOOD$\setminus s$;

По сравнению с предыдущей версией алгоритма [1], осуществлены следующие изменения:

— в процедуре SELECT новая функция «determining context(s)» замещает предыдущую версию этой функции; $context(s)$ определено выше;

— в процедуре Analysis of Extension(s) выделение всех множеств, содержащих нерасширяемое множество s, и передачи их в STGOOD (строки 7–12).

Основные улучшения представлены в следствии 3 в следующем подразделе. Положительное влияние улучшений оценивается в подразделе с обсуждаемым примером. Рассмотрим подробнее основные процедуры NIAGARA-2.

Процедура `DEBUT()`

Вход: $R(+), R(-), nt, S(+) = \{1, \ldots, nt\}$.
Выход: $S(test)$, Q, STGOOD.
1. STGOOD, $Q, S(test) \leftarrow \emptyset$;
2. **цикл** $i \in \{1, \ldots, nt\}$ **выполнять**
3. $\quad sum(i) \leftarrow 0$;
4. **цикл** $i \in \{S[1], \ldots, S[nt]\}$ **выполнять**
5. $\quad$ **цикл** $j \in \{S[i+1], \ldots, S[nt]\}$ **выполнять**
6. $\quad\quad$ **если** $to\_be\_test(t\{i,j\}) = false$ **тогда**
7. $\quad\quad\quad Q \leftarrow Q \cup \{i,j\}$
8. $\quad\quad$ **иначе**
9. $\quad\quad\quad s'' \leftarrow gnrOf(\{i,j\})$;
10. $\quad\quad\quad$ **для каждого** i *в* s'' **выполнять**
11. $\quad\quad\quad\quad sum(i) \leftarrow sum(i) + 1$;
12. $\quad\quad\quad$ вставить s'' в $S(test)$ в лексикогр. порядке;
13. **для каждого** i *в* $\overline{1, nt}$ **выполнять**
14. $\quad$ **если** $sum(i) = 1$ **тогда**
15. $\quad\quad$ найти $s : i \in s, s \in S(test)$;
16. $\quad\quad$ вставить s в STGOOD в лексикогр. порядке;
17. $\quad\quad$ удалить s из $S(test)$;
18. $nts \leftarrow \{\cup s \mid s \in S(test)\}$;

Algorithm `SELECT(`s`)`

Вход: $s, nts, Q, S(test)$, STGOOD.
Выход: множество объектов $select(s)$ для возможного расширения s.
1. Сформировать $context(s)$;
2. **если** $context(s) = \emptyset$ **тогда**
3. $\quad select(s) \leftarrow \emptyset$
4. **иначе**
5. $\quad V(s) = \{\cup s', s \subseteq s', s' \in \{\{S(test) \setminus s\} \cup \text{STGOOD}\}\}$;
6. $\quad$ **если** $V(s) = \emptyset$ **тогда**
7. $\quad\quad CAND(s) \leftarrow context(s)$;
8. $\quad$ **иначе**
9. $\quad\quad CAND(s) \leftarrow context(s) \setminus V(s)$;
10. $\quad\quad$ **если** $CAND(s) = \emptyset$ **тогда**
11. $\quad\quad\quad select(s) \leftarrow \emptyset$;
12. $\quad\quad$ **иначе**
13. $\quad\quad\quad select(s) = \{i \in CAND(s) \mid (\forall j)(j \in s), \{i,j\} \notin \{\text{STGOOD} \vee Q\}\}$

Процедура EXTENSION(s)

Вход: $s, select(s), S(test)$, STGOOD.

Выход: $ext(s)$ — множество всех расширений s' из s таких, что $t(s')$ есть тест.

1. $ext(s) = \emptyset$;
2. **до тех пор, пока** $select(s) \neq \emptyset$ **выполнять**
3. **для каждого** j *из* $select(s)$ **выполнять**
4. $s_{new} \leftarrow s \cup j$
5. **если** $to\_be\_test(t(s_{new}))$=*false* **тогда**
6. удалить s_{new};
7. **иначе**
8. $s_{new} \leftarrow gnrOf(s_{new})$;
9. вставить s_{new} в $ext(s)$ в лексикогр. порядке;

Процедура ANALYSIS_OF_EXTENSION()

Вход: $ext(s), S(test)$, STGOOD.

