DOI: 10.18720/SPBPU/2/id20-69

9. APPLICATION OF A LOGICAL-COMBINATORIAL NETWORK TO SYMBOLIC MACHINE LEARNING TASKS

Xenia Alexandrovna Naidenova, *cand. sci. (engineering), seniour researcher, Military Medical Academy, Russia, St.Petersburg, Akademika Lebedeva str., house 6, lit. Zh, BOX 194044, ksennaidd@gmail.com.*

Vladimir Andreevich Parkhomenko, *software engineer, Peter the Great St. Petersburg Polytechnic University, Russia, St.Petersburg, Politekhnicheskaya str., house 29, BOX 195251, Vladimir.Parkhomenko@spbstu.ru.*

Konstantin Vladimirovich Shvetsov, *cand. sci. (economics), associate professor, Peter the Great St. Petersburg Polytechnic University, Russia, St.Petersburg, Politekhnicheskaya str., house 29, BOX 195251, Konstantin.Shvetsov@spbstu.ru.*

Annotation. *In this chapter, the main tasks of symbolic machine learning are formulated for inferring logical rules and dependencies from data including functional and implicative dependencies, association rules, pattern satisfying some special requirements and many others. The Apriori algorithm solving the tasks listed above is described. A neural-like logical-combinatorial network is proposed to effectively realize Apriori-like universal inductive method for extracting logical dependencies from data. Functioning the neural-like logical-combinatorial network is described for the task of inferring good maximally redundant and irredundant classification tests from data. New approaches are proposed for reducing the computational complexity of algorithms. They are easily implemented on neural-like logical-combinatorial networks.*

Keywords. *Apriori-algorithm, neural-like logical-combinatorial network, machine learning, classification test.*

ПРИМЕНЕНИЕ ЛОГИКО-КОМБИНАТОРНОЙ НЕЙРОПОДОБНОЙ СЕТИ В ЗАДАЧАХ СИМВОЛЬНОГО МАШИННОГО ОБУЧЕНИЯ

***Ксения Александровна Найденова**, к.т.н., старший научный сотрудник, Военно-медицинская академия имени С.М.Кирова, Россия, Санкт-Петербург, ул. Академика Лебедева, дом 6, литера Ж, индекс 194044, ksennaidd@gmail.com.*

***Владимир Андреевич Пархоменко**, программист, Санкт-Петербургский политехнический университет Петра Великого, Россия, Санкт-Петербург, ул. Политехническая, дом 29, индекс 195251, Vladimir.Parkhomenko@spbstu.ru.*

***Константин Владимирович Швецов**, доцент, Санкт-Петербургский политехнический университет Петра Великого, Россия, Санкт-Петербург, ул. Политехническая, дом 29, индекс 195251, Konstantin.Shvetsov@spbstu.ru.*

Аннотация. *В главе формулируются основные задачи символьного машинного обучения, цель которых выделение из данных логических правил и закономерностей, включающих функциональные, импликативные зависимости, ассоциативные правила, паттерны, удовлетворяющие различным ограничениям и многие другие. Приводится Apriori-алгоритм, решающий перечисленные задачи символьного машинного обучения. Предлагается логико-комбинаторная нейроподобная обучаемая сеть, позволяющая эффективно реализовать Apriori-подобный универсальный индуктивный метод выделения зависимостей из данных. Функционирование нейроподобной логико-комбинаторной сети описывается на примерах выделения из данных хороших максимально избыточных и безызбыточных хороших классификационных тестов. Предлагаются новые подходы снижения вычислительной сложности алгоритмов, легко реализуемые на логико-комбинаторных нейро-подобных сетях.*

Ключевые слова. *Apriori-алгоритм, нейроподобная логико-комбинаторная сеть, машинное обучение, классификационный тест.*

Введение

Задачи символьного машинного обучения заключаются в извлечении из данных различных логических зависимостей, к которым относятся импликативные и функциональные зависимости, ассоциативные правила, ключевые паттерны, формальные понятия (formal concepts), часто встречающиеся

подмножества атрибутов (frequent itemsets), часто встречающиеся замкнутые подмножества атрибутов (frequent closed itemsets), хорошие классификационные тесты[9.1] и многие другие. Особенностью перечисленных зависимостей является их алгоритмическая эквивалентность, то есть один и тот же алгоритм можно применить к извлечению из данных любой из них.

Хотя для извлечения каждой зависимости существует множество разных алгоритмов, наиболее распространенным является Apriori, который использует индуктивный метод конструирования подмножеств элементов мощности $q + 1$ из подмножеств этих элементов мощности q (далее — q-подмножества), обладающих некоторым требуемым свойством. Эти подмножества называются кандидатами для формирования подмножеств мощности $q + 1$. При этом каждое подмножество мощности $q + 1$ может быть сформировано, если и только если все его собственные подмножества обладают требуемым свойством, то есть присутствуют во множестве кандидатов. Такого типа алгоритмы Apriori, AprioriTid, и AprioriHybrid были предложены в [9.10; 9.37] для извлечения из данных ассоциативных правил. Тот же самый алгоритмический прием используется в [9.9; 9.37] (алгоритм Titanic) для генерации так называемых ключевых паттернов (key patterns).

Аналогичное индуктивное правило используется в алгоритме TANE [9.38] для извлечения из данных функциональных зависимостей. К поиску функциональных зависимостей были сведены задачи построения хороших [9.27] и наилучших [9.21] классификационных (диагностических) тестов[9.2], решение которых также основано на Apriori-подобных алгоритмах. На Apriori-подобных алгоритмах основано извлечение наиболее часто встречающихся в данных замкнутых подмножеств значений атрибутов и для построения решетки формальных понятий (Iceberg Concept Lattice) [9.8; 9.35; 9.37]. Apriori-подобные алгоритмы применяются при обработке текстов (text mining) для извлечения ассоциативных отношений между словами, темы текста и контекста темы [9.15; 9.19].

С момента появления алгоритма Apriori (далее — Apriori) началось его интенсивное изучение и совершенствование. [9.10] предложили и исследовали

[9.1] Далее иногда мы будем опускать слово «классификационные» для простоты.

[9.2] Основные результаты работы обсуждаются на примере импликативных тестов, поэтому слово «диагностический» в основных разделах не указывается.

распараллеливание процесса вывода ассоциативных правил, [9.20] предложили алгоритм поиска ассоциативных правил в больших базах данных на основе пошаговой техники модификации правил с приходом новых данных. В работе [9.14] дана оценка верхней границы для числа паттернов-кандидатов.

В статье [9.7] дается обзор 11-ти модификаций Apriori, которые позволяют значительно уменьшить его вычислительную сложность. Например, в так называемом селективном подходе (selection approach), при определенной организации записей в базе данных удается снизить линейную задержку между порождением паттернов Apriori порядка $O(e^n)$ до порядка $O(n)$, где n — число записей, необходимых для генерации частотных множеств. Из других методов снижения вычислительной сложности традиционного Apriori, перечисленных в [9.7], отметим вычисление поддержки для кандидатов на частотность только в записях, длина которых больше или равна длине кандидатов, и учет встречаемости в записях пар символов, что сокращает число сканирований базы данных до одного раза.

Для реализации Apriori была предложена и описана в ряде работ нейроподобная логико-комбинаторная структура данных [9.23; 9.29]. Преимущество этой структуры перед традиционными реализациями алгоритма заключается в том, что на ней более эффективно реализуется основная операция Apriori — построение подмножеств мощности q+1 из подмножеств мощности q. Такие подмножества образуются за счет распространения возбуждения по связям сети от подмножеств нижнего уровня к подмножествам верхних уровней, а не за счет довольно сложной процедуры их конструирования в Apriori-алгоритмах.

Далее подробно остановимся на функционировании предлагаемой логико-комбинаторной нейроподобной обучаемой сети в задачах поиска импликативных зависимостей в данных в рамках вывода хороших тестов. Также будут рассмотрены возможности применения методов снижения вычислительной сложности, специфичных для вывода хороших тестов, при использовании нейроподобной логико-комбинаторной сети.

Глава организована следующим образом: в разделе 9.1 описывается традиционный Apriori алгоритм. В разделе 9.2 вводятся понятие хорошего теста в форме импликативной зависимости и определения его разновидностей.

Раздел 9.3 описывает нейроподобную логико-комбинаторную сеть и её функционирование. В 9.4 разделе даются примеры применения сети для вывода хороших тестов. Раздел 9.5 посвящается применению сети для оценки динамики интеллектуального развития кадетов. Раздел 9.6 рассматривает логико-комбинаторную сеть как когнитивную структуру. Главу завершает краткое заключение.

9.1. Традиционный Apriori алгоритм

Рассмотрим Apriori на примере извлечения совокупностей значений с наибольшей встречаемостью в записях базы данных (large itemsets, frequent itemsets). Пусть в базе данных имеется 59 записей, содержащих значения средней выработки работающего в натуральных единицах продукции (далее — СрВ), его стаж и возраст. Таблица 9.1 содержит некоторый фрагмент данных.

Чтобы рассмотреть Apriori подробно, введем некоторые обозначения. Пусть $I = \{i_1, i_2, \ldots, i_n\}$ — множество попарно различных символов, называемых в англоязычной литературе «items». Эти символы могут быть интерпретированы как значения некоторых атрибутов, с помощью которых описываются объекты, или образуются записи в базе данных.

Обозначим множество записей в базе данных через TDB. Каждая запись, обозначаемая через <tid, X>, состоит из множества символов (items), т.е. $X \subseteq I$ и имеет уникальный идентификатор tid. Будем говорить, что запись <tid, X> *содержит множество символов* Y, если $Y \subseteq X$. Число записей, содержащих X, называется *поддержкой* X (*support* of X) и обозначается как $sup(X)$: $sup(X) = \{tid \mid (tid, Y) \in TDB, X \subseteq Y\}$. Обозначим через $tidlist(X)$ множество индексов (идентификаторов) всех записей, содержащих X. При заданной минимальной поддержке minsup, запись Y называется *частотной* или *частой* (*frequent*), если $sup(Y) \geqslant minsup$.

Задача поиска частотных подмножеств символов заключается в поиске всех таких подмножеств символов в TDB или заданном множестве записей, для которых величина поддержки выше заданного минимального порога. Идея поиска частотных подмножеств в Apriori основана на следующем со-

Таблица 9.1

Иллюстративный пример

TID	СрВ	Стаж	Возраст
1	Низкая	1–4	25–35
2	Низкая	1–4	36–45
3	Низкая	1–4	46–55
4	Низкая	5–6	46–55
5	Низкая	1–4	25–35
6	Низкая	1–4	25–35
7	Низкая	1–4	25–35
8	Низкая	1–4	36–45
9	Низкая	1–4	36–45
10	Низкая	1–4	46–55
11	Низкая	5–6	36–45
12	Низкая	Свыше 10	46–55
13	Низкая	1–4	46–55
14	Низкая	1–4	46–55
15	Низкая	1–4	25–35
48	Высокая	7–10	25–35
49	Высокая	Свыше 10	25–35
50	Высокая	Свыше 10	25–35
51	Высокая	7–10	36–45
52	Высокая	Свыше 10	36–45
53	Высокая	7–10	25–35
54	Высокая	Свыше 10	25–35
55	Высокая	Свыше 10	25–35
56	Высокая	7–10	36–45
57	Высокая	Свыше 10	25–35
58	Высокая	7–10	36–45
59	Высокая	7–10	36–45

ображении: q-подмножество может быть частотным, если и только если все его собственные подмножества являются частотными.

