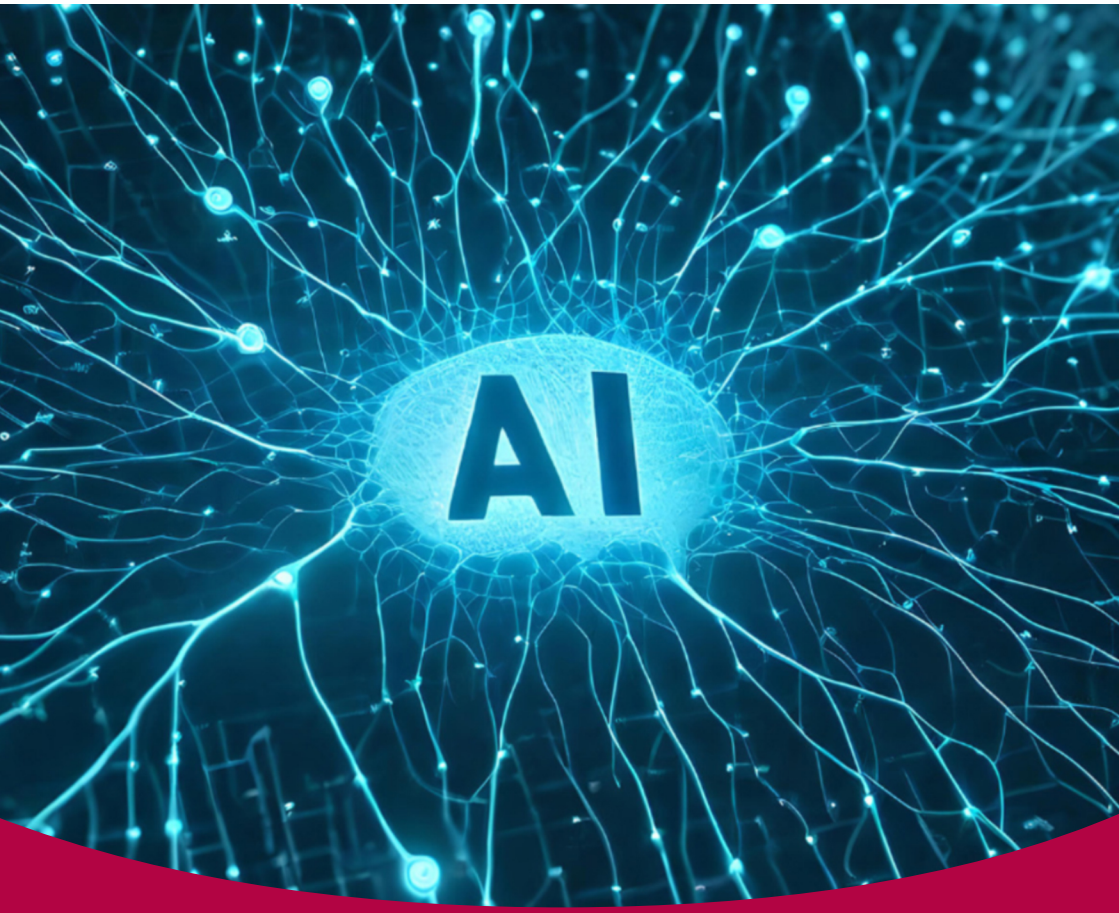
Системы искусственного интеллекта



Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации
Уральский государственный экономический университет

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Рекомендовано
Советом по учебно-методическим вопросам и качеству образования
Уральского государственного экономического университета
в качестве учебного пособия

Екатеринбург
2024

УДК 004.8(075)
ББК 32.813я73
С40

Рецензенты:

кафедра математических методов в экономике и социально-экономических наук
Уральского института фондового рынка
(протокол № 8 от 16 апреля 2024 г.);

профессор учебно-научного центра «Информационная безопасность»
Института радиоэлектроники и информационных технологий — РтФ
Уральского федерального университета имени первого Президента России Б. Н. Ельцина,
доктор технических наук, профессор
С. В. Поршнев

Авторский коллектив:
В. П. Часовских, *Е. Н. Стариков*, *Г. А. Акчурина, Е. В. Кох*

С40 **Системы искусственного интеллекта** : учебное пособие / В. П. Часовских, Е. Н. Стариков, Г. А. Акчурина, Е. В. Кох ; Министерство науки и высшего образования Российской Федерации, Уральский государственный экономический университет. — Екатеринбург : УрГЭУ, 2024. — 194 с.

В учебном пособии определяется понятие искусственного интеллекта, его место в национальных программах Российской Федерации. Рассматриваются проблемы определения искусственного интеллекта, история развития и современное состояние. Формулируется современный подход, основанный на использовании рационального агента. Уделяется большое внимание системам глубокого обучения и обучению с подкреплением. Структура и содержание пособия полностью соответствуют требованиям федеральных государственных образовательных стандартов высшего образования (ФГОС ВО).

Для студентов очной и заочной форм обучения направлений подготовки по программам бакалавриата 02.03.03 «Математическое обеспечение и администрирование информационных систем» (реализация профессиональных компетенций дисциплин «Математическое моделирование», «Архитектура аппаратных и программных средств», «Проектирование интеллектуальных информационных систем», «Формализация информации и БигДата (Big Data)»), 09.03.01 «Информатика и вычислительная техника» (реализация профессиональных компетенций дисциплин «Разработка информационных систем и баз данных», «Проектирование архитектуры программных систем»).

УДК 004.8(075)
ББК 32.813я73

Введение

Электронно-вычислительные машины (ЭВМ) широко применяются во всех видах деятельности, всех странах и на всех континентах. Распространение ЭВМ многообразно и всеобъемлюще. Современная наука информатика отвечает на все вопросы применения ЭВМ, является мощным механизмом научно-технического прогресса. Именно широкому применению ЭВМ современный мир обязан выдающимися достижениями и успехами в автоматизации всех видов производства, в создании новых технологий, в существенном изменении эффективности труда и развития управления, в новых технологиях системы высшего образования. Не подвергается сомнению роль ЭВМ в развитии науки — ЭВМ определяет новую, значимую роль в задачах фундаментальных проблем математики, физики, химии, атомной энергетики. Без применения ЭВМ нельзя проводить актуальные аэрокосмические исследования. Успехи в этих областях стимулируют создание новых технологий, получение новых материалов, совершенствование конструкторских разработок. В наши дни решение никаких важнейших научных задач немыслимо без применения высокопроизводительных и сверхвысокопроизводительных ЭВМ. В стране сформировалась инфраструктура, получившая название «информационные технологии».

В настоящее время во всем мире проходит 4-я промышленная революция [1–3]; особенностью этой революции является то, что внедрение новых технологий с использованием ЭВМ во всех видах деятельности характеризуется большой скоростью и сопровождается мощнейшей конкуренцией. Любые типы или виды ЭВМ порождают данные, объединенные или нет в группы, группы больших данных (Big Data). 4-я промышленная революция фундаментально изменяет нашу жизнь, нашу науку, образование, наш труд и наше общение. Появились и появляются новые технологии

вычислений (включая квантовые компьютеры), искусственный интеллект, блокчейн и системы распределенного реестра, аддитивное производство, виртуальная и дополненная реальность, робототехника, беспилотные транспортные средства (дроны), Интернет вещей и т. п.

В рамках Национального проекта «Цифровая экономика Российской Федерации» утвержден и действует Федеральный проект «Искусственный интеллект».

В 2024 г. утвержден Национальный проект «Экономика данных». Значимость этого Национального проекта определена указаниями нашего Президента в 2023 г.: «Наша принципиальная задача — перевести всю экономику, социальную сферу, органы власти, работу органов власти на качественно новые принципы работы, внедрить управление на новых данных — на основе больших данных... Все, что связано с данными, большими данными, принимает критически важное значение. Речь, по сути, идет о системообразующей инфраструктуре для нашего дальнейшего развития, для будущего нашей экономики в целом… Предлагаю в течение года подготовить новый национальный проект на период до 2030 г., а именно — нацпроект по формированию экономики данных»[1].

По поручению Президента Российской Федерации от 29 января 2023 г. № Пр-172 Минобрнауки разработало и разослало в вузы РФ учебный модуль «Системы искусственного интеллекта» для включения во все образовательные программы.

Учебное пособие состоит из важного теоретического материала и обобщения в конце каждой главы.

Мы стремимся сформулировать положения, полученные в фундаментальных исследованиях отечественных и зарубежных ученых по технологиям искусственного интеллекта за последние 90 лет, и все, что накопило человечество об интеллекте за последнее тысячелетие в смежных областях знаний.

Мы будем рассматривать проблемы определения искусственного интеллекта, историю развития и современное состоя-

[1] *Путин* предложил подготовить глобальный проект по формированию единой экономики данных / Российское агентство правовой и судебной информации. URL: http://www.kremlin.ru/events/president/news/71666 (дата обращения: 01.03.2024).

ние. На основе анализа проблем определим современный подход, основанный на использовании рационального агента. По существу, создание технологий искусственного интеллекта и, в конечном итоге, сильного искусственного интеллекта — это возможность разработки адекватной программы агента в данной конкретной архитектуре.

Рассматривается введение в искусственный интеллект и основные методы машинного обучения для работы с табличными данными. Уделяется большое внимание системам глубокого обучения и обучению с подкреплением.

В конце каждой главы приведены вопросы для самоконтроля. Обучающимся рекомендуется найти ответы на них с целью наилучшего усвоения теоретического материала. Также предлагается практическая работа, которую надо выполнить, используя основной теоретический материал.

Пособие предназначено для студентов очной и заочной форм обучения направлений подготовки по программам бакалавриата 02.03.03 «Математическое обеспечение и администрирование информационных систем» (реализация профессиональных компетенций дисциплин «Математическое моделирование», «Архитектура аппаратных и программных средств», «Проектирование интеллектуальных информационных систем», «Формализация информации и БигДата (Big Data)»), 09.03.01 «Информатика и вычислительная техника» (реализация профессиональных компетенций дисциплин «Разработка информационных систем и баз данных», «Проектирование архитектуры программных систем»).

Глава 1 Искусственный интеллект — возникновение, история развития и текущее состояние

1.1. Основные стратегические ориентиры в области искусственного интеллекта

Искусственный интеллект (ИИ), или, более точно, различные аспекты технологий ИИ, остается темой с нечетко определенными границами. С 26 по 30 августа 2024 г. в Москве состоялся Всемирный конгресс «СОЗНАНИЕ 2024», где собрались ведущие ученые в области искусственного интеллекта, системной теории и алгебраической биологии. В рамках конгресса было представлено 114 докладов, посвященных тематике ИИ, в которых было выделено четыре определения искусственного интеллекта и одно — естественного.

В статье А. Тьюринга «Может ли машина мыслить?» [6] предложен уникальный метод исследования искусственного интеллекта. Также значительные вклады внесены им в другие ключевые области исследований. Наш акцент будет на работах отечественных ученых.

Термин «искусственный интеллект» применяется экспертами во множестве сфер: литературе, журналистике, коммерции, научных исследованиях, при этом каждая профессиональная группа интерпретирует его по-своему.

Современное общество поляризовано в своем восприятии искусственного интеллекта: некоторые рассматривают его как высвобождающую силу, предназначенную для избавления человечества от многих его невзгод и проблем, тогда как другие видят в ИИ темную силу, напоминающую четырех Всадников Апока-

липсиса из Откровения Иоанна — символов катастроф и разрушения, способных подорвать основы человеческой цивилизации и превратить людей в подчиненных машинному разуму.

В общем понимании искусственный интеллект описывается как возможность машины выполнять задачи, требующие человеческого ума.

Этот термин можно детализировать по нескольким аспектам:

– аппарат, обладающий возможностью восприятия и осмысления окружающей среды при помощи датчиков, включая обработку визуальных и аудиоданных;

– способность разработать и воплотить в жизнь новые объекты (к примеру, графические изображения, мультимедийные видеопроекты и текстовые материалы);

– способность выполнять задания, требующие мышления (как в шахматах или го);

– или готовность переходить от одного дела к другому и креативно справляться с трудными умственными вызовами.

Для концептуализации искусственного интеллекта рекомендуется исходить из определения «естественного интеллекта», а именно — умения человеческого мозга осуществлять интеллектуальную деятельность через приобретение, запоминание и осознанное преобразование информации, а также использование полученных данных для манипулирования окружающим пространством. В данном контексте под интеллектуальными задачами понимаются такие задачи, для которых ранее не существовало известного метода решения, т. е. не был разработан конкретный алгоритм их выполнения.

На основании вышеизложенного член Академии наук Российской Федерации И. Каляев выдвигает определение: «Искусственный интеллект — это характеристика искусственных систем, способных решать задачи мышления без заранее заданного алгоритма решения» [2, с. 9].

Такой подход приводит к значимому утверждению: когда проблема разрешима с помощью компьютера, она утрачивает статус интеллектуальной задачи, так как это указывает на наличие разработанного алгоритма (учитывая, что компьютер оперирует исключительно алгоритмически).

Таким образом, многие современные приложения искусственного интеллекта на самом деле не обладают истинной когнитивной способностью, а представляют собой лишь алгоритмические системы, выполняющие задачи в соответствии с заранее заданными инструкциями.

Это инструменты работы, аналогичные молотку. Однако, если молоток расширяет физические способности человека, то программное обеспечение повышает интеллектуальные и познавательные возможности.

Поэтому то, что современность именует искусственным интеллектом, более точно было бы классифицировать как высокоразвитые интеллектуальные алгоритмы или же компьютерные системы, имитирующие интеллект. Однако терминология «искусственный интеллект» в отношении программного обеспечения укоренилась на международном уровне, стала общепринятой, и, следовательно, ее применение становится неизбежным и в нашем использовании.

Сегодня в сфере искусственного интеллекта принято различать две основные категории: нейронные сети с ограниченными способностями, известные как слабый ИИ, и развитые системы с продвинутыми когнитивными функциями, именуемые сильным ИИ.

Слабый ИИ создается для выполнения специфических функций, таких как игра в шахматы, распознавание изображений или управление транспортными средствами, что относится к области логически ориентированных разработок в домене искусственного интеллекта.

В идеальном сценарии развитие сильного искусственного интеллекта должно привести к его способности эффективно решать любые задачи, демонстрируя уровень интеллекта, сопоставимый с человеческим разумом. Это предполагает создание универсальной системы, способной эмулировать мозговую деятельность. Сильный ИИ, таким образом, анализируется через призму нейрокибернетики, подразумевающей тесное взаимодействие нейронаук и кибернетических исследований.

На данный момент в области искусственного интеллекта преобладают системы, разработанные для выполнения конкретных задач, например, для идентификации лиц, игры в шахматы

или осуществления машинного перевода. Эти системы, известные как узкоспециализированный ИИ, обладают возможностью обрабатывать и решать одну выбранную задачу. В то же время вопрос о реализации общего искусственного интеллекта (ОИИ) — машины, способной выполнять широкий спектр сложных интеллектуальных функций, аналогичных человеческим, — до сих пор остается открытым. Сложность заключается в неопределенности того, могут ли существующие технологические подходы и разработки в области ИИ стать основой для создания ОИИ.

Анализируем функции ИИ-ассистента. В идеале общий ИИ должен имитировать работу человека-помощника всесторонне, выполняя разнообразные задания, такие как управление графиками, обработка входящих звонков, адаптация к новым обязанностям через безостановочное обучение. В отличие от этого, современные разработки представляют собой более ограниченные системы, способные преобразовывать голос в текст, формировать стандартизированные ответы на электронные письма и устанавливать уведомления на основе введенных инструкций, при этом для каждой функции требуется специфический алгоритм.

С другой стороны, разработка команд для выполнения даже этих, на первый взгляд, простых заданий по сравнению с возможностями полного персонального ассистента исторически заняла значительное количество времени.

Изучим краткий исторический обзор развития интеллектуальных алгоритмов и эволюцию концепции искусственного интеллекта.

Первая половина XIX в. Ада Лавлейс стала пионером в разработке концепции аналитической машины Чарльза Бэббиджа, воплотившейся в создании первого в истории алгоритма, предназначенного для исполнения этим устройством. Этот факт утверждает ее статус как первого в мире программиста. Лавлейс была убеждена в том, что хотя такие устройства, как аналитическая машина Бэббиджа, способны выполнять задачи по заданным предписаниям, они лишены способности к самостоятельному мышлению, генерации новых идей или целеполаганию. Она аргументировала, что аналитическая машина может обрабатывать указания и проводить анализы, но не способна к самостоятельному выводу аналитических заключений или формулировке зако-

номерностей. Позднее ее замечания Алан Тьюринг охарактеризовал как «Возражение леди Лавлейс», рассматривая его в контексте возможности самообучения машин и перспектив появления искусственного интеллекта. Тьюринг предполагал, что способность машины к самомодификации программы на базе обработки данных может рассматриваться как форма обучения, открывая путь к созданию искусственного разума. В знак признания достижений Ады Лавлейс один из самых мощных языков программирования был назван в ее честь — ADA. На рис. 1 изображена механическая вычислительная машина Ч. Бэббиджа, для которой была разработана первая программа Адой Лавлейс.

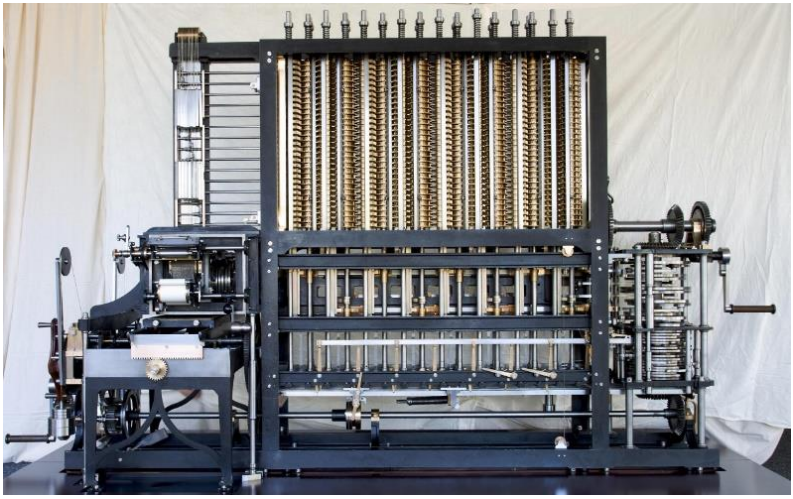


Рис. 1. Механический вычислительный аппарат, разработанный Ч. Бэббиджем

1940-е: тест Тьюринга. С момента создания в 1940-х гг. первых компьютеров возник вопрос о потенциале этих устройств и вероятности их интеллектуального равенства с человеком.

В 1950 г. выдающийся британский математик и логик Алан Тьюринг представил концепцию теста Тьюринга, который стал основополагающим критерием для оценки искусственного интеллекта на предмет его способности имитировать человеческое мышление. Сущность этого эксперимента заключается в том, что

судья общается через текстовые сообщения с двумя оппонентами, один из которых является машиной, а другой — человеком. В процессе диалога судья не знает, кто из участников является искусственным интеллектом, а кто — настоящим человеком. Главной целью судьи является разгадать, чьи ответы происходят от машины, и чьи — от человека, причем успешный ИИ должен суметь так подражать человеку, чтобы судья не смог его отличить от реального человека.

В XX в. были предприняты многочисленные попытки успешно пройти тест Тьюринга, в том числе разработка программы «Элиза» Джозефом Вейценбаумом в 1966 г., но они не смогли существенно приблизиться к достижению этой цели. В современном мире тестирование искусственного интеллекта на способность имитировать человеческое поведение происходит под эгидой различных международных организаций, причем специфика проведения такого эксперимента (временные рамки, количество необходимых «обманов» человека компьютером и т. д.) может меняться в зависимости от условий проведения.

В 2014 г. испытание, проведенное под эгидой Королевского общества в Лондоне, было успешно пройдено программой под названием «Женя Густман» (разработчик Владимир Веселов из Санкт-Петербурга), которая моделировала поведение ребенка, допускающего незнание некоторых аспектов. Программа смогла ввести в заблуждение судей в 33 % обращений, превысив порог успешности в 30 %. Тест Тьюринга, предназначенный для оценки способности машины имитировать человеческое поведение, часто становится объектом критики за то, что он не позволяет определить реальные интеллектуальные возможности искусственного интеллекта.

1943 г. — модель нейрона. В 1836 г. немецко-швейцарский физиолог Габриэль Густав Валентин осуществил открытие и описание нейронов головного мозга. В 1943 г. Уоррен Мак-Каллок и Уолтер Питтс разработали математическую модель нейрона человеческого мозга. В своем фундаментальном исследовании Мак-Каллок и Питтс стремились воспроизвести функционирование нейронных сетей мозга, применяя концепции и инструментарий математической логики. Они разработали чисто теоретическую модель, в которой мозг представлен как сеть взаимосвязанных ло-

гических узлов. Это исследование было мотивировано аналогией между биологическим нейроном и логическим пороговым элементом, который при определенных упрощающих допущениях можно описать как имеющий несколько бинарных входов и один бинарный выход.

Характеристики нейронов и нейронных сетей были установлены на основе следующих постулатов:

– активация нейрона следует принципу «абсолютность реакции»;

– время разбивается на отдельные единицы — такты.

Нейрон становится возбужденным в определенный момент времени, когда непосредственно перед этим моментом активируется строго определенное количество синапсов. Синапсы (от греч. σύναψις — соединение, связь) представляют собой точки контакта между нервными клетками, или нейронами, а также между нейронами и клетками-мишенями. Термин «синапс» ввел в употребление английский физиолог Чарльз Шеррингтон в 1897 г. Важно отметить, что для возбуждения нейрона количество активированных синапсов не зависит от их местоположения на нейроне или от предыдущей активности нейрона.

Передача сигнала между нейронами через синаптическую щель производится мгновенно, без промежутков времени, в один момент.

Синапсы классифицируются на возбуждающие и тормозящие. Тормозной синапс полностью блокирует активацию нейрона при передаче сигнала, не позволяя ему возбуждаться в данный момент. Структура нейронной сети остается стабильной со временем.

Формальные нейроны и нейронные сети, как определено Мак-Каллоком и Питтсом, соответствующие заданной аксиоматике, характеризуются следующими особенностями.

Во-первых, подтверждено, что данные формальные нейронные элементы способны выполнять любую функцию двоичной логики.

Вторым моментом является тот факт, что, используя эти моделированные нейроны, возможно создать такую нейронную сеть, которая способна имитировать любую операцию в рамках логики высказываний.

Анализируя концепцию, разработанную Джоном фон Нейманом, можно прийти к выводу, что эффективность системы измеряется ее способностью генерировать специфические ответы на внешние стимулы. Исходя из этого, применение модели нейронной сети, предложенной Уорреном Мак-Каллоком и Уолтером Питтсом, позволяет воплотить в жизнь любой процесс обработки информации, который может быть четко определен и выражен в ограниченном количестве слов.

Исследования данных ученых демонстрируют, что любую функцию, поддающуюся вычислению, можно задать через специфическую конфигурацию взаимосвязанных нейронов, а также что фундаментальные логические операции (конъюнкция «И», дизъюнкция «ИЛИ», отрицание «НЕ» и пр.) могут быть воплощены в рамках простых сетевых архитектур.

Более того, Мак-Каллок и Питтс предположили, что адекватно структурированные сети могут осуществлять процессы обучения.

1950-е гг.: Розенблаттов перцептрон. В 1950-х гг. Фрэнк Розенблатт, американский исследователь в области нейрофизиологии, создал перцептрон, который представляет собой математическую модель, имитирующую процесс восприятия информации мозгом человека.

Впоследствии стало ясно, что архитектура человеческого мозга неизмеримо более сложная, чем было представлено в исходной концепции перцептрона, подчеркивая ограниченность сравнения между ними. Проект Розенблатта, реализованный спустя несколько лет на машине IBM, проявил способность к классификации и распознаванию рукописных символов латиницы. Это достижение вызвало значительный интерес со стороны предпринимательской среды и государственных структур, стимулируя ожидания относительно будущего исследований в данной области.

Тем не менее, дальнейшее развитие перцептрона встретило серьезные препятствия, ведущие к затруднениям в финансировании исследований в сфере ИИ в последующие годы — эпоха, известная как «зима искусственного интеллекта».

1956 г. знаменует собой начало эпохи искусственного интеллекта. В ходе двухмесячного семинара, проходившего летом 1956 г. в Дартмуте, с участием выдающихся ученых в лице Джона

Маккарти, Марвина Минского, Клода Шеннона, Натаниэля Рочестера, Тренчарда Мура, Артура Самюэла, Рея Соломонова, Оливера Селфриджа, Аллена Ньюэлла и Герберта Саймона, были подробно изучены разнообразные программы логического характера и процесс обработки списков на компьютере. В результате этого глубокого обсуждения было принято единогласное решение поддержать предложение Джона Маккарти о присвоении разрабатываемой области науки названия «искусственный интеллект».

Основываясь на анализе материалов семинара, становится ясно, почему важно выделить искусственный интеллект как независимую дисциплину.

Почему невозможно представить все достижения в сфере искусственного интеллекта в контексте дисциплин, таких как теория управления, операционные исследования или теория принятия решений? Ведь их конечные задачи схожи с амбициями, присущими искусственному интеллекту.

Основная причина, по которой искусственный интеллект не квалифицируется строго как раздел математики, связана с его фундаментальной целью: имитацией и разработкой атрибутов, характерных для человеческого мышления и поведения, таких как творческие способности, способность к самообучению и обработка естественного языка. Эти аспекты выходят за рамки традиционных математических дисциплин. К тому же в искусственном интеллекте используется уникальная методология, охватывающая не только математические подходы, но и элементы информатики, лингвистики, психологии и других наук, что расширяет его область исследования и применения вне строгих математических границ.

Искусственный интеллект выделяется в ряду упомянутых сфер как ключевое направление информатики, отличающееся акцентом на разработке системного программирования (в отличие от операционных исследований, где преобладает компьютерное моделирование). Кроме того, ИИ уникален стремлением к созданию машин, способных к самостоятельным действиям в контексте постоянно эволюционирующих условий.

1969–1979: эпоха развития знаний в информационных системах. В первые 10 лет разработок в сфере искусственного интеллекта основная стратегия решения задач заключалась в ис-

пользовании универсального алгоритма поиска. Этот метод позволял соединять базовые шаги логических умозаключений в последовательность, чья цель — создание итоговых, законченных ответов на заданные вопросы.

Все теоретические понятия, связанные с данной задачей, были трансформированы из абстрактного уровня («основополагающих принципов») в прикладной формат («инструкции для применения»).

1980-е: экспертные системы. В ранние 1980-е гг. экспертные системы, программные продукты, моделирующие человеческие знания и навыки, стали широко применяться. Разработка таких программ включала экспансивный опрос экспертов для понимания их методов решения задач, в том числе по принципам, по которым психологи общаются с клиентами или медики ставят диагнозы. Эти выявленные алгоритмы затем трансформировались в архитектуру компьютерного кода.

Однако данные системы быстро исчерпали свой потенциал, в результате чего наступил второй период застоя в развитии искусственного интеллекта. Но несмотря на это, экспертные системы все еще находят применение в определенных сферах, причем особенно там, где критически важна надежность, как, например, в медицинской отрасли.

1990-е гг.: Deep Blue. В конце XX в. — начале XXI в. прогресс в области ИИ был стимулирован ростом вычислительных возможностей, интенсивным использованием математического анализа, а также фокусом на решении специфических проблем. Искусственный интеллект находит применение во многих сферах: от управления цепочками поставок до банковского дела и здравоохранения. Знаковым достижением стала разработка IBM Deep Blue, который в 1997 г. победил чемпиона мира по шахматам.

С наступлением XXI в. данные стали ключевым фактором в развитии алгоритмов ИИ, сдвигая фокус исследований в области искусственного интеллекта на методы машинного обучения, обработку больших массивов данных и сопутствующие технологии. Эта эволюция отличается от подходов, основанных на экспертных системах, которые строятся на формализации знаний экспертов. Машинное обучение, в свою очередь, осваивает инсайты непосредственно из обширных наборов данных, что при-

водит к созданию алгоритмов с высокой степенью адаптивности и применимости в разнообразных контекстах.

2010-е гг.: Watson и DeepMind. В 2011 г. корпорация IBM анонсировала разработку Watson, когнитивной системы, способной интерпретировать запросы, сформулированные на естественном языке, и предоставлять по ним точные ответы. Эффективность и продвинутые аналитические способности Watson были продемонстрированы в ходе телеигры «Jeopardy!», где он смог превзойти своих человеческих соперников. За его работой стоит инновационное сочетание информационного поиска, обработки естественного языка, технологий машинного обучения и алгоритмов строения логических моделей.

В 2015 г. DeepMind, признанный пионер в разработках искусственного интеллекта, представил AlphaGo, революционную программу, сумевшую победить мирового чемпиона по Го, Ли Седоля. Го представляет собой игру несравнимо более сложную по сравнению с шахматами, учитывая гигантское число потенциальных игровых комбинаций, что делает применение традиционных алгоритмов поиска оптимального хода, подобных используемым в Deep Blue, значительно менее эффективным. В основе AlphaGo лежат принципы машинного обучения, благодаря чему программа смогла научиться, анализируя данные из 160 тыс. партий, игранных профессионалами.

2020 г.: GPT-3, AlphaFold. В 2020 г. OpenAI, компания из США, выпустила GPT-3, систему, создающую тексты на английском языке, идентичные человеческим по сохранению контекста, грамматике и др.

В 2020 г. DeepMind анонсировала AlphaFold — технологию, способную предсказывать третичную структуру белков, что долгие годы оставалось одной из наиболее сложных и ключевых задач в биохимии; она была нерешенной на протяжении примерно 50 лет.

Современный статус разработки в сфере ИИ и нейросетей можно кратко описать так.

Прежде всего, касаясь аспекта адаптивности, текущие искусственные интеллектуальные системы, независимо от используемой модели или методологии, предназначены для работы в рамках определенного спектра задач и сценариев и не обладают

возможностью самостоятельного обучения для функционирования в радикально новой среде. Это означает, что системы ИИ находятся под управлением заранее заданного набора инструкций, даже несмотря на то, что сложность их программирования была значительно уменьшена, а их учебные потенциалы расширены.

Второй аспект касается автономности искусственного интеллекта: нынешние системы ИИ значительно зависят от человеческого вмешательства для своей работы. Это включает в себя не только запуск и выключение системы, но и ежедневное обслуживание, установление целей и адаптацию режимов работы в соответствии с различными условиями или задачами. Иными словами, эти системы не достигли полной самостоятельности в управлении. Это наиболее заметно в сферах с высоким уровнем риска или значительной неопределенностью. Примером может служить робот-пылесос, который без человеческого вмешательства не сможет эффективно функционировать даже несколько дней.

Оценивая третий аспект интеграции — умение разных программных элементов работать вместе — делаем вывод, что современные системы ИИ не являются системами с истинным интеллектом (пусть даже ограниченным), а именно снабженные:

– технологиями машинного зрения;

– технологиями обработки естественного языка;

– обработкой данных (искусственного интеллекта);

– обработкой символьных данных (логические выводы на основе информации), и т. д., т. е. не интегрирующими.

Тенденции в развитии искусственного интеллекта и перспективы в этой сфере. Научные работы в сфере искусственного интеллекта продолжаются уже свыше 70 лет и эволюционируют вдоль двух ключевых траекторий: символьного подхода и подхода, основанного на искусственных нейронных сетях.

Логическая модель разрабатывается для создания специализированного (также известного как слабый) искусственного интеллекта, или ИИ, который состоит из компьютерных алгоритмов, задачей которых является выполнение одной или ограниченного набора «умственных» функций.

Нейрокибернетическая стратегия ориентирована на разработку универсального искусственного интеллекта, соответствую-

щего человеческому интеллекту в его способности к решению разнообразных когнитивных задач.

Весь прогресс в области искусственного интеллекта связан с развитием вычислительной техники, включая суперкомпьютеры и квантовые вычислители. Характеристики суперкомпьютеров представлены в табл. 1.

Таблица 1

Список существующих суперкомпьютеров

Название	Мощность, FLOPS	Место расположения	Год
Fugaku	442	Япония	2021
Summit IBM Power Systems AC92	148	США	2022
Sierra IBM Power Systems S922LC	95	США	2020
Sunway Taihulight	93	Китай	2018
Selena NVIDIA	63	США	2020
Червоненкис	22	Россия	2021

1.2. Суперкомпьютеры

Эффективность вычислительных систем измеряется в FLOPS, что означает количество вещественных операций в секунду. PFLOPS, или петафлопс, эквивалентно 10^{15} операции в секунду.

Симуляция 1 с работы 1 % человеческого мозга на китайском суперкомпьютере Sunway Taihulight требует приблизительно 4 мин времени обработки. Этот вычислительный гигант оснащен 10,5 млн процессорных ядер, располагается на площади в 1 000 м^2 и имеет энергопотребление в 16 мВт.

ИИ и вычислительная техника. Пределы развития компонентов микропроцессоров включают:

– максимальную частоту работы до 100 ГГц при текущем достижении около 10 ГГц;

– минимально возможные размеры транзисторов в диапазоне от 3 до 5 нм, в то время как современные технологии позволяют достигать 7–10 нм.

Точность предложенных моделей естественного интеллекта по-прежнему вызывает сомнения.

ИИ и его применение в современных вычислительных системах. Начнем с анализа тех атрибутов искусственного интеллекта и натурального интеллекта, которые поддаются сравнению.

Естественный (человеческий) мозг. Число нейронов в человеческом мозге равно 80×10^9. Число синапсов в мозге человека равно 150×10^{12}. Число молекулярных переключателей на каждом синапсе составляет 10^3.

Общее число молекулярных переключателей (логических элементов) в мозге человека равно $1{,}5\times10^{17}$.

Искусственный («железный») мозг. Мощность, используемая одним логическим элементом микропроцессора за 1 ч при работе на частоте в 5 ГГц, составляет $4{,}8\times10^{-8}$ Дж. Электрическая мощность, необходимая для функционирования металлического эквивалента человеческого интеллекта, через 1 ч составляет произведение $1{,}5\times10^{17}$ на $4{,}8\times10^{-8}$, что равно $7{,}2\times10^9$ Дж. Исходя из того, что 1 МДж равен 0,277 кВт·ч, получаем 0,002 ГВт·ч.

Квантовые вычисления — ключ к развитию продвинутого искусственного интеллекта

В 2001 г. компания IBM представила миру квантовый компьютер, основанный на 7 кубитах.

В 2006 г. был разработан квантовый компьютер на базе 8 кубитов.

В 2011 г. был разработан квантовый компьютер на 16 кубитах.

В 2017 г. IBM объявила о разработке квантового компьютера на 50 кубитах.

В 2018 г. Google проинформировала о разработке квантового компьютера с 72 кубитами.

Между 2007 г. и 2017 г. компания D-Wave Systems (Канада) разработала ряд адиабатических (или аналоговых) квантовых вычислительных машин мощностью от 16 до 2 000 кубитов.

17 июня 2022 г. канадская компания D-Wave Systems заявила о разработке квантового компьютера на 7 000 кубитов.

Ограничения квантовых вычислений:

– высокая чувствительность к помехам, усиливающаяся с увеличением количества кубитов;

– проблемы в обмене данными, так как любое внешнее вмешательство способно нарушить квантовую суперпозицию системы.

Проблема доверия к ИИ

Задачи, требующие решения:

– разработка механизмов интерпретации работы искусственного интеллекта;

– проектирование системы делегирования ответственности за решения в социотехнических комплексах, интегрирующих человеческий интеллект и искусственный интеллект в единую управленческую структуру;

– создание способов структурирования и демонстрации информации через онтологии (выделение неизвестных данных и определение неочевидных связей между компонентами) для их однозначной интерпретации экспертными системами и искусственным интеллектом;

– конструирование интерфейсов для взаимодействия человека и искусственного интеллекта, охватывающее мультимодальные способы общения, включая переработку и передачу нечетко определенных сенсорных данных, таких как эмоции, интуиция, моральные и эстетические ощущения.

1.3. Прогресс в области искусственного интеллекта в России

В нашей стране прогресс в сфере информационных технологий, направленных на поддержку решений, ведет свое начало с 1970-х гг., когда начали разрабатываться экспертные системы, определяющие алгоритмы процесса принятия решений, адаптированных к определенным обстоятельствам.

Машинное обучение сменило эпоху экспертных систем, позволяя информационным системам самостоятельно вырабатывать алгоритмы и искать оптимальные решения путем анализа корреляций в предоставленных данных, исключая необходимость заранее заданных человеком решений. Это является ключевым фактором в развитии искусственного интеллекта.

В апреле 2011 г., после изучения эволюции и применения компьютерных технологий, на Международной промышленной выставке в Ганновере (Германия) представили идею четвертой индустриальной эпохи, известной как Индустрия 4.0.

В 2016 г. был опубликован труд Клауса Шваба под названием «Четвертая промышленная революция», который получил русский перевод в 2018 г. В этой работе аргументированно излагается мысль о том, что население планеты находится на грани нового этапа промышленного развития, способного радикально преобразовать привычные подходы к организации жизни и трудовой деятельности. По мнению Шваба, предстоящие изменения будут иметь беспрецедентную величину, охват и комплексность по сравнению с предыдущими промышленными переворотами, оказывая ощутимое влияние на всех и вся. В тексте приводится анализ основополагающих технологий, лежащих в основе этой революции, и оценивается их потенциальное воздействие на экономику стран, предпринимательскую среду, общественные институты и индивидуальный опыт людей, при этом выделяя значимую роль искусственного интеллекта в этом процессе.

В России внедряется программа «Цифровая экономика РФ», получившая одобрение в ходе собрания президиума Совета под эгидой Президента РФ по вопросам стратегического развития и национальных инициатив.

Цифровая экономика представляет собой экономическую сферу, где основным ресурсом выступают цифровые данные. Интенсивная работа с большими объемами информации и ее аналитическая обработка в сравнении с классическими подходами в экономике значительно увеличивает производительность в областях производства, технологических инноваций, эксплуатации оборудования, управления запасами, маркетинга, логистики и предоставления услуг.

Национальный проект «Цифровая экономика Российской Федерации»[1] содержит девять федеральных проектов:

– «Нормативное регулирование цифровой среды»;
– «Информационная инфраструктура»;
– «Кадры для цифровой экономики»;
– «Информационная безопасность»;

[1] *Национальная* программа «Цифровая экономика Российской Федерации» / Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации (Минцифры). URL: https://digital.gov.ru/target/naczionalnaya-programma-czifrovaya-ekonomika-rossijskoj-federaczii (дата обращения: 01.03.2024).

– «Цифровые технологии»;

– «Цифровое государственное управление»;

– «Искусственный интеллект»;

– «Развитие кадрового потенциала ИТ-отрасли»;

– «Обеспечение доступа в Интернет за счет развития спутниковой связи».

В России был ратифицирован официальный документ «Национальная стратегия развития искусственного интеллекта до 2030 года», закрепленный Указом Президента от 10 октября 2019 г. № 490, направленный на стимулирование развития технологий ИИ, а также Федеральный проект в области ИИ в рамках стратегии «Цифровая экономика Российской Федерации».

Были выдвинуты ключевые термины.

Искусственный интеллект представляет собой всеобъемлющую систему технологий, обеспечивающих моделирование когнитивных способностей человека, включая автономное извлечение уроков из данных и генерирование решений в отсутствие предварительно определенных алгоритмов. Эта система способна выводить результаты в рамках определенных задач, которые по качеству и эффективности находятся на уровне или превосходят человеческий интеллект. Такая система включает информационно-коммуникационные технологии, высокоразвитое программное обеспечение, в том числе алгоритмы машинного обучения, а также процедуры и сервисы, направленные на обработку информации и выработку решений.

ИИ охватывает следующие технологии: машинное обучение, нейронные сети, компьютерное зрение, обработка естественного языка, экспертные системы и системы знаний.

Инновационные технологии в области искусственного интеллекта стремятся к синтезу научно-технических новинок для достижения развитости искусственного интеллекта до уровня сильного ИИ, предполагая самостоятельность в решении множественных задач. Они включают разработку систем с автономным обучением, способностью к самостоятельному проектированию в физическом мире, использованию алгоритмов для работы с нестабильными или ограниченными данными, а также применение перспективных вычислительных архитектур и методов для интерпретаций данных, внося вклад в развитие направления.

Кроме того, 14 июля 2021 г. Минэкономразвития РФ уточнило ключевые аспекты и основополагающие термины в контексте прогрессивных областей применения ИИ.

Сильный искусственный интеллект представляет собой развитую систему ИИ, включающую продвинутые технологии и алгоритмы, способные обрабатывать большой объем данных, на их основе принимать решения и осуществлять их. Это обеспечивает возможность моделирования человеческих интеллектуальных процессов и подачу решений в понятной форме, достигая или даже превосходя множество аспектов человеческого мышления. К таким аспектам относятся анализ внешних данных и их интерпретация, интеграция этих данных для обучения и развития, а также способность к планированию и решению задач в условиях неопределенности, целеустремленно достигая поставленных целей. Это достигается через адаптацию к меняющимся обстоятельствам и взаимодействие с окружающей средой.

