Донбасская государственная машиностроительная академия

Кафедра автоматизации производственных процессов

КОМПЬЮТЕРНЫЕ СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Конспект лекций

(для студентов специальности

“Компьютерная инженерия»)

Утверждено
на заседании
кафедры АПП
Протокол № 3 от 06.11.2017

Краматорск 2017

УДК 681.3

Компьютерные системы искусственного интеллекта: конспект лекций (для студентов специальности “Компьютерная инженерия» / сост. А. А. Сердюк. – Краматорск : ДГМА, 2017. – 52 с.

Изложены вопросы теории и практики систем искусственного интеллекта. Рассмотрены процедуры создания и обучения нейронных сетей. Освещены проблемы и технологические приемы создания интеллектуальных, в частности, экспертных систем.

Составитель А. А. Сердюк, доц.

Отв. за выпуск С. П. Сус, доц.

СОДЕРЖАНИЕ

1 ОСНОВЫ ТЕОРИИ НЕЙРОННЫХ СЕТЕЙ

«Вся жизнь – от простейших до сложнейших организмов, включая человека, – есть длинный ряд все усложняющихся уравновешиваний окружающей среды».

И. П. Павлов

1.1 Современные представления об искусственном интеллекте

Термин "искусственный интеллект" был введен в результате неудачного перевода с английского "artificial intelligence", что на самом деле означает "умение рассуждать разумно".

Родоначальником искусственного интеллекта (ИИ) считают [1] средневекового испанского философа, математика и поэта Раймонда Луллия, который пытался в XIII веке создать механическую машину для решения различных задач на основе разработанной им классификации понятий, иначе говоря, структуры некоторой базы знаний. В XVIII веке идею Луллия продолжили Лейбниц и Декарт, предложившие универсальные языки классификации – первые теоретические работы в области искусственного интеллекта. Однако как наука искусственный интеллект сформировался в результате исследований Норберта Винера – отца кибернетики. В настоящее время – это одна из наиболее перспективных и престижных областей информатики.

Современная идея искусственного интеллекта формулируется следующим образом: "Единственный объект, способный мыслить, – это человеческий мозг. Поэтому любое мыслящее устройство должно каким-то образом воспроизводить его структуру" [1]. В связи с этим была принята ориентировка на программно-аппаратное моделирование структуры мозга и создание элементов, аналогичных нейронам – нервным клеткам мозга.

Первые нейронные структуры были созданы в 1956–1965 годах Розенблаттом и Мак-Каллоком. Эти устройства были названы персептронами (perceptron). Однако персептроны не получили широкого распространения, так как результаты их применения были неутешительными из-за упрощенных топологических и алгоритмических построений.

Развитие идей Розенблатта позволило создать ряд топологических вариантов нейронных сетей и алгоритмов их обучения, которые нашли применение для решения достаточно сложных практических задач. Нейронные сети строились на базе существующих компьютеров путем программного моделирования структуры нейрона. При этом база знаний нейрона создавалась путем введения обучающей выборки – *набора данных*

в виде массивов входных и желаемых выходных сигналов.

Однако при таком подходе не удавалось решать очень важные для практики задачи по машинному переводу текстовой информации (идея пословного перевода оказалась неэффективной), по управлению робототехникой (решение проблем в создании, хранении и обработке трехмерной визуальной информации), а также задачи в области принятия решений по сложным проблемам, в распознавании образов и др.

Начиная с середины 1980-х годов, интерес к искусственному интеллекту стал резко возрастать. В это время в Японии был создан первый нейрокомпьютер, а затем и транспьютеры – параллельные компьютеры с большим количеством процессоров. Транспьютерная технология позволила моделировать структуру мозга аппаратными средствами.

Параллельно с совершенствованием аппаратного обеспечения интеллектуальных систем проводятся обширные исследования в области технологии программирования, разрабатывается теоретические подходы к созданию специальных языков для представления знаний. В результате этих работ возникло новое направление – инженерия знаний.

Достижения в области нейросетевых технологий значительно повысили эффективность обработки информации и принятия решений в ситуациях, которые являются трудными как для компьютеров с традиционными средствами обработки информации, так и для человека.

Нейронные сети нашли применение в следующих областях техники:

- в промышленном производстве для управления процессами с неопределенными взаимодействиями объектов, которые участвуют в процессе, а также для диагностики работы машин, оценки качества продукции и др.;
- при передаче информации для сжатия и отображения данных, фильтрации сигналов, распознавания речи, преобразования текстовой информации и др.;
- в электронике для построения систем технического зрения, для размещения микросхем при проектировании платы, для нелинейного моделирования и др.;
- в финансовом деле для автоматического чтения документов, прогнозирования стоимости валюты, проведения общего финансового анализа;
- в автомобилестроении, космонавтике и аэронавтике для проектирования сложных систем управления, моделирования (имитации) процессов движения, построения алгоритмов управления движением.

Этот перечень не охватывает всего разнообразия решаемых задач. В настоящее время нейронные модели применяют практически во всех видах деятельности человека.

1.2 Различные подходы к построению систем ИИ

Существуют различные подходы к построению систем ИИ. Это разделение не является историческим, когда одно мнение постепенно сменяет другое, и различные подходы существуют и сейчас. Кроме того, поскольку по-настоящему полных систем ИИ в настоящее время нет, то нельзя сказать, что какой-то подход является правильным, а какой-то ошибочным.

Для начала кратко рассмотрим ***логический подход***. Основой для данного подхода служит Булева алгебра. Каждый программист знаком с нею и с логическими операторами. Свое дальнейшее развитие Булева алгебра получила в виде исчисления предикатов — в котором она расширена за счет введения предметных символов, отношений между ними, кванторов существования и всеобщности. Практически каждая система ИИ, построенная на логическом принципе, представляет собой машину доказательства теорем. При этом исходные данные хранятся в базе данных в виде аксиом, правила логического вывода как отношения между ними. Кроме того, каждая такая машина имеет блок генерации цели, и система вывода пытается доказать данную цель как теорему. Если цель доказана, то трассировка примененных правил позволяет получить цепочку действий, необходимых для реализации поставленной цели. Мощность такой системы определяется возможностями генератора целей и машиной доказательства теорем.

Конечно можно сказать, что выразительности алгебры высказываний не хватит для полноценной реализации ИИ, но стоит вспомнить, что основой всех существующих ЭВМ является бит — ячейка памяти, которая может принимать значения только 0 и 1. Таким образом было бы логично предположить, что все, что возможно реализовать на ЭВМ, можно было бы реализовать и в виде логики предикатов.

Добиться большей выразительности логическому подходу позволяет такое сравнительно новое направление, как ***нечеткая логика***. Основным ее отличием является то, что правдивость высказывания может принимать в ней кроме да/нет (1/0) еще и промежуточные значения — не знаю (0.5), пациент скорее жив, чем мертв (0.75), пациент скорее мертв, чем жив (0.25). Данный подход больше похож на мышление человека, поскольку он на вопросы редко отвечает только ***да*** или ***нет***.

Для большинства логических методов характерна большая трудоемкость, поскольку во время поиска доказательства возможен полный перебор вариантов. Поэтому данный подход требует эффективной реализации вычислительного процесса, и хорошая работа обычно гарантируется при сравнительно небольшом размере базы данных.

Под ***структурным подходом*** мы подразумеваем попытки построения ИИ путем моделирования структуры человеческого мозга. Одной из

первых таких попыток был перцептрон Френка Розенблатта. Основной моделируемой структурной единицей в перцептронах (как и в большинстве других вариантов моделирования мозга) является нейрон.

Позднее возникли и другие модели, которые в простонародье обычно известны под термином "нейронные сети" (НС). Эти модели различаются по строению отдельных нейронов, по топологии связей между ними и по алгоритмам обучения. Среди наиболее известных сейчас вариантов НС можно назвать НС с обратным распространением ошибки, сети Хопфилда, стохастические нейронные сети.

НС наиболее успешно применяются в задачах распознавания образов, в том числе сильно зашумленных, однако имеются и примеры успешного применения их для построения собственно систем ИИ.

Для моделей, построенных по мотивам человеческого мозга характерна не слишком большая выразительность, легкое распараллеливание алгоритмов, и связанная с этим высокая производительность параллельно реализованных НС. Также для таких сетей характерно одно свойство, которое очень сближает их с человеческим мозгом — нейронные сети работают даже при условии неполной информации об окружающей среде, то есть как и человек, они на вопросы могут отвечать не только "да" и "нет" но и "не знаю точно, но скорее да".

Довольно большое распространение получил и ***эволюционный подход***. При построении систем ИИ по данному подходу основное внимание уделяется построению начальной модели, и правилам, по которым она может изменяться (эволюционировать). Причем модель может быть составлена по самым различным методам, это может быть и НС и набор логических правил и любая другая модель. После этого мы включаем компьютер и он, на основании проверки моделей отбирает самые лучшие из них, на основании которых по самым различным правилам генерируются новые модели, из которых опять выбираются самые лучшие и т. д.

В принципе можно сказать, что эволюционных моделей как таковых не существует, существует только эволюционные алгоритмы обучения, но модели, полученные при эволюционном подходе имеют некоторые характерные особенности, что позволяет выделить их в отдельный класс.

Такими особенностями являются перенесение основной работы разработчика с построения модели на алгоритм ее модификации и то, что полученные модели практически не сопутствуют извлечению новых знаний о среде, окружающей систему ИИ, то есть она становится как бы вещью в себе.

Еще один широко используемый подход к построению систем ИИ — ***имитационный подход.*** Данный подход является классическим для кибернетики с одним из ее базовых понятий — "черным ящиком". Черный ящик — устройство, программный модуль или набор данных, информация о внутренней структуре и содержании которых отсутствуют полностью, но известны спецификации входных и выходных данных. Объект, поведение которого имитируется, как раз и представляет собой такой "черный ящик".

Нам не важно, что у него и у модели внутри и как он функционирует, главное, чтобы наша модель в аналогичных ситуациях вела себя точно так же.

Таким образом здесь моделируется другое свойство человека — способность копировать то, что делают другие, не вдаваясь в подробности, зачем это нужно. Зачастую эта способность экономит ему массу времени, особенно в начале его жизни.