На первом шаге алгоритма рассматриваются подмножества, состоящие из одного символа, и выделяются такие из них, которые удовлетворяют условию минимальной поддержки. Затем из выделенных символов формируются подмножества кандидатов на частотность, состоящие из двух символов. Для них также вычисляется поддержка и удаляются те из них, которые не удовлетворяют условию минимальной поддержки. Оставшиеся после удаления подмножества служат базой для формирования подмножеств на единицу большей мощности. Процесс итеративно продолжается, пока возможно образовать новое множество кандидатов на частотность.

Метод формирования подмножеств мощности $q + 1$ из подмножеств мощности q и вычисление поддержки для формируемых подмножеств являются основными подпроцессами алгоритма, определяющими его вычислительную сложность. На рисунке 9.1 приведен Apriori [9.1], в котором используются следующие обозначения: D – база данных; L_q – множество частотных подмножеств, удовлетворяющих требованию минимальной поддержки; C_q – множество потенциальных частотных подмножеств (кандидатов на частотность); t – запись (совокупность символов – itemset); C_t – множество кандидатов на частотность, содержащихся в записи t; apriori-gen(L_i) – подпрограмма формирования подмножеств мощности $q + 1$ из подмножеств мощности q.

Таблица 9.2 иллюстрирует работу Apriori для данных в таблице 9.1. Среди найденных наборов данных есть те, которые не удовлетворяют условию минимальной поддержки (например, $minsup = 3$), а, значит, подлежат удалению.

Таблица 9.2

Работа алгоритма Apriori на иллюстративном примере

q	Наборы значений X (элементы C_q)	Вхождения в записи (tidlist (X))	$sup(X)$
1	Низкая	1, 2, . . . , 15	15
1	Высокая	48, 49, . . . ,59	12
1	1 – 4	1,2,3,5,6,7,8,9,10,13,14,15	12
1	7 – 10	48, 51, 53, 56, 58, 59	6
1	5 – 6	4, 11	2
1	Свыше 10	12, 49, 50, 52, 54, 55, 57	7

Продолжение таблицы 9.2

q	Наборы значений X (элементы C_q)	Вхождения в записи (tidlist (X))	$sup(X)$
1	25 – 35	1, 5, 6, 7, 15, 48, 49, 50, 53, 54, 55, 57	12
1	36 – 45	2, 8, 9, 11, 51, 52, 56, 58, 59	9
1	46 – 55	3, 4, 10, 12, 13, 14	6
2	Низкая, 7 – 10	∅	0
2	Низкая, 5 – 6	4,11	2
2	Высокая, 5 – 6	∅	0
2	Низкая, Свыше 10	12	1
2	Низкая, 1 – 4	1, 2, 3, 5, 6, 7, 8, 9, 10, 13, 14, 15	12
2	Высокая, 7 – 10	48, 51, 53, 56, 58, 59	6
2	Высокая, Свыше 10	49, 50, 52, 54, 55, 57	6
2	Высокая, 1 – 4	∅	0
2	Низкая, 25 – 35	1, 5, 6, 7, 15	5
2	Низкая, 36 – 45	2, 8, 9, 11	4
2	Низкая, 46 – 55	3, 4, 10, 12, 13, 14	6
2	Высокая, 25 – 35	48, 49, 50, 53, 54, 55, 57	7
2	Высокая, 36 – 45	51, 52, 56, 58, 59	5
2	Высокая, 46 – 55	∅	0
2	1 – 4, 25 – 35	1, 5, 6, 7, 15	5
2	1 – 4, 36 – 45	2, 8, 9	3
2	1 – 4, 46 – 55	3, 10, 13, 14	4
2	7 – 10, 25 – 35	48, 53	2
2	7 – 10, 36 – 45	51, 56, 58, 59	4
2	7 – 10, 46 – 55	∅	0
2	Свыше 10, 25 – 35	49, 50, 54, 55, 57	5
2	Свыше 10, 36 – 45	52, 56, 58, 59	4
2	Свыше 10, 46 – 50	12	1
3	Низкая, 1 – 4, 25 – 35	1, 5, 6, 7, 15	5
3	Низкая, 1 – 4, 46 – 55	3, 10, 13, 14	4
3	Высокая, 7 – 10, 36 – 45	51, 56, 58, 59	4
3	Высокая, Свыше 10, 25 – 35	49, 50, 54, 55, 57	5

При применении Apriori алгоритма для вывода различных логических правил и зависимостей меняются подмножества рассматриваемых элементов:

1. L_1 = {frequent 1-itemsets};
2. **for** (q = 2; $L_{q-1} \neq \varnothing$; q++) **do begin**
3. C_q = apriori-gen(L_{q-1}); // New candidates
4. **for all** transactions $t \in D$ **do begin**
5. C_t = subset(C_q, t); // Candidates contained in t
6. **for all** candidates $c \in C_t$ **do**
7. c.count++;
8. **end**
9. $L_q = \{c \in C_q \mid c.count \geqslant minsup\}$
10. **end**
11. Answer = $\cup_q L_q$;

Рис. 9.1. Apriori алгоритм [9.1]

значений атрибутов, атрибутов, объектов, индексов объектов, индексов атрибутов или индексов значений. Собственно, алгоритм не меняется, меняется только свойство, которое проверяется у порождаемых подмножеств элементов. Например, проверяются такие свойства как «быть частым подмножеством», «быть ключевым паттерном», «быть тестом для принадлежности объектов к заданному классу», «быть хорошим тестом для принадлежности объектов к заданному классу», «быть правой частью функционального отношения» и некоторые другие.

Если конструируемое подмножество не обладает нужным свойством, то оно удаляется из рассмотрения и, следовательно, некоторые подмножества на единицу большей мощности не будут образованы. Множество перебираемых подмножеств и проверяемое свойство (PROPERTY) зависят от решаемой задачи. Алгоритм остается тем же самым и реализует индуктивное правило построения подмножеств мощности $q + 1$ из подмножеств мощности q. Таблица 9.3 отображает связь между извлекаемой из данных зависимостью и проверяемым в алгоритме свойством.

Таблица 9.3

Взаимосвязь между извлекаемыми зависимостями и тестируемым в алгоритме свойством образуемых подмножеств элементов

Выделяемая закономерность. Исходное множество элементов.	Порождаемые подмножества и их проверяемое свойство (Property)	Алгоритм, автор(ы)
Наилучшие функциональные зависимости, аппроксимирующие заданную классификацию K объектов. Множество атрибутов U.	Подмножества атрибутов $X \subseteq U$. Свойство: число классов $kl(P(X \cup K))$, в разбиении $P(X \cup K)$ множества объектов по значениям совокупности атрибутов $X \cup K$, где K — атрибут, задающий целевое разбиение объектов на непересекающиеся блоки.	Алгоритм Мегрецкой [9.27]
Хорошие максимально избыточные тесты, аппроксимирующие заданную классификацию K объектов. Множество идентификаторов начального множества тестов TEST-1, не различающих хотя бы пару объектов, принадлежащих одному и тому же классу в классификации K.	Подмножества идентификаторов тестов начального множества тестов TEST-1. Пусть $s = \{i_1, i_2, \ldots i_q\}$ — набор идентификаторов; определяется набор атрибутов X, являющийся пересечением строк из TEST-1, идентификаторы которых, перечислены в s; проверяется свойство набора атрибутов X «быть тестом для заданной классификации объектов K».	Алгоритм Найденовой и Полежаевой [9.21]
Частотные формальные концепты. Множество атрибутов.	Подмножества атрибутов. Свойство: «быть частотным подмножеством».	TITANIC [9.37]
Ключевые паттерны (key pattern) или минимальные генераторы замкнутых подмножеств атрибутов. Множество атрибутов.	Подмножества атрибутов. Свойство: «быть частотным подмножеством».	TITANIC [9.37]

Продолжение таблицы 9.3

Выделяемая закономерность. Исходное множество элементов.	Порождаемые подмножества и их проверяемое свойство (Property)	Алгоритм, автор(ы)
Минимальные нетривиальные функциональные зависимости (FD). Множество атрибутов.	Подмножества атрибутов. R — схема реляционного отношения, $X \rightarrow A$ функциональная зависимость (FD), где $X \subseteq R$, $A \in R$. Свойство: $(\forall A)(A \in R) \mid (\forall B)(B \in X)$: зависимость $X \backslash \{A,B\} \rightarrow \{B\}$ не выполняется.	TANE [9.38], FUN [9.31], FD-Mine [9.41].
Частотные паттерны. Множество символов, встречающихся в записях в базе данных.	Подмножества символов (itemsets), встречающихся в записях базы данных. Свойство: «быть частотным подмножеством».	[9.3]

9.2. Определения хороших классификационных тестов

9.2.1. Неформальное определение хорошего теста

Классификация как мыслительная операция связывает между собой несколько способов правдоподобного рассуждения. С одной стороны, классификация разбивает объекты на непересекающиеся блоки (классы), причем объекты, принадлежащие к одному и тому же классу, должны обладать некоторыми общими признаками, которые не были бы присущи объектам из других классов. Такие признаки позволяют делать заключения типа «если прямоугольник имеет 4 равные стороны, то это квадрат», «если у четырёхугольника равны все стороны, то это квадрат или ромб». С другой стороны, при классификации образуются иерархические структуры классов по отношению включения между ними. Отношения «класс-подкласс» дают возможность заключать, что «сосна это порода деревьев», а «деревья — вид растений».

Анализируя таблицу 9.1 и рассматривая строки таблицы как описания объектов в терминах трех атрибутов, мы можем сделать следующие заключения:

A. если Стаж = 1 – 4, то СрВ = Низкая;

B. если Стаж = 7 – 10, то СрВ = Высокая;

C. если Возраст = 46 – 55, то СрВ = Низкая;

D. если Стаж = 4 – 7, то СрВ = Низкая.

Однако между заключениями A и D есть большая разница: признак Стаж = 1 – 4 охватывает 12 объектов, а признак Стаж = 4 – 7 только два объекта. Первый признак обладает большей обобщающей способностью.

Рассмотренные утверждения, с логической точки зрения, есть импликативные зависимости. С другой стороны, как мы уже сказали, классификация есть разбиение объектов на классы. Между различными разбиениями могут выполняться отношения вложения, когда каждый класс одного разбиения вложен в один и только один класс другого разбиения. Такие отношения дают возможность строить иерархические классификации, например, деревья подразделяются на хвойные и лиственные, хвойные объединяют сосну, ель, лиственницу...; лиственные объединяют ольху, березу, липу...

