МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ПРОГРАММИРОВАНИЯ

И.Н. ВАСИЛЬЕВА
Д.Ю. ФЕДОРОВ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ

Учебное пособие

ИЗДАТЕЛЬСТВО
САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО
ЭКОНОМИЧЕСКОГО УНИВЕРСИТЕТА
2020

ББК 67.408
В19

Васильева И.Н.

В19 Интеллектуальные системы защиты информации : учебное пособие / И.Н. Васильева, Д.Ю. Федоров. – СПб. : Изд-во СПбГЭУ, 2020. – 119 с.

ISBN 978-5-7310-4986-3

Учебное пособие освещает теоретические основы и методические подходы к интеллектуализации средств защиты информации в компьютерных системах. Рассмотрены модели и методы представления знаний, основы архитектуры экспертных систем, системы нечеткого вывода, нейросетевые модели и основы интеллектуального анализа данных. Основной целью издания является формирование представлений о способах и средствах построения интеллектуальных информационных систем и их применения в задачах информационной безопасности.

Предназначено для направления подготовки бакалавров 10.03.01 «Информационная безопасность», профиль безопасность компьютерных систем (в экономике и управлении).

Vasilyeva I.N.

Intelligent Information Security Systems : a tutorial / I.N. Vasilyeva, D.Y. Fedorov. – Saint Petersburg: Publishing house of Saint Petersburg State University of Economics, 2020. – 119 p.

The manual covers the theoretical foundations and methodological approaches to the intellectualization of information security tools in computer systems. Models and methods for presenting knowledge, the basics of the architecture of expert systems, fuzzy inference systems, neural network models and the basics of data mining are considered. The main purpose of the publication is to form ideas about the ways and means of building intelligent information systems and their application to solve information security problems.

The manual is intended for bachelors of the direction of training 10.03.01 «Information security», profile «Security of computer systems (in economics and management)».

LBC 67.408

Рецензенты: д-р экон. наук, профессор **О.Л. Ким**
канд. экон. наук, доцент **С.А. Ткачев**

ISBN 978-5-7310-4986-3

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Учебное пособие разработано в соответствии с программой дисциплины «Интеллектуальные системы защиты информации» для направления подготовки бакалавров 10.03.01 «Информационная безопасность», специализация: безопасность компьютерных систем (в экономике и управлении).

Целью освоения дисциплины «Интеллектуальные системы защиты информации» является получение необходимых знаний в области построения интеллектуальных информационных систем и навыков использования принципов искусственного интеллекта в системах защиты информации.

Интеллектуализация является одной из основных современных тенденций развития информационных систем, и сфера информационной безопасности не стала исключением. Технологии интеллектуализации применяются в задачах обнаружения различного рода угроз безопасности, выявления аномалий, корреляции событий при анализе инцидентов безопасности, поддержки принятия решений при формировании оптимальной конфигурации системы защиты информации и выборе средств защиты, а также в задачах анализа рисков информационной безопасности.

Пособие освещает все темы дисциплины «Интеллектуальные системы защиты информации» и может быть использовано обучающимися в качестве дополнения к занятиям лекционного типа для освоения теоретического материала, а также служить основой для подготовки к выполнению лабораторных работ и самостоятельной работы по дисциплине.

Глава 1

ОСНОВЫ ИНТЕЛЛЕКТУАЛИЗАЦИИ ЭВМ

1.1. Основные понятия и определения искусственного интеллекта

Термин «искусственный интеллект» (Artificial Intelligence, AI) был введен в 1956 году, и в настоящее время используется для обозначения достаточно широкого направления научных и прикладных исследований.

Первоначальной идеей (и ожиданиями 1950-х — 1960-х годов) было создание компьютера, который перерабатывал бы информацию наподобие человеческого мозга. Это так называемый *искусственный интеллект общего назначения* (сильный ИИ), способный себя осознавать, самостоятельно ставить цели, заниматься творчеством, сотрудничать с человеком на равных. Это направление, как правило, ассоциируется с разумными роботами или мыслящими компьютерами, образы которых были созданы в произведениях научной фантастики.

С научной точки зрения разработка искусственного интеллекта (ИИ) общего назначения предполагает создание компьютерных моделей, имитирующих процессы решения человеком тех или иных интеллектуальных задач в целях объяснения сущности этих процессов. В настоящее время имеются определенные успехи в области нейронауки, ученые картировали мозг и стали лучше разбираться в его функциях. Однако до построения искусственного интеллекта, повторяющего архитектуру человеческого мозга, пока далеко.

Еще одна цель ИИ — автоматизация человеческой деятельности, в особенности тех ее видов, которые традиционно считались интеллектуальными. Речь идет, прежде всего, о решении плохо формализуемых задач, для которых недостаточно применения традиционных вычислений. Здесь выделяются отдельные типы задач, например, анализаторы и синтезаторы речи, автоматический перевод текстов на естественном языке, распознавание лиц и других графических образов, автоматическое управление автомобилем и др. В настоящее время, решающие подобные задачи системы имеют весьма узкие области применения. Например, программы, способные обыграть человека в шахматы, не могут отвечать на вопросы, а нейронная сеть, предназначенная для распознавания символов текста, не сможет распознать образы на картинках и т. п. Это так называемый *узкий (слабый или*

прикладной) искусственный интеллект, направленный на решение конкретных задач, и именно в этом аспекте характерно понимания термина ИИ в инженерных и технических науках. Это способность автоматизированной системы выполнять сложные, интеллектуальные, но рутинные и повторяющиеся действия вместо и вместе с человеком. Такие системы и технологии воспринимаются как новый уровень автоматизации, позволяющий на порядки повысить производительность некоторых бизнес-процессов за счет интеллектуальных помощников. Например, интеллектуальные персональные помощники, интерактивные программные боты и роботы могут выполнять большую часть работы оператора или, например, специалиста по поиску, вводу и обработке данных.

Большинство базовых технологий, применяемых в рамках этого направления, например, нейросетевые модели, разработаны еще в прошлом веке, однако рост вычислительной мощности компьютерных систем сделал возможным их широкое применение на практике. Так, лишь совсем недавно компьютеры научились распознавать образы практически на том же уровне, что и человек. Это стало возможным благодаря таким трем основным факторам, как повышение скорости вычислений за счет увеличения вычислительной мощности аппаратной части компьютера, накопление массивов данных и появление библиотек анализа данных.

Технологии *машинного обучения* (Machine Learning, ML), в особенности глубокого машинного обучения, построенные на нейросетевых моделях, позволяют обеспечить самообучение и адаптацию системы к изменяющейся среде, что позволяет в ряде случаев повысить эффективность не только самой системы, но и ее разработки. Так, например, алгоритмы классификации спама, основанные на правилах или статистике, были разработаны довольно давно. Но для того чтобы добиться высокого качества алгоритмов, разработчикам пришлось потратить много времени на их улучшение. С использованием ИИ и машинного обучения процесс определения спама стал занимать меньше времени и давать лучшие результаты, в то время как стоимость разработки подобных программ снизилась.

Еще одной возможной целью ИИ является усиление интеллекта. К этому направлению можно отнести, например, *интеллектуальный анализ данных* (ИАД), который не может быть эффективно произведен человеком ввиду слишком большого объема подлежащей анализу ин-

формации. Системы ИАД способны распознавать сложные закономерности, в том числе и скрытые, проводить классификацию и обобщение информации, делать выводы и прогнозы, осуществляя поддержку принятия решений.

Различия в подходах и трактовках терминов ИИ требует стандартизации в этой сфере. В настоящее время разрабатывается международных стандарт ISO/IEC WD 22989 Artificial intelligence — Concepts and terminology, который будет содержать терминологию, описывать концептуальные подходы и принципы построения систем с элементами ИИ, взаимосвязь ИИ с другими сквозными технологиями, а также базовые принципы и рамочные подходы к нормативно-техническому регулированию ИИ. Принятие стандарта ожидается в 2021 году.

Будем придерживаться следующего определения ИИ.

Искусственный интеллект — область информатики, которая занимается разработкой *интеллектуальных компьютерных систем*, то есть систем, обладающих возможностями, традиционно связываемых с человеческим разумом, — понимание языка, обучение, способность рассуждать, решать проблемы и т. д.

Интеллектуальные системы могут обладать различными возможностями [13], однако их основные характерные признаки:

— развитые коммуникативные способности;

— умение решать сложные, плохо формализуемые задачи;

— способность к развитию и самообучению.

Исследования в области ИИ могут быть классифицированы по задаче (или предметной области), требующей интеллектуального анализа, либо по используемому инструментарию, либо по разрабатываемой модели мышления [20]. В настоящее время наиболее актуальны следующие направления ИИ, выделяемые на основе решаемой задачи.

Обработка естественного языка (natural language processing) — технологии для анализа текста и извлечения из него информации. Эти технологии позволяют, например, анализировать тональность текста и делить отзывы на позитивные и негативные, классифицировать текст в соответствии с его смыслом, отслеживать упоминания компаний в новостях и так далее. К этому направлению также относятся:

— автоматическое реферирование и информационный поиск;

— машинный перевод;

— системы речевого общения (например, чат-боты).

Анализаторы и синтезаторы голоса — распознавание и синтез голосовой речи, голосовой биометрии, преобразование речи в текст. Речевые технологии используются в голосовых ассистентах вроде Siri и Алисы, колл-центрах для идентификации и дистанционного обслуживания клиентов и т. д.

Компьютерное зрение — извлечение информации из изображений и видеопотока. Компьютерное зрение позволяет находить, отслеживать, классифицировать и идентифицировать объекты и применяется, в частности, для распознавания лиц, в беспилотных автомобилях и умных роботах.

Биометрические анализаторы — распознавание людей по физическим или поведенческим чертам (используя образец голоса, отпечатки пальцев, фотографии лица и тела, рисунок вен, фото роговицы глаз и так далее). Биометрия в основном используется для идентификации, например, в банковском секторе.

Предиктивная аналитика и интеллектуальный анализ данных — анализ больших данных, выделение в них паттернов, определение предикторов (прогностических параметров) и построение прогнозных моделей. Предикативная (предиктивная, прогностическая) аналитика используется для оценки платежеспособности клиентов, проведения рекламных кампаний, предсказания времени отказа оборудования и решения других задач.

Развиваются и другие направления, практическое применение которых ограничивается сферой развлечений или представляет в большей мере исследовательский интерес: игровой интеллект, доказательство теорем и автоматизация научных исследований, сочинение текстов и музыки и др. Прикладные направления исследований в сфере ИИ характеризуются тем, что их значительная часть посвящена не процессам мышления, а предмету интеллектуального анализа. К примеру, для сочинения текстов изучение структуры литературных произведений имеет большее значение, чем изучение мыслительной деятельности писателя.

В сфере информационной безопасности (ИБ) технологии искусственного интеллекта и машинного обучения используются уже в течение нескольких лет. Интеллектуализация систем защиты информации является современным трендом. Машинное обучение и интеллектуальный анализ данных применяются для анализа потоков информации (например, сетевого трафика или файлов журна-

лов) для обнаружения признаков компрометации, выявления аномалий и вредоносной активности [24]. Это позволяет повысить качество мониторинга и детектирования угроз в системах предотвращения вторжений (IDS/IPS) и средствах антивирусной защиты, производить динамическую настройку защищенных периметров, обеспечить понимание сценариев атак и реализовать автоматическую машинную реакцию на атаки в сетевых инфраструктурах.

Системы распознавания образов нашли свое применение в инструментах компьютерной криминалистики, системы биометрического анализа — в системах аутентификации, а анализаторы текстов — при распознавать ключевых слов или тем в системах предотвращения утечек (DLP).

Можно перечислить следующие основные направления в ИИ, выделяемые по развиваемому в них инструментарию.

Искусственные нейронные сети — ориентированные на обработку сенсорной информации модели нервной системы, демонстрирующие, в частности, высокие способности к распознаванию образов.

Нейронные сети хорошо подходят для использования в системах обнаружения вторжений и проникновения. Есть предложения по их использованию для обнаружения DoS-атак, вредоносного ПО, спама, зомби-сетей, классификации вредоносных программ и расследования инцидентов ИБ. Причиной популярности нейронных сетей в области ИБ является их высокая скорость при аппаратной реализации на ПЛИС или графических процессорах (CPU), обеспечивающих параллельные вычисления [6]. Искусственные нейронные сети применимы в задачах распознавания образов, классификации, принятия решений для выбора ответов на атаки и т. д.

Эволюционные вычисления — модели, использующие понятие естественного отбора, обеспечивающего отсеивание наименее оптимальных согласно заданному критерию решений (генетические алгоритмы, муравьиный алгоритм).

Генетические алгоритмы применяются для обучения нейронных сетей без учителя (самообучения), а также для оптимизации топологии нейронной сети, то есть числа нейронов и межнейронных связей. Генетические алгоритмы использованы в некоторых существующих системах обнаружения вторжений, существуют работы по их применению для динамического поиска уязвимостей и активного аудита компьютерной сети, а также выбора набора защитных мер в задачах

построения системы защиты информации (ЗИ). Существуют работы, показывающие возможность применения генетических алгоритмов при решении задач криптоанализа, однако они представляют скорее научный, чем практический интерес.

Экспертные системы — системы, ориентированные на использование формальных моделей представления знаний и логический вывод.

Экспертные системы позволяют автоматизировать процесс принятия решений на основе знаний, полученных от экспертов-специалистов. Основной проблемой экспертных систем является приобретение знаний, что требует использования квалифицированного экспертного мнения. В сфере ИБ экспертные системы используются, в основном, для выбора мер защиты информации и оптимизации использования ограниченных ресурсов, оценки уровня защищенности и оценки эффективности системы защиты информации. Кроме того, экспертная система может реализовывать логику (в том числе и достаточно сложную) принятия решений средством ЗИ (например, для выбора мер реагирования в системе предотвращения вторжений).

Эвристическое программирование — системы, оптимизирующие поиск в пространстве решений за счет использования эвристик, опирающихся на имеющийся опыт.

Методы оптимизации поиска, предлагаемые ИИ, широко используются при решении различных прикладных задач, в том числе и в сфере ЗИ, однако такие системы редко позиционируются как интеллектуальные. Наибольшую известность получили эвристические методы детектирования вредоносного ПО в системах антивирусной защиты.

Мультиагентные системы — системы, использующие для решения сложной задачи или проблемы множество взаимодействующих агентов. Интеллектуальный агент — независимый объект, который наблюдает состояние управляемых им процессов через систему с помощью датчиков и осуществляет воздействие для достижения целей системы. Интеллектуальные агенты также могут использовать обучение или оперировать знаниями для достижения поставленных целей. Наиболее естественным является использование мультиагентных систем в распределенных средах. В настоящее время схематично описаны модели взаимодействующих агентов для распределенного обнаружения вторжений и DDOS-атак. При реализации мультиагентных систем ЗИ встает вопрос об обеспечении качества и защиты каналов связи между агентами, что может потребовать сотрудничества с провайдерами.

Классификация методов ИИ по разрабатываемой модели мышления включает:

— поиск в пространстве решений;

— представление знаний;

— машинное обучение.

Каждое из этих направлений сосредотачивает внимание на одном из аспектов интеллекта, отражающих логику развития ИИ в целом [20]. Исследования в области ИИ начались с парадигмы «мышление как поиск» и с разработки методов решения формально поставленных задач. Методы первой группы рассматривали поиск как ответ на ситуацию, для которой нет готового решения, и применялись в интеллектуальных играх (таких, как шахматы) и автоматическом доказательстве теорем. При этом предполагалось, что суть интеллекта состоит в решении проблем, а сам процесс решения может быть представлен как поиск пути от исходных данных к ответу в пространстве возможных решений (или как поиск пути от имеющихся средств к конечной цели через достижимые подцели). Однако подобный подход имеет серьезные ограничения, заключающиеся в том, что формализованное описание задачи должно составляться человеком. Кроме того, не удалось найти универсальных алгоритмов эффективного решения задачи поиска.

Позднее акцент сместился на проблему представления знаний, что привело к развитию экспертных систем. Экспертные системы оказались способными строить формальные описания задач, используя знания об узкой предметной области. Для систем, основанных на знаниях, поиск решения сводился к манипулированию знаниями, в чем и усматривалась теперь суть мышления. Здесь уже человек задавал описание не каждой конкретной задачи, а некоторой (хоть и достаточно узкой) предметной области, включающей целый комплекс задач. Однако с развитием систем, основанных на знаниях, встала проблема автоматического приобретения знаний (машинного обучения).

В настоящее время машинное обучение понимается как центральное направление в области ИИ. Интеллект перестал пониматься как некий готовый продукт, который можно воспроизвести, или как фиксированная способность к решению задач или манипулированию знаниями. Здесь уже машинная система получает возможность,

по крайней мере частично, строить описание предметной области самостоятельно в рамках заданного человеком представления. При метаобучении (т. е. применении методов обучения к самому процессу обучения) ставится вопрос об автоматическом построении представлений, детали которых могут существенно зависеть от предметной области.

Постановка и решение любой задачи связаны с определенной предметной областью. *Предметная область* — описание части реального мира, которое в силу своей приближенности рассматривается как ее информационная модель. Предметную область можно понимать как область человеческих знаний, в терминах которой формулируются задачи, и в рамках которой они решаются.

Тогда **знания** — это совокупность сведений о сущностях (объектах, предметах) реального мира, их свойствах и отношениях между ними в определенной предметной области. Иными словами, знания — это выявленные закономерности предметной области (принципы, связи, законы), позволяющие решать задачи в этой области.

Знания можно трактовать как один из верхних уровней иерархии способов представления информации (рис. 1).

На нижнем уровне этой иерархии находится *шум*, состоящий из информационных элементов, которые не представляют интереса и могут лишь затруднить восприятие и представление данных. На более высоком уровне находятся *необработанные данные* (raw data), которые обрабатываются информационными системами (ИС) безотносительно их содержания. На следующем уровне находится *информация*, т. е. обработанные данные, несущие определенный смысл и явно представляющие интерес для пользователей. Способ извлечения этих смыслов или использования информации и представляет *знания*. Определенные знания могут относиться и к тому, как нужно преобразовывать данные

Рис. 1. Пирамида знаний

в информацию. Например, строка 1706BD9DCCD374D0301BCE8F0 3A37349 при отсутствии знаний о ней может показаться бессмысленной, то есть проявлением шума. Однако, если есть основания полагать (или достоверно известно), что эта последовательность имеет смысл, например, является значением хэш-функции в шестнадцатеричной записи, то она рассматривается как данные. После этого к полученной информации можно применить знания. Пусть, например, имеется правило, что если значение хэш-функции файла совпадает с одним из известных значений, то файл можно считать вирусным. Тогда применив это правило к рассматриваемому значению, можно сделать заключение (принять решение) о вредоносности или необходимости дальнейшей проверки файла, к которому данное значение относится.

Над уровнем знаний находится уровень *метазнаний* — то есть знаний о знаниях. *Онтология* представляет собой метазнания, которые описывают все, что известно о рассматриваемой предметной области.

Наконец, вершиной всех знаний является *мудрость* — метазнания, позволяющие определять наилучшие цели и находить пути их достижения.

Любая *интеллектуальная компьютерная система* включает базу знаний, использующую некоторый способ представления знаний, и механизм вывода. **Механизм вывода** — обобщенная процедура поиска решения задач, реализующая некоторую общую стратегию нахождения решения (например, путем логического вывода).

Как правило, в процессе поиска решения задач применяются *логические рассуждения* и *опыт*. Общим термином, которым обозначается использование опыта для решения некоторой задачи, является *эвристика*. **Эвристика** (от древнегреческого ευρίσκω — «отыскиваю», «открываю») — совокупность приемов, методов и правил, облегчающих и упрощающих решение познавательных, конструктивных и практических задач. Эвристический опыт принято называть **рассуждениями на основе прецедентов**, с помощью таких рассуждений делается попытка найти решение задачи по данным о встречавшихся ранее аналогичных случаях, называемых прецедентами. Эвристический метод («метод проб и ошибок») основан на применении правдоподобных рассуждений и призван помочь в выборе стратегии действий в ситуации, когда исходных данных недостаточно для выработки единственного правильного решения. Это, в свою очередь, не гарантирует и абсолютной правильности полученного решения, хотя в большинстве случаев оно может оказаться достаточно хорошим

или хотя бы допустимым. Эвристики часто используются при программировании игр, имитации творческих процессов и т. п.

В отличие от эвристического вывода, логический метод рассуждений основан на применении логики. Логика может быть неформальной (используется в обычной жизни) или формальной. В *формальной логике*, называемой также символической логикой, исключительную важность имеет то, как осуществляется логический вывод и как учитываются другие факторы, которые обеспечивают доказательство истинности или ложности полученного допустимым способом заключения. В логике термин *доказательство* обозначает формальный способ, в котором применяются факты и правила логического вывода для обоснования действительного заключения. **Логический вывод** — это формальный термин, используемый для обозначения рассуждений специального типа, которые опираются не на семантику, а на правила логики. Цель логического вывода состоит в достижении действительного заключения на базе фактов с использованием доказательства в допустимой форме. **Логическое рассуждение** — это формирование действительных логических выводов. Различают следующие виды логических рассуждений.

Дедукция — рассуждение от сложного к простому, то есть получение частного правила на основе общего правила.

Индукция — рассуждение от простого к сложному, то есть синтез общего правила на основе частных примеров.

Аналогия — рассуждение на основе прошлого опыта, т. е. скрытая (неявная) закономерность, которая присуще многим на первый взгляд внешне различающимся объектам.

Процедуры приобретения знаний могут использовать до определенной степени неопределенные и противоречивые данные, поэтому и порождаемые в результате знания не обладают полной достоверностью. Как следствие, возникает проблема представления нечетких знаний, а также проведения рассуждений в условиях неопределенности. Кроме того, сами рассуждения могут носить приближенный, вероятностный характер, что приводит к использованию формализма вероятностных (индуктивных) и нечетких логик.

В настоящее время к средствам интеллектуализации относят не только механизмы моделирования собственно процесса рассуждений, но также и таких проявлений интеллекта как способность к обучению, обобщению, накоплению опыта (знаний и навыков), адаптации к из-

меняющимся условиям в процессе решения задач. Для моделирования адаптационных механизмов используются, как правило, модели на основе нейронных сетей. Следует отметить, что искусственные нейронные сети не формируют логические выводы, а осуществляют поиск в целях обнаружения в данных основополагающих образов, которые могут и не являться очевидными для человека. По существу, искусственная нейронная сеть представляет собой классификатор образов.

Несмотря активное применение методов ИИ в различных предметных областях, теория явно не определяет, что именно считать необходимыми и достаточными условиями достижения интеллектуальности ИС. Сверхзадачей ИИ в сфере ИБ является построение интеллектуальных систем защиты информации (ИСЗИ), которые могли бы решать неформализованные задачи так же эффективно, как и человек, или превосходить его.

1.2. Основные модели представления знаний

Для того, чтобы работать со знаниями в компьютерной интеллектуальной системе, их необходимо представить определенными структурами данных, соответствующими выбранной среде разработки. Универсальным средством описания (представления) знаний является естественный язык. Однако использование естественного языка в компьютерных системах наталкивается на ряд препятствий, главным из которых является отсутствие формальной семантики естественного языка. Поэтому имеющиеся знания о предметной области сначала надо структурировать и формализовать, то есть описать средствами некоторой модели представления знаний (МПЗ).

Представление знаний — это процесс (способ) описания знаний человека о проблемной области посредством выражений на формальном языке, называемом *языком представления знаний*. Цепочка представления знаний для их хранения и использования в интеллектуальной компьютерной системе представлена на рисунке 2.

Выбранная МПЗ определяет способ и язык задания (представления) знаний, а также доступный для использования способ манипулирования ими для решения задач, то есть механизм вывода. Наиболее известными МПЗ являются:

— формальные логики и нечеткие множества;

— системы продукций (системы, построенные на правилах);

— семантические сети;
— фреймы и сети фреймов;
— онтологии;
— объектно-ориентированное представление;
— нейронные сети;
— сети доверия Байеса и др.

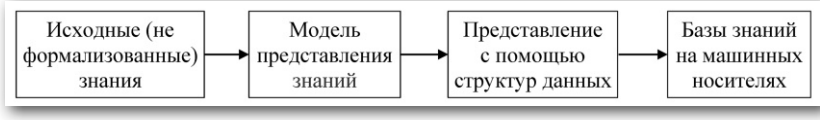

Рис. 2. Этапы представления знаний в интеллектуальной системе

Разные МПЗ используют ряд схожих понятий, характеризующих структуру знаний [12].

Индивид (экземпляр, объект) — существующий в единственном экземпляре представитель своего класса, которому можно присвоить уникальный индекс.

Концепт — смысловая единица, содержание понятия, может быть задан эксплицитно (перечислением множества конкретных индивидов) или имплицитно (содержательно) определен с помощью описания через другие уже известные концепты.

Свойство (атрибут, признак, роль, слот) — один из атрибутов концепта, характеризующих его с какой-либо точки зрения.

Отношение — множество пар, троек или кортежей n-й размерности, задающее свойства концептов и индивидов, проявляемые ими при взаимодействии с другими концептами или индивидами.

Структура (сложное отношение, формула, терм, фрейм) — система (множество) отношений, определенных на конечном множестве концептов или индивидов, может предполагать существование у ее элементов каких-либо свойств.

В основе **логических моделей представления знаний** лежит понятие *формальной системы* (*теории*). Примерами формальных систем могут служить исчисление высказываний, исчисление предикатов, формальная продукционная система. Такие системы позволяют использовать мощные методы *логического вывода — метод резолюций* и *обратный метод*. Дополнение этих методов рядом *эвристических стратегий* позволяет существенно повысить эффективность вывода.

Эти методы являются системами *дедуктивного типа*, т. е. вывод получается из заданной системы посылок с помощью фиксированной системы правил вывода. В основе дедуктивного вывода лежит *аксиоматический подход*, подразумевающий выделение некоторого небольшого множества априорно истинных утверждений (фактов, аксиом), опираясь на которые, можно вывести все другие истинные утверждения. Аксиомы представляют собой фундаментальные определения, на основании которых создаются сами формальные системы, такие как математика и логика. Используя лишь одни аксиомы, можно создавать теоремы. *Теорема* — это утверждение, которое может быть доказано путем демонстрации того, что оно следует из аксиом.

Дальнейшим развитием предикатных систем являются системы *индуктивного типа*, в которых правила вывода порождаются системой на основе обработки конечного числа обучающих примеров.

Формальная система рассматривает только синтаксис утверждений, а не их смысл (семантику), что позволяет отделить знания от рассуждений.

Под **формальной системой** понимается четверка (кортеж) $\mathbf{M} = < \mathbf{T}, \mathbf{P}, \mathbf{A}, \mathbf{F} >$, где **T** — алфавит (множество базовых элементов); **P** — множество правил построения синтаксически правильных выражений; **A** — априорно истинные выражения (аксиомы); **F** — правила вывода новых истинных выражений в рамках формальной системы **M**.

В *исчислении высказываний* (пропозициональной логике) базовым элементом является *высказывание* — логическая переменная, принимающая истинностные значения (истина либо ложь). **Высказывание** (утверждение) — предложение, истинностное значение которого может быть определено. Например, утверждение «Квадрат имеет четыре равные стороны» истинно, а «Солнце вращается вокруг Земли» — ложно. На основе логически неделимых высказываний (простых фактов) с помощью логических операций (конъюнкции $\wedge$, дизъюнкции $\vee$, отрицания $\neg$ и импликации $\rightarrow$) строятся сложные высказывания, или пропозициональные *формулы*.

Любая логическая операция может быть описана таблицей истинности, которая определяет истинностное значение результата операции в зависимости от истинностных значений ее операндов. Аналогичным образом с помощью таблицы истинности может быть описана и любая логическая формула. Формулы, которые истинны вне зависимости от значений входящих в них переменных, называются *тавтологиями*.

Пример тавтологии: $A \vee \neg A$. Такие формулы выражают логические законы и могут описывать схемы логического мышления. С другой стороны, *противоречие* — это составное высказывание, которое всегда является ложным. Пример противоречия: $A \wedge \neg A$.