Система доверенного искусственного интеллекта, также известная как система надежного ИИ, представляет собой приложение искусственного интеллекта, которое гарантирует успешное выполнение возложенных на него функций, соблюдая при этом ряд специфических условий, повышающих доверие к выдаваемым системой результатам. Эти условия включают следующее:

– надежность и понимание заключений и рекомендаций, выработанных системой и подтвержденных через проверку на верифицированных тестах;

– безопасность — подразумевает предотвращение ущерба пользователям системы на любом этапе ее функционирования, а также обеспечение защиты от кибератак, неавторизованного доступа и иных внешних угроз;

– конфиденциальность и подтверждаемость информации, используемой в системах искусственного интеллекта, в том числе управление правами доступа и прочие сопутствующие аспекты, с учетом этических норм использования ИИ.

Этические принципы в области ИИ означают комплекс стандартов и принципов, сформулированных Центром по разработке рекомендаций, которые направлены на регулирование использования технологий искусственного интеллекта в соответствии с ключевыми задачами Центра. Они заложены для того,

чтобы гарантировать уважение к правам и свободам индивидов, а также их защиту в соответствии с основополагающими положениями Конституции РФ.

Прикладная система искусственного интеллекта представляет собой компьютерную систему, разработанную с целью усиления когнитивных функций человека. Она оснащена инструментами для анализа масштабных массивов данных за короткий промежуток времени. Это позволяет ей не только генерировать решения для сложных задач, но и предоставлять эти решения пользователям в понятной форме. Система может функционировать как в режиме реального времени, так и выполнять задачи в автономном режиме.

Алгоритмы ИИ объединяют подходы для анализа и обработки информации, направленные на выявление закономерностей, понимание контента, распознавание паттернов, прогнозные модели, адаптацию и саморазвитие систем на основе методов машинного обучения, нейронных сетей, глубокого обучения, теоретических принципов когнитивного анализа, эвристического поиска. Применяются в различных областях ИИ для выполнения специфических функций, включая автоматическое обучение и оптимизацию процессов, в научных исследованиях, разработке интеллектуальных систем с целью наиболее эффективного решения задач, предполагающих аналитический подход и адаптацию к меняющимся условиям.

Математические модели в области прикладного ИИ — это структурированные, формально описанные методы представления реальных объектов, действий, явлений и интеллектуального содержания через математический язык, которые обеспечивают основу для разработки алгоритмов, программных решений и комплексов инструментов ИИ. Эти модели эффективно поддерживают процессы обучения машин, взаимодействия с пользователями, логическое мышление, постановку задач и целеполагание, а также принятие обдуманных решений на базе анализа входящих данных. Они позволяют системам ИИ извлекать важные уроки из данных, применять аккумулированные знания для достижения заданных целей, адаптируясь к меняющимся условиям и предлагая пользователю оптимальные варианты решений для реализации целевых задач.

Фреймворк представляет собой комплексное программное решение, которое упрощает процесс разработки, интегрируя различные элементы технологий искусственного интеллекта. Также включает в себя инструменты для автоматизации процессов в облачных системах, нацеленные на повышение эффективности, ускорение выполнения задач и/или настройку алгоритмов искусственного интеллекта в соответствии с требованиями, разработанными в тесном сотрудничестве с клиентом.

Прогресс искусственного интеллекта в России определяется несколькими ключевыми направлениями:

– разработка технологий ИИ в производственной сфере, создание систем искусственного интеллекта для здравоохранения;

– «Биометрические системы на основе ИИ», «Применение ИИ для улучшения управленческих процессов с целью минимизации углеродных эмиссий»;

– «Исследование натуральных языков с применением техник ИИ», «Применение ИИ в задачах развития топливно-энергетического комплекса и энергетической отрасли»;

– «ИИ для „Smart City“ и транспортных систем»;

– «Искусственный интеллект, применяемый в робототехнике и контроле беспилотных устройств»;

– «Применение ИИ в агропромышленном комплексе и пищевой промышленности»;

– «Кросс-секторальные технологии ИИ и применение ИИ в ключевых секторах экономики и социальной области»;

– использование ИИ в области кибербезопасности.

Пример 1. «Применение искусственного интеллекта для эффективного управления, нацеленного на минимизацию углеродных выбросов». Экологические проблемы и стремление к уменьшению углеродного следа становятся основной задачей для экономического развития России. Введение углеродного налога на продукцию, экспортируемую в Европейский Союз, США и Китай, оказывает давление не только на ключевые отрасли, формирующие до 20 % ВВП России, такие как угольная, нефтегазовая и химическая промышленность, но и существенно затрагивает соединенные с ними секторы экономики — финансовую сферу, информационные технологии и социальное обеспечение. Отсутствие адекватных и своевременных решений угрожает серьезной

потерей промышленного потенциала, который Россия наращивала на протяжении последних десятилетий.

Требуется создание эффективных механизмов для наблюдения и уменьшения того объема парниковых газов, что выбрасывается непосредственно различными секторами (например, производством, транспортом и при добыче полезных ископаемых), а также тех, что генерируются косвенно через потребление электричества. Задачи этого калибра включают в себя анализ информации из разнообразных источников, включая данные с сенсоров, изображения с орбитальных аппаратов, видеоматериалы с камер наблюдения и подобное. Это предполагает необходимость в использовании и дальнейшем развитии алгоритмов ИИ.

В контексте методической поддержки в сфере анализа данных для дата-центров настоятельно рекомендуется внедрить систематический подход к определению и управлению ESG-рисками (ESG — экологические, социальные и управленческие), такими как уровни выбросов метана и углекислого газа, инциденты с утечкой нефти и др., с последующим их тщательным мониторингом и контролем на уровнях отдельных предприятий до целых регионов и национального масштаба. Такие системы применяют искусственный интеллект для эффективного сбора, объединения и анализа разнородных информационных потоков, позволяя строить прогнозные модели с гибкой иерархией для точного предсказания. Этот процесс включает в себя и разработку специализированных методологий ИИ, подобных тем, что учитывают физические принципы, и методов, повышающих энергоэффективность ИИ-алгоритмов (ускоренное обучение, оптимизация хранения данных и др.), целью которых является обработка мультимодальных данных.

Реализация новаторских технологических решений в различных секторах экономики обещает не только значительное уменьшение объема выбросов парниковых газов, как прямых, так и косвенных, но и предвещает рост доходов предприятий. Это также способствует сокращению расходов на внедрение ИИ и повышает интерес инвесторов к компаниям. Такой трансфер технологических новинок в производственную сферу, их продвижение и дальнейшее развитие инновационных продуктов могут быть эффективно выполнены предприятиями при наличии профессио-

нального сопровождения от специализированных организаций, таких как инновационные центры.

Исследовательские направления Центра

Направление I. Создание соответствующих приложений искусственного интеллекта для наблюдения, анализа вероятностей и улучшения управления ESG-рисками.

Направление II заключается в разработке инновационных решений для уменьшения углеродного следа и улучшения экологической обстановки, а также в создании специализированных сервисов для промышленности России на базе разработанных средств на основе искусственного интеллекта.

Направление III. Усовершенствование базовых алгоритмов искусственного интеллекта и создание высокопроизводительных программных решений для интеграции (объединения данных) разнородных источников информации (включая данные, полученные с помощью математических расчетов, основанных на фундаментальных законах, информацию с датчиков и данные из космического мониторинга) с целью прогностического анализа изменений в природной среде.

Исследования, проведенные Центром, могут находить применение, включая следующее.

1. В рамках разработки эффективных систем мониторинга окружающей среды выделяется значимость использования искусственного интеллекта для создания интегрированных гибридных решений. Эти системы объединяют в себе данные, получаемые с помощью спутникового дистанционного зондирования планеты, а также современные сенсоры и датчики. Эти устройства обеспечивают передачу данных в реальном времени, что позволяет осуществлять непрерывный мониторинг состояния окружающей среды, анализировать тенденции в изменении уровня выбросов углерода и разрабатывать наиболее эффективные стратегии и мероприятия для сокращения воздействия на климат.

2. Разработка и усовершенствование методов для эффективного изъятия и хранения углекислого газа в сочетании с использованием традиционных энергоносителей (газ, нефть, уголь) и проектов по выращиванию лесов в рамках климатических инициатив, включая интегрированные модели для оценки затрат на захват, транспортировку и последующую переработку/утилиза-

цию/захоронение CO_2 в промышленных масштабах для улучшения финансовой эффективности мероприятий, направленных на минимизацию углеродного отпечатка предприятий.

3. Для улучшения энергетической эффективности и экологичности в сфере нефтяного сервиса, применяется подход, основанный на моделировании и оптимизации ключевых параметров работы промышленных объектов и деятельности по извлечению ископаемых. Это способствует уменьшению влияния человека на природу и снижает вероятность возникновения экологических катастроф.

4. Оценка влияния инвестиций в инфраструктуру и финансирования ESG-перехода, особенно для разработки всесторонних методик определения соответствия организаций принципам устойчивого развития, используя искусственный интеллект и объединяя масштабные разнообразные данные.

Пример 2. Сфера действия: «Применение трансдисциплинарных ИИ-технологий и разработок в сфере искусственного интеллекта для ключевых экономических отраслей и областей социального благополучия».

Центр фокусируется на работе в следующих одной или более специализированных областях.

1. Разработка специализированных платформ на основе мультидисциплинарных ИИ-технологий.

2. Отраслевые платформенные решения объединяют различные программные инструменты и продукты, которые унифицированы по типам входящих и исходящих данных, методологиям и стандартам. Это включает в себя аспекты, связанные с искусственным интеллектом, обеспечивающие возможность адресовать множество специфических отраслевых задач, превосходя функциональность стандартного прикладного ПО. Данные решения, хотя и могут быть разработаны для конкретной отрасли, основываются на универсальных элементах, таких как компоненты, инструменты, методики и алгоритмы ИИ, которые могут находить применение в разнообразных секторах для выполнения технологически похожих задач — это так называемые межотраслевые технологии ИИ. Примеры использования включают в себя разработку и внедрение сложных систем для корпоративного, отраслевого, муниципального и федерального уровня управления.

При разработке платформенных продуктов ключевую роль занимает реализация основополагающих принципов машинного обучения. Эти принципы не только являются краеугольным камнем для разнообразия вопросов в пределах одной отрасли, но и содействуют интеграции идей между разными сферами. Данное взаимодействие стимулирует создание и развитие новаторских технологий, которые охватывают широкий спектр областей, включая создание инновационных материалов, хемоинформатику, нейронауки, автоматизацию и геоинформационные системы, за счет применения искусственного интеллекта.

3. Разработка индустриальных приложений для различных секторов, особенно для ключевых экономических сфер, не охваченных компетенциями специализированных центров, таких как образование, банковские и финансовые услуги, а также публичное администрирование и государственные сервисы. Основная задача таких решений заключается в оптимизации организационных процедур — планирования, анализа, прогнозирования и управления, а также в улучшении сервиса для клиентов посредством персонализированного подхода.

Центр проводит анализ ситуации, определение причин проблемы, разработку стратегий, выбор наилучших решений, планирование и реализацию.

Направления работы Центра:

– разработка искусственного интеллекта при переменных данных для секторных платформ;

– адаптация ИИ-моделей с небольшим объемом обучающих данных для использования в сферах с ограниченным доступом к большим датасетам;

– применение методов машинного обучения в дизайне экспериментов для выбора идеального набора данных для обучения, включая использование в технологиях на стыке дисциплин;

– разработка методов машинного обучения с возможностью интерпретации, особенно для секторов, где стоимость ошибочного решения высока;

– интеграция классического моделирования, оптимизационных алгоритмов и искусственного интеллекта для наилучших итогов, объединяющих доменные теоретические основы с анализом данных;

– предварительный анализ данных, способствующий оптимизации привлечения высококвалифицированных экспертов в домене ИИ и машинного обучения.

В разработке секторальных приложений часто используются методы анализа данных с использованием распределенного интеллекта, имитации динамики систем с множеством элементов, а также средства для помощи в управленческом решении.

Алгоритмы искусственного интеллекта предназначены для работы с широким спектром данных, включая визуальные материалы, тексты, аудиозаписи, а также временные ряды, охватывая при этом и гетерогенные источники данных.

В рамках деятельности Центра крайне важно заложить основу для создания алгоритмов, методик и инструментария, способствующих эффективному решению ключевых проблематик в сфере машинного обучения, которые затрудняют разработку высокопроизводительных приложений и платформ для разнообразных секторов экономики. Применяя достигнутые научные достижения, необходимо способствовать прогрессу в определенных технологических аспектах, особенно в тех областях, где искусственный интеллект представляет собой радикально новый подход к решению задач.

Учитывая широкий спектр и трудности, присущие взаимосвязям в обозначенных областях, а также объемы обрабатываемой информации и высокие требования к ее обработке и безопасности, ключевую роль в разработке и внедрении технологий искусственного интеллекта стали занимать системы, базирующиеся на распределенной обработке данных. Эти системы обеспечивают комплексный подход к анализу информации, моделированию сложных взаимодействий между многочисленными объектами и поддержке процессов принятия стратегически важных управленческих решений.

В процессе деятельности Центра могут быть созданы методики и инструментарий для первичной аналитики данных, применяя технологии искусственного интеллекта, включая модели прогнозирования и рекомендательные системы, которые объединяют классическое математическое моделирование и алгоритмы машинного обучения.

При разработке специализированных решений для конкретных отраслей можно создавать методологии и инструментарий для первоначального анализа данных, применяя технологии искусственного интеллекта. Также разрабатываются стратегии для объединения информационных систем в рамках одной отрасли и для синтеза межотраслевых связей. В этом контексте возникают примеры создания или прототипирования систем прогнозирования и рекомендаций, которые сочетают в себе классические подходы математического моделирования и современные алгоритмы машинного обучения.

Указанные эффекты охватывают всю сферу информационных систем на всех этапах их создания и применения, где используются искусственный интеллект и его технологии. Особенно это касается систем с высокими стандартами безопасности, а также тех, что обрабатывают чувствительные данные, в том числе личную информацию пользователей.

Выводы по главе 1

Сегодня нет единого определения искусственного интеллекта, признаваемого нейробиологами, инженерами-программистами, философами и экспертами из других областей знания. В данной книге мы опираемся на работы российских исследователей.

В обобщенном понимании искусственный интеллект определяется как способность машин выполнять задачи, требующие человеческого ума. В соответствии с трактовкой ученого И. Каляева, искусственный интеллект является характеристикой созданных систем, направленной на решение сложных задач без заранее известного метода решения. Таким образом, по мере того как компьютер находит решение задачи, она перестает ассоциироваться с интеллектуальной деятельностью, предполагая разработку конкретного алгоритма выполнения, ведь компьютерные системы фактически действуют на основе алгоритмов.

Таким образом, подавляющее большинство систем, которые сегодня определяются как искусственный интеллект, фактически не обладают интеллектуальными способностями в полном

смысле этого слова. Они представляют собой программные решения, выполняющие задачи на основе алгоритмов, разработанных человеком.

В области искусственного интеллекта принято различать концепции слабого и сильного ИИ. Под слабым ИИ понимают системы, способные решать специализированные задачи, в то время как сильный ИИ подразумевает системы с когнитивными функциями, схожими с человеческими.

Эксперты в области искусственного интеллекта предсказывают, что разработка сильного ИИ станет возможной примерно в 2030-х гг.

Вопросы для самоконтроля

1. Что предложили Ч. Бэббидж и А. Лавлейс и когда?
2. Какова цель предложенного А. Тьюрингом теста?
3. Когда и кто предложил модель нейрона головного мозга?
4. Какую функции реализуют искусственные нейроны?
5. Место искусственного нейрона в логике высказываний.
6. Кто предложил искусственные сети нейронов и их обучение?
7. Что такое перцептрон Розентблатта и первая реализация сети перцептронов?
8. Когда появился термин «искусственный интеллект»?
9. Является ли наукой искусственный интеллект?
10. Когда появились программные системы, основанные на знаниях? Каковы их структура и цели?
11. Когда появились экспертные системы? Каковы их структура и цели?
12. Когда появился компьютер Deep Blue? Каковы его возможности?
13. Когда была создана система Watson? Каковы ее возможности?
14. Когда появилась система GPT-3? Назовите ее возможности и перспективы.
15. Текущее состояние дел в области искусственного интеллекта.
16. Какие существуют направления исследований в области искусственного интеллекта и каковы их перспективы?
17. Как взаимосвязаны результаты в области разработок искусственного интеллекта и быстродействие компьютеров?
18. В чем состоит проблема доверия к искусственному интеллекту?
19. В чем различия Национальной программы «Цифровая экономика Российской Федерации» и Индустрии 4.0?

20. Место искусственного интеллекта в «Цифровой экономике Российской Федерации».
21. Перечислите основные направления развития искусственного интеллекта в России.
22. Назовите области применения и технологии искусственного интеллекта.
23. Для решения каких задач предназначено машинное обучение?
24. Назовите области применения Data Mining.

Глава 2 Введение в искусственный интеллект и машинное обучение

2.1. Концепция, развивающаяся через использование технологий рационального агента

Мы будем следовать концепции, разработанной Стюартом Расселом и Питером Норвигом [5], которая получила широкое распространение на мировом уровне и применяется в академических кругах по всем направлениям искусственного интеллекта.

Предположим, что любой субъект, осуществляющий действие, рассматривается как Агент (от лат. *agens*, *agentis* — «действующий»).

Мы предполагаем, что понятие агентов в информационных технологиях охватывает широкий спектр способностей, выходящих за рамки обыденных программ вычислительной техники. Ключевыми особенностями агентов являются их способность к самостоятельной работе без постоянного внешнего управления, распознавание и адаптация к условиям вычислительной среды, продолжительная активность без прямого вмешательства, эффективное реагирование на изменения условий и разработка стратегий для достижения заданных пользователем целей.

Определим рационального агента как сущность, выполняющую вычисления таким образом, что позволяет оптимизировать достижение целей, тем самым гарантируя максимально благоприятный исход в условиях неопределенности. В контексте информатики агентом является любой объект, который эффективно воспринимает и взаимодействует со своей (часто изменчивой) окружающей средой.

Функциональное качество (функциональность) информационного агента (далее — агента) состоит из операций (вычислений), которые агент выполняет в ответ на определенный (хотя и не всегда конкретизированный) набор восприятий.

Агент непрерывно анализирует и категоризирует окружающую среду с помощью датчиков и генерирует реакцию, оказывающую влияние на окружение через исполнительные устройства.

Очевидно, определение, относится ли программа к искусственному интеллекту, напрямую зависит от трактовки понятия естественного интеллекта. Учитывая отсутствие универсального определения естественного интеллекта и, как следствие, отсутствие моделей для ЭВМ, совместимых с понятием естественного интеллекта, можно оперировать только категорией технологий ИИ. Сегодня все, что причисляется к искусственному интеллекту, подпадает под определение технологий ИИ, что и будет нашим исходным предположением.

На самом деле разработка искусственного интеллекта (специализированного программного обеспечения для компьютеров) представляет собой процесс создания оптимальной программы агента для специфического использования.

При таком определении процесс разработки ИИ в теории становится выполнимым, поскольку для любого цифрового устройства с областью хранения данных в k бит теоретически существует в точности 2^k вариантов программных реализаций агентов. В этом сценарии для идентификации оптимального варианта агента предполагается возможность перебора каждого варианта вдоль и поперек. Однако данный подход представляется невыполнимым при значительных показателях k.

2.2. Технология агентных систем в области искусственного интеллекта

В области изучения и разработки искусственного интеллекта были разработаны модели, отличающиеся от логической парадигмы.

В контексте нейронных сетей, которые рассматриваются в качестве одной из разновидностей технологий ИИ, существует

гипотеза о том, что человеческий мозг способен адаптироваться к окружающей среде через процесс установления специально адаптированных связей между определенным набором нейронов. В таких системах предполагается, что знание закодировано не через эксплицитные логические формулы, а кроется в неявном виде в форме ассоциативных связей и характеристик, формирующих уникальные конфигурации взаимосвязей между нейронами.

Альтернативная концепция умственной деятельности, вдохновленная биологическим механизмом адаптации видов к изменениям внешней среды, применяет принципы эволюции для решения комплексных задач. Этот подход не опирается на логические выводы; вместо этого создаются генерации конкурирующих решений, которые подвергаются эволюционным процессам: отбору, мутации и генетическому кроссингу. Неэффективные варианты исключаются, в то время как потенциально успешные варианты продолжают развиваться, комбинируя в себе лучшие характеристики предшественников для создания оптимизированных решений.

Социальные системы предоставляют альтернативный метод моделирования интеллекта через общее поведение, способствующее решению задач, недоступных для решения индивидуальными участниками.

Примером взаимодействия в сфере научных исследований служат ученые из академических кругов, промышленности или государственных институтов, которые объединяют усилия для изучения актуальных проблем. Они обмениваются наработками через научные конгрессы и специализированные издания, являющиеся ключевыми каналами коммуникации. Таким образом, они коллективно вкладывают в решение задач, имеющих значимость на общественном уровне. Хотя эти ученые часто действуют в рамках собственных проектов, финансовая поддержка в виде грантов часто направляет их исследования к общей цели.

Сегодня мы наблюдаем обилие теорий и определений, касающихся агентов, а также разнообразие моделей агентных систем и вычислений. Основное внимание уделяется эволюции и текущему положению в области искусственных агентов, а также знаковым работам в этой сфере от таких ученых, как Стюарт Рассел и Питер Норвиг.

Рассмотрим стандартную модель агента, который взаимодействует с окружающей его средой через ряд сенсоров и вносит изменения в конфигурацию этой среды посредством использования исполнительных устройств согласно рис. 2.

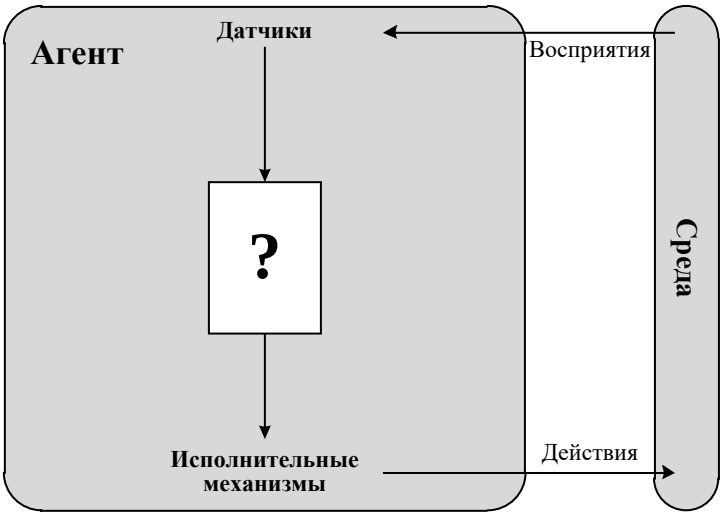


Рис. 2. Общая схема агента

Индивидуум, воспринимаемый как агент, обладает зрением, а также дополнительными сенсорными системами, при этом его экзекутивные функции выполняются конечностями вроде рук, ног, оральным аппаратом и прочими элементами анатомии.

Автономный робот, действующий в роли исполнительного устройства, оборудован для восприятия окружающего пространства видеокамерами и ИК-дальномерами, в то время как для взаимодействия с физическим миром использует моторы различного типа.

ПО, функционирующее как агент, анализирует входящие сигналы, включая коды клавиш, данные файлов и сетевой трафик, и реагирует, отображая информацию на экране, сохраняя файлы и осуществляя передачу данных через сеть.

Предположим, что каждый субъект способен осознавать собственные поступки (хотя и не всегда их исходы).

Важно подчеркнуть, что термин «агент» используется как инструментальное средство в аналитике систем, а не как окончательная схема классификации, разделяющая мир на категории агентов и не-агентов.

К примеру, простой калькулятор, выбирающий операцию и выводящий на экран цифру «6» после обработки ввода «3+3=», может служить примером агента, однако такой подход мало что дает для понимания механизмов работы калькулятора.

Предположим, что рациональный агент осуществляет адекватные действия.

Учитывая, что рациональный агент в итоге представляет собой программное обеспечение и набор данных для вычислительной машины, для рационального агента каждая строка в таблице, описывающей функционал агента, заполнена корректно и не включает в себя никаких ошибок.

Ясно, что совершение адекватных поступков предпочтительнее, чем допущение заблуждений, однако что включает в себя фраза «совершение адекватных поступков»?

В инициальном анализе утверждается, что адекватным действием считается то, что способствует оптимальной работоспособности субъекта действия. Следовательно, необходим метод оценки эффективности. Параметры эффективности в сочетании с характеристиками окружающей среды, а также сенсорными входами и механизмами реализации действий субъекта формируют обширное описание задач перед агентом. Рассмотрение этих элементов позволяет более точно интерпретировать концепцию «рациональности».

В любой заданный момент адекватность поведения агента определяется на основе четырех ключевых элементов, изложенных ниже:

– показатели производительности, которые определяют критерии успеха;

– знания агента о среде, приобретенные ранее;

– действия, которые могут быть выполнены агентом;

– последовательность актов восприятия агента, которые произошли до настоящего времени.

Исходя из вышеизложенного, определим рационального агента как объект, который принимает решения и предпринимает

действия для достижения наилучшего возможного результата на основе накопленных знаний.

Для каждой потенциальной ситуации, возникающей в процессе взаимодействия с окружающей средой, рациональный агент определяет наиболее подходящее действие, при этом ориентируясь на цель максимизации своей эффективности. Это достигается путем анализа возможных вариантов действий, основанных на текущем восприятии среды и всем обширном объеме знаний, ранее собранных агентом. В табл. 2 представлены образцы таких агентов и их характеристики.

Таблица 2

Возможные типы агентов

Тип агента	Показатели производительности	Среда	Исполнительные механизмы	Датчики
Система медицинской диагностики	Эффективное выздоровление больного, сокращение расходов, предотвращение оснований для исков	Пациент, больница, персонал	Генерация запросов, экзаменационных материалов, медицинских заключений, советов, рефералов	Регистрация симптоматики, данных лабораторных анализов и ответов больного в системе
Спутниковая система для обработки получаемых изображений	Аккуратное категоризирование визуального контента	Линия связи между космическим аппаратом и наземной станцией	Отображение результатов классификации конкретного сегмента изображения на экране	Матрицы пикселей, хранящие информацию о цвете
Робот-сортировщик деталей	Процентные показатели безошибочной сортировки по лоткам	Ленточный конвейер с движущимися на нем деталями, лотки	Шарнирный манипулятор и захват	Видеокамера, датчики углов поворота шарниров
Программируемое устройство очистных сооружений	Оптимизация уровня очистки, эффективности и безопасности	Система очистки, персонал управления	Вентили, помпы, обогреватели, экраны	Термометрия, барометрия, анализаторы компонентов

Окончание табл. 2

Тип агента	Показатели производительности	Среда	Исполнительные механизмы	Датчики
Программа динамического обучения английскому языку	Улучшение результатов студентов на тестированиях	Многочисленные обучающиеся, оценочное учреждение	Представление на экране тренировочных заданий, советов, корректировок	Ввод с клавиатуры

В отличие от рассмотренных случаев, определенные программные агенты, иначе именуемые программными роботами или софтботами, функционируют внутри сложных и неограниченных проблемных сред.

Рассмотрим алгоритм автоматического управления, разработанный для симулятора, воссоздающего впечатляюще точную и комплексную виртуальную среду большого пассажирского лайнера. В этой симуляции виртуально воспроизводится активность других воздушных суден и эффективность работы аэродромных служб. Задачей программного модуля является оперативный выбор оптимальных решений из разнобразных доступных опций, основываясь на динамически изменяющихся условиях.

Другим примером служит программное обеспечение-бот, созданное для мониторинга новостных сайтов в Интернете и отображения клиентам выбранных ими новостей.

Для эффективности его работы необходимы специфические навыки в области обработки естественного языка. Это включает в себя возможность в процессе обучения адаптироваться к интересам различных клиентов и гибкость в меняющихся условиях, например, при потере доступа к новостным источникам или появлении новых. Интернет выступает в качестве сложного информационного пространства, сравнимого с реальным миром, где присутствует большое количество искусственных сущностей.

Визуализируем четыре ключевых архетипа агентов, осуществляющих фундаментальные принципы, на которых строится большинство современных интеллектуальных систем и технологий искусственного интеллекта:

– обычные или простые рефлексные агенты (рис. 3);

– рефлексные агенты, основанные на модели (рис. 4);

– агенты, основанные на цели (рис. 5);

– агенты, основанные на полезности (рис. 6).

Рассмотрим методы преобразования агентов всех представленных категорий в агентов, способных к обучению.

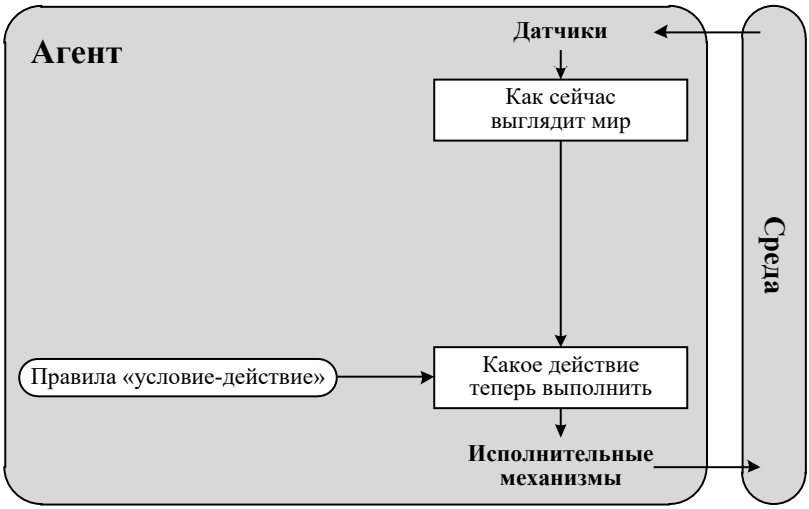


Рис. 3. Структура простого рефлексного агента

Обычные / простые рефлексные агенты просты в использовании и высоконадежны, однако обладают ограниченными когнитивными возможностями. Они вырабатывают действие только на основе текущего акта восприятия, игнорируя всю остальную их историю.

Рефлексные агенты, основанные на модели. В режиме частичной прозрачности среды одним из ключевых элементов эффективного функционирования агента является его способность отслеживать ту часть окружающей среды, которая в данный момент доступна его перцептивным способностям. Это подразумевает, что агент должен поддерживать актуализированное внутреннее моделирование реальности, которое формируется на основе накопленного опыта взаимодействия с окружающей средой и способно компенсировать дефицит непосредственно доступной информации. Для построения и корректировки внутреннего пред-

ставления о состоянии мира в алгоритмах агента используется два основных типа знаний: одни отражают изменения в среде, не зависящие от действий агента, в то время как другие отображают, как взаимодействие агента со средой изменяет последнюю. Основываясь на первом типе знаний, строится модифицированная модель среды (рис. 4).

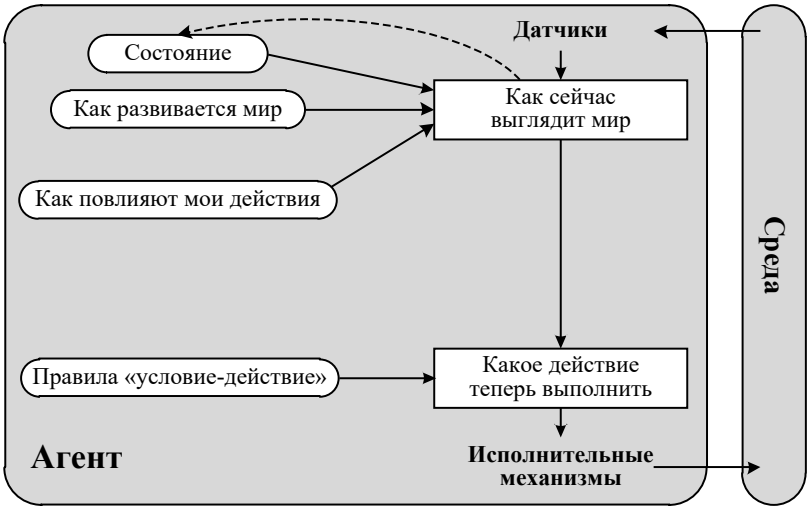


Рис. 4. Структура рефлексного агента, основанного на модели

Агенты, основанные на цели. В реализации решений в сложных ситуациях информация из внешнего мира часто оказывается недостаточной. Рассмотрим сценарий, где индивид стоит перед выбором одного из трех возможных путей на перекрестке, где решение напрямую зависит от конечной цели. Это подразумевает, что для эффективного выбора действий необходимо учитывать не только внешние условия и состояние агента, но также и его целевые установки, определяющие предпочтительные исходы. В таком контексте алгоритмы искусственного интеллекта агента могут интегрировать разнообразные данные, включая цели, для формирования стратегии достижения заданных результатов.

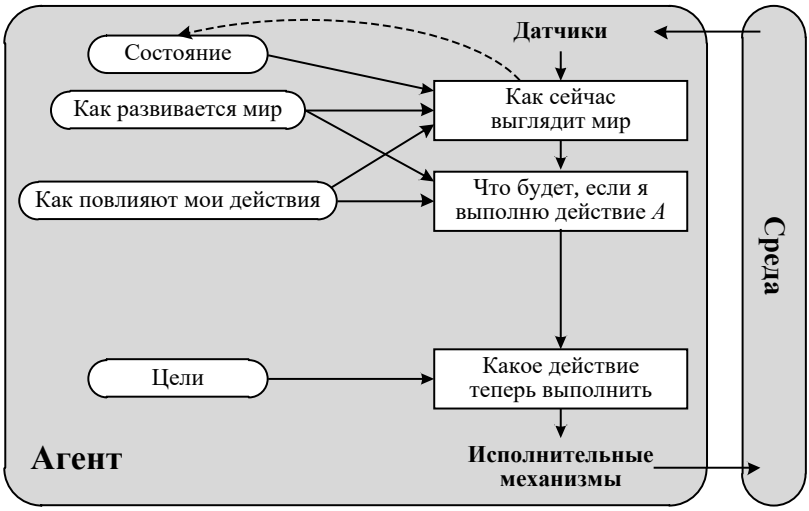


Рис. 5. Структура агента, основанного на модели и на цели

Агенты, основанные на полезности. В реальности для создания качественного поведения учет только целей не всегда достаточен. Определение целей приводит к примитивной дихотомии между «удовлетворением» и «неудовлетворением», в то время как детализированные метрики эффективности нужны для дифференциации между разнообразными мировыми состояниями на основе степени потенциального удовлетворения агента от их достижения.

Функция полезности преобразует определенное состояние или серию состояний в реальное число, выражающее уровень удовлетворенности субъекта.

Детальное определение функции полезности позволяет рационально решать задачи в двух следующих контекстах, когда цели не дают этого сделать:

– когда сталкиваются противоречивые стремления, когда достижимы лишь некоторые из них (как выбор между скоростью и безопасностью), функция полезности способствует идентификации приемлемого компромисса;

– когда агент преследует множество целей, достижение которых не гарантировано, функция полезности служит эффектив-

ным инструментом для оценки вероятности успеха, учитывая значимость каждой цели.

На рис. 6 представлена общая структура агентов, функционирующих на базе принципа полезности.

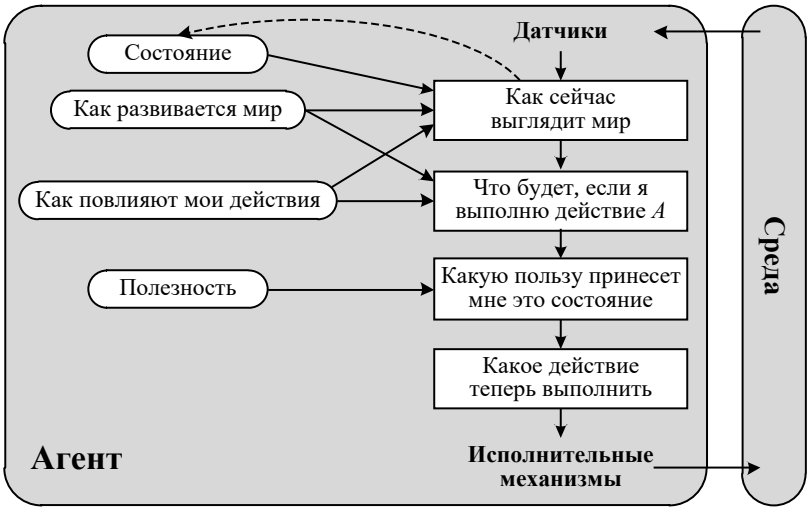


Рис. 6. Структура агентов, основанных на полезности

Обучающиеся агенты. Обозначенные типы агентов объединяет ключевая проблема: их неспособность к обучению, что представляет собой значительный недочет в контексте интеллектуальных систем. Элементы, необходимые для процесса обучения, представлены на рис. 7.

Интеллектуальный агент состоит из четырех основных частей. Роль критика состоит в анализе действий агента на основе установленного критерия качества. Этот компонент критически важен, так как без него агент не способен самостоятельно оценить результативность своих действий. Критерий качества должен оставаться неизменным. Компонент обучения занимается внедрением улучшений на основе обратной связи, в то время как исполнительский компонент выбирает действия во внешней среде. Обучающий элемент использует оценки критика для коррекции и определения будущих действий агента. Этот процесс обучения

тесно связан с исполнительным компонентом. При проектировании такого агента ключевым является понимание того, какой исполнительный компонент окажется наиболее подходящим после завершения процесса обучения агента.

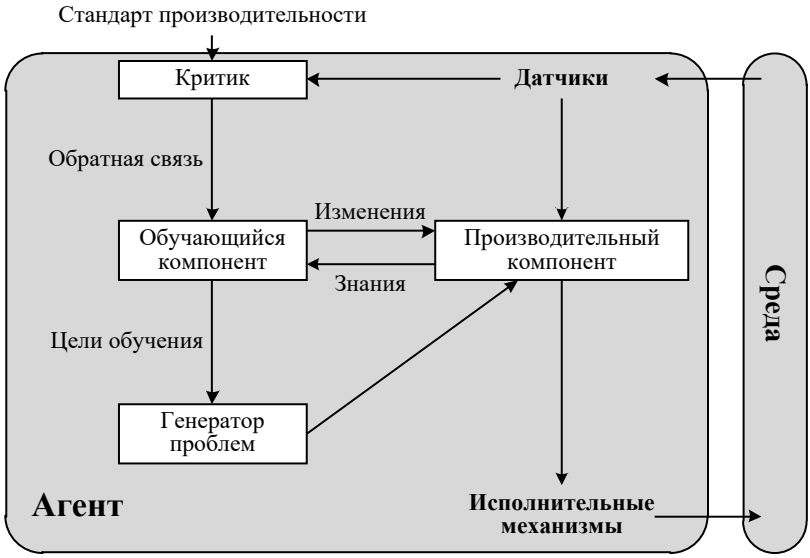


Рис. 7. Общая модель обучающихся агентов

Генератор проблем имеет задачу идентификации стратегий, ведущих к получению уникального информационного опыта. Производственный компонент, отбирая лишь наиболее эффективные методы, может стать жертвой рутины, постоянно повторяя их в предположении их оптимальности. В этом контексте генератор проблем выступает как механизм для инициации начальных, возможно, менее эффективных стратегий, которые, тем не менее, могут привести к наилучшим исходам. Этот процесс обеспечивает системе возможность для экспериментов и открытия наиболее подходящих решений.

Теперь можно уточнить или выделить следующие элементы.

Агент представляет собой сущность, способную к восприятию и взаимодействию в некотором контексте. Его задача заклю-

чается в определении реакции на серию восприятий через конкретное действие.

Метрики эффективности измеряют действия агента в контексте. Разумный агент стремится оптимизировать свои результаты по этим метрикам, основываясь на цепочке воспринятых событий до текущего времени.

Концепция проблемной среды охватывает критерии эффективности, характеристики внешних условий, устройства выполнения и сенсоры. Началом разработки агента должно стать всестороннее выявление характеристик проблемной среды.

Проблемные среды могут быть категоризированы в соответствии с рядом критических характеристик. Сюда входит их полная или частичная наблюдаемость, определенность или вероятностная природа, однократность или последовательность задач, неизменность или изменчивость с течением времени, структура данных (дискретная или непрерывная), а также — включают ли они одного агента или множество агентов в их динамике.

Агентская программа выполняет функционал агента, включая множество базовых архитектур, которые соответствуют специфике обрабатываемой информации, критической для процессов принятия решений. Разнообразие архитектур обусловлено их различной производительностью, объемом и адаптируемостью. Оптимальный выбор конкретной агентской программы обусловлен спецификой окружающей среды.