Каждому уровню иерархической классификации соответствует разбиение объектов на непересекающиеся классы, которому сопоставляется некоторый атрибут. Зависимость между атрибутами, которые порождают такие разбиения объектов, называется функциональной и обозначается как $X \rightarrow K$, где K – атрибут, порождающий разбиение объектов на меньшее число классов, чем совокупность атрибутов X, то есть $P(X) \subseteq P(K)$, где $P(X)$, $P(K)$ обозначают соответствующие разбиения. Анализируя таблицу 1, мы можем обнаружить функциональную зависимость: Стаж, Возраст → СрВ, которая определяется следующими правилами:

– если Стаж = 1 – 4, Возраст = 25 – 35, то СрВ = Низкая;
– если Стаж = 1 – 4, Возраст = 36 – 45, то СрВ = Низкая;
– если Стаж = 1 – 4, Возраст = 46 – 55, то СрВ = Низкая;
– если Стаж = 4 – 7, Возраст = 46 – 55, то СрВ = Низкая;
– если Стаж = 4 – 7, Возраст = 36 – 45, то СрВ = Низкая;
– если Стаж = Свыше 10, Возраст = 46 – 55, то СрВ = Низкая;
– если Стаж = 7 – 10, Возраст = 25 – 35, то СрВ = Высокая;

– если Стаж = 7 – 10, Возраст = 36 – 45, то СрВ = Высокая;
– если Стаж = Свыше 10, Возраст = 25 – 35, то СрВ = Высокая;
– если Стаж = Свыше 10, Возраст = 36 – 45, то СрВ = Высокая.

Атрибуту СрВ соответствует разбиение записей в таблице 9.1 на 2 класса: $P(\text{СрВ}) = \{(1, 2, 3, \ldots, 15), (48,49, 50, \ldots, 59\}$; совокупности атрибутов Стаж, Возраст соответствует разбиение на 10 подклассов: $P(\text{Стаж, Возраст}) = \{(1, 5, 6, 7, 15), (2, 8, 9), (3), (4), (10, 13, 14), (11), (12), (48, 53), (49, 50, 54, 55, 57), (51, 56, 58, 59)\}$.

Разбиение, порождаемое некоторой совокупностью атрибутов, имеет блоки, внутри которых объекты не различаются (по значениям атрибутов), а объекты разных блоков отличаются по значению хотя бы одного атрибута этой совокупности.

Понятие хорошего классификационного теста[9.3] базируется на идее нахождения наилучших[9.4] аппроксимаций заданной классификации (разбиения) объектов и на эквивалентности трех определений теста, доказательство которой дано в [9.24].

Чтобы пояснить понятие наилучшей аппроксимации заданной классификации объектов, введем следующие обозначения: пусть $U = \{A_1, A_2, \ldots, A_m\}$ — множество атрибутов, X, Y и Z — подмножества U, $T = \{t_1, t_2, \ldots, t_n\}$— таблица описаний некоторого множества объектов в терминах значений атрибутов множества U. Пусть K — заданная классификация объектов в таблице T. Значения атрибута K могут рассматриваться как имена классов объектов в классификации, порождаемой этим атрибутом. Пусть $N = \{1, 2, \ldots, n\}$ — множество индексов (номеров) объектов в таблице и $t_i[K], t_i[X]$ — значения атрибута K и атрибутов совокупности X в описании i-ого объекта.

Набор $X \subseteq U$ атрибутов является *диагностическим* тестом для заданной классификации в таблице $T(U)$, если и только если выполняется одно из следующих эквивалентных условий [9.24; 9.25]:

1. $\forall(i, j), i, j \in N, i \neq j$: $t_i[K] \neq t_j[K] \rightarrow t_i[X] \neq t_j[X]$
   $(t_i[X] = t_j[X] \rightarrow t_i[K] = t_j[K])$;
2. $P(X) \subseteq P(K)$.
3. $X \rightarrow K$.

[9.3] Как диагностического на функциональных отношениях, так и импликативного.
[9.4] С точки зрения максимального покрытия объектов.

Условие 1 описывает отношения сходства-различия между объектами разных классов в классификации.

Обозначим через $Q(K)$ множество всех диагностических тестов для K в таблице T : $Q(K) = \{X \mid (X \subseteq U)\&(P(X) \subseteq P(K))\}$.

Если $P(X) = P(K)$, тогда X есть идеальная аппроксимация классификации K. Набор $X \subseteq U$ есть хороший диагностический тест или хорошая аппроксимация классификации в T, если и только если удовлетворяются следующие условия:

A. $X \in Q(K)$;

B. не существует такого набора атрибутов Z, $Z \subseteq U$, $Z \neq X$, что $Z \in Q(K)$ и $P(X) \subset P(Z) \subseteq P(K)$.

Каждое значение атрибута задает разбиение заданного множества объектов на 2 непересекающихся блока (объекты, в описании которых это значение встречается, и объекты, в описаниях которых оно не встречается). В силу этого оказывается возможным свести задачу поиска *импликативных зависимостей* между значениями атрибутов к нахождению хороших тестов для заданной классификации объектов на два класса.

Рассмотрим импликацию «Если Стаж = 1 – 4, то СрВ = Низкая» (из таблицы 9.1). Значение 1 – 4 атрибута Стаж не встречается ни в одной записи со значением СрВ = Высокая. Рассмотрим записи $\{1, 2, \ldots, 15\}$, соответствующие значению СрВ = Низкая. Этот класс разбивается значением Стаж = 1 – 4 на три подкласса $\{(1, 2, 3, 5, 6, 7, 8, 9, 13, 14, 15), (4, 11), (12)\}$.

Значение Стаж = 5 – 6 также не встречается со значением СрВ = Высокая. Мы имеем две импликативные зависимости: (Стаж = 1 – 4) → (СрВ = Низкая) и (Стаж = 5 – 6) → (СрВ = Низкая). Однако первая импликация имеет значительно большую поддержку, чем вторая. Далее мы будем рассматривать классификационные тесты как зависимости между значениями атрибутов, порождающие импликации с наибольшей поддержкой.

9.2.2. Определения классификационных тестов через соответствия Галуа

Обозначим через M множество значений всех рассматриваемых атрибутов $M = \bigcup_{a \in U} \mathrm{rng}(a)$, где $\mathrm{rng}(a)$ есть множество значений атрибута a. Пусть $G =$

$G_+ \cup G_-$ обозначает множество объектов, где G_+ и G_- есть не пересекающиеся множества положительных и отрицательных объектов, соответственно.

Обозначим через $\delta(g)$ описание объекта $g \in G$, тогда $D_+ = \{\delta(g) \mid g \in G_+\}$ и $D_- = \{\delta(g) \mid g \in G_-\}$ будут описаниями положительных и отрицательных объектов, соответственно.

При поиске классификационных тестов используются отображения $2^G \rightarrow 2^M$ и $2^M \rightarrow 2^G$, называемые соответствиями Галуа [9.32] и определяемые для объектов с символьными описаниями [9.12; 9.25] следующим образом: $obj(B) = \{g \in G \mid B \subseteq \delta(g), B \subseteq M\}$, эта операция возвращает все объекты, в описание которых входит совокупность значений B, и операция $val(A) = \{m \in M \mid m \in \bigcap \delta(g), g \in A, A \subseteq G\}$, которая возвращает пересечение всех описаний объектов, входящих во множество A. Свойства этих операций даны в [9.5]:

1. $A_1 \subseteq A_2 \rightarrow val(A_2) \subseteq val(A_1)$ для всех $A_1, A_2 \subseteq G$;
2. $B_1 \subseteq B_2 \rightarrow obj(B_2) \subseteq obj(B_1)$ для всех $B_1, B_2 \subseteq M$;
3. $A \subseteq obj(val(A))$ и $val(A) = val(obj(val(A)))$ для всех $A \subseteq G$;
4. $B \subseteq val(obj(B))$ и $obj(B) = obj(val(obj(B)))$ для всех $B \subseteq M$;
5. $val(\cup A_j) = \cap val(A_j)$ для всех $A_j \subseteq G$; $obj(\cup B_j) = \cap obj(B_j)$ для всех $B_j \subseteq M$.

Свойства 1, 2 относятся к расширению наборов A, B. Расширение A индексом j^* некоторого нового объекта ведет к получению более общего признака объектов: $A \subseteq (A \cup j^*)$, следовательно $val(A \cup j^*) \subseteq val(A)$.

Расширение B некоторым новым значением m ведет к уменьшению числа объектов, обладающих признаком $\{B \cup m\}$ по сравнению с числом объектом, обладающих более общим признаком B: $B \subseteq (B \cup m)$, следовательно, $obj(B \cup m) \subseteq obj(m)$.

Две операции замыкания были определены следующим образом [9.25]: $generalization\_of(B) = val(obj(B))$ и $generalization\_of(A) = obj(val(A))$. Множество A замкнуто, если $A = obj(val(A))$. Множество B замкнуто, если $B = val(obj(B))$.

При последовательности отображений $B \rightarrow obj(B) \rightarrow val(obj(B))$ получим, что $B \subseteq val(obj(B))$. Эта операция генерализации дает максимальный общий признак объектов, принадлежащих $obj(B)$.

При последовательности отображений $A \rightarrow val(A) \rightarrow obj(val(A))$ получим, что $A \subseteq obj(val(A))$. Эта операция генерализации дает максимальное множество объектов, обладающих признаком $val(A)$. Соответствия Галуа и операции замыкания являются мыслительными операциями, которые каждый человек часто использует. Дадим пример этих операций.

Пример 9.1. Предположим, что некто видел два фильма (s) с участием Жерара Депардье ($val(s)$). После этого он хочет узнать все фильмы с его участием $obj(val(s))$. Допустим, что некто знает, что Жерар Депардье играет вместе с Пьером Ришаром (t) в нескольких фильмах ($obj(t)$). Тогда он может обнаружить, что это фильмы одного и того же режиссёра Франсиса Вебера $val(obj(t))$.

Именно эти операции генерализации (замыкания) используются при поиске хороших тестов. Они позволяют рассматривать классификацию как двойственный процесс образования подмножеств объектов и соответствующих им общих признаков или процесс образования подмножеств признаков и соответствующих подмножеств объектов.

В Анализе формальных понятий (далее — АФП) [9.11] пара (A, B), где A и B замкнуты, называется *формальным понятием* и A есть *объем* понятия, а B его *содержание*. *Содержательным* и *объемным полупонятием* называют пары вида $(obj(B), B)$ и $(A, val(A))$, соответственно.

В АФП принято рассматривать бинарные описания объектов, имеющие только два значения признаков 0 и 1. Триплет (G, M, I), где I есть бинарное отношение между G и M, называют *формальным контекстом*, для которого определены операции замыкания и соответствия Галуа, обозначаемые в различных публикациях по-разному: через одно обозначение $(\cdot)'$ или через два $(\cdot)^{\uparrow}$ и $(\cdot)^{\downarrow}$ [9.40]. В отличных от бинарных контекстах вводятся иные соответствия Галуа и операции замыкания, например, в узорных структурах вводится одно обозначение соответствий Галуа через $(\cdot)^{\diamond}$ [9.12], другие обозначения можно найти в [9.17; 9.30].