Выход: модифицированные $S(test)$ и STGOOD.

1. **если** $ext(s) = \emptyset$ **тогда**
2. **если** $s \subset s*$, $s* \in S(test)$ **тогда**
3. перенести $s*$ из $S(test)$ в STGOOD в лексикогр. порядке;
4. удалить s;
5. **иначе**
6. перенести s из $S(test)$ в STGOOD в лексикогр. порядке;
7. **если** $||ext(s)|| = 1$ **тогда**
8. $s = s_{new}, s_{new} \in ext(s)$;
9. **если** $s \not\subset s*$, $s* \in S(test)$ **тогда**
10. перенести s из $S(test)$ в STGOOD в лексикогр. порядке;
11. **иначе**
12. перенести $s*$ из $S(test)$ в STGOOD в лексикогр. порядке;
13. удалить s;
14. **иначе**
15. **для каждого** s_{new} *в* $ext(s)$ **выполнять**
16. вставить s_{new} в $S(test)$ в лексикогр. порядке;
17. delete s;

4.3. Главные характеристики алгоритмов NIAGARA и NIAGARA-2

Предлагаемый алгоритм основан на генерации замкнутых множеств. Пусть X незамкнутое множество и $c(X) = s(t(X))$ его замыкание. Рассмотрим две возможности: $c(X) = X$ и $X \subset c(X)$. В первом случае X замкнуто и должно быть расширено. Во втором случае X не замкнуто.

Утверждение 1. Если $X \subset c(X)$, *тогда* $t(c(X)) \subset t(X)$, *но* $t(X) \subset \subset t(c(X)) \rightarrow t(X) = t(c(X))$.

Следствие 1. Если $t(X) = t(c(X))$, *тогда* $c(X)$ *может заменить* $X \in \in S(test)$ *и* X *может быть удалено из* $S(test)$ *без потери какого-либо решения.*

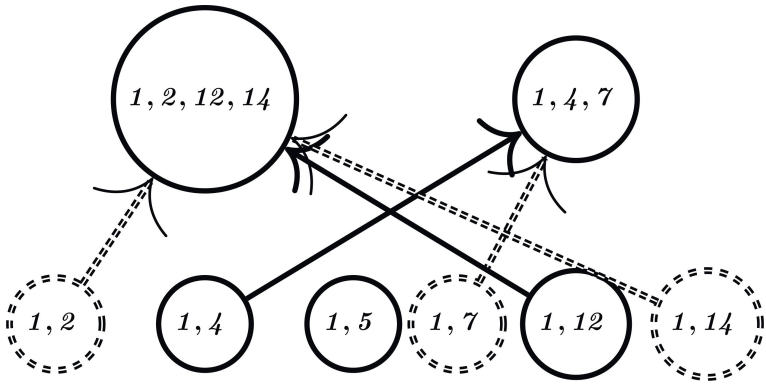

Иллюстрация использования утверждения 1 и следствия 1.

Утверждение 2. Если индекс объекта включен в одно и только одно множество в S(test), тогда это множество не может быть расширено.

Следствие 2. Если индекс объекта включен в одно и только одно множество в S(test), тогда это множество является хорошим тестом.

Следующее утверждение лежит в основании нового правила правдоподобных рассуждений, что увеличивает эффективность алгоритма NIAGARA-2 по сравнению с предыдущей версией.

Утверждение 3. Если замкнутое множество X индексов объектов содержит нерасширяемое подмножество Y, тогда X также нерасширяемое множество.

Следствие 3. Если множество X индексов объектов замкнуто, является тестом и содержит некоторое нерасширяемое подмножество, тогда X является хорошим тестом.

Назовем следствия 1, 2 и 3 правилами рассуждения 1, 2 и 3 соответственно. Хотя первое и второе утверждения (и следствия) впервые сформулированы, соответствующие правила уже использовались в NIAGARA.

Третье правило делает возможным избежать расширения множеств, содержащих нерасширяемые подмножества. Это правило с логической точки зрения является запрещающим правилом. С точки зрения решетки паттернов это правило значит, что если паттерн нерасширяемый, то все элементы его принципиального фильтра в решетке паттернов также нерасширяемы.