Реактивные агенты принимают решения, опираясь исключительно на текущие восприятия, в то время как модельные рефлексные агенты формируют и поддерживают внутреннее представление окружающей среды, позволяя отслеживать изменения, которые не заметны в данный момент. Целеориентированные агенты стремятся выполнить определенные задачи или достигнуть заданных целей, в то время как агенты, максимизирующие полезность, руководствуются стремлением максимально повысить ожидаемую получаемую выгоду или удовлетворение от своих действий.

В технологическом контексте различные виды агентов склонны модифицировать и оптимизировать свою функциональность с использованием образовательных ресурсов.

2.3. Основы машинного обучения — концептуальный подход

Данные

Машинное обучение зависит от наличия данных, которые необходимы для корректировки алгоритмов. Альтернативой являются экспертные системы, где прогнозы основываются на профессиональных знаниях экспертов, однако они часто уступают машинным алгоритмам в точности. При реализации процессов автоматизации и точного прогнозирования в сферах, где компания имеет длительный опыт (например, в колл-центрах, кредитном скоринге или анализе спроса), предполагается, что за это время был собран значительный объем данных, пригодных для машинного обучения.

Иногда в компании отсутствует необходимая информация из-за того, что ее либо не архивировали, либо задача никогда не ставилась. В этих ситуациях решением может стать поиск публичных данных в сети или приобретение/заказ специфических данных.

Еще одной потенциальной трудностью является обеспечение безопасности: хотя данные и присутствуют в организации, предоставление доступа к ним разработчикам может быть недопустимо. Необходимо рассмотреть меры по урегулированию данных вопросов еще до старта проекта.

Также критическое значение имеет объем информации: большее количество данных повышает точность алгоритма, а сложные задачи требуют большего объема информации.

Ресурсы. Тренировка алгоритмов машинного обучения зачастую задействует ресурсоемкие вычислительные операции. Работа с небольшим объемом данных остается выполнимой на домашнем ПК, однако анализ информации, охватывающей миллионы потребителей, требует выделенного сервера с высокими характеристиками. Даже при его использовании время обработки может простираться от нескольких дней до целых недель.

Большие компании обычно приобретают серверное оборудование и привлекают специализированный персонал для его эксплуатации, в то время как малые предприятия могут предпочесть аренду виртуальных серверов, доступ к которым осуществляется через Интернет.

Специалисты. Для использования методов машинного обучения требуются определенные знания, поэтому конфигурирование модели предпочтительнее поручить эксперту в данном сегменте.

Большие компании часто формируют команды из экспертов в области Data Science, что является мультидисциплинарной сферой, объединяющей знания статистики, математического анализа, информатики и основ искусственного интеллекта, предназначенной для анализа и обработки массивов данных. В этом контексте заказчик предоставляет ученым по данным необходимую информацию, детализирует проблему и определяет ключевые показатели эффективности, в то время как специалисты занимаются разработкой и возможной интеграцией моделей. Однако создание простейших моделей на основе ограниченного количества данных доступно даже начинающим, которые освоили исходные принципы через базовые онлайн-курсы. В таком случае критически важно, чтобы обрабатываемые данные были гомогенными, высококачественными и свободными от ошибок.

Следует осознавать, что профессионалы в домене машинного обучения часто не обладают глубокими знаниями в конкретной сфере применения технологий, из-за чего для достижения целей проекта критически важно, чтобы заказчик был осведомлен о ключевых принципах и ограничениях использования машинного обучения. Это облегчит взаимопонимание и поможет в эффективном разрешении возникающих препятствий.

Метрики. Важно иметь четкое представление о целях решаемой задачи, определить ключевые показатели (метрики) для оптимизации, такие как точность прогнозов, уровень удовлетворенности клиентов и прибыльность, а также установить приемлемые для бизнеса значения данных метрик.

Регулярно высокие оценки по ключевым показателям эффективности (KPI) моделей машинного обучения не всегда означают значительный успех в деловой среде. Критическим барьером для интеграции технологий машинного обучения в стратегические бизнес-процессы является диссонанс между техническими метриками, используемыми разработчиками алгоритмов, и операционными показателями, на основе которых принимают решения руководители высшего звена.

Даже при обещаниях математика о росте прибыли важно распознать долю улучшений, пришедших благодаря алгоритмам машинного обучения, в отличие от улучшений, вызванных внешними переменными. Также важно учесть, что достижение определенных показателей эффективности может обходиться слишком дорого.

Примерно 50 % инициатив в области искусственного интеллекта завершаются неуспехом, что объясняется завышенными ожиданиями, дефицитом качественных данных и сложностями в data management.

К сожалению, даже при обеспечении всех предпосылок для успеха, включая обширные объемы данных, достаточные вычислительные мощности и оптимизацию алгоритма, проект все еще может закончиться неудачей, не достигнув целевых показателей качества.

Здесь имеется в виду независимость переменных: когда на основании входных данных невозможно спрогнозировать результаты.

Ситуации подобного рода часто возникают на рынках ценных бумаг, где изменение индексов определяется обширным набором параметров, делая процесс прогнозирования исключительно запутанным. Тем не менее, в сфере финансовых торгов интенсивно используются разработки в области искусственного интеллекта.

Ключевые задачи машинного обучения охватывают применение процессного фреймворка CRISP-DM для анализа и интерпретации данных.

Диаграмма на рис. 8 представляет стандартный процесс для анализа данных из разных отраслей, CRISP-DM (Cross-Industry Standard Process for Data Mining), охватывающий следующие этапы.

1. Понимание бизнес-целей.
2. Начальное изучение данных.
3. Подготовка данных.
4. Моделирование.
5. Оценка.
6. Внедрение.

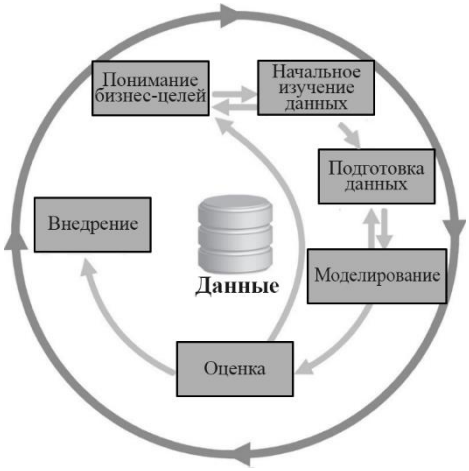


Рис. 8. Модель процесса интеллектуального анализа данных CRISP-DM

Понимание бизнес-целей является начальным этапом в обработке информации, где критически важно осмыслить цели сбора и анализа информации, а также определить, какие именно данные требуются для достижения поставленных целей.

Формулирование целевых установок и первичных предположений по анализу информации станет фундаментом для реализации проекта.

В рамках этапа «Понимание бизнес-целей» первоочередной задачей является уточнение стратегических целей компании. Далее следует анализ настоящего положения дел организации, выявление и формулировка конкретных целей для дальнейшего аналитического исследования. Завершающим этапом является разработка подробного плана предстоящего проекта.

На этапе **начального изучения данных**, который является вторым шагом в процессе обработки информации, ключевой задачей является анализ качества используемого массива данных. Это включает в себя проверку на полноту информации, выявление ошибок, недостатков и пропущенных значений. Необходимо оценить характеристики имеющихся данных, сформулировать по ним вопросы и разработать предварительные гипотезы относительно возможных закономерностей, скрытых в данных.

На этапе **подготовки данных** осуществляется создание конечного датасета для анализа. Этот процесс включает очистку данных от аномалий и ошибок, а также стандартизацию форматов, преобразуя первоначально разрозненные и многоформатные данные в унифицированный вид. В рамках этого этапа задачи, такие как селекция данных (выделение необходимых таблиц, записей, атрибутов), очистка данных, включая конвертацию и подготовку к анализу, создание производных значений, агрегация и форматирование данных, могут выполняться итеративно в любой последовательности без строгого предварительного планирования.

Моделирование — на данном этапе требуется определение подхода для анализа данных, создание аналитической модели.

Модель должна адекватно отображать весь алгоритм аналитического исследования: от того, какие цели вы ставите перед сбором данных, до спецификации используемой информации, ее структурированности, методов обработки и т. д. В процессе работы вам может потребоваться возврат к этапу предварительной обработки данных, поскольку разные аналитические подходы предъявляют к данным специфические требования к формату.

Этап моделирования включает в себя выбор подхода к моделированию, разработку тестов для проверки модели, создание самой модели, а также ее оценку.

Оценка — это процесс выявления, успешно ли были реализованы поставленные задачи через применение созданной модели и анализа выходных данных.

Этот этап позволяет проверить соответствие ожидаемых результатов изначальным целям. В процессе такого анализа можно обнаружить ключевые задачи бизнеса, которые ранее были пропущены. Цели этапа оценки включают анализ достигнутых результатов, критический осмотр примененных методик работы и определение направления дальнейших действий.

Внедрение — процесс может быть легким или трудоемким, в соответствии с задачами компании. В основном это включает создание и применение решений, полученных через анализ данных. Это может охватывать подготовку отчетности или автоматизацию процедур, направленных на достижение ваших задач. На этапе внедрения необходимо: разработать план деплоймента; составить план поддержки и наблюдения за функционированием

системы; подготовить итоговую документацию; провести оценку и анализ реализованного проекта.

Изучим более детально различные элементы процессов подготовки и анализа данных, а также инструментарий для обработки данных и методы построения моделей.

Сбор данных

На данном этапе критическое значение имеет поиск информации для создания запроса:

– что ищем;

– запрос советов для поддержки в нахождении ресурсов поисковых данных;

– самостоятельный поиск;

– запрос и получение данных.

Формулировать вопросы нужно с особым вниманием к контексту задачи, учитывая тип необходимых данных и четко определяя рамки запроса.

Где консультируемся?

– среди профессионалов в специализированной сфере;

– в консалтинговых компаниях;

– у органов власти;

– в социальных сетях.

Где ищем сами?

– ГИС;

– консалтинговые компании;

– научные порталы и учреждения;

– информационные ресурсы отрасли;

– специализированные союзы и альянсы представителей власти;

– СМИ.

Как запрашиваем данные?

– запрос предложения бизнес-услуг;

– обращение за информацией к государственным учреждениям, ссылаясь на Федеральный закон от 9 февраля 2009 г. № 8 «Об обеспечении доступа к информации о деятельности государственных органов и органов местного самоуправления»;

– автоматическое извлечение информации (data scraping);

– доступ к открытым данным, приобретение информации через публичное предложение.

Хранение данных

Хранение информации представляет собой методику, обеспечивающую доступность данных, их неприкосновенность и надежную безопасность.

Существует множество методов хранения информации:

– чтобы извлечь информацию с твердотельного накопителя, будь он съемного или встроенного типа, требуется физический доступ к самому устройству хранения данных или к устройству, в котором он установлен;

– сервера баз данных;

– облачное хранилище позволяет хранить данные в Интернете, обеспечивая доступ к ним с любой точки мира.

Определение метода сохранения информации обусловливается размером данных, требуемой скоростью их извлечения, регулярностью обновления информации, численностью пользователей с разрешением на доступ к ней, а также затратами на хранение заданного объема информации.

База данных представляет собой централизованный метод организации информации. Системы управления базами данных (СУБД) обеспечивают операции чтения, записи, модификации и удаления данных. SQL Server 2019 рекомендуется ввиду его возможностей аналитики больших данных и интеграции машинного обучения.

Обработка данных

Под подготовкой данных подразумевается специализированный набор процедур обработки данных, направленных на выделение дополнительных знаний путем переоценки и корректировки полученных данных, аналитических выводов, расчетов и т. д.

В начальной фазе процесса анализа данных происходит их первичная обработка, включающая стандартизацию форматов, выявление ключевых атрибутов и организацию данных в структурированный порядок. После этого определяется оптимальная модель обработки данных для решения поставленной задачи:

– селективная обработка активных задач означает выполнение операций с отдельными категориями;

– реально временная потоковая обработка данных означает выполнение аналитических операций с массивами информации,

которые постоянно поступают, обновляя исходные результаты анализа при каждом приходе нового набора данных;

– обработка данных в пакетном режиме, собранных за установленный период времени.

В соответствии с выбором модели для решения определенной задачи проводится отбор технологий и типов баз данных, предназначенных для наиболее эффективного использования в данном контексте.

К методам управления данными относят: генерацию данных; их изменение; извлечение сведений; анализ для выбора вариантов; формирование отчетности; составление документации; укрепление защиты информации.

В процессе анализа данных акцентируется внимание на качественных характеристиках информации.

В анализе данных принято различать данные «чистые» и «грязные». Отличительными характеристиками «грязных» данных являются присутствие посторонней информации, не относящейся к исходным данным, предварительная обработка и дефицит исходного материала. Эти факторы затрудняют процесс анализа, поскольку «грязные» данные уже включают в себя элементы аналитической обработки, избавиться от влияния которых невозможно.

Визуализация данных

Визуализация данных — это методика отображения информации в собранном и легко усваиваемом формате.

Визуализация может быть:

– для публичной презентации перед зрителями;

– исследовательской — подготовленной к получению первичных итогов анализа данных.

Визуализация играет ключевую роль на различных стадиях обработки данных, включая отображение начальных наработок, демонстрацию промежуточных выводов, а также представление итоговых результатов.

Из-за большого количества данных визуализация становится обязательным методом преобразования данных в формат, доступный для понимания. Следовательно, инструменты визуализации играют ключевую роль в обработке данных.

Виды визуализации данных:

– линейные диаграммы, диаграммы рассеяния и другие виды графиков;

– графики (вертикальные столбцы, секторный, ступенчатый, донат, радиальный, визуализация ключевых слов и пр.);

– инфографика;

– схемы;

– презентации;

– картография охватывает изображение местности (фотокарты), земные поверхности (геокарты), транспортные маршруты (дорожные карты), специфические данные (тематические карты), статистическое распределение (картограммы);

– дашборды — это комплексные информационные панели, которые не только представляют данные наглядно, но и осуществляют их глубокий анализ. Они, например, позволяют студентам отслеживать свои академические достижения, затем сравнивать их с успеваемостью сверстников и анализировать общую статистику по предметам;

– иллюстрации.

Анализ метаданных подхода CRISP-DM

Методика состоит из шести шагов: анализ бизнес-процессов; предварительный анализ данных; их обработка; построение моделей; оценка результатов; интеграция.

Процесс извлечения информации состоит из следующих шагов: создание запроса; обращение за помощью; независимый поиск; отправка запроса; получение результатов.

Данные могут храниться на жестких дисках, в серверных системах или в облачном хранилище.

Обработка информации охватывает исходную обработку и очистку, определение ключевых атрибутов, сжатие данных, а также подбор подходящей модели для анализа.

Обработка информации — набор процессов, которые ученый проводит для того, чтобы вывести конкретные выводы о свойствах изучаемого объекта на основе рассматриваемой информации.

Визуализация данных — это методология отображения информации.

Искусственный интеллект обозначает проблематику. В дисциплине машинного обучения созданы алгоритмы для поиска решений типизированных заданий, обладающих определенными параметрами на входе и желаемыми результатами на выходе, где важно четко определить эти параметры и результаты для конкретного случая. Часто встречаемой задачей здесь является классификация.

Например, мы опубликовали вакансию, но кандидатов на должность слишком много. Чтобы сэкономить время на просмотр резюме, разработаем софт, задача которого — сегментировать поданные резюме, отфильтровывая их на основе специфичных характеристик. Это разделяет претендентов на группы: квалифицированные и неквалифицированные. В сфере искусственного интеллекта такой процесс определяется как классификационная задача.

Разработаем алгоритмическую последовательность действий, обеспечивающую анализ резюме соискателей на предмет наличия в них критически важных требований для вакансии. В случае обнаружения этих параметров резюме будет автоматически перенаправлено в директорию «На собеседование», а при их отсутствии — в директорию «Отклонено».

В настоящий момент на ЭВМ возлагается задача задать параметры, по которым будет осуществляться выбор.

К примеру, предоставим алгоритму набор резюме, подготовленных вручную, служащих тренировочным набором данных.

В данном наборе данных присутствуют как одобренные заявители, так и те, кто был отклонен.

Эффективность алгоритма оценивается с применением тестового набора данных, состоящего из аннотированных резюме.

В тестовом наборе данных каждому резюме присваивается метка истинности в зависимости от точности алгоритма: True, если классификация верна, и False, если ошибочна. Алгоритм категоризирует резюме на две группы: подходящие (позитивные) и неподходящие (негативные).

Различные комбинации данных параметров и категорий приводят к четырем видам исходов:

– TP (True Positive) — это когда алгоритм корректно идентифицирует резюме как соответствующее требованиям;

– TN (True Negative) означает, что алгоритм корректно классифицировал резюме как несоответствующие требованиям;

– FP — ложноположительный результат, когда алгоритм неверно идентифицирует резюме как подходящее, хотя в нем отсутствуют необходимые качества;

– FN, или ложноотрицательный результат, возникает, когда алгоритм неверно отвергает адекватное резюме.

Через указанные критерии мы в состоянии вычислить специфические показатели, дающие возможность анализировать эффективность алгоритма. Ключевым является выбор самого соответствующего показателя.

Наиболее базовый показатель эффективности модели — это точность, измеряемая количеством случаев, когда предсказания алгоритма соответствуют истинным значениям в контрольном наборе данных.

Метрика эффективна, когда равное число адекватных и неадекватных резюме получено.

В этом контексте данная метрика оказывается неэффективной: очевидно, что число подходящих претендентов ограничено, и производительность алгоритма в большей мере определяется аккуратным подбором кандидатов, приглашенных на интервью.

Такая метрика, как точность, отражает пропорцию правильно идентифицированных подходящих резюме к общему числу кандидатов, выбранных ЭВМ для собеседования. Это значит, что среди приглашенных ЭВМ могут быть кандидаты, не соответствующие требованиям.

В дополнение к точности существует также метрика, измеряющая полноту. Полнота отражает процент правильно идентифицированных соответствующих резюме от общего количества резюме, которые, по оценке программиста, заслуживают быть приглашенными на собеседование. Это указывает на возможность компьютером пропустить подходящих кандидатов.

С учетом того, что высокие показатели точности и полноты одновременно повышают качество отбора кандидатур, достижение идеальных значений этих параметров в практике оказывается задачей со множеством сложностей, требующей нахождения компромисса. В этом контексте широко применяется F-мера, вычисляемая как гармоническое среднее точности и полноты, что поз-

воляет достигнуть баланса между этими показателями, минимизировав влияние общего числа релевантных и нерелевантных объектов. Данный подход и его аппликация детально освещены в статье «Оценка классификатора — точность, полнота, F-мера»[1].

Этот этап образовательного процесса может быть завершен или же продолжен, ведь с увеличением объема информации точность выводов, которые мы получаем, повышается.

2.4. Данные машинного обучения

Табличные данные

В области машинного обучения зачастую применяются данные в таблицах, которые также известны как структурированные данные.

В Excel-таблице и SQL Server 2019, который поддерживает функции машинного обучения, существует возможность интеграции машинного обучения напрямую благодаря встроенной поддержке языка R. В коммерческой сфере расширение этого языка известно как RevoScaleR, что усиливает его аналитические возможности.

Разработана платформа для машинного обучения на основе данных, хранящихся в SQL Server, с последующей ее интеграцией с Python. Это расширило функциональность до того, что R и Python теперь функционируют как единые сервисы машинного обучения в рамках SQL Server (SQL Server Machine Learning Services).

SQL Server ML Services расширяют функциональность, предоставляя средства для разработки, развертывания и запуска приложения в рабочей среде масштабируемых моделей машинного обучения, основываясь на следующих принципах:

– приложения, использующие машинное обучение, работают на одном и том же устройстве, где размещен SQL Server, однако они выполняются в процессах, которые функционируют независимо от основного процесса SQLSERVR.EXE;

---

[1] *Оценка* классификатора (точность, полнота, F-мера) / bazhenov.me. URL: https://www.bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html (дата обращения: 01.03.2024).

– SQL Server предоставляет интерфейс T-SQL посредством системной хранимой процедуры sp_execute_external_script для выполнения кода машинного обучения;

– SQL Server предоставляет архитектуру для интеллектуального обмена данными и масштабируемости, используя специальный программный код для машинного обучения.

В таблице данных объекты располагаются по строкам, представляя собой базисные единицы данных, для которых требуется осуществить прогнозирование.

В коммерческих проектах центральным элементом часто выступает покупатель, но объектом анализа могут быть различные сущности: компания (прогнозирование финансового краха), продукция (классификация по категориям), комментарий потребителя (анализ сентимента отзыва).

В анализе данных встречаются объекты повышенной сложности, такие как, например, в прогнозировании спроса, где объект анализа представляет собой комбинацию «продукт — временной период».

Колонки в таблице представляют атрибуты сущностей — свойства, которые определяют уникальные черты каждой сущности и выделяют ее среди остальных.

В анализе клиентов банковских учреждений рассматриваются такие ключевые характеристики, как ежемесячный доход, возрастная категория, занимаемая должность, степень образования, место проживания и продолжительность профессионального опыта. Для компании характерными особенностями являются дата создания, объем уставного фонда, форма собственности, объем продаж за год, доходность. Обычно значение атрибута выражается численно, как, к примеру, размер зарплаты клиента, составляющей 50000. Применяются и альтернативные типы характеристик, для которых создают методики трансформации в числовую форму (табл. 3).

Эффективность алгоритмов машинного обучения напрямую зависит от объема доступных данных. К примеру, в информационных системах банковской сферы могут быть аккумулированы данные о множестве клиентов; иными словами, таблицы данных могут содержать миллионы записей, что представляет существенный объем информационных массивов.

Таблица 3

Информация для обучения ИИ

Заработная плата, ден. ед.	Возраст, годы	Должность	Уровень образования	Город проживания	Стаж работы, годы	Вернет ли клиент кредит
100000	26	Риелтор	Высшее	Санкт-Петербург	5	Да
50000	20	Продавец-консультант	Высшее	Москва	1	Нет
35000	39	Автомеханик	Среднее специальное	Воронеж	8	Нет
25000	23	Программист	Высшее	Самара	2	Да
75000	41	Юрист	Среднее	Москва	14	Да

Наоборот, в организации здравоохранения могут присутствовать данные только об ограниченном числе больных, скажем, нескольких сотнях, что представляет собой ограниченный набор информации, и на этой основе эффективно настроить алгоритм может оказаться затруднительно.

Число атрибутов — колонок в датасете — может колебаться, обычно находясь в диапазоне от десятков до сотен.

Алгоритм классификации

После выбора типа данных следующим шагом будет формулировка задачи классификации.

В данной задаче классификации объекты (записи в исходном датасете) принадлежат к определенным категориям или классам, выбранным из фиксированного списка. Например, в анализе прогнозирования процента пользователей, который теряется за определенный период, объект для анализа представляет собой клиента, и существует две возможные категории результата: «клиент ушел» и «клиент остается», причем каждый клиент может быть отнесен только к одному из этих классов.

В задаче классификации клиентских отзывов каждый отзыв рассматривается как отдельный объект, который требуется распределить по предопределенным категориям, таким как «жалоба», «предложение», «позитивный отзыв», гарантируя, что каждый из них будет отнесен к соответствующему классу.

Классификационные задачи в области машинного обучения способствуют созданию алгоритмов, которые анализируют харак-

теристики объектов для прогнозирования принадлежности к определенной категории, например, предсказание вероятности отказа клиента от услуги.

Базовая модель прогнозирования оттока клиентов может быть сформулирована следующим образом: при условии, что клиент начал взаимодействие с компанией после 2023 г. и зафиксировал менее трех транзакций в течение прошедшего года, вероятно, этот клиент прекратит пользоваться услугами. В противном случае предполагаем, что клиент останется.

В другом выражении, учитывая атрибуты клиента, такие как «год первого обращения» и «количество транзакций за последние 12 месяцев», мы определяем методику для прогнозирования принадлежности к определенной категории. Этот метод, или алгоритм, должен исполняться на компьютере в полностью автоматизированном режиме, следовательно, он требует высокой точности формулировки.

Конечно, программа может обладать более высокой сложностью, включать множественные условные операторы, выполнение вычислений на основе атрибутов (как пример, агрегирование значений атрибутов) и пр., обрабатывая при этом сотни атрибутов, но всегда должна подчиняться строго определенной спецификации.

Возникновение вопроса о разработке алгоритма для классификации поведения клиентов подразумевает интерес к механизмам принятия решений экспертами в предсказании оттока. Опрос профессионалов об их методиках диагностики и последующая трансляция этих методик в программный код представляют собой один из подходов к построению такого алгоритма.

Этот метод известен как система искусственного интеллекта, специализирующаяся на знаниях.

Тем не менее, методика обладает рядом ограничений: консультации с экспертами могут выявить несоответствия во мнениях, их компетенция может ограничиваться взаимодействием с определенной категорией клиентов, без осведомленности о потребностях других групп, и, наконец, их знания и навыки могут быть вызовом для стандартизации в форме детальной программы, особенно когда решения основаны на интуитивном подходе.

Машинное обучение предоставляет метод, исходящий из анализа данных: используя обширную базу примеров поведения клиентов, включая тех, кто прекратил взаимодействие, система самостоятельно разрабатывает алгоритм для прогнозирования того, останется ли новый клиент или уйдет. Это означает, что на фундаменте собранных данных программа проектирует другую программу: первая идентифицируется как алгоритм обучения, вторая — как алгоритм прогнозирования.

За счет обучения на обширном наборе реальных случаев алгоритм накапливает достаточно опыта для эффективного прогнозирования поведения новых клиентов (рис. 9).

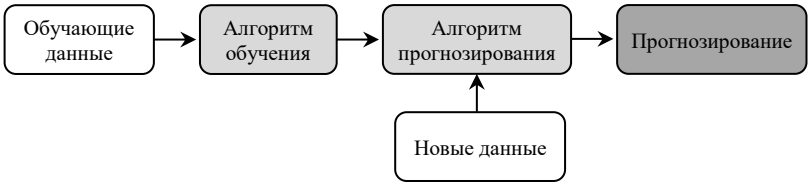


Рис. 9. Модель обучения и прогнозирования

Данные для обучения организованы в формате таблицы, где строки соответствуют объектам, а столбцы — атрибутам, дополнительно выделен столбец для категорий этих объектов.

На основе собранных данных модель машинного обучения автоматически разрабатывает прогностический алгоритм, способный по атрибутам (колонкам) нового элемента классифицировать его принадлежность к определенной категории.

Необходимо подчеркнуть, что датасеты должны быть комплексными, что подразумевает наличие данных по всем (или большинству) характеристикам для каждой единицы наблюдения.

При минимальном количестве отсутствующих данных в атрибутах применимо восполнение средними арифметическими показателями. Однако когда пропущенная информация обширна, выявить корреляции в наборе данных становится более трудоемким процессом.

Таким образом, для адекватной модели машинного обучения, необходимо иметь доступ к данным по всем (или большинству) атрибутам для каждого анализируемого экземпляра. Рас-

сматривая, к примеру, задачу прогнозирования характеристик будущих финансовых операций клиента, параметр «величина операции» может представлять значительный интерес. Однако до момента выполнения операции как ее сущность, так и объем финансовых средств остаются неопределенными, делая данный атрибут неприменимым.

Функции машинного обучения, такие как классификация, выступают связующим звеном между деловыми операциями и методами машинного обучения.

Для применения методов решения классификационных задач необходимо идентифицировать объект исследования, определить набор характеристик для его описания, выбрать целевой класс, который будет предсказан, и компилировать набор данных с аннотацией и без пропусков.

Задача регрессии

В регрессионном анализе, также как и в классификации, целью является предсказание значений, однако регрессия фокусируется на прогнозировании непрерывных, количественных переменных.

В задачах классификации датасет организован в форме таблицы, где каждая строка соответствует отдельному экземпляру или наблюдению, а столбцы представляют характеристики или атрибуты, по которым описываются эти экземпляры. Эти характеристики инвариантны для всех экземпляров, однако их конкретные значения уникальны для каждого объекта. Ключевым отличием в таких задачах является добавление специального столбца для меток или классов, к которым принадлежат объекты. Этот столбец также именуют как столбец целевой переменной, который содержит данные, на основе которых модель обучается выявлять закономерности для прогнозирования принадлежности новых объектов к определенным классам.

В регрессионных задачах данные аналогичны, однако целевая переменная представлена в числовом формате. В сущности, регрессионный анализ направлен на вычисление континуального исхода, основываясь на множестве атрибутов, т. е. определение количественного результата для каждого элемента в выборке. В контексте прогнозирования спроса цель заключается в анализе и прогнозировании объема продукции, который будет востребо-

ван в ритейлерском магазине за определенный период, скажем, за конкретную неделю.

Еще один пример — прогнозирование цены на апартаменты, где цена выражается в рублях и представлена как количественный показатель.

Другой пример включает оценку возраста индивида на основе фотографии (возраст представлен в виде числа).

На первый взгляд, различия между классификацией и регрессией кажутся минимальными и, по существу, они таковыми и являются. Тем не менее, как станет видно далее, подходы и методики в прогнозировании и обучении для этих двух типов задач существенно отличаются.

Аналогично задачам классификации, для решения требуется набор обучающих данных, содержащий признаки объектов и соответствующие известные значения количественной целевой переменной. Рассмотрим задачу прогнозирования цены жилья, используя данные, представленные в табл. 4.

Таблица 4

Информация для прогнозирования цены квартиры

Признаки				Целевая переменная
Площадь, м^2	Этаж	Число комнат	Число лет с последнего ремонта	Стоимость, млн ден. ед.
115	3	4	2	46.0
55	5	2	5	10.3
72	6	3	12	17.7
55	20	1	0	32.0

В данной задаче анализируется объект — квартира, с характеристиками: общей площадью, количеством комнат, расположением этажа и временем, прошедшим с момента последнего ремонта. Целью является предсказание переменной, которая представляет собой стоимость данной квартиры.

Используя данные для обучения, модель машинного обучения создает прогностический алгоритм. Этот алгоритм анализирует характеристики новой квартиры, включая площадь, количество комнат, этажность и наличие ремонта, для оценки ее рыночной цены.

Исходя из предпосылки о концептуализации задачи как регрессионной модели и компиляции набора обучающих данных, фокусируемся на методологии обучения на основе этих данных для разработки предиктивной модели.

Наиболее элементарный, а фактически бесперспективный метод прогнозирования, который возможно разработать, заключается в постоянном предсказании идентичного результата для любого элемента анализа.

В данном контексте нашей задачи это подразумевает, что цена предсказывается идентичной для всех апартаментов.

Чтобы оценить эту стоимость, можно воспользоваться тренировочным набором данных, а именно — рассчитать среднюю цену квартир в этом наборе. Процесс вычисления средней цены является сутью обучения; это значительно превосходит любые попытки предсказывать стоимость как нулевую.

Несмотря на то, что описанный алгоритм вряд ли окажется эффективным в реальных условиях из-за отсутствия анализа уникальных характеристик объектов, он все же может служить отправной точкой или примитивным эталоном. Любой усовершенствованный алгоритм должен демонстрировать более высокую точность по сравнению с этим базовым вариантом. Оценка качества предсказаний алгоритма может производиться путем вычисления средней абсолютной ошибки.

Линейные модели

Линейные модели являются наиболее распространенным подходом к регрессии. Линейная модель предсказания работает по принципу комбинации произведений весов и признаков: каждый признак умножается на свой вес, результаты суммируются. Представим, что наша цель — спрогнозировать цену жилья, основываясь на данных параметрах, представленных в табл. 5.

Таблица 5

Факторы, влияющие на стоимость квартиры

Площадь, м^2	Этаж	Число комнат	Число лет с последнего ремонта
70	2	3	5

Таким образом, предполагается, что вес коэффициентов признаков распределен следующим образом: 20 % приходится на

площадь, 150 % — на этаж, 63 % — на количество комнат, и наличие ремонта влияет отрицательно, уменьшая стоимость на 20 %.

Получается, что вес коэффициента указывает на значимость фактора для прогнозирования итоговой цены: коэффициент 0,20 свидетельствует о том, что увеличение площади квартиры на 1 $м^2$ поднимает ее цену на 0,20 ед., а показатель 0,63 говорит о том, что добавление одной комнаты увеличивает стоимость на 0,63 ед. Отрицательный вес, например, −0,2, демонстрирует, что каждый год после последнего ремонта снижает цену объекта на 0,2 ед.

Таким образом, проведя прогноз цены, который равен сумме произведений характеристик на их соответствующие коэффициенты, получаем: $0{,}20 \times 70 + 1{,}5 \times 2 + 0{,}63 \times 3 - 0{,}2 \times 5 = 17{,}89$ усл. ед.

Значительное абсолютное значение коэффициента указывает на высокую релевантность признака для модели прогнозирования, тогда как коэффициент, стремящийся к нулю, свидетельствует о минимальном вкладе данного признака в процесс формирования прогноза.

Поиск значений весов

В процессе обучения модель автоматически определяет оптимальные веса на основе данных, чтобы минимизировать ошибку прогноза, сравнивая результаты с различными наборами весов.

Таким образом, величина ошибки определяется на основе тренировочного набора данных: увеличение количества тренировочных экземпляров (квартир в наборе данных) способствует более высокой эффективности работы алгоритма машинного обучения в разнообразных ситуациях.

При наличии большого количества учебных данных адаптированные в ходе обучения параметры модели окажутся более эффективными, нежели настройки, предложенные экспертом по недвижимости, ввиду того, что машинный алгоритм анализировал значительно более обширный массив информации, чем мог охватить специалист.

Ранговый метод

Ранг обозначает позицию элемента во множестве с аналогичными характеристиками, классифицированного согласно определенному атрибуту. В контексте ранговой методологии экспер-

там предложено ранжировать цели внутри конкретных групп в порядке уменьшения значимости каждой цели по сравнению с другими.

Следовательно, в случае, когда в определенном наборе присутствует n объектов, каждому из этих объектов будет присвоен уникальный ранг r_i, где $i = 1, \ldots, n$.

Для эффективного распределения приоритетов целесообразно применять методику последовательного отбора наивысшего приоритета среди заданных задач, назначая ему наименьший ранг, а затем удалять выбранную цель и присвоенный ей ранг из дальнейшего анализа.

Предположим, что сформулированы следующие целевые установки: ц1, ц2, ц3, ц4.

Создадим таблицу для ранжирования (табл. 6).

Таблица 6

Шаблон для ранжирования целей

Номер цели	1	2	3	4
Ранг				

Пусть ц2 — наиболее важная цель, тогда $r_2 = 1$, начинаем заполнять нашу таблицу (табл. 7).

Таблица 7

Шаблон для ранжирования целей с указанием наиболее приоритетной цели

Номер цели	1	2	3	4
Ранг		1		

Аналогично рассмотрим оставшиеся цели. Заполненная таблица будет выглядеть так (табл. 8).

Таблица 8

Итоговая таблица ранжированных целей

Номер цели	1	2	3	4
Ранг	3	1	4	2

В процессе оценки значимости, веса целей устанавливаются исходя из их важности. Анализируется индивидуальная значимость каждой цели.

$$V_{i,k} = \frac{2}{n}\left(1 - \frac{r_{i,k}}{n+1}\right), \quad (1)$$

где n — количество целей, $r_{i,k}$ — ранг, присвоенный i-й цели k-м экспертом.

Коллективное определение важности целей выражается следующей формулой:

$$V_i = \frac{2}{n}\left(1 - \frac{\sum_{r=1}^{m} r_{i,k}}{m(n+1)}\right), \quad (2)$$

где m — количество экспертов.

Метод ранжирования отличается низкой сложностью выполнения, достаточно высоким уровнем консенсуса между оценивающими и невысокой детализацией решений.

Метод парных сравнений

При использовании этого метода каждому эксперту предлагается оценить относительную важность достижения каждой из двух рассматриваемых целей заданной совокупности. Таким образом, эксперт выполняет $\frac{n}{2(n-1)}$, заполняя матрицу предпочтений следующего вида (табл. 9).

Таблица 9

Матрица предпочтений метода парных сравнений

	1	2	3	…	*j*	…	*n*
1	1	$\frac{[1]}{[2]}$	$\frac{[1]}{[3]}$		$\frac{[1]}{[j]}$		$\frac{[1]}{[n]}$

Окончание табл. 9

	1	2	3	…	*j*	…	*n*
2		1	$\frac{[2]}{[3]}$		$\frac{[2]}{[j]}$		$\frac{[2]}{[n]}$
3			1				
…				1			
i					1		$\frac{[i]}{[n]}$]
…						1	
n							1

Обычно в процессе выполнения анализа устанавливают формат, в котором выражается значимость между целями *i* и *j* как отношение целых чисел в диапазоне от 1 до 10. Для каждого участника создается матрица предпочтений, которая трансформируется в ранжированную матрицу ($r_{i,j}$) на основе определенных соотношений. Если $i < j$, то расчеты проводят по формуле (3), если $i > j$, то по формуле (4).

$$r_{i,j} = \frac{2P_{i,j}}{P_{i,j}+1}, \tag{3}$$

где $P_{i,j}$ — значение предпочтения эксперта, отданное *i*-му объекту по отношению к *j*-му объекту.

$$r_{i,j} = \frac{2}{P_{i,j}+1}. \tag{4}$$

Оценки личной важности целей от *i*-го специалиста:

$$V_{i,k} = \frac{\sum_{j=1}^{n} r_{i,j,k}}{\sum_{i=1}^{n}\sum_{j=1}^{n} r_{i,j,k}}. \tag{5}$$

Групповые оценки значимости V_i:

$$V_i = \frac{\sum_{k=1}^{m}\sum_{j=1}^{n} r_{i,j,k}}{\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} r_{i,j,k}}. \quad (6)$$

Метод парных сравнений влечет за собой значительно больше усилий и предполагает меньше единодушия среди участников, но предоставляет высокую точность в оценках. В случаях, когда нет предварительных данных о глубине понимания проблематики и уровне компетенции участников, рекомендуется использовать метод ранжирования.

Единство мнений в группе экспертов является ключевым фактором для достижения надежных результатов. Важность этого аспекта подчеркивается использованием показателя конкордации W (7), который служит мерой согласованности между оценками участников. Этот метод позволяет оценить степень универсального согласия внутри экспертной группы, что является залогом обоснованности и весомости выносимых рекомендаций. Он не только отражает сходство и расхождение мнений в числовом выражении, но и выступает как ключевой фактор в оценке достоверности и устойчивости коллективных выводов.

Это выражение представляет собой математическую запись, возможно, указывающую на суммирование, разность, деление, возведение в степень числовых значений и переменных. Такие выражения часто встречаются в различных областях науки и инженерии, отражая комплексные вычисления, связанные, например, с алгебраическими операциями, физическими расчетами или программированием.

$$W = \frac{12\sum_{i=1}^{n}\left(\sum_{r=1}^{m} r_{i,j,k} - \frac{m(n+1)}{2}\right)^2}{m^2 n(n^2-1)}. \quad (7)$$

W изменяется от 0 до 1.

При значении W, превышающем 0,67, целесообразно организовать дополнительное исследование, применяя технику парного сравнения. В случае меньшего значения следует выяснить факторы, влияющие на снижение консенсуса. Для этого можно привлечь анализ мнений, выраженных одиночными специалистами или коллективами экспертов.

Спирменовский ранговый коэффициент корреляции

Для вычисления Спирменовского рангового коэффициента корреляции (ρ) применяется формула

$$\rho = 1 - \frac{6\sum_{i=1}^{n}\left(R_{i,1} - R_{i,2}\right)^2}{n(n^2 - 1)}, \tag{8}$$

где $R_{i,1}$ и $R_{i,2}$ — ранги, присвоенные i-й цели соответственно первым и вторым экспертами (или группами экспертов).

Интервал вариаций ρ находится между −1 и +1.

Учитывая представленную информацию, структура обучения будет представлена в соответствии с рис. 10.

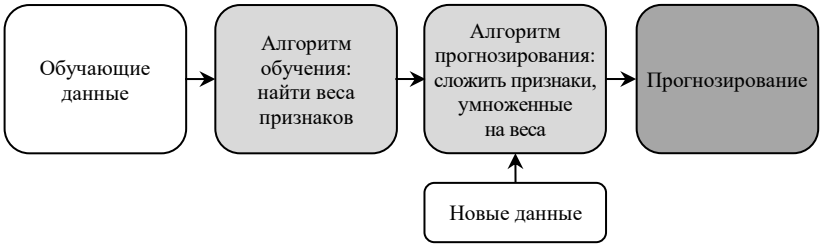


Рис. 10. Схема обучения

Выводы по главе 2

В учебном пособии основное внимание уделяется подходу к понятию искусственного интеллекта, разработанному С. Расселом и П. Норвигом. Ключевую роль в этой концепции играет технология агентов.