Основным недостатком имитационного подхода также является низкая информационная способность большинства моделей, построенных с его помощью.

Заканчивая ознакомление с различными методами и подходами к построению систем ИИ, нужно отметить, что на практике очень четкой границы между ними нет. Очень часто встречаются смешанные системы, где часть работы выполняется по одному типу, а часть по другому.

1.3 Вспомогательные системы ИИ

Для того, чтобы человек сознательно воспринял информацию (для примера возьмем чертеж), она должна пройти довольно длительный цикл предварительной обработки. Вначале свет попадает в глаз. Пройдя через всю оптическую систему фотоны в конце концов попадают на сетчатку — слой светочувствительных клеток — палочек и колбочек.

Уже здесь — еще очень далеко от головного мозга, происходит первый этап обработки информации, поскольку, например, у млекопитающих, сразу за светочувствительными клетками обычно находятся два слоя нервных клеток, которые выполняют сравнительно несложную обработку.

Теперь информация поступает по зрительному нерву в головной мозг человека, в так называемые "зрительные бугры". То, что именно сюда приходит видеоинформация для дальнейшей обработки, показывают многочисленные опыты над людьми во время различных операций, в ходе которых производилась трепанация черепа.

Некоторые исследователи пошли дальше, и вживляли слепым людям целую матрицу электродов, напряжения на которых соответствовали освещенности соответствующих участков видеокамеры, размещенной на голове пациента. После операции, слепые начинали различать крупные фигуры (квадрат, треугольник, круг) и даже читать текст (при вживлении матрицы 10*10). Широкому распространению данного метода лечения слепоты препятствуют как недостаточно высокий наш технический уровень, так и чрезвычайно высокая опасность операций на открытом мозге. Такого рода опыты проводятся только попутно с операцией, вызванной другими причинами.

Далее зрительная информация поступает в отделы мозга, которые уже выделяют из нее отдельные составляющие — горизонтальные, вертикальные, диагональные линии, контуры, области светлого, темного, цвет-

ного. До этих пор мы можем без труда смоделировать работу мозга применяя различные графические фильтры. Постепенно образы становятся все более сложными и размытыми, но графический образ картины пройдет еще долгий путь, прежде чем достигнет уровня сознания. Причем на уровне сознания у нас будет не только зрительный образ, к нему примешаются еще и звуки, запахи (если картина представляет собой натюрморт) и вкусовые ощущения. Дальнейшие ассоциации каждый может додумать сам.

Смысл всего сказанного заключается в том, чтобы показать, что в системах ИИ имеются подсистемы, которые мы уже сейчас можем реализовать даже не зная о том, как они реализованы у человека. Причем можем это сделать не хуже, чем у прототипа, а зачастую и лучше. Например, искусственный глаз (а равно и блок первичной обработки видеоинформации, основанные на простейших фильтрах или сравнительно несложных устройствах) не устает, может видеть в любом диапазоне волн, легко заменяется на новый, видит при свете звезд.

Устройства обработки звука позволяют улавливать девиацию голоса человека в 1-2 Герца. Данное изменение частоты происходит при повышенном возбуждении вегетативной нервной системы, которое в свою очередь часто обусловлено волнением человека. На данном принципе основаны современные детекторы лжи, которые позволяют обнаружить с высокой вероятностью даже записанные на пленку много лет назад ложные высказывания.

Современные системы управления электродвигателем позволяют с высокой точностью держать заданные координаты даже при ударном изменении нагрузки. А ведь это примерно тоже, что держать на длинной палке баскетбольный мяч, по которому то слева, то справа кидают теннисные мячи. За одно и тоже время, компьютер произведет гораздо больше арифметических операций и с большей точностью, чем человек.

Антиблокировочная система (АБС) на автомобилях позволяет держать тормоза на грани заклинивания колеса, что дает наибольшее трение с дорогой, а это без АБС по силам только очень опытным водителям.

В принципе такие примеры, где техника оказывается ничуть не хуже человека, можно продолжать до бесконечности. Общий же смысл сказанного в том, что при конструировании ИИ, мы не связаны одним набором элементарных составляющих, как природа. В каждом конкретном случае желательно применять то, что даст самый большой эффект. В той области, где у человека господствуют рефлексы (чихание, быстрое напряжение мышцы, переваривание пищи, регулировка температуры), мы вообще можем применить жесткие системы управления, с раз и навсегда заданным алгоритмом функционирования. При этом вполне можно ожидать увеличения точности и уменьшение времени обучения их до нуля. При этом ядро нашей системы ИИ будет решать уже не настолько глобальные задачи.

Данный принцип разбиения задачи на подзадачи уже давно используется природой. К примеру, мы далеко не полностью используем все воз-

можности наших мышц в области разнообразия движений. Мы не можем заставить наши глаза смотреть в разные стороны, не говоря уже о том, чтобы делать это на разном уровне (левый глаз — влево-вверх, правый — вправо-вниз). При ходьбе мы часто используем далеко не оптимальный набор движений и далеко не все сочетания вариантов напряжения мышц мы опробуем. Попробуйте к примеру сделать волну животом. В принципе здесь нет ничего сложного, поскольку каждый пучок мышц пресса иннервируется отдельно, но если Вы этого не делали ранее, то получить необходимый результат будет не просто — в повседневной жизни это действие не нужно, а значит его нет и в "словаре движений", а на обучение необходимо определенное время. А по поводу оптимальности походки существуют расчеты, что если бы человек всегда рассчитывал оптимально траекторию движения, в которой существует более 200 степеней свобод, то он бы не ходил, а в основном бы только думал о том, как надо ходить.

На самом деле наша система управления построена по иерархическому принципу, когда задача распределяется между несколькими уровнями. Высший уровень нервной системы (связанный с большими полушариями мозга) ставит лишь общую задачу, скажем, переложить книгу на стол. Этот уровень вообще не контролирует действие отдельных двигательных единиц, направленных на решение поставленной задачи. Здесь уместна аналогия: командующий армией, ставя перед своими войсками некую общую задачу, отнюдь не предписывает каждому солдату и офицеру, что именно он должен делать в каждый момент операции.

Детализация построения движений у человека происходит на уровнях более низких, чем командный уровень коры больших полушарий. Более того, в некоторых случаях (когда мы отдергиваем руку, прикоснувшись к горячему предмету, даже не успев осознать ситуацию) все управление формируется на нижележащих уровнях, связанных с различными отделами спинного мозга.

В общем ситуация схожа с той, когда программист использует библиотеку подпрограмм. При этом ему не важно, какой алгоритм они используют, если программа работает нормально. А на написание своей библиотеки тратится драгоценное время. Кроме того, еще не известно, будет ли она работать так же хорошо.

Общий вывод состоит в том, что в настоящее время существуют методы, алгоритмы и устройства, которые позволяют нам довольно неплохо смоделировать нижние уровни человеческого интеллекта, причем совсем не обязательно на таком же физическом принципе.

1.4 Особенности и проблемы формирования знаний

Стремления ученых и специалистов в области автоматического управления сводятся к тому, чтобы построить системы, которые могли бы принимать решения и вырабатывать управление на основе сведений об

окружающей среде, а также с учетом опыта и мотиваций. Для этого система должна обладать необходимыми знаниями.

Знания обычно подразделяются на поверхностные и глубинные.

Поверхностные знания – это знания о видимых взаимосвязях между событиями и фактами в предметной области, например в представлении электрического звонка: "если нажать на кнопку, то звонок зазвенит".

Глубинные знания – это абстракции, аналогии, схемы, отображающие природу процессов, протекающих в предметной области, например знание принципиальной схемы звонка и принципа его работы.

По категориям знания разделяются на:

- концептуальные (понятийные) – это знания, отражающие мыслительные способности;
- фактуальные (предметные) – это сведения о качественных и количественных характеристиках, то есть информация и данные;
- алгоритмические (процедурные) – это знания о технологии, методах и программах, ориентированные на конкретную область применения.

Из этого перечисления видно, что концептуальные знания являются базой для построения других категорий знаний.

В обычных системах автоматизации функции деятельности реализуются на синтаксическом (формальном) уровне, который определяется информацией (данными) и алгоритмом управления процессом. При этом поведение системы определено априорно, а изменения производственной среды учитываются в известных пределах при создании алгоритма. Модель внешнего мира (производственной среды) в памяти системы отсутствует [2].

Для интеллектуализации процесса управления *функции управления* должны задаваться на семантико-прагматическом уровне, который позволяет построить процесс управления не на основе анализа базы данных, а на основе анализа базы знаний. При этом в системе управления создается новый стиль представления информации, в том числе и модели внешнего мира, при изменении которой формируется наиболее правильная реакция-ответ.

В решении какой-либо задачи исходные *данные* и *знания* поэтапно *трансформируются* [1].

Данные (факты), которые характеризуют объекты, процессы и явления предметной области, а также их свойства, трансформируются в следующей последовательности:

1) D1 – результаты измерений и наблюдений;
2) D2 – материальные носители информации (таблицы, протоколы, справочники);
3) D3 – модели (структуры, графики и т. п.);

4) D4 – данные в компьютере на языке описания данных;

5) D5 – базы данных в компьютере.

Знания – это метаданные, которые представляют собой закономерности предметной области, обычно основаны на данных теоретических исследований и данных, полученных эмпирическим путем. Другими словами, знания – это продукт мыслительной деятельности человека. При обработке на ЭВМ знания трансформируются аналогично данным в такой последовательности:

1) Z1 – знания в памяти человека как результат его мышления;

2) Z2 – материальные носители знаний, под которыми понимают учебники, методические пособия, инструкции, руководящие материалы и т. п.);

3) Z3 – поле знаний как условное описание или модель объектов предметной области, необходимое для перехода к формальному представлению знаний;

4) Z4 – знания, описанные на языках их представления, то есть семантические конструкции и сети, фреймы, формальные логические модели;

5) Z5 – база знаний на машинных носителях информации как основа интеллектуальной системы.

Приведенные здесь термины имеют следующие определения:

- семантическая конструкция – это смысловое обозначения понятия;
- семантическая сеть – это ориентированный граф, вершины которого – понятия, а дуги – отношения между понятиями;
- фрейм (каркас, рамка) – это структура знаний, абстрактный образ, представляющийся некоторым стереотипом.