Все формальные понятия, построенные по заданному формальному контексту, образуют *решетку формальных понятий* (решетку Галуа) [9.32].

Для задания классификационного контекста добавляется дополнительный целевой атрибут со множеством значений $\varepsilon \in \{+,-\}$. Целевой атрибут

задает разбиение рассматриваемых объектов на два класса: положительных и отрицательных объектов. В соответствии с целевым атрибутом в АФП разбиение формального контекста задаётся на два подконтекста: $K_\varepsilon := (G_\varepsilon, M, I_\varepsilon)$, $I_\varepsilon := I \cap (G_\varepsilon \times M)$, где $\varepsilon \in \{+, -\}$. Если необходимо, то можно добавить символ τ, который обозначает объекты без указания их принадлежности к классу положительных или отрицательных объектов [9.16]. Далее мы будем рассматривать два классификационных подконтекста: K_+, K_-, отображаемых с помощью таблицы, строки которой соответствуют описаниям объектов G_+, G_-, соответственно, а столбцы — атрибутам с символьными значениями.

Дадим определения классификационного теста и хороших классификационных тестов на *импликациях значений атрибутов*. При этом задание теста для положительных объектов (с учетом данных отрицательных объектов) будем обозначать *обучающим контекстом* $K_\pm$ (и $K_\mp$ для обратного случая)[9.5].

Определение 9.1. *Тест* для $K_\pm$ есть пара (A, B), такая что $B \subseteq M$, $A = obj(B) \neq \varnothing$, $A \subseteq G_+$ и $B \cap \delta(g) = \varnothing$ для всех $g \in G_-$.

Отметим, что тест из определения 9.1 есть содержательное полупонятие по терминологии АФП [9.18].

Определение 9.2. Тест (A,B) для $K_\pm$ является *максимально избыточным (замкнутым)*, если $obj(B \cup m) \subset A$, $m \in M \setminus B$.

Определение 9.3. Тест (A,B) для $K_\pm$ является *хорошим*, если и только если любое расширение $A^* = A \cup i$, $i \in G_+ \setminus A$ влечет, что $(A^*, val(A^*))$ не является тестом для $K_\pm$.

Хороший максимально избыточный тест (ХМИТ) является формальным понятием, а его содержание совпадает с определением минимальной положительной гипотезы в [9.16]. Другие похожие символьные классификаторы представлены в работах [9.4; 9.6; 9.13; 9.22; 9.42].

Определение 9.4. Тест (A,B) для $K_\pm$ является *безызбыточным*, если $\forall m, m \in B$ $(obj(B \setminus m), B \setminus m)$ не является тестом для $K_\pm$.

Хороший безызбыточный тест (ХБТ) является формальным понятием только в том специальном случае, когда он одновременно и максимально избыточный тест. Построение ХБТов основано на следующем соображении:

[9.5] Далее по тексту будет опущено упоминание обучающих контекстов для простоты.

ХБТ (A,B) содержится в одном и только одном ХМИТе (A^*, B^*), таком что $A^* = A$.

Пример 9.2. Обратимся к таблице 9.1. Выберем в качестве положительного класса объекты со значением «Низкая» атрибута СрВ. Тогда в этой таблице заключены следующие хорошие тесты для положительных объектов: $(\{2,8,9\}, \{(1-4), (36-45)\})$ — максимально-избыточный, но не хороший тест; $(\{1,2,3,5,6,7,8,9,10, 13, 14,15\}, \{(1 - 4)\})$ — максимально-избыточный и одновременно безызбыточный хороший тест; $(\{4,11\}, \{(5 - 6)\})$ — максимально-избыточный и одновременно безызбыточный хороший тест.

9.3. Логико-комбинаторная нейроподобная сеть для генерации подмножеств мощности $q + 1$ из подмножеств мощности q

Элементам комбинаторно-сетевой структуры соответствуют подмножества конечного множества S. Эти элементы располагаются в сети по слоям, так что каждый слой q соответствует подмножествам мощности q. Все элементы q–слоя имеют q входов или связей с элементами предыдущего $(q - 1)$–уровня. Каждый элемент возбуждается если и только если все элементы предыдущего уровня, с ним связанные, являются активными (возбужденными). Вес связей, идущих от возбужденного элемента, устанавливается равным 1, вес связей, идущих от невозбужденных элементов, устанавливается равным 0. Элемент q–слоя возбуждается, если и только если сумма весов его входов равна q. Возможное число N_q узлов в каждом слое известно заранее как число сочетаний из S по q. В процессе функционирования сети число узлов может только уменьшаться.

Преимущество сети состоит в том, что её функционирование не требует сложной техники для изменения весов связей между её элементами. Узлы сети интерпретируются в зависимости от решаемой проблемы. Свойства, проверяемые для узлов сети, тестируются с помощью присоединенных процедур.

Если возбужденный узел не обладает требуемым свойством, то он исключается из рассмотрения путем установки в 0 весов связей, идущих от него к узлам более высокого уровня.

Функционирование сети при прямом (восходящем) распространении возбуждения происходит следующим образом:

A. Шаг 1. Активируются узлы первого уровня, веса исходящих от них связей устанавливаются в 1;
B. Шаг 2. Начиная со второго уровня, если все входящие связи для узла соответствующего уровня равны 1, то проверяем, обладает ли этот узел требуемым свойством.
   - Если да, то приводим узел в активное состояние, путем установки его выходных связей в 1.
   - Если требуемое свойство для узла не удовлетворяется, то приводим этот узел в неактивное состояние путем установки его выходных связей в 0.
C. Шаг 3. Переход к анализу узлов следующего уровня.

Процесс распространения возбуждения останавливается, если ни один из узлов следующего уровня не может быть приведен в активное состояние. Отметим, что если некоторый узел не активен, то все узлы, достижимые из него, также будут неактивными, и для них не нужно тестировать выполнение требуемого свойства.

Установку в 0 весов связей, исходящих из некоторого узла, будем называть запретом, а соответствующий узел будем называть запрещенным. Установку в 1 весов связей, исходящих из некоторого узла, будем называть разрешением или допущением, а соответствующий узел будем называть допустимым.

Обратное (нисходящее) распространение возбуждения происходит от узлов верхнего уровня, которые определены как допустимые. Это может происходить при распознавании объектов по их описаниям или при восходящем процессе, когда через присоединенные процедуры непосредственно определяется допустимость некоторых узлов более высокого уровня без распространения возбуждения.

Функционирование сети при обратном (нисходящем) распространении возбуждения происходит следующим образом:

A. Шаг 1. Устанавливаются в 1 веса связей, входящих в допустимый узел (активация узла). Начиная со следующего более низкого уровня,

B. Шаг 2. Для всех узлов, к которым ведут связи, установленные в 1 на предыдущем шаге, устанавливаем их входные связи в 1.

C. Шаг 3. Переход к анализу узлов следующего более низкого уровня.

Процесс распространения возбуждения останавливается при достижении узлов самого низкого исходного уровня.

В [9.26] предложен подход двунаправленного построения классификационной решётки как взаимодействия нисходящего (от признаков) и восходящего (от объектов) процессов. Этот подход приложим и в случае построения рассматриваемой в работе логико-комбинаторной нейроподной сети.

9.4. Примеры функционирования сети

Проиллюстрируем функционирование сети на примерах поиска ХМИТов и ХБТов.

9.4.1. Вывод ХМИТов при прямом и обратном распространении возбуждения

Узлам первого уровня в сети соответствуют индексы объектов с указанием, к какому классу каждый объект принадлежит. Каждый следующий слой сети соответствует подмножествам из q индексов (объектов), где $q = \{2, \ldots, N\}$, N — число рассматриваемых объектов.

Для тестирования свойства узла, необходимо применить одно из соответствий Галуа, а именно $B = val(A)$, где $A \subseteq G_+$ есть подмножество индексов объектов, соответствующее узлу, B — подмножество значений атрибутов, полученное как пересечение описаний объектов, соответствующих узлу. Присоединенная процедура $PROPERTY(A)$ проверяет: если $val(A) \not\subseteq \delta(g)$ для всех $g \in G_-$, тогда «true», иначе «false».

В таблицах 9.4 и 9.5 даны описания положительных и отрицательных объектов для иллюстрации функционирования сети. Эти описания взяты из реальной задачи нахождения отличий семян с хорошей всхожестью (положительные примеры) от семян с плохой всхожестью (отрицательные примеры) по значениям различных признаков семян $(m_1, \ldots, m_{26})$).

Таблица 9.4

Описания положительных примеров

G	D_+	G	D_+
1	$m_1\ m_2\ m_5\ m_6\ m_{21}\ m_{23}\ m_{24}\ m_{26}$	2	$m_4\ m_7\ m_8\ m_9\ m_{12}\ m_{14}\ m_{15}\ m_{22}\ m_{23}\ m_{24}\ m_{26}$
3	$m_3\ m_4\ m_7\ m_{12}\ m_{13}\ m_{14}\ m_{15}\ m_{18}\ m_{19}\ m_{24}\ m_{26}$	4	$m_1\ m_4\ m_5\ m_6\ m_7\ m_{12}\ m_{14}\ m_{15}\ m_{16}\ m_{20}\ m_{21}\ m_{24}\ m_{26}$
5	$m_2\ m_6\ m_{23}\ m_{24}$	6	$m_7\ m_{20}\ m_{21}\ m_{26}$
7	$m_3\ m_4\ m_5\ m_6\ m_{12}\ m_{14}\ m_{15}\ m_{20}\ m_{22}\ m_{24}\ m_{26}$	8	$m_3\ m_6\ m_7\ m_8\ m_9\ m_{13}\ m_{14}\ m_{15}\ m_{19}\ m_{20}\ m_{21}\ m_{22}$
9	$m_{16}\ m_{18}\ m_{19}\ m_{20}\ m_{21}\ m_{22}\ m_{26}$	10	$m_2\ m_3\ m_4\ m_5\ m_6\ m_8\ m_9\ m_{13}\ m_{18}\ m_{20}\ m_{21}\ m_{26}$
11	$m_1\ m_2\ m_3\ m_7\ m_{19}\ m_{20}\ m_{21}\ m_{22}\ m_{26}$	12	$m_2\ m_3\ m_{16}\ m_{20}\ m_{21}\ m_{23}\ m_{24}\ m_{26}$
13	$m_1\ m_4\ m_{18}\ m_{19}\ m_{23}\ m_{26}$	14	$m_{23}\ m_{24}\ m_{26}$