Мы установили, что концепция компьютерных агентов обогащена целым спектром особенностей, дифференцирующих ее от обыденного программного обеспечения, включая их способность к самостоятельным вычислениям, адаптации к окружающей их вычислительной среде, длительной работе без внешнего вмешательства, а также к анализу изменений в этой среде и разработке стратегий для оптимизации принятия решений.

Рационального агента мы определили как сущность, выполненные вычисления которой направлены на достижение оптимального исхода, что в условиях неопределенности гарантирует получение максимально благоприятного предполагаемого результата. В качестве агента в системе обработки данных рассматривался любой объект, который эффективно осуществляет восприятие и выполняет действия в определенной (параметризованной) среде.

Функциональное назначение агента в информационных системах определяется его способностью выполнять вычислительные операции в ответ на конкретную последовательность сенсорных воздействий, не всегда строго заданную.

Агент постоянно анализирует и оценивает окружающую его ситуацию через сенсоры и создает ответную реакцию, влияющую на эту среду при помощи исполнительных устройств.

Очевидно, что классификация программы как примера искусственного интеллекта зависит от истолкования термина естественного интеллекта. Учитывая отсутствие универсального определения и адекватных моделей естественного интеллекта для электронно-вычислительных машин, целесообразно применять более узкое понятие «технологии искусственного интеллекта». Сегодня любые системы, именуемые ИИ, фактически представляют собой разновидности таких технологий.

Было установлено, что разработка искусственного интеллекта, представляющего собой специализированное программное обеспечение для компьютеров, заключается в создании оптимизированной версии программного агента для конкретного использования. В качестве примера агента в человеческом понимании выступает человек со своими сенсорными органами, вроде глаз, обеспечивающими восприятие окружающего мира, и эффекторными элементами, такими как руки и ноги, выполняющими активные действия в окружающей среде.

Автономный робот-агент может быть оснащен видеонаблюдением и ИК-дальномерами для восприятия окружения, а для взаимодействия с ним можно использовать разнообразные моторы.

Предполагается, что оценка рациональности действий агента в определенный момент времени опирается на четыре ключевых фактора, включая:

– показатели производительности, которые определяют критерии успеха;

– знания агента о среде, приобретенные ранее;

– действия, которые могут быть выполнены агентом;

– последовательность актов восприятия агента, которые произошли до настоящего времени.

Анализируются четыре ключевых типа агентских программ, олицетворяющие фундаментальные концепции, на которых базируется большинство систем искусственного интеллекта:

– обычные или простые рефлексные агенты;

– рефлексные агенты, основанные на модели;

– агенты, основанные на цели;

– агенты, основанные на полезности.

Фундаментальным принципом машинного обучения является использование данных для настройки и оптимизации алгоритмов. Без достаточного объема информации эффективное обучение моделей невозможно. В качестве альтернативного подхода выделяются экспертные системы, базирующиеся на знаниях и опыте профессионалов, но их эффективность зачастую ниже, чем у моделей, обученных на данных. Это особенно заметно в задачах автоматизации и прогнозирования в устоявшихся сферах бизнеса, таких как работа колл-центров, кредитный скоринг или анализ потребительского спроса, где за долгие годы деятельности компаний удается собрать обширные массивы данных, предоставляющие ценный материал для тренировки алгоритмов машинного обучения.

В процессе анализа задач ИИ используется подход CRISP-DM, который охватывает следующие этапы.

1. Понимание бизнес-целей.
2. Начальное изучение данных.
3. Подготовка данных.
4. Моделирование.

5. Оценка.
6. Внедрение.

В области машинного обучения обычно используются данные, представленные в табличном формате, известные также как структурированные данные.

Анализируются таблицы в Excel и SQL Database 2019, который интегрирует функции машинного обучения. Связь машинного обучения с SQL Server возможна через использование языка R; для коммерческого применения доступна версия RevoScaleR.

Исследована платформа для обучения моделей на данных, хранящихся в SQL Server, с применением интеграции с Python. Это привело к появлению SQL Server Machine Learning Services, объединяющих в себе возможности R и Python как компоненты машинного обучения в рамках SQL Server.

Основные цели машинного обучения включают классификацию и регрессию. Чтобы оценить важность критериев, анализируются ранговая методика (наиболее базовая) и техника парных сопоставлений. Сформулированы формулы для расчета весов (важности) критериев и апробации значимости используемых подходов.

Вопросы для самоконтроля

1. В чем особенность концепции Стюарта Рассела и Питера Норвига?
2. Что такое рациональный агент?
3. Что определяет функция агента?
4. Роль агента в искусственном интеллекте.
5. Агентные вычисления и нейронная сеть в искусственном интеллекте.
6. Зачем принципы биологической эволюции применяются для решения сложных проблем?
7. Как в социальных системах представляется модель интеллекта?
8. Сколько существует концепций и определений агента, моделей и агентных вычислений?
9. В чем особенность общей схемы агента?
10. Как рассматривается программное обеспечение, выступающее в роли агента?
11. Можно ли рассматривать понятие агента как инструмент для анализа систем, почему?

12. Какой агент является рациональным агентом?
13. Что является правильным действием агента?
14. От каких факторов зависит оценка рациональности действий агента?
15. Как можно определить рационального агента?
16. Особенности программного робота, предназначенного для управления тренажером.
17. Особенности программного робота, предназначенного для просмотра источников новостей в Интернете.
18. Основные виды программ агентов.
19. Как следует понимать простые рефлексные агенты?
20. Как следует понимать рефлексные агенты, основанные на модели?
21. Особенность агентов, действующих на основе цели.
22. Как следует представлять агентов, действующих на основе полезности?
23. Как можно определить обучающихся агентов?
24. Могут ли агенты улучшать свою работу благодаря обучению.
25. Данные, ресурсы, метрики и специалисты в концепции машинного обучения.
26. Общая схема методологии по исследованию данных CRISP-DM.
27. Сколько этапов включает методология CRISP?
28. Какие структурированные данные машинного обучения вам известны?
29. Постановка задачи классификации. Является ли эта задача существенной в технологии ИИ?
30. Постановка задачи регрессии. Насколько она интересна в экономике?
31. Веса (значимость) показателей в задачи регрессии. Как их определяют?
32. Отличие рангового метода от метода парных сравнений.
33. В чем смысл коэффициента конкордации?

Глава 3 Модели, поиск в машинном обучении, технология ML.NET

3.1. Алгоритмы и методы обучения в искусственном интеллекте

В рамках линейных моделей предполагается, что функция отклика (прогноз) вычисляется как линейная комбинация характеристик объекта, где каждая характеристика умножается на соответствующий коэффициент (вес).

В рамках используемого метода обучения веса устанавливаются на основе величин характеристик и определяют степень влияния атрибутов.

Привлекательность линейных моделей заключается в легкости интерпретации их прогнозов, однако ограниченность линейного подхода не гарантирует наивысшую точность в предсказаниях будущего.

Задачи прогнозирования величин и методы их решения

Термин «алгоритм» описывает простую последовательность операций, которую для внедрения в образовательные технологии требуется трансформировать в программный код или, другими словами, закодировать.

В реальных ситуациях не всегда нужно программировать с нуля, важно использовать существующие технологии, содержащие оптимизированные и высокопроизводительные функции.

В языке программирования Python для выполнения задач классификации и регрессии широко используется библиотека Scikit-learn, предоставляющая обширный арсенал необходимых алгоритмов и функций. Для работы с линейными моделями также

применим пакет функций Vowpal Wabbit, который поддерживает эффективную реализацию подобных задач.

Библиотека Sklearn в контексте машинного обучения на Python предлагает широкий набор алгоритмов для комплексного анализа данных, включая классификацию, кластеризацию и регрессионный анализ.

В Vowpal Wabbit включены функции, важные для сферы машинно-обученческих систем, особенно при реализации режима online learning, методов сокращения измерений и обучения с подкреплением.

Vowpal Wabbit также способен обрабатывать задачи, где объем данных превышает объем доступной оперативной памяти компьютера.

Функции Vowpal Wabbit обеспечивают потребности как широкого круга пользователей, так и экспертов в сфере машинного обучения, включая выполнение регрессионного анализа, задач классификации и факторизацию матриц.

Переобучение

Тренировочные данные применяются для разработки алгоритма прогнозирования.

В процессе тренировки алгоритма машинного обучения ожидается, что он сможет делать достаточно точные прогнозы на данных, на которых он был обучен, ведь основная цель обучения заключается в повышении его способности к обобщению информации. Но может возникнуть ситуация, когда алгоритм успешно прогнозирует результаты только для тренировочного набора данных, не обнаружив универсальных закономерностей, связывающих входные признаки с целевым значением (категорией или числом). Это свидетельствует о том, что алгоритм просто запоминает конкретные примеры, вместо того, чтобы учиться на них. В результате при его реализации в реальных условиях такой алгоритм окажется неэффективным, что иллюстрирует явление переобучения.

Для предотвращения переобучения обычно используют разделение данных на обучающий и тестовый датасеты.

Первый датасет применяется для тренировки модели машинного обучения, а второй — для верификации точности модели на данных, не участвовавших в ее обучении.

В задачах регрессии необходимо определить:

1) признаки по весам;

2) класс по признакам;

3) число по признакам;

4) веса по классам.

В линейных представлениях целевой функции обучение включает:

– поиск средней целевой переменной;

– определение подходящего веса;

– определение порога измерений;

– вычисление произведения признаков на веса.

О метриках

Мы изучили основы классификации и регрессии, которые представляют собой два ключевых типа задач в области машинного обучения.

Чтобы разработать алгоритм регрессии или классификации, обычно используют линейные модели, ансамбли деревьев принятия решений или искусственные нейронные сети.

После завершения обучения алгоритма критически важно провести оценку его эффективности, т. е. оценить точность и надежность предоставляемых им прогнозов.

Давайте разберемся с оценкой качества в контексте регрессионного и классификационного анализа.

Следует подчеркнуть значимость адекватного определения критериев оценки эффективности.

Эксперт в области машинного обучения должен проводить оценку эффективности модели, используя адекватную метрику, иначе результативность разработанной модели может оказаться неверной и непригодной.

В повседневной деятельности используются как Интернет, так и традиционные показатели оценки.

Для измерения фактического воздействия внедрения определенной стратегии в продукт используются онлайн-метрики, такие, как увеличение прибыли или выполнение задачи, поставленной пользователем.

В зависимости от сферы применения используются различные показатели эффективности онлайн-маркетинга. В секторе интернет-маркетинга, например, активно применяется коэффици-

ент кликабельности (Click-Through Rate, CTR), который необходим для оценки соотношения количества кликов по рекламным объявлениям к общему числу просмотров, что позволяет измерить эффективность рекламной кампании и уровень вовлеченности аудитории. Этот показатель отражает результативность интернет-маркетинговых усилий, выраженную через соотношение числа кликов по рекламному объявлению к его общему количеству показов, измеряемое в процентном выражении. Если рекламное объявление появляется 20 раз и привлекает 5 кликов, то коэффициент кликабельности составит 25 %, что считается довольно высоким уровнем эффективности.

Онлайн-платформы анализируют среднюю длительность сессии посетителя на сайте.

В контексте корпоративной экономики рассматривается показатель пожизненной ценности клиента (LTV, Lifetime Value), представляющий собой общий доход, полученный от одного покупателя на протяжении всего периода взаимодействия. Эффективность этого показателя повышается за счет снижения объема отказов от услуг. Для максимизации LTV компании применяют стратегии лояльности, включающие предложения специальных условий, скидок и подарков, что существенно экономичнее, чем затраты на привлечение новых клиентов, зачастую превышающие исходные затраты в 8−10 раз.

Актуальные инструменты маркетинга, включая CRM (Customer Relationship Management — управление взаимоотношениями с клиентами) и advertising platforms (рекламная площадка), в сочетании с продвинутыми методами обработки данных, основанными на Больших данных и машинном обучении, способствуют повышению показателя LTV для компаний.

Цифровая проверка качества зачастую включает в себя проведение тестирования, т. е. реализацию модели для сегмента пользователей и анализ результативности.

Очевидно, что разработка и интеграция онлайн-метрик является сложной задачей, и в случае создания неэффективного алгоритма это может привести к значительному недовольству среди некоторых клиентов.

При разработке алгоритма рекомендуется применять офлайн-метрики, поскольку они позволяют оценить точность про-

гнозирования, а не финансовые результаты, обеспечивая быструю оценку без необходимости реализации.

Офлайн-метрики

Задачи регрессионного анализа включают в себя моделирование и прогнозирование количественных данных, таких как цена на жилье, возраст покупателя или продолжительность периода без начисления процентов по кредиту.

Пример. Осуществим анализ прогнозирования потребительского спроса: целью является определение объема продукции, который будет необходим на конкретном торговом объекте на протяжении последующей недели.

В процессе оценки качества используются данные из таблицы, включающей в себя два основных столбца: фактически верные данные и соответствующие им прогнозируемые значения.

Как было установлено раньше, апробация эффективности алгоритма должна проводиться на данных, которые не использовались в процессе его тренировки, а именно — на тестовой выборке. Таким образом, в таблицу вносятся прогнозы по тестовым объектам и связанные с ними реальные значения из набора данных. Для удобства анализа возьмем четыре объекта для тестирования (табл. 10), хотя в реальности тестовые наборы данных могут насчитывать тысячи или даже миллионы записей.

Таблица 10

Тестовые объекты

Номер товара	Реальный спрос из выборки (объем потребления товара, тыс. ед.)	Прогнозируемая величина, тыс. ед.
1	22	20
2	17	21
3	16	14
4	17	27

Визуальный анализ эффективности прогнозов показывает высокую точность для первого и третьего продукта с минимальным отклонением в 2 тыс. ед. В то время как второй продукт демонстрирует увеличенную ошибку прогнозирования в 4 тыс. ед., ошибка для четвертого продукта значительно выше — 10 тыс. ед.

Однако, столкнувшись с практическими случаями, когда анализируются тысячи данных, выполнение визуального анализа становится невозможным — требуется агрегированный индикатор качества.

Подсчет метрик регрессии

MAE (средняя абсолютная ошибка) — необходимо посчитать модуль разницы между прогнозом и реальным значением для всех объектов, а затем поделить разницу на число объектов.

MSE (среднеквадратическая ошибка) — необходимо посчитать разницу между прогнозом и реальным значением для каждого объекта, а затем возвести каждую в квадрат, сложить результаты и разделить на число объектов.

RMSE (квадратный корень из среднеквадратической ошибки) — необходимо посчитать разницу между прогнозом и реальным значением для каждого объекта, возвести каждую в квадрат, сложить результаты, поделить на число объектов, а затем взять корень из получившегося среднего значения.

MAPE (средняя процентная ошибка) — необходимо посчитать разницу между прогнозом и реальным значением, а затем поделить ее на реальное значение, получив среднее в виде %.

Для вычисления совокупного индикатора качества разумно применить операцию усреднения ошибок по всем элементам, суммируя дисперсии между прогнозируемыми значениями и фактическими и последующим делением полученной суммы на общее число элементов. В рассматриваемом случае результатом будет $\frac{2+4+2+10}{4}=4{,}5$ тыс. ед. Иными словами, в среднем алгоритм отклоняется на 4,5 тыс. ед.: в некоторых случаях больше, в других — меньше.

Этот показатель известен как средняя абсолютная ошибка (mean absolute error, MAE) и рассчитывается по формуле

$$\mathrm{MAE}\left(Y^{\mathrm{true}}, Y^{\mathrm{pred}}\right)=\frac{1}{n}\sum_{i=1}^{n}\left|y_i^{\mathrm{true}}-y_i^{\mathrm{pred}}\right|, \qquad (9)$$

где Y^{true} — правильные ответы для выборки; Y^{pred} — прогнозируемые значения для аналогичного набора данных; y_i^{true} — представ-

ляет собой верный ответ или целевое значение для *i*-го элемента; y_i^{pred} — прогноз для *i*-го элемента; *n* — всего объектов в выборке.

Часто применяется аналогичная метрика, известная как среднеквадратичная ошибка (mean squared error, MSE), для расчета которой производят усреднение квадратов разностей между прогнозными и реальными значениями.

$$\text{MSE}(Y^{\text{true}}, Y^{\text{pred}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i^{\text{true}} - y_i^{\text{pred}})^2. \quad (10)$$

В данном случае, исходя из примера, среднеквадратическая ошибка равна $\frac{4+16+4+100}{4} = 31.$

Однако вместо запрашиваемого числа товаров мы получили значения, представленные в квадратах тысяч. Для того чтобы привести их к первоначальным единицам измерения, требуется применение операции извлечения квадратного корня, что приводит к понятию среднеквадратической ошибки. В данной ситуации RMSE равен квадратному корню среднеквадратической ошибки, которая составляет 31, в результате чего получаем $\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{31} = 5{,}57$ ед.

Относительно RMSE, MAE воспринимается проще, поскольку происходит усреднение абсолютных разностей, однако RMSE предпочтительнее для тренировки моделей. Тем не менее, обучение с использованием MAE также эффективно.

Одним из преимуществ метрики средней абсолютной ошибки является ее повышенная устойчивость к аномалиям данных по сравнению с корнем среднеквадратической ошибки. Это объясняется тем, что в случае наличия одного аномального значения с значительным отклонением от остальной выборки влияние этого отклонения на итоговое значение MAE будет меньше. Причина в том, что RMSE усиливает эффект от выбросов за счет возведения разностей в квадрат, тогда как MAE оперирует абсолютными значениями разностей.

В приведенном примере аномальным объектом выступает четвертый продукт, при этом показатель средней абсолютной

ошибки на 1 тыс. ед. ниже, чем значение корня среднеквадратической ошибки.

Для достижения целей бизнес-проекта важно обеспечить примерно одинаковые показатели искажений во всех аспектах (избегая выбросов), что делает использование RMSE предпочтительным методом. В случае, если приемлемо допустить ошибку в нескольких случаях ради снижения искажений в большей части данных, следует оптимизировать процесс, используя MAE.

Ошибки, ведущие к отклонениям результатов в пониженную или повышенную стороны, могут оказывать различное воздействие на бизнес-операции. Возможность учета этой особенности может быть интегрирована в аналитическую метрику.

К примеру, прогнозируя на 1 тыс. ед. продукции меньше фактического спроса, мы навлекаем убытки: дефицит товара приведет к тому, что часть покупателей останется без покупки.

В случае прогнозирования объемов продукции, на 1 тыс. ед. превышающих фактическую потребность, возникнут дополнительные расходы, связанные с ее складированием.

Если продукция компактна и не требует значительных затрат на складирование, предпочтительнее приобрести ее с избытком, нежели столкнуться с нехваткой. В таких ситуациях для расчета экономической эффективности применяют дифференцированный подход к оценке излишков и дефицита товаров, используя коэффициенты 0,5 для излишков и 1,5 для недостатков.

В данном случае множитель 1,5 будет использоваться для товаров 1 и 3, поскольку их значения (18 и 12 соответственно) меньше предельных значений 20 и 14. Для оставшихся двух товаров будет применяться коэффициент 0,5. Таким образом, расчетное значение метрики будет определено на основе этих коэффициентов:

$$1,5\times 2+0,5\times 4+1,5\times 2+0,5\times 104=60.$$

Из-за повышенного значения коэффициента алгоритм скорее минимизирует ошибки для элементов 1 и 3, нежели для 2 и 4. Такой показатель обычно именуют квантильным смещением.

Интерпретация метрик

К оценке качества регрессионных моделей можно подходить, анализируя отношение значений метрик, таких как MSE или MAE, к среднему значению зависимой переменной. Это особенно актуально, когда предсказываемые значения находятся в пределах определенного порядка величин, а допустимый разброс ошибки заключен в меньшем масштабе. Для преобразования значений ошибок в процентное выражение и получения более интуитивно понятной интерпретации результатов моделирования применяют метрики, включающие нормирование, что позволяет обозначить среднюю степень отклонения предсказаний от фактических данных в процентном соотношении.

Например, индикатор средней абсолютной процентной ошибки (MAPE) вычисляет среднее из отношений абсолютной ошибки к значению целевой переменной.

$$\mathrm{MAPE}(Y^{\mathrm{true}}, Y^{\mathrm{pred}}) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i^{\mathrm{true}} - y_i^{\mathrm{pred}}}{y_i^{\mathrm{true}}}\right|. \qquad (11)$$

В нашем примере получится следующее:

$$\frac{\frac{2}{20}+\frac{4}{15}+\frac{2}{14}+\frac{10}{15}}{4} \approx 29\,\%.$$

Таким образом, точность выполнения алгоритма составляет приблизительно 71 %.

Крайне важно осознавать, что в области машинного обучения отсутствуют абсолютно совершенные алгоритмы или метрики ошибок, стремящиеся к нулю: этот раздел вычислительной техники принципиально ориентирован на генерацию аппроксимаций будущих событий.

Установление допустимого порога ошибки происходит по усмотрению заказчика и выражается через конкретные показатели выбранной метрики. Так, в контексте такого параметра, как MAPE, заказчик может установить, что значения ошибки не должны превышать 20 % на тестовом наборе данных. При пре-

вышении этого порога по результатам обучения модели предстоит либо дополнительная оптимизация алгоритма, либо расширение объема обучающих данных для улучшения точности.

Оценка ошибки сверху

Приемлемые значения метрик варьируются в зависимости от целей бизнеса.

К примеру, совершение сделок по продаже недвижимости с отклонением в стоимости на 100 тыс. р. в большую или меньшую сторону является приемлемым и часто фиксировалось в операциях компании, в то время как разброс в 300 тыс. р. выходит за рамки допустимого.

Дополнительно в качестве максимально допустимой ошибки можно выбрать показатели ошибки базового решения (бейзлайн — простой алгоритм или эвристика, служащие критерием для оценки эффективности разрабатываемой модели. Этот первоначальный критерий помогает аналитикам устанавливать ожидаемый минимум эффективности для решения задачи). Примитивный бейзлайн — это константная модель, прогнозирующая одно и то же значение для любого входного примера. В случае, если ошибка модели превысила показатель константного бейзлайна, значит, в процессе тренировки модели была допущена оплошность.

Следует подчеркнуть, что вне зависимости от бизнес-задач следует выбор методик оценки без определенных ограничений. Варианты включают широко применяемые среднюю абсолютную ошибку или среднюю абсолютную процентную ошибку, возможность интегрирования нескольких показателей (для примера, применение коэффициентов для оценки среднеквадратичной вместо абсолютной ошибки) или разработку уникальных критериев оценки.

В случаях, когда важность представляет не сам размер ошибки, а ее соответствие определенному пределу, уместно применение метрики «доля ситуаций с ошибкой ниже установленного критерия». Это особенно актуально для задач, подобных прогнозированию спроса, когда товары размещаются в упаковках (при этом даже одиночный продукт требует целой упаковки для хранения) и есть ограничение в виде возможности хранения только одной упаковки для каждого наименования товара.

Агенты для решения задачи

Проанализируем категории целенаправленных агентов, их классификацию будем проводить как «агенты, выполняющие задачи». Агенты, задача которых найти решение, определяют последовательность действий, приводящих к цели.

Предположим, что ИИ-агенты способны оптимизировать свою эффективность. Эта задача облегчается, когда агент может взять на себя ответственность за достижение цели.

Вообразите ситуацию, когда профессионал решает, где ему работать. Критерии, по которым он оценивает потенциальное место работы, обширны: ему важно, чтобы базовый оклад был высоким с возможностью его увеличения через стимулирующие выплаты; стремится получить доступ к корпоративному фитнесу; желает, чтобы работа приносила удовольствие; хочет минимизировать негативные аспекты трудовой деятельности и пр.

Данное решение требует учета обширного ряда компромиссных моментов и глубокого анализа нормативных актов организации. Агенту рекомендуется нацелиться на оптимизацию своего выбора рабочего места, ориентируясь на максимально выгодный совокупный критерий.

Цели структурируют действия, ограничивая альтернативы промежуточных действий, к которым стремится агент. Определение цели с учетом контекста и метрик эффективности агента является первым шагом к решению задачи.

Обсудим цель как набор возможных состояний реальности, где эта цель является достигнутой. Миссия агента заключается в выявлении последовательности шагов, которые направят его к достижению конечной цели. Перед этим агенту необходимо идентифицировать необходимые для анализа действия и состояния. Определение задачи включает в себя процесс уточнения, какие именно действия и состояния необходимо взять во внимание для достижения определенной цели. Следовательно, агент анализирует состояния, представляющие собой выборку определенных организационных структур.

Таким образом, наш представитель поставил цель изучить предложения ряда организаций. В общих терминах, агент, сталкивающийся с несколькими альтернативами, чьи результаты ему неизвестны, может определить наилучший курс действий, сна-

чала изучая различные потенциальные цепочки действий, приводящие к исходам, о которых он имеет информацию, а после — выбирая оптимальную из них.

Алгоритм, который был изучен для идентификации определенной последовательности, будем определять как процесс поиска. Каждый поисковый алгоритм принимает на вход специфическую проблему и выдает решение, представленное в виде шагов. После обнаружения решения выполняются предписанные алгоритмом шаги. Этот процесс происходит в фазе исполнения.

Для данного агента можно реализовать базовую схему работы, основанную на классическом цикле действий: формулирование задачи — поиск решения — выполнение действий. Сначала агент определяет цель и задачу, для выполнения которой инициирует процесс поиска оптимального пути решения. Полученное решение становится основой для его последующих действий, позволяя выбрать и выполнить начальный этап предложенного курса действий (обычно это первый шаг в плане) и последовательно исключать его из дальнейшей последовательности действий.

Сразу по завершении действия агент определяет следующий объект задачи (рис. 11).

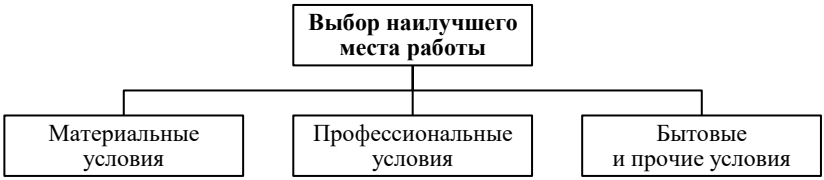


Рис. 11. Формулировка новой цели

Задача может быть описана через четыре основных элемента.

Начальное положение агента, откуда он начинает задание. Так, в контексте нашего агента исходное состояние при выборе места работы определяется как набор потенциальных компаний.

Определение доступных операций, которые может выполнить агент, осуществляется через функцию функция-преемник. Эта функция, находясь в контексте задачи искусственного интел-

лекта и автоматизированного принятия решений, предоставляет способ оценки возможных переходов между состояниями в пространстве состояний. Конкретизируя для любого заданного состояния *s*, функция-преемник (*s*) выделяет набор допустимых переходов в виде упорядоченных пар <действие, преемник>, где каждое действие является легальным в рамках данного состояния *s*, а соответствующий преемник описывает возможное последующее состояние, достижимое путем применения данного действия. Этот процесс аналогичен выбору наилучшего пути из множества возможных вариантов развития событий на основе доступных альтернатив.

Комбинация начального положения и функции определения последующего элемента неявно формирует пространство всех возможных состояний, доступных из начальной точки. Это пространство представляет собой граф, где каждое состояние — это узел, а переходы между состояниями отображаются как ребра, обозначающие действия. Маршрут через это пространство состоит из ряда состояний, связанных последовательностью примененных действий.

Валидация целевого состояния, представляющая собой процесс идентификации того, соответствует ли рассматриваемое состояние определенному критерию цели. В определенных случаях, когда цели заранее заданы как конечный набор возможных состояний, эта проверка упрощается до сопоставления текущего состояния с одним из предопределенных вариантов. В контексте взаимодействия агента в динамической среде цель обычно предстает в виде уникального состояния, к достижению которого стремится агент.

Порой цель определяется через абстрактное качество, а не через конкретный перечень состояний. Так, в шахматах задача заключается в достижении матового положения, когда король соперника находится под угрозой и лишен возможности избежать атаки.

Функция «Материальные условия» присваивает численное значение каждой организации. Агент, занимающийся решением проблемы, выбирает эту функцию в соответствии с его индивидуальными показателями эффективности.

Указанные ранее компоненты являются определяющими для формулировки задачи и могут быть интегрированы в комплексную структуру данных, которая используется как входные параметры в процессе выполнения соответствующего алгоритма.

Решение задачи представляет собой последовательность шагов, приводящих от начального к целевому состоянию.

Игра в восемь, демонстрация которой представлена на рис. 12, включает в себя поле размером 3×3 с восемью пронумерованными фишками и одной пустой клеткой. Фишка, находящаяся в непосредственной близости к этой свободной клетке, имеет возможность быть перемещенной на нее. Требуется достичь указанного целевого состояния, подобного тому, которое показано в правой части рисунка. Общепринятое описание данной головоломки изложено далее.

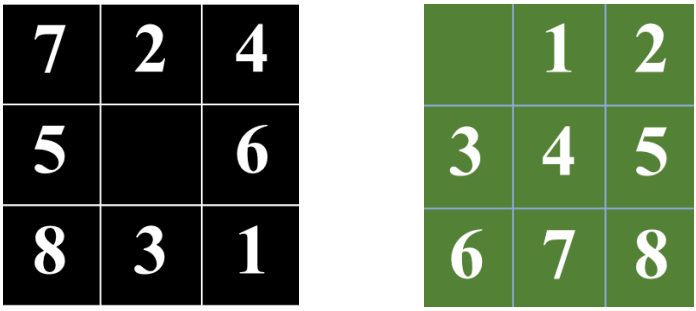


Рис. 12. Задача игры в восемь

Состояния. Для описания состояния используется информация о расположении всех восьми фишек и пустого участка на одной из девяти ячеек.

Исходная точка. Укажем, что в качестве стартовой позиции может быть выбрано любое условие. Важно подчеркнуть, что заданная цель может быть эффективно реализована из 50 % всех потенциальных исходных позиций.

Функция генерации преемников. Она формирует возможные состояния, полученные путем выполнения возможных действий: движение влево, вправо, вверх или вниз.

Проверка цели. Она позволяет определить соответствие текущего состояния установленным целям.

Стоимость пути. Каждый этап имеет стоимость 1, поэтому стоимость пути равна количеству этапов в пути.

Какие абстрактные концепции заложены?

Абстракция действий сводится к определению исходного и целевого состояний, минуя рассмотрение всех промежуточных стадий во время перемещения игрового элемента.

При разработке абстрактного описания были исключены операции, такие как встряхивание доски, позволяющее передвинуть застрявшую фишку, или использование инструмента, например, ножа, для извлечения фишек с последующим их размещением обратно в ящик. Исключено также описание правил игры, что позволяет обойтись без рассмотрения подробных сведений о физических манипуляциях. Задача игры в восемь имеет $9!/2 = 181\,440$ достижимых состояний и легко решается.

Игра в пятнадцать является частью группы головоломок на основе передвижения плиток, широко применяемых в области искусственного интеллекта для тестирования разработанных методов поиска решений.

Этот широкий спектр задач классифицируется как NP-полные, означая маловероятность обнаружения алгоритмов, способных гарантированно работать существенно эффективнее других методов поиска в худших сценариях.

Игра в пятнадцать, которая представляет собой головоломку, где нужно упорядочить 15 плиток в заданную конфигурацию на игровом поле 4×4, имеет около 1,3 трлн состояний, и случайно выбранные ее экземпляры могут быть решены оптимальным образом за несколько миллисекунд с помощью наилучших алгоритмов поиска [5, с. 118].

Задача игры в двадцать четыре (на доске 5×5) имеет количество состояний около 10^{25}, и случайно выбранные ее экземпляры все еще весьма нелегко решить оптимальным образом с применением современных компьютеров и алгоритмов [5, с. 118]).

Реальные задачи

Алгоритмы нахождения пути применяются в широком спектре областей — от маршрутизации данных в сетевых технологиях до оперативно-тактического планирования в военной

сфере и организации авиаперелетов. Разработка и оптимизация эффективных алгоритмов для таких сложных задач часто требует значительных усилий.

Разберем упрощенный вариант задачи составления плана авиаперелетов, который определен следующим образом.

Состояния определяются местонахождением (например, аэропортом) и текущим временем.

Исходное положение определяется в параметрах задания.

Процедура вычисления последующего состояния. Этот алгоритм вычисляет и предоставляет набор возможных ситуаций, проистекающих из реализации любых запланированных авиаперелетов, учитывая переменные, такие как категория и номер места, исключая рейсы, запланированные на отправление до текущего момента времени. Это учитывает временные рамки для перемещений между терминалами и трансферов между аэропортами.

Верификация достижения цели. Достигли ли мы пункта назначения в установленный срок?

Стоимость поездки определяется ценой билета, временем ожидания вылета, длительностью перелета, таможенными и пограничными формальностями, удобством кресел, временем отправления, моделью авиалайнера, бонусами для часто летающих клиентов и пр.

В сфере коммерческого туристического консалтинга применяются сложные алгоритмы планирования маршрутов, учитывающие многообразные и сложные модели тарификации, принятые в авиалиниях. Эти системы пытаются оптимизировать поиск вариантов перелетов, но каждый бывалый турист осознает, что реальность часто вносит свои коррективы в любые, даже самые продуманные планы.

Высококачественная система должна предусматривать планы действий в непредвиденных ситуациях (такие, как страховочное бронирование билетов на альтернативные рейсы) в объеме, соизмеримом с затратами и вероятностью отклонения от изначального пути.

Задачи формирования маршрута тесно переплетены с задачами навигации, однако существует ключевое отличие.

Задача коммивояжера (TSP) является ключевым элементом в области оптимизации маршрутов, требуя посещения каж-

дого города однократно для минимизации общего пути. Она относится к классу NP-трудных проблем, указывая на ее вычислительную сложность в контексте булевых функций и унарных операторов на машинах Тьюринга. Вопреки трудностям, разработке эффективных алгоритмов для TSP было уделено множество ресурсов. Применение таких алгоритмов находится далеко за пределами классических маршрутов коммивояжера, распространяясь на оптимизацию процессов в промышленности, включая автоматизацию сверления микросхем и логистику в цехах, повышая их эффективность.

Задача разработки СБИС (сверхбольших интегральных схем) включает в себя определение расположения миллионов транзисторов, резисторов и других элементов, а также их взаимосвязей на кристалле с целью уменьшения занимаемой площади, сокращения времени прохода сигналов, уменьшения эффектов паразитной емкости и увеличения процента выхода качественной продукции. Этот процесс размещения возникает после этапа создания логической схемы и обычно делится на этап размещения функциональных блоков (ячеек) и этап прокладки соединительных путей (маршрутизация).

В процессе разработки интегральных схем элементарные компоненты устройства организуются в ячейки, каждая из которых служит для исполнения определенной функциональной задачи. Эти ячейки имеют фиксированные габаритные характеристики, включая размеры и общую занимаемую площадь, а также требуют установления специфического количества электрических соединений с другими ячейками. Основная задача заключается в оптимизации расположения этих ячеек на кристалле микросхемы так, чтобы обеспечить их взаимное неперекрывание и выделить достаточно места для маршрутизации соединительных проводников, что обеспечит надежную и эффективную работу микросхемы.

В процессе разметки каналов на печатных платах выполняется подбор оптимальных путей для проводников через промежутки между компонентами. Эти операции по определению путей представляют собой высокосложные задачи, однако инвестиции в их решение безусловно окупаются.

Задача управления навигацией робота является расширенной версией проблемы поиска пути, описанной ранее. В данной задаче мы переходим от дискретного набора путей к сценарию, где робот движется в континуальном пространстве, предполагая бесконечное количество потенциальных движений и состояний.

Для осуществления регулярного движения робота на плоской поверхности измерения пространства эффективно сводятся к двум измерениям. Однако когда робот оснащен как верхними, так и нижними конечностями или колесами, которыми необходимо манипулировать, размерность пространства для исследования увеличивается, делаясь многомерной. Чтобы сделать такое пространственное поле для поиска ограниченным, применяются сложные аналитические методы.

Исходная трудность проекта обостряется, поскольку в процессе регулирования действующих роботов критично принимать в расчет неточности измерительных устройств и несовершенства в функционировании приводных механизмов.

Задачи автоматического упорядочения сборки сложных объектов. Цель состоит в определении последовательности, в которой должны быть собраны детали некоторого объекта. Если выбрана неправильная последовательность, то в дальнейшем нельзя будет найти способ добавления некоторой детали к этой последовательности без отмены определенной части уже выполненной работы. Проверка возможности выполнения некоторого этапа в последовательности представляет собой сложную геометрическую задачу поиска, тесно связанную с задачей навигации робота. Поэтому одним из дорогостоящих этапов решения задачи упорядочения сборки является формирование допустимых преемников. Любой практически применимый алгоритм должен предотвращать необходимость поиска во всем пространстве состояний, за исключением крошечной его части. Еще одной важной задачей сборки является ***проектирование молекулы белка***, цель которой состоит в определении последовательности аминокислот, способных сложиться в трехмерный белок с нужными свойствами, предназначенный для лечения некоторых заболеваний.

В настоящее время увеличился спрос на разработку программного обеспечения для автоматических агентов, способных

выполнять **поиск в Интернете**, анализировать данные, экстрагировать нужные сведения, отвечать на запросы пользователей и осуществлять операции на электронных торговых площадках. Это отличное приложение для алгоритмов поиска, поскольку Всемирную паутину удобно воспринимать как граф, включающий вершины (веб-страницы), связанные между собой через гиперссылки.

После того, как задача будет четко определена, следует искать способы ее решения.

Реализовать задуманное можно с помощью техники перебора возможных вариантов.

Сосредоточим внимание на алгоритмах поиска, использующих специально созданное дерево для определения пути. В таком дереве важны два элемента: начальное положение объекта поиска и функция, определяющая следующий шаг (функция преемства), которые вместе формируют все пространство возможных состояний, через которые может проходить поиск. Важно отметить, что если система позволяет существование различных маршрутов для достижения одного и того же конечного состояния, тогда древовидную структуру целесообразно заменить на графовую, обеспечив таким образом более комплексный и всесторонний подход к поиску решения задачи.

На рис. 13 развернутые узлы затенены; узлы, которые были сформированы, но еще не развернуты, выделены полужирным контуром; узлы, которые еще не были сформированы, обозначены тонкими штриховыми линиями.

Важно осознавать различия между пространством состояний и деревом поиска.

Количество состояний в множестве ограничено, в то время как множество траекторий является неограниченным.

Таким образом, бинарное дерево поиска потенциально содержит бесконечное количество узлов. Разнообразие форматов узлов предполагает, что каждый узел — это структура данных, включающая в себя набор элементов.

Необходимо осознавать разницу между узлом и множеством его состояний. Узел — это элемент данных, применяемый для создания структуры дерева поиска, тогда как состояние отражает состояние внешних условий.

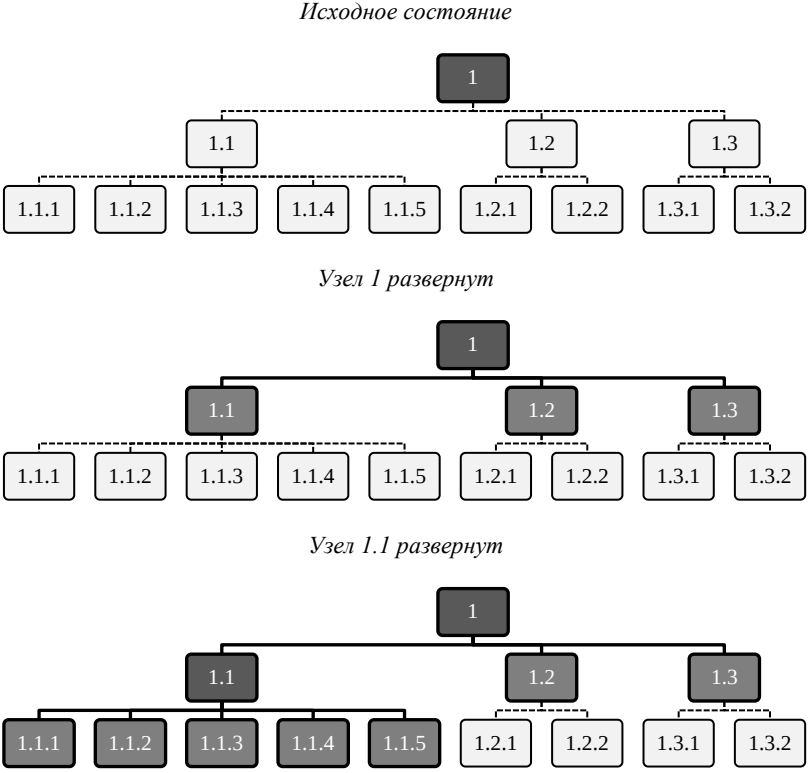


Рис. 13. Дерево поиска, используемое алгоритмом определения пути до конечной точки

Очевидно, что узлы находятся в точно определенных позициях, обозначаемых при помощи указателей, в то время как состояния не имеют фиксированных мест. Возможна ситуация, когда одна и та же конфигурация внешней среды может быть ассоциирована с двумя различными узлами, предполагая, что их можно идентифицировать через разные маршруты поиска.