Рисунок дает пример работы следствия 1. Круги и сплошные стрелки, выполненные полужирным, означают замкнутые множества и связи между ними (см. диаграмму в [8]). Круги и стрелки, выполненные пунктиром, означают незамкнутые множества и связи с их замыканиями. Эти множества должны быть удалены и заменены на их замыкания.

Можно удалить из рассмотрения множества $\{1,2\}$, $\{1,7\}$, $\{1,14\}$ потому, что $t(\{1,2\}) = t(\{1,2,12,14\}$, $t(\{1,7\}) = t(\{1,4,7\})$, $t(\{1,14\}) = t(\{1,2,12,14\})$ и, тем самым, избежать расширения незамкнутых множеств, потому что по-

Таблица 1. Пример булевого представления множеств

	1	2	4	5	7	12	14	Замкнутый?	Удалить?
1	1	1				1	1	1	
2	1	1						0	1
3	1		1		1			1	
4	1			1				1	
6	1				1			0	1
7	1					1		1	
8	1						1	0	1

лучим тот же результат при расширении их замкнутых супер множеств. Таблица 1 иллюстрирует преимущество использования булевого представления множеств индексов объектов. Данные в табл. 1 согласованы с рисунком.

5. Оценка работы алгоритма

5.1. Иллюстративный пример

Данные для работы алгоритма приведены в [1]. В этом разделе приводятся результаты применения алгоритма NIAGARA-2 на множестве следующих на-

Таблица 2. Расширения элементов S(test)

S	$Context(s)$	$CAND(s)$	$Select(s)$	$Ext(s)$	Delete $s \in S(test)$	STGOOD
1, 4	5, 7, 12	5, 12	12	∅	1	1, 4, 7
1, 4, 7						
1, 5	12	∅	∅	∅	1	1, 5, 12
1, 5, 12						
1, 12	∅	∅	∅	∅	1	
2, 3, 4	7, 8.10	7, 8, 10	7, 8	2, 3, 4, 7	1	2, 3, 4, 7
2, 7	8, 10	8	8	2, 7, 8	1	2, 7, 8
2, 8	10	∅	∅	∅	1	
2, 10	∅	∅	∅	∅	1	2, 10
3, 7	8, 10, 11, 12	8, 10, 11	8, 11	∅	1	3, 7, 12
3, 7, 12						
3, 8	10, 11	10, 11	10, 11	∅	1	3, 8
3, 10	11	11	11	∅	1	3, 10
3, 11	∅	∅		∅	1	3, 11
4, 6, 8, 11	12	∅	∅	∅	1	4, 6, 8, 11
4, 6, 11	12	∅	∅	∅	1	
4, 7	8, 11, 12	8, 11, 12	8, 11, 12	4, 7, 12	1	4, 7, 12
4, 8	11, 12	11, 12	11	4, 8, 11	1	
4, 11	12	12	∅	∅	1	
4, 12	∅	∅	∅	∅	1	
7, 8	11, 12	11, 12	11	7, 8, 11	1	7, 8, 11
7, 11	12	12	∅	∅	1	
7, 12	∅	∅	∅	∅	1	
8, 10	11	11	∅	∅	1	8, 10
8, 11	∅	∅	∅	∅	1	

чальных данных. *Вход*: $S = \{1, 2, \ldots, 14\}$; $T = \{A_1, \ldots, A_{26}\}$; STGOOD $= \emptyset$; $S(test) = \emptyset$; $Q = \emptyset$. *Выход*: после применения процедуры DEBUT имеем те же множества $S(test)$, Q и STGOOD, как в таблицах 21, 22 и 23 из [1] соответственно. Таблица 2 представляет результат расширения элементов множества $S(test)$. Различие между этой таблицей и соответствующей таблицей в [1] обсуждается в следующем разделе. Результат работы NIAGARA и NIAGARA-2 одинаковый (табл. 26 в [1]).