Таблица 9.5

Описания отрицательных примеров

G	D_-	G	D_-
15	$m_3m_8m_{16}m_{23}m_{24}$	16	$m_7m_8m_9m_{16}m_{18}$
17	$m_1m_{21}m_{22}m_{24}m_{26}$	18	$m_1m_7m_8m_9m_{13}m_{16}$
19	$m_2m_6m_7m_9m_{21}m_{23}$	20	$m_{19}m_{20}m_{21}m_{22}m_{24}$
21	$m_1m_{20}m_{21}m_{22}m_{23}m_{24}$	22	$m_1m_3m_6m_7m_9m_{16}$
23	$m_2m_6m_8m_9m_{14}m_{15}m_{16}$	24	$m_1m_4m_5m_6m_7m_8m_{16}$
25	$m_7m_{13}m_{19}m_{20}m_{22}m_{26}$	26	$m_1m_2m_3m_5m_6m_7m_{16}$
27	$m_1m_2m_3m_5m_6m_{13}m_{18}$	28	$m_1m_3m_7m_{13}m_{19}m_{21}$
29	$m_1m_4m_5m_6m_7m_8m_{13}m_{16}$	30	$m_1m_2m_3m_6m_{12}m_{14}m_{15}m_{16}$
31	$m_1m_2m_5m_6m_{14}m_{15}m_{16}m_{26}$	32	$m_1m_2m_3m_7m_9m_{13}m_{18}$
33	$m_1m_5m_6m_8m_9m_{19}m_{20}m_{22}$	34	$m_2m_8m_9m_{18}m_{20}m_{21}m_{22}m_{23}m_{26}$
35	$m_1m_2m_4m_5m_6m_7m_9m_{13}m_{16}$	36	$m_1m_2m_6m_7m_8m_{13}m_{16}m_{18}$
37	$m_1m_2m_3m_4m_5m_6m_7m_{12}m_{14}m_{15}m_{16}$	38	$m_1m_2m_3m_4m_5m_6m_9m_{12}m_{13}m_{16}$
39	$m_1m_2m_3m_4m_5m_6m_{14}m_{15}m_{19}m_{20}m_{23}m_{26}$	40	$m_2m_3m_4m_5m_6m_7m_{12}m_{13}m_{14}m_{15}m_{16}$
41	$m_2m_3m_4m_5m_6m_7m_9m_{12}m_{13}m_{14}m_{15}m_{19}$	42	$m_1m_2m_3m_4m_5m_6m_{12}m_{16}m_{18}m_{19}m_{20}m_{21}m_{26}$
43	$m_4m_5m_6m_7m_8m_9m_{12}m_{13}m_{14}m_{15}m_{16}$	44	$m_3m_4m_5m_6m_8m_9m_{12}m_{13}m_{14}m_{15}m_{18}m_{19}$
45	$m_1m_2m_3m_4m_5m_6m_7m_8m_9m_{12}m_{13}m_{14}m_{15}$	46	$m_1m_3m_4m_5m_6m_7m_{12}m_{13}m_{14}m_{15}m_{16}m_{23}m_{24}$
47	$m_1m_2m_3m_4m_5m_6m_8m_9m_{12}m_{14}m_{16}m_{18}m_{22}$	48	$m_2m_8m_9m_{12}m_{14}m_{15}m_{16}$

Рисунок 9.2 отображает фрагмент сети при выводе ХМИТов для объектов G_+ (таблица 9.4). На этом и следующих рисунках используются следующие условные обозначения:

- пунктирные стрелки имеют вес равный 1, а сплошные — 0;
- активированные узлы обведены линией типа «проволока»;
- активированные узлы и связи, которые инициируют активацию дальнейших узлов и связей, но обозначают предыдущий проход в сети, выделены полужирным красным цветом и обведены линией типа «проволока»;
- в узлах приведены индексы объектов (за исключением рисунка 9.4, где в узлах — индексы атрибутов);
- проверяемым свойством в узлах является «быть тестом» (за исключением рисунка 9.4, где в узлах проверяемым свойством является «*не* быть тестом»).

На рисунке 9.2 все узлы второго уровня сети возбуждаются, но узлы $\{4,10\}, \{7,10\}, \{1,8\}$ и $\{1,10\}$ не удовлетворяют тестируемому свойству и запрещаются установкой в 0 их выходных связей. На третьем уровне сети возбуждаются узлы $\{4,7,8\}$ и $\{1,4,7\}$, из которых первый узел не удовлетворяет тестируемому свойству. В результате мы имеем только два допустимых узла: $\{8,10\}$, $\{1,4,7\}$. В этом фрагменте сети проверялись на удовлетворение заданного свойства 12 узлов, не требовали такой проверки 14 узлов.

Если при восходящем процессе функционирования сети некоторый узел оказывается допустимым, то для него можно выполнять операцию замыкания по соответствиям Галуа, то есть определять $A^* = obj(val(A))$. Может оказаться, что $A \subseteq A^*$, то есть совокупность индексов A^* соответствует допустимому узлу более высокого уровня (см. свойство операций замыкания 2). Тогда при нисходящем распространении возбуждения в сети, активируются все узлы, достижимые из узла, соответствующего A^*, и, следовательно, позже, при восходящем процессе распространения возбуждения, эти узлы уже будут допустимыми и непроверяемыми.

Таким образом, в отличие от традиционного Apriori, в котором можно двигаться только в направлении от $q = 1$ до $q = n$, предлагаемый в главе подход

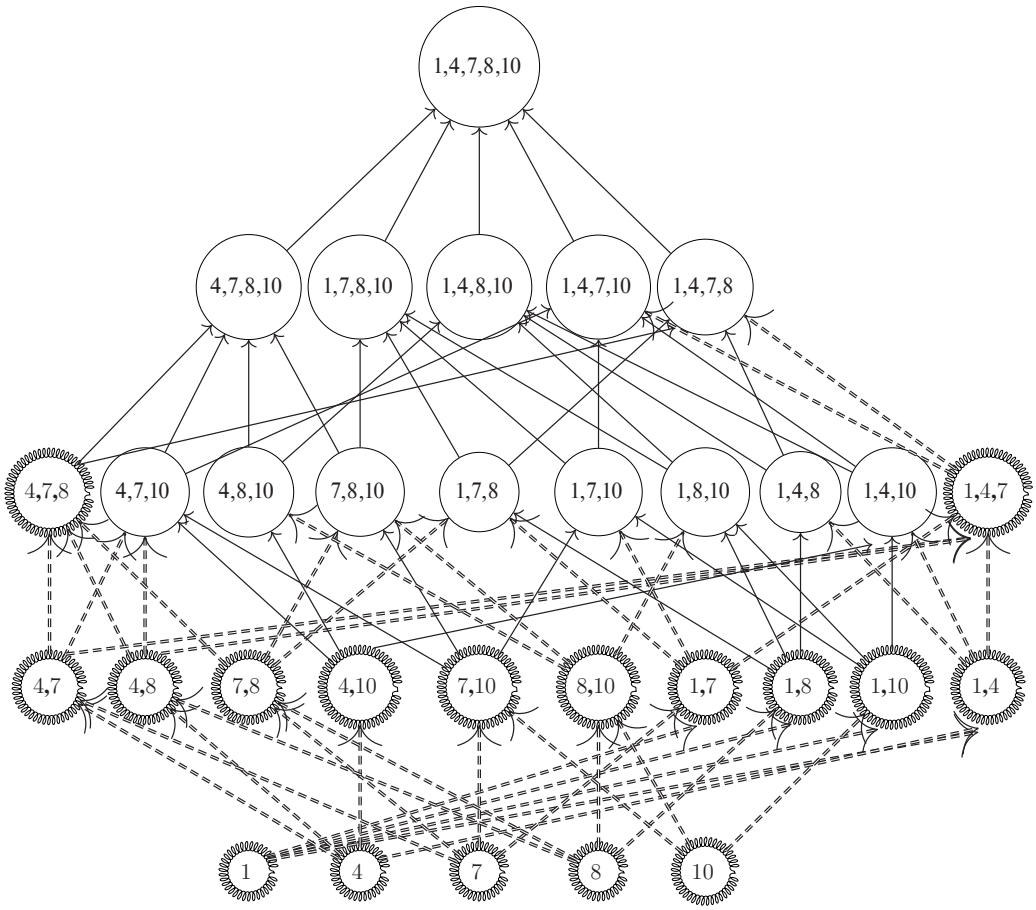

Рис. 9.2. Фрагмент восходящей генерации ХМИТов

позволяет двигаться в обратном направлении (от больших значений *n* к меньшим), а также поочередно, в обоих направлениях.

На рисунке 9.3 изображён фрагмент сети, активация которой началась с возбуждения от узла (6,4) к узлу (4,6,8,11) (стрелка и узел выделены красным). При нисходящем возбуждении сети возможна активация некоторых узлов верхнего уровня, которые, в свою очередь, могут повлечь активацию узлов нижнего уровня и так далее до того момента, когда дальнейшая активация узлов прекратится.»

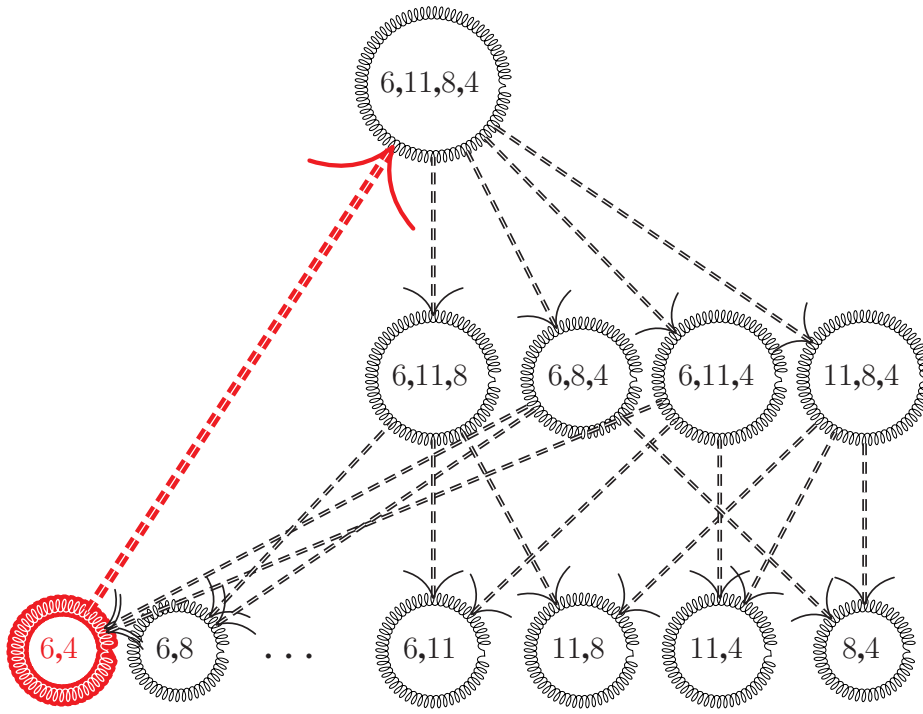

Рис. 9.3. Фрагмент нисходящей генерации ХМИТов как обратной реакции на возбуждение узла (6,4)

9.4.2. Извлечение ХБТов из некоторого ХМИТа

Пусть $X = \{m_4, m_{12}, m_{14}, m_{15}, m_{24}, m_{26}\}$ является содержанием некоторого хорошего замкнутого теста для $K_{\pm}$. В таблице 9.6 дано начальное множество отрицательных примеров, используемых для нахождения ХБТов, содержания которых находятся в X.