Пропозициональная логика имеет существенные ограничения. В частности, пропозициональная логика оперирует только полными высказываниями, с ее помощью нельзя исследовать внутреннюю структуру высказывания. В целях обеспечения возможности анализировать более общие случаи была разработана *логика предикатов*. В своей простейшей форме она принимает вид *логики предикатов первого порядка*.

Предикат первого порядка — это n-местное отношение между объектами — аргументами предиката, принимающее истинностные значения (истина либо ложь). Количество аргументов ($n = 1, 2, 3, ...$) называется арностью, или местностью предиката.

Из переменных, обозначающих аргументы предиката, а также из функций формируются *термы*. *Формулы* в исчислении предикатов формируются на основе логических операций и предикатов от термов.

Например, если f_1 и f_2 — двуместные функции, то запись $f_1(f_2(m_1,m_2),m_3)$ — терм. А запись $A(f_1(m_1,m_2),m_3)) \rightarrow B(f_2(m_1,m_2),m_3))$, где A, B — двуместные предикаты, — формула.

Кроме того, в исчислении предикатов используются кванторы существования $\exists$ («существует») и всеобщности $\forall$ (интерпретируется как «для любого», «для каждого» или «для всех»). Например, если triangle(x) — предикат, обозначающий, что x является треугольником, а polygon(x) — многоугольником, то запись $\forall x(\text{triangle}(x) \rightarrow \text{polygon}(x))$ может быть прочитана как «любой треугольник является многоугольником» («для любого x верно, что если x является треугольником, то x является многоугольником»).

Основным методом логического вывода в исчислении предикатов первого порядка является *метод резолюций*. Метод резолюций является обобщением метода «доказательства от противного». Для его эффективного применения формулы представляются в специальном виде — в виде дизъюнктов Хорна (хорновских дизъюнктов), а точнее — их конъюнкции.

Будем считать предикат атомарной или элементарной формулой. Литералом будем называть атомарную формулу или отрицание атомарной формулы. Атом называется положительным литералом, а его

отрицание — отрицательным. *Дизъюнкт* — это дизъюнкция конечного числа литералов. Если дизъюнкт не содержит литералов, его называют пустым (□).

Хорновский дизъюнкт — дизъюнкт с не более чем одним положительным литералом.

Будем считать, что все литералы P и L_i — положительны. Тогда хорновский дизъюнкт может принять один из следующих видов:

— L_i (атомарная формула, факт);

— $\neg L_1 \vee \neg L_2 \vee \neg ... \vee \neg L_n$ (цель, запрос — отрицание утверждения $L_1 \wedge L_2 \wedge ... \wedge L_n$, истинность которого требуется доказать);

— $P \vee \neg L_1 \vee \neg L_2 \vee \neg ... \vee \neg L_m$ (определенный дизъюнкт, правило, эквивалентен формуле $L_1 \wedge L_2 \wedge ... \wedge L_m \rightarrow P$).

Идея **метода резолюций** заключается в том, чтобы к имеющемуся набору фактов и правил добавить отрицание утверждения, истинность которого нужно доказать, и попытаться получить противоречие — вывести из получившегося расширенного набора фактов тождественно ложное утверждение (□).

Правило резолюции базируется на тавтологии $(A \vee C) \wedge (B \vee \neg C) \Rightarrow A \vee B$ и может быть сформулировано следующим образом: если могут быть найдены два факта вида $A \vee C$ и $B \vee \neg C$, то нужно добавить резольвенту — правило $A \vee B$, которое далее можно использовать для построения новых резолюций. Для исчисления высказываний резолютивный вывод сводится, таким образом, к последовательности вычеркиваний контрарных литералов. Предикаты же могут содержать переменные, поэтому предварительно требуется провести унификацию дизъюнктов — замену переменных термами, такую, что после подстановки отрезаемые литералы стали бы контрарными.

Таким образом, из множества, состоящего из посылок и отрицания цели, последовательно выбираются дизъюнкты, содержащие унифицируемые контрарные литералы, вычисляются их резольвенты, которые добавляются к исследуемому множеству дизъюнктов. Этот процесс следует продолжать до тех пор, пока не будет выведен пустой дизъюнкт. Это случится за конечное число шагов, если множество дизъюнктов невыполнимо (то есть истинность целевого утверждения можно доказать на основе существующих посылок). В противном случае процесс либо никогда не завершится, либо на каком-то шаге больше не найдется дизъюнктов, к которым можно применить правило резолюции.

Резолютивный вывод реализован в декларативном языке Prolog (Пролог), в котором конечное непустое множество хорновских дизъюнктов (фактов и правил) представляет логическую программу. Таким образом, программирование на языке Пролог заключается в составлении базы знаний — то есть описании фактов предметной области и правил, позволяющих получать новые знания на основе уже известных. Определение фактов заключается в описании свойств и отношений между конкретными объектами (индивидами), а правила задаются формулами вида: список посылок → следствие (для записи используется синтаксис: следствие:-список посылок). Очень часто правила задают рекурсивное соотношение.

Рассмотрим в качестве примера описание иерархической структуры дерева каталогов для файловых систем Windows (или Linux). Можно определить отношение `содержит(x,y)`, которое определяет, что объект *y* расположен непосредственно в каталоге *x* (*y* является прямым потомком *x*). Тогда фактами базы знаний будет набор пар объектов файловой системы «родитель»-«потомок» для конкретной реализации ИС, например:

```
содержит('C:','Windows').
содержит('C:','Пользователи').
содержит('Windows','System32').
содержит('Windows','Resourses').
содержит('Windows','bootstat.dat').
содержит('System32','AppLocker').
содержит('System32','AutExt.dll').
содержит('System32','AppxSip.dll').
    ...
содержит('C:','Program Files').
содержит('Windows','INF').
содержит('Windows','Logs').
содержит('Windows','explorer.exe').
содержит('Windows','PFRO.log').
содержит('System32','drivers').
содержит('System32','auditpol.exe').
содержит('drivers','cdfs.sys').
```

Здесь неявно предполагается, что имена папок и файлов уникальны в пределах ИС, хотя в реальных системах это не так. Чтобы обойти это ограничение, можно, например, рассматривать в качестве объектов не имя файла/папки, а номер соответствующего индексного узла, а для получения символического имени дополнительно ввести предикат `имя(x,y)`, задающий для объекта *x* символическое имя *y*.

Затем можно ввести отношение `путь(x,y)`, получающее значение истина, если объект *y* является вложенным в *x* (то есть существует относительный путь от *x* до *y*). Определить отношение путь можно с помощью рекурсивного правила:

```
путь(x,y) :- содержит(x,z), содержит(z,y).
```

Введя свойство корень(*x*) для выделения корневых объектов файловой системы (например, логических томов C:, D: для Windows или / для Linux) и задав соответствующие факты, например:

```
корень('C:'). корень('D:').
```

можно проверять полные пути с помощью правила:

```
полный_путь(x,y) :- путь(x,y), корень(x).
```

Запросы к этой базе знаний могут иметь вид:

путь('Windows', 'cdfs.sys'). — имеется ли в папке Windows и ее подпапках файл cdfs.sys?

содержит('System32',*x*). — какие объекты содержатся непосредственно в каталоге System32?

`путь('System32',x).` — какие объекты имеются в дереве каталога System32?

`путь(x,'cdfs.sys').` — какие объекты составляют полный путь для файла cdfs.sys?

`полный_путь(x,'cdfs.sys').` — на каком логическом томе находится файл cdfs.sys?

`полный_путь(x,y).` — какие объекты имеются в системе? Будет осуществлен поиск всех вариантов — Пролог последовательно будет пытаться согласовывать вопрос с имеющимися в программе предложениями от первого до последнего.

`путь(x,_).` — показать все непустые папки (чтобы не выводить вложенные объекты, используется анонимная переменная).

Построение полного пути к файлу как последовательности объектов файловой системы в Пролог может быть осуществлено с помощью списков, являющихся основной структурой данных языка.

Логика предикатов и метод резолюций применимы во многих случаях, язык Пролог эффективно применяется для разработки систем автоматического перевода и создание естественно-языковых интерфейсов, реализации символьных вычислений и автоматического доказательства теорем, разработки экспертных систем и оболочек экспертных систем.

Вместе с тем классические логические МПЗ имеют ряд существенных ограничений. Так, механизм логического вывода не предусматривает возможности использования неполной информации об объектах

предметной области. Как правило, используется предположение о ложности неизвестных фактов, что не во всех случаях отвечает реальности.

С помощью классической логики невозможно выразить такие категории как «большинство» (то есть «больше половины») — для этого в логике должны быть предусмотрены предикаты, обеспечивающие подсчет количества элементов. Классическая логика не позволяет выражать зависимости, которые могут быть истинными только иногда, но не всегда.

Такие ограничения снимаются в рамках многозначной или нечеткой логики, где утверждения помимо классических истинностных значений (истина или ложь) могут принимать другие значения (например, «не определено» или степень уверенности). Это, однако, влечет дополнительное усложнение логической системы, а механизмы вывода хуже разработаны.

Продукционные системы (наборы правил) похожи на логические системы представления знаний, однако имеют не такие строгие ограничения. Продукционная МПЗ основана на *правилах* вида: «Если (условие), то (действие)», где условие — это образец, по которому осуществляется поиск в множестве фактов и знаний о текущей ситуации (входной информации), а действие — действия или операторы, которые будут выполняться при успешном исходе поиска. При этом форма задания как условий, так и действий может существенно различаться в зависимости от реализации продукционной системы. Действия правил могут заключаться в модификации набора фактов в базе знаний (например, в добавлении новых фактов) либо во взаимодействии с внешней средой.

Кроме наборов правил продукционные системы обычно включают и механизмы вывода (манипулирования знаниями). Программа, управляющая перебором правил, называется *машиной вывода*. Может использоваться *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения к данным) вывод. Под данными понимаются исходные факты, на основании которых запускается машина вывода.

В качестве примера рассмотрим правила для выбора схемы резервного копирования (выбрать тип = {«полное», «инкрементное», «дифференциальное»}) в зависимости от объема резервируемой информации (данные = {«много», «мало»}, требуемой частоты создания архивов (период = {«несколько раз в день», «ежедневно», «раз в неделю»})

и критичности скоростных характеристик процедур создания архива либо восстановления данных (приоритет = {«запись», «восстановление»}).

```
ЕСЛИ данные = «мало» И приоритет = «восстановление» ТО выбрать тип = «полное».
ЕСЛИ период = «раз в неделю» И приоритет = «восстановление» ТО выбрать тип = «полное».
ЕСЛИ данные = «много» И приоритет = «восстановление» ТО выбрать тип = «дифференциальное».
ЕСЛИ данные = «много» И приоритет = «запись» ТО выбрать тип = «инкрементное».
ЕСЛИ данные = «мало» И приоритет = «запись» ТО выбрать тип = «дифференциальное».
ЕСЛИ период = «несколько раз в день» И приоритет = «запись» ТО выбрать тип = «инкрементное».
```

Если добавить к этому набору правила для определения приоритета, например, на основе критичности уровня доступности информационной инфраструктуры (требования доступности = {«низкая», «средняя», «высокая»}) и пропускной способности канала передачи данных (канал = {«много», «мало»}), то можно делать уже двухшаговый вывод.

```
ЕСЛИ требования доступности = «низкая» ТО задать приоритет = «запись».
ЕСЛИ требования доступности = «средняя» И канал = «мало» ТО задать приоритет = «запись».
ЕСЛИ (требования доступности = «средняя» ИЛИ требования доступности = «высокая») И канал = «много» ТО задать приоритет = «восстановление».
ЕСЛИ требования доступности = «высокая» И канал = «мало» И данные «много» ТО задать приоритет = «запись».
ЕСЛИ требования доступности = «высокая» И канал = «мало» И данные «мало» ТО задать приоритет = «восстановление».
```

Как видно, правила в первом наборе (для определения типа резервного копирования) не являются взаимоисключающими. Например, в ситуации, когда резервированию подлежит небольшой объем информации, при этом процедура архивации информации проводится часто, а скорость создания архива является критичным параметром, может быть применено как последнее, так и предпоследнее правило набора. Похожая ситуация складывается при редком резервировании больших объемов информации и критичности скорости восстановления данных из архива — в этом случае применимы второе и третье правило.

В продукционных системах не обязательно на каждом шаге применяется только одно правило. Существование на каждом шаге вывода нескольких правил, которые могут быть применены в текущих

условиях, является типичным для продукционных систем. Устранение неоднозначности выбора правила (*разрешение конфликтов*) производится с помощью *эвристик* (стратегий разрешения конфликтов) или *метаправил*. Могут использоваться эвристики специфичности (применение в первую очередь наиболее частного правила), эвристики, заключающиеся в применении еще не применявшегося в текущем выводе правила или самого нового правила (если база правил является открытой, то есть может пополняться в процессе эксплуатации системы). *Метаправила* являются частью базы знаний и описывают правила разрешения конфликтов между обычными продукциями, определяемые спецификой предметной области.

Продукционная модель чаще всего применяется в промышленных экспертных системах (ЭС). Наиболее типичным является использование продукционных систем в задачах диагностики и классификации, особенно в медицинских ЭС. В очень примитивной форме диагностирующие продукционные системы распространены в виде компьютерных помощников для помощи пользователям в решении проблем или в настройке программного обеспечения (ПО).

Наиболее известной средой для разработки продукционных систем является система с открытым исходным кодом CLIPS (C Language Integrated Production System), обладающая высокой скоростью и эффективностью, а также допускающая интеграцию с C- и Java-приложениями. В настоящее время предпочтение отдается нечетким продукционным системам (к посылкам и заключениям добавляются значения уровня доверия), которые могут быть реализованы средствами MatLab Fuzzy Logic Toolbox, Wolfram Mathematica Fuzzy Logic, FuzzyTECH или другими средствами нечеткого моделирования.

Логическое следствие (операция импликации) выражает лишь связь между значениями истинности своих операндов, однако она не может установить причинно-следственную связь или связь типа класс-экземпляр. В то же время, при использовании связки «если..., то...» в естественном языке, подразумевают именно такие отношения. Человек имеет склонность к ассоциированию, то есть определению смысла некоторых понятий через его ассоциативные связи с другими понятиями, которые в совокупности образуют своего рода сеть. Понятия являются основой нашего знания о мире, поэтому такую ассоциативную сеть можно рассматривать в качестве представления знаний.

Однако простая ассоциативная связь недостаточно выразительна, так как необходимо различать не только сами понятия, но и разнообразные типы отношений между ними. Как результат, возникла МПЗ, названная *семантическими сетями*.

Семантическая сеть представляет собой (ориентированный) граф, узлы которого соответствуют некоторым сущностям (объектам, событиям, процессам, явлениям, характеристикам или значениям), а дуги — отношениям между ними. При этом, как узлы, так и дуги имеют метки, указывающие, какое именно понятие помещено в узле, или какое именно отношение обозначает дуга.

В качестве примера семантической сети можно привести схему связи понятий ИБ (рис. 3), приведенную в стандарте ГОСТ Р ИСО/МЭК 15408–1-2012. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель.

Семантические сети первоначально использовались для моделирования смысла (семантики) высказываний на естественном языке, тем не менее они являются МПЗ широкого назначения. В зависимости от назначения семантической сети (ситуационная, целевая, функциональная, классификационная) могут использоваться те или иные типовые связи. Основными типовыми связями являются: «это» (IS-A, «экземпляр», A-Kind-Of, «является») или «например» (частный

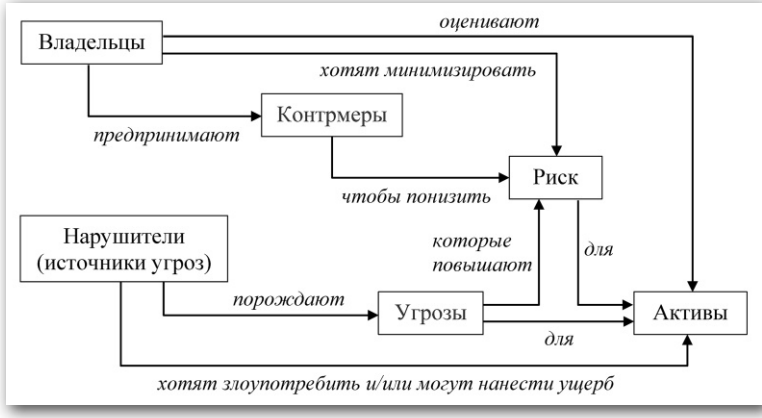

Рис. 3. Связь высокоуровневых понятий ИБ

случай) — связь вида элемент-множество, частное-общее, «состоит из» (HAS-A, «имеет частью») или «является частью» (PART-OF) — связь вида часть-целое, функциональные связи: «свойство» (связь вида объект-свойство), «значение» (связь вида свойство-значение), причинно-следственные связи — «является следствием». Характеристики узла повторяются (наследуются) в его потомках (от общего к частному), что позволяет исключать дублирование информации в семантических сетях.

Семантические сети могут использоваться как для описания общих знаний о некоторой предметной области, так и для описания структуры единичных высказываний.

Так, например, семантическая структура знания о событии «29.10.2019 в 10.17 администратор остановил службу Active Directory Domain Services на сервере DC01.univer.ru с целью применения обновлений операционной системы» представляется с помощью лингвистического падежного отношения (падежного фрейма) — рис. 4. Для описания падежных фреймов используется собственный набор типизированных отношений (агент, объект, источник, приемник, время, место, цель и др.). Такая структура более однозначна, чем предложение на естественном языке. Падежные фреймы отражают глубинные (ролевые) взаимосвязи между элементами некоторой ситуации.

Особенность семантической сети как МПЗ состоит в неотделимости механизма вывода от самой базы знаний. При формировании запроса

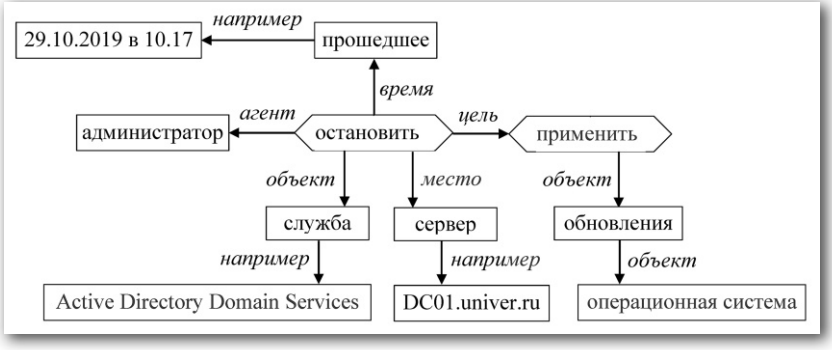

Рис. 4. Пример семантической сети, описывающей событие

к базе знаний сначала строится семантическая сеть, отражающая структуру запроса (например, рис. 5), а затем производится сопоставление общей сети и сети для запроса. Это обуславливает необходимость перебора и низкую эффективность вывода в семантических сетях, особенно если ответ на запрос является отрицательным.

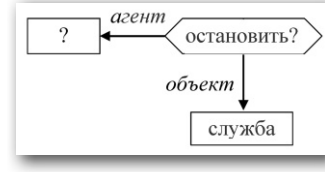

Рис. 5. Подсеть запроса «Кто остановил службу?»

Семантические сети обеспечивают высокую степень наглядности представления знаний, приближены к описанию на естественном языке, обеспечивают как возможность объединения различных фрагментов сети, так и выделения некоторых ее участков, охватывающих необходимые смысловые характеристики. Семантические сети могут быть легко преобразованы в программу на языке Пролог.

Семантические сети используются, как правило, для автоматического распознавания текстов и понимания их смысла, например, в системах автоматического перевода. Увеличение мощности аппаратных платформ позволяет в настоящее время использовать для решения подобных задач методы глубинного обучения на основе нейронных сетей. Применение нейросетевых технологий обработки естественного языка привело к созданию ряда успешных систем машинного перевода, например, Google Translate. Тем не менее, использование семантических сетей остается актуальным для решения ряда частных задач, как в области языкового анализа (например, для разрешения лексической многозначности), так и не имеющих непосредственного отношения к обработке естественного языка (например, разметка объектов на изображениях).

В литературе термины «семантическая сеть» и «онтология» встречаются в достаточно близких контекстах, связанных с инженерией знаний. Однако они означают два разных понятия. *Онтология* задает предмет описания, то есть формальную спецификацию концептуализации.

В информатике онтологии используются для описания знаний о некоторой предметной области. Онтология описывает понятия предметной области, а также отношения, которые имеются между этими понятиями. Онтологии рассматриваются в качестве ключевого элемента в проекте семантического веба.

Онтология как модель выражает определенный взгляд (разработчика) на некоторую предметную область и формально может быть описана следующим набором множеств:

$$\mathbf{O} = \langle \mathbf{X}, \mathbf{R}, \mathbf{Ф} \rangle,$$

где **X** — конечное множество концептов (понятий) предметной области, которую представляет онтология **O**. Например, для предметной области ИБ такими понятиями являются: угроза, риск, опасность, безопасность, атака, инцидент и т. д.

R — конечное множество отношений между концептами (понятиями, терминами) заданной предметной области. Например, угроза_риск, риск_опасность, атака_инцидент и т. д. Здесь имя отношения формируется из имен, связываемых этим отношением сущностей и знака нижнего подчеркивания.

Ф — конечное множество функций интерпретации, заданных на концептах и/или отношениях онтологии. Роль функции интерпретации может играть словесное пояснение термина (аннотация), формула для вычисления значения термина, алгоритмическое описание, а также определение в виде логической формулы.

Пример ядра онтологии компьютерных угроз, включающего основные понятия, приведен на рисунке 6. В основе этой онтологии лежат следующие принципы:

— на систему осуществляется «Атака», которая создает «Угрозу безопасности», оказывающую влияние на «Состояние защищенности системы»;

— «Атака» использует «Уязвимость системы» и направлена на ее «Актив»;

Рис. 6. Ядро онтологии компьютерных угроз

— «Инцидент безопасности» подразумевает последствия, т. е. «Реакцию системы» на несанкционированные действия извне;
— использование какой-либо «Уязвимости» может открыть другую «Уязвимость».

Из всего множества отношений в онтологии выделяется специальный класс — простая *таксономия*. Под таксономической структурой понимается иерархическая система понятий, связанных между собой отношением «это» (IS-A, A Kind Of — «быть элементом класса» или «быть подклассом класса»). Это отношение позволяет организовать иерархию понятий онтологии в виде дерева.

Потребность в разработке онтологий объясняется следующими причинами:

— совместное использование людьми или программными агентами общего понимания предметной области;
— разработка и управление терминологией;
— возможность повторного использования знаний в предметной области;
— получение логической теории, которая состоит из словаря и набора утверждений на некотором языке логики, что позволяет на основе этой теории получать вывод новых знаний, явно не заложенных в онтологии.

Онтологии могут быть разработаны с использованием языков Gellish, OWL (Web Ontology Language), а также стандарта RDF (Resource Description Framework) — для семантического web, и сред Ontolingua, Protégé, Ontostudio и др. Универсальным средством описания формальной семантики является язык XML (eXtensible Markup Language).

Рассмотренные МПЗ плохо приспособлены для описания сложно структурированных объектов или явлений — множества логических формул или продукционных правил никак не структурированы. Семантические сети, хотя и могут выражать связи класс-объект и объект-экземпляр, формально рассматривают узлы на одном уровне абстракции, то есть не способны выражать иерархию в явном виде, поскольку расположение узлов никак не упорядочено. Представлению объекта как единой целостности уделяется основное внимание во **фреймовой МПЗ**.

Под **фреймом** понимается абстрактный образ или ситуация. Например, термин «угроза» понимается как «потенциальное воздействие, которое может, прямо или косвенно, нанести ущерб безопасности».

Из этого определения убрать ничего нельзя, например, убрав слово «потенциальное», получим определение атаки, а не угрозы, если уберем «ущерб» будет просто действие и т. п. Но в этом определении есть «слоты» или «щели» — незаполненные значения некоторых атрибутов, например, тип и объект воздействия, тип и размер ущерба и др. Такой образ, как и его формализованная модель, называется *фреймом*.

Первоначально фреймы предназначались для представления стереотипных знаний. Фрейм аналогичен структуре записи в высокоуровневом языке программирования (таком как C или Pascal), при этом полям записи соответствуют такие компоненты фрейма, как *слоты*, а значениям — *заполнители* слотов. Фрейм представляет собой структуру вида:

```
Имя фрейма
Имя первого слота: значение первого слота
...
Имя n-го слота: значение n-го слота
```

В качестве значений слотов могут выступать числа или математические соотношения, тексты на естественном языке или процедуры, правила вывода или ссылки на другие слоты данного фрейма или на другие фреймы (связь A-Kind-Of, «является»). Например:

```
Уязвимость
Наименование: CVE-2012-2559
Характер: Позволяет удаленным нарушителям выполнить произвольный код или вызвать отказ в обслуживании (неправильная запись указателя) через специально созданный пакет к TCP-порту 5678
Вероятность использования: средняя
```

Значением слота может быть и другой фрейм (то есть набор слотов) более низкого уровня. Таким образом, могут создаваться объекты со сложной иерархической структурой. Например:

```
Актив
Наименование: Граничный маршрутизатор ST-200
Вид актива: аппаратное обеспечение
Уровень критичности: высокий
Права доступа (фрейм Пользователя_1)
Права доступа (фрейм Пользователя_2)
...
```

Фрейм с незаполненными слотами называется фреймом-*образцом*, или *прототипом*. Прототип, как правило, представляет собой не просто перечень слотов, но также и некоторые значения, заданные для них по умолчанию. Могут указываться и ограничения на значение

слота. Некоторые слоты могут не иметь значений по умолчанию. Прототипы хранятся в базе знаний и описывают некоторое стереотипное представление об объекте или ситуации.

Когда наблюдаются конкретный объект или ситуация, для их отображения на основе поступающих данных создается фрейм-*экземпляр*. Сначала значения слотов установлены по умолчанию, но по мере накопления данных эти значения уточняются. Таким образом обеспечивается специализация фреймов применительно к конкретным ситуациям и создание более конкретных фреймов.

Системы фреймов проектируются таким образом, чтобы более универсальные фреймы находились ближе к вершине иерархии. Важнейшим свойством фреймов является наследование свойств по A-Kind-Of (AKO) связям. Слот AKO указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся значения аналогичных слотов, причем наследование свойств может быть частичным.

С каждым слотом может быть связана одна или несколько процедур, осуществляющих внутреннюю интерпретацию данных фрейма и описывающих их использование. Как правило, используются процедуры типов: если-добавлено (if-added) — выполняется при помещении новой информации в слот; если-удалено (if-removal) — выполняется при удалении информации из слота; если-нужно (if-needed) — выполняется, когда запрашивается информация из слота, а он пуст, либо значение, установленное по умолчанию, неприменимо. Кроме того, процедурная информация помогает использовать произвольные типы отношений (пространственные, временные, причинно-следственные) между фреймами. Таким образом, фреймы могут входить между собой в различные отношения, образуя структуру, подобную семантической сети.

Пример фреймовой модели для решения задач оценки рисков ИБ [1] приведен на рисунке 7.

Для работы с фреймами разработан ряд языков специального назначения, таких как FRL, SRL, KRL, KEE, HP-RL; кроме того, средства работы с фреймами (LOOPS и FLAVORS) включены в язык LISP. LISP (LISt Processing language, Лисп) — «язык обработки списков» — использует представление программ и данных в виде линейных списков и является традиционным инструментальным средством ИИ.