5.2. Сравнение работы алгоритмов NIAGARA и NIAGARA-2

Иллюстративный пример был обработан с помощью NIAGARA и NIAGARA-2, чтобы найти новые ХМИТ. Преимущество NIAGARA-2 определяется двумя основными оптимизациями: функцией контекста и новым правилом 3 на основе утверждения 3. Для сравнения алгоритмов вычисляются следующие меры:

1) общее число объектов, вовлеченных в контексты (Conts);

2) общее число объектов, включенных в множества $CAND(s)$ для всех расширяемых наборов (SCand);

3) общее число объектов, включенных в множества $Select(s)$ для всех расширяемых наборов (SSelect);

4) число наборов объектов (индексов объектов), которые необходимо было расширить (SExt).

Результат сравнения представлен в табл. 3.

Таблица 3. Сравнение NIAGARA и NIAGARA-2

Алгоритм	Conts	SCand	SSelect	SExt
NIAGARA	59	55	16	25
NIAGARA-2	29	22	14	20

Меры Const и SCand уменьшились почти на 50% и 40% соответственно. Мера SSelect для новой версии алгоритма изменилась незначительно. Мера SExt уменьшилась из-за ХМИТ, которые переводились из $S(test)$ в STGOOD по новому правилу (соответствующие расширения не производились). Эти ХМИТ с номерами 8, 10, 12 в табл. 26 из [1].

6. Работы с близкой тематикой

В первую очередь авторов интересуют методы, с помощью которых решается основная проблема алгоритмов: как избежать повторяющейся генерации одного и того же объекта или как проверить единственность сгенерированного объекта. Применительно к этой проблеме некоторые методы обобщены в [9, 10]. Некоторые другие работы приведены в обзоре [11].

Важность использования контекстов аргументированно показана в [12]. Также можно отметить, что идея использования деревьев решений (класси-

фикационной структуры) для организационного принципа извлечения онтологии из текстов предложена в [13].

Правила рассуждений широко используются во всех алгоритмах, имеющих дело с генерацией частых замкнутых наборов элементов. Например, правило, аналогичное правилу 1 в NIAGARA-2, поддерживается леммой 3 в [14]. Тем не менее правила рассуждений 2 и 3 являются оригинальными и используются только для генерации GMRT.

Во многих алгоритмах применяется индуктивное правило, основанное на поуровневом расширении множеств атрибутов, наборов значений атрибутов или объектов (индексов объектов) таким образом, что множества размера q строятся из множеств размера $q-1$ предыдущего уровня. Каждое множество может быть построено тогда и только тогда, когда на предыдущем уровне есть все его собственные подмножества. В алгоритмах проверяются различные свойства множеств. Если множество не обладает требуемым свойством, то его исключают из рассмотрения. Это сильно уменьшает количество множеств всех последующих уровней, которые нужно построить.

Поуровневая генерация множеств элементов используется для извлечения ассоциативных правил из данных. Алгоритм извлечения ассоциативных правил AIS впервые был введен в [15]. Новые алгоритмы Apriori, AprioriTid и AprioriHybrid являются улучшенными версиями первого алгоритма.

Теоретико-решеточное обоснование задачи извлечения ассоциативных правил из данных на основе анализа формальных понятий было сделано в [16]. Было показано, что множество часто встречающихся понятий однозначно определяет все часто встречающиеся паттерны. Решетка частых понятий может также быть использована, чтобы получить правило генерации множеств, из которых можно вывести все ассоциативные правила.

Существуют две стратегии генерации понятий с помощью пакетных алгоритмов: нисходящая (или сверху вниз) и восходящая. Однако важно, состоит ли ведущий процесс непосредственно в генерации всех подмножеств объектов (объемов понятий) или всех подмножеств атрибутов (содержаний понятий). В [17] используется стратегия «сверху вниз» для вывода понятий в «breadth first» (сначала в ширину) стиле. Ведущим процессом этого алгоритма является генерация подмножеств объектов заданного контекста с уменьшением их мощности. В алгоритме NextClosure [18] ведущим процессом является построение лексически упорядоченных подмножеств атрибутов.

Можно заключить, что каждый алгоритм, генерирующий паттерны (наборы элементов) и подразумевающий формирование логических правил (импликация, ассоциативное правило, функциональные зависимости, формальные понятия и многие другие), использует так или иначе правила правдоподобного рассуждения, и этот факт дает право утверждать, что искомые алгоритмы можно рассматривать как модели мыслительных процессов человека.