Нам понадобится следующее определение.

Определение 9.5. Пусть $B \subseteq M$, такое что $(\mathrm{obj}(B), B)$ есть тест для $K_{\pm}$. Значение $m \in M$, $m \in B$ назовём *существенным значением* в B, если $(\mathrm{obj}(B\backslash m), (B\backslash m))$ не тест для $K_{\pm}$.

Заметим, что содержания ХБТов должны состоять только из существенных значений (см. определение 9.4).

Опишем процедуру, на которой основано *нахождение некоторого подмножества существенных значений в содержании любого теста* для $K_{\pm}$. Представ-

Таблица 9.6

Начальное множество отрицательных примеров

G	D_-
17	$m_{24}m_{26}$
23	$m_{14}m_{15}$
30, 48	$m_{12}m_{14}m_{15}$
31	$m_{14}m_{15}m_{26}$
37,40,41,43,44,45	$m_4m_{12}m_{14}m_{15}$
39	$m_4m_{14}m_{15}m_{26}$
42	$m_4m_{12}m_{26}$
46	$m_4m_{14}m_{15}m_{24}$
47	$m_4m_{12}m_{14}$

ляет интерес нахождение частично-максимального[9.6] по числу значений подмножества $smax(B) \subset B$ (далее — *ЧМ*), такого что $(obj(B), B)$ есть тест, но $(obj(smax(B)), smax(B))$ не тест для $K_{\pm}$. Тогда $smin(B) = B \setminus$ ЧМ есть некоторое подмножество существенных значений в B.

Следующая эвристическая[9.7] процедура позволяет найти *ЧМ* в X. Начинаем с первого значения m_1 в X. Если это значение является содержанием некоторого теста для $K_{\pm}$, то оно пропускается, если нет, то прибавляем к нему значение m_2 и оцениваем функцию $to\_be\_test((obj(m_1, m_2), (m_1, m_2))$. Если значение функции равно false, тогда прибавляем следующее значение m_3 и оцениваем функцию $to\_be\_test((obj(m_1, m_2, m_3), (m_1, m_2, m_3))$. Если значение функции $to\_be\_test((obj(m_1, m_2), (m_1, m_2))$ равно true, тогда m_2 пропускается и оценивается функция $to\_be\_test((obj(m_1, m_3), (m_1, m_3))$. Процесс продолжается таким образом, пока не достигается последнее значение в X.

В этой процедуре $to\_be\_test(A,B)$: **if** $B \not\subset \delta(g), \forall g \in G_-$, **then** true **else** false.

В нашем примере находим, что ЧМ — это $\{m_4, m_{12}, m_{14}, m_{15}, m_{24}\}$ и $smin(X)$ состоит только из одного значения m_{26} (удаление этого значения приводит к тому, что оставшаяся часть $X \setminus m_{26} = \{m_4, m_{12}, m_{14}, m_{15}, m_{24}\}$ равна в точности

[9.6] Частично потому, что в тесте могут содержаться два и более максимальных по отношению частичного порядка, но не сравнимых между собой множества, не являющимися тестами.

[9.7] Поиск осуществляется с полиномиальной задержкой.

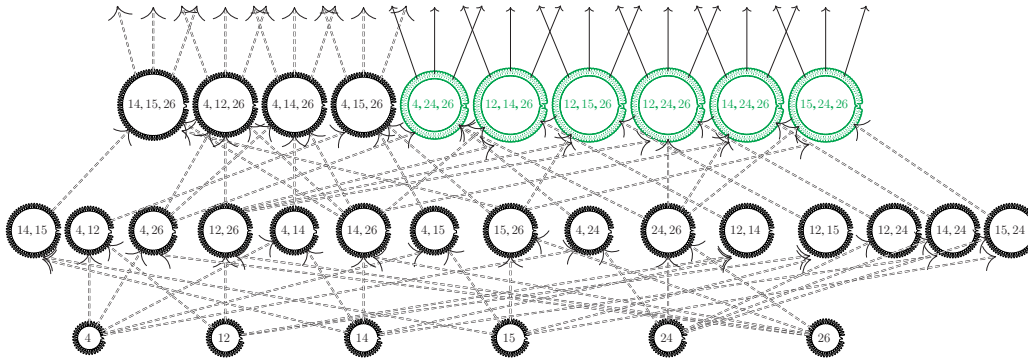

Рис. 9.4. Фрагмент сети (от нетестов к тестам) для порождения ХБТов из существенного значения m_{26}

описанию отрицательного объекта 46 (см. таблицу 9.6). Это означает, что m_{26} должно входить в содержание любого ХБТ для $K_{\pm}$.

Таким образом, можно создать начальную конфигурацию сети, содержащую только те узлы, в которые входит m_{26}. Рисунок 9.4 отображает сеть именно с такой конфигураций. В узлах сети — индексы значений атрибутов. Проверяемым свойством является «*не* быть тестом для $K_{\pm}$». Узлы, *не* обладающие этим свойством (т.е. удовлетворяющие свойству «быть тестом»), выделены полужирным зелёным курсивом.

В таблице 9.7 приведено число узлов в полной сети равное C(6,5) + C(6, 4) + C(6,3) + C(6,2) + C(6,1), где C(n,i) обозначает число сочетаний из n по i, и число действительных узлов в сети с начальной конфигурацией.

Таблица 9.7

Сравнение полной и действительно используемой конфигураций сети

Число узлов в полной сети	Число узлов в реальной сети
C(6,5) = 6	0
C(6,4) = 15	0
C(6,3) = 20	10
C(6, 2) = 15	15
C(6,1) = 6	6

В результате мы получили содержания для шести ХБТов: $\{m_{14},m_{24},m_{26}\}$, $\{m_{15},m_{24},m_{26}\}$, $\{m_{4},m_{24},m_{26}\}$, $\{m_{12},m_{14},m_{26}\}$, $\{m_{12},m_{15},m_{26}\}$, $\{m_{12},m_{24},m_{26}\}$. Конструирование начальной конфигурации сети остается вне рассмотрения в настоящей главе.

Для больших данных размер сети может быть также большим. Однако декомпозиция исходного классификационного контекста на подконтексты, предложенная в [9.29], может существенно сократить требования к размеру сетей, соответствующих подконтекстам.

Перечислим главные преимущества логико-комбинаторной нейроподобной сети:

- максимально возможный размер сети вычисляется заранее;
- возможна начальная конфигурация сети, значительно сокращающая число узлов;
- возможна декомпозиция сети на автономные подсети;
- активность узлов может управляться через присоединённые процедуры, позволяющие изменять интерпретацию узлов и самого процесса построения сети;
- сеть может использоваться при распознавании новых объектов, не участвующих в обучении сети.

9.5. Применение сети для оценки динамики интеллектуального развития кадетов

9.5.1. Описание данных

В эксперимент были вовлечены 33 женщины-кадета. Первое множество данных было сформировано в момент их поступления в вуз (2009 год), второе множество было сформировано в конце второго года обучения (2011 год). Классификационный атрибут «динамика интеллектуального развития кадет» (ДИН) основан на анализе результатов тестирования по интеллектуальным методикам Аналогии, Кубы, Силлогизмы, и Вербальная память[9.34].

Для каждой женщины вычислялась разность оценок по каждой интеллектуальной методике для двух соответствующих моментов тестирования, при-

нимая во внимание знак разности. Затем эти разности суммировались по всем методикам. Если знак суммы был «плюс», то динамика считалась положительной. Если знак разности был «минус» и абсолютное значение разности было больше, чем 2, то динамика считалась отрицательной. Если сумма была в пределах от нуля до 2, то динамика считалась нулевой. Для 33-х женщин мы получили 5, 10, и 18 персон с нулевой, отрицательной и положительной динамикой, соответственно.

Структурная модель атрибутов основана на MMPI вопроснике, адаптированном в [9.36]. Каждое значение атрибута MMPI преобразовано в значение по Т-шкале при использовании ключей вопросника и коррекционной шкалы K. Обучающая выборка дана в таблице 9.8, где используются следующие сокращения: L, F, и K это шкалы Лжи, Достоверности, и Коррекции, Hs (Ипохондрия), D (Депрессия), Hy (Истерия), Pd (Тревожность), Mf (Мужественность/Женственность), Pa (Паранойя), Pt (Психастения), Sc (Шизофрения), Ma (Сверхконтроль) и Si (Интроверсия).

В обучающей выборке (таблица 9.8) 17 женщин-кадетов с положительной (Class = 1) и 8 женщин кадетов с отрицательной (Class = 2) динамикой. Измерения по тесту MMPI проводились при поступлении в ВУЗ для того, чтобы изучить отличия кадетов с отрицательной и положительной динамикой по их личностным качествам.

В таблице 9.8 значения характеристик преобразованы из Т-шкалы в 5-балльную шкалу с применением правил, представленных в таблице 9.9 [9.36].

В таблице 9.10 приведены объемы и содержания всех ХМИТов, отличающих женщин-кадетов с отрицательной динамикой от женщин-кадетов с положительной динамикой интеллектуального развития по их личностным характеристикам.