Методы поиска без использования знаний о предметной области

Алгоритм поиска в ширину (BFS) представляет собой один из методов обхода графа, который начинает свое действие с исследования корневого узла, после чего переходит к исследованию

всех его непосредственных потомков (преемников), а затем — к их потомкам, и так продолжается вплоть до исследования всех доступных узлов. Этот метод требует, чтобы перед переходом к узлам следующего уровня глубины были полностью развернуты и исследованы все узлы текущего уровня, обеспечивая таким образом систематическое и последовательное покрытие всех уровней в дереве или в графе поиска.

Алгоритм поиска в глубину (DFS) предпочтительно исследует самый дальний узел на текущем границе графа. Этот принцип задействует стратегию максимального погружения в граф, отдавая приоритет узлам, находящимся на наибольшем расстоянии от корня, до достижения тех, которые не имеют исходящих вершин. Как только такие узлы обработаны и исключены из контура, алгоритм выполняет обратный ход, возвращаясь к более поверхностному уровню с целью исследования оставшихся вершин с непросмотренными потомками.

Основная проблема алгоритма поиска в глубину заключается в риске принятия ошибочного решения, когда выбирается направление, ведущее к тупиковому узлу. Это может привести к погружению в глубокую ветвь дерева поиска, которая оказывается чрезвычайно длинной или даже бесконечной, тогда как альтернативный маршрут, возможно, был бы кратчайшим путем к цели, расположенной вблизи исходной точки.

При применении метода поиска с ограниченной глубиной, где d обозначает глубину наименее удаленного от корня решения, а l обозначает максимальную глубину поиска, возможно предотвратить циклическое погружение в бесконечность. Однако, устанавливая ограничение глубины, вносится дополнительный фактор неполноты в алгоритм: риск пропустить действительное решение при выборе l меньше d, особенно когда точная глубина d неизвестна заранее. Это может привести к ситуации, когда верное решение не будет найдено из-за его расположения за установленными границами поиска.

Вдобавок применение поиска с ограничением по глубине окажется неэффективным, если заданное ограничение превышает оптимальную глубину решения d.

3.2. Технология ML.NET

ML.NET, предложенная Microsoft в 2019 г., представляет собой инновационный инструмент для машинного обучения, специально созданный для разработчиков, работающих в среде .NET. Эта платформа предоставляет возможность интегрировать функционал машинного обучения в существующие приложения без необходимости глубоких знаний в области data science и математики, тем самым открывая доступ к развитым алгоритмам анализа данных и обработки информации.

ML.NET обертывает алгоритмы машинного обучения, позволяя чаще всего реализовывать использование этих алгоритмов через однократный вызов функции.

ML.NET реализует:

– технологии для классификационных задач и категоризационных задач;

– технологии регрессионного анализа для прогнозирования последовательностей;

– технологии выявления аномалий;

– рекомендательные технологии;

– технологии временные рядов;

– методы распознавания образов в изображениях;

– техники категоризации и анализа текстовых данных.

ML.NET — это фреймворк для машинного обучения в рамках Visual Studio 2019, представляющий собой компонент технологии искусственного интеллекта.

ML.NET задействует алгоритмы и статистические подходы из библиотек Visual Studio 2022 для автоматического обучения систем на базе .NET, минуя нужду во внешнем программном обеспечении.

В Visual Studio 2022 ML.NET используется для тренировки машинного автомата и предсказания будущих значений целевой переменной системы.

В Visual Studio 2022 ML.NET в основном применяется для обучения моделей и работы с данными.

В ходе машинного обучения через Visual Studio 2022 данные подаются в систему, после чего в этой же среде определяется наиболее подходящая модель ML.NET для тренировки системы

под .NET. По завершении этого процесса ML.NET в Visual Studio 2022 способна осуществлять прогнозы и отображать результаты визуально.

ML.NET поддерживает обучение с учителем и без учителя.

ML.NET представляет собой передовую платформу для машинного обучения, доступную в качестве проекта с открытым исходным кодом, совместимого с различными операционными системами, включая Windows, Linux и macOS. Это решение позволяет разрабатывать настраиваемые модели машинного обучения для широкого спектра приложений, включая консольные и настольные приложения, веб- и мобильные платформы, игры, а также устройства, работающие в рамках концепции Интернета вещей. Одной из ключевых особенностей ML.NET является его интеграция с другими инструментами и фреймворками для машинного обучения, такими как TensorFlow, Accord.NET и Microsoft Cognitive Toolkit (CNTK), что расширяет возможности разработчиков в создании сложных систем ИИ. В последних обновлениях ML.NET были реализованы функции для упрощения загрузки и обработки данных из реляционных баз данных (например, SQL Server, Oracle, MySQL), облегчая тем самым подготовку данных для обучения моделей. Кроме того, последняя версия ML.NET акцентирует внимание на развитии возможностей AutoML, что значительно упрощает разработку моделей машинного обучения, делая их более доступными для разработчиков без глубокой специализации в данной области.

ML.NET предлагает основную концепцию, важную для разработки приложений на основе машинного обучения.

Для точного прогнозирования результатов в ML.NET мы должны загружать обширные наборы данных для тренировки модели. Возможности ML.NET позволяют использовать различные источники данных для обучения и тестирования, включая текстовые файлы (CSV/TSV), реляционные базы данных (с поддержкой SQL Server, Oracle, MySQL и др.), двоичные файлы, IEnumerable и др.

Образование. Выбор адекватного алгоритма машинного обучения очень важен для разработки эффективной модели, соответствующей заданным целям и задачам, для последующего обучения и предсказания исходов.

Анализ. Для разработки и применения модели машинного обучения обязательно необходимо определить ее задачу. Для группировки данных по схожим характеристикам применяется метод кластеризации. Для предсказания количественных значений, например, цены или стоимости, идеально подходит регрессионный анализ. В то время как для определения принадлежности данных к определенной категории или классу, например, для анализа эмоциональной окраски отзывов, используется классификация.

Ожидаемые исходы. Используя информацию из этапов обучения и тестирования, конечное предсказание делается через приложение ML.NET с применением обученной модели. Эта модель, сохраненная в бинарном файле, может быть легко встроена в другие .NET-приложения, обеспечивая тем самым их расширение и интеграцию с возможностями машинного обучения.

Обсудим актуальные архитектуры и подходы в контексте ML.NET.

В Visual Studio 2022 для установки среды ML.NET используются инструменты пакетного менеджера NuGet.

Перечисляем требуемые программные модули.

```
using Microsoft.ML/*/;
using Microsoft.ML.Data/*/;
using Microsoft.ML.Models;
using Microsoft.ML.Data;
using Microsoft.ML.Trainers;
utilizing Microsoft Machine Learning Transformations;
using System;
utilizing System.Collections.Generic namespace;
using System.Linq;
using C = System.Console;
```

Для набора данных (dataset) создаем структуру хранения в виде класса.

Для обучения и проверки модели используем обучающую и тестовую выборки соответственно, организованные в виде классов.

Разрабатываем класс UnitData для хранения исходных данных и класс UnitPrediction для хранения прогнозируемых результатов.

```
public class UnitData/*/
{
   [Column("0")]/*/
   public float UnitA/*/;

   [Column("1")]
   public float UnitS/*/;

   [Column("2")]
   public float Volume;

   [Column("3")]
   public string Status;

   [Column("4")]
   public float Level;
}

public class UnitPrediction
{
   [ColumnName("Score")]
   public float Level;
}
```

Формируем набор данных, используя функционал GenerateUData.

```
internal static List<UnitData> CreateUDataEntries(int count)
{
   var pdlist = new List<UnitData>();
   Random randomGenerator = new Random();
for (int index = 0; index < limit; index++)
   {
      var pd = new UnitData
      {
         UnitA = random.Next(minValue: 0, maxValue: 20),
         UnitS = random.Next(minValue: 0, maxValue: 10),
Volume = random.Next(4, 30)
      };

      if ((pd.UnitA > 16 || pd.UnitS > 7) && pd.Volume is > 23)
         pd.Status is set to "Warning";
      else if ((pd.UnitA < 5 || pd.UnitS < 3) && pd.Volume < 15)
      pd.Status = "Warning";
      else
pd.Status = "Standard";

// Выражение вычисляет атрибут «Level» объекта «pd» путем
// суммирования значений атрибутов «UnitA», «UnitS» и «Объем»
```

```
// того же объекта, а затем вычитания случайного целого числа
// от 1 до 2 включительно, сгенерированного «random.Next(1, 3)»
     pdlist.Add(pd);
   }
   return pdlist;
}
```

Затем мы формируем файл для оптимальной идентификации и проводим тестирование набора данных. Файл для улучшенной идентификации включает в себя 2000 записей, в то время как тестовый набор содержит 20 записей.

```
var dataSet = CreateUniformData(2000);
var sampleData = CreateUserData(20);
Console.WriteLine("Contents of the Dataset:");
C.WriteLine("identifier,UnitAlpha,UnitSigma,Capacity,Condition,
Grade");
foreach (var entry in trainData.Take(20))
C.WriteLine(string.Join(",", item.UnitA, item.UnitS, item.Volume,
item.State, item.Grade));
```

Для оптимизации процесса распознавания содержимое файла трансформируется в набор данных, представленный через класс CollectionDataSource.

```
var trainDataset = CollectionDataSource.Initialize(trainData);
```

Разработка модели. Инициируем экземпляр LearningPipeline() и интегрируем в него необходимые элементы модели. В арсенале ML.NET представлен широкий спектр регрессионных алгоритмов для анализа данных. Для наших целей применим алгоритм регрессии Generalized Additive Model Regressor.

```
var dataProcessingPipeline = new LearningPipeline();
// Добавляем компоненты
// learningPipe, который включает набор данных trainCollection
// в свою последовательность
learningPipe.Add(new ColumnMapping(("Level", "Target")));
// Добавьте в learningPipe экземпляр CategoricalOneHotVec-
torizer для колонки "Status"
learningPipe.Add(new ColumnConcatenator("Attributes",
"UnitA", "UnitS", "Capacity", "Condition"));
// Добавляем алгоритм регрессии
learningPipe.Append(new GeneralizedAdditiveModelRegressor());
```

Упомянутые элементы в обучающем потоке развиваются через применение функции Train().

```
C.WriteLine("\nInitiating model training:");
// Переменной "model" присваивается результат процесса
// обучения, выполняемого "learningPipe", который использует
// "UnitData" в качестве входных данных и прогнозирует
// результаты, указанные "UnitPrediction"
```

Оценка точности. Класс RegressionEvaluator() предназначен для анализа регрессионных моделей, позволяя измерять их производительность через расчет среднеквадратичной ошибки и коэффициента детерминации.

```
var evaluator = new RegressionEvaluator();
var evaluationResults = evaluator.Assess(model, trainingDataset);
Console.WriteLine("\nModel Evaluation:");
Console.WriteLine("Root Mean Square: " + metrics.RootMeanSquare);
Console.WriteLine("Coefficient of Determination (R^2): " +
metrics.RSquared);
```

Предсказание результатов на тестовом наборе данных. Создаем прогностическую модель для тестового набора. Для сопоставления фактических и прогнозных значений выводим их одновременно.

```
var forecast = model.Forecast(testData);

// Отображение тестового набора данных вместе
// с прогнозируемыми результатами
var outcomes = testData.Zip(predicted, (actual, forecast) =>
Tuple.Create(t.UnitA, t.UnitS, t.Volume, t.Status, t.Level,
p.Level));

int i = 1;
Console.WriteLine("\nPredicted outcome:");
C.WriteLine("identifier,UnitAlpha,UnitSigma,Capacity,Condi-
tion,InitialLevel,ForecastedLevel");
foreach (var item in results)
{
C.WriteLine(string.Concat(i, ",", item.Item1,item.Item2,
item.Item3, item.Item4, item.Item5, item.Item6));
i++;
}
```

Демонстрировалось применение модели регрессии на языке C# в ML.NET.

Давайте проанализируем пример использования классификации с помощью ML.NET на языке C#.

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Models;
using Microsoft.MachineLearning.Api;
using Microsoft.ML.Trainers;
utilizing Microsoft.ML.DataTransforms;
using System;
utilizing System.Collections.Generic;
using System.Linq;
using C = System.Console;

namespace mltest
{
  Public class UnitData
  {
    [Column("0")]
    public float UnitA;

    [Column("1")]
    public float UnitS;

    [Column("2")]
    public float Volume;

    [Column("3")]
    public string Status;

    [Column("4")]
    public float Level;
  }

  public class ForecastingUnit
  {
    [ColumnName("Score")]
    public float Level;
  }

  class Program
  {
static void Main(string[] args)
    {
// Создание обучающих и оценочных наборов данных
      var trainSet = CreateTrainingData(2000);
```

```
        var testData = CreateUserData(20);
        C.WriteLine("Contents of the dataset:");
C.WriteLine("identifier,UnitAlpha,UnitSigma,Capacity,Condition,
Tier");
        foreach (var element in trainData.Take(20))
          C.WriteLine(string.Concat(item.UnitA, ",", item.UnitS,
item.Volume, item.State, item.Tier));

// Преобразование списка в совокупный источник данных
        var trainCollection =
CollectionDataSource.Establish(trainData);

        // Инициализируем модель машинного обучения
        var learningPipe = new LearningPipeline();
        // Добавляем компоненты
        learningPipe.Incorporate(trainCollection);
        learningPipe.Append(new ColumnCopyingEstimator(("Level",
"Label")));
        learningPipe.Add(new OneHotEncodingTransformer("Status"));
        learningPipe.Add(new ColumnConcatenator("Features",
"UnitA", "UnitS", "Capacity", "Condition"));
        // Реализация модели линейной регрессии
        learningPipe.Include(new Generalized Additive Model
Regressor());

        // Обучение модели
        C.WriteLine("\nInitiating model training:");
        var model = learningPipe.Fit<UnitData, UnitPrediction>();

        // Оценка производительности модели и проверка точности
        var evaluator = new RegressionEvaluator();
        var metrics = evaluator.CalculateMetrics(model,
trainingData);
        C.WriteLine(("\nAssessment of Model Performance:");
        Console.WriteLine("Root Mean Square: " +
metrics.RootMeanSquare);
        C.WriteLine displays the squared coefficient of
determination, indicated by metrics.RSquared.

        // Результаты прогноза по набору данных
var forecasted = model.Forecast(testingData);

        // Отображение тестового набора данных вместе
        // с результатами прогноза
        var outcome = testData.Zip(predicted, (trueValue,
predictedValue) =>
Tuple.Create(t.UnitA, t.UnitS, t.Volume, t.Status, t.Level,
p.Level));
```

```
       int i = 1;
       C.WriteLine("\nForecasted outcome:");
C.WriteLine("identifier,UnitAlpha,UnitSigma,Capacity,
Condition,InitialLevel,ForecastLevel");
       for each (var element in outcomes)
       {
         C.WriteLine(string.Join(",", i, item.Item1,
item.Item2, item.Item3, item.Item4, item.Item5, item.Item6));
         i++;
       }

       C.ReadLine();
     }

     internal static List<UnitData> ProduceUnitDataList(int
quantity)
     {
       List<UnitData> pdlist = new List<UnitData>();
       Random random = new Random();
       for (int index = 0; index < count; index++)
       {
var productDetails = new ProductInfo
         {
           UnitA = random.Next(0, 20),
           UnitS = random.Next(0, 10),
// Переменная "Volume" инициализируется случайным целым
// значением, которое генерируется в диапазоне от 4 до 29
// включительно с использованием "random.Next"
         };

// Если свойства pd, в частности, UnitA, превышают 16 или
// UnitS превышают 7, одновременно с объемом pd больше 23,
// условие выполняется
           pd.Status = "Caution";
         else if ((pd.UnitA < 5 || pd.UnitS < 3) && pd.Volume
< 15)
           pd.Status = "Warning";
         else
pd.Status = "Regular";

pd.Level = (pd.UnitA + pd.UnitS + pd.Volume) — ran-
dom.Next(1, 3);

         pdlist.Add(pd);
       }
       return pdlist;
     }
   }
}
```

Классификация с ML.NET на C#. В Visual Studio инициируем создание консольного приложения и приступаем к интеграции ML.NET, используя NuGet Package Manager для удобной загрузки и установки соответствующего пакета в среду разработки.

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Models;
using Microsoft.MachineLearning.Api;
using Microsoft.ML.Trainers;
implementing Microsoft.ML.Transforms;
using System;
utilizing System.Collections.Generic;
using System.Linq;
```

В процессе создания тестового и обучающего наборов данных для задачи классификации мы компилируем целевые значения в определенные категории. В этом контексте класс ProcessData служит в качестве репозитория для хранения обучающих и тестовых данных, в то время как класс ProcessPrediction предназначен для хранения результатов предсказаний, выполненных на основе этих данных.

```
public class ProcessData
{
   [Column("0")]
   public float UnitA;

   [Column("1")]
   public float UnitS;

   [Column("2")]
   public float Volume;

   [Column("3")]
   [ColumnName("Label")]
   public string Label;
}
public class ProcessForecast
{
[ColumnName("PredictedOutcome")]
public string PredictedLabels;
}
```

Применяем метод GeneratePData() для создания примеров данных, что облегчает подготовку файла для точного анализа

и оценку качества данных. В этом процессе доступна проверка алгоритма генерации данных.

```
private static List<ProcessData> CreateProcessDataList(int
count, bool test = false)
{
  List<ProcessData> pdlist = new List<ProcessData>();
  var randomGenerator = new Random();
  for (int index = 0; index < count; index++)
  {
    ProcessData pd = new var
    {
// UnitA присваивается значение с помощью генератора
// случайных чисел в диапазоне от 0 до 119
UnitS = random.Next(0, 60),
      Volume = random.NextInt64(40, 280)
    };

    if (!test)
    {

pd.Label = "Alert";
else if ((pd.UnitA < 25 || pd.UnitS < 20) && pd.Volume < 100)
        pd.Label = "Warning";
      else
        pd.Label = "Typical";
    }
    else
pd.Label = "";

    pdlist.Add(pd);
  }
  return pdlist;
}
```

Для верификации данных внутри используем метод Print(), выводящий элементы списка на экран.

```
private static void Display(IEnumerable<ProcessData> outcomes)
{
  int i = 1;
  Console.WriteLine("id,UnitA,UnitS,Volume,Label");
  for (var item in results)
  {
    Console.WriteLine(string.Join(",", i,item.UnitA, item.UnitS,
item.Amount, item.Description));
    i++;
  }
}
```

Мы генерируем файл, содержащий 2000 элементов для улучшения качества распознавания, и набор из 50 элементов для проверки, используя функцию GeneratePData().

```
var trainData = ProducePseudoData(2000);
Print(trainData.Take(20));

var testData = CreatePseudoData(50, isTest: true);
Print(testData.Take(20));
```

Мы подготовили наборы данных для обучения и тестирования. Теперь следующим шагом является разработка модели с использованием класса LearningPipeline из фреймворка ML.NET.

```
var learningPipeline = new LearningPipeline();
```

Содержимое списка необходимо конвертировать в формат CollectionDataSource для последующей интеграции в структуру LearningPipeline.

```
var trainCollection = CollectionDataSource.From(trainData);
learningPipe.Append(trainCollection);
```

Конвертация меток в числовые индексы.

```
learningPipe.Add(new DictionaryLabeler("Label"));
```

Определение столбцов функций.

```
learningPipe.Add(new ColumnConcatenator("Features", new[] {
"UnitA", "UnitS", "Volume" }));
```

Реализация метода классификации.

```
learningPipe.Add(new SDCARegressionTrainer());
// Интеграция конвертера прогнозируемых меток
PredictedLabelColumnOriginalValueConverter в learningPipe
{ PredictedLabelColumn = "ForecastOutcome" });
```

Обучение модели

Интегрируем ClassificationEvaluator() для оценки эффективности модели, затем инициируем выполнение программы для ее валидации. С использованием этого инструмента анализа мы способны извлечь данные по матрице ошибок и показателям потерь.

```
var evaluator = new ClassificationEvaluator();
metrics = evaluator.CalculatePerformance(model,
trainingDataset);
Console.WriteLine("Micro-accuracy: " + metrics.MicroAccuracy);
Console.WriteLine("Displaying Log Loss metric: " +
metrics.LogLoss);
```

При использовании функции автоматического внедрения процедуры нормализации MinMaxScaler применяйте параметры «norm=Warn» или «norm=No» для деактивации данного механизма.

Сборка тестовых и предсказанных данных производится перед их выводом на печать.

```
var outcomes = testData.Zip(predicted, (t, p) => new AnalyzeData
{
  UnitA = t.UnitA,
  UnitS = t.UnitS,
  Volume = t.Volume,
  Label = p.PredictedLabels

}).ToList();

Print(results);
```

Реализации регрессионного анализа на языке C#

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Models;
using Microsoft.ML.Legacy;
using Microsoft.ML.Trainers;
using Microsoft.ML.DataTransforms;
using System;
utilizing System.Collections.Generic;
using System.Linq;

namespace mltest
{
  public class ProcessData
  {
    [Column("0")]
    public float UnitA;

    [Column("1")]
    public float UnitS;

    [Column("2")]
    public float Volume;
```

```
    [Column("3")]
    [ColumnName("Label")]
    public string Label;
  }

  public class ForecastProcess
  {
    [AttributeName("ForecastedOutcome")]
    string PredictedLabels;
  }

  class Program
  {
    static void Main(string[] args)
    {
      // Создание обучающих и оценочных наборов данных
var trainingDataset = CreatePredictiveData(2000);
// Отображаем первые 20 записей из набора trainData
      var testData = ProducePseudoData(50, testMode:true);

// Создание объекта для проекта машинного обучения
      var learningPipe = new LearningPipeline();
      var trainCollection =
CollectionDataSource.From(trainData);
      learningPipe.Include(trainCollection);
// Преобразование текстовых меток в числовые индексы
      learningPipe.Add(new Dictionarizer("Target"));

      learningPipe.Add(
        new ColumnConcatenator("Features", "UnitA",
"UnitS", "Volume"));

      // Процесс категоризации
      learningPipe.Add(new SDCARegressionTrainer());

      // Преобразование прогнозируемого значения столбца
      // меток
      learningPipe.Add(new
PredictedLabelColumnOriginalValueConverter()
{ PredictedLabelColumn = "PredictedLabel" });

      // Модель обучения
      var predictionModel = learningPipeline.Fit<ProcessData,
ProcessPrediction>();

      // Оценка эффективности модели и проверка точности
      ClassificationEvaluator evaluator = new
ClassificationEvaluator();
      var measurements = evaluator.Assess(model, trainingSet);
```

```
       Console.WriteLine("AccuracyMicro: " +
metrics.AccuracyMicro);
       Console.WriteLine("Logarithmic Loss: " +
metrics.LogLoss);

       // Прогноз результатов на основе оценки данных
       var prediction = model.Forecast(testData);

       // Получаем тестовые данные и классы прогнозов
       var outcomes = testData.Zip(predicted, (test, pred)
=> new ProcessData
       {
         UnitA is assigned the value of t.UnitA,
UnitSize = time.UnitSize,
         Volume = transaction.Volume,
Variable = p.PredictedTags

       }).ToList();

       // Отображение результатов выполнения
DisplayOutput(results);

// Console.ReadLine() метод в C# используется для чтения
// следующей строки символов из стандартного ввода
       }

       private static void Display(IEnumerable<ProcessData>
outcomes)
       {
         int i = 1;
Console.WriteLine(string.Join(",", new string[] {"id",
"UnitA"}));
"MeasurementUnits", "Capacity", "Identifier"));
         foreach (var item in results)
         {
           Console.WriteLine($"{i},{item.UnitA},{item.UnitS},");
item.Capacity, item.Identifier));
           i++;
         }
       }

       private static List<ProcessData>
CreateProcessDataList(int quantity, bool isTestMode = false)
       {
List<ProcessData> pdList = new();
         Random random = new Random();
for (int index = 0; index < total; index++)
         {
var pd = new ProcessData
           {
             UnitA = random.NextInt32(0, 120),
```

```
                UnitS = random.Next(minValue: 0, maxValue: 60),
                Volume = random.randint(40, 280)
            };

            if (!test)
            {
If either pd.UnitA exceeds 75 or pd.UnitS is above 30, and
concurrently, pd.Volume surpasses 230
                pd.Label = "Caution";
              otherwise, if (either pd.UnitA is below 25 or
pd.UnitS falls short of 20) and pd.Volume is less than 100
                pd.Label = "Warning";
              else
                pd.Label = "Typical";
            }
            else
              pd.Label = "";

            pdlist.Insert(pd);
        }
        return pdlist;
      }
   }
}
```

Особенности ML.NET. Microsoft ML.NET обеспечивает широкие возможности для машинного обучения, включая классификацию текста, прогнозирование числовых данных, обнаружение аномалий, рекомендательные системы и обработку изображений.

Используя DotNet, можно разработать программное обеспечение для машинного обучения благодаря фреймворку ML.NET.

Для программирования с ML.NET подходят C# и F#.

ML.NET — это кросс-платформенная, открытая разработка с исходным кодом.

ML.NET поддерживает разработку и выполнение на платформах Windows, Linux и macOS.

Активно применяется в Microsoft Windows, Bing, Azure, и поддерживает внедрение в сторонние среды, например, TensorFlow, CNTK и Accord.NET.

ML.NET обеспечивает создание приложений на основе машинного обучения для веб-разработок, мобильных платформ, настольных систем, игровых проектов и Интернета вещей.

ML.NET сохраняет тренированную модель в формате бинарного файла, который затем может быть интегрирован в различные DotNet-приложения.

Компания Microsoft неустанно обогащает свои продукты рядом новшеств, например интеграцию продвинутых методов глубокого обучения с использованием TensorFlow и Cognitive Toolkit (CNTK).

В раннем выпуске ML.NET 0.2 был внедрен набор функциональностей для выполнения задач кластеризации в рамках машинного обучения. В предварительной версии ML.NET 0.5 был включен механизм оценки моделей TensorFlow. В предварительной версии ML.NET 0.6 расширены функции, позволяя оценивать уже обученные ONNX модели.

С выпуском версии ML.NET 0.7 платформа расширила свою поддержку, включив в себя архитектуры как x86, так и x64. ML.NET, находясь в стадии предварительного выпуска, постоянно обновляется Microsoft, при этом в платформу интегрируются новые функциональные возможности. В то время как более ранние версии ML.NET были сосредоточены исключительно на поддержке x64, начиная с текущей версии, разработчики могут вести разработку под обе архитектуры — как x86, так и x64.

Первая тестовая версия ML.NET 0.7 внедряет на стадии испытаний интеграционные Python-библиотеки NimbusML для ML.NET.

В предварительной версии ML.NET 0.7 представлены функционалы для выявления аномалий.

ML.NET строится вокруг модели машинного обучения, задающей последовательность действий для генерации прогнозов на базе вводимой информации. Эта платформа дает возможность разработать индивидуальные модели с выбором нужного алгоритма и поддерживает загрузку готовых моделей TensorFlow и ONNX.

Разработанную модель можно интегрировать в программное обеспечение и применять ее для прогнозирования исходов.

ML.NET функционирует на операционных системах Linux, Windows и macOS при использовании .NET Core, а на Windows также совместим с .NET Framework. Версия для 64-битных систем доступна на всех перечисленных платформах, в то время как

32-битная версия ограничена работой на Windows и не поддерживает определенные возможности, связанные с LightGBM, TensorFlow и ONNX.

ML.NET поддерживает генерацию прогнозов разнообразных категорий.

Классификация/категоризация — автоматизированная классификация отзывов от клиентов на позитивные и негативные.

Прогнозирование регрессии для предсказания непрерывных переменных — например, оценка цены недвижимости с учетом ее площади и местоположения.

Обнаружение аномалий — к примеру, выявление мошеннических операций по банковским счетам.

Рекомендации — например, рекомендация товаров, соответствующих интересам клиентов онлайн-магазина, исходя из истории их предшествующих заказов.

Серии данных по времени/последовательности данных — к примеру, метеорологические прогнозы и прогнозирование объемов продаж.

Классификация изображений — например, диагностика аномалий на изображениях, полученных с медицинского оборудования.

Классификация текста — например, проведите категоризацию текстов на основании их контента.

Сходство предложений — можно оценить степень схожести двух фраз.

Пример

Обсудим участок кода из базового приложения на ML.NET. В демонстрации создается модель линейной регрессии, предназначенная для предсказания стоимости недвижимости на основе ее площади и первоначальной цены.

```
Using System;
using Microsoft.ML;
using Microsoft.ML.Data;

class Program
{
  class HouseData
  {
    public float Dimensions { get; set; }
```

```
    public float Price { get; set; }
  }

public interface Prediction
  {
    [AttributeName("Score")]
public float Price { get; set; }
  }

Define Main(string[] args) as Empty
  {
    MLContext mlContext = new MLContext();

    // 1. Получение или разработка наборов данных
    // для обучения
    HouseData houseData[] = {
    () { Dimension = 1.1F, Cost = 1.2F },
new HouseData() { Area = 1.9F, Cost = 2.3F },
new HouseData() { Dimension = 2.8F, Cost = 3.0F },
      new HouseData() { Size = 3.4F, Price = 3.7F } };
IDataView trainingSet =
Context.Data.LoadFromEnumerable(PropertyData);

// 2. Описание схемы предварительной обработки данных
// и разработки модели
Pipelinee = Context.Data.Preprocessing.Combine("Features",
new[] { "Size" })
.Attach(mlContext.Regression.Trainers.Sdca(labelColumnName:
"Price", maximumIterations: 100));

    // 3. Обучение модели
    var model = pipeline.Train(trainingData);

    // 4. Прогнозирование результатов
  var size = new HouseData() { Size = 2.5f };
// Переменной "var price" присваивается результат
// использования метода Predict, который создан для
// прогнозирования моделей данных о домах, созданных
// на основе "Context.Model" с переданной конкретной моделью.
// Этот механизм, предназначенный для прогнозирования
// значений на основе данных о доме, использует заданный
// параметр "Size" для своего прогнозирования

    Console.WriteLine($"Estimated market value for a property
with {size.Size*1000} square feet area: {price.Price*100:C}
thousand dollars");

The estimated market value for a property spanning 2500
square feet is approximately $261,980.
  }
}
```

Процесс использования программного обеспечения. На рис. 14 демонстрируется архитектура программного обеспечения и цикличный процесс создания модели:

– коллекция и интеграция тренировочных данных в структуру IDataView;

– определение последовательности операций по извлечению признаков и использование алгоритма машинного обучения;

– тренировка модели через вызов метода Fit() в рамках шагов, которые выполняются, чтобы создать и запустить программу или приложение;

– установление эквивалентности достижений заданным нормам и стандартам модели и циклы оптимизации для повышения ее качества;

– сериализация модели в бинарный формат для интеграции в приложение;

– инициализация ITransformer из сохраненной модели;

– прогнозирование данных через метод CreatePrediction-Engine.Predict().

Модель ML.NET. Эта модель — сущность, включающая набор трансформаций для выработки предсказаний, основываясь на анализе данных, представленных пользователем.

Basic. Наиболее элементарная форма — это бивариантная линейная регрессия, представляющая собой взаимосвязь, где одна непрерывная переменная изменяется прямо пропорционально другой.

Более сложная модель. Усовершенствованная модель категоризации транзакций применяет анализ текстовых данных для классификации операций.

Для анализа транзакций их описания декомпозируют на составляющие элементы, исключая нерелевантный текст и символы, а затем производится количественный подсчет слов и символов. Эти составляющие элементы служат базой для тренировки линейной регрессионной модели, основанной на множестве категорий из тренировочного датасета. Схожесть новых описаний с уже существующими в тренировочном наборе данных повышает вероятность корректного категоризирования аналогичных транзакций.

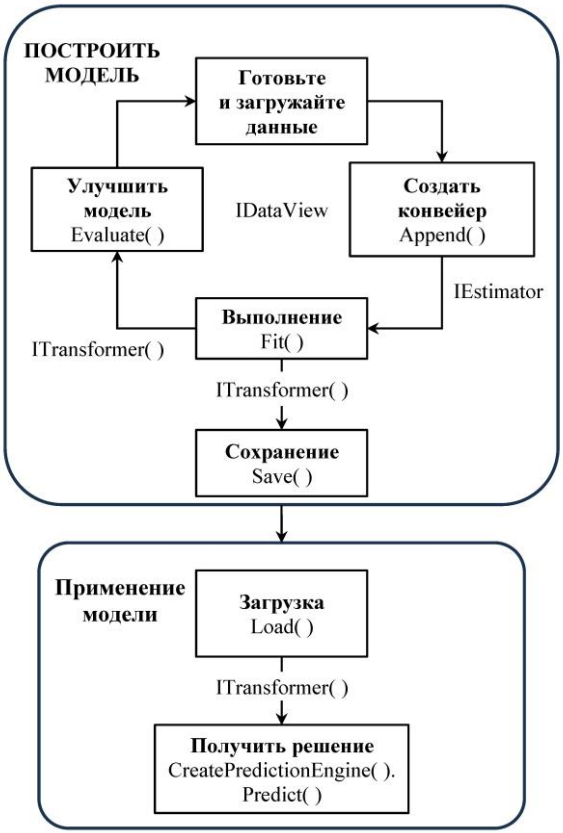


Рис. 14. Цикличный процесс создания модели

В примере, где приведены линейные модели (табл. 5), модели прогнозирования стоимости жилья и классификации текстов основаны на линейных алгоритмах. Однако варьирование типов данных и поставленных целей требует рассмотрения других подходов, например, использования деревьев решений, обобщенных аддитивных моделей и прочих методик.

Подготовка данных. В большинстве ситуаций данные не сразу подходят для тренировки алгоритмов машинного обучения в своем первоначальном формате. Обработка данных представляет собой ключевой этап, предшествующий их использованию

для оптимизации модели. Часто необходимо выполнить ряд преобразований, включая конвертацию из текстовых форматов в числовые, удаление лишней информации для очистки данных, регулировку объема данных, их масштабирование или нормализацию для улучшения качества и эффективности обучения модели.

Архитектура ML.NET. Разработка на ML.NET начинается с создания экземпляра MLContext. Это фабрика по созданию компонентов для загрузки данных, преобразования данных, обучения моделей, прогнозирования и операций сохранения/загрузки моделей.

Инициализация платформы ML.NET и архивирование данных

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v4.0.1.

Класс, используемый для создания компонентов, которые работают с данными, но не являются частью тренировочных этапов создания и запуска модели. Включает компоненты для загрузки, сохранения, кеширования, сортировки, перемешивания и разбиения данных.

C#

```
public sealed class DataOperationsCatalog
```

Наследование: Object → DataOperationsCatalog.

Подготовка данных

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для создания экземпляров компонентов преобразования.

C#

```
public sealed class TransformsCatalog
```

Наследование: Object → TransformsCatalog.

Алгоритмы обучения

Двоичная классификация

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для создания экземпляров компонентов двоичной классификации, таких как средства обучения и калибраторы.

C#

```
public sealed class BinaryClassificationCatalog :
Microsoft.ML.TrainCatalogBase
```

Наследование: Object → TrainCatalogBase → BinaryClassificationCatalog.

Многоклассовая классификация

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для создания экземпляров компонентов многоклассовой классификации, таких как обучающие объекты.

C#

```
public sealed class MulticlassClassificationCatalog :
Microsoft.ML.TrainCatalogBase
```

Наследование: Object → TrainCatalogBase → MulticlassClassificationCatalog.

Обнаружение аномалий

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, применяемый для создания экземпляров компонентов в среде MLContext, включая инструменты для обучения и анализа в задачах выявления аномалий.

C#

```
public sealed class AnomalyDetectionCatalog :
Microsoft.ML.TrainCatalogBase
```

Наследование: Object → TrainCatalogBase → AnomalyDetectionCatalog.

Прогнозирование

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый MLContext для создания экземпляров компонентов прогнозирования.

C#

```
public sealed class ForecastingCatalog :
Microsoft.ML.TrainCatalogBase
```

Наследование: Object → TrainCatalogBase → Forecasting-Catalog.

Ранжирование

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, применяемый в MLContext для создания элементов ранжирования, включая обучающие модели и модули оценки.

C#

```
public sealed class RankingCatalog :
Microsoft.ML.TrainCatalogBase
```

Наследование: Object → TrainCatalogBase → Ranking-Catalog.

Регрессия

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, применяемый в MLContext для создания элементов регрессии, включая обучающие алгоритмы и анализаторы результатов.

C#

```
public sealed class RegressionCatalog :
Microsoft.ML.TrainCatalogBase
```

Наследование: Object → TrainCatalogBase → Regression-Catalog.

Использование модели

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для сохранения и загрузки обученных моделей.

C#

```
public sealed class ModelOperationsCatalog
```

Наследование: Object → Model Operations Catalog.

Через любую из вышеупомянутых категорий доступен доступ к соответствующим способам разработки. В Visual Studio для отображения каталогов применяется функционал IntelliSense.

Сборка конвейера

В любом директории представлен набор функций-расширений. Рассмотрим их применение в процессе построения обучающего конвейера.

```
var pipeline = mlContext.Data.Processors.Combine("Attributes",
new[] { "Dimension" })
.Add(mlContext.Regression.Trainers.SdcaRegression(labelColumn-
Name: "Price", maximumNumberOfIterations: 100));
```

Эта фаза ограничивается только разработкой сущностей, не переходя к их реализации.

Обучение модели

После инициализации объектов в процессе обработки данных их можно применять для тренировки алгоритма.

```
var trainedModel = pipeline.TrainOn(trainingDataSet);
```

Использование метода Fit() позволяет применять наши тренировочные данные для настройки параметров модели — процесс, известный как ее обучение. В контексте обсуждаемой линейной регрессии модель определяется двумя ключевыми параметрами: смещением и весом. Выполнение метода Fit() обеспечивает определение конкретных значений этих параметров. Стоит заметить, что для большинства моделей машинного обучения количество параметров значительно превышает два.

Анализируем процесс согласно структуре машинного обучения.

Разбиение данных на тренировочные и тестовые выборки. Задача алгоритма машинного обучения заключается в выявлении структур в тренировочном наборе данных. Эти выявленные структуры применяются для генерации прогнозов, основываясь на данных, которые не встречались ранее.

Произведем разбиение исходного массива данных на две части: обучающую и тестовую, с применением функции TrainTestSplit. Это действие приведет к созданию экземпляра TrainTestData, который будет включать в себя два элемента типа IDataView: первый элемент предназначен для обучающей выборки, второй — для тестовой. Процентное соотношение данных, отводимых под тестовую выборку, устанавливается через аргумент testFraction. В приведенном ниже примере кода для тестового набора данные выделены в объеме 20 % от всего массива данных.

```
DataOperationsCatalog.TrainTestData dataPartition =
mlContext.Data.TrainTestSplit(data, testFraction: 0.2);
IDataView trainingDataset = dataSplit.TrainingData;
IDataView testData = dataSplit.TestSet;
```

Подготовка данных. Перед тренировкой модели ИИ следует выполнить предобработку данных. ML.NET-алгоритмы требуют специфичных форматов входящих колонок. Отсутствующие значения автоматически получают стандартные имена для входных и выходных колонок.

Управление предполагаемыми типами данных в столбцах. В ML.NET алгоритмы машинного обучения требуют ввода в форме вектора с дробными числами фиксированной длины. При условии, что все данные уже конвертированы в числа и должны быть обработаны параллельно (например, пиксели изображений), используйте атрибут VectorType для соответствующей структуры данных.

Когда столкнетесь с неоднородными данными, а задача требует применить различные методы трансформации к каждому атрибуту, воспользуйтесь функцией Concatenate после обработки индивидуальных атрибутов. Это позволит агрегировать обработанные данные в единую структуру, формируя новый атрибут из векторов, представляющих собой обработанные значения каждого столбца.

В данном кодовом сегменте колонки «Size» и «HistoricalPrices» агрегируются в единый вектор элементов, представляющий собой исходные данные для создания новой колонки, именуемой «Features». В силу однородности масштабов данных метод «NormalizeMinMax» используется к колонке «Features» в целях выполнения нормализации значений.