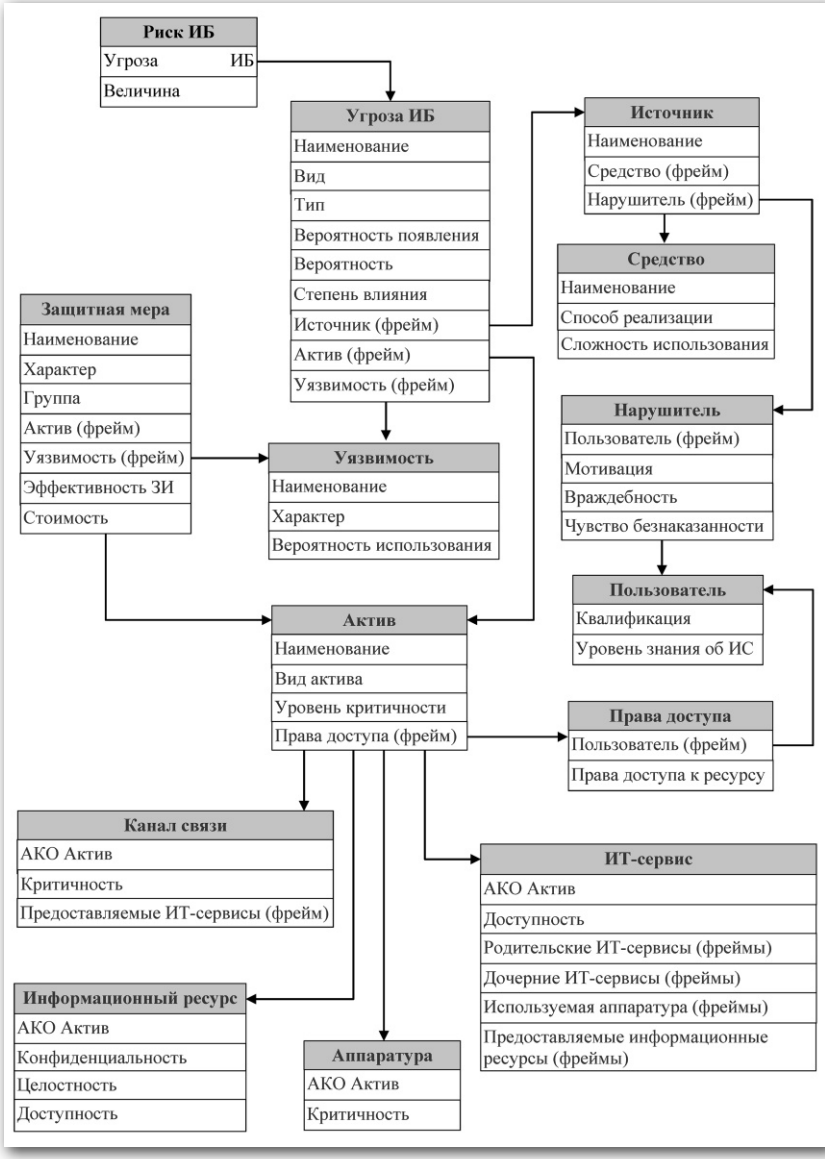

Рис. 7. Фреймовая модель для оценки рисков ИБ

Достаточно близким к фреймовому является **объектно-ориентированный подход** (ООП) к представлению знаний, в основе которого, как и во фреймовой МПЗ, лежит понятие *объекта*, как целостного образования, обладающего внутренней структурой и наделенного определенными свойствами и поведением. Идеи данного подхода проявляются в принципах объектно-ориентированного программирования (инкапсуляция, наследование, полиморфизм), которое, однако, не может быть отождествлено с данным подходом в целом. В отличие от фреймов, в ООП класс не описывает стереотипное представление об объекте или ситуации (в отличие от фрейма-прототипа), а лишь задает каркас для создания объектов.

Многообразие подходов к представлению знаний обусловлено потребностью конкретных предметных областей, специфика которых определяет не только конкретику наполнения базы знаний, но и ее структуру, то есть тип используемого представления, а также методы манипулирования знаниями.

В традиционных МПЗ описание базы знаний создается путем формализации знаний, полученных от специалистов в узкой предметной области — экспертов, что делает их разработку и актуализацию достаточно трудоемкой. Неэффективность такого подхода становится особенно заметной при разработке знаний о динамично меняющихся предметных областях, например, в задачах обнаружения вторжений, где постоянно появляются новые уязвимости, технологии эксплуатации и схемы проведения атак. Необходимость автоматизации процесса приобретения знаний и извлечения знаний, в том числе и неявных, из больших массивов данных приводит к смещению акцента на методы машинного обучения, в том числе, обучения без учителя (самообучения), а также интеллектуального анализа данных. Это, в свою очередь, требует рассмотрения вопросов представления нечетких, неполных и недостоверных знаний и моделирования рассуждений в условиях неопределенности.

1.3. Экспертные системы

Экспертная система (ЭС) — это ИИС, работающая на основе логического вывода. ЭС использует экспертные знания для обеспечения высокоэффективного решения неформализованных задач в узкой проблемной области. Поэтому ЭС называют также *системами,*

основанными на знаниях. При помощи ЭС, как правило, решаются следующие задачи:

— интерпретация данных — определение смысла данных, результаты которого должны быть согласованными и корректными (например, определение конкретных объектов или их типов по данным описания);

— диагностика — обнаружение неисправностей, отклонений или ошибок в некоторой системе (например, диагностика ошибок в аппаратуре и математическом обеспечении компьютера);

— мониторинг — непрерывная интерпретация данных в режиме реального времени и сигнализация о выходе значений тех или иных параметров за допустимые пределы (например, контроль аварийных датчиков, системы обнаружения вторжений или SIEM-системы);

— проектирование — подготовка спецификаций на создание объектов (например, конфигураций компьютерных систем) с заранее определенными свойствами;

— планирование — нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции (например, планирование эксперимента);

— обучение — диагностирование ошибки при изучении какой-либо дисциплины и объяснение правильного решения (системы-тренажеры).

Основу ЭС составляет база знаний о предметной области, которая накапливается в процессе построения и эксплуатации ЭС. Накопление и организация знаний — важнейшее свойство всех ЭС. Типовая ЭС (рис. 8) включает следующие компоненты:

— база знаний;

— подсистема сбора знаний и интерфейс общения с экспертом;

— интерфейс общения с пользователем;

— решатель — машина логического вывода;

— подсистема объяснений.

База знаний состоит из фактов и набора правил, по которым в зависимости от входной информации принимается то или иное решение. Факты представляют собой краткосрочную информацию, которая может изменяться в процессе решения задачи. Правила представляют более долговременную информацию о том, как порождать новые факты и гипотезы из имеющихся данных. Как правило, в базе знаний ЭС

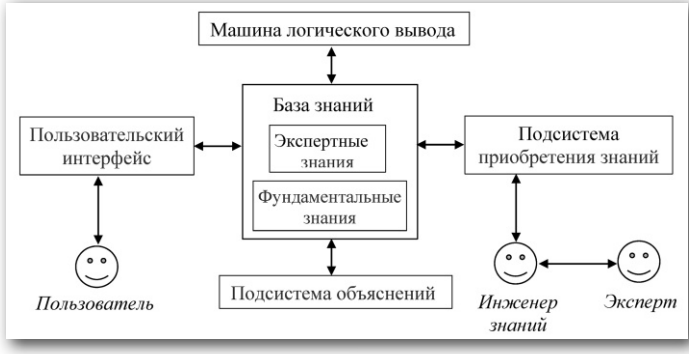

Рис. 8. Обобщенная схема экспертной системы

интегрируются знания, поступающие от экспертов — специалистов в конкретной предметной области (экспертные знания), а также фундаментальные (энциклопедические) знания, составляющие суть общеизвестных научных теорий и моделей.

Система приобретения знаний — механизм формализации знаний эксперта, пополнения базы знаний и обучения (самообучения) ЭС. Эксперт формулирует правила, по которым работает система, как правило, он взаимодействует с системой приобретения знания не напрямую, а через инженера знаний, который аккумулирует экспертные знания и решает, каким образом они будут представлены в ЭС. Другой подход — извлечение знаний непосредственно из массивов данных с помощью механизмов машинного обучения.

Машина логического вывода — программный модуль, моделирующий ход рассуждений эксперта на основании знаний, содержащихся в базе знаний.

Пользователь ставит цели для логического вывода и предоставляет ЭС информацию о конкретном объекте или ситуации, как правило, взаимодействуя с ЭС через вопросно-ответный интерфейс. Диалог с пользователем обычно ведется на языке, приближенном к естественному, программные инструменты интеллектуального интерфейса воспринимают сообщения пользователя и преобразуют их в форму внутреннего представления базы знаний, а также осуществляют обратное преобразование при получении от машины результатов обработки запросов пользователя.

Система объяснений предназначена для показа пользователю всего процесса рассуждений, в результате которого было найдено (или не найдено) решение.

Описанная структура характерна для *статических* ЭС, которые могут использоваться в том случае, когда можно не учитывать изменения окружающего мира за время решения задачи. Однако существует целый ряд задач, в которых требуется учитывать фактор времени и (или) изменять в процессе решения данные об окружающем мире. Такие задачи могут быть решены в рамках *динамических* ЭС [16]. Динамическая ЭС дополнительно включает подсистему моделирования внешнего мира и подсистему сопряжения с внешним миром, позволяющую получать данные о состоянии внешней среды через систему контроллеров и датчиков (рис. 9).

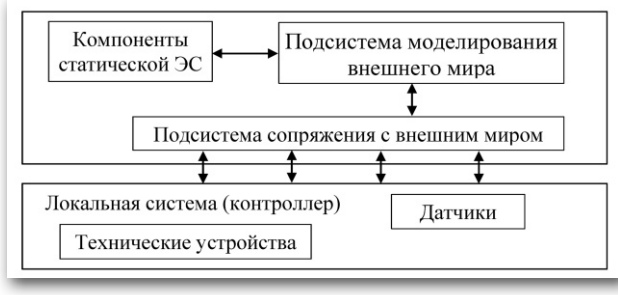

Рис. 9. Архитектура динамической ЭС

Кроме того, сама база знаний и механизмы вывода существенно изменяются, чтобы отразить временную логику происходящих в реальном мире событий.

В работе ЭС можно выделить два основных режима: режим приобретения знаний и режим решения задачи (режим консультации или режим использования). В режиме *приобретения знаний* общение с ЭС осуществляет эксперт (при помощи инженера по знаниям). В режиме *консультаций* общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения.

Глава 2

МОДЕЛИ И МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В СИСТЕМАХ ЗАЩИТЫ ИНФОРМАЦИИ

2.1. Нечеткие знания

Необходимость введения понятия нечеткости

Рассмотренные в предыдущей главе модели представления знаний, базирующиеся на классическом логическом выводе, имеют существенные ограничения, что затрудняет их эффективное использование для решения практических задач. Во-первых, в классической логике, любое утверждение должно принимать истинностное значение, то есть быть либо истинным, либо ложным. Однако в реальных ситуациях истинностное значение утверждения может быть неизвестным, либо определяться с некоторым *коэффициентом уверенности* (*коэффициент доверия*), отражающим степень уверенности в истинностном значении высказывания, то есть степень его неопределенности — число, выражающее вероятность или степень уверенности, с которой можно считать данный факт или правило достоверным или справедливым. Вместе с тем, люди, обладая здравым смыслом, могут делать рациональные заключения и в условиях неопределенности.

Неточные рассуждения с применением коэффициентов доверия впервые были реализованы в медицинской экспертной системе MYCIN [7]. В ней каждое продукционное правило снабжалось значением коэффициента достоверности CF гипотезы (заключения правила) при наличии определенных свидетельств (посылок правила). При этом коэффициент достоверности может быть рассчитан как $CF = MB\text{-}MD$[1], где MB — мера повышения степени доверия к гипотезе в силу наличия свидетельств, а MD — мера повышения степени недоверия к гипотезе в силу наличия тех же свидетельств. Значения MB и MD находятся в промежутке [0,1]. Для истинных гипотез $MB = 1$, $MD = 0$, $CF = 1$; для ложных гипотез $MB = 0$, $MD = 1$, $CF = -1$; если же свидетельства отсутствуют, то все эти величины равны нулю. Отрицательные значения CF означают, что имеющиеся свидетельства опровергают гипотезу (или говорят в пользу отрицания гипотезы). При этом $CF(H) \neq 1 - CF(\neg H)$,

[1] Позднее коэффициент достоверности стали рассчитывать как $CF = \dfrac{MB - MD}{1 - \min(MB, MD)}$

как это следовало бы ожидать в теории вероятностей, но $CF(H) = -CF(\neg H)$, то есть если свидетельство подтверждает гипотезу с некоторым коэффициентом достоверности, то с тем же значением оно опровергает отрицание гипотезы.

Истинность утверждений, содержащихся в предпосылках правила, может также иметь нечеткий характер. Таким образом, в общем виде правила системы MYCIN имеют вид: <Если условие 1 выполнено с достоверностью x_1 и ... и условие m выполнено с достоверностью x_m, то справедливо заключение 1 с достоверностью y_1 и ... и заключение n с достоверностью y_n>. Для того, чтобы отбросить малозначимые гипотезы в системе MYCIN активация правил происходила только в том случае, если значение CF превысило некоторое пороговое значение (0,2). Коэффициенты достоверности правил, имеющих одно и то же следствие, вычисляются с помощью комбинирующей функции.

Несмотря на успешное практическое применение коэффициентов достоверности в экспертной системе MYCIN, существуют сложности с их теоретическим обоснованием. Позднее было показано, что теория коэффициентов достоверности фактически является аппроксимацией к классической теории вероятностей. Такой подход может применяться только в случае коротких цепей логического вывода и простых гипотез.

Еще одним ограничением классической логики и классических продукционных систем является так называемое *предположение о замкнутости мира* (базы знаний), в рамках которого полагается, что база содержит все знания о предметной области. Это означает, что любое предположение, статус которого неизвестен, полагают ложным, что позволяет сделать логическую систему знаний полной и непротиворечивой, а вывод в ней — монотонным. Тогда может быть определена истинность любого утверждения, причем ранее выведенные утверждения остаются истинными вне зависимости от того, какие утверждения будут выведены в дальнейшем. С другой стороны, это предполагает относительную неизменность базы знаний, что значительно снижает возможность адаптации интеллектуальной системы к изменяющимся условиям и затрудняет ее применение в динамических, быстро меняющихся предметных областях.

Для преодоления указанных недостатков требуется ввести учет неопределенности, которая может выражаться в следующем [15]:

— неопределенность границ, например, при разграничении таких состояний как «маленький — большой», «низкий — высокий», «старый — новый», «медленный — быстрый» и т. п.;
— неопределенность семантики отдельных понятий и терминов, вызванной присущей естественным языкам неоднозначности или многозначности смысла (полисемии), когда одно и то же слово может обозначать разные объекты / понятия в зависимости от контекста, например, карта сети (как топология) или сетевая карта (как устройство);
— неполнота модельных представлений (знаний) о системе, особенно при решении плохо формализуемых задач;
— противоречивость отдельных знаний, модельных представлений о системе или требований, например, требование решить задачу за минимальное время и с минимальными затратами вычислительных ресурсов содержит определенное противоречие (поскольку временная и емкостная сложность решения задачи обычно находятся в обратной зависимости);
— неопределенность, вызванная действием факторов случайного характера (стохастическая неопределенность).

Случайное событие может произойти или не произойти. Если существует возможность каким-то образом задать вероятностную меру (вероятностное распределение), то появляется возможность получения усредненной количественной оценки возможности наступления такого хорошо описанного события. При этом предсказать каждый конкретный исход не представляется возможным. Например, если говорят, что вероятность сбоя технической системы составляет 7%, это не значит, что в данный конкретный момент времени она на 93% работоспособна и 7% ее компонентов отказали. Смысл этого высказывания заключается в том, что в среднем 93% времени система будет работоспособна, и лишь в 7% случаях будет обнаружен отказ.

С другой стороны, на основании этого высказывания невозможно утверждать, что сбой произойдет (или не произойдет) в какой-то конкретный момент времени, например, завтра. Кроме того, неопределенными остаются и ответы на ряд вопросов. Например, если сбой системы все же произойдет завтра, можем ли мы утверждать, что работоспособность системы будет восстановлена через 7% времени суток? Будет ли система полностью выведена из строя или сбой затронет отдельный компонент системы, приведя к невозможности выполнение

или ухудшению выполнения некоторых функций при сохранении в целом ее работоспособности? Сколько компонентов системы должны отказать и к каким последствиям это должно привести, чтобы считать это событие сбоем? Как видно, при детальном анализе вероятностного высказывания, при кажущейся его простоте и очевидности обнаруживается еще один тип неопределенности, отличающейся от стохастического. Эта неопределенность относится к лингвистическому описанию события или ситуации, а не к вероятностной оценке возможности их возникновения.

Таким образом, *стохастическая неопределенность* описывает неопределенность того, произойдет ли некоторое хорошо описанное событие в будущем. С течением времени эта неопределенность может измениться или вообще исчезнуть (в случае, если событие уже произошло). Напротив, *лингвистическая неопределенность* связана с неточностью описания самого события или ситуации независимо от времени их рассмотрения.

Лингвистическая неопределенность связана как с неточностью семантики естественного языка, так и с субъективностью представлений и оценок, даваемых разными людьми. Для примера можно рассмотреть такие фразы как «красивое лицо», «трудный день», «высокооплачиваемая работа», «дорогое программное обеспечение», «хороший антивирус», «безопасная система». Очевидно, что сопоставить им какие-то точные количественные определения весьма затруднительно. В рамках *психолингвистики* принято считать, что в подобных фразах люди используют слова в качестве *субъективных категорий*, которые дают возможность классифицировать объекты, обладающие такими свойствами как высота, длина, вес, температура, цвет и др.

Рассмотрим утверждение «Скорее всего, в новом году безопасность нашей компании останется на высоком уровне». Здесь не определено как само событие (неопределенность понятий «безопасность», «высокий уровень»), так и значение вероятности не имеет количественного выражения. Выражение вероятности («скорее всего») здесь также является субъективной категорией.

Высказывания, использующие субъективные категории, играют важную роль в процессе принятия решений в повседневной жизни людей. Даже при том, что большинство субъективных категорий не может быть точно определено, люди могут достаточно эффективно использовать их для комплексных оценок и принятия решений, ос-

нованных на учете многих разнотипных факторов. Кроме того, в разговоре люди иногда сознательно используют неопределенность для придания своим высказываниям дополнительной гибкости. При этом люди могут рассуждать приблизительно или неточно, что позволяет им с накоплением опыта принимать адекватные решения даже в тех случаях, когда полное описание реальной системы требует наличия более детальных данных, чем человек в состоянии получить и интерпретировать. Компьютерные системы, использующие бинарную логику, такой способностью не обладают. Для возможности интерпретации субъективных категорий необходимо обеспечить сравнение конкретных числовых значений некоторой характеристики с заданным порогом (пороговым значением). Другим подходом является разработка специального математического аппарата, позволяющего использовать не точные количественные, а до некоторой степени неопределенные (нечеткие) значения.

Таким математическим аппаратом стала теория *нечетких множеств* и *нечеткий логический вывод*, использующий *лингвистические переменные*. Теория нечетких множеств предложена американским ученым Л.А. Заде в 1965 г. Л.А. Заде рассматривал теорию нечетких множеств как аппарат анализа и моделирования систем с участием человека. Его подход основан на том, что элементами мышления человека являются не числа, а элементы некоторых нечетких множеств или классов объектов, при этом переход от «принадлежности» к этому классу до не «непринадлежности» происходит не скачкообразно, а непрерывно. Л.А. Заде использовал термин «fuzzy set» (нечеткое множество). На русский язык термин fuzzy можно перевести как нечеткий, размытый, расплывчатый.

Нечеткие системы предназначены, прежде всего, для моделирования неопределенности, связанной с неточностью или неясностью описания границ системы или ее отдельных состояний. По существу, этот подход применим для решения таких задач, в которых неопределенность вызвана отсутствием четких критериев, позволяющих однозначно отнести элементы к тому или иному классу. Именно в этом и заключается различие между *нечеткостью* и *случайностью*.

Использование вероятностно-статистических моделей позволяет получать значения вероятностей некоторых событий на основе известных значений вероятностей других событий. Большинство современных ЭС, использующих вероятно-статистический подход,

интерпретируют вероятность как субъективную степень (меру) доверия к суждению об истинности некоторого высказывания. Такой подход реализуется с помощью Байесовских *сетей доверия* и позволяет уточнять первоначальные оценки вероятностей (как правило, задаваемые экспертным путем) при поступлении новой информации о наступлении тех или иных событий, являющихся реализациями случайных величин [3].

Базовыми же методами построения нечетких моделей являются *теория нечетких множеств* и *нечеткая логика*, которые в свою очередь, являются обобщением классической теории множеств и классической формальной логики. В настоящее время показано, что нечеткая мера включает как частный случай вероятностную меру [15], а нечеткие множества естественно рассматривать как «проекции» случайных множеств [19]. Таким образом, теория нечетких множеств тесно связана с теорией случайных множеств. Математический аппарат теории нечетких множеств достаточно громоздок и не позволяет в должной мере учитывать различные зависимости между моделируемыми с его помощью понятиями (объектами). Вместе с тем, использование математического аппарата случайных множеств предоставляет такие возможности.

Нечеткие множества

Математическая теория нечетких множеств позволяет описывать нечеткие понятия и знания, оперировать этими знаниями и делать нечеткие выводы. В основе теории нечеткости лежит понятие нечеткого множества. *Нечеткое множество* — совокупность элементов произвольной природы, относительно которых нельзя с полной определенностью утверждать, принадлежат ли они данному множеству или нет.

Более строго с математической точки зрения нечеткое множество может быть отождествлено со своей функцией принадлежности.

Пусть X — некоторое множество.

Подмножество A множества X в классическом понимании определяется как множество упорядоченных пар $A = \langle x, m_A(x) \rangle$, где $m_A(x)$ — характеристическая функция:

$$\mu_A(x) = \begin{cases} 1, & x \ A, \\ 0, & x \ A. \end{cases}$$

Нечеткое подмножество B множества X — множество пар $B = <x, \mu_B(x)>$, где $\mu_B(x)$ — характеристическая *функция принадлежности*, принимающая значения в некотором вполне упорядоченном множестве M. Как правило, рассматривается $M = [0, 1]$, $\mu_B(x): X \to [0,1]$. Далее будем рассматривать лишь этот случай. Вещественное значение $\mu_B(x) \in [0,1]$ показывает степень принадлежности элемента x нечеткому множеству B. Нечеткое множество описывает неопределенность, соответствующую элементу x — он одновременно и входит, и не входит в нечеткое множество B. За его вхождение — $\mu_B(x)$ шансов, а за не вхождение — $1-m_B(x)$ шансов. При этом значение $\mu_B(x) = 1$ означает, что элемент $x \in X$ определенно принадлежит нечеткому множеству B, а значение $\mu_B(x) = 0$ означает, что x определенно не принадлежит нечеткому множеству B.

Таким образом, нечеткие множества допускают возможность частичной принадлежности к ним, степень принадлежности объекта к нечеткому множеству определяется соответствующим значением функции принадлежности. Обычные множества (так же, как и интервальные множества) являются частным случаем нечетких (при $M = \{0,1\}$).

В теории нечетких множеств сохраняют свой смысл некоторые специальные виды классических множеств, такие как пустое множество и универсум (множество всех элементов).

Пустое нечеткое множество $\varnothing$ — нечеткое множество, функция принадлежности которого тождественно равна нулю: $\forall\, x \in X$, $\mu_{\varnothing}(x) = 0$.

Универсальное нечеткое множество X — нечеткое множество, функция принадлежности которого тождественно равна единице: $\forall\, x \in X$, $\mu_X(x) = 1$. При этом характеристическая функция обычного универсального множества также тождественно равна единице для всех элементов $x \in X$.

Носителем нечеткого множества B является обычное множество BS, состоящее только из тех элементов универсума, для которых значения функции принадлежности соответствующего нечеткого множества отличны от нуля: $BS = \{x \in X \mid \mu_X(x) > 0\}$.

Носитель нечеткого пустого множества является пустым. Носитель универсума совпадает с самим универсумом. Нечеткое множество конечно, если его носитель является конечным множеством. Конечная мощность нечеткого множества совпадает с числом элементов его носителя. Бесконечное нечеткое множество — такое,

носитель которого не является конечным множеством. Аналогичным образом можно определить счетные и несчетные нечеткие множества.

Для сокращения записи нечеткого множества обычно указывают лишь элементы его носителя и значения функции принадлежности для них, предполагая, что для всех остальных элементов значение функции принадлежности равно нулю. Нечеткие множества могут быть заданы в виде списка с явным перечислением всех элементов и соответствующих значений функции принадлежности (например, $B = \{x_1|\mu_B(x_1), x_2|\mu_B(x_2), ..., x_n|\mu_B(x_n)\}$, либо аналитически в виде математического выражения или графика кривой для функции принадлежности. Формально следует указывать также универсум, из которого выбираются элементы нечеткого множества, однако на практике универсум, как правило, не указывают, ограничивая его элементами рассматриваемой предметной области или решаемой задачи.

Два нечетких множества *эквивалентны* $A \equiv B$, когда $\forall\, x \in X$: $\mu_A(x) = \mu_B(x)$.

Нечеткое множество A *содержится в нечетком множестве* B: $A \subset B$ (или B доминирует над A — рис. 10), тогда и только тогда, когда $\forall\, x \in X$: $\mu_A(x) \leqslant \mu_B(x)$.

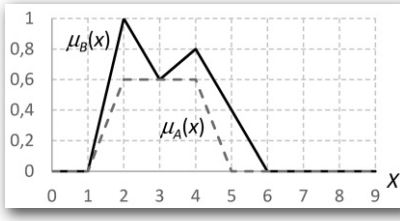

Рис. 10. Пример доминирования для нечетких множеств

Нечеткое множество является *нормальным*, если хотя бы один элемент этого множества имеет коэффициент принадлежности, равный 1. Нечеткие множества, не являющиеся нормальными, называются *субнормальными*.

Ядром нечеткого множества B (core B) называется четкое подмножество универсального множества X, элементы которого имеют степени принадлежности равные единице:

$$\text{core } B = \{x \in X \,|\, \mu_B(x) = 1\}.$$

Подмножество B множества X называется *выпуклым* нечетким множеством (рис. 11) тогда и только тогда, когда для произвольных x_1 и x_2, принадлежащих B и произвольного числа $\lambda \in [0,1]$ выполняется условие:

$$\mu_B(\lambda x_1 + (1 - \lambda)\, x_2) \geqslant \mu_B(x_1) \cap \mu_B(x_2) = \min(\mu_B(x_1), \mu_B(x_2)).$$

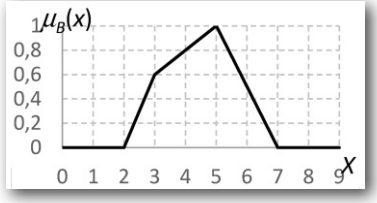

Рис. 11. Пример выпуклого нечеткого множества

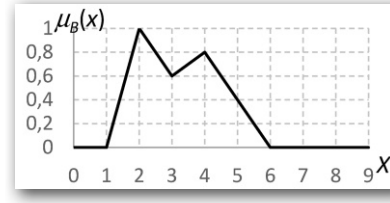

Рис. 12. Пример вогнутого нечеткого множества

Подмножество B множества X называется *вогнутым* нечетким множеством (рис. 12) тогда и только тогда, когда для произвольных x_1 и x_2, принадлежащих B и произвольного числа $\lambda \in [0,1]$ выполняется условие:

$$\mu_B(\lambda x_1 + (1-\lambda)\, x_2) \geqslant \mu_B(x_1) \cup \mu_B(x_2) = \max(\mu_B(x_1), \mu_B(x_2)).$$

Пусть A и B — два непустых нечетких подмножества X с функциями принадлежности $\mu_A(x)$ и $\mu_B(x)$ соответственно. Дадим определения теоретико-множественных операций над нечеткими множествами. Отрицанием $\overline{A}$, пересечением $A \cap B$, произведением AB, объединением $A \cup B$ и суммой $A+B$ называются нечеткие подмножества X с функциями принадлежности:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x);$$
$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x));$$
$$\mu_{AB}(x) = \mu_A(x)\mu_B(x);$$
$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x));$$
$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x),\ x \in X.$$

Графическая интерпретация логических операций для некоторого нечеткого множества A и его отрицания $\overline{A}$ приведено на рисунке 13: а — нечеткое множество A, б — отрицание $\overline{A}$, в — $A \cap \overline{A}$; г — $A \cup \overline{A}$.

Операции пересечения и объединения нечетких множеств обладают свойствами коммутативности, ассоциативности и идемпотентности. Также выполняются тождества:

$$A \cup \varnothing = A;\ A \cap \varnothing = \varnothing;$$
$$A \cup E = E;\ A \cap E = A,$$

где E — универсум.