При попытке формализовать человеческие знания возникает проблема, связанная с отсутствием аппарата для их описания. Кроме этого, некоторые описания не могут быть интерпретированы определенными знаниями. В связи с этим возникает проблема "нечетких знаний". Для решения такой проблемы американским математиком Лотфи Заде был предложен аппарат нечеткой (fuzzy) логики.

Центральная парадигма интеллектуальных технологий сегодня – это обработка знаний. Поэтому ***интеллектуальными системами*** называют системы, в которых ядро составляют *база знаний или модель предметной области, описанная на языке высокого уровня, приближенного к естественному*.

2 НЕЙРОН И АРХИТЕКТУРА НЕЙРОННЫХ СЕТЕЙ

2.1 Принципы функционирования биологического нейрона

Информацию об окружающей среде и внутреннем состоянии человек получает от сенсорной системы (анализатора) – части нервной системы, состоящей из периферических рецепторов (органов чувств), которые связаны с центральной нервной системой нервными волокнами.

В сенсорных органах энергия воздействия преобразуется в рецепторный потенциал, который трансформируется в импульсную активность нервной клетки – потенциал действия. Этот потенциал подается в сенсорный центр, клетки которого преобразуют (перекодируют) нервный сигнал. Сенсорные центры связаны с двигательными и ассоциативными отделами мозга. Идентифицируя свойства сигналов, они подают в двигательные центры импульсы, которые вызывают возбуждение или торможение (действие или бездействие).

Центральным звеном в этой системе является мозг, состоящий из нервных клеток – нейронов, количество которых превышает 100 миллиардов и каждый из которых имеет в среднем 10 тыс. связей с другими нейронами.

Биологический нейрон, схема которого приведена на рис. 1.1, имеет ядро клетки, тело клетки, дерево входов – дендриты и выход – аксон.

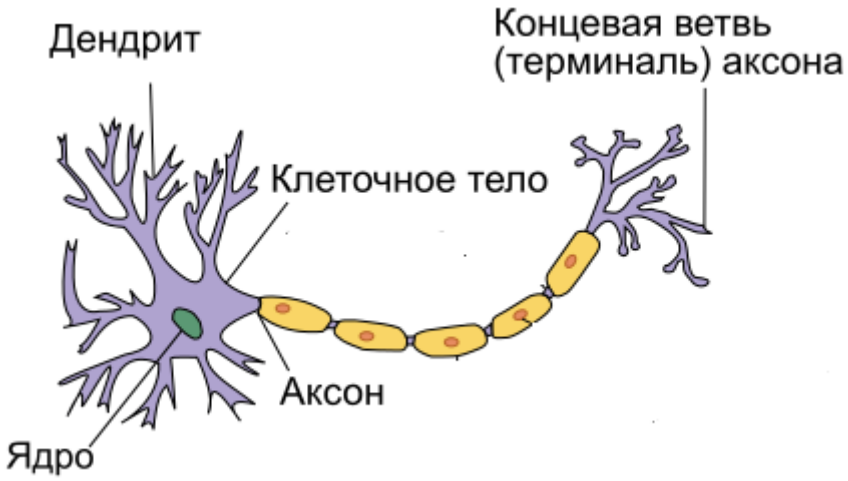


Рисунок 1.1 – Строение биологического нейрона

Длина дендритов может достигать одного миллиметра, а длина аксона – сотен миллиметров. Дендриты связаны с окончаниями нервных клеток узлами, которые называются синапсами. Передача импульса в синапсе представляет собой процесс освобождения определенного химического

вещества – нейротрансмиттера.

Нейротрансмиттер диффундирует через синаптическую щель, возбуждая или затормаживая в зависимости от типа синапса способность нейрона-приемника генерировать электрические импульсы. Синапс может настраиваться проходящими через него импульсами, то есть активность синапса определяется предисторией и действует как память.

Нейроны взаимодействуют посредством серии импульсов в течение нескольких миллисекунд. Информация передается путем частотной модуляции импульсов в диапазоне от единиц герц до нескольких сотен герц.

При этом за одну передачу пересылается несколько бит информации. Несмотря на сравнительно низкую скорость и малые объемы пересылок, сложные решения могут приниматься быстро в связи с огромным параллелизмом потоков.

Нейроны делятся на три группы:

- рецепторные, обеспечивающие преобразование воздействий окружающей среды в информацию для мозга;
- промежуточные, образующие центральную нервную систему и обеспечивающие ассоциативное преобразование информации и принятие решений;
- эффекторные, обеспечивающие связь центральной нервной системы с мышечной системой.

Все связи между нейронами физически параллельны. Один слой клеток проецируется на другой, волокна разветвляются и сливаются, то есть происходит дивергенция и конвергенция связей.

Практическая реализация такой системы связей представляет пока серьезную проблему, так как существующие в настоящее время нейропроцессоры имеют ограниченную систему команд и связей (нейросетевой базис). Поэтому на практике применяют упрощенные системы, реализованные либо на нейрокомпьютерах, либо путем программного моделирования на обычных компьютерах.

Для описания алгоритмов и устройств в нейроинформатике выработана специальная "схемотехника", в которой элементарные устройства – сумматоры, синапсы, нейроны и т.п. объединяются в сети, предназначенные для решения задач.

2.2 Модель нейрона

Математическая модель нейрона представляется в следующем виде:

$$a = f(wp + b), \tag{1.1}$$

где a – выходной сигнал, p – входной сигнал, w – вес входа, b – смещение аргумента функции нейрона, f (*) – функция активации нейрона.

Математической модели нейрона сопоставлена структурная схема, приведенная на рисунке 1.2.

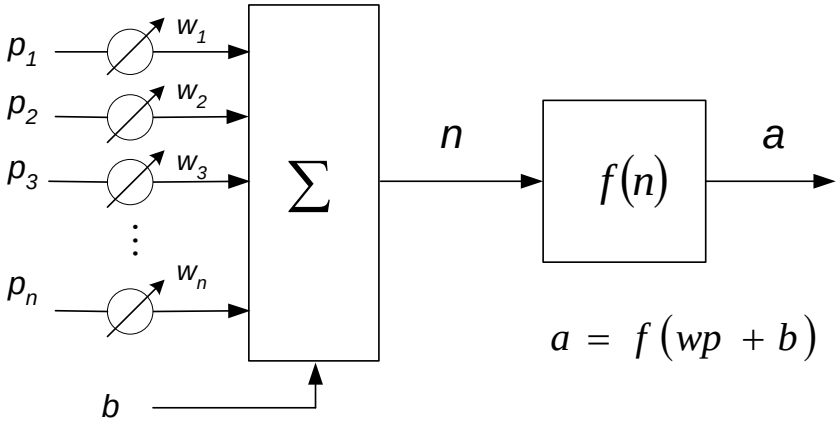
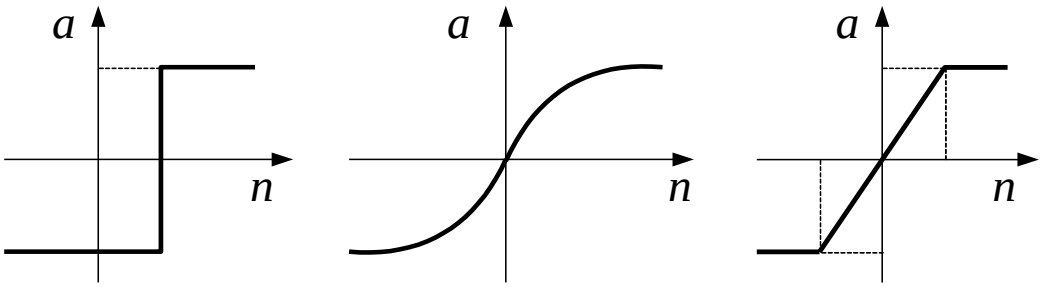


Рисунок 1.2 – Структурная схема модели нейрона

Блок функционального преобразования $f(n)$ реализует одну из функций активации, примеры которых приведены на рис. 1.3. Вид функции активации выбирается с учетом особенностей решаемых задач.



а б в

а – пороговая со смещением; б – сигмоидная; в – линейная с насыщением

Рисунок 1.3 – Виды некоторых функций активации нейрона

Недостатком пороговой и линейной функций (рис. 1.3, *а*, *в*) является резкий разрывный характер зависимости выходного сигнала от входного, что неестественно для условий многих процессов.

Компромиссом двух рассмотренных функций является сигмоидная функция (рис. 1.3, *б*), которую можно задать, например, выражением:

$$a = \frac{1}{1+e^{-n}}. \qquad (1.2)$$

Структуру нейрона, показанную на рисунке 1.2, принято изображать укрупненной схемой (рис. 1.4).

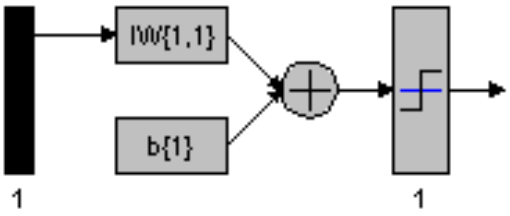


Рисунок 1.4 – Укрупненная структурная схема нейрона

Здесь вход нейрона изображается в виде темной вертикальной черты, под которой указывается количество элементов входа R. Вектор входа умножается на вектор-строку $\mathbf{W}$ длины R. Смещение b представляет собой скалярную величину с константой (1). Результат суммирования преобразуется функцией активации, условно изображенной на выходном блоке.

Параллельное включение нейронов образует слой нейронной сети.

2.3 Архитектура однослойных и многослойных нейронных сетей

Архитектура (топология) сети – это графическая иллюстрация соединения нейронов.

Различают однослойную и многослойную архитектуру. В однослойных сетях нейроны соединяются каждый с каждым или регулярно, причем каждый из нейронов может быть входным и выходным (рис. 1.5).

В многослойных сетях нейроны группируются в слои. Классической схемой связи является связь нейронов одного слоя с каждым нейроном другого слоя. Внутри слоя нейроны связей друг с другом не имеют (рис. 1.6). Если количество слоев в сети больше двух, внутренние слои называются скрытыми.