Отметим, что личностный профиль женщин-кадетов с отрицательной динамикой соответствует так называемому «углубленному» профилю (по Собчик) в отличие от «выпуклого» профиля для женщин-кадетов с положительной динамикой. Для «углубленного» профиля характерны более низкие оценки в середине профиля (в данном случае нормальные оценки по шкалам Истерия, Тревожность и Паранойя) и более высокие оценки (выше нормы) по шкалам Ложь, Достоверность и Сверхконтроль. Причем завышенная оцен-

Таблица 9.8

Исходные данные для моделирования динамики развития кадетов

No	L	F	K	Hs	D	Hy	Pd	Mf	Pa	Pt	Sc	Ma	Si	Cl	Dyn
1	4	3	5	3	4	3	3	4	3	4	4	4	2	2	-7
2	4	4	5	3	4	3	3	3	2	4	4	4	2	2	-3
3	4	3	4	3	3	3	3	3	3	3	3	4	3	2	-3
4	5	4	5	3	4	3	4	2	4	3	4	3	3	2	-5
5	4	3	4	3	3	3	3	4	3	3	3	3	3	2	-4
6	3	3	4	3	3	3	3	3	3	3	3	4	2	2	-2
7	5	3	5	4	4	4	4	3	4	4	4	4	2	2	-7
8	4	3	4	3	3	3	3	4	3	3	3	3	2	2	-2
9	5	3	5	3	3	3	4	2	2	3	4	4	2	2	-2
10	4	3	4	3	2	2	3	2	2	3	3	4	3	2	-2
1	3	3	5	3	4	4	4	4	3	4	4	4	2	1	3
2	2	3	4	3	2	3	3	3	3	3	3	3	2	1	2
3	3	3	5	3	3	3	3	2	4	4	4	3	3	1	3
4	3	3	4	3	3	3	4	4	2	3	3	5	3	1	4
5	3	3	5	3	3	4	4	4	3	4	4	3	3	1	6
6	4	2	4	3	4	4	4	4	2	3	3	3	2	1	4
7	3	3	3	2	4	2	3	4	3	2	3	5	2	1	2
8	3	3	4	2	3	3	4	4	3	3	4	3	2	1	2
9	2	4	5	3	4	4	3	4	4	4	4	4	2	1	1
10	3	3	5	3	2	3	3	2	4	3	3	4	2	1	1
11	3	4	4	3	3	3	3	4	2	3	3	4	2	1	4
12	3	3	4	3	3	4	2	4	3	3	3	4	2	1	10
13	5	3	5	4	3	4	4	4	4	4	4	4	2	1	4
14	3	3	4	3	4	3	4	4	2	4	4	4	2	1	5
15	3	3	4	3	3	3	3	2	2	3	3	4	3	1	2
16	5	3	4	3	4	2	3	3	4	3	3	3	3	1	3
17	3	3	5	3	4	4	3	5	4	4	4	3	2	1	5
18	5	4	5	3	4	3	4	1	3	4	4	4	3	1	1
1	4	4	5	3	3	3	3	1	3	3	4	4	3	3	-1
2	3	4	4	4	3	4	4	3	4	3	4	5	3	3	-1
3	4	3	4	3	2	2	4	2	2	3	3	4	2	3	0
4	3	4	4	3	2	3	4	1	2	3	4	4	3	3	0
5	4	5	3	2	2	2	3	2	3	2	3	5	2	3	0

Таблица 9.9

Шкала интервалов для MMPI методики

Балл	Характеристика интервала	Границы
1	Существенно ниже нормы	$\leqslant 30$T
2	Ниже нормы	[31 – 44]T
3	Норма	[45 – 55]T
4	Выше нормы	[56 – 69]T
5	Существенно выше нормы	$\geqslant 70$T

Таблица 9.10

Хорошие тесты для 2-го класса кадетов (Шаг 7)

Номер теста	L	F	K	Hy	Pd	Mf	Pa	Pt	Ma	Cl	Номера кадетов
1	4			3	3					2	{1,2,3,5,8}
2						3			4	2	{2,3,4,7}
5		4	5	3						2	{2,6}
6		3		3	3		3		4	2	{1,3,4}

ка по шкале Сверхконтроль не сочетается с завышенной оценкой по шкалам Лжи, Достоверности и Коррекции в некоторых малочисленных подгруппах женщин-кадетов. Полученные результаты должны уточняться с расширением обучающей выборки.

9.5.2. Функционирование сети при построении ХМИТ-ов в задачах анализа личностных характеристик женщин-кадетов с положительной и отрицательной динамикой интеллектуального развития

Рассмотрим три варианта функционирования логико-комбинаторной нейроподобной сети для вывода ХМИТ-ов по исходным данным в таблице 9.8:

1. с прямым распространением возбуждения;
2. с применением операции замыкания Галуа;
3. с вычислением начального множества ХМИТов.

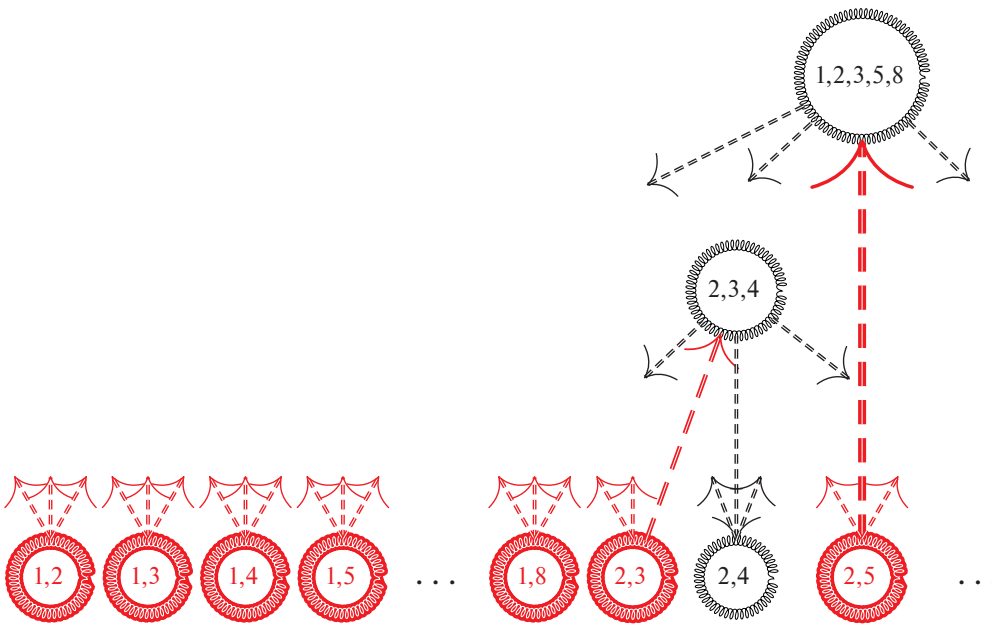

Рис. 9.5. Восходящие возбуждения до достижения узла (2,5)

Как и ранее, узлам сети соответствуют подмножества индексов объектов (в данном случае женщин-кадетов).

В первом варианте было образовано: узлов из двух индексов — 28, из них 17 соответствуют тестам (один из которых ХМИТ); узлов из трех индексов — 31, из них 15 соответствуют тестам (один из которых ХМИТ); узлов из четырех индексов — 6, из них 1 соответствует ХМИТу; узлов из пяти индексов — 1, который соответствует ХМИТу.

Во втором варианте после проверки первых 10-ти узлов, состоящих из двух индексов, и применения к ним операции замыкания Галуа были получены подмножества индексов (1,3,4), (2,3,4) и (1,2,3,5,8) для узлов более высоких уровней в сети и соответствующих тестам (рисунок 9.5). Распространение возбуждения от этих узлов к нижним слоям сети позволило возбудить без проверки 24 узла (5 узлов из четырех индексов, 12 узлов из трех индексов и 7 узлов из двух индексов), что отображено на рисунке 9.6.

В третьем варианте такой прием сокращения перебора как вычисление начального заполнения множества хороших тестов [9.28] позволил получить в данном случае сразу (без функционирования сети) все ХМИТы для отличе-

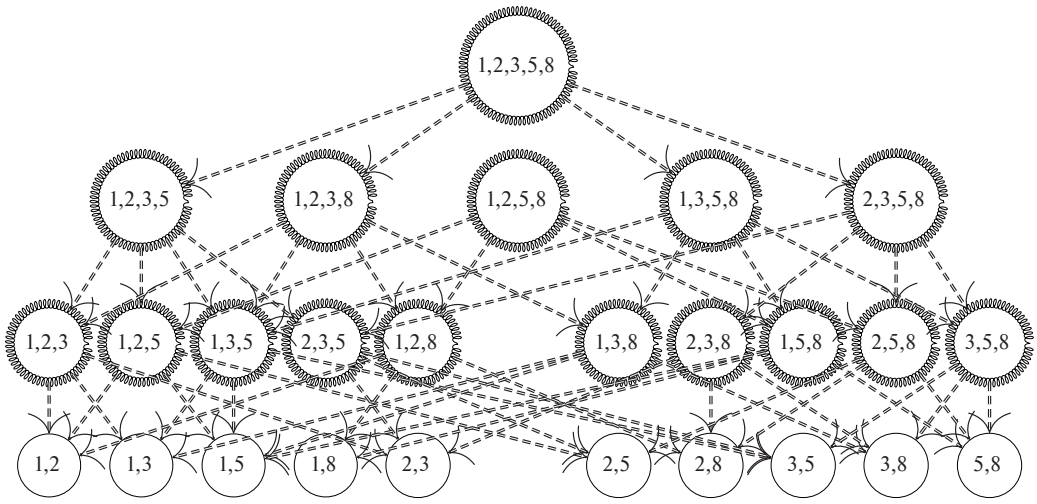

Рис. 9.6. Нисходящее распространение возбуждения узлов сети

ния женщин-кадетов с отрицательной динамикой от женщин-кадетовов с положительной динамикой по их личностным характеристикам. Иллюстрация этого приема дана в таблице 9.11.

9.6. Логико-комбинаторная сеть как когнитивная структура

Отметим, что модель нейрона персептронного типа с суммированием весов связей, широко используемая в работах по искусственному интеллекту, критикуется специалистами, изучающими физиологию человека, так как эта модель не может объяснить работу функциональных систем организма человека [9.2].

В работе [9.39, с. 248] функция нейрона определяется как преобразование значений предикатов $P_1, P_2, \ldots, P_k$, которые обозначают возбуждения, приходящие через аксоны на вход нейрона, в значения некоторого выходного предиката P_0, то есть по существу как логическая функция. Известно, что каждый нейрон имеет рецептивное поле, рецепторы которого вызывают возбуждения безусловно (без обучения).

В процессе обучения нейрон получает подкрепление или запрет в зависимости от различных условий и целевых установок организма.

Таблица 9.11

Содержание значений атрибутов в объектах и объемах тестов

Имя атрибута	m	$Obj(m)$	Объемы тестов, содержащихся в $Obj(m)$
L	4	1, 2, 3, 5, 8	1,2,3,5,8
L	3	4	∅
L	5	6,7	∅
F	3	1,3,4,5,7,8	1,3,4
F	4	2,6	2,6
K	5	1,2,6,7	2,6
K	4	3,4,5,8	∅
Hy	3	1,2,3,4,5,6,8	1,2,3,5,8
Hy	4	7	∅
Pd	3	1,2,3,4,5,8	1,2,3,5,8
Pd	4	6,7	∅
Mf	4	1,5,8	∅
Mf	3	2,3,4,7	2,3,4,7
Mf	2	6	∅
Pa	3	1,3,4,5,8	1,3,4
Pa	2	2	∅
Pa	4	6,7	∅
Pt	4	1,2,7	∅
Pt	3	3,4,5,6,8	∅
Ma	4	1,2,3,4,7	1,2,3
Ma	3	5,6,8	5,8

Один и тот же нейрон может участвовать в работе различных функциональных систем организма. Можно предположить, что отдельный нейрон не имеет какой-то определенной специфики (или, по крайней мере, что существуют такие нейроны). Среди возбуждающих входов нейрона существуют мотивационные и эмоциональные входы, и в каждый момент времени в зависимости от целей и состояния организма нейрон срабатывает в рамках одной какой-нибудь функциональной системы. Также весьма вероятно [9.39], что существует механизм, управляющий частотой возбуждения нейрона, опреде-

ляемой как отношение числа стимулирующих возбуждений на входе к числу полученных подкреплений.