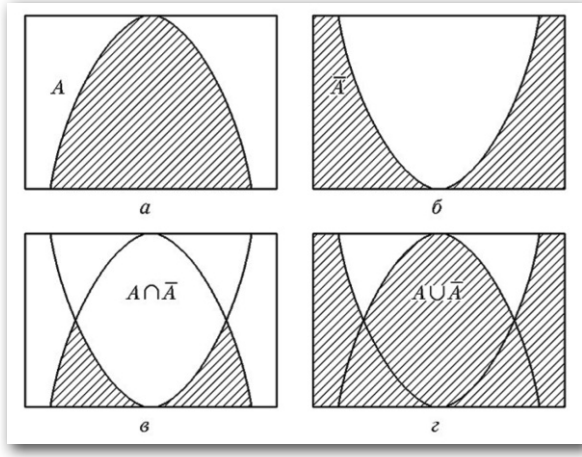

Рис. 13. Графическая интерпретация операций
над нечеткими множествами A и $\overline{A}$

Вместе с тем, в отличие от четких множеств, для нечетких множеств в общем случае $A \cup \overline{A} \neq E$; $A \cap \overline{A} \neq \varnothing$.

Для нечетких множеств справедлива теорема де Моргана в виде: $\overline{A \cup B} = \overline{A} \cap \overline{B}$; $\overline{A \cap B} = \overline{A} \cup \overline{B}$; $\overline{A + B} = \overline{A}\,\overline{B}$; $\overline{AB} = \overline{A} + \overline{B}$.

Логические операции над нечеткими множествами эквивалентны в нечеткой логике (логике Заде) логическим операциям над нечеткими или лингвистическими переменными: пересечение $\cap$ — логическому «И», объединение $\cup$ — логическому «ИЛИ», дополнение — логическому отрицанию «НЕ». Например, они позволяют определить нечеткие значения для таких субъективных категорий как «не высокая скорость», «низкий или средний уровень защищенности» (рис. 14), «хорошее и недорогое антивирусное решение» и т. п.

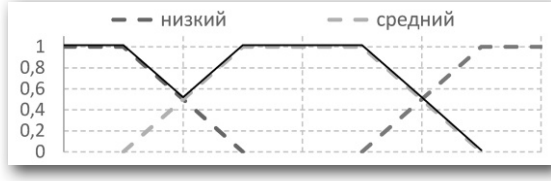

Рис. 14. Графическое определение субъективной категории
«низкий или средний уровень защищенности»

Как известно, связки «И», «ИЛИ», «НЕ» образуют полную группу, то есть с их помощью может быть выражена любая логическая формула. Следовательно, с помощью рассмотренного механизма может быть вычислена функция принадлежности заключения, когда известна строгая логическая зависимость заключения от посылок, выражающаяся логической формулой.

При вероятностном подходе свойства функции принадлежности эквивалентны свойствам вероятности случайной величины. Операции «И» (эквивалентной вероятности произведения случайных событий) соответствует операция умножения (алгебраического произведения) нечетких множеств, а операции «ИЛИ» (сумма случайных событий) — операция алгебраического сложения +. Следует отметить, что вероятность может быть определена как нечеткое значение, особенно тогда, когда ее значение оценивается приблизительно.

На основе операции алгебраического умножения для нечетких множеств могут быть определены умножение на число и возведение в степень (при положительном показателе степени). На основе последней определяются две специальные операции: концентрирование и растяжение, применяющиеся для усиления или ослабления лингвистических признаков.

— CON(A) = A2 — операция концентрирования (уплотнения), применяется для усиления значения признаков (аналогично слову «очень», например, «очень высокий», «очень старый»).

В результате применения этой операции к нечеткому множеству A уменьшаются степени принадлежности элементов к этому множеству. Причем если значение $\mu_A(x)$ близко к 1, то это уменьшение мало, а для элементов с малой степенью принадлежности — относительно велико (рис. 15). Например, если A = {1/0,2, 2/0,4, 3/0,8, 4/1, 5/0,8, 6/0,2}, то CON(A) = {1/0,04, 2/0,16, 3/0,64, 4/1, 5/0,64, 6/0,04}.

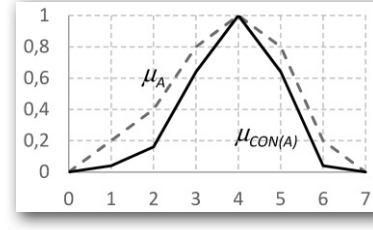

Рис. 15. Пример применения операции концентрации

Применение операции концентрации означает уменьшение нечеткости или неопределенности в задании множества, что на практике соответствует уточнению некоторых аспектов предметной области.

— DIL(A) = A0,5 — операция растяжения, обратная операции

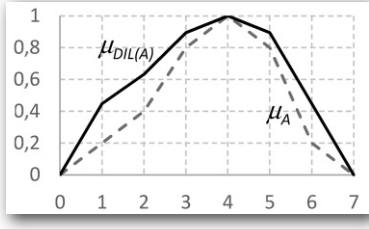

Рис. 16. Пример применения операции растяжения

концентрации, применяется для ослабления значения признаков (аналогично слову «довольно», «более-менее» — «довольно высокий», «более-менее новый»).

Применение операции растяжения означает усиление неопределенности в задании множества (рис. 16), что может интерпретироваться как потеря части информации или как поступление информации, «размывающей» ранее принятую модель ситуации.

Можно ввести и другие аналогичные по смыслу операции, позволяющие модифицировать значения признака. Так, слово «более чем» можно определить как $A^{1,25}$, «очень очень» — как $\mathrm{CON}(\mathrm{CON}(A)) = A^4$.

Кроме того, используя декартово произведение нечетких множеств (многомерные нечеткие множества), можно задать *нечеткие отношения* и операции над ними [20].

Нечетким отношением R между непустыми четкими множествами $X_1, X_2, ..., X_n$ называется нечеткое подмножество, определенное на декартовом произведении множеств $X_1 \times X_2 \times ... \times X_n$, где функция принадлежности $\mu_R(x_1,x_2,...,x_n)$ показывает степень выполнения отношения R (силу связи) между соответствующими элементами $x_1 \in X_1$, $x_2 \in X_2$, ... и $x_n \in X_n$.

С помощью нечетких отношений можно выражать степень различия или сходства ряда объектов, например, такие высказывания как «x намного меньше, чем y». Нечеткие отношения являются более гибкими по сравнению с традиционными отношениями. Они позволяют задать не только сам факт выполнения отношения, но и указывать степень его выполнения, что является крайне важным при решении ряда практических задач.

Рассмотрим, например, бинарное отношение «сходное использование» для следующего набора средств защиты информации: {антивирусное средство (А), межсетевой экран (МСЭ), система обнаружения вторжений (СОВ), средство контроля съемных носителей (СК), DLP-система (DLP), фильтрующий прокси-сервер (П)}. Элементы этого набора являются метками шкалы нечеткого множества «использование средств ЗИ». Тогда бинарное отношение задается как матрица, позволяющая сравнить сходство использования каждой пары средств из

этого набора. Очевидно, по главной диагонали этой матрицы будут стоять 1, а сама она будет симметричной (рис. 17).

	А	МСЭ	СОВ	СК	DLP	П
А	1	0,7	0,8	0,1	0	0
МСЭ	0,7	1	0,9	0	0,2	0,4
СОВ	0,8	0,9	1	0	0	0,1
СК	0,1	0	0	1	0,2	0
DLP	0	0,2	0	0,2	1	0,3
П	0	0,4	0,1	0	0,3	1

Рис. 17. Пример нечеткого отношения «сходное использование средств ЗИ»

Наиболее важными частными случаями нечетких отношений являются нечеткие отношения порядка, подобия, различия, сходства/несходства.

Таким образом, нечеткие множества обладают широкими возможностями для выражения многообразия человеческих понятий в формализованном виде.

Нечеткие и лингвистические переменные

Нечеткое число задается нечетким множеством, если оно выпуклое, нормальное, кусочно-непрерывное, его ядро содержит одну точку (рис. 18).

Нечеткий интервал задается нечетким множеством, если выполняются все условия для нечеткого числа, кроме последнего (рис. 19).

Нечеткая переменная описывается произвольным нечетким множеством. Формально, **нечеткая переменная** — это тройка $<\alpha, X, A>$, где α — имя переменной, X — универсальное множество (область определения переменной α), A — нечеткое множество на X, описывающее ограничения на значения нечеткой переменной α (т. е. $\mu_A(x)$).

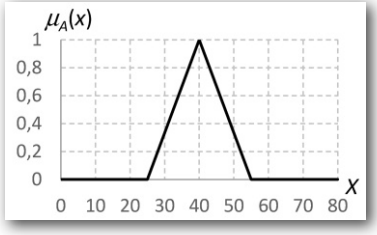

Рис. 18. Нечеткое число «около 40»

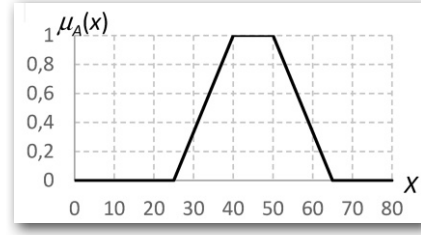

Рис. 19. Нечеткий интервал «примерно от 40 до 50»

В теории нечетких множеств, помимо значений и переменных цифрового типа, определены еще лингвистические переменные.

Лингвистической переменной (linguistic variable) называется переменная, значения которой определяются набором вербальных (словесных) характеристик некоторого свойства. Например, лингвистическая переменная «уровень защищенности» может принимать значения «низкий», «средний», «высокий». Слова или словосочетания некоторого естественного или искусственного языка, используемые для характеристики значений лингвистической переменной, называются *термами*.

Значения лингвистической переменной определяются через нечеткие множества, а именно каждое значение, которое может принимать лингвистическая переменная является нечетким множеством. Формально, лингвистическая переменная — это набор $<\beta, T, X, G, M>$, где:

β — имя лингвистической переменной,

T — множество ее значений-термов (базовое терм-множество), представляющее имена нечетких переменных, каждая из которых определена на множестве X;

G — синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения). Множество $T \cup G(T)$, где $G(T)$ — множество сгенерированных с помощью процедуры G термов, называется расширенным терм-множеством лингвистической переменной;

M — семантическая процедура, позволяющая сопоставить каждому новому значению лингвистической переменной, образуемому процедурой G, соответствующее нечеткое множество.

Например, для лингвистической переменной «уровень защищенности» можно сформировать новые значения «очень низкий» и «очень высокий» (задав их функции принадлежности с помощью операции концентрирования $\mathrm{CON}(A)$).

В общем случае значение лингвистической переменной представляет собой *составной терм*, содержащий сочетание *элементарных термов*, которые могут принадлежать одной из четырех базовых категорий:

— первичные термы — значения из базового терм-множества;

— отрицание НЕ, логические связки И, ИЛИ;

— модификаторы вида «очень», «весьма», «слабо», «более-менее», «достаточно», «вполне», «скорее» и т. п.;

— маркеры (чаще всего — вводные слова).

Логические связки и неопределенности выражаются через операции, определенные на нечетких множествах.

Для проектирования нечеткой системы необходимо все переменные описать как лингвистические, то есть определить для каждой переменной множество термов, а каждый терм описать как нечеткое множество со своей функцией принадлежности. Нечеткое множество определяется через некоторую базовую шкалу B и функцию принадлежности $\mu(x)$, $x \in B$, принимающую значения в интервале [0,1]. Формальное определение нечеткого множества не накладывает никаких ограничений на вид функции принадлежности. Однако с практической точки зрения именно вопрос построения функции принадлежности является наиболее важным.

Существует различные подходы к заданию значений функции принадлежности, которые можно подразделить на две группы: *экспертное* и *экспериментальное* задание. При использовании экспертных методов вид функции принадлежности так или иначе зависит от субъективного понимания и даже восприятия реальности человеком-экспертом. Функция принадлежности в этом случае отражает субъективную степень уверенности экспертов в том, что данное конкретное значение базовой шкалы соответствует определяемому нечеткому множеству. Вторая группа методов основывается на проведении экспериментов, чаще всего физических, и поэтому такие методы считаются более объективными и точными, но они, как правило, требуют больших затрат и потому не всегда выполнимы.

1. *Прямое экспертное* задание — эксперт (или группа экспертов) непосредственно определяет виды и параметры функции принадлежности, основываясь на особенностях предметной области. Такие методы используются для измеримых категорий, таких как скорость, время, расстояние, давление, температура и т. д.

Например, пусть лингвистическая переменная «Специалист по защите информации» может принимать значения «опытный специалист» и «неопытный специалист» в зависимости от того, сколько лет человек проработал по специальности. Пусть проведен опрос экспертов, результаты которого приведены в таблице 1.

Таблица 1

Экспертные оценки опытности специалиста

Стаж работы по специальности	1	2	3	4	5	10
Доля экспертов, ответивших «опытный»	0%	20%	30%	60%	80%	100%

Тогда нечеткое множество «опытный специалист» может быть определено как $A = \{1/0, 2/0{,}2, 3/0{,}3, 4/0{,}6, 5/0{,}8, 10/1\}$ (рис. 20).

2. *Косвенное экспертное* задание — когда проводят опрос экспертов, в процессе которого они отвечают «ДА» или «НЕТ» на заданные вопросы. Эти методы определения значений функции принадлежности обычно используют, когда нет элементарных измеримых свойств, через которые может быть определено интересующее нас нечеткое множество.

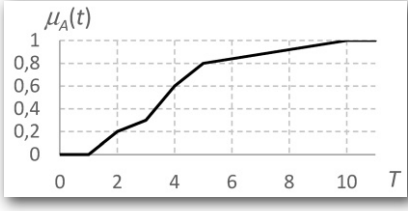

Рис. 20. Пример задания функции принадлежности для нечеткого множества «опытный специалист»

К этой группе методов, в частности, относится *метод парных сравнений*, когда эксперт формирует матрицу $A = \{a_{ij}\}$ отношений функций принадлежности элементов нечеткого множества. При этом диагональные элементы равны 1, а элементы, симметричные относительно диагонали, определяются как $a_{ij} = \frac{w_i}{w_j}$, $i \neq j$, т. е. если один элемент оценивается экспертом в a раз значимее, чем другой, то последний должен быть в $1/a$ раз сильнее, чем первый. При этом задача сводится к поиску вектора w, удовлетворяющего уравнению вида $Aw = \lambda_{max}$, где λ_{max} — наибольшее собственное значение матрицы A. Поскольку матрица A является положительной по определению, то решение данной задачи существует.

Вместе с тем, на практике экспертная оценка степени отношения для пар объектов является не менее сложной задачей, чем прямая экспертная оценка признака или ранжирование объектов по степени его выраженности. Поэтому чаще всего метод парных сравнений используют в следующем виде. Для каждой пары объектов сравнение производится с помощью отношения: «лучше» (первый

объект предпочтительнее, лучше, имеет более ярко выраженное значение оцениваемого признака, >), «хуже» (<) или «сопоставим» (объекты неразличимы, степень выраженности оцениваемого признака для обоих объектов одинакова, =). В такой постановке приняты три стандартных способа задания числовых меток для матрицы сравнений (табл. 2).

Таблица 2

Варианты задания числовых значений в матрице сравнений

лучше	2	1	1
хуже	1	0	0,5
сопоставим	0	–1	0

Примем первый из этих вариантов. Тогда сформированная экспертом матрица сравнений антивирусных средств может выглядеть следующим образом (рис. 21).

Антивирус	Kaspersky	Avira	AVG	Dr.Web
Kaspersky	1	2	2	2
Avira	0	1	1	2
AVG	1	1	1	2
Dr.Web	0	0	1	1

Рис. 21. Пример парного сравнения антивирусных средств

Формально, матрица парных сравнений должна быть симметрична относительно главной диагонали в том смысле, что если $a_{ij} = a+y$, где a — значение элементов на главной диагонали, а y — некоторое число, то для симметричного элемента должно выполняться $a_{ji} = a-y$. Однако на практике оценки эксперта могут быть непоследовательны и даже противоречивы, а несогласованность матриц парных сравнений — обычное явление. Так, на рис. 21 в строке 3 (AVG) указано, что объект AVG предпочтительнее Dr.Web, однако в строке 4 (Dr.Web) объект Dr.Web сопоставим с AVG. Как будет видно далее, незначительная несогласованность матрицы сравнений не оказывает существенного влияния на результат, однако если рассогласование достаточно велико, может потребоваться специальный анализ результатов экспертизы.

После составления матрицы парных сравнений рассчитываются значения столбцов P_K с использованием итерационной формулы $P_K = AP_{K-1}$, P_0 принимается равным единичному столбцу (все элементы которого равны 1). Интерес представляют нормированные значения столбца P_K: $P'_{iK} = \dfrac{P_{iK}}{\sum P_{iK}}$, которые рассматриваются как относительный вес (сила) объектов и могут задавать значения функции принадлежности μ_A.

Расчет может быть произведен с заданной точностью ε, то есть выполняется столько итераций K, что $\max\limits_i \left| P'_{iK} - P'_{iK-1} \right| < \varepsilon$.

Например, потребуем ε = 0,001. Тогда на первом шаге получаем:

$$P_1 = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 5 \\ 2 \end{pmatrix}, \qquad \sum P_{i1} = 18, \; P'_1 = \begin{pmatrix} 0{,}3889 \\ 0{,}2222 \\ 0{,}2778 \\ 0{,}1111 \end{pmatrix}.$$

Продолжая этот итерационный процесс, получаем:

$$P_2 = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 7 \\ 4 \\ 5 \\ 2 \end{pmatrix} = \begin{pmatrix} 29 \\ 13 \\ 20 \\ 7 \end{pmatrix}, \qquad \sum P_{i2} = 69, \qquad P'_2 = \begin{pmatrix} 0{,}4203 \\ 0{,}1884 \\ 0{,}2899 \\ 0{,}1014 \end{pmatrix}.$$

$$P_3 = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 29 \\ 13 \\ 20 \\ 7 \end{pmatrix} = \begin{pmatrix} 109 \\ 47 \\ 76 \\ 27 \end{pmatrix}, \qquad \sum P_{i3} = 259, \qquad P'_3 = \begin{pmatrix} 0{,}4208 \\ 0{,}1884 \\ 0{,}2934 \\ 0{,}1042 \end{pmatrix}.$$

$$P_4 = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 109 \\ 47 \\ 76 \\ 27 \end{pmatrix} = \begin{pmatrix} 409 \\ 177 \\ 286 \\ 103 \end{pmatrix}, \qquad \sum P_{i4} = 975, \qquad P'_4 = \begin{pmatrix} 0{,}4194 \\ 0{,}1815 \\ 0{,}2933 \\ 0{,}1056 \end{pmatrix}.$$

$$P_5 = \begin{pmatrix} 1 & 2 & 2 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 409 \\ 177 \\ 286 \\ 103 \end{pmatrix} = \begin{pmatrix} 1541 \\ 669 \\ 1978 \\ 389 \end{pmatrix}, \quad \sum P_{i5} = 3677, \quad P'_5 = \begin{pmatrix} 0{,}4191 \\ 0{,}1819 \\ 0{,}2932 \\ 0{,}1058 \end{pmatrix}.$$

Получили требуемую точность вычислений.

При обработке оценок, полученных от нескольких экспертов, сначала вычисляют математическое ожидание (среднее арифметическое) для каждого коэффициента a_{ij} по всем сформированным экспертами матрицам. Определение весов различных объектов производится затем по рассмотренной выше схеме.

Согласно результатам теории полезности, отношение предпочтения инвариантно относительно выбора начала координат и масштаба порядковой шкалы [3]. Поэтому для того, чтобы определить значения функции принадлежности нечеткого множества «хороший антивирус», достаточно знать какие-либо два значения, к которым можно привязать итоговые веса. Например, если эксперт считает, что одному из оцениваемых антивирусов можно сопоставить значение μ_A(Kaspersky) = 0,9, а другому — μ_A(Avira) = 0,7, получим следующие значения функции принадлежности для рассматриваемых объектов (рис. 22).

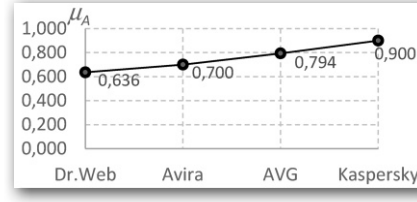

Рис. 22. Пример определения функции принадлежности нечеткой переменной «хороший антивирус»

Если сравнение пар объектов производится отдельно по различным показателям, то по каждому показателю составляется своя таблица результатов парных сравнений.

Метод парных сравнений часто используется для определения весовых коэффициентов или значимости тех или иных признаков. Однако с ростом числа объектов для сравнения метод становится слишком трудоемким.

3. Выбор одной из *типовых форм* функций принадлежности с возможным дальнейшим уточнением параметров функции в процессе эксперимента. Такой подход определяется высокой трудоемкостью вычислений при решении реальных задач моделирования

сложных систем с применением аппарата нечетких множеств. Для повышения эффективности выполнения большого объема операций над разного рода лингвистическими и другими нечеткими переменными желательно работать с функциями принадлежности стандартного вида.

В частности, при использовании пакетов прикладных программ, реализующих операции с нечеткой логикой (например, MatLab), предлагаются встроенные стандартные функций принадлежности (рис. 23).

Выбор вида стандартной функции во многом субъективен и определяется в большой степени опытом, интуицией и предпочтениями эксперта.

Существенным является тот факт, что для этих функций заранее известны их аналитические представления, что позволяет вычислить их значения в любой точке области определения. В то же время могут возникнуть определенные трудности при вычислении параметров аналитического представления функции принадлежности для конкретных лингвистических значений. При выборе стандартных функций принадлежности и определении их числовых параметров может быть использована следующая простая методика [22].

Рис. 23. Виды стандартных (типовых) функций принадлежности с указанием имен, принятых в среде MatLab

Сначала следует определить значения или границы (a, b) промежутков значений x, для которых $\mu_A(x) = 0$. В этом случае эксперт может однозначно определить носитель нечеткого множества, или базовое множество, соответствующее определенному лингвистическому значению.

Если же нулевое значение функции принадлежности достигается только в пределе: $\lim\limits_{|x|\to\infty} \mu_A(x) \to 0$. В таком случае эксперт должен ответить на вопрос типа «Какое минимальное значение должна иметь функция принадлежности $\mu_A(x)$, чтобы не считать элемент x принадлежащим данному множеству?» Ответ на этот вопрос определит в дальнейшем параметры функции принадлежности.

Следующий шаг — определение координат (c, d) плато функции принадлежности, то есть ответ на вопрос, какие значения x с наибольшей степенью уверенности принадлежат нечеткому множеству — функция принадлежности для них достигает своих максимальных значений (рис. 24).

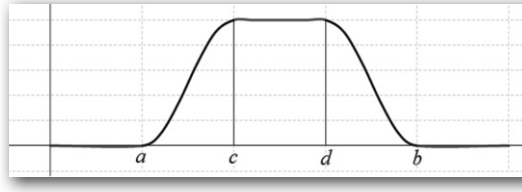

Рис. 24. Ключевые значения для задания параметров функции принадлежности

В большинстве случаев этих параметров достаточно для задания параметров аналитического представления стандартных функций принадлежности, и, как следствие, расчета значений функций принадлежности в любой точке. Однако для некоторых видов функций (например, экспоненциальных, параболических и гиперболических) этого оказывается недостаточно. В этом случае следует дополнительно определить *точку перехода* (crossover point) нечеткого множества, то есть такие значения x, для которых $\mu_A(x) = 0{,}5$. Для определения точки перехода может быть использована методика, основанная на анализе двух соседних лингвистических значений (термов) лингвистической переменной.

В этом случае для построения восходящей ветви для терма эксперта просят указать точку, относительно которой он испытывает наибольшие трудности при соотнесении ее с текущим и предыдущим термом. А при определении нисходящей ветви — с текущим и последующим (рис. 25).

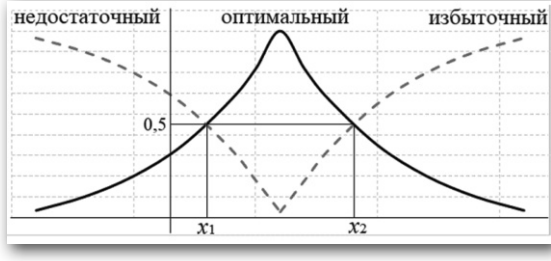

Рис. 25. Пример определения точек перехода
для термов лингвистической переменной
«уровень защищенности»

Пусть, например, лингвистическая переменная «уровень защищенности» имеет три терм-значения — «оптимальный», «недостаточный» и «избыточный». Точку перехода x_1 эксперт определяет, как значение, которое он одинаково может отнести как к нечеткому множеству «оптимальный уровень защищенности», так и к нечеткому множеству «недостаточный уровень защищенности». Так же эксперт испытывает одинаковые сомнения относительно того, является ли значение x_2 «оптимальным» или «избыточным».

В частности, знание значений аргументов для нулевого значения, плато (1) и значения 0,5 функции принадлежности позволяет использовать для ее построения стандартные S-образные функции. Это, в свою очередь позволяет при непрерывном носителе рассчитывать значения функции принадлежности для произвольного значения лингвистической переменной, не включенного в исходное базовое множество.

4. *Экспериментальное* задание — вид функций принадлежности аппроксимируется по статистическим данным проводимого эксперимента. Такой метод задания характерен для моделирования систем автоматического управления, например, в автомобильной промышленности.

Проверка построенной нечеткой модели на известных данных позволяет проверить ее адекватность и, при необходимости, уточнить вид функции принадлежности или откорректировать значения ее параметров.

Нечеткий логический вывод

Существенным ограничением классической логики является использование только двух истинностных значений: «истина» и «ложь» (или 1 и 0 в числовом выражении). Для преодоления этого недостатка был предложен целый ряд многозначных логик, однако *нечеткая логика* моделирует, прежде всего, приближенные рассуждения, а не точные многозначные рассуждения. Это значит, что нечеткая логика позволяет делать вывод заключений, которые могут оказаться не точными, из посылок, которые также могут быть не точны. Могут быть определены различные варианты нечеткой логики. В нечеткой логике, предложенной Л. Заде, истинностные значения высказывания могут принадлежать вещественному интервалу [0,1]. Численное значение из этого интервала является количественной оценкой степени истинности высказывания.

Нечеткое высказывание — утверждение, относительно истинности которого можно судить лишь с некоторой степенью уверенности. Нечеткие высказывания могут иметь следующий вид.

1. Высказывание <β есть β′>, где β — наименование лингвистической переменной, β′ — ее значение, которому соответствует нечеткое множество на универсальном множестве *X*. Например, «объем данных большой», предполагает, что лингвистической переменной объем данных сопоставлено значение «большой», для которого задано нечеткое множество на универсальном множестве *X*.

2. Высказывание <β есть *m*β′>, где *m* — модификатор («очень», «довольно», «более-менее» и др.). Например, «объем данных очень большой», «риск не велик» «антивирус достаточно хороший» и т. п.

Такие утверждения как «число соединений намного больше обычного» или «число соединений близко к обычному» требуют определения нечетких отношений R_1 («намного больше, чем») и R_2 («близко к»), заданных на декартовом произведении $X \times X$. Подобные утверждения рассматриваются как высказывания, которым будут соответствовать нечеткие множества, индуцированные нечеткими отношениями R_1 и R_2.

Фактически, нечеткое отношение, при интерпретации значений функции принадлежности как значений истинности, задает нечеткий предикат. При означивании переменных нечеткого предиката (то есть присвоении им конкретных значений) он превращается в некоторое нечеткое высказывание. На практике нечеткое обобщение логики предикатов первого порядка не нашло широкого применения, вместо этого, как правило, рассматривается нечеткое обобщение продукционных правил, использующих означенные лингвистические переменные.

3. Из высказываний рассмотренных типов могут быть образованы составные высказывания с использованием связок «И», «ИЛИ», «ЕСЛИ..., ТО...», «ЕСЛИ..., ТО..., ИНАЧЕ...».