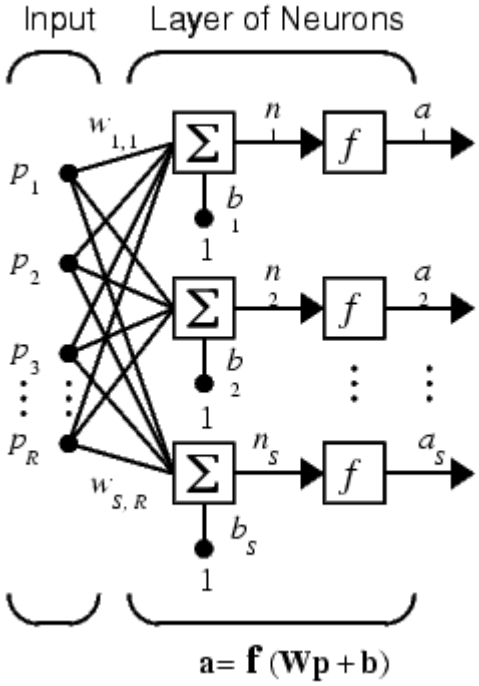


Рисунок 1.5 – Топология однослойной сети

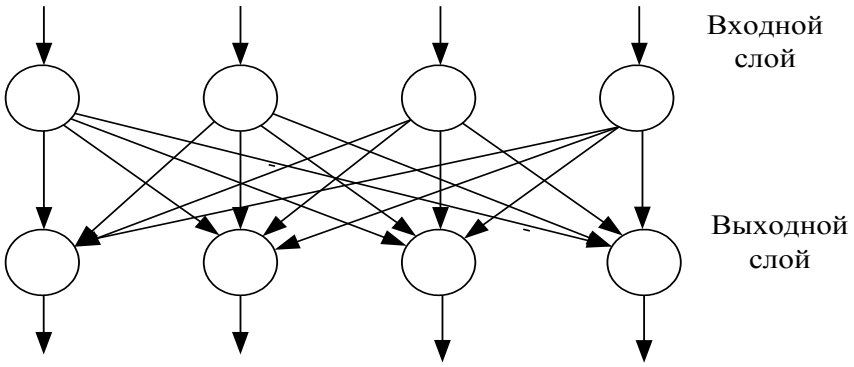


Рисунок 1.6 – Двухслойная сеть с прямыми связями

При конструировании сети необходимы следующие *данные*:

- размерности векторов входных и выходных сигналов;
- формулировка задачи;
- требуемая точность решения задачи.

На основании этих данных разработчик должен *выбрать*:

- тип топологии сети;
- число нейронов по слоям;
- функцию активации нейронов;
- способ задания и изменения коэффициентов синаптической связи.

По способу решения задач сети делятся на следующие *классы*:

- формируемые сети, которые проектируются для формализуемых задач, имеющих в нейросетевом базисе четкий алгоритм решения;

- сети с формируемой матрицей связей, которые проектируются для трудноформализуемых задач и в которых ассоциативная память создается по обучающей выборке путем задания матрицы коэффициентов;
- обучаемые сети, которые проектируются для неформализуемых задач и в которых ассоциативная память формируется постепенно по мере обучения.

Таким образом, топология сети непосредственно определяется особенностями решения задачи.

2.4 Создание, инициализация и адаптация нейронной сети

2.4.1 Формирование архитектуры нейронной сети

Архитектура сети состоит из описания того, сколько слоев имеет сеть, сколько нейронов в каждом слое, какой вид функции активации каждого слоя, а также как соединены слои друг с другом. Архитектура сети зависит от конкретной задачи, которую должна решать сеть.

Работа сети заключается в вычислении значений, устанавливаемых на выходе сети, по значениям входов и значениям желаемых результатов. Таким образом, конкретная задача определяет количество входов и выходов сети.

Входной слой может состоять из одного нейрона или нескольких нейронов, так как каждый нейрон может иметь R входов. Однако выходной слой, в котором каждый нейрон имеет один выход, должен состоять из количества нейронов, равного количеству выходов.

Таким образом, для разработчика сети важно решить задачу о количестве нейронов во входном слое и в скрытых слоях. Увеличение количества нейронов в скрытых слоях обеспечивает повышение мощности сети.

Выбор функции активации зависит от требуемого вида отображения. Для линейных отображений применяют линейные функции активации, для нелинейных отображений – нелинейные функции активации.

Смещение нейронов позволяет сформировать более сложные связи между входными и выходными сигналами – даже при нулевых входных сигналах путем обучения можно получить произвольную величину выходного сигнала.

В многослойных сетях обычно применяют нелинейные функции активации – сигмоидальную или гиперболического тангенса. Однако при применении этих функций, возможно, потребуется ограничить число выходов. Для исключения такой ситуации предпочтительно применять линейные функции активации.

В пакете прикладных программ Neural Network Toolbox для формирования сетей предусмотрены следующие функции (табл. 1.1).

Таблица 1.1 – Функции формирования архитектуры сети

Операторы	Назначение
`network`	Создание шаблона нейронной сети
`newlin`	Создание линейного слоя
`newff`	Создание сети прямой передачи
`newfftd`	Создание сети прямой передачи с запаздыванием
`newrb`	Создание радиально-базисной сети
`newlvg`	Создание нейронной сети для задач классификации

ППП `Neural Network Toolbox` использует специальный класс объектов `network objekt`. Эти объекты представляются в виде массивов записей, поля которых определяют их свойства, характеристики и параметры. С помощью массивов записей задается вычислительная модель нейронной сети, для которой используется имя `net`, являющееся также именем массива записей.

Свойства сети определяются следующими записями:

- `numInputs` – количество векторов входа сети;
- `inputs{i}.size` –количество элементов вектора входа;
- `numLayers` – количество слоев;
- `blasConnect` – одномерная булева матрица связности размером $N_l \times 1$ для указания наличия (1) или отсутствия (0) смещений в каждом слое;
- `inputConnect` – булева матрица связности размером $N_l \times N_i$, где N_l – количество слоев, N_i – количество входов, которая задает наличие (1) или отсутствие (0) веса при связывании слоя с входом;
- `layerConnect` – булева матрица связности для слоев размером $N_l \times N_l$, которая задает наличие (1) или отсутствие (0) связей для установки весов входов каждого слоя;
- `outputConnect` – одномерная матрица связности для выходов размера $1 \times N_l$, которая задает наличие или отсутствие выхода в слое;
- `targetConnect` – одномерная матрица связности для целей размера $1 \times N_l$, которая показывает наличие или отсутствие целевого выхода.

После формирования архитектуры задаются начальные значения весов и смещений, иными словами – сеть инициализируется.

2.4.2 Процедуры инициализация сети

Функции инициализации сети, приведенные в табл. 1.2, включены в описание свойств объекта `net`.

Таблица 1.2 – Функции инициализации сети

Оператор	Назначение
`init`	Инициализация нейронной сети
`initzero`	Инициализация нулевых значений весов и смещений
`initcon`	Инициализация равных смещений
`initlay`	Функция готовности сети к инициализации

Процедура инициализации может быть выполнена, например, с помощью метода `init` для объектов класса `network` следующим способом:

```
net = init(net);
```

Способ инициализации зависит от выбора параметров сети `net.initFcn` или `net.layers{i}.initFcn`, которые устанавливают ту или иную функцию инициализации.

Для сети с прямой передачей сигналов по умолчанию используется функция инициализации `initlay`, которая разрешает для каждого слоя сети применять собственную функцию инициализации, обычно `initwb` или `initnw`.

Функция `initwb` позволяет использовать собственные функции инициализации для каждой матрицы весов входа и вектора смещений путем задания параметров `net.inputWeights{i,j}.initFcn` и `net.biases{i}.initFcn`.

Функция `initnw` применяется для слоев, использующих сигмоидальные функции активации. Она генерирует начальные веса и смещения для слоя так, что активные области нейронов распределяются равномерно относительно области значений входа, тем самым обеспечивая ускорение процесса обучения.

2.4.3 Процедуры адаптации сети

Подготовка обучающей выборки. При решении прикладных задач обучение нейронных сетей производится по специальной выборке данных, которая должна быть достаточной и представительной.

Обучающий набор данных – это массив наблюдений, содержащий наиболее важные признаки изучаемого объекта.

Выбор признаков – это наиболее ответственный этап подготовки обучающей выборки, так как если важный признак не будет учтен, то процесс обучения может оказаться неэффективным.

Объем данных определяет эффективность обучения сети. Если выборочные значения взяты из ограниченного диапазона, то это может создать проблемы обучения за пределами диапазона.

Еще один важный аспект подготовки выборки – определение и соблюдение интервала выборочных значений. Сеть может не адаптироваться в процессе обучения, если имеются значительные пропуски в интервалах выборочных значений.

Существует примерная оценка количества данных, необходимых для обучения сети, – просто число связей умножается на 10. Однако с ростом числа признаков требуемое количество данных для обучения возрастает по нелинейному закону. Следует также учесть, что редко встречаются задачи, для которых требуется менее 100 наблюдений.

Начальная процедура обучения, которая может рассматриваться как обратная моделированию, производится после установки начальных значений весов и смещений. Такая процедура называется адаптацией сети.

Адаптация сети. В ППП Neural Network Toolbox реализовано два способа адаптации сети:

- последовательный способ, когда обучающая последовательность вектора входов и целевого вектора представляется в виде массива ячеек формата `cell : p = [`p_1`; `p_2` ];`
- групповой способ, когда входы и целевой вектор задаются в формате `double :p = [`$p_{1,1}$ $p_{1,2}$ $p_{1,3}$ …`; `$p_{2,1}$ $p_{2,2}$ $p_{2,3}$ …`].`

Процедура адаптации реализуется на основе метода `adapt`. При этом используется свойство `net.adaptFcn`, которое задает метод адаптации.

Для статических сетей по умолчанию применяется функция настройки режима `adaptwb`, которая позволяет выбирать произволные функции для настройки весов и смещений.