```
// 1. Метод предварительной оценки данных.
// Объединяем атрибуты размера и истории в единый вектор
// признаков, сохраненный в недавно созданном столбце
// с именем «Features».
// 2. Стандартизировать набор функций
IEstimator<ITransformer> dataPreparationEstimator =
mlContext.Transforms.CombineColumns("Features", new[]
{"Size", "HistoricalPrices"})
.Append(mlContext.Transforms.NormalizeMinMax("Features"));

// Инициализация преобразования подготовки данных
ITransformer dataPreparationTransformer =
dataPreparationEstimator.Fit(trainingDataSet);
// Выполнение преобразования на обучающем наборе данных
IDataView processedTrainingData =
dataPreparationTransformer.Apply(trainData);
```

Обработка стандартных наименований колонок. При отсутствии явного указания названий колонок система машинного обучения ML.NET автоматически присваивает стандартные имена. Для обработки входных данных каждый экземпляр алгоритма обладает параметром под наименованием featureColumnName. Когда это необходимо, предусмотрен дополнительный параметр labelColumnName для определения целевого значения, к которому должна стремиться модель. По умолчанию эти параметры называются Features и Label соответственно.

В процессе предобработки данных, когда применяется операция Concatenate для формирования объединенного столбца под названием Features, нет необходимости явно указывать имя этого столбца в настройках алгоритма машинного обучения, поскольку он уже включен в подготовленный набор данных в формате IDataView. Что касается столбца с метками, именуемого CurrentPrice, его имя автоматически модифицируется в Label благодаря использованию атрибута ColumnName в структуре данных. Это действие, совершенное библиотекой ML.NET, избавляет пользователя от необходимости задавать имя столбца с метками (labelColumnName) при настройке оценки производительности выбранной модели машинного обучения.

При отказе от использования предопределенных наименований столбцов необходимо явно указать названия атрибутов и целевых переменных, передавая их как аргументы при конфигурации модели машинного обучения.

Это демонстрируется в приведенном ниже коде:

```
var UserDefinedColumnSdcaEstimator =
mlContext.Regression.Trainers.Sdca(labelColumnName:
"MyLabelColumnName", featureColumnName: "MyFeatureColumnName");
```

Кеширование данных. По умолчанию в процессе обработки данных их внесение или стриминг происходит в режиме «ленивой» загрузки. Это подразумевает, что алгоритмы машинного обучения в процессе тренировки могут многократно осуществлять чтение данных с накопителя и последующую обработку. С целью уменьшить объем операций чтения данных адекватным решением является применение кеширования данных в оперативной памяти. Кеширование осуществляется в рамках EstimatorChain через механизм AppendCacheCheckpoint.

Для повышения эффективности обработки данных перед применением алгоритмов машинного обучения рекомендуется располагать метод AppendCacheCheckpoint в начале конвейера данных.

Через внедрение метода AppendCacheCheckpoint в класс EstimatorChain перед использованием StochasticDualCoordinate-Ascent алгоритм обучения оптимизирует процесс, сохраняя вычисления предшествующих оценщиков для дальнейшего применения.

```
// 1. Объединить атрибуты Size и Historical в единое
// векторное представление и назначить этот результирующий
// вектор новому введенному столбцу с названием Features
// 2. Стандартизировать вектор Attributes
// 3. Кешировать подготовленные данные
// 4. Реализовать алгоритм Sdca для обучения модели
IEstimator<ITransformer> preparationPipeline =
mlContext.Transforms.Concatenate("Features", new[] {"Size",
"HistoricalPrices"})
.Add(mlContext.Transforms.NormalizeMinMax("Features"))
.AddCacheCheckpoint(mlContext);
.AddTrainer(mlContext.Regression.Trainers.SdcaRegression());
```

Тренировка алгоритма искусственного интеллекта. После выполнения этапа подготовки данных следует применить функцию Fit() для тренировки модели машинного обучения, используя алгоритм регрессии Stochastic Dual Coordinate Ascent.

```
// Инициализация стохастической двухкоординатной
// регрессионной модели оценки
var regressionEstimator =
mlContext.Regression.Trainers.SdcaNonCalibrated();
```

Построение алгоритма искусственного интеллекта

```
var model = sdcaEstimator.Train(transformedTrainingDataSet);
```

Извлечение параметров модели. После тренировки модели экспортируйте получившиеся ModelParameters для анализа или дальнейшего обучения. Параметры линейной регрессии (LinearRegressionModelParameters) включают смещение.

```
var trainedModelParameters =
trainedModel.Model as LinearRegressionModelParameters;
```

Использование модели. Данные можно трансформировать в предсказания двумя способами: параллельно или последовательно. В нашем эксперименте мы применили обе методики: параллельную трансформацию для оценки эффективности модели и последовательную для генерации индивидуальных предсказаний. Давайте подробнее рассмотрим процесс формирования отдельных прогнозов.

Функция CreatePredictionEngine() требует указать классы для входных и выходных данных. Соответствие между полями классов и столбцами в датасете устанавливается через имена полей или атрибуты в коде, что критически важно для обучения и последующего предсказания модели.

Модели и схема данных. Основой архитектуры машинного обучения ML.NET являются объекты DataView.

Каждый этап в потоковой обработке данных требует определенной входной структуры (описание требуемых данных по имени, типу и объему) и генерирует результат с заданной выходной структурой (конкретизация результата работы по имени, типу и объему данных).

Когда выходные данные одного этапа преобразования в конвейере обработки данных не совпадают с ожидаемыми входными данными следующего этапа, ML.NET генерирует ошибку в виде исключения.

Структура данных для представления информации организована в таблицу, состоящую из колонок и записей. Каждая колонка имеет уникальное наименование, определенный тип данных и размер. В качестве примера, рассматривая обработанный и структурированный массив данных с информацией о стоимости жилья, можно выделить колонки «Size» и «Price». Эти атрибуты характеризуются конкретными значениями и классифицируются как скалярные данные, в отличие от векторных (рис. 15).

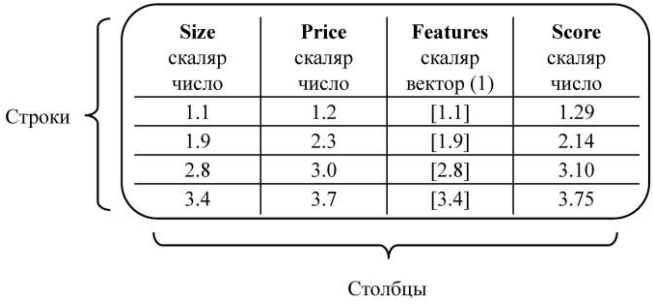


Рис. 15. Структура данных для представления информации о стоимости жилья

ML.NET framework обрабатывает данные через векторные столбцы. Стандартное наименование такого столбца — «Features». В демонстрационном проекте по прогнозированию стоимости жилья мы интегрировали столбец «Size» в вектор «Features» для анализа.

Дополнительно после завершения процесса прогнозирования различные алгоритмы машинного обучения порождают дополнительные колонки данных. Названия этих дополнений строго определены и зависят от конкретного применяемого метода в машинном обучении. Например, в контексте выполнения регрессионного анализа одна из таких добавленных колонок именуется, как «Score», что стало причиной для того, чтобы мы именовали наш целевой признак данных, как «Price».

```
public class Prediction
{
  [ColumnName("Score")]
  public float Price { get; set; }
}
```

Ключевой характеристикой компонентов DataView в библиотеке ML.NET является их простая вычислительная модель. Это означает, что данные, представленные этими объектами, активно загружаются и применяются исключительно в процессах обучения, оценивания моделей и в ситуациях, когда требуется генерация прогнозов. В процессе разработки и тестирования решений, основанных на ML.NET, разработчики имеют возможность воспользоваться возможностями отладчика Visual Studio, чтобы осуществить визуализацию и анализ любого объекта DataView при помощи специализированного метода для предварительного просмотра данных.

```
var debugLog = testPriceDataView.GeneratePreview();
```

В отладчике возможно провести анализ значения переменной debug, оценив ее данные. Отказ от применения функции предпросмотра в коде рекомендуется из-за ее негативного влияния на эффективность выполнения программы.

Развертывание модели. В практических реализациях задач машинного обучения процессы тренировки и оценки эффективности модели обычно разграничиваются с этапом прогнозирования. Такой подход обусловлен тем, что выполнение данных операций часто находится в ведении различных команд специалистов.

```
mlContext.Model.Save(model, trainingData.Schema, "model.zip");
```

Выводы по главе 3

Обсуждается важнейшая задача национальной программы «Цифровая экономика Российской Федерации»: создание и адаптация датасетов для использования в искусственном интеллекте через механизмы машинного обучения. В контексте этой программы, управление и анализ объемных данных становится критически значимым аспектом. Это обсуждение касается основы, которая будет поддерживать прогресс и экономическое будущее страны. В настоящий момент решение сложностей, связанных с обработкой больших объемов информации, осуществимо благодаря применению разработок в сфере искусственного интеллекта.

Изучены линейные алгоритмы, предполагающие, что прогноз формируется как взвешенная сумма атрибутов объекта, умноженных на их веса. Веса атрибутов определяются на этапе обучения с использованием ML.NET, отражая влияние атрибутов на результат.

Демонстрируется, что широкое применение моделей на основе линейных корреляций опирается на легкость их понимания и проверки; подчеркивается, что точность прогнозов не всегда удовлетворяет потребности пользователя.

В учебных материалах студенты изучают регрессионные и классификационные модели, представляющие собой ключевые инструменты в области искусственного интеллекта.

В главе описаны методы оценки эффективности для регрессионных и классификационных моделей, включая анализ не связанных с Интернетом показателей.

В области агентских технологий ключевыми целями являются задания, включающие в себя алгоритмы определения необходимых действий, создающие цепочки операций для достижения заданных результатов.

В пособии рассмотрена передовая платформа машинного обучения ML.NET (Microsoft, 2019), предназначенная для разработчиков в среде .NET. Этот инструмент дает возможность разработчикам обогащать функционал своих приложений.

Отмечено, что ML.NET создает передовые алгоритмы машинного обучения, которые значительно упрощают применение этих алгоритмов до уровня простого вызова функции.

Вопросы для самоконтроля

1. Модели и поиск в машинном обучении. Варианты моделей.
2. Инструменты для решения задачи регрессии.
3. Назначение библиотек Vowpal Wabbit и Sklearn.
4. Что нужно предсказать в задаче регрессии?
5. В чем состоит обучение линейной модели?
6. Что такое метрики и варианты метрик?
7. Офлайн-метрики для задачи регрессии.
8. Подсчет метрик регрессии.
9. Приведите примеры интерпретации метрик.

10. Суть технологии ИИ «Агенты, решающие задачи».
11. Что подается на вход алгоритма поиска?
12. Как и кто определяет дерево целей и критериев задачи поиска?
13. Алгоритмы поиска решений?
14. Что известно о стратегии неинформированного поиска?
15. Что такое технология ML.NET?
16. Какие задачи ИИ можно решать в ML.NET?
17. Базовая концепция ML.NET, которую необходимо использовать для разработки приложений машинного обучения.
18. Особенности применения и возможности Microsoft ML.NET.
19. Особенность решения задач классификации/категоризации.
20. Особенности решения задач регрессии для прогнозирования непрерывных значений.
21. Последовательность работы с кодом приложения машинного обучения.
22. Что представляет собой модель ML.NET — базовая модель?
23. Подготовка данных и оценка модели.
24. Назначение архитектурных шаблонов ML.NET.
25. Обращение к основным библиотекам ML.NET.
26. Сборка конвейера и обучение модели.
27. Зачем необходимо разделение данных на обучающий и проверочный наборы?
28. Что дает работа с ожидаемыми типами столбцов?
29. Что значит «работа с именами столбцов по умолчанию»?
30. Зачем необходимо обучение модели машинного обучения?
31. Построитель моделей и принципы его работы.
32. Какие проекты создает Model Builder? Чем полезен консольный вариант?

Глава 4 | Нейронные сети

4.1. Искусственный нейрон как основа нейронных сетей

Головной мозг человека представляет сложнейшую биологическую систему, содержащую множество биологических нейронов, функционирующую и представляющую естественный интеллект, принимающий данные от органов чувств (глаза, уши, нос), каким-то образом в результате обработки порождает информацию, позволяющую узнавать лица, распознавать речь, ощущать запахи и т. п.

На основе полученных данных мозг дает команды различным исполнительным органам (пожать руку знакомому человеку, пойти к доске по просьбе учителя, покинуть помещение при появлении дыма).

Будем предполагать, что нейроны головного мозга образуют некоторый вариант сети, похожую на искусственную компьютерную модель. В головном мозге человека физиологи определили порядка 90 млрд нейронов, которые соединены в различные структуры миллиардными связями. Упрощенное строение биологического нейрона показано на рис. 16.

Дендрит — получает информацию. **Синапс** — место контакта между двумя нейронами и передачи импульса. **Аксон** — импульсы идут от тела клетки к другому нейрону.

Биологический нейрон — это нервная клетка, которая состоит из тела, дендритов и аксона.

Тело соединено со множеством коротких и толстых отростков, которые называются дендритами. Это входные отростки — они принимают импульсы с тех нейронов, которые находятся в сети раньше, чем этот нейрон.

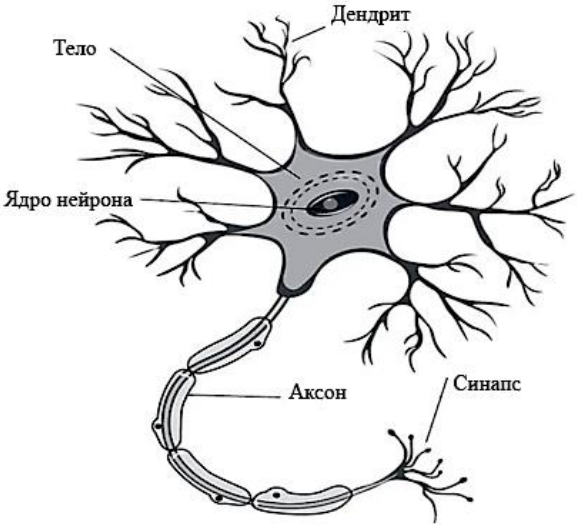


Рис. 16. Упрощенная схема нейрона

Имеется один очень длинный и тонкий отросток, который называется аксоном. Это выходной отросток, по которому нейрон передает электрохимический импульс следующим нейронам в сети. Через множество дендритов нейрон получает входные сигналы, в теле нейрона они обрабатываются, а через единственный аксон выходной импульс от нейрона передается дальше. Окончание аксона имеет разветвления, через которые выходной сигнал может быть передан нескольким следующим нейронам. Реальный биологический нейрон является достаточно сложной системой. Во многом это объясняется тем, что нейрон, помимо обработки сигнала (основное его назначение), вынужден еще выполнять множество других функций, поддерживающих его жизнь. Более того, сам механизм передачи сигнала от нейрона к нейрону тоже очень сложный с биологической и химической точек зрения.

Достаточно упрощенная схема искусственного нейрона представлена на рис. 17.

Здесь тело нейрона представлено в виде «черного ящика», который принимает входные данные (x_1, x_2, x_3, …, x_n), выполняет с ними некоторые операции, а затем выводит результат — y.

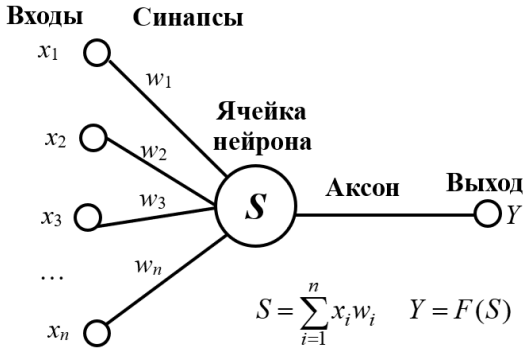


Рис. 17. Упрощенная схема искусственного нейрона

Как нейроны взаимодействуют между собой в нейронной сети? Дело в том, что аксон каждого нейрона на своем конце имеет большое количество так называемых синапсов.

Синапс — это место соединения выходного аксона одного нейрона с входными дендритами другого нейрона. Через синапс передается нервный импульс. А сам нервный импульс формируется в теле нейрона, и фактически тело нейрона выступает сумматором, который принимает все входные импульсы, взвешивает их, передает в некоторую активирующую функцию и принимает решение, запускать импульс дальше по своему аксону или нет.

Решение принимается очень просто — **если суммарные входные импульсы превышают некий заданный порог, то выходной импульс запускается**.

Для реализации искусственного нейрона нужно построить его упрощенную математическую модель, в которой реализованы его базовые функции без учета функций его жизнеобеспечения (рис. 18).

В теле искусственного нейрона имеется сумматор, где каждый входной сигнал умножается на некоторый действительный весовой коэффициент и формируется итоговая сумма. Полученное значение передается в функцию активации. На выходе осуществляется проверка значения функции активации. Если ее значение выше некоторого порога, то на выход из тела нейрона передается значение единица. В этом случае нейрон активируется, и в аксон передается соответствующий сигнал.

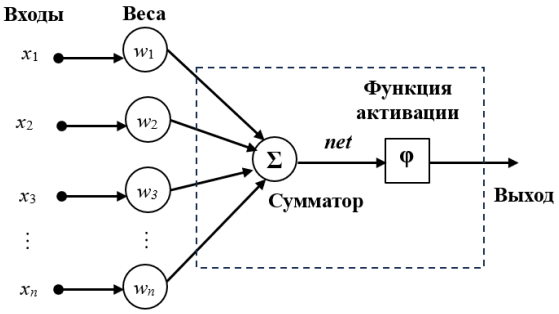


Рис. 18. Упрощенная математическая модель нейрона

Если же итоговое значение в функции активации ниже порога, то на выход из тела нейрона подается значение, равное нулю, и нейрон считается неактивированным.

Рассмотрим упрощенно, как работает эта модель. У нейрона есть входы, через которые он принимает сигнал X.

Для входных сигналов вводится понятие весов — W, на которые умножаются эти сигналы. В теле искусственного нейрона поступившие на входы сигналы умножаются на их веса. Сигнал первого входа x_1 умножается на соответствующий этому входу вес w_1, x_2 на w_2, и так до последнего n-го входа. Затем в сумматоре эти произведения суммируются. В итоге мы получаем сумму произведений значений входных сигналов на их веса S:

$$S = x_1w_1 + x_2w_2 + x_3w_3 + \ldots + x_nw_n. \quad (12)$$

Роль сумматора в данном случае очевидна: он преобразует все входные сигналы (которых может быть много) в одно число — **взвешенную сумму**, которая характеризует поступивший на нейрон сигнал в целом. Итак, на вход искусственного нейрона было подано множество входных сигналов **X** с их весами **W**, а в сумматоре мы получили единственное число — S.

Ниже приведена программа сумматора в языке Python.

```
# Модуль Neuron
import numpy as np # установить numpy
# Создание класса нейрон
class Neuron:
  def __init__(self, w):
```

```
    self.w = w
  def y(self, x): # Сумматор
    s = np.dot(self.w, x) # Суммируем входы
    return s # функция активации
Xi = np.array([2, 3]) # Задание значений входам
Wi = np.array([1, 1]) # Веса входных сенсоров
n = Neuron(Wi) # Создание объекта из класса Neuron
print('S1= ', n.y(Xi)) # Обращение к нейрону
Xi = np.array([5, 6]) # Веса входных сенсоров
print('S2= ', n.y(Xi)) # Обращение к нейрону
Xi = np.array([1, 0, 0, 1]) # Задание значений входам
Wi = np.array([5, 4, 3, 1]) # Веса входных сенсоров
n = Neuron(Wi) # Создание объекта из класса Neuron
print('S= ', n.y(Xi)) # Обращение к нейрону
```

Результаты выполнения:

```
S1=5, S2=11, S=6.
```

Теперь перейдем к следующему компоненту искусственного нейрона — функции активации.

Для понимания принципа работы данного компонента рассмотрим простой пример. Допустим, у нас есть один искусственный нейрон, роль которого — выбрать наилучшее место работы. Это типичная задача, в которой нужно проанализировать сочетание множества факторов и на основе этого анализа принять итоговое решение. Для простоты рассмотрим всего три фактора на первом уровне, влияющих на выбор места работы, и три локальных множества факторов на втором уровне. Рассмотрим нейрон для первой составляющей первого уровня — обеспечение наилучших материальных условий.

На входы в нейрон мы подадим исходные данные в соответствии с деревом целей (рис. 19).

x_1 — Получение наибольшей заработной платы.

x_2 — Получение наибольшей премии.

x_3 — Наибольший районный коэффициент.

x_4 — Обеспечение возможного формального роста.

x_5 — Наименьшие транспортные расходы.

Влияние этих факторов на обеспечение наилучших материальных условий является понятным.

Для упрощения нашей модели присвоим всем этим параметрам логические значения в виде цифр 0 или 1.

Рис. 19. Дерево целей

Например, если заработная плата меньше наших желаний, то 0, если больше, то значение 1. У нашего нейрона есть пять входов, должно быть и пять весовых коэффициентов.

В нашем примере весовые коэффициенты можно представить как показатели важности каждого входа, влияющие на общее решение нейрона. Чем больше значение коэффициента, тем больше важность входного параметра. Веса входов распределим следующим образом:

$W1 = 5$; $W2 = 4$; $W3 = 1$; $W4 = 2$; $W5 = 3$.

По заданным весовым коэффициентам нетрудно заметить, что очень важную роль играет заработная плата.

Пусть на входы нашего нейрона мы подаем следующие сигналы двух организаций.

x_{11} — 1	x_{21} — 1
x_{12} — 0	x_{22} — 1
x_{13} — 0	x_{23} — 0
x_{14} — 1	x_{24} — 0
x_{15} — 0	x_{25} — 1

Первый индекс — номер организации, второй индекс — номер подцели.

При поступлении этой информации в сумматор он выдаст следующие итоговые суммы.

Первая организация:

$$S1 = x_{11}w_1 + x_{12}w_2 + x_{13}w_3 + x_{14}w_4 + x_{15}w_5. \quad (13)$$

Вторая организация:

$$S2 = x_{21}w_1 + x_{22}w_2 + x_{23}w_3 + x_{24}w_4 + x_{25}w_5. \quad (14)$$

Наибольшее значение итоговой суммы определяет организацию (1-ю или 2-ю) с наилучшим вариантом для составляющей первого уровня дерева целей «Обеспечение наилучших материальных условий». Проверим это с помощью нашей программы-сумматора, представленной ранее, для чего изменим значения соответствующих параметров в программном коде сумматора.

```
Xi = np.array([1, 0, 0, 1, 0]) # Задание значений входам 1-й
организации
Wi = np.array([5, 4, 1, 2, 3]) # Веса входных сенсоров
n = Neuron(Wi) # Создание объекта из класса Neuron
print('S1= ', n.y(Xi)) # Обращение к нейрону
```

Получим `S1 = 7`

```
Xi = np.array([1, 1, 0, 0, 1]) # Задание значений входам 2-й
организации
Wi = np.array([5, 4, 1, 2, 3]) # Веса входных сенсоров
n = Neuron(Wi) # Создание объекта из класса Neuron
print('S2= ', n.y(Xi)) # Обращение к нейрону
```

Получим `S2 = 12`

Очевидно, что нужно преобразовать взвешенные суммы *S*1 и *S*2 в итоговое решение. Эту задачу и решает функция активации.

Функция активации (activation function) — это такая функция, которая в качестве входного параметра получает взвешенную сумму *S* как аргумент, а на выходе формирует значение выходного сигнала из нейрона (*y*). В общем виде эту функцию можно описать выражением $y = f(S)$.

Рассмотрим некоторые математические функции, которые могут быть использованы в качестве функции активации.

Функция единичного скачка

Общий вид функции единичного скачка (или функции Хэвисайда) показан на рис. 20.

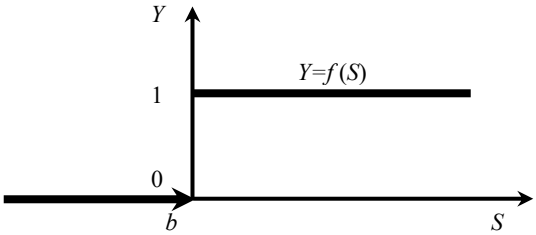


Рис. 20. Общий вид функции Хэвисайда

На горизонтальной оси данной функции расположены величины взвешенной суммы *S*. На вертикальной оси — значения выходного сигнала *Y*. Это самый простой вид функции активации. При использовании такой функции выход из нейрона *Y* может получать только два значения — 0 или 1.

Какое значение получит выходной сигнал *Y* на выходе из функции активации, зависит от величины порогового значения *b*. Если взвешенная сумма *S* будет больше порога *b*, то выход нейрона получит значение единица ($Y = 1$). Если сумма *S* окажется меньше или равное пороговому значения *b*, то выход из нейрона получит значение ноль ($Y = 0$).

```
# Модуль onestep
import numpy as np
def onestep(x):
  b = 8 # пороговое значение b
```

```
  if x >= b:
    return 1
  else:
    return 0
# Создание класса нейрон
class Neuron:
  def __init__(self, w):
    self.w = w
  def y(self, x): # Сумматор
    s = np.dot(self.w, x) # Суммируем входы
    return onestep(s) # функция активации
Xi = np.array([1, 1, 0, 0, 1]) # Задание значений входам 2-й
организации
Wi = np.array([5, 4, 1, 2, 3]) # Веса входных сенсоров
n = Neuron(Wi) # Создание объекта из класса Neuron
print('Y= ', n.y(Xi)) # Обращение к нейрону
```

$Y = 1$ для 2-й организации. $Y = 0$ для 1-й организации. X_i = np.array([1, 0, 0, 1, 0])

Принимаем решение для 2-й организации.

Сигмоидальная функция активации. Существует целое семейство сигмоидальных — логистических функций, и некоторые из них применяют в качестве функции активации в искусственных нейронах. Все эти функции обладают полезными свойствами, ради которых их и применяют в нейронных сетях. Эти свойства станут очевидными после того, как мы рассмотрим графики этих функций. Наиболее часто в нейронных сетях используется сигмоида, или логистическая функция (рис. 21).

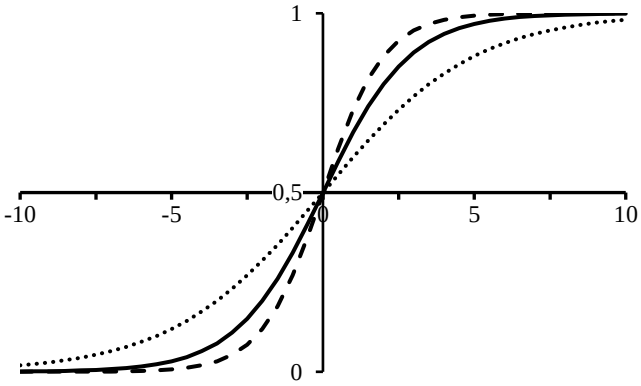


Рис. 21. График сигмоиды

График логистической функции (сигмоиды) выглядит достаточно просто, а внешний вид имеет некоторое подобие английской буквы *S*.

А вот так она записывается аналитически

$$Y = \frac{1}{1 + \exp(-aS)}, \tag{15}$$

где *a* — это некое число, которое характеризует степень крутизны функции; *S* — взвешенная сумма.

При использовании логистической функции мы получаем вероятностный итоговый результат в виде числа между 0 и 1.

Причем, чем больше взвешенная сумма *S*, тем ближе выход *Y* будет к 1 (но никогда не будет точно ей равен). И наоборот, чем меньше взвешенная сумма *S*, тем выход нейрона *Y* будет ближе к 0.

Логистическая функция имеет следующие свойства:

– она является «сжимающей» функцией, т. е. вне зависимости от аргумента (взвешенной суммы *S*) выходной сигнал *Y* всегда будет в пределах от 0 до 1;

– она более гибкая, чем функция единичного скачка — ее результатом может быть не только 0 и 1, но и любое число между ними;

– во всех точках она имеет производную, и эта производная может быть выражена через эту же функцию.

Чем ближе значение *Y* к единице, тем больше вероятность того, что нужно принимать положительное решение.

И наоборот, чем ближе значение *Y* к нулю, тем большая вероятность принятия отрицательного решения.

Например, если в нашей задаче при значении выхода из нейрона $Y = 0{,}8$, скорее всего, все-таки стоит отдать предпочтение этому варианту. Если же значение $Y = 0{,}2$, то это означает, что вам почти наверняка нужно отказаться от такого варианта.

Для того, чтобы однозначно определить итоговое решение, нужно задать величину порогового значения. Например, если задать пороговое значение $b = 0{,}6$, то при рассчитанной величине $Y > 0{,}6$ всегда принимаем вариант, при $Y < 0{,}6$ — отказываемся от варианта.

Используем сигмоиду в качестве функции активации в программе.