Нечеткие множества, соответствующие составным высказываниям вида <α есть α′ И β есть β′> и <α есть α′ ИЛИ β есть β′>, определяются по следующим правилам:

$$\mu_{A\cap B}(x,y) = \min(\mu_A(x),\mu_B(y));$$
$$\mu_{A\cup B}(x,y) = \max(\mu_A(x),\mu_B(y)).$$

Что касается импликации <ЕСЛИ α есть α′, ТО β есть β′>, то ей соответствует нечеткое отношение R, заданное на декартовом произведении множеств $X\times Y$, функция принадлежности $\mu_R(x,y)$ которого зависит от выбранного способа задания нечеткой импликации [20]. Чаще всего импликация определяется как $\mu_R(x,y) = \min(\mu_A(x),\mu_B(y))$.

Нечетким логическим выводом (fuzzy logic inference) называется получение заключения в виде нечеткого множества, соответствующего текущим значениям входов, с использованием нечеткой базы знаний и нечетких операций.

Основу нечеткого логического вывода составляет *композиционное правило Заде*. Оно формулируется в общем виде и универсально, однако платой за универсальность является высокая сложность системы. Поэтому различными исследователями предпринимались попытки определения нечеткой импликации и разработки соответствующих правил вывода. Наибольшую известность получили алгоритмы логического вывода Е. Мамдани (Ebrahim Mamdani) и Такаги-Сугено (Takagi-Sugeno).

Выбор того или иного метода нечеткого логического вывода определяется решаемой задачей. Если решается задача извлечения знаний из данных (в виде лингвистических правил) или производится поиск ассоциативных связей во множестве данных, то для этих целей лучше использовать нечеткую систему типа Мамдани. В моделях Мамдани

правила базы знаний строятся из высказываний вида <ЕСЛИ x есть A, ТО y есть B> и допускает использование связок И в посылках.

Если решается задача интерполяции или аппроксимации, причем точность является определяющим фактором, то выбор должен быть сделан в пользу нечеткой системы типа Такаги-Сугено, который обеспечивает большую точность на больших выборках данных. В алгоритме Такаги-Сугено база знаний строится в виде <ЕСЛИ x есть A И то y есть B, ТО $z = ax+bx$>, здесь a и b — четкие значения входных переменных, соответствующие высказываниям A и B соответственно, z — четкое значение выходной переменной. Алгоритм Сугено применяется, когда известна не форма функции соответствия выходного параметра, а весовые коэффициенты влияния входных параметров.

Прежде всего требуется задать базу правил нечетких продукций — конечное множество нечетких продукций в форме «ЕСЛИ..., ТО...», согласованных относительно используемых лингвистических переменных. Каждое правило снабжается значением коэффициента определенности или весовым коэффициентом правила (CF). Кроме того, должны быть определены функции принадлежности для соответствующих лингвистических термов.

Чтобы база знаний была полной, правила должны удовлетворять следующим условиям:

— существует хотя бы одно правило для каждого лингвистического терма выходной переменной;

— для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве посылки.

Пусть в базе знаний имеется k правил вида «ЕСЛИ..., ТО...», в которых посылки и заключения представляют собой нечеткие высказывания вида 1 или 2, связанные словами «И», «ИЛИ». С помощью правил преобразования дизъюнктивной и конъюнктивной формы описание системы можно привести к виду:

$$L_1\text{: ЕСЛИ } <A_1>\text{, ТО } <B_1> (CF_1),$$
$$L_2\text{: ЕСЛИ } <A_2>\text{, ТО } <B_2> (CF_2),$$
$$\ldots$$
$$L_k\text{: ЕСЛИ } <A_k>\text{, ТО } <B_k> (CF_k),$$

где A_i, B_i — нечеткие множества, заданные на декартовом произведении универсальных множеств входных и выходных лингвистических переменных соответственно, $i = 1, \ldots, k$.

В общем случае механизм нечеткого логического вывода включает следующие основные этапы.

1. Этап фаззификации (введения нечеткости) — функции принадлежности, определенные для входных переменных, применяются к их фактическим значениям для определения степени истинности каждой предпосылки каждого правила.

2. Этап непосредственного нечеткого вывода — вычисленные значения истинности предпосылок применяются для получения заключения каждого правила. Это приводит к получению одного нечеткого подмножества для каждой выходной переменной в каждом правиле.

3. Этап композиции (агрегации, аккумуляции) — выходы всех правил вычисляются отдельно, но в правой части нескольких из них может быть указана одна и та же нечеткая переменная. Нечеткие подмножества, назначенные каждой выходной переменной, объединяются вместе для формирования одного нечеткого подмножества.

4. Необязательный этап дефаззификации — нахождение обычного (точного) значения для каждой из выходных лингвистических переменных. Дефаззификация может потребоваться, когда результаты должны быть использованы устройствами или процедурами, внешними по отношению к нечеткой системе. Используются следующие алгоритмы перехода от нечеткости к точным значениям [20]:

— метод центра тяжести (center of gravity) — нахождение абсциссы центра тяжести площади, ограниченной графиком функции принадлежности выходной переменной $x_0 = \int x\mu(x)dx / \int xdx$. Для приближенного вычисления центра тяжести может быть использована формула $x_0 = \sum \mu(x_i)x_i / \sum x_i$.

— метод центра максимумов (mean of maximums) — центр тяжести вычисляется только по подмножеству значений выходной переменной, соответствующих максимальному значению функции принадлежности $x_0 = \int xdx / \int dx$. Для приближенного вычисления используется среднее значение $x_0 = \frac{1}{n}\sum x_i$.

— метод первого максимума (first maximum) — берется наименьшая абсцисса (самое левое значение выходной переменной), соответствующая максимальному значению функции принадлежности $x_0 = \min\{x \mid \mu_A(x) = \max\mu_A\}$.

Рассмотрим простой пример нечеткого логического вывода по правилам, который достаточно просто и полно описывает его сущность.

Пусть система мониторинга рисков использует знания эксперта о том, что работа в корпоративной информационной системе в нерабочее время подозрительна. Это знание можно представить с помощью нечеткого продукционного правила «если ..., то...» следующим образом:

ЕСЛИ «сеанс начат в не рабочее время», ТО «повысить уровень риска».

Здесь как предпосылка, так и заключение выражены в виде нечетких отношений, т. е. нет данных о том, какое конкретно сейчас время и на сколько надо повысить уровень риска. Однако сам эксперт это знает. Например, время сразу после окончания рабочего дня не будет подозрительным, поскольку некоторые работники могут немного задержаться для того, чтобы завершить какие-то операции. В то же время начало активности в ночное время может вызвать сильное подозрение. При этом интерпретация в виде нечеткого множества, например, $A = \{18/0, 20/0{,}2, 22/0{,}4, 24/0{,}9, 2/1, 4/1, 6/0{,}9, 7/0{,}6, 8/0{,}2, 9/0\}$ более точно отражает оценку эксперта, нежели строгая интерпретация: с 9 до 18 часов — сеанс начат в рабочее время, а с 18.01 до 8.59 — в не рабочее. Выразив время как как задержку после окончания рабочего дня, можно переписать $A = \{0/0, 2/0{,}2, 4/0{,}6, 6/0{,}9, 8/1, 10/1, 12/0{,}9, 13/0{,}6, 14/0{,}2, 15/0\}$.

Аналогично, повышение уровня риска может быть выражено как нечеткое множество $B = \{0/0, 0{,}5/0{,}3, 1/0{,}5, 1{,}5/0{,}7, 2/1\}$.

Пусть наблюдаем, что подключения к системе производились в течение 2–3 часов после окончания работы. Эксперт считает, что ЕСЛИ «задержка после работы не очень большая», ТО «надо несколько повысить уровень риска». Здесь целесообразно выразить терм «не очень большая» с помощью нечеткого множества, например, следующим образом: $A' = \{0{,}5/0, 1/0{,}6, 2/1, 3/0{,}4, 4/0{,}2, 6/0\}$.

Таким образом имеется два правила:

ЕСЛИ «сеанс начат в не рабочее время», ТО «повысить уровень риска»;

ЕСЛИ «задержка после работы не очень большая», ТО «надо несколько повысить уровень риска».

Это заключение эксперта и есть, по сути, нечеткий вывод, а точнее — приближенные рассуждения.

Проиллюстрируем вывод графически, опираясь на формальное описание операций с нечеткими множествами. Здесь полное пространство предпосылок — время задержки начала сеанса работы пользователя после окончания рабочего дня (X), а полное пространство заключений — значения прибавки к уровню риска (Y). Используя заданные нечеткие множества A и B, исходное нечеткое продукционное правило можно графически изобразить как на рисунке 26.

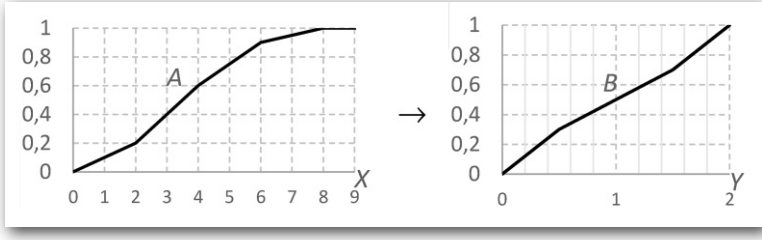

Рис. 26. Графическое представление нечеткого правила

Графическое представление нечеткого множества «задержка после работы не очень большая» приведено на рисунке 27.

Тогда процесс классического нечеткого логического вывода по правилам может быть графически представлен как на рисунке 28.

Сначала получен результат приближенного сопоставления посылки правила A и данных наблюдения A' как $A \cap A'$. Затем максимальное значение a рассматривается как некая мера сопоставления $A \cap A'$ и выполняется редукция по этой мере заключения B. В качестве способа редукции B выбрано отсечение по мере сопоставления a (на рис. 28 запись aY означает, что $\mu_{aY}(y) = a$ для любого $y \in Y$). Как результат вывода получается нечеткое множество B'.

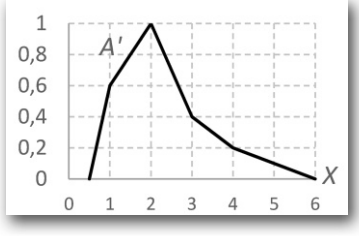

Рис. 27. Нечеткое множество «задержка не очень большая»

На этапе дефаззификации следует получить единственное числовое значение выходного параметра. Если для этих целей используется алгоритм первого максимума, то четкое значение для операции «повысить уровень риска» равно 0,75 (рис. 28).

Чаще всего, для дефаззификации используется метод центра тяжести.

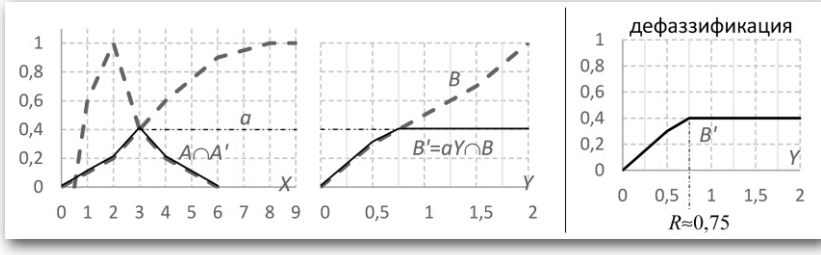

Рис. 28. Графическое представление нечеткого логического вывода

Тогда вычисляется (точно или приближенно) площадь фигуры под кривой функции принадлежности выходного параметра, а затем находится такая абсцисса, которая делила бы эту площадь на две равные части (рис. 29, а). В случае применения метода центра максимумов пополам делится сумма длин отрезков на оси абсцисс, соответствующих максимальным значениям функции принадлежности (рис. 29, б).

Следует отметить, что для работы с нечеткими моделями, как правило, используются специализированные пакеты прикладных программ (например, расширение FuzzyLogic Toolbox для MatLab, Wolfram Mathematica Fuzzy Logic, FuzzyTECH, CubiCalc, FisPro и др.), реализующих нечеткий логический вывод. В таком случае работа с нечеткой моделью сводится к заданию нечетких правил и описанию входных и выходных переменных (путем задания для них функций принадлежности).

В заключение рассмотрим несколько примеров фрагментов нечетких баз знаний, которые могут быть применены в области информационной безопасности.

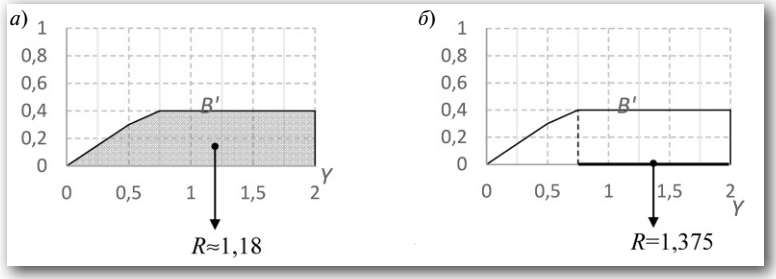

Рис. 29. Пример дефаззификации с использованием метода
а) центра тяжести б) центра максимумов

Так, рассмотренный в п. 1.2. набор продукционных правил, описывающих выбор типа резервного копирования, по сути, уже содержит лингвистическую переменную «данные» (X_1) со значениями «мало» и «много». Переменную «приоритет» можно заменить на две лингвистические переменные «требования_к_скорости_восстановления» (X_2) и «требования_к_скорости_записи» (X_3) с терм-значением «высокие», «низкие». В качестве заключения правил можно выбирать частоту проведения полного и инкрементного копирования. Тогда потребуется ввести две лингвистические переменные «частота_полного_копирования» (Y_1) и «частота_инкрементного_копирования» (Y_2) с терм-значениями «часто», «редко», «никогда».

Тогда база правил выбора типа копирования может выглядеть следующим образом.

ЕСЛИ X_1 = «мало» И X_2 = «очень высокие» И X_3 = «низкие» ТО Y_1 = «часто» И Y_2 = «никогда».

ЕСЛИ X_1 = «мало» И X_2 = «высокие» ТО Y_1 = «часто» И Y_2 = «редко».

ЕСЛИ X_1 = «мало» И X_3 = «высокие» ТО Y_1 = «редко» И Y_2 = «часто».

ЕСЛИ X_1 = «много» И X_2 = «высокие» И X_3 = «не высокие» ТО Y_1 = «часто» И Y_2 = «не часто».

ЕСЛИ данные = «много» И X_2 = «высокие» И X_3 = «высокие» ТО Y_1 = «не часто» И Y_2 = «не часто».

ЕСЛИ X_1 = «много» И X_2 = «не высокие» И X_3 = «высокие» ТО Y_1 = «очень редко» И Y_2 = «часто».

В [2] приведен набор нечетких правил для обнаружения DDos атак типа SYN Food. Для анализа используются TCP-флаги (SYN), адреса отправителя и получателя, направленные пакеты (входящие или исходящие), время поступления пакета, порт получателя, корректность заголовка. Используются четыре лингвистические переменные:

X_1 — среднее время поступления одного пакета;

X_2 — процент пакетов с различными внешними IP-адресами;

X_3 — процент пакетов с различными внешними портами;

X_4 — процент пакетов с некорректными заголовками.

Выходным параметром является переменная Y — степень уверенности в атаке. Экспертные правила для обнаружения атаки задаются следующим образом.

ЕСЛИ X_1 = «мало» И X_2 = «мал» И X_3 = «мал», ТО Y = «средняя».

ЕСЛИ (X_1 = «мало» ИЛИ X_1 = «среднее») И (X_2 = «велик» ИЛИ X_3 = «велик» ИЛИ X_4 = «велик»), ТО Y = «высокая».

ЕСЛИ X_1 = «велико», ТО Y = «низкая».

В [4] сделана попытка построения нечеткой модели оценки рисков нарушения политики информационной безопасности, в частности, положений о запрете использования ресурсов корпоративной информационной системы в нерабочее время. При этом предполагается, что политикой установлена периодичность мониторинга файлов. База нечетких правил использует следующие лингвистические переменные:

X_1 — число активных пользователей системы;

X_2 — время;

X_3 — время, прошедшее с момента последнего анализа файлов мониторинга

Y_1 — возможность проникновения;

Y_2 — возможность распространения атаки;

Y_3 — нарушение политики мониторинга;

Y_4 — уровень риска.

База знаний сформулирована следующим образом.

ЕСЛИ X_1 = «несколько» И X_2 = «нерабочее», ТО Y_1 = «большая».

ЕСЛИ X_1 = «очень много», ТО Y_1 = «большая» И Y_2 = «высокая».

ЕСЛИ X_3 = «значительное», ТО Y_3 = «большое».

ЕСЛИ Y_1 = «большая», ТО Y_4 = «высокий».

ЕСЛИ Y_2 = «большая» И Y_3 = «большое», ТО Y_4 = «очень высокий».

Не затрагивая содержательную часть этой нечеткой модели, отметим, что в данном примере можно снизить размерность задачи и сократить цепочку логического вывода, исключив переменную Y_3 (и третье правило, в котором она выступает в роли заключения) и переформулировав последнее правило как

ЕСЛИ Y_2 = «большая» И X_3 = «значительное», ТО Y_4 = «очень высокий».

Несомненным достоинством нечетких моделей является возможность упрощенного описания и решения плохо формализуемых задач с использованием средств вычислительной техники. Нечеткие правила понятны и легко интерпретируются экспертами. Вместе с тем необходимо четко понимать и их недостатки.

Основной проблемой является субъективность нечетких моделей, поскольку исходный набор нечетких правил формулируется экспертом. Как следствие, база правил может оказаться неполной или содержать

противоречия. Если вид и параметры функций принадлежности, описывающие входные и выходные переменные системы, выбираются субъективно, то они могут неадекватно отражать реальность.

Усиление нечеткости модели, то есть увеличение числа входных нечетких переменных приводит к резкому росту числа правил, что затрудняет восприятие базы знаний и экспоненциально увеличивает сложность вычислений. Кроме того, в настоящее время отсутствует стандартная методика проектирования и расчета нечетких систем.

Практический интерес представляет использование нейро-нечетких систем (fuzzy-neural networks), осуществляющих вывод на основе нечетких правил, параметры функций принадлежности для которых настраиваются автоматически в процессе обучения нейронной сети.

2.2. Нейронные сети

Биологические и искусственные нейроны

Высокоорганизованные живые организмы вынуждены приспосабливаться к постоянно изменяющимся условиям внешней среды, в основе этой функции лежат адаптационные возможности нервной системы. В основе функционирования нервной системы лежит взаимодействие нервных клеток — *нейронов*, связанных между собой нервными волокнами, которые служат для передачи сигналов. Каждый нейрон имеет несколько отростков, по которым принимаются сигналы, — *дендритов* и один выходной — *аксон*. Импульсы, поступившие к нейрону одновременно по нескольким дендритам, суммируются. Если суммарный импульс превышает некоторый порог, нейрон возбуждается, формирует собственный импульс и передает его далее по аксону. Аксон соединен через специальные образования — *синапсы* с дендритами других нейронов, которым передается выходной сигнал.

Синапсы могут изменять свою проводимость (сопротивление) со временем, усиливая или ослабляя передаваемые сигналы — так реализуется процесс обучения. *Весом синапса* называют то, во сколько раз изменяется сила импульса при прохождении синапса. Значения величин сформированных синаптических связей могут сохраняться неопределенно долго, что обеспечивает сохранение памяти в течении жизни.

Описанный таким образом нейрон представляет собой «черный ящик» со многими входами и одним выходом. Адаптация к изменениям окружающей среды заключается в изменении весов синапсов.

Первые модели искусственных нейронных сетей появились еще в 40-х годах прошлого века, представляя нейрон как линейную комбинацию входов, которая затем поступает на вход нелинейности в виде «ступеньки», сравнивающей результат с некоторым порогом. При этом отмечалась бинарная природа нейронной активности (нейрон либо «включен», либо «выключен», практически без промежуточных состояний), что допускает описание в терминах пропозициональной логики.

В современной интерпретации эту модель реализует **формальный нейрон**, состоящий из элементов трех типов: *умножителей* (*синапсов*), *сумматора* и *нелинейного преобразователя*, и реализующий две основные функции: взвешенное суммирование и нелинейное преобразование (рис. 30).

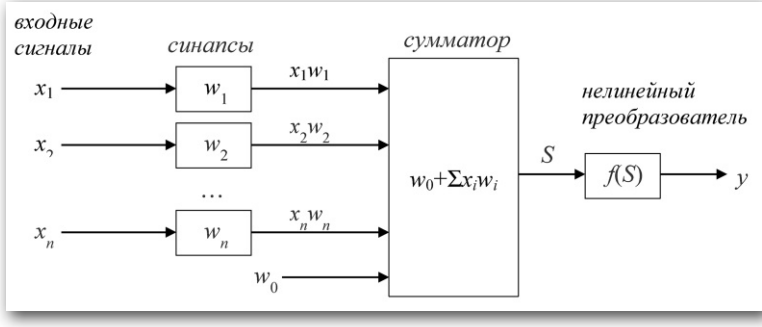

Рис. 30. Структурная схема искусственного нейрона

Синапсы умножают входной сигнал (x_i) на число, характеризующее силу связи, — вес синапса (w_i). Затем в сумматоре вычисляется величина возбуждения S, полученного нейроном, как

$$S = w_0 + \sum_{i=1}^{n} x_i w_i ,$$

где n — число входов нейрона, w_0 — смещение нейрона, используемое для задания порогового значения функции активации.

В некоторых сетях величина возбуждения нейрона рассчитывается по формуле $S = w_0 + \sum_{i=1}^{n} (w_i - x_i)^2$. При некоторых ограничениях на входные вектора и синаптические веса можно показать эквивалентность

задач подбора весов, максимизирующих первую функцию и минимизирующих вторую.

Следующим шагом возбуждение S пропускается через преобразующую (активационную) функцию $f(S)$, в результате чего определяется выходной сигнал

$$y = f(S) = f(w_0 + \sum_{i=1}^{n} x_i w_i) .$$

Существует много разновидностей функций активации, однако чаще всего используются две из них:

— пороговая (ступенчатая) функция $f(S) = \begin{cases} 1, \text{ при } S > \delta \\ 0, \text{ при } S \le \delta \end{cases}$, где δ — порог срабатывания нейрона;

— нелинейная логистическая (сигмоидная) функция $f(S) = \frac{1}{1+e^{-aS}}$, где $a > 0$, $0 < f(S) < 1$ (рис. 31), близкая по своим свойствам к передаточным характеристикам биологических нейронов.

Пороговая функция более удобна при аппаратной реализации нейрона, а сигмоидная — в аналитических исследованиях.

Рассмотрена модель искусственного нейрона является наиболее распространенной, однако это не единственный способ комбинирования входных сигналов, и к настоящему времени предложено достаточно много моделей искусственных нейронов.

Различные способы объединения отдельных искусственных нейронов порождают более сложные образования, в которых воспроизводится свойство обучения, — **искусственные нейронные сети** (ИНС). Искусственная нейронная сеть обладает новым уровнем функциональности, намного превосходящим возможности отдельных нейронов и, кроме того, отличным от возможностей традиционных вычислительных систем.

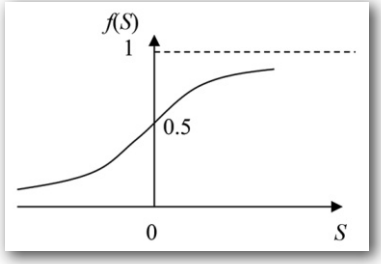

Рис. 31. Логистическая (сигмоидная) функция

В настоящее время предложено множество различных ИНС, отличающихся типом используемых нейронов, структурой связей, методами обучения и назначением.

Несмотря на то, что теория ИНС стала разрабатываться достаточно давно, прорыв в их практическом применении стал заметен лишь в последние годы, чему во многом способствовала возможность аппаратной реализации с высокой степенью параллелизма. Так, первые реализации нейронных сетей на графических процессорах появились в 2009 году, известны реализации на ПЛИС, кроме того, компании разрабатывают специальные аппаратные платформы (например, Intel Nervana Neural Network Processor, IBM SyNAPSE, Google Tensor Processing Unit). Другими факторами стали доступность больших выборок обучающих данных и открытых библиотек машинного обучения.

В настоящее время ИНС с успехом используются, для решения, прежде всего, задач распознавания образов: классификации — определения принадлежности входного образа заданным классам и кластеризации — группировки близких подобных образов в один кластер. При наличие пропущенных данных можно говорить об аппроксимации и прогнозировании. В том числе традиционным приложением ИНС стали следующие области:

— распознавание речи (использование в системах голосовой помощи, голосового поиска), распознавание изображений (например, рукописного текста, лиц, распознавание видео с камер слежения);
— обработка естественного языка (автоматический перевод, контекстный анализ, генерация текстов), в то же время до действительного понимания текста, то есть обработки содержащейся в тексте информации на основе ее семантики, еще далеко;
— обработки сигналов, в частности визуальных, в роботике (например, в беспилотных автомобилях и других устройствах).

В области ИБ это направление находит применение при реализации биометрических систем аутентификации, а также в различного рода системах обнаружения угроз (IDS/IPS, SIEM, DLP) при определении типа угрозы, выявлении признаков аномальной активности и злоупотреблений. Кроме ИНС для построения таких СЗИ часто используются генетические алгоритмы и искусственные иммунные системы.

Нейросетевые системы могут найти свое применение и в компьютерной криминалистике для проведения ретроспективного анализа больших объемов данных (например, содержимого лог-файлов), а также определения цифровых подделок, созданных с использованием

глубокого обучения (дипфейков). В настоящее время с высокой степенью достоверности могут быть определены сгенерированные ИНС тексты на естественном языке (gltr.io).

Вместе с тем, настоящие нейроны устроены значительно сложнее, чем используемые модели. Следует отметить асинхронность и очень высокую степень параллелизации работы биологической нервной системы, а также ее пластичность за счет формирования новых связей между нейронами и их постоянным обновлением (нейрогенезисом), а также за счет возможности переобучения уже существующих нейронов для обработки совершенно новых сигналов. На современном этапе не удалось построить полноценной модели нервной системы даже для достаточно простых организмов (типа червя нематоды, имеющей всего порядка 300 нейронов).

Существуют и количественные различия, как в скорости передачи сигналов (миллисекунды в биологических против наносекунд в искусственных нейронных сетях), так и в количестве нейронов в сети. Количество нейронов в мозге человека составляет величину не менее 10^{11} с порядка 10^{15} связей между ними, причем все нейроны работают параллельно. В компьютерных системах реально могут быть реализованы ИНС с тысячами нейронов и десятками тысяч связей.

ИНС, так же, как и их биологический прототип, могут адаптироваться к изменяющейся внешней среде. ИНС способны обучаться на основе опыта, обобщать предыдущие прецеденты на новые случаи и извлекать существенные свойства из поступающей информации, содержащей зашумленные и искаженные данные. Кроме того, некоторые типы ИНС предназначены не только для анализа входящей информации, но и для воспроизведения ее из своей памяти, то есть автоматической генерации правдоподобных объектов (например, воспроизведение элементов человеческого поведения, генерация текстов на естественном языке, музыки, изображений).

В ходе обучения сеть настраивается так, чтобы обеспечивать требуемую реакцию. Отклик сети после обучения может быть до некоторой степени нечувствителен к небольшим изменениям входных сигналов. Эта внутренне присущая способность ИНС видеть образ сквозь шум и искажения крайне важна при решении практических задач.

Основной проблемой использования нейросетевых моделей называют невозможность объяснения принимаемого ИНС решения, хотя нередко доля ошибок ИНС оказывается существенно ниже доли

ошибок, вызванных «человеческим фактором». Разработчики ИНС не могут гарантировать отсутствия ошибок, даже критических, только за счет конструктивных особенностей системы. Невозможность исключить подобные ситуации на этапе проектирования системы и фактически определение проблем опытным путем на этапе эксплуатации — экспериментальной или рабочей, порождают проблему доверия.

Классификация искусственных нейронных сетей

По способу объединения нейронов в сеть можно выделить три основных вида архитектуры ИНС:

— полносвязные сети (рис. 32, а), в которых каждый нейрон связан со всеми остальными (на входы каждого нейрона подаются выходные сигналы всех остальных нейронов);

— слабосвязные сети с локальными связями (рис. 32, б), где нейроны располагаются в узлах прямоугольной или гексагональной решетки; каждый нейрон связан с четырьмя, шестью или восемью ближайшими соседями;

— многослойные сети (рис. 32, в), в которых нейроны объединены в слои.