Функции настройки задаются свойствами:

- `net.inputWeights{i,j}.learnFcn;`
- `net.layerWeights{i,j}.learnFcn;`
- `net.biases[i,j].learnFcn`.

Для модификации весов и смещений необходимо также задать скорость настройки. При задании нулевой скорости адаптация не производится. Величина скорости определяется экспериментально. При задании малой скорости процесс адаптации затягивается во времени, а при большой – не достигается заданный малый уровень ошибки.

2.4.4 Процесс обучения сети

Процесс обучения сети – это *настройка весов* связей и алгоритма функционирования с учетом пяти элементов – входа сети, выхода сети, множества функций фактического результата, желаемого результата, функции ошибки.

Существует *четыре правила обучения*.

1 Правило коррекции по ошибке: задается желаемый результат, с которым сравнивается реальный выход и производится модификация весов в направлении уменьшения ошибки.

2 Правило Больцмана – это разновидность первого правила, применяемая для анализа вероятностных процессов.

3 Правило Хебба: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи должна возрастать.

4 Правило соревнования: «победитель получает всё», иными словами – наиболее активный нейрон получает наибольший вес синаптическрой связи, что упрощает алгоритм обучения.

Искусственный интеллект является моделью интеллекта человека. Поэтому задача обучения сводится к двум подзадачам:

1) создание программы «малого ребенка»;
2) задача воспитания «ребенка», то есть совершенствование алгоритма программы.

При решении прикладных задач с помощью нейронных сетей обычно используется метод обучения «с учителем», то есть формируется некоторый объем наблюдений и желаемых результатов, с помощью которых обучается сеть.

В процедуре обучения важно правильно выбрать количество слоев сети и число нейронов в каждом слое. Обычно эти величины определяются имеющимся объемом данных для обучения. После этого необходимо назначить первоначальные значения весов и смещений, которые в процессе обучения будут изменяться по критерию минимальной ошибки решения.

Критерием качества обучения чаще всего принимается минимальная сумма квадратов ошибок $\min \sum E^2$. Геометрической интерпретацией критерия является поверхность функции ошибок. Цель обучения состоит в том, чтобы найти глобальный минимум на поверхности функции ошибок.

В линейных сетях в качестве алгоритма поиска глобального минимума обычно применяется ***алгоритм обратного распространения ошиб-***

ки, разработанный специально для нейронных сетей.

Суть этого алгоритма заключается в том, что после определения ошибки вычисляется вектор градиента как функция весов и смещений. Этот вектор указывает направление кратчайшего пути по поверхности ошибки к точке минимума функции. Алгоритм действует итеративно и его шаги принято называть ***эпохами*** или ***циклами.***

В каждом цикле на вход сети подаются обучающие наблюдения, после чего вычисляются ошибки, а также градиенты, которые используются сетью для корректировки весов и смещений. Величина шага определяется параметром скорости настройки. На практике шаг выбирается пропорционально крутизне склона (градиенту). Процесс обучения заканчивается тогда, когда-либо величина ошибки достигла малого установленного значения, либо когда ошибка перестает уменьшаться.

В нелинейных сетях с нелинейными функциями активации достижение глобального минимума не гарантируется, так как поверхность функции ошибок может иметь ряд неблагоприятных для поиска минимума свойств: седловые точки, длинные узкие овраги, плоские участки и т. п.

Переобучение сети. Возможна ситуация, когда сеть подстроилась под определенную выборку и очень точно выполняет целевую функцию в определенном диапазоне, но недостаточно точно при отклонении от этого диапазона. Такая ситуация называется переобучением сети.

Для выявления эффекта переобучения используется механизм контрольной проверки. С этой целью часть обучающих наблюдений, не используемых в обучении, резервируется. Если ввод резервного массива увеличивает ошибку сети, то сеть является слишком мощной для этой задачи и необходимо уменьшить количество нейронов или слоев. Если после ввода резервного массива возникает ошибка, которая уменьшается до прежнего значения ошибки сети, то такая сеть работает нормально.

Для оценки качества работы сети анализируется также свойство обобщения – способность сети правильно работать как с хорошими, так и с плохими обучающими значениями.

2.5 Алгоритмы обучения нейронных сетей

2.5.1 Алгоритм обучения однослойной сети

Сеть может быть обучена решению различных прикладных задач – аппроксимации функций, идентификации параметров и управления объектами, распознавания образов, классификации объектов и др. Процесс обучения требует, чтобы при изменении информации на входах $\mathbf{p}$ информация на выходах $\mathbf{a}$ соответствовала желаемым значениям $\mathbf{t}$ с минимальной

ошибкой **e**.

Здесь и далее используются следующие обозначения:

- скалярные величины обозначены курсивными строчными буквами: a, b, c;
- векторные величины обозначены прямыми строчными полужирными буквами: **a, b, c**;
- матрицы – прямыми прописными полужирными буквами: **A, B, C.**

Для настройки весов и смещений в однослойных нейронных сетях ($M = 1$) наиболее часто применяется метод градиента. В этом случае выражение для функционала ошибки принимает вид:

$$J = \frac{1}{2}\sum_{q=1}^{Q}\sum_{i=1}^{S^1}\left(t_i^q - a_i^{qS^1}\right)^2 = \frac{1}{2}\sum_{q=1}^{Q}\sum_{i=1}^{S^1}\left(t_i^q - f\left(n_i^q\right)\right)^2, \quad (1.3)$$

где $f\left(n_i^q\right)$ – функция активации; $\left(n_i^q\right) = \sum_{j=0}^{R} w_{ij} p_j^q$ – сигнал на входе функции активации для i-го нейрона ($i = \overline{1, S}$); $\mathbf{p^q} = \left[p_i^q\right]$ – вектор входного сигнала; R – число элементов вектора входа; S – число нейронов в слое; w_{ij} – весовые коэффициенты сети.

Для учета смещения включим его вектор в состав матрицы входов и дополним этот вектор элементом, равным 1:

$$\mathbf{W} = \left[w_{ij}\right],\ i = \overline{1, S}, \quad j = \overline{1, R}. \quad (1.4)$$

Применяя правило дифференцирования сложной функции, вычислим градиент функционала ошибки:

$$\frac{\partial J}{\partial w_{ij}} = -\sum_{q=1}^{Q}\left(t_i^q - f\left(n_i^q\right)\right)\frac{\partial\left(f\left(n_i^q\right)\right)}{\partial w_{ij}} = -\sum_{q=1}^{Q}\left(t_i^q - f\left(n_i^q\right)\right)f'\left(n_i^q\right)p_j^q. \quad (1.5)$$

Введем обозначение:

$$\varepsilon = \Delta_i^q = \left(t_i^q - f\left(n_i^q\right)\right) = \left(t_i^q - a_i^q\right), \quad i = \overline{1, S} \quad (1.6)$$

и преобразуем функцию градиента следующим образом:

$$\frac{\partial J}{\partial w_{ij}} = -\sum_{q=1}^{Q} \Delta_i^q f'\left(n_i^q\right) p_j^q, \quad i = \overline{1,S} \tag{1.7}$$

Полученные выражения в применении для линейной сети упрощаются, так как в этом случае справедливо условие $f'\left(n_i^q\right) = 1$. Тогда функция градиента приобретает вид:

$$\frac{\partial J}{\partial w_{ij}} = \left(t_i^q - a_i^q\right) p_j^q, \quad i = \overline{1,S}, \quad j = \overline{0,R}. \tag{1.8}$$

Это выражение положено в основу алгоритма WH, применяемого для обучения линейных нейронных сетей.

2.5.2 Алгоритм обучения многослойной сети

Архитектура многослойной сети существенно зависит от решаемой задачи. Суммарное количество весов и смещений связано с длиной обучающей последовательности и может достигать нескольких тысяч. Поэтому для таких сетей были разработаны специальные методы обучения «без учителя». Наибольшее распространение получил метод обратного распространения ошибки.

Термин «обратное распространение» относится к процессу вычисления производных функционала ошибки по параметрам сети.

При описании многослойных сетевых структур применяются следующие обозначения:

- весовая матрица – $\mathbf{W}(\mathbf{t})$;
- элемент матрицы – $w_{ij}(t)$, где i – номер строки, j – номер столбца, t – время или итерация;
- вектор смещений – $\mathbf{b}(t)$;
- элемент вектора смещения – $b_i(t)$;
- вектор-столбец – $\mathbf{w_j}(t)$, вектор, соответствующий столбцу j матрицы $\mathbf{W}$;
- вектор-строка – $\mathbf{w}_i(t)$, вектор, соответствующий строке i матрицы $\mathbf{W}$.

При описании слоев нейронной сети применяются следующие индексы:

- верхний индекс из ***одного символа*** применяется для того, чтобы

указать принадлежность некоторого элемента слою. Например, вектор входа слоя 3 обозначается как $\mathbf{p}^3$;

- верхний индекс из ***двух символов*** применяется для того, чтобы указать источник сигнала (l) и пункт назначения (k) (он используется в матрицах входа и матрицах весов).

Для сетей с прямой передачей сигнала при обучении рассчитывается некоторый функционал, характеризующий качество обучения:

$$J = \frac{1}{2}\sum_{q=1}^{Q}\sum_{i=1}^{S^M}\left(t_i^q - a_i^{qS^M}\right)^2, \qquad (1.9)$$

где J – функционал; Q – объем выборки; M – число слоев сети; q – номер выборки; S^M – число нейронов выходного слоя; $\mathbf{a^q} = \left[a_i^{qM}\right]$ – вектор сигнала на выходе сети; $\mathbf{t^q} = \left[t_i^q\right]$ – вектор желаемых (целевых) значений сигналов на выходе сети для выборки с номером q.

Рассмотрим выражение для градиента критерия качества у выходного слоя M по весовым коэффициентам:

$$\frac{\partial J}{\partial w_{ij}^M} = \frac{\partial}{\partial w_{ij}^M}\left(\frac{1}{2}\sum_{q=1}^{Q}\sum_{k=1}^{S^M}\left(t_k^q - a_k^{qM}\right)^2\right) = -\sum_{q=1}^{Q}\sum_{k=1}^{S^M}\left(t_k^q - a_k^{qM}\right)\frac{\partial a_k^{qM}}{\partial w_{ij}^M}, \qquad (1.10)$$

где S^M – число нейронов в слое; a_k^{qM} - k-й элемент вектора выхода слоя M для элемента выборки с номером q.