Модель нейрона, как логического элемента сети, представленная в настоящей работе, совпадает во многих аспектах с формальной моделью нейрона, описанной в работах [9.39, с. 248; 9.33, с. 101–102]:

- нейрон (узел в сети), получает возбуждения от нейронов предыдущего слоя и передает возбуждение следующему слою в зависимости от подкрепляющего сигнала;
- подкрепление нейрона реализуется в рамках определенной задачи при условии, что достигается требуемая цель (в нашей модели, это свойство различимости объектов разных классов). Нейрон не может передать возбуждение, если необходимое условие для его выходного сигнала не выполняется (нейрон получает запрещающий сигнал).

Выводы

В работе предлагается нейроподобная логико-комбинаторная обучаемая сеть, на основе которой решаются основные задачи символьного машинного обучения, цель которых выделение из данных логических правил и закономерностей. Описывается функционирование этой сети и даются примеры её использования для вывода из данных максимально избыточных и безизбыточных классификационных тестов. Предлагаемая сеть реализует универсальное индуктивное правило порождения подмножеств мощности $q + 1$ некоторого основного множества из подмножеств на единицу меньшей мощности. Показывается, что эта реализация на сети обладает рядом свойств, позволяющих значительно снизить вычислительную сложность индуктивного правила по сравнению с его реализацией на основе Apriori. Узел нейроподобной логико-комбинаторной обучаемой сети можно рассматривать как модель нейрона, реализующего при передаче возбуждения логическую функцию, управляемую с помощью присоединенных процедур.

Благодарности

Работа выполнена при финансовой поддержке РФФИ (проект № 18-07-1800098А). This research work was supported by the Academic Excellence Project 5-100 proposed by Peter the Great St. Petersburg Polytechnic University.

Библиографический список

9.1. *Agraval R.*, *Sricant R.* Fast algorithm for mining association rules in large databases // Proceedings of the 20th International Conference on Very Large Databases (VLDB'94). — San-Francisco, 1994. — P. 487–499.

9.2. *Anohin P. K.* System analysis of an integrative activity of a neuron. — Moscow: Science, 1974. — 157 p. — (In Russian).

9.3. *Anshu C.*, *Raghuvanshi C.* An algorithm for frequent pattern mining based on Apriori // International Journal on Computer Science and Engineering. — 2010. — Vol. 2, no. 4. — P. 942–947.

9.4. *Baskakova L.*, *Zhuravlev Y.* A model of recognition algorithms with representative samples and systems of supporting sets. // U.S.S.R. Comput. Math. Math. Phys. — 1981. — Vol. 21, no. 5. — P. 189–199.

9.5. *Birkhoff G.* Lattice Theory. — Third. — Providence, R.I.: Amer. Math. Soc., 1967. — 423 p.

9.6. *Buzmakov A.*, *Kuznetsov S. O.*, *Napoli A.* A new approach to classification by means of jumping emerging patterns // CEUR Workshop Proceedings. Vol. 939. — 2012. — P. 15–22.

9.7. *Christopher R.*, *Raj S.* Modified versions apriori algorithm: A survey // Intern. J. of Emerging Trends in Engineering and Development. — 2015. — Vol. 3, issue 5. — P. 385–388.

9.8. Constructing Iceberg Lattices from Frequent Closures Using Generators / L. Szathmary [et al.] // Proceedings of International Conference on Discovery Science. — Springer, 2008. — P. 136–147.

9.9. Data Mining Using Apriori Algorithm / A. Palekar [et al.] // Intern. Journal of Eng. Trends and Technology. — 2015. — Vol. 28, no. 4. — P. 190–192.

9.10. Fast discovery of association rules / R. Agrawal [et al.] // Advances in knowledge discovery and data mining. — 1996. — Vol. 12, no. 1. — P. 307– 328.

9.11. *Ganter B.*, *Wille R.* Formal concept analysis: mathematical foundations. — Springer, Berlin, 1999. — P. 284.

9.12. *Ganter B.*, *Kuznetsov S. O.* Pattern Structures and Their Projections // Conceptual Structures: Broadening the Base. Vol. 2120 / ed. by H. Delugach, G. Stumme. — Springer, 2001. — P. 129–142. — (Ser.: LNCS).

9.13. *Ganter B.*, *Kuznetsov S. O.* Formalizing Hypotheses with Concepts // Conceptual Structures: Logical, Linguistic, and Computational Issues: Proceedings of the 8th International Conference on Conceptual Structures. — 2000. — P. 342–356.

9.14. *Geerts F.*, *Goethals B.*, *Van den Bussche J.* A tight upper bound on the number of candidate patterns // Proceedings 2001 IEEE International Conference on Data Mining. — IEEE. 2001. — P. 155–162.

9.15. *Kulkarni M.*, *Kulkarni S.* Knowledge Discovery in Text Mining using Association Rule Extraction // International Journal of Computer Applications. — 2016. — Vol. 143, no. 12. — P. 30–35.

9.16. *Kuznetsov S. O.* Mathematical aspects of concept analysis // Journal of Mathematical Sciences. — 1996. — Vol. 80, no. 2. — P. 1654–1698.

9.17. *Liquiere M.*, *Sallantin J.* Structural machine learning with Galois lattice and Graphs. // Proceedings of the 5th International Conference on Machine Learning. Vol. 98. — 1998. — P. 305–313.

9.18. *Luksch P.*, *Wille R.* A Mathematical Model for Conceptual Knowledge Systems // Proceedings of the 14th Annual Conference of the Gesellschaft fur Klassifikation (GfKl 1990) / ed. by H.-H. Bock, P. Ihm. — 1991. — P. 156–162.

9.19. *Mahmood S.*, *Shahbaz M.*, *Guergachi A.* Negative and positive association rules mining from text using frequent and infrequent itemsets // The Scientific World Journal. — 2014. — Vol. 2014. — DOI http://dx.doi.org/10.1155/2014/973750.

9.20. Maintenance of discovered association rules in large databases: An incremental updating technique / D. W. Cheung [et al.] // Proceedings of the twelfth international conference on data engineering (ICDE'96). — IEEE. 1996. — P. 106–114.

9.21. *Megretskaya I.* Construction of natural classification tests for knowledge base generation // The Problem of the Expert System Application in the National Economy: Reports of the Republican Workshop. Kishinev. — 1988. — P. 89–93.

9.22. *Mitchell T. M.* Version Spaces: A Candidate Elimination Approach to Rule Learning // Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1. — Cambridge, USA: Morgan Kaufmann Publishers Inc., 1977. — P. 305–310. — (Ser.: IJCAI'77).

9.23. *Naidenova X.* A neural network-like combinatorial data structure // CEUR Workshop Proceedings. — 2012. — Vol. 871. — P. 67–77.

9.24. *Naidenova X.* Machine Learning as a diagnostic task // Knowledge-Dialog-Solution, Materials of the Short-Term Scientific Seminar / ed. by I. Arefiev. — St Petersburg, Russia: State North-West Technical University Press, 1992. — P. 26–36. — (in Russian).

9.25. *Naidenova X.* The Data-Knowlege Transformation // Text Processing and Cognitive Technologies. Vol. 3 / ed. by V. Solovyev. — Pushchino, 1999. — P. 130–151.

9.26. *Naidenova X.*, *Chebocksarova T.* A strategy of two-direction search in solving of the applied tasks // Semiotics and informatics – Informatsionnye protsessy i sistemy. — 1979. — No. 12. — P. 125–128. — (in Russian).

9.27. *Naidenova X.*, *Polegaeva J.* SISIF–the System of knowledge acquisition from experimental facts // Industrial Applications of Artificial Intelligence. — 1991. — P. 87–92.

9.28. *Naidenova X.* Constructing Galois Lattices as a Commonsense Reasoning Process // Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems / ed. by X. Naidenova, D. I. Ignatov. — IGI Global, 2013. — P. 34–70. — DOI 10.4018/978-1-4666-1900-5.ch003. — URL: http://www.igi-global.com/chapter/constructing-galois-lattices-commonsense-reasoning/69404/.

9.29. *Naidenova X.*, *Parkhomenko V.* An Approach to Incremental Learning Good Classification Tests // Contributions to the 11th International Conference on Formal Concept Analysis / ed. by P. Cellier, F. Distel, B. Ganter. — Technische Universitat Dresden, 2013. — P. 51–64.

9.30. *Nguifo E. M.*, *Njiwoua P.* IGLUE: A lattice-based constructive induction system // Intelligent data analysis. — 2001. — Vol. 5, no. 1. — P. 73–91.

9.31. *Novelli N.*, *Cicchetti R.* FUN: An efficient algorithm for mining functional and embedded dependencies // International Conference on Database Theory. — Springer. 2001. — P. 189–203.

9.32. *Ore O.* Galois connections // Trans. Amer. Math. Soc. — 1944. — Vol. 55. — P. 494–513.

9.33. *Polyakov G.* About the principles of neural brain organization. — Moscow State Univercity, 1965. — 293 p.

9.34. *Reshetnikov M.*, *Kulagin B.* Investigation of general level development of cognitive psychological processes. — VMedA, 1987. — 27 p. — (in Russian).

9.35. *Smith D. T.* A Formal Concept Analysis Approach to Data Mining: The QuICL Algorithm for Fast Iceberg Lattice Construction // Computer and Information Science. — 2014. — Vol. 7, no. 1. — P. 10–32.

9.36. *Sobchick L.* Standardized multivariate method of personality investigation: a practical guide. — Rech, 2007. — 224 p. — (in Russian).

9.37. *Stumme G.* Efficient Data Mining Based on Formal Concept Analysis // DEXA. Vol. 2453 / ed. by A. Hameurlain, R. Cicchetti, R. Traunmüller. — Springer, 2002. — P. 534–546. — (Ser.: LNCS).

9.38. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies / Y. Huhtala [et al.] // Comput. J. — 1999. — Vol. 42, no. 2. — P. 100–111.

9.39. *Vityaev E.* Knowledge acquisition from data. Cognition via computers. Models of cognitive processes. — Novosibirsk State University, 2006. — 293 p. — (Ser.: Essays on physiological functional systems).

9.40. *Wille R.* Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts // Ordered Sets: Proceedings of the NATO Advanced Study Institute held at Banff, Canada, August 28 to September 12, 1981. Vol. 83 / ed. by I. Rival. — Springer Netherlands, 1982. — P. 445–470.

9.41. *Yan H.*, *Hamilton H.*, *Butz C.* Fdmine: discovering functional dependencies in a database using equivalences // Proceedings 2002 IEEE International Conference on Data Mining. — 2002. — P. 729–732.

9.42. *Zhang X.*, *Dong G.*, *Ramamohanarao K.* Information-Based Classification by Aggregating Emerging Patterns // Intell. Data Eng. Autom. Learn. – IDEAL 2000. Data Mining, Financ. Eng. Intell. Agents. Vol. 1983 / ed. by K. Leung, L.-W. Chan, H. Meng. — Springer, 2000. — P. 48–53. — (Ser.: LNCS).