Слой — это совокупность нейронов с единым входным сигналом. Внешние входные сигналы подаются на входы нейронов первого слоя, а выходами сети являются выходные сигналы нейронов последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько промежуточных (скрытых) слоев. Входной слой может рассматриваться как вырожденный, служащий только для ввода значений входных переменных. Среди многослойных нейронных сетей выделяют следующие типы.

1. *Монотонные* — частный случай слоистых сетей с дополнительными условиями на связи и нейроны. Каждый слой кроме последнего

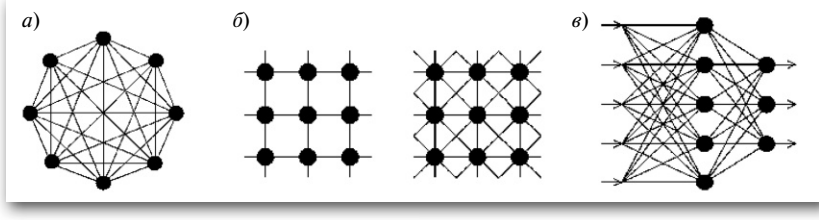

Рис. 32. Основные типы архитектур ИНС: а) полносвязная сеть, б) слабосвязные сети, в) многослойная сеть с последовательными связями

(выходного) разбит на два блока: возбуждающий и тормозящий. Связи между блоками тоже разделяются на тормозящие и возбуждающие. Для нейронов монотонных сетей зависимость выходного сигнала нейрона от параметров входных сигналов должна описываться монотонной функцией.

2. *Сети без обратных связей* (сети *прямого распространения* сигналов) — в таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя. Обычно каждый выходной сигнал k-го слоя подается на вход всех нейронов $(k+1)$-го слоя, однако возможен и вариант соединения k-го слоя с произвольным $(k+p)$-м слоем. Среди многослойных сетей без обратных связей различают *полносвязные* (выход каждого нейрона k-го слоя связан с входом каждого нейрона $(k+1)$-го слоя) и *частично полносвязанные*.

3. *Сети с обратными связями* (*рекуррентные* сети, recurrent neural network, RNN) — информация с последующих слоев передается на предыдущие слои.

Среди сетей с обратными связями наиболее часто встречаются:

— слоисто-циклические сети — слои замкнуты в кольцо: последний слой передает свои выходные сигналы первому; все слои равноправны и могут как получать входные сигналы, так и выдавать выходные;

— слоисто-полносвязанные сети — состоят из слоев, каждый из которых представляет собой полносвязную сеть, а сигналы передаются как от слоя к слою, так и внутри слоя; в каждом слое цикл работы распадается на три части: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача к следующему слою;

— полносвязанно-слоистые — по своей структуре аналогичны слоисто-полносвязанным, но функционируют по-другому: на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих.

Отсутствие обратных связей в сетях прямого распространения гарантирует их безусловную устойчивость. Однако, такие сети обладают более ограниченными возможностями, чем сети с обратными связями.

Многослойные сети можно классифицировать по *числу слоев*. ИНС с многими скрытыми слоями называются *глубокими* нейронными се-

тями (deep neural network, DNN). Теоретически число слоев и число нейронов в каждом слое может быть любым, однако фактически оно ограничено ресурсами компьютера или специализированных микросхем, на которых реализована нейронная сеть. Чем сложнее сеть, тем более сложные задачи она может решать.

Помимо классификации по архитектурным особенностям, ИНС могут быть классифицированы:

— по типу входной информации (входная информация представлена в двоичном виде или в виде вещественных чисел);
— по способу преобразования входной информации (автоассоциативные сети, в которых входной вектор сигналов является искажением или сжатием некоторого эталона, и гетероассоциативные, в которых входной вектор отличен от выходного);
— по виду сигналов на выходе нейронов (бинарные сети, в которых выходной сигнал представлен в двоичном виде, и аналоговые, в которых выходы нейронов являются вещественными числами);
— по однородности состава (гомогенные, состоящие из нейронов с одинаковой функцией активации, и гетерогенные, нейроны которых имеют различные функции активации);
— по методам обучения (с учителем или без учителя);
— по времени срабатывания нейронов (синхронные сети, в которых одновременно состояние изменяют все нейроны одного слоя, и асинхронные).

Наиболее распространенными являются сети прямого распространения (многослойные однонаправленные сети или *персептроны*), сети с обратными связями (например, нейронная сеть с долгой краткосрочной памятью, LSTM), самоорганизующиеся *сети Кохонена* и *сети встречного распространения*.

Выбор структуры ИНС осуществляется в соответствии с особенностями и сложностью задачи. Кроме выбора типа сети требуется задать ее основные параметры — число слоев, число нейронов в слое, вид функции активации нейронов. Для решения отдельных типов задач уже исследованы оптимальные конфигурации, например, для анализа временных рядов лучше всего подходят рекуррентные сети, в частности, сети с долгой краткосрочной памятью (long short-term memory, LSTM). Если же задача не может быть сведена ни к одному из известных типов ИНС, приходится решать сложную проблему

синтеза новой конфигурации. При этом необходимо принимать во внимание следующие особенности ИНС:

— возможности сети возрастают с увеличением числа нейронов сети, плотности связей между ними и числом слоев;
— введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о ее динамической устойчивости;
— сложность алгоритмов функционирования сети, введение нескольких типов синапсов способствует усилению мощности нейронной сети;
— увеличение масштаба сети требует больших объемов данных для обучения;
— усложнение сети, в особенности увеличение числа слоев, приводит к существенному увеличению трудоемкости ее обучения.

Обучение нейронных сетей

Применение ИНС для решения любой задачи включает два этапа: этап *обучения* и этап *распознавания*.

Под **обучением ИНС** обычно понимается настройка параметров сети, то есть оптимизация синаптических весов, при которых минимизируется ошибка предсказания выходных значений. Топология же сети обычно считается неизменной. Обучение сети может производиться по одному из двух базовых сценариев: обучение с учителем и обучение без учителя.

Обучение *с учителем* (supervised training) производится на примерах, называемых *обучающей выборкой*, и подразумевает, что для каждого вектора значений входных переменных примера обучающего множества известен требуемый вектор значений выходных переменных. Такие данные еще называют *размеченными*.

Модификацией метода обучения с учителем является обучение с последовательным *подкреплением* знаний. Обучение с подкреплением тренирует алгоритм при помощи системы стимулов. В этом случае после анализа обучающего примера сеть получает обратную связь в виде оценки, показывающей, насколько правильно она работает на этом примере.

Обучение с учителем применяется для решения двух типов задач: классификации и прогнозирования. Основным же ограничением является потребность в наличии внушительного набора достоверных размеченных данных. Недостаток данных — наиболее часто встречающаяся проблема в машинном обучении.

К типу задач обучения с учителем могут также относиться задачи выявления логических закономерностей и построения совокупности диагностических правил (шаблонов) для последующего распознавания.

Для многих алгоритмов обучения с учителем существует опасность *переобучения*, в результате которого сеть подстраивается исключительно под обучающую выборку, теряя способность к обобщению. При этом сеть учитывает также и шум, характерный для конкретного обучающего набора данных, делая тем самым попытку описать конкретные данные точнее, чем это в принципе позволяет уровень шума в данных и погрешность самой модели. В результате сеть теряет работоспособность на реальных данных.

Для того, чтобы контролировать обобщающие способности ИНС, рекомендуется разделить весь набор размеченных данных на три выборки (например, в соотношении 70:20:10) — обучающую, тестовую и проверочную (валидационную). Обучение производится на обучающей выборке, при этом качество работы нейросетевой модели периодически проверяется на тестовой выборке. Для финальной оценки необходимо использовать данные проверочной (валидационной) выборки.

Обучение *без учителя* (unsupervised learning) реализует механизм настройки весов сети в случае, когда доступны только неразмеченные данные, то есть известны только значения входных переменных для примеров из обучающего множества. ИНС в этом случае пытается самостоятельно найти корреляции (зависимости) в данных, извлекая полезные признаки и анализируя их.

Обучение без учителя используется при решении задач кластеризации (разбиения множества данных на подмножества сходных объектов без заранее заданных классов), обнаружения аномалий и поиска ассоциаций (поиска связанных объектов по некоторым ключевым признакам выбранного объекта). Обучение без учителя может применять в том случае, если размеченные данные ненадежны или их получение слишком дорого. Платой является сложность оценки точности полученных результатов.

Обучение *с частичным привлечением учителя* (semi-supervised learning) является «золотой серединой» между этими двумя подходами — в этом случае обучающая выборка содержит как размеченные, так и неразмеченные данные. Такой подход особенно полезен, когда трудно заранее определить важные признаки или разметка всех объектов выборки является слишком трудоемкой.

Обучение с частичным привлечением учителя нередко используют для решения медицинских задач (например, для анализа медицинских изображений, таких как рентгеновские снимки, снимки компьютерной томографии или МРТ), где наличие небольшого количества размеченных данных может привести к значительному повышению точности результата.

Популярный метод обучения, для которого требуется лишь относительно небольшой набор размеченных данных, заключается в использовании генеративно-состязательной сети (generative adversarial network, GAN). Этот алгоритм построен на комбинации двух состязающихся нейронных сетей, одна из которых (генератор, G) генерирует образцы, а другая (дискриминатор, D) старается отличить правильные («подлинные») образцы от «поддельных» (то есть сгенерированных сетью). Эти две сети взаимодействуют и циклично совершенствуются, поскольку дискриминатор старается лучше отделять подделки от оригиналов, а генератор пытается создавать убедительные подделки. Основной проблемой при реализации GAN является соблюдение баланса между ее частями.

По использованию элементов случайности, алгоритмы машинного обучения делятся на детерминированные и стохастические. В *детерминированных* методах шаг за шагом осуществляется процедура коррекции весов сети, основанная на использовании их текущих значений, входов сети, выходов нейронов и некоторой дополнительной информации, например, требуемых значений выходов сети. В основе *стохастических* методов обучения лежит процедура случайных изменений весов нейронов, при этом сохраняются лишь те изменения, которые ведут к улучшениям. Стохастические методы позволяют преодолеть некоторые недостатки, свойственные детерминированным методам обучения («застревание» в локальных экстремумах целевой функции), однако они значительно медленнее.

В общем случае при обучении многослойных ИНС наилучшие результаты достигаются применением стохастических методов совместно с детерминированными алгоритмами локальной оптимизации. Это позволяет добиться достаточной точности подбора весов сети за приемлемое время обучения.

На этапе распознавания на вход ИНС поступает заранее неизвестный входной вектор. При этом выходной вектор является результатом распознавания или решения другой задачи, для которой предназначена и обучена данная ИНС.

Реализацию и обучение ИНС рекомендуется производить с использованием готовых библиотек или специализированных инструментальных средств. Из коммерческих продуктов наибольшую известность получили:

— пакеты расширения MatLaB: Neural Network Toolbox (NNTool) и Neuro Solutions;

— STATISTICA Automated Neural Networks (SANN) от компании StatSoft (statsoft.ru).

Доступны многочисленные свободно распространяемые библиотеки на языке Python и других высокоуровневых языках программирования:

— TensorFlow (tensorflow.org) от Google — поддерживается Python, Java, C++ и C, доступны версии для JavaScript и Android;

— Keras (keras.io) — высокоуровневый API, написанный на Python и способный работать поверх библиотек TensorFlow (используется по умолчанию), Microsoft CNTK или Theano — поддерживается Python;

— Caffe (caffe.berkeleyvision.org) от исследовательской лаборатории Berkeley AI Research (BAIR) — поддерживается Python, C++ и C;

— Microsoft CNTK (https://github.com/Microsoft/CNTK) — поддерживается Python и C++;

— OpenCV (opencv.org) — поддерживается C++, C, Java, Python;

— PyTorch (pytorch.org) от компании Facebook — поддерживается C++, C и Python;

— Dlib (dlib.net) — поддерживается C++ и Python, и др.

Рассмотрим далее некоторые специальные виды ИНС.

Персептрон

Многослойная однонаправленная сеть (сеть прямого распространения сигнала) или *персептрон* (рис. 33) была разработана в 1958 г. и стала первой ИНС.

Характерными особенностями персептрона являются:

— наличие одного или нескольких слоев (помимо входного, который служит только для ввода значений рассматриваемых переменных);

— отсутствие обратных связей;

— наличие связей между нейронами только соседних слоев.

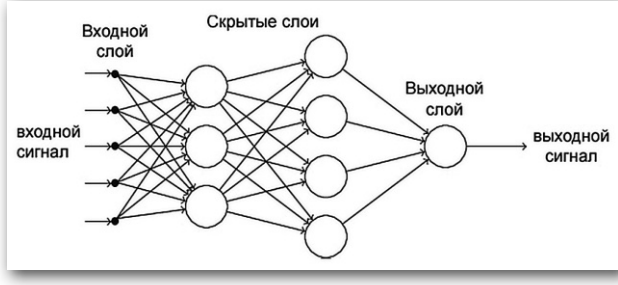

Рис. 33. Трехслойный полносвязный персептрон
с пятью входами и одним выходом

Для однослойных и двухслойных сетей существуют не решаемые задачи, причем число таких задач растет по мере увеличения размерности входа. И только сети, содержащие не менее трех слоев, свободны от подобного недостатка.

Персептрон может научиться всему, что он способен представлять. Понятие представляемости относится к способности сети моделировать определенную функцию. Обучаемость же требует наличия систематической процедуры настройки весов сети для реализации этой функции.

Персептрон обучают, подавая множество образов по одному на его вход и подстраивая веса до тех пор, пока для всех образов не будет достигнут требуемый выход (с заданной точностью). В многослойной сети правильные (эталонные) значения имеются только для нейронов выходного слоя. Правильные значения выходов нейронов скрытых слоев неизвестны. Поэтому для обучения многослойного персептрона используют метод *обратного распространения ошибки*, основной задачей которого является получение оценки ошибок для нейронов скрытых слоев. Свое название метод получил потому, что при обучении сети ошибка распространяется от выходного слоя к входному, то есть в направлении, противоположном направлению распространения сигнала.

Скорость обучения сети с помощью алгоритма обратного распространения ошибки во многом определяется видом функции ошибки. Теории выбора оптимального вида функции ошибки не существует. Обычно в качестве ошибки используют среднеквадратическую ошибку (СКО) выходного слоя $E = \frac{1}{2}\sum_{j=1}^{n}\left(z_j - y_j\right)^2$, где y_j —

выход j-го нейрона выходного (K-го) слоя при предъявлении очередного входного вектора, z_j — эталонное (требуемое) значение выхода этого нейрона при предъявлении данного входного вектора, n — число нейронов K-го слоя. Иногда ошибку рассчитывают по другому критерию. Подстройка синаптических весов осуществляется методом градиентного спуска.

Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других патологических случаев. Например, если всем весам придать одинаковые начальные значения, а для требуемого функционирования нужны неравные значения, то сеть не сможет обучиться.

Метод обратного распространения ошибки реализует обучение с учителем и может использоваться как для обучения многослойных сетей прямого распространения, так и для сетей с обратными связями. Доказано, что метод способен научить сеть аппроксимировать любую функцию с любой точностью. Метод обратного распространения ошибок является модификацией классического метода градиентного спуска и требует, чтобы функция активации была дифференцируемой (что затрудняет использование двоичных сигналов). Кроме того, он работает достаточно медленно. Алгоритм обратного распространения ошибки предполагает выполнение для каждой обучающей пары следующих шагов:

1. Выбрать очередную обучающую пару (x, z). Подать входной вектор на вход сети.

2. Вычислить выход сети y.

3. Вычислить величину ошибки между выходом y и эталонным выходом z.

4. Подкорректировать синаптические веса сети для выходного, K-го слоя и для остальных слоев в последовательности от $k = K–1$ до $k = 2$.

Для обучения могут использоваться и более быстрые и надежные алгоритмы оптимизации: алгоритм переменной метрики (BFGS-метод), метод Левенберга-Марквардта, метод сопряженных градиентов, которые могут быть, однако, весьма затратными с точки зрения памяти. Из стохастических методов обычно отдают предпочтение методу Коши.

Основными достоинствами многослойного персептрона (MLP) являются его простота, хорошая изученность и теоретическая разработанность, а также гарантированное получение ответа после прохода данных по слоям. Такие сети хорошо работают в задачах распознавания объектов из заранее заданных классов, в то же время, персептрон не может решать задачи чистого обобщения, то есть классифицировать объект с принципиально новыми, отличными от запомненных сетью признаками. То есть обученный персептрон не может «дообучиться» или «переобучиться» чему-то новому.

Сверточные сети

Сеть свертки (convolutional neural network, CNN) представляет собой многослойную нейронную сеть прямого распространения, специально созданную для распознавания двумерных поверхностей с высокой степенью инвариантности к преобразованиям, масштабированию, искажениям и прочим видам деформации. Сверточные сети используются преимущественно для распознавания изображений, хотя могут применяться также для работы с любыми двумерными наборами данных, которые можно представить в виде матриц (например, аудио или видео).

Архитектура сети CNN основана на чередовании слоев двух видов: слоев свертки и вычислительных слоев. Первый слой в CNN всегда сверточный. В заключении используют полносвязный слой, выполняющий роль классификатора, то есть определяющий вероятности принадлежности анализируемого образа к различным классам.

Каждый нейрон *слоя свертки* получает входной сигнал от локального рецептивного поля (поля восприятия), представляющего собой несколько смежных нейронов предыдущего слоя (рис. 34), извлекая таким образом его локальные признаки. Нейрон, получающий информацию, следит не за всем входным пространством сигналов, а только за его частью, такая область и называется его *рецептивным полем*. Для следующих нейронов сверточного слоя рецептивное поле будет сдвигаться, пробегая предыдущий слой во всех возможных позициях. Это позволяет найти характерный образ (признак) в любом месте, где бы он ни появился.

Вычислительный слой сети состоит из набора *карт признаков*, каждая из которых имеет форму плоскости (или матрицу), на которой все нейроны должны совместно использовать одно и то же множество синаптических весов (рис. 35).

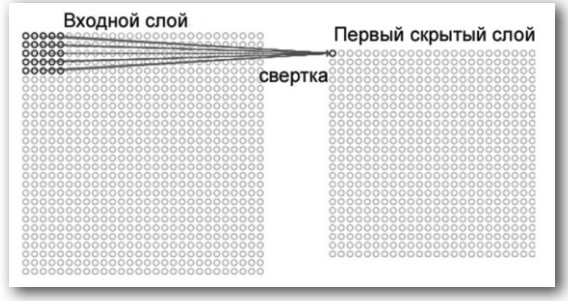

Рис. 34. Действие слоя свертки с указанием
рецептивного поля нейрона размером 5×5

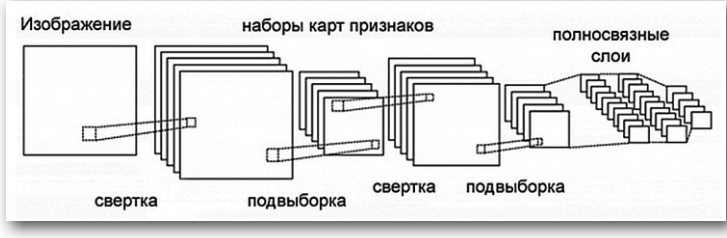

Рис. 35. Обобщенная архитектура сверточной сети

За каждым слоем свертки следует вычислительный слой, осуществляющий подвыборку и локальное усреднение или вычисление максимума, что позволяет снизить размерность карты признаков, отбросив излишние подробности, если признак уже выделен (рис. 36). Этот прием позволяет снизить чувствительность выходного сигнала к различного рода искажениям.

2	0	2	1
5	7	6	4
3	1	0	1
2	2	4	5

→

7	6
3	5

Рис. 36. Пример действия вычислительного слоя
с подвыборкой 2×2 и шагом 2 и функцией максимума

Таким образом, прохождение пары слоев свертка-вычисление позволяет одновременно уменьшить количество хранимой в памяти информации и выделить опорные признаки изображения, такие как ребра, контуры или грани. На следующем уровне обработки из этих ребер и граней можно распознать повторяемые фрагменты текстур, которые дальше могут сложиться в фрагменты изображения.

Самое существенное отличие сверточной сети от полносвязного многослойного персептрона — значительно меньшее количество используемых весов при том же количестве нейронов.

Работа CNN обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к еще более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное.

Для обучения сверточной сети могут применяться как алгоритмы с учителем на маркированных данных (например, метод обратного распространения ошибки), так и алгоритмы обучения без учителя. Строение CNN позволяют выполнять операций внутри одного слоя параллельно, что обуславливает высокую эффективность их аппаратных реализаций, например, на графических процессорах.

Сверточные сети считаются в настоящее время одним из лучших инструментов распознавания и классификации изображений. Распознавание объектов на фото и видео с помощью CNN применяется в беспилотном транспорте, в системах видеонаблюдения, системах контроля доступа, системах «умного дома» и т. д. Кроме того, CNN используются для распознавания речи, обработки аудиосигналов, обработки временных рядов, для анализа смысла текстов.

Описано использование CNN в задачах распознавания рукописных символов и лиц в системах биометрической аутентификации. В работе [25] рекомендовано использование глубоких сверточных сетей для преобразования изображения лица человека в криптографический ключ пользователя, что позволяет повысить защищенность системы аутентификации.

Вместе с тем, существует критика сверточных сетей, основанная на том, что CNN жестко зависят от типа задачи — для распознавания образов каждого типа (голос, подпись, клавиатурный почерк) придется использовать свою архитектуру сверточной сети.

Кроме того, целенаправленное внесение определенных изменений (зачастую незаметных для человеческого глаза) в распознаваемые изображения позволяет «обмануть» нейронную сеть так, чтобы получить нужный результат классификации. Эти изменения специфичны для конкретных нейросетевых реализаций.

Самоорганизующиеся сети Кохонена

Самоорганизующиеся сети могут применяться для широкого круга задач преобразования данных, однако, наиболее эффективным является их использование для визуализации многомерных данных в пространстве меньшей размерности и кластеризации данных.

Задача кластеризации сводится к выделению групп (кластеров) из имеющегося набора объектов таким образом, чтобы объекты из одного кластера имели больше сходства друг с другом, чем с объектами из других кластеров. При этом должен задаваться какой-либо критерий или мера сходства. Для решения задачи кластеризации используется обучение без учителя.

Самоорганизующаяся сеть, руководствуясь определенными правилами функционирования, сама определяет некоторую меру сходства между предъявляемыми ей данными и, на основании этого сходства, относит их к той или иной группе.

В 1982 году финский ученый Тойво Кохонен (Kohonen) предложил специальную конфигурацию нейронной сети с обучением без учителя для решения задач кластеризации данных. При этом не требуется ни знать семантику обрабатываемых данных, ни высказывать предположений или гипотез о том, как эти данные структурированы. Достаточно задать только число классов, на которые нужно разбить данные.

При этом выделяют сети с *неупорядоченными нейронами* (*слои Кохонена*) и сети *с упорядочением нейронов* (*самоорганизующиеся карты*, self-organizing map, SOM).

Сеть (слой) Кохонена — однослойная сеть, каждый нейрон которой соединен со всеми компонентами входного вектора, задающего описание объекта, подлежащего кластеризации. Количество нейронов в этой сети совпадает с количеством кластеров, которое должна выделить сеть. В качестве нейронов сети Кохонена применяются линейные взвешенные сумматоры.

Формирование самоорганизующихся сетей начинается с инициализации синаптических весов, которым обычно присваиваются малые

случайные значения. Для обучения сети применяются *механизмы конкуренции*, основанные на принципе «победителю достается все». При подаче на вход сети вектора данных X побеждает тот нейрон, вектор весов W которого в наименьшей степени отличается от входного вектора (с учетом заданной меры близости). Этот нейрон имеет единичный выходной сигнал, остальные — нулевые. Если условие активации одновременно достигается для нескольких нейронов, то либо все они имеют выход, равный единице, либо только первый в списке (по соглашению).

Чаще всего в качестве меры близости используется евклидово расстояние $d(X,W)=\sqrt{\sum_i (x_i - w_i)^2}$, но возможно применение и иных метрик.

Вокруг нейрона-победителя образуется *радиус обучения*, определяющий, сколько еще нейронов, кроме нейрона-победителя, участвуют в обучении (т. е. корректируют свои веса) на данной итерации. Радиус в данном случае рассчитывается как расстояние в пространстве векторов весов нейронов. Радиус обучения максимален на первой итерации и уменьшается с увеличением числа итераций таким образом, что в конце обучения корректирует свои веса только нейрон-победитель. Веса нейронов, находящихся за пределами радиуса обучения, не изменяются.

Как правило, обучение состоит из двух основных этапов: на первом производится подстройка весов для всех нейронов окрестности нейрона-победителя с постепенным уменьшением этой окрестности. На втором этапе производится точная коррекция весов малыми значениями для обеспечения сходимости сети.

Одной из особенностей обучения с конкуренцией заключается в том, что некоторые нейроны оказываются незадействованными. Нейроны, имеющие начальные веса, значительно удаленные от входных векторов объектов обучающей выборки, никогда не выигрывают конкуренции, независимо от того, как долго продолжается обучение. Такие нейроны называют «мертвыми», поскольку они не выполняют никакой полезной функции. При наличии в сети «мертвых» нейронов входные данные будут интерпретироваться меньшим числом нейронов, а погрешность квантования увеличится. Поэтому необходимо дать шанс победить всем нейронам. Для этого алгоритм обучения модифицируют таким образом, чтобы нейрон-победитель со временем терял активность.

Карты Кохонена (*самоорганизующиеся карты признаков*) предназначены для визуального представления на двумерной карте многомерных объектов с близкими свойствами. Карта Кохонена состоит из ячеек прямоугольной или шестиугольной формы (рис. 37), каждой ячейке соответствует нейрон сети Кохонена.

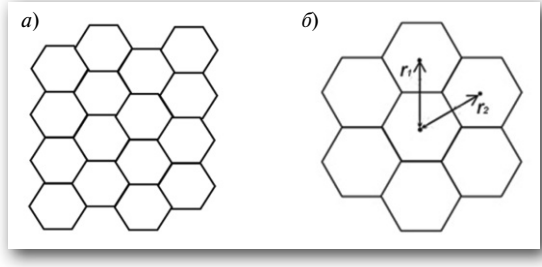

Рис. 37. Карта Кохонена: а) сеть с шестиугольными ячейками, б) расстояния между соседними шестиугольными ячейками

Объекты, векторы признаков которых близки, попадают в одну ячейку или в соседние ячейки. Таким образом, двумерная карта Кохонена отражает на плоскости близость многомерных векторов признаков. Шестиугольные ячейки более корректно отображают расстояние между объектами на карте, так как для этих ячеек расстояние между центрами смежных ячеек одинаковы.

Для облегчения визуального анализа используется раскраска карт Кохонена. Если требуется провести анализ сходства объектов по каким-то конкретным параметрам, то для каждого параметра строится своя карта Кохонена. Ячейки карты раскрашиваются в разные цвета (или оттенки серого цвета) в зависимости от значения параметров, соответствующих каждой ячейке. Поскольку в каждую ячейку в общем случае попадает несколько объектов, то значение параметра для объектов в ячейке усредняется (как варианты возможно вычисление минимального или максимального значения). Если же ячейка пуста («мертвый» нейрон), то в качестве значения берется вес нейрона, соответствующий рассматриваемому параметру.

Обучение карт Кохонена производится таким же образом, как и обучение слоя Кохонена. Скорость самообучения карт Кохонена

существенно превышает скорость персептрона при той же размерности входных данных.

После обучения сети активация победившего нейрона свидетельствует о принадлежности объекта соответствующему кластеру. Таким образом, сеть способна к классификации неизвестных образов на основе их сходства с образами, предъявляемыми в процессе обучения. Таким образом, имея перед собой карту и зная информацию о некоторой из части исследуемых объектов, можно достаточно достоверно судить о свойствах новых объектах. Если после обучения карта встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его аномальность.

Традиционно метод Кохонена рассматривается как эмпирический алгоритм, сходимость которого не может быть строго математически обоснована, а выводы о структуре данных делаются на основе визуального анализа карт. Основным ограничением сети Кохонена является необходимость знать заранее число кластеров. Кроме того, окончательный результат работы сети Кохонена зависит от ее начальной случайной инициализации. Поэтому использование сетей Кохонена, как правило, рекомендуется только для разведочного анализа данных.