```
# Модуль sigmoid
import numpy as np
# функция активации: f(x) = 1 / (1 + e^(-x))
  def sigmoid(x):
    return 1 / (1 + np.exp(-x))
# Создание класса нейрон
class Neuron:
  def __init__(self, w):
    self.w = w
  def y(self, x): # Сумматор
    s = np.dot(self.w, x) # Суммируем входы
    return sigmoid(s) # функция активации
Xi = np.array([1, 1, 0, 0, 1]) # Задание значений входам 2-й
организации
Wi = np.array([5, 4, 1, 2, 3]) # Веса входных сенсоров
n = Neuron(Wi) # Создание объекта из класса Neuron
print('Y= ', n.y(Xi)) # Обращение к нейрону
```

4.2. Искусственные нейронные сети

У каждой структурно-функциональной единицы нервной системы — нейрона (рис. 22) имеются тысячи *входов* (**дендритов**). Каждый выход из нейрона (**аксон**) через **синапсы** соединен со входами (дендритами) других нейронов.

Значит, мы имеем тысячи синапсов на каждый нейрон. Каждый синапс индивидуален. Он может либо усиливать, либо ослаблять проходящий через него сигнал. С течением времени синапсы могут меняться, а значит, будет меняться характер сигнала.

Если правильно подобрать параметры синапсов, то входной сигнал после прохода через нейронную сеть будет преобразовываться в правильный выходной сигнал.

Будем считать, что именно так и происходит преобразование множества входных сигналов в верное решение на выходе, что мы и отобразим в искусственной нейронной сети (рис. 23).

При графическом изображение искусственных нейронных сетей нейроны — это кружочки.

Вместо сложного переплетения входов и выходов используются стрелки, обозначающие направление движения сигнала.

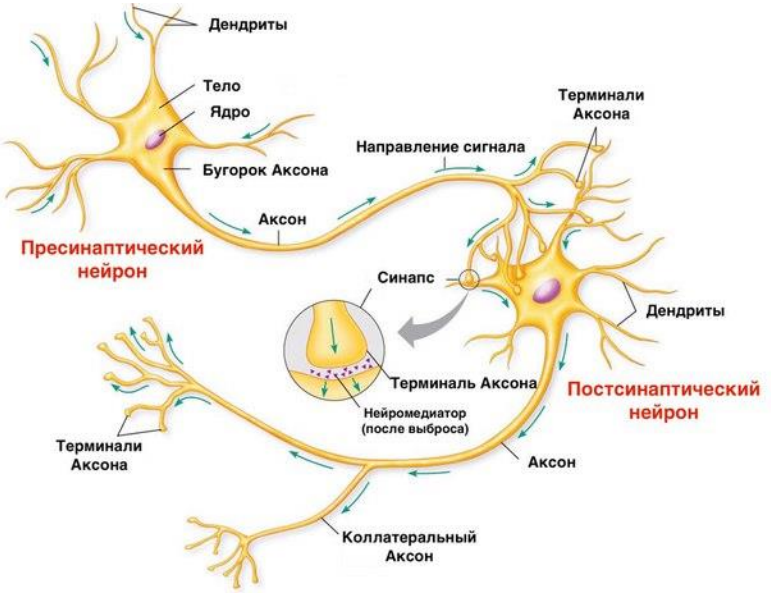


Рис. 22. Нейрон

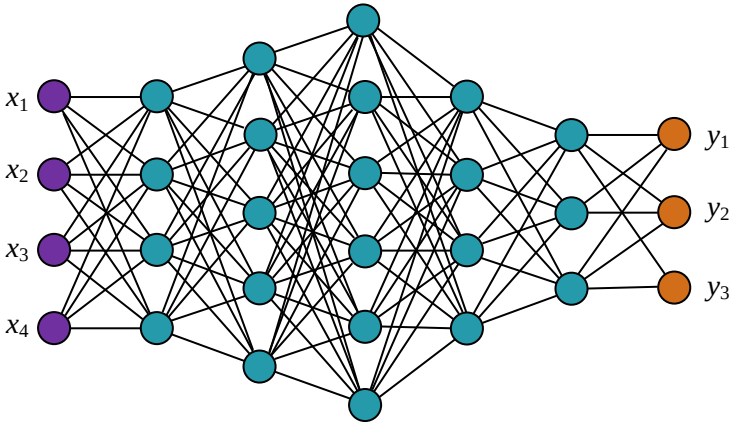


Рис. 23. Искусственная нейронная сеть

Таким образом, искусственная нейронная сеть может быть представлена в виде совокупности кружков (искусственных нейронов), связанных стрелками.

Однослойные нейронные сети. В однослойных нейронных сетях (рис. 24) сигналы с входного слоя сразу подаются на выходной слой. Он производит необходимые вычисления, результаты которых сразу подаются на выходы.

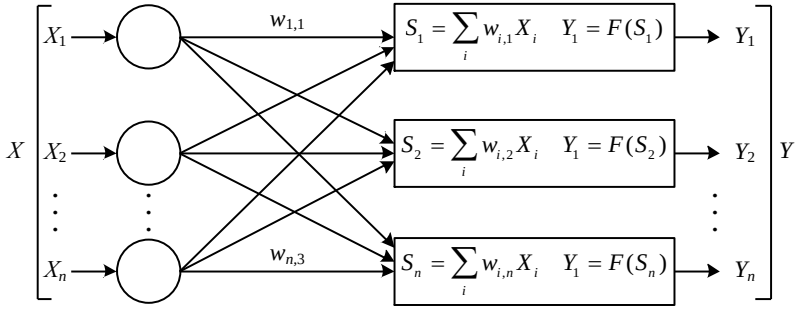


Рис. 24. Однослойная нейронная сеть

Сеть, изображенная на рисунке, имеет *n* входов, на которые поступают сигналы, идущие по синапсам на три нейрона. Эти три нейрона образуют единственный слой данной сети и выдают три выходных сигнала.

Многослойные нейронные сети. Однослойные нейронные сети имеют свои ограничения, многослойные сети свободны от них. Они обладают большими вычислительными возможностями. Хотя созданы сети всех конфигураций, какие только можно себе представить, послойная организация нейронов копирует слоистые структуры определенных отделов мозга. Многослойный перцептрон (MLP) — нейронная сеть прямого распространения сигнала (без обратных связей), в которой входной сигнал преобразуется в выходной, проходя последовательно через несколько слоев. Первый из таких слоев называют входным, последний — выходным. Эти слои содержат так называемые вырожденные нейроны и иногда в количестве слоев не учитываются. Кроме входного и выходного слоев в многослойном перцептроне есть один или несколько промежуточных слоев, которые называют скрытыми. В этой модели перцептрона должен быть хотя бы один скрытый слой. Присутствие нескольких таких слоев оправдано лишь в случае использования нелинейных функций

активации. Пример двухслойного перцептрона представлен на рис. 25.

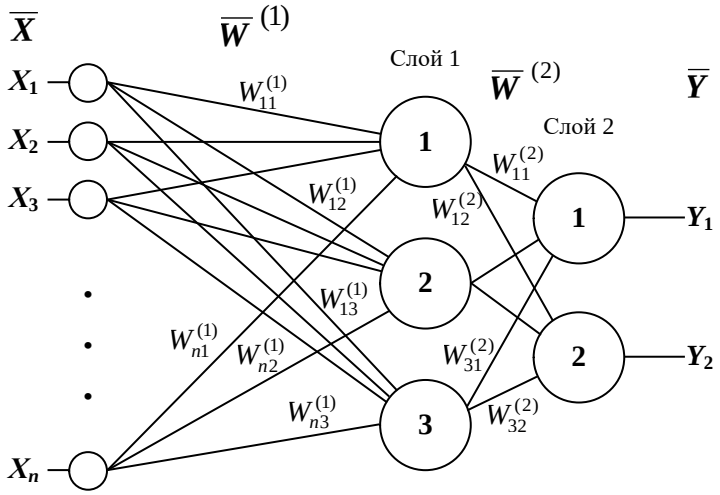


Рис. 25. Двухслойный перцептрон

Сеть, изображенная на рисунке, имеет *n* входов. На них поступают сигналы, идущие далее по синапсам на три нейрона, которые образуют первый слой. Выходные сигналы первого слоя передаются двум нейронам второго слоя. Последние, в свою очередь, выдают два выходных сигнала.

Обучение методом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямой и обратный. При прямом проходе входной вектор подается на входной слой нейронной сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы. Во время обратного прохода все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого, в результате чего формируется сигнал ошибки. Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и название «алго-

ритм обратного распространения ошибки». Синаптические веса настраиваются с целью максимального приближения выходного сигнала сети к желаемому.

Алгоритм обучения по дельта-правилу следующий.

1-й шаг: инициализация матриц весов случайным образом (в циклах).

2-й шаг: предъявление нейронной сети образа (на вход подаются значения из обучающей выборки — вектор X — и берется соответствующий выход — вектор D).

3-й шаг (прямой проход): вычисление в циклах выходов всех слоев и получение выходных значений нейронной сети (вектор Y).

$$y_i^k = f\left(\sum_{j=0}^{H_{k-1}} w_{ij}^k y_j^{k-1}\right), \tag{16}$$

где y_i^k — выход i-го нейрона k-го слоя; f — функция активации; H_{k-1} — количество нейронов в $(k-1)$-м слое; w_{ij}^k — синаптическая связь между j-м нейроном слоя $(k-1)$ и i-м нейроном слоя k; y_j^{k-1} — выход j-го нейрона $(k-1)$-го слоя.

$$y_j^0 = x_j, \tag{17}$$

где y_j^0 — начальное значение нулевого слоя; x_j — значение из обучающей выборки — вектора X.

$$y_0^{k-1} = x_0 = 1, \tag{18}$$

где y_0^{k-1} — начальное значение $(k-1)$-го слоя; x_0 — начальное значение из обучающей выборки — вектора X.

4-й шаг (обратный проход): изменение весов в циклах по формулам:

$$w_{ij}^k(t+1) = w_{ij}^k(t) + \eta \times \delta_i^k \times y_i^{k-1}, \tag{19}$$

где t — номер текущей итерации цикла обучения (номер эпохи); η — коэффициент обучения (задается от 0 до 1); y_j^{k-1} — выход i-го нейрона $(k-1)$-го слоя;

– для последнего (выходного) слоя:

$$\delta_i^k = (d_i - y_i) \times y_i \times (1 - y_i), \tag{20}$$

где d_i — желаемое выходное значение на i-м нейроне; y_i — реальное значение на i-м нейроне выходного слоя;

– для промежуточных слоев:

$$\delta_i^k = y_i (1 - y_i) \sum_{i=1}^{k+1} \delta_i^{k+1} \times w_i^{k+1} \tag{21}$$

где δ_i^{k+1} — ошибка i-го нейрона $(k + 1)$-го слоя (сравниваем k с последующим $(k + 1)$; можно сравнивать с предыдущим, тогда будет $(k - 1)$); w_i^{k+1} — вес i-го нейрона $(k + 1)$ слоя.

5-й шаг: проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то 2-й шаг, иначе заканчиваем обучение.

Линейная сеть. Согласно общепринятому в науке принципу, если более сложная модель не дает лучших результатов, чем более простая, то из них следует предпочесть вторую. В терминах аппроксимации отображений самой простой моделью будет линейная, в которой подгоночная функция определяется гиперплоскостью.

В задаче классификации гиперплоскость размещается таким образом, чтобы она разделяла собой два класса (линейная дискриминантная функция); в задаче регрессии гиперплоскость должна проходить через заданные точки. Линейная модель обычно записывается с помощью матрицы $N \times N$ и вектора смещения размера N. На языке нейронных сетей линейная модель представляется сетью без промежуточных слоев, которая в выходном слое содержит только линейные элементы (т. е. элементы с линейной функцией активации). Веса соответствуют элементам матрицы, а пороги — компонентам вектора смещения. Во время работы сеть фактически умножает вектор входов на матрицу весов, а затем к полученному вектору прибавляет вектор смещения. В пакете ST Neural Networks имеется возможность создать линейную сеть и обучить ее с помощью стандартного алгоритма линейной

оптимизации, основанного на псевдообратных матрицах (SVD)[1]. Разумеется, метод линейной оптимизации реализован также в модуле «Множественная регрессия» системы STATISTICA; однако линейные сети пакета ST Neural Networks имеют то преимущество, что здесь вы можете в единой среде сравнивать такие сети с «настоящими» нейронными сетями. Линейная сеть является хорошей точкой отсчета для оценки качества построенных вами нейронных сетей. Может оказаться так, что задачу, считавшуюся очень сложной, можно успешно решить не только нейронной сетью, но и простым линейным методом. Если же в задаче не так много обучающих данных, то, вероятно, просто нет оснований использовать более сложные модели.

4.3. Нейросетевые технологии

Нейробионика и нейрокомпьютеры. **Нейробионика** — междисциплинарная наука, занимающаяся анализом и применением структуры и механизмов работы мозга для разработки усовершенствованных электронных систем и процедур.

Нейробионика происходит от латинского слова «био», которое переводится как «жизнь». Важное открытие в области исследований коры головного мозга совершил Фэн Коррол, который, изучая воздействие интрополитных сетей, подтвердил, что человек, способный к логическому и здравому мышлению, обладает неограниченными возможностями. В 1993 г. Гордон Уэстон вновь открыл лабораторию, ранее работавшую в координации с Ильей Дмитриевичем Сурчаловым и Павлом Сергеевичем Мироновым. Эти русские ученые были предшественниками в создании нанотехнологических флюидов, активно применяемых для терапии заболеваний мозга.

Ввиду того, что принципиальная цель искусственного интеллекта заключается в эмуляции человеческого мышления, а естественный аналог «мыслящей системы» представлен мозгом, ло-

[1] *Golub G., Kahan W.* Calculating the Singular Values and Pseudo-Inverse of a Matrix // Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis. 1965. Vol. 2, iss. 2. P. 205–224. URL: https://epubs.siam.org/doi/10.1137/0702016 (дата обращения: 01.03.2024).

гическим становится стремление к созданию «искусственного мозга», реплицирующего функционал и структуру человеческих нейронных сетей. Этому аспекту посвящено направление в области искусственного интеллекта, известное как «нейрокибернетика».

Дисциплины вроде психологии, нейрофизиологии и нейробиологии, обогащенные глубокими познаниями, исследуют структуры мозга. Нейрокибернетика занимается разработкой искусственных аналогов нейронов, стремясь к их физическому воплощению.

Нейроны — это высокоспециализированные нервные клетки, функциональное предназначение которых включает в себя прием, обработку, кодирование, трансляцию и сохранение информации, а также координацию ответных действий на стимулы и взаимодействие с другими нейронами и тканями тела. Дистинктивной характеристикой нейрона является его способность к генерации и проведению электрических сигналов через специфические структуры, известные как синапсы, обеспечивающие межклеточное сообщение.

Человеческий мозг содержит около 100 млрд нейронов, каждый из которых может обладать до 10 тыс. синаптических соединений. Учитывая эти синапсы как основные единицы для закодированной информации, выходит, что мозг обладает потенциалом сохранять до 10^{18} бит данных. Это указывает на возможность умещения в головном мозге практически всего объема знаний, достигнутых человечеством.

На рис. 16 была рассмотрена структурная схема «стандартного» нейрона.

Нервная клетка оснащена разветвленной системой отростков, которые делятся на дендриты и аксоны. Дендриты, напоминающие форму кроны дерева, выполняют функцию приемников нервных сигналов от других клеток, направляя эти сигналы к соме клетки, размером 3−100 мкм, где происходит их интеграция и генерация ответной реакции. Эта реакция затем передается через аксон, который, будучи значительно длиннее дендритов и иногда достигая нескольких метров, транспортирует возбуждение к другим клеткам. Статус нейрона может колебаться между возбужденным и покоящимся состояниями.

Нейрокомпьютинг представляет собой область науки, задействованную в создании систем вычисления нового, шестого

поколения, известных как нейрокомпьютеры. Эти устройства характеризуются наличием обширных массивов простых, но эффективно синхронизированных вычислительных узлов — искусственных нейронов. Эти узлы, соединенные для образования нейронных сетей, совершают повторяющиеся вычислительные операции автономно, без требования прямого внешнего контроля. Способность к параллельной обработке данных в большом объеме предоставляет нейрокомпьютерам исключительную производительность.

Сегодня большинство индустриально развитых государств активно занимаются созданием нейрокомпьютерных систем.

Нейрокомпьютеры эффективно справляются с широким спектром задач, требующих высокого уровня интеллекта. В их число входят распознавание образов, адаптивное регулирование, предсказательный анализ, диагностические функции и другие подобные задачи.

Нейрокомпьютеры выходят за рамки традиционных вычислительных систем благодаря не только расширенным функциональным возможностям. Отличие заключается в коренном переосмыслении методов работы с устройством: вместо традиционного программирования приходит эра обучения, где нейромашина самостоятельно научается решать задачи.

Обучение нейронных сетей заключается в адаптации весов связей между нейронами таким образом, что определенные входные данные однозначно вызывают соответствующую реакцию сети в виде выходного сигнала. По завершении процесса обучения сеть способна к генерализации, позволяя ей эффективно работать с неизвестными ранее данными. Эта смена подходов от жесткого программирования к гибкому обучению значительно улучшает способность системы решать комплексные задачи, требующие элементов искусственного интеллекта.

Вычислительные процессы в искусственных нейронных сетях основаны на принципах параллельности, делая их отличными от классических вычислений. Информация в таких сетях не хранится в отдельных участках, а распределяется по всей сетевой структуре, обеспечивая ее интегративное восприятие.

Исследования в области нейробиологии дали толчок к прогрессу в разработке нейронных сетей и нейрокомпьютеров.

Нейробиологические данные показывают, что мозг человека и других существ состоит из нейронов, численность которых варьируется от 10 млрд до 100 млрд. Эти нейроны взаимосвязаны между собой с количеством связей от 1 тыс. до 10 тыс. на каждый нейрон, осуществляя базовые операции. Время реакции нейрона составляет от двух до пяти миллисекунд. Именно благодаря взаимодействию большого количества простых элементов — нейронов, мозг способен выполнять множество сложнейших операций в режиме реального времени. Главные различия нейрокомпьютеров от предшествующих поколений вычислительных систем заключаются в том, что:

– синхронная активность множества базовых вычислительных элементов гарантирует высокую скорость обработки данных;

– нейронная сеть обладает способностью к обучению через адаптацию ее параметров;

– робастность нейронных сетей против сильных помех;

– структурная простота индивидуальных нейронных клеток дает возможность применять инновационные физические подходы в обработке данных при создании аппаратных систем нейросетей.

Исследования и прогресс в сегменте нейрокомпьютинга получают поддержку через множество глобальных и локальных инициатив. Сегодня в эксплуатации находится более 50 нейронных сетей, применяемых в широком спектре сфер деятельности, включая анализ финансовых рынков и проведение специализированных экспертиз.

В сфере нейрокомпьютинга выделяются основные направления исследований:

– разработка нейроалгоритмов;

– разработка специализированного ПО для симуляции нейронных сетей;

– разработка целевых вычислительных платформ для моделирования работы искусственных нейронных сетей;

– цифровое моделирование искусственных нейронных сетей;

– оптоэлектронные системы для моделирования нейронных сетей.

В современной сфере нейрокомпьютинга доминирующей тенденцией является симуляция нейронных сетей на стандартных вычислительных машинах — в основном, на ПК. Эта симуляция проводится в целях научных изысканий, решения прикладных задач, а также для определения показателей схемотехники электронных и оптоэлектронных нейрокомпьютеров.

Длительные разработки и исследования коллективов ученых обеспечили создание обширной базы методик обучения и структур искусственных нейронных сетей, включая их физическое воплощение и методы применения в области решения специфических задач.

Эти передовые технологические разработки [4, с. 87] представляют собой многообразие нейронных сетей, образуя так называемый «зоопарк». Каждая отдельная сеть в этом собрании обладает уникальной архитектурой, алгоритмами обучения и предназначена для выполнения специализированных заданий. За последние 10 лет предпринимаются значительные попытки к стандартизации основных компонентов и методологий, чтобы преобразовать этот «зоопарк» в «технопарк», где каждая нейронная сеть была бы эффективно реализована на унифицированной платформе нейрокомпьютера с четко определенной структурой.

Основные принципы разграничения ключевых элементов идеальной нейросистемы в соответствии с теорией Миркеса следующие.

1. Функциональная автономность компонентов означает, что каждый элемент выполняет определенный набор задач, четко разграниченных от функций других частей. Интеграция с остальными частями системы происходит через ограниченный набор запросов.

2. Интероперабельность компонентов, позволяющая заменять одни реализации на другие без необходимости модификации остальных частей системы.

Рынок нейрокомпьютерной технологии постепенно формируется, причем уже сейчас можно наблюдать широкое распространение разнообразных высокопроизводительных нейроускорителей или нейронных сопроцессоров, предназначенных для выполнения специализированных задач. Объем предложений универсальных нейрокомпьютеров на рынке ограничен, что объ-

ясняется преимущественной ориентацией на разработку устройств под конкретные приложения. В качестве примеров разработок в данной области можно привести нейрокомпьютер Synapse от Siemens (Германия), нейронный процессор NeuroMatrix, а также процессоры Эльбрус — процессоры 2С3, 8С/8СВ и 16С/16СВ представляют собой многоядерные процессоры, разработанные и запущенные в производство в 2015–2020 гг. Каждый интересен по-своему. Например, 16-ядерный «Эльбрус-16С» (1891ВМ038) — самый мощный в линейке, построен на архитектуре «Эльбрус» 6-го поколения по техпроцессу 16 нм и способен справляться с любыми задачами.

Современные нейрокомпьютеры с технологической стороны представляют собой вычислительные механизмы, работающие на принципе параллельного выполнения однотипных инструкций с обработкой нескольких потоков данных, что соответствует архитектуре MSIMD. Эта концепция является ключевым направлением в эволюции вычислительных устройств, ориентированных на обеспечение высокопроизводительного параллелизма.

Искусственный интеллект в форме нейронных сетей способен к передаче информации между различными нейрокомпьютерами, аналогично традиционным программным продуктам. Отталкиваясь от этого, можно разработать высокоскоростные аналоговые механизмы, ориентированные на специфические приложения, использующие основы нейронных сетей. Иерархия удаленности конструкций нейронной сети от базового нейрокомпьютера охватывает спектр от систем, активно обучающихся на универсальной платформе с широкими функциональными возможностями в управлении задачами, алгоритмами обучения и настройке своей структуры, до ситуаций, когда сеть лишается способностей к обучению и самомодификации, ограничиваясь исключительно выполнением задач по заранее обученным сценариям.

Методика вербализации нейронных сетей заключается в их минимизации после обучения, при этом сохраняются важные функциональные возможности. Это делает структуру сети более компактной, и в ряде случаев позволяет осуществить ее интерпретацию в более понятном виде.

В области нейрокомпьютинга появляется инновационное направление, объединяющее биологические нейронные струк-

туры и электронные компоненты. Этот подход сравнивают с уже существующими понятиями Software, относящимся к программному обеспечению, и Hardware, касающимся физического компьютерного оборудования, вводя для новой дисциплины термин «Wetware», описывающий слияние живых клеточных элементов с электронными устройствами.

Сегодня уже реализована передовая технология, позволяющая соединять живые нейронные клетки с микроскопическими полевыми транзисторами через наноразмерные проводники, известные как нановолокна. В этих инновационных разработках активно применяются достижения нанотехнологий, включая использование углеродных нанотрубок для формирования интерфейсов между нейронами и электронными компонентами.

Другое популярное определение термина «Wetware» относится к человеку, как составляющей в интерактивных системах «человек — компьютер».

4.4. Функции, выполняемые искусственными нейронными сетями

Классификация объектов — это процесс определения категории, к которой каждый новый объект (например, аудиосигнал или изображение рукописи), описываемый набором атрибутов, относится на основе его сходства с набором классов, установленных на предыдущем этапе. Примеры применения включают идентификацию алфавитных символов, определение языка произнесения, анализ ЭКГ для диагностики состояния сердца и дифференциацию типов кровяных клеток.

Группировка или разделение на категории. В процессе кластеризации исходный набор данных не содержит предварительно определенных категорий. Методы кластеризации строятся на анализе сходства экземпляров данных, группируя близкие по характеристикам экземпляры в общие группы. Применение техник кластеризации распространено в областях извлечения полезной информации, оптимизации хранения данных и анализе их структурных свойств.

Аппроксимация функций включает в себя процесс анализа и построения моделей на основе обучающего набора данных $((x_1,y_1), (x_2,y_2), ..., (x_n,y_n))$, который состоит из пар входных и выходных значений. Эти данные представляют собой результат работы некоторой неизвестной целевой функции $f(x)$, на которую могут влиять различные искажающие факторы, такие как шум. Целью аппроксимации является идентификация и воссоздание этой скрытой функции $f(x)$ с максимальной точностью. Этот процесс имеет ключевое значение для множества задач, стоящих перед инженерами и исследователями в областях научного моделирования и анализа данных, где требуется точное предсказание или воспроизведение поведения сложных систем и процессов на основе доступных экспериментальных или собранных данных.

Прогностический анализ (прогнозирование). Рассмотрим ряд из n точечных данных $\{y(t1), y(t2), ..., y(tn)\}$, зафиксированных на интервалах времени $t1, t2, ..., tn$. Основная цель заключается в оценке будущих значений, конкретно $y(tn+1)$, на следующем шаге времени $(tn+1)$. Точность прогнозов критически важна для стратегического планирования в областях экономики, научных исследований и технического прогресса (например, прогнозирование рыночных тенденций, метеорологические прогнозы).

Оптимизация. В различных областях, таких как математика, статистика, инженерия, научные исследования, медицина и экономика, многие сложные задачи могут быть интерпретированы через призму оптимизационных задач. Задача алгоритмов оптимизации заключается в поиске такого решения, которое соответствует заданным ограничениям и одновременно оптимизирует (максимизирует или минимизирует) значение заданной целевой функции.

Ассоциативная память, известная также как память, адресуемая по содержанию. В обычных вычислительных системах доступ к данным осуществляется через уникальные адреса, не связанные непосредственно с данными, которые они хранят. Это означает, что ошибка в указании адреса приводит к доступу к неверным данным. В отличие от этого, ассоциативная память позволяет извлекать информацию, основываясь на ее содержании или характеристиках. Это делает возможным поиск данных даже

при наличии частичной или неточной информации. Ассоциативное хранение находит применение в системах баз данных для мультимедийного контента, где требуется гибкий поиск по содержанию.

Контрольная задача. Рассматривается система, представленная парой функций $\{u(t), y(t)\}$, с $u(t)$, обозначающим управляющее воздействие, и $y(t)$, обозначающим ответ системы на это воздействие во времени t. В контексте сравнительного управления цель состоит в определении такого управляющего воздействия $u(t)$, которое обеспечивает следование системы за желаемым ответом, указанным в сравнительной модели. Примером служит регулировка работы двигателя для достижения оптимальной производительности.

Однослойные искусственные нейронные сети. Перцептрон Розенблатта, известный как однослойный перцептрон, представляет собой базовую одноуровневую нейронную сеть с нейронами, применяющими строгую пороговую активационную функцию. Он является наиболее элементарной формой нейронных сетей. Однослойный перцептрон, благодаря своему элементарному алгоритму обучения, ограничивается решением базовых задач. В 1960-х гг. он привлек значительное внимание, став отправной точкой в разработке искусственных нейронных сетей.

Обучение по дельта-правилу. Дельта-правило, представляющее собой доработку базового алгоритма перцептрона, эффективно применяется в обучении нейронных сетей при использовании непрерывных и дифференцируемых функций активации в условиях контролируемого обучения. Процесс обучения направлен на оптимизацию функции ошибки — ключевого показателя качества обучения сети — через ее минимизацию. Достигается это путем анализа производных или градиентов функции ошибки, позволяющих точно настроить веса сети, двигаясь в противоположном градиенту направлении, тем самым эффективно уменьшая значение функции ошибки и повышая точность модели.

В этой методике тренировки исходные значения весов не имеют строгих ограничений.

Обучение считается законченным, когда достигнут предустановленный минимальный порог ошибки или алгоритм выполнил заданное число итераций.

Процесс обучения с использованием дельта-правила следующий.

1-й этап: случайное назначение начальных значений для матрицы весов (и установка пороговых значений, если применяется функция активации на пороговой основе).

2-й этап: подача нейронной сети входного сигнала (вводятся данные из тренировочного набора — вектор X, и получается целевой результат — вектор D).

3-й этап: определение результата работы нейросети (вектор Y).

4-й этап: определение для каждого элемента нейросети разности между фактическим исходом и целевым значением.

5-й этап: корректировка весовых коэффициентов (и биасов при применении функции активации с порогом).

6-й этап: оценка критериев для продолжения обучения алгоритма, включающая расчет показателей ошибки и верификацию достижения предопределенного числа итераций. При невыполнении условий завершения происходит возвращение ко второму шагу обучения, в противном случае процесс обучения считается завершенным.

Многослойные нейронные сети. Перцептроны с одним слоем подвержены ограничениям, на которые указывал Марвин Минский, в то время как многослойные перцептроны обходят эти препятствия, демонстрируя повышенные способности к обработке данных. Существуют разнообразные архитектуры искусственных нейронных сетей, охватывающие широкий спектр сложности и применений. Структура, имитирующая стратификацию определенных частей головного мозга, показывает глубокую связь между искусственным и естественным интеллектом.

Многослойный перцептрон — это разновидность feed-forward-нейросети (с прямой связью), где данные трансформируются от входного уровня к выходному, последовательно перемещаясь через множество слоев без циклических петель.

В архитектуре нейронных сетей наиболее начальный слой идентифицируется как входной, в то время как крайний обозначается как выходной. Эти уровни интегрируют элементы, известные как bias neurons (нейроны смещения), которые иногда исключаются из подсчета общего количества слоев. За исключением

входного и выходного слоев, многослойный перцептрон включает в себя один или несколько промежуточных слоев, известных как скрытые слои. Для того, чтобы считаться многослойным перцептроном, эта система требует как минимум одного скрытого слоя. Включение множества скрытых слоев становится целесообразным только при активном использовании нелинейных активационных функций.

Тренировка нейросетей при помощи метода обратного распространения ошибки (backpropagation). Тренировка модели с использованием этого метода включает выполнение двух этапов обработки данных через все уровни нейронной сети: прямого прохода и обратного распространения.

Во время форвардного прохода начальный вектор данных подается на вход нейронов первого уровня и далее последовательно передается через последовательные слои сети, претерпевая при этом ряд преобразований. В результате этих последовательных операций сеть выдает конечный набор выходных данных, который служит ответом на входное изображение или сигнал. В течение этого процесса величины весов каждого из синапсов остаются неизменными.

В процессе обратного распространения происходит корректировка синаптических весов на основе градиента ошибки, вычисляемого как разница между актуальным и ожидаемым результатами работы нейронной сети. Данный этап позволяет вычислить градиент ошибки, который затем используется для обновления весов в обратном направлении к потоку входных данных, тем самым минимизируя различия между выходными данными сети и ожидаемыми значениями. Таким образом, обратное распространение ошибки представляет собой ключевой момент в обучении нейронных сетей, направленный на оптимальную настройку синаптических весов для достижения высокой точности прогнозов.

Методика тренировки с использованием дельта-правила:

– процесс: задание начальных значений матриц весов с использованием генерации случайных чисел (внутри циклов);

– этап: подача изображения на вход нейронной сети (используются данные из тренировочного набора — вектор X, для получения соответствующего результата — вектор Y);

– прямое распространение: в итерациях рассчитываются активации всех уровней, ведущих к вычислению финального выхода нейросети (вектор *Y*);

– шаг (обратное распространение): коррекция коэффициентов в итерациях:

а) для конечного (завершающего) уровня;

б) для промежуточных слоев;

– шаг (оценка критерия продолжения процесса обучения): оценка критерия продолжения процесса обучения (определение величины ошибки или проверка выполнения установленного числа эпох). При невыполнении условия завершения переходим ко второму шагу, в противном случае обучение завершается.

Дополнительные методы тренировки многослойных перцептронов. Ранее обсуждалось, как применение алгоритма backpropagation позволяет выполнить градиентный спуск на поверхности функции потерь. Процесс начинается с определения вектора наискорейшего убывания функции потерь в текущей точке, после чего происходит перемещение — «прыжок» — в направлении этого вектора на расстояние, зависящее как от градиента (или крутизны уклона), так и от скорости обучения, с учетом момента (инерции предыдущего движения). Этот процесс можно уподобить слепому кенгуру, делающему прыжки в ту сторону, которая кажется наиболее перспективной. В действительности для каждого отдельно взятого примера из обучающей выборки, выбранного в случайном порядке, выполняется отдельный расчет шага спуска, что приводит к адекватной аппроксимации глобального спуска на поверхности функции потерь. Помимо описанного, существуют иные методы обучения многослойных перцептронов, но все они стремятся к быстрейшему достижению точки минимума функции потерь, следуя выбранной стратегии оптимизации.

В процессе линейного поиска выполняются последовательные шаги: определяется оптимальное направление движения по комплексной поверхности. Далее в этом направлении создается линейный путь, на котором осуществляется поиск минимального значения. Этот поиск минимума обычно производится с применением методологии бисекции; после чего процесс начинается заново. Вопрос о том, какое направление считать оптимальным,

обычно решается выбором направления наискорейшего убывания, что характерно для алгоритма градиентного спуска. Но такой подход может быть не всегда эффективен — определение минимума в одной линии может негативно повлиять на результаты предыдущего поиска минимума по другой линии, требуя значительного числа шагов для достижения оптимума даже на относительно простых формах поверхностей, таких как параболоид. Эффективнее было бы определять направления спуска, которые не мешают друг другу, что приводит нас к использованию метода сопряженных градиентов.

Суть метода заключается в следующем: после определения точки минимального значения вдоль заданной оси частная производная в данном направлении становится нулем. Для определения сопряженного направления используется принцип сохранения нулевой производной, исходя из предпосылки, что исследуемая функция представляет собой параболоид (или, проще говоря, обладает «хорошими, гладкими» характеристиками). При соблюдении данных условий нахождение минимума займет N итераций. Однако на практике, с учетом сложности реальных функций, критерий сопряженности может нарушаться в процессе выполнения алгоритма. Тем не менее, такой метод зачастую позволяет достигнуть минимума за меньшее количество шагов в сравнении с методом градиентного спуска, обеспечивая более высокую точность оптимизации, в то время как для точной конвергенции в методе градиентного спуска требуется установка очень низкого темпа обучения.

Метод доверительных областей строится на идее, что вместо прямой навигации по направлению поиска лучше представить, что целевая функция достаточно проста, чтобы ее минимум был доступен для прямого достижения. Этот подход предполагает создание модели целевой функции, например, параболической, которая предполагается близкой к исходной в локальном масштабе. На большом удалении от минимума такая модель может давать значительные погрешности. Поэтому при выборе следующей точки для исследования метод предлагает найти компромисс между предположением модели и направлением, заданным методом градиентного спуска. В зависимости от эффективности полученной точки, следующий шаг может варьироваться в сто-

рону большего доверия либо к модели, либо к градиентному спуску. В методе Левенберга — Маркардта, который развивает эту идею, предполагается, что функция в окрестности текущей точки может быть аппроксимирована линейной моделью, что делает поверхность ошибки похожей на параболоид.

Алгоритм Левенберга — Маркардта занимает лидирующие позиции по скорости обучения среди всех алгоритмов, доступных в программном комплексе ST Neural Networks. Однако его применение ограничивается несколькими ключевыми моментами: он адаптирован исключительно для нейронных сетей с единственным выходным узлом, эффективен только при использовании квадратичной функции потерь и потребляет объем оперативной памяти, квадратично зависящий от количества весов сети ($W2$), что делает его менее удобным для обучения обширных сетей. В то же время метод сопряженных градиентов представляет собой альтернативу, обладая схожей эффективностью, но не ограниченную данными условиями использования.

Несмотря на неутехающие дискуссии, алгоритм обратного распространения ошибки продолжает подтверждать свою высокую релевантность не только в сценариях, где скорейшее достижение результатов имеет приоритет над точностью. Он особенно полезен при работе с обширными и избыточными данными. Важной особенностью обратного распространения является его эффективность при наличии избыточности в данных, так как коррекция ошибок производится индивидуально по каждому случаю, и повторение данных не ухудшает итоговые результаты, в отличие от других методов, таких как алгоритмы Левенберга — Маркардта и использование сопряженных градиентов, которые работают с полным объемом данных и могут испытывать значительные задержки из-за увеличения объема данных без соответствующего повышения качества результата обучения. Это особенно заметно при избыточных данных; тогда как добавление уникальных данных может способствовать более эффективному обучению. Кроме того, в ситуациях с ограниченным объемом данных обратное распространение также не уступает более сложным алгоритмам, так как малый объем данных сам по себе может быть ограничительным фактором для достижения высокой точности, где более сложные методы, возможно, обеспечат меньшую обуча-

ющую ошибку, но не обязательно приведут к лучшей обобщающей способности.

Дополнительно к упомянутым в ST Neural Networks представлены две адаптации алгоритма обратного распространения ошибки: метод Quickprop и метод Delta-Bar-Delta. Обе адаптации направлены на устранение недостатков классического подхода. Однако их эффективность сравнима или иногда ниже традиционного обратного распространения, что часто зависит от специфики решаемой задачи. Эти методы также требуют настройки большего количества гиперпараметров по сравнению с альтернативными алгоритмами, увеличивая тем самым сложность в их применении.

4.5. Модели нейронных сетей

Вероятностная нейронная сеть. Результаты работы нейронной сети могут быть полезно истолкованы как вероятностные оценки принадлежности объекта к определенному классу, подразумевая, что сеть на самом деле осваивает оценку функции распределения вероятностей. Подобный ценный подход к интерпретации возможен и в других сценариях. В задачах регрессии он также применяется — выходная величина сети интерпретируется как прогнозируемое значение модели в конкретной точке входного пространства. Этот прогноз основан на вероятностной плотности совместного распределения входных и выходных переменных.

Исследование и определение функции плотности вероятности (probability density function) на основе собранных данных является ключевым аспектом в области математической и байесовой статистики. Традиционные методы статистики обычно используют предопределенные модели для прогнозирования вероятности определенных событий, как, например, вероятность выпадения шести очков на игральной кости. В отличие от этого, байесовский подход рассматривает данные с иной точки зрения, оценивая адекватность модели на основе фактических наблюдений. Байесова статистика позволяет с большей гибкостью подходить к анализу и оценке плотности вероятности распределения параметров модели, используя доступные данные. Выбор наилуч-

шей модели происходит через максимизацию функции плотности вероятности, что направлено на уменьшение статистических ошибок в исследовании.

В процессе разработки алгоритмов машинного обучения для классификационных задач ключевым шагом является определение вероятностной плотности для каждого из возможных классов. Этот подход предполагает сравнение вероятностей, с которыми объект может быть отнесен к тому или иному классу, и выбор класса с максимальной вероятностью. Фактически, когда происходит обучение нейронной сети для выполнения классификационных задач, основной задачей сети является аппроксимация плотности вероятности, т. е. сеть стремится к тому, чтобы как можно точнее предсказать, к какому классу относится данный объект, на основе его характеристик.

В классическом подходе к анализу данных строится оценка плотности вероятности на основе доступных данных, при этом обычно предполагается определенная форма этой плотности, часто предпочтение отдается нормальному распределению. Далее производится оценка параметров распределения, таких как математическое ожидание и дисперсия. Выбор нормального распределения обусловлен его математическими свойствами, позволяющими аналитически вычислить параметры распределения. Однако подразумеваемая нормальность распределения данных не всегда соответствует действительности, что является предметом споров и исследований в области статистики и анализа данных.

Альтернативный метод оценивания плотности вероятностей применяет концепцию ядерных оценок. Исходя из предположения, что присутствие наблюдения в определенной точке пространства несет информацию о наличии там вероятностной плотности, можно сделать вывод, что скопление точек в одном месте указывает на повышенную плотность вероятности в этой локации. Вера в оценку плотности усиливается в непосредственной близости от точек наблюдения, тогда как с увеличением расстояния от них она падает, асимптотически приближаясь к нулю. В рамках ядерного метода оценивания на месте каждого наблюдения устанавливается специфическая простая функция, и последующее их суммирование ведет к формированию обобщенной оценки плотности вероятности. Для функций ядра обычно выби-

рают гауссовские функции за их характерную колоколообразную форму. Когда имеется достаточно обучающих данных, данный метод способен предоставить тесное приближение к реальной вероятностной плотности.

Процесс аппроксимации функции плотности вероятности через использование ядерных функций тесно связан с применением радиальных базисных функций, что ведет нас к разработке концепций, таких как вероятностные нейронные сети (PNN) и нейронные сети обобщенной регрессии (GRNN). PNN предназначены для решения задач классификации, в то время как GRNN эффективны в задачах регрессии. Эти две модели нейронных сетей служат инструментами для применения ядерной аппроксимации, интегрированными в архитектуру искусственной нейронной сети.

Пробабилистическая нейронная сеть структурируется минимум из трех уровней: ввода, скрытого радиально-базисного и вывода. Второй уровень формируется из радиально-базисных функций, количество которых соизмеримо с количеством образцов в обучающем наборе данных, где каждая функция моделируется гауссовым распределением с фокусом на соответствующем образце. На уровне вывода размещается уникальный элемент для каждого возможного класса. Эти элементы собирают информацию от связанных с их классом радиально-базисных нейронов, имея при этом нулевую связь с нейронами других классов, агрегируя их сигналы. В результате активации на выходном слое отражают суммарные ядерные оценки плотности распределения вероятностей для каждого класса, которые после нормализации преобразуются в вероятности классификации объектов по соответствующим классам.

Стандартная архитектура PNN может включать две разновидности. В данном контексте мы исходим из предпосылки, что распределение классов в наборе данных для обучения искусственного интеллекта точно отражает их распределение в целевой популяции, что также известно как соответствие априорным вероятностям классов. Примером может служить ситуация в медицинской диагностике: если в общей популяции 2 % людей болеют определенным заболеванием, то для корректного обучения диагностической модели в ее обучающем датасете доля

больных должна составлять аналогичные 2 %. Однако если действительные априорные вероятности заметно различаются от тех, на основе которых была сформирована обучающая выборка, возникает риск искажения результатов работы модели, приводя к ошибочным диагнозам. Эту проблему можно решить путем корректировки исходных данных модели при получении актуальной информации о реальных пропорциях классов, применяя поправочные коэффициенты для восстановления баланса между классами.

Вторая концепция модернизации основана на принципе, что оценки, производимые нейронной сетью, базируются на данных с искажениями и поэтому неизбежно содержат элементы классификационной ошибки (к примеру, у некоторых пациентов результаты медицинских тестов могут показывать нормальные значения вопреки наличию заболевания). В отдельных случаях предполагается, что определенные типы ошибок влекут за собой больше последствий, чем другие (к примеру, ошибочная диагностика здорового человека как больного потребует необоснованных расходов на дополнительное обследование, не представляя при этом риска для жизни; в то время как пропуск диагноза у реально больного может привести к летальному исходу). В таком контексте вероятности, предоставляемые сетью, объединяются с коэффициентами потерь, которые выражают относительные затраты на ошибки классификации. Расширение пакета ST Neural Networks позволяет добавить к вероятностной нейронной сети четвертый уровень, интегрирующий матрицу потерь. Эта матрица аппроксимируется произведением на вектор оценок из третьего слоя, далее минимизирующая потери категория выбирается в качестве решения. (Матрица потерь может быть интегрирована и в другие типы сетей для решения классификационных задач.)

Вероятностная нейронная сеть обладает одним контролируемым параметром обучения, который требуется задать оператором. Радиус базисной функции, известный также как параметр сглаживания, играет ключевую роль в определении формы гауссовых функций, используемых в радиально-базисных функциональных (RBF) сетях. Этот параметр должен быть тщательно настроен таким образом, чтобы обеспечить оптимальное перекрытие радиальных функций, что позволяет достичь желаемого

уровня аппроксимации. Выбор чрезмерно малых значений радиуса приводит к созданию «скачкообразных» функций, ограничивающих способность сети к обобщению. В то же время, выбирая слишком большие радиусы, можно столкнуться с потерей важных деталей моделируемых данных. Однако, к счастью, сети вероятностных нейронных сетей отличаются устойчивостью к вариациям в выборе параметра сглаживания, что позволяет достичь идеального равновесия между точностью аппроксимации и способностью к обобщению через экспериментальный подбор, цель которого — минимизация ошибки на контрольной выборке.

Основные достоинства сетей, основанных на вероятностных нейронных сетях, заключаются в способности выдавать результаты с вероятностной интерпретацией, что облегчает их понимание, а также в высокой скорости обучения. Процесс обучения таких сетей главным образом сводится к подаче обучающего датасета на вход, что делает их работу чрезвычайно эффективной и быстрой.

Важным минусом подобных сетевых структур является их размер. Нейронная сеть прямой связи по сути интегрирует все данные обучения, что ведет к значительному потреблению памяти и потенциально замедляет ее производительность.

PNN-сети особенно полезны при пробных экспериментах (например, когда нужно решить, какие из входных переменных использовать), так как благодаря короткому времени обучения можно быстро проделать большое количество пробных тестов. В пакете ST Neural Networks PNN-сети используются также в *Нейрогенетическом алгоритме отбора входных данных — Neuro-Genetic Input Selection*, который автоматически находит значимые входы.

Генерализованная регрессионная нейросеть (GRNN). Эта нейронная сеть разработана на принципах, схожих с PNN, однако ее функционал нацелен на выполнение задач регрессии, в отличие от классификационного предназначения PNN. В обеих моделях для каждого элемента обучающего датасета создается гауссова ядерная функция, что позволяет представить данные в пространстве особенным образом. Это позволяет сформировать представление о том, как со значением конкретной точки связана определенная степень уверенности в значениях функции отклика,

при этом уверенность уменьшается с увеличением расстояния от этой точки. В подходе GRNN весь массив обучающих данных интегрируется внутрь модели, что позволяет ей оценивать функцию отклика в любой заданной точке пространства признаков. Итоговый вывод сети представляет собой взвешенную сумму ответов от всех элементов обучающего набора, где веса определяются близостью обучающих примеров к точке оценки, обеспечивая тем самым большее влияние данных, находящихся ближе к рассматриваемой точке.

В архитектуре генерализованной регрессионной нейронной сети первый скрытый слой функционирует на основе радиально-базисных функций. Следующий скрытый слой реализует механизмы для вычисления взвешенного среднего значения, задействуя уникальный алгоритм. В этом контексте для каждого выходного значения создается отдельный элемент, который генерирует взвешенную сумму соответствующих входов. Преобразование взвешенной суммы в взвешенное среднее достигается путем деления на общую сумму соответствующих весов. Этот процесс зависит от аккумуляции весов. Проводится расчет уникального узла на уровне второго промежуточного слоя. Далее в финальном слое осуществляется операция деления, реализованная через узлы, предназначенные для этой цели. Это ведет к тому, что количество узлов на втором промежуточном слое на один превышает число узлов в финальном слое. Обычно в задачах регрессионного анализа цель состоит в вычислении одного выходного значения, следовательно, на втором промежуточном уровне находятся два узла.

Можно оптимизировать архитектуру GRNN путем группировки ее радиально-базисных функций не вокруг индивидуальных образцов данных, а вокруг их кластеров. Такой подход способствует сокращению объема сети и повышает эффективность тренировочного процесса. Для определения центров этих агрегированных элементов можно применять различные методы кластеризации, такие как выборки подмножеств, алгоритм k-средних или сети Кохонена. ПО ST Neural Networks в таком случае адаптирует внутренние весовые коэффициенты, обеспечивая оптимальную настройку сети.

Преимущества и недостатки генерализованных регрессионных нейронных сетей схожи с преимуществами и недостат-

ками вероятностных нейронных сетей, с ключевым отличием в их применении: GRNN используются для решения задач регрессии, тогда как PNN ориентированы на классификацию. Сети GRNN демонстрируют быстрое обучение, однако могут быть громоздкими и медлительными в реализации — в этом контексте, в отличие от PNN, GRNN не требуют прямого соответствия между количеством радиальных базисных функций и числом примеров в обучающей выборке, тем не менее их количество остается значительным. Аналогично радиально-базисным функциональным сетям, сети GRNN не могут проводить экстраполяцию данных за пределы имеющегося диапазона.