7. Заключение

В статье проанализировано применение правдоподобных рассуждений в NIAGARA и NIAGARA-2 для генерации ХМИТ. Более того, некоторые новые процедуры, основанные на правдоподобных рассуждениях, помогают сделать алгоритм NIAGARA-2 более эффективным. Производительность алгоритма по времени улучшена за счет следующих новых процедур: расширение текущих наборов целевых объектов (с использованием импликации, основанной на свойствах замкнутых хороших тестов, запретов, правил расширения) и сокращение пространства поиска. Доказана их корректность. Будущие работы включают в себя проведение экспериментов, показывающих эффективность предложенных оптимизаций.

СПИСОК ЛИТЕРАТУРЫ

1. *Naidenova X.* An incremental learning algorithm for inferring logical rules from examples in the framework of the common reasoning process / Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, Massive Comp., Springer. 2006. V. 6. P. 89–147.
2. *Найденова К.А., Полежаева Ю.Г.* Алгоритм нахождения наилучших диагностических тестов // Сб. научн. тр. 4 Всесоюзн. конф. “Применение методов математической логики”. Институт кибернетики АН ССР. 1986. С. 87–92.
3. *Naidenova X., Buzmakov A., Parkhomenko V., Schukin A.* Notes on relation between symbolic classifiers / CEUR-WS. 2017. V. 1921. P. 88–103.
4. *Ore O.* Galois connections // Trans. Amer. Math. Soc. 1944. V. 55. P. 494–513.
5. *Ganter B., Wille R.* Formal concept analysis: mathematical foundations. Berlin/ Heidelberg: Springer, 1999.
6. *Финн В.* О машинно-ориентированной формализации правдоподобных рассуждений в стиле Ф. Бекона-Д. С. Милля // Семиотика и информатика. 1983. Т. 20. С. 35–101.
7. *Kuznetsov S.* Mathematical aspects of concept analysis // J. Math. Sci. 1996. V. 80. No. 2. P. 1654–1698.
8. *Ganter B., Kuznetsov S.* Hypotheses and version spaces // LNCS. 2003. V. 2746. P. 83–95.
9. *Carpineto C., Romano G.* Concept Data Analysis: Theory and Applications. Chichester, UK: John Wiley & Sons, Ltd, 2004.
10. *Kuznetsov S.O., Obiedkov S.A.* Comparing performance of algorithms for generating concept lattices // J. Experiment. Theoret. Artific. Intelligence. 2002. V. 14. No. 2-3. P. 189–216.
11. *Poelmans J., Ignatov D.I., Kuznetsov S.O., Dedene G.* Formal concept analysis in knowledge processing: A survey on applications // Expert Syst. Appl. 2013. V. 40. No. 16. P. 6538–6560.
12. *Galitsky B., Ilvovsky D.I., Goncharova E.* Organizing contexts as a lattice of decision trees for machine reading comprehension // Proc. of the 10th Int. Worksh. “What can FCA do for Artificial Intelligence?”. 2022. P. 75–87.

13. *Goncharova E., Ilvovsky D.I., Galitsky B.* Concept-based chatbot for interactive query refinement in product search // Proc. of the 9th Int. Worksh. “What can FCA do for Artificial Intelligence?”. 2021. P. 51–58.
14. *Wang J., Han J., Pei J.* Closet+: Searching for the best strategies for mining frequent closed itemsets // Proc. ACM SIGMOD Int. Conf. Knowl. Discov. Data Mining. 2003. P. 236–245.
15. *Agrawal R., Imieliński T., Swami A.* Mining association rules between sets of items in large databases // Proc. ACM SIGMOD Int. Conf. Management Data. 1993. P. 207–216.
16. *Zaki M.J., Ogihara M.* Theoretical foundations of association rules / 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery ACM. 1998. P. 71–78.
17. *Bordat J.P.* Calcul pratique du treillis de galois d’une correspondence // Math. Sci. Humaines. 1986. V. 96. P. 31–47.
18. *Ganter B.* Two basic algorithms in concept analysis / Formal Concept Analysis. Berlin/Heidelberg: Springer. 2010. P. 312–340.

Статья представлена к публикации членом редколлегии А.А. Галяевым.

Поступила в редакцию 08.07.2023

После доработки 13.10.2023

Принята к публикации 20.01.2024