Карты Кохонена нашли широкое применение в геоинформационных системах, поскольку для этих систем легко реализовать информационную цепочку от исходной таблицы к узлам решетки, а от них — к конкретным координатам на географической карте.

Примеры нейросетевых моделей в системах защиты информации

В настоящее время существует ряд задач ИБ, эффективное решение которых требует применения нейросетевых моделей или средств интеллектуального анализа данных. Это задачи, связанные с обработкой множества сигналов и больших массивов данных в разных представлениях и форматах, которые что делает их анализ в «ручном режиме» практически неосуществимым или слишком медленным. Это затрагивает системы сетевой безопасности, проактивного обнаружения скомпрометированных узлов и системы сбора и анализа событий безопасности.

Многослойный персептрон используется в задачах классификации, там, где может быть проведено предварительное обучение сети с учителем. Примером являются биометрические системы аутентификации.

При регистрации нового пользователя ему ставится в соответствие отдельный класс, а затем происходит обучение системы, для чего пользователь неоднократно предъявляет свои биометрические характеристики, в качестве принадлежащих этому классу объектов. После этого система может работать в режиме распознавания.

В зависимости от типа анализируемого биометрического признака, в системах аутентификации могут быть использованы и другие виды ИНС, например, сверточные сети для анализа трехмерного цифрового изображения лица.

Многослойный персептрон может быть использован и в различных системах обнаружения угроз ИБ. При этом возможны два подхода [17]:

— дополнение существующих экспертных правил нейронной сетью для настройки параметров правил и фильтрации поступающих сообщений с целью снижения числа ложных срабатываний,

— самостоятельное использование ИНС для выявления признаков злоупотреблений.

Как правило, при первом подходе для этих целей используют нейро-нечеткие системы, которые позволяют явным образом отразить в структуре ИНС опыт экспертов, формализованный в виде системы нечетких предикатных правил, параметры которых автоматически корректируются в процессе обучения сети. Примеры таких реализаций приведены, в частности, в [2, 10, 11].

Применительно к анализу сетевых пакетов эти правила могут включать определенные значения отдельных полей заголовка пакета (IP-адрес и порт источника или получателя, установленные флаги, размер пакета и т. д.). При анализе журналов регистрации событий правила могут ограничивать время регистрации пользователя в системе, количество попыток неправильного ввода пароля в течение короткого промежутка времени, а также наличие изменений в критических файлах системы.

Например, набор нечетких правил для обнаружения DDos атак типа SYN Food, приведенный в конце параграфа 2.1, может быть отражен в структуре многомерного персептрона, как это представлено на рис. 38 [2].

Описанный таким образом нейронечеткий классификатор задается четырехслойным персептроном. Здесь входной слой выполняет лишь распределительные функции, подавая на вход первого скрытого слоя нейронов вычисленные значения параметров трафика.

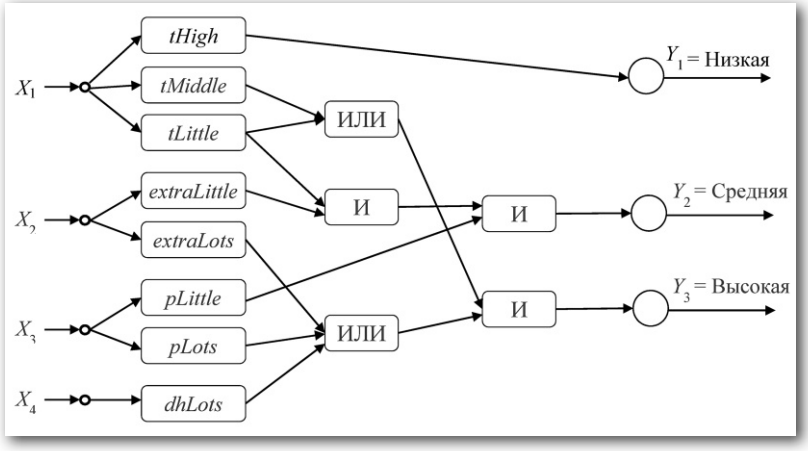

Рис. 38. Пример задания нейронечеткого классификатора

Первый скрытый слой нейронов выполняет фаззификацию переменных, распределяя входные значения на основе функций принадлежности $m(X_i)$ по нечетким множествам:

tHigh — «среднее время поступления одного пакета велико»;

tMiddle — «среднее время поступления одного пакета среднее»;

tLittle — «среднее время поступления одного пакета мало»;

extraLittle — «процент пакетов с различными внешними IP-адресами мал»;

extraLots — «процент пакетов с различными внешними IP-адресами велик»;

pLittle — «процент пакетов с различными внешними портами мал»;

pLots — «процент пакетов с различными внешними портами велик»;

dhLots — «процент пакетов с некорректными заголовками велик».

Следующие два скрытых слоя реализуют логику нечеткого правила, а функция активации формируется по правилам выполнения операций алгебраического сложения и умножения нечетких множеств. Выходной слой состоит из нейронов с обычной сигмоидной функцией активации и формирует значение выходной переменной Y — степени уверенности в атаке. Обучение сети производится методом обратного распространения ошибки.

Рассмотренный пример нацелен на обнаружение одного вида атак, однако основным достоинством ИНС является возможность использования одной и той же сети для детектирования атак разного типа. В этом случае речь идет о самостоятельном использовании ИНС. Любое действие пользователя или приложения должно быть представлено в виде вектора признаков, которые подаются на вход ИНС. Экспертные знания могут быть полезны при выделении состава значимых признаков, формирующих вектор входных данных. Все нейроны сети имеют стандартные (например, сигмоидные) функции активации, а настройка паттернов (шаблонов) угроз происходит за счет изменения синаптических весов в ходе обучения сети.

Например, в [10, 11] в качестве основы системы обнаружения вторжений использован четырехслойный персептрон с 8 нейронами во входном и с 2 нейронами в выходном слоях. На вход сети подается следующие 8 характеристик сетевых соединений: ID протокола, номер порта (хост), номер порта (гость), IP адрес (хост), IP адрес (гость), протокол ICMP, объем передаваемых данных в байтах, TCP-флаги. Выход классифицирует два состояния: нормальное функционирование и реализация атаки. Обучение сети с известными атаками производилось на основе данных анализа сетевого трафика с помощью шаблонов IDS Bro.

Основной проблемой при использовании сетей, обучающихся на данных об известных атаках, является способ поддержания актуальности информации о киберугрозах, который заключается в необходимости добавления новых правил, описывающих шаблоны угроз, и дообучения сети. Для возможности учета известных угроз, не отраженных экспертными правилами, сеть должна иметь способность к кластеризации.

В [2] для этих целей предлагается использовать сеть встречного распространения, объединяющую слой Кохонена и слой Гроссберга (рис. 39). Эти слои обучаются отдельно. Сначала производится обучение слоя Кохонена без учителя, в результате чего производится кластеризация входных данных с характеристиками сетевого трафика. На втором шаге производится обучение с учителем для слоя Гроссберга.

Слой Гроссберга содержит нейроны, функция активации которых представляет собой взвешенную сумму выходов слоя Кохонена. Поскольку лишь один нейрон в слое Кохонена будет давать ненулевой выход, то в процессе обучения будут изменяться только те веса нейронов слоя Гроссберга, которые связаны с этим нейроном.

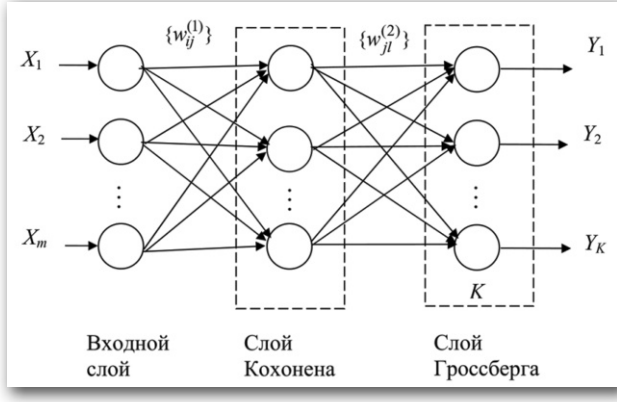

Рис. 39. Сеть встречного распространения

После обучения каждый нейрон слоя Гроссберга фактически выдает величину веса, который связывает этот нейрон с единственным ненулевым нейроном слоя Кохонена. Слой Гроссберга нужен, чтобы привести полученные в результате самообучения слоя Кохонена кластеры к известным классам атак.

Высокая обобщающая способность сети встречного распространения позволяет правильно детектировать атаки даже в случае неполных или искаженных входных данных.

В [21] предлагается использовать три различные ИНС: многослойный персептрон (MLP), сеть радикально-базисных функций (RBF — сеть прямого распространения с особым видом функции активации нейрона) карты Кохонена (SOM), а затем сопоставлять полученные результаты для повышения точности детектирования сетевых атак и снижения ложных срабатываний. В качестве входных данных предлагается использовать формат NLS-KDD, представляющий стандартный набор данных IDS: длительность соединения, протокол соединения, тип подключения, служба и порт назначения, флаг успешного подключения, количество неудачных попыток авторизации, количество запросов на подключение с правами администратора и т. д.

Основная специфика задач обнаружения разного рода угроз ИБ заключается в том, что обучение на старых данных не эффективно, поскольку уже известные атаки, которые системы защиты информации

(СЗИ) умеют отражать, уже не опасны. Для обнаружения таких атак могут быть использованы более простые и менее затратные способы, например, сигнатурный анализ. В то же время наибольшую проблему представляет выявление принципиально новых атак, не только не имеющие сигнатур, но и с ранее неизвестными сценариями реализации, против которых СЗИ, использующие известные паттерны атак, бессильны. При этом время, в течение которого функциональные возможности СЗИ могут оставаться неизменными, постоянно сокращается (от нескольких месяцев еще несколько лет назад до нескольких часов сегодня).

Поэтому все большее внимание уделяется обнаружению аномальной активности, то есть не сопоставлению с паттерном атаки, а выявление отклонений от паттернов нормального функционирования системы.

Например, при анализе работы с веб-приложением каждая страница сайта имеет свой пользовательский сценарий. Так, для интернет-магазина обычными пользовательскими действиями будет выбор товара, сохранение его в корзине, просмотр корзины, добавление и удаление из нее товаров, изменение их количества, выход без заказа, попытки оплаты, ввод адреса и времени доставки и т. п. Точно также имеется типичный трафик к сайту, а для каждого поля — типичный вводимый тип данных. Отклонение запросов от типичных значений по любому из таких параметров как: тип данных, частота, источник запросов — служит поводом для подозрений. Например, если типичными пользовательскими запросами являются `GET /?p = 1`, `GET /?p = 5`, `GET /?p = 9`, `GET /?p = 254`, то запрос `GET /?p = ' OR 1 = 1 --` (попытка SQL-инъекции) будет расценен как аномальный.

Построение поведенческих моделей может осуществляется только на основе собранных в течение определенного временного промежутка обращений к веб-приложению (проверка на аномальность), либо с использованием дополнительно базы данных атак (проверка на аномальность и наличие злоупотреблений).

Другой пример — выявление обращений к серверу базы данных со стороны нелегитимного ПО, когда политикой безопасности предписана работа только с помощью типового клиентского рабочего места. На основе рекомендаций экспертов в качестве входных данных сети были выделены следующие признаки:

— объем информации, загружаемой из базы данных за контрольный период (полученное значение необходимо нормализовать, поскольку считываемый из базы данных объем заранее не известен и индивидуален для каждой задачи и для каждого пользователя; в качестве нормализации можно применить оценку трафика по десятибалльной шкале);
— количество транзакций в минуту;
— количество операций модификации данных в минуту (в приведенном примере стандартное ПО использует «короткие транзакции», то есть в рамках одной транзакции обычно бывает 1–2 операции модификации данных);
— признаки обращения к словарю базы данных (большинство клиентских рабочих мест к словарю не обращаются, что отличает их от средств разработки и администрирования; признаки будут формироваться для каждой из таблиц словаря базы).

Выходной слой ИНС содержит один нейрон, классифицирующий обращение как аномальное.

Если выделение влияющих признаков представляет проблему, а размерность входных данных велика, то на предварительном этапе может быть применен метод главных компонент или реализующая его рециркуляционная нейронная сеть (сеть с «бутылочным горлышком») [2, 8], позволяющая выделить заранее заданное число наиболее значимых признаков в произвольном наборе данных.

При всей действенности подхода на основе выявления аномалий, существует несколько проблем, связанных с применением нейросетевых моделей в задачах такого рода.

Первая из них связана с качеством данных, используемых для обучения. Если обучение ведется на неразмеченных «живых» данных, полученных в ходе непосредственной работы с приложениями (что чаще всего и происходит на практике), нужно быть уверенным, что в собранных данных изначально отсутствует аномальная активность, в противном случае ИНС будет в дальнейшем диагностировать ее как нормальную и, следовательно, не обнаружит.

Вторая проблема связана с непрозрачностью принятия решений ИНС, которая не имеет встроенных механизмов для объяснения, почему та или иная активность является аномальной. В ряде случаев это существенно затрудняет интерпретацию результатов работы ИНС. Зачастую аналитику требуется немало времени, чтобы понять, действи-

тельно ли выявленная активность связана с вредоносными действиями нарушителей или же явилась следствием каких-то других факторов, например, изменений в самом приложении, пересмотра практик работы или политик безопасности, или привлечения новых сотрудников.

Высокая скорость обновления ландшафта угроз ИБ порождает спрос на системы, постоянно находящиеся в режиме обучения, при условии сохранения СЗИ возможности обнаруживать и отражать атаки.

2.3. Интеллектуальный анализ данных

Сущность, задачи и основные этапы ИАД

Интеллектуальный анализ данных (ИАД) охватывает всю область проблем, связанных с извлечением знаний из массивов данных. В области ИБ это задачи, связанные с анализом сетевого трафика для выявления атак и аномального поведения, обнаружением СПАМ-сообщений в почтовой рассылке и мошеннических действий (fraud), выявлением корреляции событий на основе разнородных данных (данные подсистем аудита сети и операционной системы, файлы журналов файловых систем, СЗИ и приложений) в системах мониторинга событий и при расследованиях инцидентов ИБ. Это связано с тем, что электронная почта, финансовые транзакции, сканирование сетей, DoS-атаки представляют собой многократные повторения сетевых операций, выявление которых можно автоматизировать.

Часто ИАД рассматривают как трансформацию термина Data Mining, однако методы Data Mining сосредотачиваются главным образом на процессе анализа данных и интеллектуальном моделировании, в то время как ИАД охватывает весь спектр проблем, связанный с процессом получения знаний из данных, включая вопросы извлечения данных из различных источников, их консолидацию, профайлинг, трансформацию, очистку и обогащение.

Современная концепция ИАД предполагает, что:

— данные могут быть разнородными, неточными, неполными (содержать пропуски), противоречивыми, косвенными;
— данные могут иметь очень большие объемы (big data);
— используются интеллектуальные алгоритмы анализа данных, в частности, способные к обучению на основе прецедентов, обладающие обобщающей способностью (то есть позволяющие делать общие выводы на основе частных наблюдений);

— процессы переработки сырых данных в информацию, а информации в знания требуют нетривиальной автоматизации.

Выделяют следующие типовые этапы, сопровождающие решение задач интеллектуального анализа данных [9]:

1. Анализ предметной области, формулировка целей и задач исследования.

2. Извлечение и сохранение данных.

3. Предварительная обработка данных:

— очистка — исключение дубликатов и фиктивных значений, обработка противоречий, случайных выбросов и помех (сглаживание, подавление шума), восстановление и заполнение пропусков;

— консолидация — объединение данных из нескольких возможных источников в одном хранилище, обеспечение необходимого уровня их информативности и качества, преобразование к единому формату;

— профайлинг — анализ информации о структуре данных (тип поля, длина его значений и их допустимый диапазон, анализ шаблонов), проверка полей источника данных на соответствие заданным ограничениям и исправление обнаруженных проблем в соответствии с заданным сценарием (например, проверка форматов представления и способов записи числовых данных и приведение их в соответствие с принятым синтаксисом);

— оптимизация данных — сокращение размерности (отбор значимых признаков);

— обогащение данных — привлечение дополнительной информации из внешних источников или получение ее с помощью дополнительных действий с имеющимися данными (например, подсчет средних значений или математических ожиданий), с целью повышения ценности и значимости входных данных с точки зрения решения конкретной задачи анализа;

— преобразование (трансформация) — оптимизации представлений и форматов данных с точки зрения решаемых задач и целей анализа, может включать агрегирование и сжатие данных, их нормализацию (приведение входных значений к некоторому заданному диапазону), дискретизацию атрибутов и т. п.

4. Содержательный анализ данных — установление общих закономерностей или решение более конкретных, частных задач анализа:

— классификация — отнесение объектов к какому-либо из заранее заданных классов (групп, типов);
— аппроксимация и предсказание непрерывных значений признака — получение в качестве результата неизвестного значения выходного признака на основе анализа известных данных;
— кластеризация — выделение групп объектов по признаку сходства, анализ и прогнозирование общих черт объектов в кластере;
— анализ ассоциаций — поиск взаимозависимостей или корреляционных связей;
— выявление последовательностей (событий, связанных друг с другом во времени), прогнозирование следующего возможного события;
— прогнозирование — поиск существующих временных рядов и прогнозирование динамики значений в них на будущие периоды времени.

5. Представление результатов анализа в удобной для восприятия и интерпретации форме (визуализация и отбор полезных паттернов, формирование информативных графиков и таблиц), интерпретация результатов.

6. Использование новых знаний для принятия решений.

Таким образом, сначала обрабатываются и подготавливаются данные. Потом составляется краткий перечень соответствующих целям и задачам исследования алгоритмов. Затем производятся выбор и настройка параметров этих алгоритмов. И наконец, строятся аналитические модели для выбора лучшей из них. Полученные результаты представляются в доступной форме и интерпретируются в соответствии с целями анализа.

Подготовка данных для анализа

Главную роль в ИАД играют сами данные. Если качество данных низкое, то даже самый изощренный анализ не сможет дать приемлемый результат.

Обычно для задач анализа используют табличное представление данных. Каждая строка представляет собой элемент данных с описанием отдельного наблюдения, а каждый столбец несет переменную для его описания. Переменные также называются атрибутами, признаками или размерностями.

Существуют четыре основных типа переменных (таблица 3). Каждый тип обрабатываемых данных предполагает применение определенных методов обработки и анализа, поэтому важно убедиться, что к входным данным применимы выбранные алгоритмы.

Таблица 3

Основные типы переменных, используемых в ИАД

Тип переменной	Описание
Бинарная (двоичная)	Простейший тип переменных только с двумя вариантами значения
Категориальная (качественная или порядковая)	Если вариантов больше двух, информация может быть представлена категориальной переменной. Этот тип не является числовым, однако на его значениях может быть задано отношение порядка. В этом случае говорят о порядковых величинах
Целочисленная (количественная)	Такой тип используется, когда информация может быть представлена целым числом
Непрерывная (количественная)	Наиболее информативный тип переменной, значения представимы в виде вещественных чисел

В качестве примера рассмотрим перечень бинарных переменных для анализа сетевого трафика протокола Telnet [24]:

— признак зашифрованных данных;
— неудачная попытка регистрации;
— успешная попытка регистрации;
— попытка получения прав доступа суперпользователя root;
— факт получения прав доступа суперпользователя root;
— признак гостевой учетной записи;
— выполненная операция создания файла.

Каждый из перечисленных признаков может принимать лишь два значения — да/нет (True/False, 1/0).

Хотя в первоначальном наборе данных может быть много разных переменных, применение в алгоритме слишком большого их числа ведет к замедлению вычислений или к ошибочным предсказаниям из-за информационного шума. Поэтому необходимо сформировать

короткий список самых важных переменных, вносящих наибольший вклад в результат. Выбор переменных часто делается методом проб и ошибок либо с помощью методов *сокращения размерности* (dimension reduction). Их имеет смысл добавлять и убирать, учитывая промежуточные результаты. Для начала можно использовать простые графики для выявления корреляций между переменными, отбирая самые многообещающие для дальнейшего анализа.

В процессе ИАД часто приходится сталкиваться с неполными данными. Неполные данные могут существенно затруднить анализ, сделав невозможность применения ряда аналитических методов, поэтому при любой возможности следует решить вопрос их дальнейшего использования с применением одним из следующих способов.

Приближение. Если пропущено значение переменной бинарного или категориального типа, его можно заменить самым типичным значением (модой) переменной. Для целочисленных или непрерывных переменных используется медиана.

Вычисление. Пропущенные значения могут быть вычислены на основе более продвинутых алгоритмов обучения с учителем. Хотя такие вычисления требуют времени, они обычно приводят к более точным оценкам неполных значений. Причина в том, что вместо приближения к наиболее распространенному значению в целом, для оценки значения отбираются только сходные с оцениваемой записи.

Удаление. В качестве крайней меры, при невозможности использования других средств, строки с неполными значениями могут быть удалены. Тем не менее этого обычно избегают, чтобы не уменьшать объем данных, доступных для анализа. Более того, исключение элементов данных может привести к искаженным результатам в отношении отдельных (в особенности, немногочисленных) групп.

После того как набор входных данных обработан, можно приступать к его анализу.

Алгоритмы ИАД

Выбор алгоритма анализа данных зависит от задачи, которую хотим решить, а она, в свою очередь, определяет возможность применения того или иного подхода к обучению в рамках ИАД. Для каждого их таких подходов существует набор применимых алгоритмов (таблица 4).

Таблица 4

Основные алгоритмы ИАД

Тип обучения	Алгоритмы
Обучение без учителя	Метод *k*-средних Метод главных компонент Ассоциативные правила Анализ социальных связей
Обучение с учителем	Регрессионный анализ Метод *k*-ближайших соседей Метод опорных векторов Дерево решений Случайные леса Нейронные сети
Обучение с подкреплением	Многорукие бандиты

Задача: найти закономерности в данных.

Метод обучения: без учителя.

Когда требуется выявить скрытые закономерности в наборе данных без каких-либо предварительных знаний об анализируемых данных, можно воспользоваться алгоритмами обучения без учителя. Так называют алгоритмы, используемые тогда, когда не знаем, какие закономерности искать, и предоставляем их поиск самим алгоритмам (таблица 5).

Рассмотрим основные алгоритмы обучения без учителя.

Метод k-средних. Сходные элементы данных объединяются в группы, число которых задается равным *k*. Метод *k*-средних выполняет кластеризацию следующим образом [26].

1. Назначается число групп (*k*), на которые должны быть разбиты данные. Случайно выбирается *k* объектов исходного набора как первоначальные центры кластеров (центроиды).

2. Каждое наблюдение относится к определенной группе по самому близкому центроиду, т. е. на основании критерия минимизации расстояния между ними (обычно используется евклидово расстояние).

3. Пересчитываются координаты центроидов всех кластеров (обычно рассчитываются как средние значения) и вычисляются внутригрупповые разбросы (within-cluster variation).

4. Минимизируется общий внутригрупповой разброс, для чего шаги 2 и 3 повторяются многократно, пока объекты не перестанут

перераспределяться по группам или не будет достигнуто заданное число итераций.

Методом *k*-средних — очень простой и эффективный алгоритм, имеющий, однако, две существенные проблемы. Во-первых, итоговые результаты чувствительны к начальному случайному выбору центров групп. Возможное решение этой проблемы состоит в многократном выполнении алгоритма с различным случайным назначением начальных центроидов и сравнению результатов. Вторая проблема — необходимость априори задавать фиксированное число *k* кластеров для разбиения. Оптимальный выбор числа *k* возможен с использованием методов «локтя» или с помощью статистики разрыва (gap statistic), которая генерируется на основе имитационных моделей Монте-Карло.

Таблица 5

Алгоритмы обучения без учителя

Переменные	Тип значений или результата	Кластеризация методом *k*-средних	Метод главных компонент	Ассоциативные правила	Лувенский метод	PageRank
Вход	Бинарные значения					
	Непрерывные значения					
	Узлы и ребра					
Выход	Категории					
	Ассоциации (связи)					
	Ранги					

Метод главных компонент. Количество переменных для анализа снижается путем комбинирования наиболее информативных из них в новые переменные, называемые главными компонентами. Главные компоненты представляют собой новое множество не коррелирующих между собой признаков, каждый из которых получен как некоторая линейная комбинация исходных влияющих признаков.

В [9] отмечается, что выделение главных компонент может повлечь значительные искажения исходного признакового пространства, что может привести к снижению разделимости для объектов в новом пространстве признаков и снизить итоговое качество классификации.

Ассоциативные правила представляют собой механизм нахождения логических закономерностей между связанными элементами (событиями или объектами). Метод позволяет обнаружить ассоциации среди элементов данных, например, товары, которые часто покупают вместе.

Ассоциативные правила имеют следующий вид: если <условие> то <результат>, где <условие> — не логическое выражение, а набор объектов, с которыми связаны (ассоциированы) объекты, включенные в <результат> данного правила. Кроме правил указанного вида, существуют косвенные ассоциативные правила, ассоциативные правила с отрицанием, временные ассоциативные правила для событий, связанных во времени, и другие.

Существуют три типичные ассоциативные метрики:

— поддержка {X} показывает, как часто появляется X;

— достоверность {X–>Y} показывает, как часто Y появляется в присутствии X;

— лифт {X–>Y} показывает то, как часто X и Y появляются вместе, в сравнении с тем, как часто они появляются по отдельности.

Алгоритмы поиска ассоциативных правил предназначены для нахождения всех правил, связывающих X и Y, причем поддержка и достоверность этих правил должны быть выше некоторых наперед определенных порогов. Алгоритмы поиска ассоциативных правил не тривиальны, одной из проблем является вычислительная сложность при увеличении числа элементов данных.

Лувенский метод. Метод, который идентифицирует кластеры в графе (например, социальной сети) путем максимизации числа внутрикластерных связей и минимизации связей между кластерами. Степень удовлетворения этим двум условиям известна как модулярность (modularity), более высокая модулярность — признак более оптимального разбиения на кластеры.

Метод состоит из двух стадий [27]. На первой происходит поиск «малых» сообществ путем оптимизации модулярности на локальном уровне. На второй стадии узлы одного кластера (сообщества) агрегируются, и строится новая сеть большего масштаба, после чего эти стадии повторяются до тех пор, пока не будет достигнут максимальный

уровень модулярности. Таким образом, после каждого этапа формируются все более крупные кластеры.

Простота и эффективность делают лувенский метод популярным решением для кластеризации сети. Однако он имеет свои ограничения. Итеративный процесс слияния кластеров может привести к тому, что значимые, но небольшие кластеры могут быть поглощены. Определение оптимального разбиения на кластеры может быть затруднено для сетей, содержащих перекрывающиеся или вложенные кластеры.

PageRank. Алгоритм, который определяет доминирующие узлы в сети. Он ранжирует узлы, основываясь на количестве связей, а также на их силе и источнике. Такие узлы определяют кластеры как области высокой концентрации взаимодействий, сформированные вокруг этих узлов.

Метод PageRank использовался Google для ранжирования вебсайтов.

Задача: использовать для прогнозирования заданные шаблоны.

Метод обучения: с учителем.

При прогнозировании не ставится цель вскрыть структуру внутренних взаимосвязей между предикторами или сделать сравнительную оценку силы их влияния на отклик. Основная задача — предсказать значение целевого признака на основании наблюдаемой вариации значений влияющих признаков. Когда требуется прогноз, могут использоваться алгоритмы обучения с учителем, то есть алгоритмы, предсказания которых основаны на уже существующих шаблонах (таблица 6).

Рассмотрим алгоритмы обучения с учителем.

Регрессионный анализ — методы восстановления зависимости между результирующей переменной и предикторами. Находится линия наилучшего соответствия, пролегающая максимально близко к наибольшему числу элементов данных. Такая линия тренда вычисляется на основе взвешенной комбинации влияющих признаков.

Многомерный регрессионный анализ является весьма полезной и широко применяемой моделью для прогнозирования количественных значений выходного признака. Регрессионный анализ позволяет выявлять зависимости между признаками, решает задачи прогнозирования, а также задачи классификации (в случае использования регрессионной зависимости для задания разделяющей плоскости между классами).