Правило функционирования слоя M описывается выражением:

$$a_k^{qM} = f_M\left(\sum_{l=0}^{S^{M-1}} w_{kl}^M a_l^{q(M-1)}\right), \quad m = \overline{1, S^M}. \qquad (1.11)$$

Определим производную выходного сигнала по весовому коэффициенту:

$$\frac{\partial a_k^{qM}}{\partial w_{ij}^M} = \begin{cases} 0, & \text{если номер выхода отличается} \\ & \text{от номера нейрона в слое: } k \neq i; \\ f'\left(n_i^{qM}\right)\cdot a_j^{q(M-1)}, & \text{если } k = i \end{cases} \qquad (1.12)$$

$$i = \overline{1, S^M}, \quad j = \overline{0, S^{M-1}}$$

После подстановки (1.12) в (1.10) получим по объему выборки Q:

$$\frac{\partial J}{\partial w_{ij}^M} = -\sum_{q-1}^{Q} \left(t_i^q - a_i^{qM}\right) f_k'\left(n_i^{qM}\right) \cdot a_i^{q(M-1)}. \quad (1.13)$$

Обозначим:

$$\Delta_i^{qM} = \left(t_i^{qM} - a_i^{qM}\right) f_M'\left(n_i^{qM}\right), \quad i = 1,\ldots S^M. \quad (1.14)$$

Тогда получим:

$$\frac{\partial J}{\partial w_{ij}^M} = -\sum \Delta_i^{qM} a_j^{q(M-1)}; \; i = \overline{1, S^M}, \quad j = \overline{0, S^{M-1}}. \quad (1.15)$$

Для настройки весов w_{ij}^{M-1} слоя $M-1$ получим следующее соотношение:

$$\frac{\partial J}{\partial w_{ij}^{M-1}} = -\sum_{q=1}^{Q}\sum_{k=1}^{S^M} \left(t_k^q - a_k^{qM}\right) f_M'\left(n_k^{qM}\right) \frac{\partial\left(n_k^{qM}\right)}{\partial\left(a_i^{q(M-1)}\right)} \frac{\partial\left(a_k^{q(M-1)}\right)}{\partial w_{ij}^{M-1}} a_i^{q(M-1)} =$$

$$= -\sum_{q=1}^{Q}\sum_{k=1}^{S^M} \left(t_k^q - a_k^{qM}\right) f_M'\left(n_k^{qM}\right) w_k^M \frac{\partial\left(n_k^{qM}\right)}{\partial a_i^{q(M-1)}} f_{M-1}'\left(n_i^{q(M-1)}\right) a_j^{q(M-2)} =$$

$$= -\sum \Delta_i^{q(M-1)} a_j^{q(M-2)}, \quad (1.16)$$

где

$$\Delta_i^{q(M-1)} = \sum_{k=1}^{S^M} \left(t_k^q - a_k^{qM}\right) f_M'\left(n_k^{qM}\right) w_k^M f_{M-1}'\left(n_i^{q(M-1)}\right) =$$

$$= \left(\sum_{k=1}^{S^M} \Delta_k^{qM}\right) f_{M-1}'\left(n_i^{q(M-1)}\right), \quad i = \overline{1, S^{M-1}}. \quad (1.17)$$

Для слоев $M-2,\ M-3,...,1$ вычисление частных производных критерия J по элементам матриц весовых коэффициентов выполняется аналогично. В итоге получаем следующую формулу:

$$\frac{\partial J}{\partial w'_{ij}} = -\sum_{q=1}^{Q} \Delta_i^{q(r-1)} a_j^{q(r-1)} , \qquad (1.18)$$

где $r = \overline{1, M}$, $i = \overline{1, S'}$, $j = \overline{0, S^{r-1}}$,

$$\Delta_i^{qr} = \left(\sum_{k=1}^{S^{r+1}} \Delta_k^{q(r+1)} w_{ki}^{r+1} \right) f_r'\left(n_i^{qr}\right), \quad r = \overline{1, M-1},$$

$$\Delta_i^{qM} = \left(t_i^q - a_i^{qM}\right) f_M'\left(n_i^{qM}\right), \quad i = \overline{1, S^M} .$$

Выражение (1.18) представляет собой алгоритм настройки весовых коэффициентов нейронов, то есть обучения сети.

2.5.3 Алгоритм рекуррентной сети с обратной связью

Всякий целевой вектор можно рассматривать как набор характерных признаков некоторого объекта. Для такого набора в замкнутом гиперкубе можно построить систему линейных уравнений первого порядка, имеющую решения в вершинах этого гиперкуба.

Если создать сеть, положения равновесия которой совпадали бы с решениями указанной системы линейных уравнений, то при поступлении на вход сети некоторого входного набора ее выход можно было бы ассоциировать с соответствующим этому набору объектом. Такая сеть может рассматриваться как ассоциативная память.

Ассоциативными возможностями обладают рекуррентные сети Хопфилда. Свойство их рекурсии проявляется в том, что выходы сети подается обратно на входы. Структурная схема сети Хопфилда приведена на рис. 1.7.

Сеть состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон имеет связь с остальными нейронами посредством синапсов. Выходные сигналы образуются на аксонах. Нейроны принимают решения асинхронно, все связи симметричны, то есть соблюдается условие:

$$w_{ij} = w_{ji} . \tag{1.19}$$

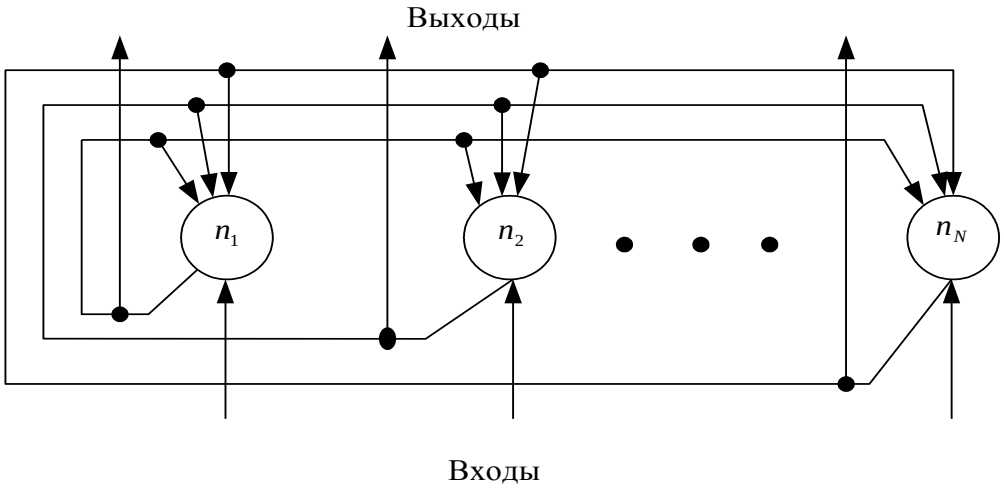


Рисунок 1.7 – Структурная схема сети Хопфилда

Рекуррентные нейронные сети обладают тем свойством, что за конечное число тактов времени они из произвольного начального состояния приходят в состояние устойчивого равновесия, называемое *аттрактором*. Вокруг аттрактора существует область притяжения. Если начальное состояние попадает в область притяжения, то система начинает двигаться к нему, что интерпретируется таким образом: ***при задании частичной информации система «вспоминает» эталон.***

Ряд аттракторов рассматривается как ассоциативная память. Количество таких аттракторов определяет объем ассоциативной памяти.

Состояния сети образуют некое подобие холмистой поверхности, в которой поведение сети подобно шарику, пущенному на эту поверхность – шарик двигается по склону в ближайший локальный минимум.

Каждая точка поверхности соответствует некоторому сочетанию активности нейронов, а высота этой точки характеризует энергию такого сочетания:

$$E = -\sum_{i \neq j} w_{ij} x_i x_j = \min . \tag{1.20}$$

Энергия интерпретируется как некоторая обобщенная энергия. Если в активных сочетаниях нейроны имеют большой положительный коэффициент связи, то такие сочетания будут характеризоваться низким уровнем вносимой нейроном энергии, и именно к ним будет стремиться сеть. И наоборот, нейроны с отрицательной связью при активации добавляют к энергии сети большую величину, и сеть будет стремиться избегать подоб-

ных состояний.

Важнейшее свойство сети заключается в том, что она может хранить и воспроизводить столько эталонов, сколько в ней имеется нейронов.

Архитектура сети Хопфилда показана на рисунке 1.8, а функция активации – на рисунке 1.9.

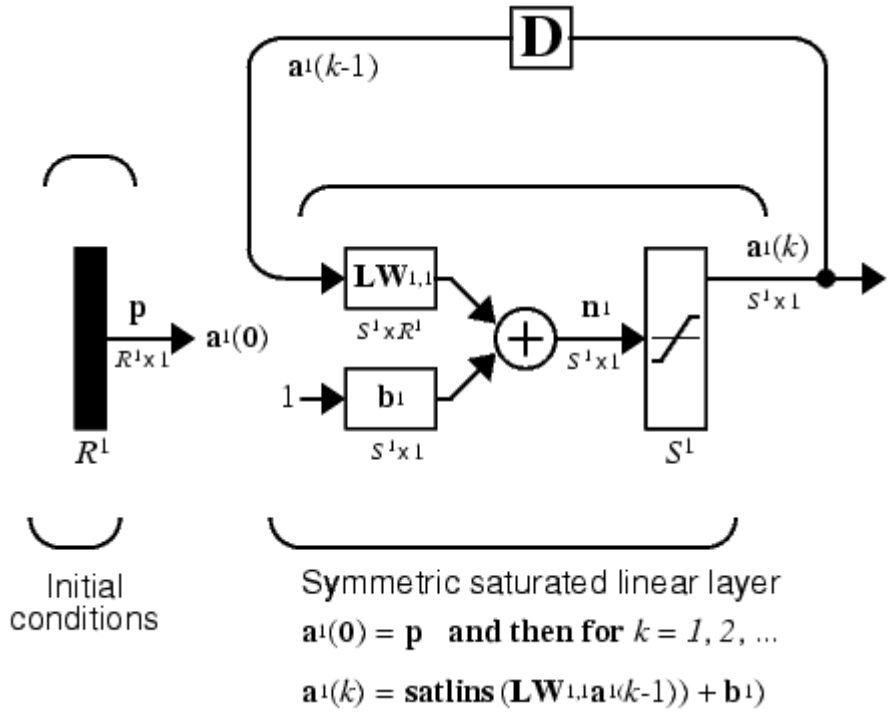


Рисунок 1.8 – Архитектура сети Хопфилда

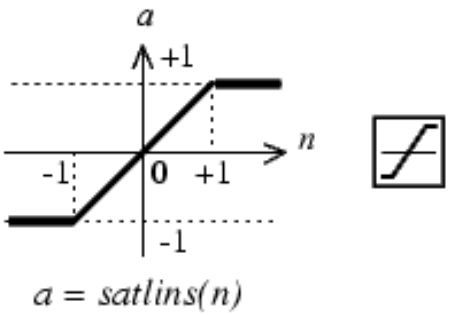


Рисунок 1.9 – Функция активации нейронов сети Хопфилда

Коэффициенты связей устанавливаются при обучении и могут принимать любые значения, как положительные, так и отрицательные.