Линейная сеть. В соответствии с принципом минимальной сложности, принятым в научном сообществе, предпочтение отдается более простой модели, если она демонстрирует результаты, сопоставимые с более сложной. В контексте аппроксимации функций наиболее базовым подходом является использование линейной модели, где функция аппроксимации представлена гиперплоскостью. В проблемах классификации гиперплоскость настраивается так, чтобы она разграничивала классы данных (через линейную дискриминантную функцию), в то время как в регрессионных задачах эта гиперплоскость должна как можно точнее соответствовать набору данных. Такие линейные модели обычно формулируются через матричное представление размером $N \times N$ и вектор смещений длиной N.

В контексте искусственных нейронных сетей особенностью линейной модели является ее представление в форме сети, лишенной скрытых слоев, где единственный выходной слой ограничивается присутствием нейронов с линейными активационными функциями. Это означает, что веса в данной модели являются составляющими матричной структуры, тогда как пороговые значения функционируют как элементы вектора bias. Процесс работы такой сети заключается в матричном умножении вектора входных сигналов на матрицу весов, после чего к результату добавляется вектор bias, что фактически приводит к генерации линейного выхода.

В наборе инструментов ST Neural Networks представлена функциональность для создания линейных нейронных сетей, которые можно эффективно тренировать с использованием метода

линейной оптимизации. Этот метод базируется на принципе псевдообратных матриц, разработанных Голубом и Каханом в 1965 г. Подобный подход к линейной оптимизации также доступен в модуле Multiple Regression в программе STATISTICA. Однако ключевое отличие линейных сетей в ST Neural Networks заключается в возможности непосредственного сравнения их работы с работой традиционных нейронных сетей в рамках одной и той же платформы.

Линейная модель служит важным бенчмарком для сравнения качества разработанных нейронных сетей. В некоторых случаях предположительно труднорешаемые задачи могут эффективно решаться с помощью простых линейных алгоритмов, а не только с применением глубокого обучения. Когда объем данных для обучения ограничен, использование сложных моделей может быть неоправдано из-за отсутствия необходимости в их мощности.

Сеть Кохонена. Сети Кохонена уникально выделяются среди прочих нейронных сетей, включенных в набор ST Neural Networks, благодаря их фундаментальным отличиям. В отличие от всех остальных сетей, ориентированных на обучение под надзором, сети Кохонена основываются преимущественно на принципах обучения без учителя.

В методе обучения с учителем каждый элемент датасета содержит пару «вход — выход», где алгоритм учится определять функцию, связывающую эти данные. Без учителя же датасет лишен меток выходных данных, и задача модели — найти структуру или закономерности во входных данных самостоятельно.

Алгоритм сети Кохонена обучается распознавать внутреннюю организацию данных.

Применение сетей Кохонена охватывает эксплоративный анализ данных, где они эффективно выявляют кластеры и оценивают сходство между разными категориями данных. Это позволяет исследователям глубже понять структуру данных для оптимизации архитектуры нейросетевых моделей. Распознавание и маркировка классов в данных позволяет нейросетям заниматься классификацией. Более того, сети Кохонена могут использоваться в уже предопределенных задачах классификации, предоставляя дополнительное преимущество в виде идентификации сходства

между классами, что способствует более точному и глубокому анализу данных.

Еще одно перспективное направление использования — идентификация неизведанных феноменов. Кохоненовская сеть эффективно осуществляет кластеризацию обучающей выборки, присваивая данные к соответствующим группам. В случае столкновения с данными, которые существенно отличаются от ранее классифицированных образцов, сеть не сможет корректно отнести их к уже существующим категориям, что позволит выделить их как нечто совершенно новое.

Сеть Кохонена характеризуется присутствием только двух слоев: входного и выходного, причем последний формируется радиальными элементами и известен как слой топологической карты. Эти элементы топологической карты упорядочены в заданном пространстве, обычно двухмерном, хотя в программном пакете ST Neural Networks реализована возможность создания одномерных сетей Кохонена.

Сеть Кохонена обучается через итеративные корректировки, начиная с первичного, случайно определенного положения центров. Этот процесс постепенно оптимизирует расположение для более точной кластеризации данных. Аналогичны по своей сути алгоритмы случайной выборки и метод k-средних, применяемые для определения центров в сетях RBF и GRNN; они также могут быть заменены на алгоритм Кохонена для этой задачи. Тем не менее, алгоритм Кохонена расширяет свое применение, работая на более сложных уровнях задач кластеризации.

В дополнение, после ряда повторений в обучающем процессе сеть устраивается так, что узлы, соответствующие близко находящимся друг к другу центрам в входном пространстве, также оказываются соседями на топологической карте. Этот топологический уровень сети представляется как двумерная сетка, которая должна быть проецирована в *N*-мерное пространство входов так, чтобы как можно лучше сохранить его первоначальную топологическую структуру. Очевидно, при проецировании *N*-мерного пространства на двумерное некоторая информация теряется; тем не менее, этот метод бывает полезен, так как предоставляет возможность визуализации и понимания данных, иначе недоступных для осмысления.

Алгоритм Кохонена, который является основополагающим примером итеративного метода, последовательно перебирает серию эпох. В ходе каждой эпохи алгоритм последовательно обрабатывает весь набор обучающих данных, после чего применяется следующий вычислительный процесс:

– определить нейрон-победитель (т. е. тот, который находится на минимальном расстоянии от входного образца);

– модифицировать активированный нейрон путем приближения его параметров к данным входного образца, используя взвешенное среднее между текущим состоянием центра нейрона и данными обучающего образца.

В процессе вычисления взвешенной суммы применяется адаптация, где применяется постепенно снижающийся коэффициент обучения. Это обеспечивает более деликатную коррекцию весов с каждой новой эпохой обучения. В итоге это приводит к тому, что позиция центра кластера адаптируется к оптимальному положению, адекватно представляющему ту группу данных, на которых данный нейрон показал наилучший результат.

В алгоритмическом процессе достижения топологической упорядоченности ключевую роль играет концепция окрестности. Это область вокруг победившего нейрона, включающая нескольких его соседей. Аналогично изменению скорости обучения, радиус этой окрестности тоже сокращается по мере продвижения времени, начиная от обширной группы нейронов, потенциально охватывающей бóльшую часть топологической карты, до минимальной окрестности, ограниченной единственным выигравшим нейроном, на завершающих этапах. В контексте алгоритма Кохонена модификация весов осуществляется не только для выигравшего нейрона, но и для всех нейронов, находящихся в его актуальной окрестности в данный момент.

В процессе адаптации к новым условиям существенные части нейронной сети активно переориентируются к поданным примерам обучения, образуя первичную, грубую топологическую структуру, которая позволяет схожим входным данным стимулировать соседние нейроны в топологическом пространстве. С течением времени, по мере продолжения обучения, процесс адаптации сети утончается за счет постепенного уменьшения темпов обучения и радиусов влияния нейронов, позволяя тем самым вы-

являть все более деликатные различия в данных. Обычно этот процесс делится на две ключевые стадии: начальную, характеризующуюся высокой интенсивностью обучения и обширным взаимодействием между нейронами, и продвинутую, с фокусом на тонкую настройку и минимальной — практически нулевой — активностью соседних нейронов.

После обучения нейронной сети на определение структур данных, она становится эффективным инструментом для визуализации в процессе анализа данных. Анализируя данные о частоте побед каждого нейрона (Win Frequencies), который показывает, как часто каждый нейрон был наиболее активен при обработке обучающего набора данных, можно выявлять разделение данных на кластеры. Дополнительно, обрабатывая индивидуальные данные, можно наблюдать изменения в топологии карты, что помогает выявить логическую связь между кластерами. Часто для понимания смысловой нагрузки кластеров требуется вернуться к изначальным задачам анализа данных. Выявленные кластеры и иногда даже отдельные данные могут быть помечены метками на основе их смысла или функциональности. Когда топологическая карта полностью сформирована таким образом, в систему могут быть введены новые наблюдения. Если нейрон, который активируется для нового наблюдения, имеет предварительную маркировку, соответствующую классу, сеть выполняет классификацию. В случае, если нейрон не маркирован, сеть не делает однозначного вывода о классификации.

В методе Кохоненовских самоорганизующихся карт для задач классификации применяется концепция порога активации. Этот порог определяет максимально допустимое значение для расстояния между вектором входных данных и вектором весов нейрона, за которым следует успешное распознавание. Если активация победившего нейрона превышает установленный порог, то рассматривается, что сеть не способна классифицировать входной вектор. Это обеспечивает возможность Кохоненовским картам функционировать как детекторам новизны, отбрасывая объекты, которые существенно отличаются от предыдуще обученной модели, тем самым указывая на потенциально новые или аномальные явления при условии, что все нейроны были корректно идентифицированы и пороги активации настроены адекватно.

Концепция нейронных сетей Кохонена была разработана на основе схожести с определенными характеристиками человеческого мозга. Кора головного мозга, представляющая собой обширный плоский лист примерно 0,5 м^2 и для удобства размещения в черепной коробке сформированная в складчатую структуру, обладает уникальными топологическими особенностями. Эти особенности включают в себя связь между физическим расположением различных участков коры и функциями тела, которые они контролируют; например, область, управляющая движением руки, находится рядом с областью, отвечающей за управление движениями пальцев, создавая непрерывное и органическое отображение человеческого тела на этой двумерной плоскости.

Системы распознавания образов и машинного зрения. Сегодня системы оптического распознавания символов (OCR) и компьютерного зрения активно применяются в индустрии и строительстве для автоматизации процессов, связанных с переносом данных из бумажного формата и обработкой изображений, включая снимки, сделанные видеокамерами. Это особенно актуально при обработке материалов, полученных в ходе аэрофотосъемки и тахеометрических исследований.

Ключевые положения и единство восприятия. Фонтанное преобразование базируется на концепции холистического восприятия, согласно которой любой анализируемый объект рассматривается как интегральная структура, скомпонованная из связанных компонентов в рамках определенной системы отношений. Примером служит текстовая страница, которая разделяется на рубрики, рубрика включает в себя заголовок и текст, текст разбивается на абзацы, абзацы — на предложения, предложения — на слова, а слова — на буквы. Вся эта иерархия элементов текста объединена через специфичные пространственные и семантические связи.

Чтобы понять, что представляет собой целое, необходимо идентифицировать составляющие его элементы. Эти элементы, впрочем, анализируются исключительно как компоненты целого. Следовательно, процесс целостного восприятия предполагает формулировку гипотезы о характере объекта в качестве целого. Исходя из этой гипотезы осуществляется выделение и толкование составных частей объекта. Дальнейшая задача заключается в по-

пытке реконструкции целого из его частей для верификации первоначальной гипотезы. Стоит отметить, что объект восприятия может быть воспринят и как элемент обширного контекста или большего целого. Таким образом, в процессе чтения индивид идентифицирует буквы, осмысливает слова, объединяет их в грамматические структуры и постигает значение.

В инженерных системах процесс распознавания текста не является простым выбором между вариантами; он основывается на методике, предполагающей итеративное формирование и верификацию различных предположений с использованием как специфических знаний о анализируемом объекте, так и контекстной информации. Эффективность восприятия определенной категории объектов обусловлена двумя ключевыми критериями: ни один элемент вне этой категории не соответствует заданному критерию, однако каждый элемент внутри нее соответствует ему без оговорок. К примеру, дефиниция категории изображений буквы «К» должна включать все возможные варианты изображений «К», исключая при этом изображения других букв. Такой подход характеризуется качеством отображаемости — способностью системы создавать визуальный или звуковой эквивалент исходного объекта: стандарт буквы для OCR-технологий позволяет ее визуализировать, образец слова в системах распознавания голоса дает возможность его произнести, а модель структуры фразы в синтаксических анализаторах обеспечивает генерацию корректных предложений. С точки зрения практического применения возможность такого отображения критична, так как поддерживает высокое качество системных дефиниций.

Два метода описания целостности включают шаблонный и структурный подходы. В первой ситуации описывается графическое представление, выполненное в растровой или векторной графике, с указанием категории трансформаций, таких как репликация, масштабирование и другие подобные операции. В альтернативном подходе представление информации выполняется через графическую структуру, где узлы символизируют компоненты начального объекта, связанные дугами, обозначающими их взаимное пространственное расположение. Компоненты, в свою очередь, могут быть многоуровневыми, предполагая наличие подробного внутреннего описания.

Безусловно, применение шаблонных методов в практике значительно упрощает задачу по сравнению со структурными методами. Тем не менее, их эффективность снижается при необходимости анализа объектов, характеризующихся большим разнообразием. В качестве примера можно привести то, что шаблонные методы подходят для идентификации типографских знаков, в то время как структурные методы способны адаптироваться и к анализу рукописного текста.

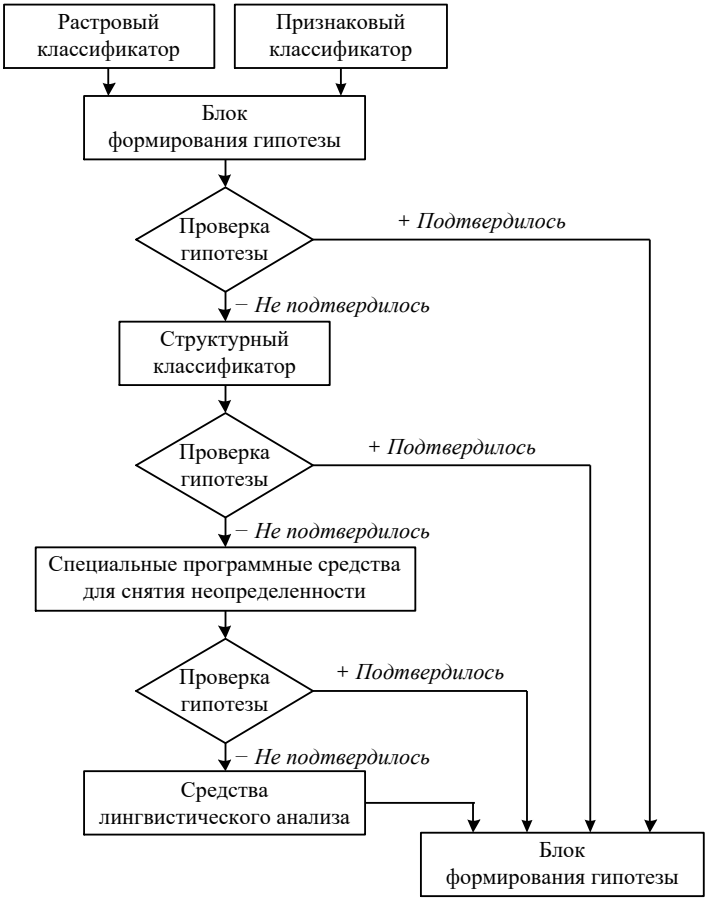


Рис. 26. Обобщенная диаграмма функционирования системы идентификации изображений

Согласно концепции холистического подхода к обработке информации выделяются две ключевые стратегии, как показано на рис. 26.

В первую очередь необходимо обеспечить параллельную работу всех доступных источников данных. Это означает, что процедура распознавания текста должна происходить синхронно с лингвистической и контекстуальной обработкой, избегая последовательного подхода, который исключает возможность использования обратной связи для корректировки распознавания на основе контекста. Второй ключевой момент заключается в необходимости воспринимать и анализировать объект исследования как единое целое, что позволяет улучшить точность интерпретации данных.

Первый этап, называемый восприятием, означает построение предположений относительно объекта наблюдения. Эти предположения могут базироваться как на заранее имеющихся знаниях о объекте, включая контекстуальные данные и оценку прошлых предположений (методология «топ-даун»), так и на начальном разборе самого объекта (подход «боттом-ап»). Второй этап заключается в детализации восприятия через верификацию выдвинутой гипотезы, включая глубокий анализ объекта с целью его понимания в соответствии с предложенной концепцией, при этом активно используется контекстуальная информация.

Для облегчения анализа и обработки данных требуется их предварительная подготовка, при которой ключевые детали остаются нетронутыми. Этот этап включает конвертацию исходных данных в формат, приемлемый для последующих операций (как пример, преобразование изображений в векторный формат), либо извлечение различных вариантов разбиения данных на составные элементы, среди которых верный вариант выбирается путем формулирования и тестирования различных предположений. Создание и оценка гипотез должны быть четко включены в структуру программного обеспечения. Каждая предположение представляется как отдельная сущность для возможности ее оценки или сравнения. Обычно гипотезы генерируются поочередно, собираются в один перечень и ранжируются в соответствии с их предварительным анализом. При этом для окончательного выбора оптимального предположения активно задействуются дополнительный контекст и внешние данные.

Распознавание символов. Оптическое распознавание символов (OCR) представляет собой процесс преобразования изображений текста — будь то рукописный, печатный или набранный на машинке — в машиночитаемый текстовый формат, т. е. в последовательность символов, которые компьютеры могут обрабатывать, например, в документах текстовых процессоров. Эта технология находит широкое применение в преобразовании бумажных изданий, документов в цифровой формат, автоматизации документооборота в бизнесе и публикации текстов в Интернете. OCR дает возможность вносить изменения в тексты, искать необходимые слова или выражения, сокращать объем хранения информации, без потери качества выводить на печать, анализировать данные, а также позволяет применять к тексту машинный перевод, форматирование или преобразование текста в устную речь. Оптическое распознавание символов является предметом исследований в таких дисциплинах, как распознавание образов, искусственный интеллект и компьютерное зрение.

Сегодня наиболее популярными являются так называемые «умные» технологии, обладающие высоким уровнем аккуратности в идентификации различных шрифтов. Определенные программы оптического распознавания текста обладают возможностью реконструкции оригинального форматирования документа, восстанавливая не только текст, но и графические изображения, макеты с колонками и другие визуальные элементы.

На сегодняшний день в области оптического распознавания символов различают три основных метода: шаблонный, структурный и признаково-ориентированный. Однако только шаблонный и структурный методы соответствуют принципу целостности обработки информации.

Применение шаблонных описаний облегчает процесс реализации, но ограничивает возможность отображения сложно структурированных объектов с широким спектром уникальных особенностей. Эта методология обычно находит применение в распознавании типографских символов, в отличие от структурного подхода, который адаптируется к рукописным текстам. В рукописных текстах встречается значительное разнообразие стилей написания, что требует гибкого и сложного подхода, каким и является структурное описание.

Шаблонные системы. Шаблонные системы распознавания символов функционируют, превращая входящее изображение конкретного знака в его пиксельное представление, после чего осуществляют поиск по базе данных для нахождения наиболее сходного шаблона, опираясь на минимальное количество отличающихся пикселей. Эти системы отличаются прочностью к повреждениям на изображении и способностью быстро обрабатывать информацию, однако их эффективность ограничивается исключительно теми шрифтами, данные по которым уже занесены в их базы. В ситуациях, когда шрифт несколько отличается от того, который заложен в системе в качестве эталона, шаблонные механизмы могут допускать ошибки при анализе даже высококачественных изображений, что подчеркивает их ограниченную адаптивность к новым или нестандартным шрифтам.

Структурные системы. В этих системах представление объекта производится посредством графовой структуры, где элементы объекта отображаются в виде вершин, а взаимосвязи в пространственном контексте между этими элементами — в виде ребер. Такие системы, применяющие данный метод, предпочтительно используют векторные форматы изображений, поскольку они позволяют точно и эффективно описывать графические объекты через их составные структурные части, например, линейные элементы, составляющие символы. В контексте буквы «р» структурные элементы будут представлены вертикальной линией и прилегающей к ней дугой.

Одним из главных недостатков структурных подходов в распознавании образов является их высокая восприимчивость к искажениям на изображении, которые могут нарушать структуру компонент. Векторизация, используемая в этих системах, может внести дополнительные артефакты, ухудшая качество распознавания. В отличие от шаблонных и признаковых методов, до настоящего времени не разработаны высокоэффективные методы машинного обучения для структурных систем, что существенно затрудняет их автоматизацию. В частности, для реализации в программе FineReader структурные модели приходится разрабатывать вручную, что существенно увеличивает трудозатраты и снижает гибкость процесса настройки под новые задачи.

Признаковые системы. В таких методах представление каждого символа осуществляется путем интеграции его в *n*-мерное пространство особенностей, где определяется набор ключевых характеристик, подлежащих вычислению для распознаваемых изображений. Этот процесс включает в себя сопоставление вычисленного *n*-мерного вектора атрибутов с эталонными значениями для определения наиболее соответствующего символа. Однако методы, основанные на признаковом описании, часто сталкиваются с проблемой интегральности: чтобы класс объектов (в нашем контексте — группа изображений одного символа) был адекватно описан, к нему должны относиться все принадлежащие ему экземпляры без включения внешних. Ввиду потери значительного объема информации при анализе характеристик сложно достичь условия, при котором класс будет включать исключительно соответствующие ему элементы.

4.6. Новое в нейросетях — нейроморфные системы

Искусственные нейронные сети стали неотъемлемой частью современной жизни, находя широкое применение в многих областях. Они лежат в основе работы голосовых ассистентов, алгоритмов для обработки запросов в поисковиках, разработки беспилотных транспортных средств, создания виртуальных изображений и автоматизированной генерации текста.

Сегодня многие знают термин «нейросеть», в основном, благодаря продуктам типа генерированных изображений и текстов. Чем же нейросеть отличается от базового искусственного интеллекта?

Искусственный интеллект охватывает широкий спектр научных исследований и технологий, позволяя различным экспертам предлагать уникальные определения. Среди множества подходов к реализации ИИ нейронные сети выступают как одна из ключевых технологий. Для аналогии рассмотрим анализ финансовых графиков, где тренды могут быть идентифицированы через аппроксимацию данных линейной моделью, характеризующейся коэффициентами направления и перемещения.

Искусственная нейронная сеть представляет собой комплекс взаимодействующих искусственных нейронов, которые выполняют функции на основе сложных математических алгоритмов. Даже в базовой модели нейронной сети число таких параметров измеряется сотнями или тысячами, в то время как в более сложных системах оно может доходить до 300 млрд. Основной задачей является настройка этих параметров путем обучения сети, что позволяет задавать точные значения каждому из них. Как результат, когда в систему подается новая информация, эти многомерные взаимосвязи обеспечивают выдачу адекватного вывода. Это может быть, например, идентификация объектов на фотографии как кошек или собак, выполнение прогнозов на базе анализа данных или даже предсказание химического состава вещества по заданным характеристикам.

Обучение нейросети включает процесс подстройки весов между нейронами для минимизации разницы между фактическими и прогнозируемыми выходными значениями на основе входных данных.

В области искусственного интеллекта методы машинного обучения делятся на две крупные категории: с учителем и без учителя. Обучение с учителем требует обширной базы данных с аннотациями от эксперта для каждого образца, включая правильные ответы. Например, если передать нейросети изображение для классификации среди 10 категорий, таких как кошки, собаки и т. д., нейросеть должна активировать соответствующий выход. Если выход нейросети не совпадает с заданным экспертом, активируется механизм обратного распространения ошибок, настраивающий веса нейронов для корректировки результата. Этот процесс последовательного предъявления образцов и корректировки весов повторяется многократно, до достижения минимального значения функции потерь, указывающего на оптимизацию параметров сети. Для валидации результатов обучения и избежания переобучения используется часть данных, не участвовавшая в тренировке, а качество модели оценивается по доле правильных ответов. В методе обучения без учителя задействуются механизмы вознаграждения и наказания, мотивирующие систему к самостоятельной адаптации весов в процессе функционирования без предварительно заданных эталонных ответов.

Нейроморфная система — это тип вычислительной технологии, имитирующий структуру и принципы работы мозга. В отличие от традиционных искусственных нейросетей, которые разрабатываются для работы на обычных компьютерах и серверах, нейроморфные системы используют специализированные аппаратные средства для более эффективного и энергоэффективного моделирования нейронных сетей.

В традиционной вычислительной практике математические разработки специализированных алгоритмов и формул для определенных вычислительных задач играют ключевую роль. Эти задачи подразумевают прямую взаимосвязь между входными данными и результатами, требуя значительных усилий и ресурсов для создания и тестирования формулы. В контрасте с этим искусственные нейронные сети предлагают альтернативный подход к решению задач, особенно тех, что сложно формализовать. В таких случаях правила для вычислений не полностью определены или слишком сложны для явного выражения. В этом контексте разработчику нейросетей предстоит собрать обширный набор данных для тренировки, явно указывая ожидаемый исход для каждого примера — это процесс, известный как разметка данных. После этого происходит обучение нейросети на основе этих данных. Таким образом, процесс разработки переносит акцент с формулировки точных алгоритмов на подготовку и подачу обучающих данных, выбор подходящей архитектуры нейросети, ее слоев и нейронов, а также настройку других параметров, оставляя выполнение обучения и последующих вычислений за вычислительными системами.

Для тренировки и функционирования нейронных сетей активно применяются два вида вычислительных устройств: традиционные ускорители на базе архитектуры фон Неймана, разработанной в 1940-е гг., и более новаторские нейроморфные вычислители. Структура процессора влияет на связи между вычислительными элементами и устройствами для хранения данных, или памятью. Процессоры, предназначенные изначально для выполнения широкого спектра задач (CPU), с течением времени претерпели эволюцию к более специализированным устройствам, таким как процессоры для обработки цифровых сигналов (DSP) и графические процессоры (GPU), нацеленные на обработку вектор-

ной графики высокой точности и реализма. Поскольку ключевыми операциями в нейронных сетях являются умножение и сложение предыдущих результатов, GPU, адаптированные к подобным вычислениям, оказываются исключительно подходящими для этих целей.

В настоящее время на графических процессорах происходит большая часть процесса обучения нейронных сетей.

Основной недостаток графических процессоров заключается в их повышенном потреблении электроэнергии. В связи с этим активно развивается альтернативное направление — создание нейронных ускорителей. Эти миниатюрные интегральные схемы предназначены для эффективной работы с уже тренированными искусственными нейронными сетями. Вопреки всему данные устройства по-прежнему основываются на традиционной архитектуре нейрона, несмотря на то, что область нейробиологии продолжает активно развиваться и приносить новые открытия.

В определенный момент исследования ученых в области нейробиологии привели к открытию, что нейроны в человеческом мозге функционируют за счет прерывистых электрических сигналов, известных как спайки, активируясь только при необходимости, в отличие от непрерывного сигнала, предполагаемого в традиционных моделях искусственных нейронных сетей. Это открытие побудило как математиков, так и специалистов в области электроники сфокусироваться на новом направлении исследований, получившем название «нейроморфная инженерия», подразумевая разработку технологий, в основе которых лежит принцип работы мозга.

Ключевая сложность традиционных вычислительных архитектур заключается в их фундаментальном разделении вычислительного ядра и блока памяти, из-за чего последняя значительно отстает в скорости, а именно — в десятки раз медленнее, чем центральное процессорное устройство. Это создает заметные задержки в передаче данных. В контрасте, нейроморфная инженерия стремится к слиянию функций хранения и обработки данных в одном элементе, тем самым эмулируя работу нейронов мозга. Это принципиально новое построение компьютерной архитектуры позволяет значительно повысить как скорость обработки данных, так и энергоэффективность системы, поскольку устраня-

ется необходимость в отдельных циклах чтения и записи между процессором и памятью.

В сфере традиционных (стандартных) искусственных нейронных сетей сложилась обширная и эффективно функционирующая инфраструктура, включающая в себя не только передовые алгоритмы и программное обеспечение, но и инструментарий для разработки, известный как фреймворки, а также специализированные вычислительные решения для их тренировки и выполнения. Именно в этой области сосредоточено внимание академического сообщества, индустрии образования и предпринимательского сектора. В то время как сектор нейроморфных вычислений демонстрирует значительный прогресс, видимый через научные работы и исследования, этот сегмент по-прежнему не достиг уровня массовой коммерциализации. В нашей стране действуют несколько выдающихся научно-исследовательских групп, активно работающих над созданием уникальных фреймворков для традиционных нейросетей и их интеграцией с местными аппаратными средствами ускорения вычислений для нейросетевых применений.

Развитие экосистем для нейроморфной технологии представляет большие сложности. Такие процессоры разнообразны и могут быть классифицированы в следующие группы: на основе стандартных транзисторов с инновационной архитектурой, возможным использованием принципиально новых видов памяти; комбинированные цифро-аналоговые системы; а также чисто аналоговые системы, выполняющие операции непосредственно в элементах памяти, например, на базе мемристоров. В рамках отечественных достижений стоит выделить цифровой нейроморфный процессор «Алтай», однако комплексной платформы для его эффективного использования пока что не разработано. Международный гигант Intel представил в 2018 и 2021 гг. две модификации своего цифро-аналогового нейроморфного процессора Loihi, сопровождаемого развитым программным обеспечением Lava. Отечественные аналоги в этом сегменте на данный момент недоступны. Параллельно поле аналоговых нейроморфных процессоров претерпевает интенсивное развитие как за пределами России, так и внутри страны.

В 1971 г. Леон Чуа, работая в области электротехники, выдвинул теорию о существовании четвертого фундаментального

электронного компонента в дополнение к уже известным — емкости, индуктивности и сопротивлению. Этот элемент он предложил назвать мемристором, уникальным тем, что его электрическое сопротивление не является постоянной величиной, а зависит от предыдущего прохождения через него электрического заряда. Таким образом, мемристор обладает способностью «запоминать» свое предыдущее электрическое состояние даже после отключения питания. В 2008 г., команда исследователей из компании Hewlett Packard объявила о реализации мемристора на практике, что стало значимым прорывом в области электроники. У мемристоров выделены шесть основных характеристик, определяющих их функциональность и применимость в различных электронных устройствах. Однако до сих пор не существует мемристора, который бы удовлетворял всем барьерам производительности по этим ключевым параметрам. Мемристоры лежат в основе разработки аналоговых нейроморфных процессоров, имитирующих работу нейронных сетей мозга, что открывает новые перспективы в создании вычислительных систем и искусственного интеллекта.

Разрабатываются электронные схемы для обеспечения функционирования матрицы таких мемристоров. В нейросети производятся векторно-матричные умножения, а матрица мемристоров как раз и выполняет их на аппаратном уровне на максимальной скорости в аналоговой форме, оперируя напряжениями и токами по законам Ома и Кирхгофа. Интерфейсная электронная схема позволяет преобразовать двоичный код в импульсные сигналы, обрабатывать напряжения и токи, возвращать результат вычислений мемристивной матрицы обратно в двоичный код.

Нейроморфные системы способны решать разнообразные задачи, такие как распознавание образов, обработка естественного языка, оптимизация процессов и предсказание временных рядов, благодаря их способности к быстрому и эффективному обучению на основе моделирования процессов, происходящих в человеческом мозге.

Системы, работающие на основе цифровых и смешанных цифро-аналоговых нейроморфных процессоров, продвигаются в выполнении сложных операций, аналогичных тем, что доступны специализированным нейронным ускорителям. Они отличаются повышенной энергоэффективностью и оперативностью

благодаря инновационным архитектурам и компонентам. В то время как объем доступных для аналоговых нейроморфных процессоров вычислительных матриц все еще оставляет желать лучшего для выполнения подобного круга задач, эти процессоры находят применение в специфических областях. Например, возьмем голосовые ассистенты, такие как «Алиса», которые требуют значительной вычислительной мощности для анализа аудиопотока, особенно при обработке данных локально, без использования cloud-серверов. В таких случаях важно активировать основной процессор лишь после улавливания активационной команды, например, «Алиса». На помощь приходит компактная, энергоэффективная нейроморфная микросхема, непрерывно мониторящая окружающий звуковой фон, которая обнаруживает данную фразу и активирует мощный процессор для дальнейшей обработки.

Во многих странах проекты Илона Маска в области интеграции микрочипов в организм человека открывают новые возможности для людей с проблемами здоровья, применяя роботизированные технологии в повседневной деятельности. Пропуская подробности его исследований, стоит упомянуть, что применение мемристоров также предоставляет возможность создания интерфейса между биологическими тканями и полупроводниковой электроникой. Исследователи из Нижнего Новгорода обозначили это направление как нейроэлектроника, предложив наглядный термин. Использование нейронных сетей способствует решению задач, априори недоступных для человеческого мозга, в то время как человек способен интуитивно и мгновенно анализировать и синтезировать информацию из окружающей среды, выявляя причинно-следственные связи. Хотя нейроморфные системы могут не всегда достичь человеческого уровня в понимании сложных связей из-за ограниченного объема нейронных взаимосвязей, они представляют собой значимый шаг в области искусственного интеллекта, обеспечивая возможность компенсировать или усиливать физиологические функции посредством нейроинтерфейсов. Эти системы могут функционировать как «умные помощники», восполняя недостающие или улучшая существующие способности человека.

В разработке экзоскелетов предусмотрена интеграция нейроинтерфейса, позволяющая управлять протезом через непосред-

ственное взаимодействие с нервной системой. В настоящее время функционирование протезированной конечности обеспечивается за счет электродвигателя, который регулирует ее положение исходя из разницы между текущим и желаемым состояниями. Для индивидов, переживших инсульт, существенным ограничением является утрата способности мускулатуры реагировать на мозговые команды, в то время как прямое электростимулирование с использованием сигналов определенной амплитуды и частоты может инициировать желаемое движение. В обозримом будущем развитие нейронных сетей даст возможность манипулировать мышечными реакциями с помощью специфических электрических импульсов для достижения необходимой моторики. Исследования показывают, что систематическое тренировочное воздействие способно восстановить реактивность мускулатуры на центральные нервные стимулы, тем самым восстанавливая потерянные физиологические функции.

Академик Г. Я. Красников уверен в том, что человечество будет эволюционировать в направлении постепенного превращения в кибернетические организмы, т. е. киборгов [3].

Выводы по главе 4

Рассмотрены проблемы построения искусственных нейронных на основе модели нейрона головного мозга человека.

Используя математическую модель биологического нейрона, приведена упрощенная схема искусственного нейрона, которая была предложена У. Мак-Каллоком (нейробиолог) и У. Питтсом (логик). В 1950 г. нейрофизиолог Ф. Розенблатт разработал математическую модель перцептрона. Перцептрон стал одной из первых моделей искусственных нейросетей.

Для нейрона как сумматора приведена программа в языке Python. Приведена подробная интерпретация работы искусственного нейрона.

Приведена функция активации как функция, которая в качестве входного параметра получает взвешенную сумму S как аргумент, а на выходе формирует значение выходного сигнала из нейрона (y). Рассмотрены некоторые математические функции,

которые могут быть использованы в качестве функции активации. Определено целое семейство сигмоидальных — логических функций, и некоторые из них применяют в качестве функции активации в искусственных нейронах.

Показаны нейронные сети, графическая интерпретация искусственных нейронных сетей. Рассмотрены однослойные и многослойные искусственные нейронные сети.

Рассмотрена генерализованная регрессионная нейронная сеть, построенная как вероятностная нейронная сеть, предназначенная для решения задач регрессии, а не классификации, как PNN.

В учебном пособии рассматривается новая технология и модели технологий искусственного интеллекта — нейробионика и нейрокомпьютеры.

Принципиальная цель искусственного интеллекта заключается в эмуляции человеческого мышления, а естественный аналог «мыслящей системы» представлен мозгом, логическим становится стремление к созданию «искусственного мозга», реплицирующего функционал и структуру человеческих нейронных сетей. Этому аспекту посвящено направление в области искусственного интеллекта, известное как нейрокибернетика.

Нейрокомпьютеры эффективно справляются с широким спектром задач, требующих высокого уровня интеллекта. В их число входят распознавание образов, адаптивное регулирование, предсказательный анализ, диагностические функции и другие подобные задачи.

Исследования в области нейробиологии дали толчок к прогрессу в разработке нейронных сетей и нейрокомпьютеров.

В учебном пособии показаны главные отличия нейрокомпьютеров от традиционных поколений вычислительных систем и основные направления исследований:

– разработка нейроалгоритмов;

– разработка специализированного ПО для симуляции нейронных сетей;

– разработка целевых вычислительных платформ для моделирования работы искусственных нейронных сетей;

– цифровое моделирование искусственных нейронных сетей;

– оптоэлектронные системы для моделирования нейронных сетей.

Рассмотрены основные функции, выполняемые искусственными нейронными сетями.

Приведены основные свойства и функции наиболее интересной нейронной сети — сети Кохонена.

Подробно рассмотрены концептуальные аспекты машинного обучения и технологии обработки изображений и распознавания образов.

Приведена обобщенная диаграмма функционирования системы идентификации изображений.

Подробно рассматривается новое направление в нейронных сетях — нейроморфные системы.

Вопросы для самоконтроля

1. Что такое биологическая нейронная сеть?
2. Искусственная нейронная сеть как модель биологической нейронной сети.
3. Упрощенная схема искусственного нейрона. Зачем она нужна?
4. Упрощенная математическая модель нейрона.
5. Назначение весовых коэффициентов нейрона.
6. Что такое функция активации нейрона искусственной сети?
7. Интерпретация сигмоидальной функции активации.
8. Искусственная нейронная сеть и графическая интерпретация.
9. Что является ключевым моментом в концепции искусственных нейронных сетей?
10. Однослойная нейронная сеть, ее свойства и назначение.
11. Многослойная нейронная сеть, ее свойства и назначение.
12. Вероятностная нейронная сеть и ее интерпретация.
13. Обобщенно-регрессионная нейронная сеть и ее свойства.
14. Понятие нейробионики и нейрокомпьютера.
15. Основные задачи, решаемые нейроморфными системами. Способы решения разнообразных интеллектуальных задач.
16. Обобщенные составляющие функционирования системы идентификации изображений.

Заключение

В настоящее время в России поставлены и решаются принципиальные задачи цифровой экономики (4-й промышленной революции) — перевести всю экономику, социальную сферу, органы власти, работу органов власти на качественно новые принципы работы, внедрить управление на новых данных — на основе больших данных и технологиях искусственного интеллекта.

Сейчас технологии ИИ стали массовыми и повсеместными, проникли в нашу повседневную жизнь и вряд ли ее покинут. Они используются в поисковых и рекомендательных системах, транспорте, логистике, банковском деле, планировании бизнес-процессов, производстве и научных исследованиях. Они уже давно не ограничиваются цифровой реальностью, проникая в быт. Нас начинают окружать домашние роботы, беспилотные аппараты, «умные» дома и города, не говоря уже о приложениях с элементами ИИ для смартфонов и персональных компьютеров.

Существующие технологии ИИ открывают перед нами огромные перспективы. Они способны придать новый импульс развитию мировой экономики, оказать позитивное влияние на все сферы нашей жизни. Все, что связано с данными, большими данными, принимает критически важное значение, а технологии ИИ позволяют эффективно их обрабатывать. Речь, по сути, идет о системообразующей инфраструктуре для нашего дальнейшего развития, для будущего нашей экономики в целом. Научиться методам и подходам в данной области — технологиях ИИ, знаниям как рабочему инструменту можно только при полном овладении приемами и технологией решения задач. Эту цель и преследует данное учебное пособие, предлагая необходимое количество теории для решения практических задач, при этом представленной таким образом, чтобы процесс реализации теоретических блоков в виде программ и алгоритмов на вычислительных машинах был наименее трудоемким.

Технологии ИИ участвуют в глобальных преобразованиях технологической индустрии. Цифровая сфера развивается ускоренными темпами благодаря появлению технологий искусственного интеллекта и большим данным. Эти составляющие представлены в платформе ASP.NET Core MVC и среде Visual Studio 2019, 2022 как машинное обучение — ML.NET и SQL Server 2019 и 2022. В современной экономике нам необходимо понимать, что такое искусственный интеллект и каковы перспективы его в будущем.

Библиографический список

Список использованной литературы

1. *Душкин Р. В.* Искусственный интеллект. М.: ДМК Пресс, 2019. 280 с.

2. *Каляев И. А.* Искусственный интеллект: камо грядеши? // Экономические стратегии. 2019. Т. 21, № 5 (163). С. 6−15.

3. *Красников Г. Я.* Человек будет постепенно становиться небольшим киборгом // Коммерсантъ Наука. 29 сент. 2021. № 33. С. 45. URL: https://www.kommersant.ru/doc/5005683.

4. *Остроух А. В., Суркова Н. Е.* Системы искусственного интеллекта: монография. 2 е изд., стер. СПб.: Лань, 2021. 228 с.

5. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход: пер. с англ. 2-е изд. СПб.: Диалектика, 2020. 1408 с.

6. *Тьюринг А.* Может ли машина мыслить? М.: Государственное издательство физико-математической литературы, 1960. 68 с.

Рекомендуемая литература

7. *Брайтон Г.* Искусственный интеллект в комиксах / ил. Говарда Селины; пер. с англ. Д. Кудряшова. М.: Эксмо, 2018. 176 с.

8. *Брайтон С. Л., Куц Дж. Н.* Анализ данных в науке и технике / пер. с англ. А. А. Слинкина. М.: ДМК Пресс, 2021. 542 с.

9. *Бутл Р.* Искусственный интеллект и экономика: работа, богатство и благополучие в эпоху мыслящих машин: пер. с англ. М.: Альпина ПРО, 2023. 424 с.

10. *Гатман А. Дж., Голдмейер Дж.* Разберись в Data Science. Как освоить науку о данных и научиться думать как эксперт / пер. с англ. М. А. Райтмана. М.: Эксмо, 2023. 304 с.

11. *Искусственный* интеллект — надежды и опасения: сборник / пер. с англ. В. Желнинова; под ред. Дж. Брокмана. М.: АСТ, 2020. 384 с.

12. *Люгер Дж. Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем. 4-е изд.: пер. с англ. М.: Вильямс, 2003. 864 с.

13. *Маркс Р., Дембски У., Эверт У.* Введение в эволюционную информатику / пер. с англ. В. С. Яценкова. М.: ДМК Пресс, 2020. 276 с.

14. *Марр Б., Уорд М.* Искусственный интеллект на практике. 50 кейсов успешных компаний / пер. с англ. Е. Петровой. М.: Манн, Иванов и Фербер, 2020. 320 с.

15. *Мишра П.* Объяснимые модели искусственного интеллекта на Python. Модель искусственного интеллекта. Объяснения с использованием библиотек, расширений и фреймворков на основе языка Python / пер. с англ. С. В. Минца. М.: ДМК Пресс, 2022. 298 с.

16. *Нейман, Дж. фон.* Вычислительная машина и мозг / пер. с англ. А. Чечиной. М.: АСТ, 2018. 192 с.

17. *О'Коннелл М.* Искусственный интеллект и будущее человечества / пер. с англ. М. Кудряшова. М.: Эксмо, 2019. 272 с.

18. *Оливейра А.* Цифровой разум: как наука меняет человечество / пер. с англ. К. Чистопольской; под науч. ред. М. Фаликман. М.: Изд. дом «Дело», 2022. 448 с.

19. *Падуа С.* Невероятные приключения Лавлейс и Бэббиджа. (Почти) правдивая история первого компьютера. М.: Манн, Иванов и Фербер, 2017. 320 с.

20. *Сейновски Т.* Антология машинного обучения: важнейшие исследования в области ИИ за последние 60 лет / пер. с англ. М. А. Райтмана, Е. В. Сазановой. М.: Эксмо, 2022. 304 с.

21. *Сильный* искусственный интеллект. На подступах к сверхразуму / М. С. Бурцев, О. Л. Бухвалов, А. А. Ведяхин и др. М.: Интеллектуальная Литература, 2021. 232 с.

22. *Тьюринг А.* Игра в имитацию: о шифрах, кодах и искусственном интеллекте / пер. с англ. Ю. Данилова. М.: Родина, 2019. 192 с.

23. *Уинстон П.* Искусственный интеллект / пер. с англ. В. Л. Стефанюка; под ред. Д. А. Поспелова. М.: Мир, 1980. 520 с.

24. *Финн В. К.* Искусственный интеллект: методология, применения, философия / науч. ред. М. А. Михеенкова. М.: КРАСАНД, 2018. 448 с.

25. *Холмс У., Бялик М., Фейдел Ч.* Искусственный интеллект в образовании: Перспективы и проблемы для преподавания и обучения: пер. с англ. М.: Альпина ПРО, 2022. 304 с.

26. *Часовских В. П.* Библиотека методических указаний 2024−2025 учебный год. URL: http://vikchas.ru/Public/Bibliotekas/Index.

27. *Шваб К.* Четвертая промышленная революция: пер. с англ. М.: Эксмо, 2018. 208 с.

28. *Шваб К., Дэвис Н.* Технологии Четвертой промышленной революции: пер. с англ. М.: Эксмо, 2018. 320 с.

29. *Labunets V. G.* Clifford Algebras as Unified language for multicolor image processing and pattern recognition // Computational Noncommutative Algebra and Applications. NATO/Advanced Study Institute. 2003. P. 197−225.

30. *Mukherjee S.* ML.NET Revealed: Simple Tools for Applying Machine Learning to Your Applications. Apress, 2020. 192 s.

Информация об авторах

Часовских Виктор Петрович — профессор кафедры шахматного искусства и компьютерной математики УрГЭУ, доктор технических наук, профессор
e-mail: u2007u@ya.ru

Стариков Евгений Николаевич — заведующий кафедрой шахматного искусства и компьютерной математики УрГЭУ, кандидат экономических наук, доцент
e-mail: starikov_en@usue.ru

Акчурина Галия Абдулазисовна — старший преподаватель кафедры шахматного искусства и компьютерной математики УрГЭУ
e-mail: gaa20111@yandex.ru

Кох Елена Викторовна — доцент кафедры шахматного искусства и компьютерной математики УрГЭУ, кандидат сельскохозяйственных наук, доцент
e-mail: elenakox@mail.ru

Оглавление

Учебное издание

Часовских Виктор Петрович,
Стариков Евгений Николаевич,
Акчурина Галия Абдулазисовна,
Кох Елена Викторовна

Системы искусственного интеллекта

Учебное пособие

Корректор *В. К. Матвеев*
Компьютерная верстка *И. В. Засухиной*

Поз. 49. Подписано в печать 24.12.2024.
Формат 60 × 84 1/16. Бумага офсетная. Печать плоская.
Уч.-изд. л. 8,4. Усл. печ. л. 11,4. Печ. л. 12,3. Заказ 273. Тираж 23 экз.
Издательство Уральского государственного экономического университета
620144, г. Екатеринбург, ул. 8 Марта / Народной Воли, 62/45

Отпечатано с готового оригинал-макета в подразделении оперативной полиграфии
Уральского государственного экономического университета

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