Таблица 6

Алгоритмы обучения с учителем

Категория	Основные характеристики	Регрессионный анализ	Метод *k*-ближайших соседей	Метод опорных векторов	Деревья решений	Случайные леса	Нейронные сети
Прогнозирование	Бинарные переменные	✓	✓	✓	✓	✓	✓
	Категориальные переменные		✓		✓	✓	✓
	Возможные классы	✓	✓		✓	✓	✓
	Непрерывные переменные	✓	✓		✓	✓	✓
	Нелинейные отношения		✓	✓	✓	✓	✓
Анализ	Большое число переменных			✓	✓	✓	✓
	Простота использования	✓			✓	✓	
	Высокая скорость вычислений	✓	✓		✓		
Результаты	Высокая точность					✓	✓
	Интерпретируемость	✓	✓		✓		

Выделяют следующий обобщенный многоэтапный подход к регрессионному анализу [9]:

— формулировка задачи — высказываются предварительные гипотезы о зависимости исследуемых явлений;

— определение зависимых и независимых (влияющих) переменных;

— сбор статистических данных — данные должны отражать изменения каждой из переменных, включенных в регрессионную модель;

— формулировка гипотезы о форме связи (простая или множественная, линейная или нелинейная) и составление регрессионного уравнения;

— определение функции регрессии — расчет численных параметров регрессионного уравнения;
— оценка точности регрессионного анализа;
— интерпретация полученных результатов — сравнение полученной зависимости с предварительными гипотезами, оценка корректности и правдоподобия;
— предсказание неизвестных значений зависимой переменной.

Регрессия чувствительна к искажению значений влияющих признаков и практически нечувствительна к искажению значений результирующей переменной.

Метод k-ближайших соседей. Элементы данных классифицируются исходя из близости к соседним элементам. Число ближайших соседей задается равным k.

В основе метода k-ближайших соседей лежит гипотеза компактности, которая предполагает, что тестируемый объект будет иметь такую же метку класса, как и обучающие объекты в локальной области его ближайшего окружения. При этом каждый объект относится к преобладающему классу его ближайших соседей.

При $k = 1$ достигается абсолютно правильное распознавание примеров обучающей выборки (самый ближайший сосед — сам объект), однако часты ошибки на неизвестных данных. При увеличении $k>1$ до некоторых пределов качество распознавания на контрольной выборке будет возрастать. Оптимальное в смысле точности предсказаний значение k может быть найдено с использованием перекрестной проверки. Для этого для различных фиксированных значений k строится модель k-ближайших соседей и оценивается ошибка классификации. Затем выбирается значение k, соответствующее наименьшей ошибке.

Метод опорных векторов. Классифицирует элементы данных в две группы, граница между которыми прокладывается между периферийными элементами данных, то есть опорными векторами обеих групп. Основная идея классификатора на опорных векторах заключается в том, чтобы строить разделяющую поверхность с использованием только небольшого подмножества точек, лежащих в зоне, критической для разделения, тогда как остальные верно классифицируемые наблюдения обучающей выборки вне этой зоны игнорируются (точнее, являются «резервуаром» для оптимизационного алгоритма). Собственно *опорными векторами* называются наблюдения, лежащие непосредственно на границе разделяющей поверхности,

либо на неправильной для своего класса стороне относительно границ зазора.

При работе с изогнутыми границами для учета нелинейности обычно расширяют пространство переменных, включая различные функциональные преобразования исходных предикторов — функции ядра. Каждое ядро характеризуется параметрами, которые подлежат оптимизации. Основная идея использования ядер заключается в том, что при отображении данных в пространство более высокой размерности исходное множество точек может стать линейно разделимым. Решение оптимизационной задачи в расширенных пространствах огромных размерностей оказалось возможным потому, что ядро формируется только для ограниченного набора опорных векторов. Это позволяет строить модели с использованием разделяющих поверхностей самой различной формы.

Поскольку на расположение гиперплоскости оказывают влияние только те наблюдения, которые лежат на границах зазора или нарушают его, то решающее правило такого классификатора довольно устойчиво к выбросам большинства точек, расположенных вне «критической зоны» разделения. Это свойство выгодно отличает метод опорных векторов от других классификаторов.

Дерево решений. Метод, который строит прогноз путем формирования последовательности бинарных вопросов, постепенно разбивающих элементы данных на однородные группы.

Дерево решений имеет одну вершину (корень), а завершается вершинами, из которых не исходит ни одна дуга, — листьями. При этом принято, что дерево решений растет вниз (а не вверх, как настоящее дерево). Дерево решений представлено связным ациклическим графом, имеющим различные метки:

— узлы (вершины, не являющиеся листьями) — переменные набора данных;
— на дугах (ветвях) отмечают атрибуты (значения переменных), от которых зависит целевая функция;
— в листьях отмечают значения целевой функции.

В общем случае для решения задачи классификации необходимо спуститься по дереву от вершины до листа, выполняя соответствующие действия в узлах и выбирая при этом соответствующую дугу.

Деревья решений не требуют предварительной обработки данных, способны работать с категориальными переменными, просты для визуализации и понимания, отличаются сравнительно высокой на-

дежностью и точностью. Метод отличают быстрый процесс обучения (построения дерева), а также высокая вычислительная эффективность обработки значительных объемов данных, однако он подвержен переобучению. Кроме того, задача построения оптимального дерева требует построения всех возможных вариантов и полного их перебора (т. е. является неустранимо переборной).

Применение деревьев решений лучше подходит для задач с дискретными (категориальными) значениями с четким набором отличных атрибутов, а также в тех случаях, когда важно понимать логику получения и интерпретации результатов.

Случайный лес. Метод, при котором строится ансамбль деревьев решений, каждое из которых само по себе дает очень невысокое качество классификации, но в совокупности за счет их большого количества результат получается хорошим. Для обучения каждого дерева используются несколько разные обучающие множества [5]. Для формирования каждой ветви дерева используется случайная комбинация бинарных вопросов. В результате деревья будут похожими, но не будут являться точными копиями. Когда поступает новая выборка для предсказания, она передается каждому из деревьев в ансамбле. Затем собираются прогнозы отдельных деревьев, и наиболее популярное предсказание используется в качестве результата.

Нейронная сеть. Использует слои нейронов для передачи активации, благодаря чему возможно обучение и прогнозирование. Из-за своей сложности результаты не поддаются интерпретации, хотя зачастую обладают высокой точностью.

Задача: использовать закономерности в данных, постоянно улучшая прогнозирование по мере появления новых результатов.

Метод обучения: с подкреплением.

В отличие от обучения с учителем и без, где модели проходят обучение и после применяются без дальнейших изменений, модель обучения с подкреплением постоянно развивается, используя результаты обратной связи.

Настройка параметров алгоритмов ИАД

Многочисленные алгоритмы, доступные в ИАД, приводят к огромному числу потенциальных моделей, которые можно построить. Но даже один такой алгоритм способен генерировать различные результаты в зависимости от настройки его параметров.

Параметры задают тонкую регулировку алгоритма. У разных алгоритмов свои параметры настройки. Если алгоритм слишком чувствителен и принимает случайные отклонения данных за закономерности — то такой алгоритм склонен к *переобучению* (overfitting). Такая модель точна для прогнозирования по уже имеющимся данным, но меньше подходит для анализа новой или будущей информации.

Если алгоритм, наоборот, слишком нечувствителен и основные закономерности упустил, то эта порождает проблему *недообучения* (underfitting). Такая модель может пренебрегать важными тенденциями и дает менее точные предсказания как для текущих, так и для будущих данных.

Но, когда параметры настроены хорошо, алгоритм достигает равновесия, определяя главные тенденции, сбрасывая со счетов мелкие отклонения и предлагая хорошую прогностическую модель. В таблице 7 представлен список параметров для разных алгоритмов, которые подлежат настройке.

Таблица 7

Параметры настройки моделей ИАД

Метод анализа	Параметры настройки
Регрессионный анализ	Параметр регуляризации (для лассо или ридж-регрессии)
Метод *k*-ближайших соседей	Число ближайших соседей *k*
Метод опорных векторов	Параметр стоимости Параметры ядра Параметр эластичности
Дерево решений	Минимальный размер конечных узлов Максимальное число конечных узлов Максимальная глубина дерева
Случайные леса	Все параметры деревьев решений Число деревьев Число переменных для выбора в каждой разбивке
Нейронные сети	Число скрытых слоев Число нейронов в каждом слое Число итераций обучения Коэффициент скорости обучения Первоначальные веса связей

Оценка результатов работы алгоритмов

После того как модель построена, требуется оценить качество полученных результатов и их соответствие целям анализа.

Выборе модели основывается на оценках смещения, точности и надежности прогноза. Под *смещением* (bias) понимается различие между рассчитанным по модели прогнозом и истинным неизвестным значением моделируемой переменной. *Точность* (accuracy) — различие между оценками, основанными на данных выборки и истинным значением реализации оцениваемой величины. *Надежность* (precision) — различие между оценкой у по разовой выборке и средним из прогнозов по всему множеству выборок, которые могут быть взяты из той же совокупности.

Для сравнения моделей по степени точности предсказаний используются *метрики оценки*. Эти метрики определяют типы прогностических ошибок и штрафуют за них по-разному.

Рассмотрим три оценочные метрики, используемые чаще всего. В зависимости от целей исследования могут быть разработаны новые метрики.

1. Метрики классификации и прогнозирования.

Эта группа содержит две основные метрики.

— Процент верных прогнозов — доля достоверно правильных предсказаний, это простейшая мера точности прогнозирования.

— Матрица неточностей (confusion matrix) дает представление о том, где модель прогнозирования преуспела и где она потерпела неудачу (пример матрицы неточностей приведен в таблице 8).

Таблица 8

Оценка точности детектирования (предсказания) СПАМ-письма с помощью матрицы неточностей

Прогноз / Факт	не СПАМ	СПАМ
не СПАМ	32,5%	3,4%
СПАМ	28,0%	36,2%

В представленной таблице 32,5% писем были правильно отнесены к письмам, не содержащим СПАМ, т. е. к обычным письмам. 36,2% писем классифицированы правильно как СПАМ. 28,0% ошибочно

отнесены к обычным письмам (то есть угроза не выявлена) и 3,4% классифицированы как СПАМ, хотя являются обычными.

2. Метрика регрессии.

Речь идет о традиционной мере отклонения, широко используемой в статистике, эконометрике и теории риска — среднеквадратичном отклонении (СКО, Root Mean Squared Error, RMSE).

Поскольку при регрессии используются непрерывные числовые значения, то ошибки обычно измеряют количественно, как разницу между предсказанными и реальными значениями, распределяя штрафы и исходя из величины ошибки. СКО — популярная метрика регрессии, особенно полезная в случаях, когда требуется избежать крупных ошибок: каждая из них возводится в квадрат, что усиливает значимость такой ошибки. Это делает метрику крайне чувствительной к резко отклоняющимся значениям, за которые она штрафует модель.

Валидация модели

Метрики не дают полной картины эффективности модели.

Из-за эффекта переобучения, модели, хорошо себя показавшие на уже имеющихся данных, могут не справиться с новыми. Чтобы этого избежать, необходимо подвергать модели оценке, используя надлежащую процедуру валидации.

Валидация (validation) — это оценка того, насколько хорошо модель предсказывает новые данные.

Чаще всего, вместо того, чтобы ждать поступления новых данных, для проверки модели текущий набор данных разбивают на два сегмента.

Первый выступит в роли *обучающего набора* данных (training dataset), а второй послужит заменой для новой информации в качестве *тестового набора* данных (test dataset) для оценки точности прогностической модели. Лучшей моделью признается та, которая дает самые точные предсказания на тестовом наборе. Чтобы процесс валидации был эффективен, необходимо выбирать элементы для обучающего и тестового набора данных случайно и беспристрастно.

Однако если изначальный набор данных мал, аналитик не сможет позволить себе отложить их часть для формирования тестового набора, поскольку тогда пришлось бы пожертвовать точностью, которая может существенно снизиться при сокращении доступного объема данных. По этой причине, вместо использования двух различных наборов данных для испытания одного набора проверкой другим, мож-

но обойтись изначальным набором, устроив перекрестную проверку — кросс-валидацию.

Кросс-валидация (cross-validation) позволяет полностью задействовать данные путем разделения их набора на несколько сегментов для поочередной проверки модели. За одну итерацию все сегменты, кроме одного, используются для обучения модели, которая сама проверяется на последнем сегменте. Этот процесс повторяется до тех пор, пока каждый сегмент не отработает в роли тестового (рис. 40).

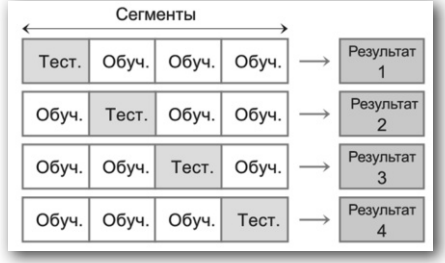

Рис. 40. Пример кросс-валидации набора данных

На рис. 40 набор данных разделен на четыре сегмента, а итоговая точность прогнозирования может быть рассмотрена как среднее значение четырех результатов. Поскольку для предсказаний на каждой итерации использовались разные сегменты, их прогнозы могут разниться. Приняв во внимание эту вариативность, можно дать более здравую оценку действительным прогностическим способностям модели. А в качестве итоговой оценки точности модели принимают среднее значение за все итерации.

Если результаты кросс-валидации показывают, что прогностическая точность модели невысока, можно вернуться к настройке параметров или обработать данные иначе.

Следует отметить, что даже высокие оценки кросс-валидации не дают гарантии получения хороших модельных результатов на новых данных.

Практический пример ИАД — анализ мошеннических транзакций

Рассмотрим процесс машинного обучения, используя в качестве примера набор данных транзакции при онлайновых покупках, собранные компанией розничной электронной торговли [24]. Краткое

описание процесса построения модели с исходным текстом на языке Python представлен на GitHub[2].

Исходный набор размеченных данных представлен в файле формата CSV[3] и содержит описание 39 221 транзакций, для каждой из которых приведено 5 ключевых характеристик, а также бинарная метка (0 или 1), которая показывает, является ли транзакция мошеннической (1) или нет (0).

Формат представления данных в виде текста, разделенного запятыми (CSV), является стандартным для аналитических исследований, такой формат может быть обработан даже MS Excel. В первой строке файла содержатся имена полей (характеристик) (рис. 41): accountAgeDays, numItems, localTime, paymentMethod,paymentMethodAgeDays, label. Последующие строки представляют собой записи данных.

```
accountAgeDays,numItems,localTime,paymentMethod,paymentMethodAgeDays,label
29,1,4.745402,paypal,28.2048611111,0
725,1,4.742303,storecredit,0.0,0
845,1,4.921318,creditcard,0.0,0
503,1,4.886641,creditcard,0.0,0
2000,1,5.040929,creditcard,0.0,0
119,1,4.962055,paypal,0.0,0
2000,1,4.921349,paypal,0.0,0
371,1,4.876771,creditcard,0.0,0
2000,1,4.748314,creditcard,0.0,0
4,1,4.461622,creditcard,0.0,0
26,1,4.745402,paypal,0.0,0
2000,1,4.505662,creditcard,0.000694444444444,0
```

Рис. 41. Пример размеченных данных о транзакциях

Пример произвольно выбранной строки:

```
196, 1, 4.962055, creditcard, 5.10625, 0
```

Представим строку в более удобной для чтения форме.

accountAgeDays:	196
numItems:	1
localTime:	4.962055
paymentMethod:	creditcard
paymentMethodAgeDays:	5.10625
label:	0

[2] https://github.com/dm-fedorov/ml/blob/master/logistic-regression-fraud-detection.ipynb
[3] https://github.com/dm-fedorov/ml/blob/master/datasets/payment_fraud.csv

Данная транзакция была выполнена пользователем, учетная запись которого создана 196 дней назад (accountAgeDays). Пользователь приобрел один экземпляр товара (numItems) приблизительно в 4:58 утра по местному времени (localTime). Оплата произведена с помощью кредитной карты (paymentMethod), и этот способ оплаты был добавлен за 5 дней до транзакции (paymentMethodAgeDays). Значение 0 метки (label) свидетельствует о том, что данная транзакция не мошенническая.

Пометка о мошеннической транзакции (значение label равно 1) выставляется, если владелец кредитной карты подал заявление о возврате денег (chargeback) по этой транзакции.

Финансовые потери от компенсации мошеннических транзакций можно существенно уменьшить, если найти способ определения, в какой степени транзакция похожа на мошенническую, и блокировать подозрительные транзакции. Одно из решений — использовать некоторые правила, например: «Если способ оплаты был добавлен в день транзакции и количество экземпляров товара больше 10, то транзакция мошенническая». Однако при таком подходе может возникнуть слишком большое количество ложноположительных срабатываний (когда обычная транзакция будет принята за мошенническую).

В таком случае на помощь приходит машинное обучение, а задача примет следующий вид — получить прогнозирующий алгоритм, умеющий определять мошенническую транзакцию по пяти признакам из набора имеющихся данных.

Поскольку набор данных содержит метку, соответствующую цели прогнозирования, такой массив называется «размеченным набором данных» (labeled dataset), что позволяет выполнять на нем обучение с учителем.

Создаваемая система выявления мошенничества будет принимать на входе признаки транзакции, а на выходе — возвращать вероятностную оценку (probability score) того, в какой степени исследуемая транзакция может считаться мошеннической.

Воспользуемся библиотекой scikit-learn (scikit-learn.org), включающей реализацию алгоритмов машинного обучения, и написанной на языке Python. Кроме того, для обработки данных используем модуль Pandas (pandas.pydata.org).

Следующая команда выполняет преобразование из текстового формата CSV в тип данных DataFrame[4]:

```
import pandas as pd
df = pd.read_csv('datasets/payment_fraud.csv')
```

Воспользуемся функцией sample для извлечения из DataFrame трех случайных строк (таблица 9).

Таблица 9

Пример представления данных в DataFrame

№	accountAge-Days	numItems	localTime	paymentMethod	paymentMetho-dAgeDays	label
12603	2	1	4.965339	creditcard	1.306944	0
25926	2000	1	4.921349	paypal	0.008333	0
14212	1087	1	4.876771	creditcard	19.909722	0

Столбец paymentMethod, соответствующий признаку «тип оплаты», содержит данные нечислового типа. В исследуемом наборе данных для этого признака допустимы только три значения: creditcard, paypal и storecredit. Многие алгоритмы машинного обучения требуют, чтобы все признаки были числовыми. Существуют различные методы преобразования категорий в числовые значения, однако они требуют знания функциональной зависимости между качественными градациями и результирующим признаком, или хотя бы задания отношения порядка на множестве качественных градаций.

Другой подход заключается в увеличении числа переменных путем ввода для каждой градации качественной шкалы новой переменной. Каждая переменная будет иметь значение 1, если исходный признак принимает значение, равное соответствующей градации, и 0 — в противном случае. Такая методика кодирования категориальной переменной называется унитарным кодированием (one-hot encoding), а определяемые по этой методике переменные называются фиктивными (dummy variables) в терминологии математической статистики. Для того, чтобы повышение размерности задачи не приводило к снижению точности результата, это должно компенсироваться большим объемом выборки данных.

[4] Описание формата приведено на странице документации https://pandas.pydata.org/pandas-docs/stable/reference/api/ pandas.DataFrame.html

Для преобразования категориальных переменных в числовую форму воспользуемся функцией get_dummies[5].

```
df = pd.get_dummies(df, columns = ['paymentMethod'])
df.sample(3)
```

В структуру данных DataFrame будут добавлены три столбца с бинарными значениями: paymentMethod_creditcard, paymentMethod_paypal и paymentMethod_storecredit. Рассмотрим значения, принимаемые новыми переменными для рассмотренных ранее данных из таблицы 9 (таблица 10). Каждый новый признак является бинарным, и для каждой записи (строки) данных только один из них принимает значение 1.

Таблица 10

Пример значений новых столбцов DataFrame

№	paymentMethod_creditcard	paymentMethod_paypal	paymentMethod_storecredit
12603	1	0	0
25926	0	1	0
14212	1	0	0

Теперь можно разделить общий массив данных на тренировочный (X_train) и тестовый (X_test) наборы с помощью функции train_test_split [28].

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
     df.drop('label', axis = 1), df['label'],
     test_size = 0.33, random_state = 17)
```

В первом аргументе функции передается значение `df.drop ('label', axis = 1)`. Разделение на наборы `X_train, X_test` выполняется в соотношении 67:33, поскольку передан аргумент `test_size = 0.33, random_state` — начальное значение, используемое генератором случайных чисел.

Столбец label исключается из набора *X* перед разделением на `X_train, X_test`. После этого метки (`label`) также будут разделены в том же соотношении на наборы `y_train и y_test`.

[5] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

Применим стандартный алгоритм обучения с учителем — логистическую регрессию (logistic regression).

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression().fit(X_train, y_train)
```

Здесь создается объект класса LogisticRegression[6], затем вызывается функция `fit()`, в которую передаются обучающие наборы данных `X_train`, `y_train`. В переменную `clf` записывается тренировочная модель классификатора.

Для прогнозирования с использованием полученной модели необходимо передать непомеченные признаки в функцию `predict`.

```
y_pred = clf.predict(X_test)
```

Проверяя `y_pred`, можно увидеть, что прогнозируемые метки присвоены каждой строке в тестовом наборе данных `X_test`. Отметим, что во время тренировки классификатор не имел доступа к набору меток `y_test`, поэтому прогнозы в `y_pred` являются исключительно результатами обобщений, выведенных при работе с тренировочным набором.

Затем применим функцию `accuracy_score()` для оценки точности выполненных прогнозов.

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred, y_test))
```

Результат свидетельствует о высокой точности прогноза на тестовых данных: 0.999922738159623 (максимально возможное значение — 1, минимальное — 0).

Сгенерируем матрицу несоответствий (таблица 11).

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

Таблица 11

Матрица несоответствий для классификации транзакций

Прогноз / Действительно	не мошеннические	мошеннические
не мошеннические	12753	0
мошеннические	1	189

[6] Описание класса и методов работы с ним приведено https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Здесь обнаружен только один случай неправильной классификации на всем тестовом наборе. 189 транзакций верно помечены как мошеннические. В одном случае мошенническая транзакция не была выявлена. Ложноположительных срабатываний нет.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Аникин И.В., Потапов А.С. Программный комплекс оценки рисков информационной безопасности на основе продукционно-фреймовой модели // Научно-технические ведомости СПбГПУ, №5 Информатика. Телекоммуникации. Управление. — 2010. — С. 98–102.

2. Васильев И.В. Интеллектуальные системы защиты информации. — 2-е изд., испр. и доп. — М.: Машиностроение, 2013. — 172 с.

3. Васильева И.Н. Управлением рисками информационной безопасности: учебное пособие. — СПб.: Изд-во СПбГЭУ, 2016. — 177 с.

4. Ганиев А.А., Касимова Г.И. Анализ моделей и алгоритмов обнаружения компьютерных атак на основе положений политики безопасности // Молодой ученый — № 9 (113), 2016. — С. 54–57.

5. Глассner Э. Глубокое обучение без математики. Т. 1: Основы — М.: ДМК-Пресс, 2019. — 578 с.

6. Гниденко И.Г., Морозов С.К., Федоров Д.Ю. Многопроцессорные системы и параллельное программирование: учебное пособие. — СПб.: Изд-во СПбГЭУ, 2019. — 68 с.

7. Джарратано Д., Райли Г. Экспертные системы. Принципы разработки и программирование, 4-е издание. — М.: Вильямс, 2007. — 1152 с.

8. Емельянова Ю.Г., Талалаев А.А., Тищенко И.П., Фраленко В.П., Нейросетевая технология обнаружения сетевых атак на информационные ресурсы // Программные системы: теория и приложения. — Том 2, выпуск 3, 2011. — С.3–15.

9. Замятин А.В. Интеллектуальный анализ данных: учебное пособие. — Томск: Издательский Дом Томского государственного университета, 2016. — 120 с.

10. Защита информации в компьютерных системах: монография / под ред. Е.В. Стельмашонок, И.Н. Васильевой. — СПб.: Изд-во СПбГЭУ, 2017. — 163 с.

11. Ишанханов С.Р., Чернокнижный Г.М. Система обнаружения вторжений OS NEOS // Интеллектуальные и информационные технологии в формировании цифрового общества: сборник научных статей международной научной конференции. Санкт-Петербургский государственный экономический университет. — СПб.: Изд-во СПбГЭУ, 2017. — С. 157–159.

12. Каширин Д.И., Каширин И.Ю. Модели представления знаний в системах искусственного интеллекта // Вестник РГРТУ. № 1 (выпуск 31). Рязань, 2010. — С. 36–43.

13. Козлов А.Н. Интеллектуальные информационные системы: учебник. — Пермь: Изд-во ФГБОУ ВПО Пермская ГСХА, 2013. — 278 с.

14. Котельников Е.В., Колеватов В.Ю. Методы искусственного интеллекта в задачах обеспечения безопасности компьютерных сетей, 2008 [Электронный ресурс]. URL: http://window.edu.ru/resource/166/56166/files/62320e1-st07.pdf.

15. Леоненков А.В. Нечеткое моделирование в среде MATLAB и FuzzyТЕСН. — СПб.: BXB-Санкт-Петербург, 2005. — 736 с.

16. Морозова В.А., Паутов В.И. Представление знаний в экспертных системах: учебное пособие. — Екатеринбург: Изд-во Урал. ун-та, 2017. — 120 с.

17. Нестерук Г.Ф., Осовецкий Л.Г., Харченко А.Ф. Информационная безопасность и интеллектуальные средства защиты информационных ресурсов (Иммунология систем информационных технологий). — СПб.: Изд-во СПбГУЭФ, 2003. — 364 с.

18. Нестерук Ф.Г., Нестерук Г.Ф., Осовецкий Л.Г. Основы организации адаптивных систем защиты информации. — СПб.: СПб ГУ ИТМО, 2008. — 112 с.

19. Орлов А.И. Теория принятия решений: учебное пособие. — М.: Изд-во «Март», 2004. — 656 с.

20. Потапов А.С. Технологии искусственного интеллекта: учебное пособие. — СПб: СПбГУ ИТМО, 2010. — 218 с.

21. Фролов П.В., Чухраев И.В., Гришанов К.М. Применение искусственных нейронных сетей в системах обнаружения вторжений // Системный администратор. Вып. №9 (190), 2018 [Электронный ресурс]. URL: http://samag.ru/archive/article/3724.

22. Чернов В.Г. Основы теории нечетких множеств. Решение задач многокритериального выбора альтернатив: учебное пособие. — Владимир: Изд-во Владимирского гос. ун-та, 2005. — 100 с.

23. Чернокнижный Г.М. Защита сетевых информационных технологий: учебное пособие — СПб.: Изд-во СПбГЭУ, 2018. — 128 с.

24. Чио К., Фримэн Д. Машинное обучение и безопасность. — М.: ДМК Пресс. — 2020. — 390 с.

25. Чуйков А.В., Вульфин А.М., Васильев В.И. Нейросетевая система преобразования биометрических признаков // Доклады ТУСУР, 2018, том 21, № 3 — С.35–41.

26. Шитиков В.К., Мастицкий С.Э. Шитиков В.К., Мастицкий С.Э. Классификация, регрессия, алгоритмы Data Mining с использованием R, 2017 [Электронный ресурс]. URL: https://github.com/ranalytics/data-mining/blob/master/Book/ Shitikov_Mastitsky_2017_DM_R.pdf.

27. Ын А., Су К. Теоретический минимум по Big Data. Все, что нужно знать о больших данных. — СПб.: Питер, 2019. — 208 с.

28. Cross-validation: evaluating estimator performance [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/cross_validation.html.

Учебное издание

Васильева Ирина Николаевна
Федоров Дмитрий Юрьевич

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ

Учебное пособие

Редактор В.М. Макосий

Подписано в печать 09.06.2020. Формат 60×84 1/16.
Усл. печ. л. 7,5. Тираж 50 экз. Заказ 2176.

Издательство СПбГЭУ. 191023, Санкт-Петербург, Садовая ул., д. 21.

Отпечатано на полиграфической базе СПбГЭУ