При обучении сети используется правило Хебба, в соответствии с которым реализуется следующий алгоритм:

1 Вход **p** осуществляет начальную установку в соответствии с выражением:

$$\mathbf{a}^1(0) = \mathbf{p}. \tag{1.21}$$

2 В сети используется линейная функция активации с насыщением `satlins`, которая устанавливает выходы сети в соответствии с условиями:

$$a = satlins(n) = \begin{cases} -1, & \text{при} \quad n < -1; \\ n, & \text{при} \quad -1 \le n \le 1; \\ 1, & \text{при} \quad n > 1. \end{cases} \tag{1.22}$$

3 После задания начальных условий сеть генерирует выходные сигналы, которые по обратной связи поступают на вход. Этот процесс повторяется до тех пор, пока изменение выходного сигнала в текущей итерации не станет меньше установленного значения ошибки. Динамическая модель рекуррентного слоя сети Хопфилда описывается при этом следующим образом:

$$a^1(k) = saltins\left(\mathbf{LW}^{11}\mathbf{a}^1(k-1) + \mathbf{b}^1\right), \tag{1.23}$$

где $\mathbf{LW}^{11}$ – матрица весов динамической системы, $\mathbf{b}^1$ – вектор передачи единичного входа (смещения).

Можно полагать, что каждый вектор выхода, в конечном счете, установится в одной из точек равновесия, проявляющей аттрактные свойства.

3 РАЗРАБОТКА СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

3.1 Технология создания интеллектуальных систем

3.1.1 Особенности создания прикладных интеллектуальных систем

В настоящее время наиболее распространенными системами искусственного интеллекта являются экспертные системы [1, 4], которые вырабатывают советы:

- при управлении сложными диспетчерскими пультами, например распределением электроэнергии – Alarm Analyser [1987 г.];
- при постановке медицинских диагнозов – NEUREX [1988 г.];
- при диагностике электронных приборов – Intelligence Ware [1990 г.];
- при управлении перевозками – AIRPLAN [1983 г.];
- при формировании пакета инвестиций – RAD [1992 г.].

Приведенные экспертные системы на практике подтвердили свою эффективность.

Современные экспертные системы – это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и распространяющие их эмпирический опыт на консультирование менее квалифицированных пользователей.

Главные условия для применения экспертных систем [1]:

- дефицит квалифицированных кадров, способных принимать решения;
- выполнение небольшой задачи требует многочисленного коллектива "узких" специалистов;
- решение задачи требует анализа сложного набора условий, а обычный специалист не в состоянии это делать;
- большое расхождение между результатами решений хороших и плохих исполнителей;
- наличие конкурентов, которые лучше справляются с возникающими задачами.

Главное отличие интеллектуальных экспертных систем от других программных средств – это наличие *базы знаний*, представленной в такой форме, которая понятна специалистам предметной области и которая может быть изменена или дополнена в той же понятной форме. Так как интеллектуальные системы применяются обычно для решения сложных задач со слабо формализованными знаниями специалистов-практиков, то обработка знаний осуществляется с помощью специальных языков сверхвысо-

кого уровня, в которых операции со смысловой информацией превалируют над вычислительными операциями.

В настоящее время разработано ряд языков и моделей представления знаний, наибольшее распространение из которых получили следующие [1]:

- модели продукции – ROSIE, 1982 г.;
- семантические сети – NET, 1985 г.;
- фреймы – FRL, 1988 г.;
- язык логического программирования – ПРОЛОГ, 1990 г.;
- объектно-ориентированные языки – CLIPS 1980 г., CLOS, 1993 г.

На базе перечисленных выше моделей разработаны программы и реальные коммерческие экспертные системы.

Наибольшие трудности в разработке экспертных систем вызывает не процесс создания программы, а домашинный этап анализа знаний и проектирование начальной базы знаний. Этим занимается специальная наука – инженерия знаний.

3.1.2 Структура и классификация экспертных систем

Структура современной экспертной системы показана на рисунке 2.1 [1]. Реальные экспертные системы могут иметь более сложную структуру, однако они будут обязательно содержать блоки, изображенные на этом рисунке.

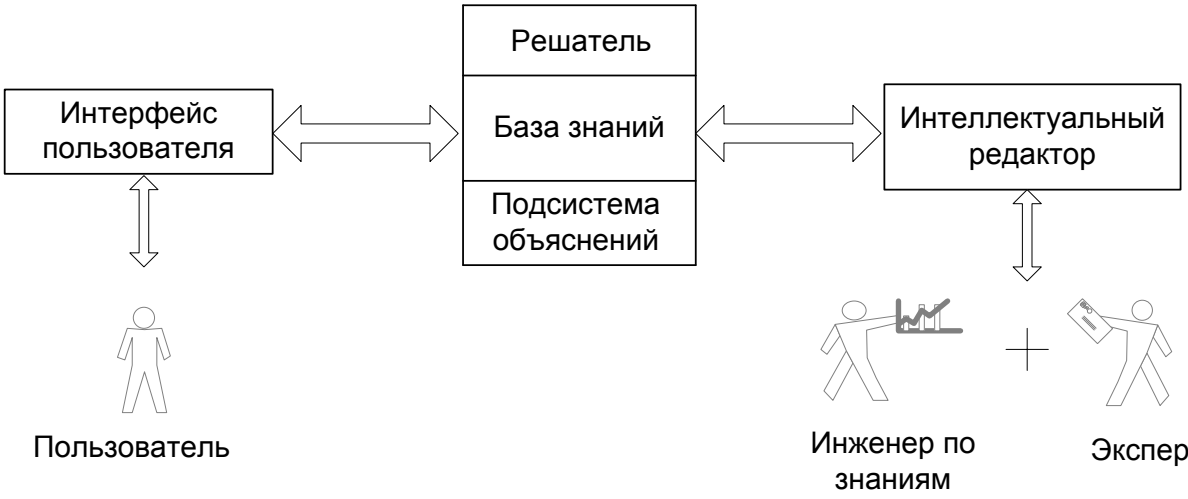


Рисунок 2.1 – Структурная схема экспертной системы

В экспертную систему входят следующие элементы:

Пользователь – специалист предметной области, нуждающийся в поддержке своей деятельности со стороны экспертной системы.

Интерфейс пользователя – комплекс программ, реализующих диа-

лог пользователя.

Решатель – программа, моделирующая ход рассуждений эксперта с помощью базы знаний.

База знаний – ядро экспертной системы, представленное машинным носителем на языке, понятном эксперту и пользователю.

Подсистема объяснений – программа, позволяющая пользователю получить ответ на вопрос о том, как была получена рекомендация (раскрытие трассы цепи умозаключений).

Инженер по знаниям – специалист в области искусственного интеллекта, выступающий в роли буфера между экспертом и базой знаний (синонимы: аналитик, инженер-интерпретатор, когнитолог).

Интеллектуальный редактор – программа, которая предоставляет инженеру по знаниям и эксперту возможности пополнять или корректировать базу знаний в диалоговом режиме.

Процесс функционирования экспертной системы можно представить так: пользователь с помощью интерфейса посылает запрос на необходимую информацию; решатель, используя базу знаний, генерирует пользователю рекомендацию, объясняя ход своих рассуждений.

Экспертные системы обычно классифицируют по виду решаемой задачи:

- ***интерпретация*** данных, например определение основных свойств личности по результатам тестирования;
- ***диагностика*** – оценка качества функционирования биологических, технических и иных систем;
- ***мониторинг*** – непрерывная интерпретация данных в реальном масштабе времени, например с целью предотвращения аварийных ситуаций;
- ***проектирование*** – получение четкого структурного описания знаний об объекте, формирование обоснований при принятия решений, например при синтезе электрических цепей;
- ***прогнозирование*** – предсказание последствий событий или явлений, например при оценке будущего урожая, а также экономические прогнозы;
- ***планирование*** – нахождение плана действий при проведении работ, например планирование производственных заказов;
- ***обучение*** – использование компьютера для обучения по какой-либо дисциплине, когда с помощью экспертной системы диагностируются характерные ошибки и находятся средства их ликвидации;
- ***управление*** – нахождение наилучшей стратегии управления сложными системами в соответствии с заданными свойствами;
- ***поддержка принятия решений*** – подача такой информации ли-

цу, принимающему решения, которая бы облегчала выбор лучшего решения среди множества возможных вариантов решений.

В общем случае все системы делятся на *системы, решающие задачи анализа*, и *системы, решающие задачи синтеза*.

3.1.3 Этапы создания экспертной системы

Уровень и численность разработчиков экспертной системы определяются поставленной задачей. Обычно численность группы составляет от четырех до 10 человек.

При создании базы знаний от специалистов требуются [1]:

- умение реферировать и аннотировать тексты;
- владение методами кластерного и факторного анализа;
- знание аппарата математической логики;
- умение формального представления знаний специальными языками.

Процесс разработки экспертной системы практически для любой предметной области можно разделить на шесть этапов [1]:

1 Выбор проблемы. Этот этап включает в себя определение предметной области, поиск эксперта, формирование коллектива разработчиков, анализ расходов и прибылей, подготовка плана работ.

2 Разработка прототипа экспертной системы. В качестве прототипа выступает усеченная версия экспертной системы с несколькими десятками правил, фреймов или примеров. При разработке прототипа производится компоновка базы знаний и её структурирование (разрабатывается поле знаний). Затем производится формализация знаний и составляется программа-прототип, которая проходит тестирование и оценивается экспертом.

3 Доработка прототипа. Результаты тестирования и оценки прототипа позволяют выявить то, что необходимо включить в последующую доработку системы. Обычно система дополняется недостающими знаниями, которые увеличивают количество правил для анализа отдельных случаев.

4 Оценка экспертной системы. Оценка проводится с целью определения точности работы системы и её полезности. Оценка производится по следующим критериям: удобство пользовательского интерфейса; качество решений, предлагаемых системой; наличие тупиковых ситуаций, чувствительность к изменению условий, производительность и т. п.

5 Стыковка системы. На этом этапе производится согласование работы системы с коммуникационной средой, обучение персонала, который будет работать с системой, обучение персонала, который будет обслужи-

вать систему.

6 Поддержка экспертной системы. Так как предметная область неизбежно изменяется с течением времени, то эти изменения нужно отражать в базе знаний системы.

3.2 Теоретические основы инженерии знаний

3.2.1 Описание основных понятий

Условное неформальное описание основных понятий и связей между понятиями предметной области представляется полем знаний [1].

Поле знаний (Pz) – это первый шаг к формализации знаний путем моделирования терминов и знаков конкретной науки или повседневного языка специалиста, в результате чего создается новый язык, например язык L.

Так как стандарта этого языка не существует, то каждый инженер по знаниям должен создавать его сам, придерживаясь следующих рекомендаций.

1 Язык должен содержать строгие определения понятий, быть ближе к языку математики.
2 В языке нежелательно использование терминов других наук, так как это может вызвать недоразумение.
3 Желательно применение символов либо графических объектов (схем, рисунков, пиктограмм и т. п.).
4 Учитывая необходимую последующую формализацию языка L, следует конструировать этот язык на основе семиотической модели.

Прикладная семиотика возникла в XVIII веке в результате работ Г. Ламберта. Семиотика включает в себя синтаксис (совокупность правил построения языка), семантику (отношения между знаками языка и реальностью) и прагматику (отношения между знаками языка и их пользователями).

Поле знаний может быть представлено некоторой семиотической моделью в виде графа, таблицы, диаграммы, формулы, текста (рис. 2.2).

Включение компонентов I и O в Pz обусловлено тем, что составляющие и структура этих интерфейсных компонентов имплицитно (неявно) присутствуют в модели репрезентации в памяти эксперта. Операциональная модель M может быть представлена как совокупность концептуальной структуры S_k, отражающей понятийную структуру предметной области, и функциональной структуры S_f, моделирующей схему рассуждений эксперта:

$$M = (S_k, S_f). \quad (2.1)$$

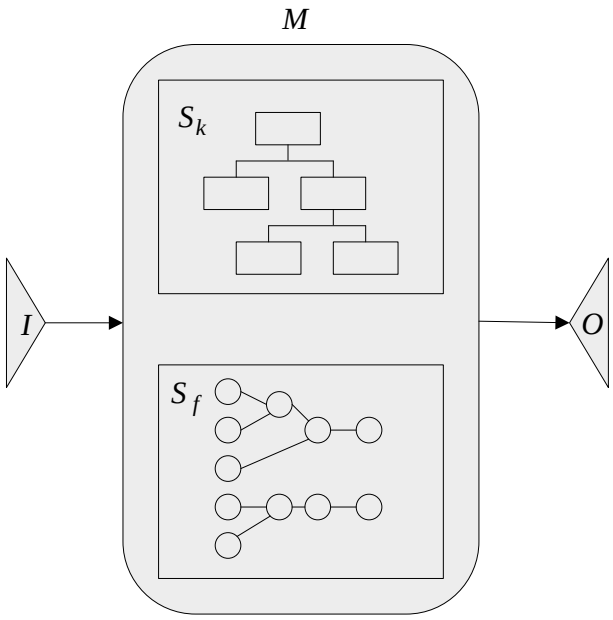


Рисунок 2.2 – Структура поля знаний

Концептуальная структура S_k выступает как статическая составляющая Pz, в то время как функциональная составляющая S_f представляет динамическую составляющую.

Формирование S_k основано на выявлении понятийной структуры предметной области. Структура S_f содержит понятия предметной области и моделирует основные функциональные связи между понятиями структуры S_k. Эти связи отражают модель принятия решения в выбранной предметной области.

Семантика. Семантика – это набор правил интерпретации предложений и формул языка. Семантика языка L зависит от особенностей предметной области.

Семантику поля знаний Pz можно рассматривать на двух уровнях. На первом уровне P_{Zg}^{i} существует семантическая модель знаний эксперта i о некоторой предметной области O_g.

На втором уровне любое поле знаний Pz является моделью некоторых знаний реальной действительности. Схема, отображающая отношения между реальной действительностью и полем знаний, можно, например, представить так, как показано на рисунке 2.3.

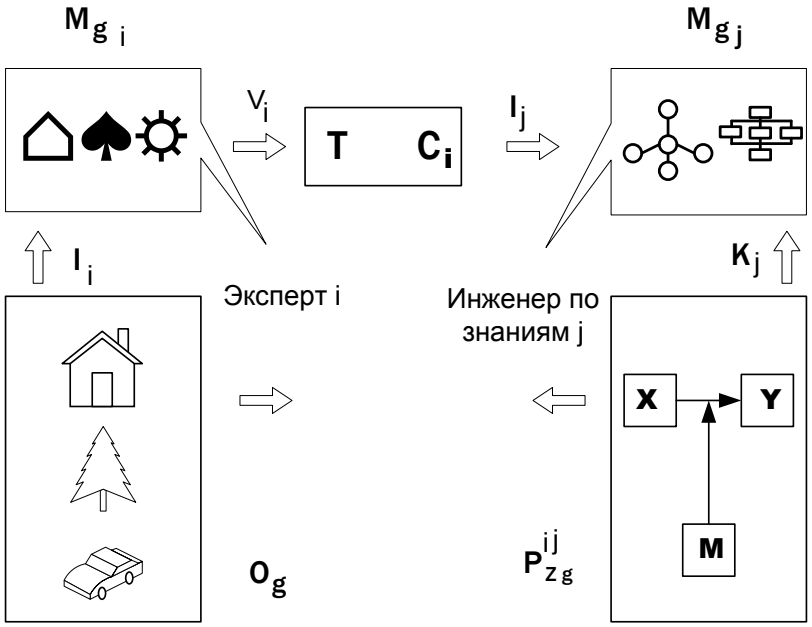


Рисунок 2.3 – «Испорченный телефон» при формировании поля знаний

Как видно из рисунка, поле знаний $P_{z_g}^{ij}$ – это результат, полученный после четырёх трансляций, если выражаться на языке информатики.

- Первая трансляция I_1 – это восприятие и интерпретация действительности O предметной области g экспертом i. В результате в памяти эксперта образуется модель M_{gi} как семантическая репрезентация действительности и его личного опыта по работе с ней.

- Вторая трансляция V_i – это вербализация опыта i-го эксперта, когда он пытается объяснить свои рассуждения и передать свои знания инженеру по знаниям. В результате V_i образуется либо текст T_i, либо речевое сообщение C_i.

- Третья трансляция I_j – это восприятие и интерпретация сообщений T_i или C_i инженером по знаниям j. В результате в памяти инженера по знаниям образуется модель мира $M_{g\,j}$.

- Четвертая трансляция K_j – это кодирование и вербализация модели $M_{g\,j}$ в форме поля знаний $P_{z_g}^{ij}$.

Схема напоминает игру в "испорченный телефон" – стоит заменить, например, инженера по знаниям и получится совсем другая картина. Поэтому при построении экспертной системы необходимо стремиться к максимальному соответствию $M_{g\,j}$ и $P_{z_g}^{ij}$.

Прагматика. В качестве прагматической составляющей семиотической модели следует рассматривать технологии проведения структурного анализа предметной области, то есть от хаотических черновиков знаний перейти к стройной и ясной модели.

Структурирование знаний производится с применением иерархического подхода как методологического приема расчленения системы знаний на уровни. На высшем уровне иерархии располагаются наименее детализованные представления, отражающие только самые общие черты и особенности проектируемой системы. На следующих уровнях степень подробности структуры возрастает, и отдельные представления отображаются блоками и модулями.

На каждом уровне вводятся свои представления о системе и элементах. Элемент k-го уровня является системой для уровня $k-1$. Продвижение от уровня к уровню имеет строгую направленность проектирования – сверху вниз или снизу вверх. При этом нисходящая стратегия называется *дедуктивной* и предполагает декомпозицию объектов, а восходящая называется *индуктивной* и предполагает синтез с постепенным обобщением понятий. На рисунке 2.4 показан пример дуальной концепции при проектировании предметной области S_k для экспертной системы помощи оператору энергетического блока.

Нисходящая концепция (top-down) декларирует движение от $n \Rightarrow n+1$, где n – уровень иерархии понятий предметной области. При этом стратегия STR_{td} детализации понятий представляется в виде:

$$STR_{td} : P_i^n \Rightarrow P_1^{n+1},...,P_{k_i}^{n+1}, \tag{2.2}$$

где i – номер порождающего концепта; k_i – число порождающих концептов.

Восходящая концепция (bottom-up) предписывает следующую стратегию STR_{bu} движения от $n \Rightarrow n-1$:

$$STR_{bu} : P_1^n,...,P_{k_i}^n \Rightarrow P_i^{n-1}. \tag{2.3}$$

Основанием для прекращения агрегирования и дезагрегирования является полное использование словаря терминов, которыми пользуется эксперт. При этом число уровней иерархии является фактором успешности структурирования понятий.

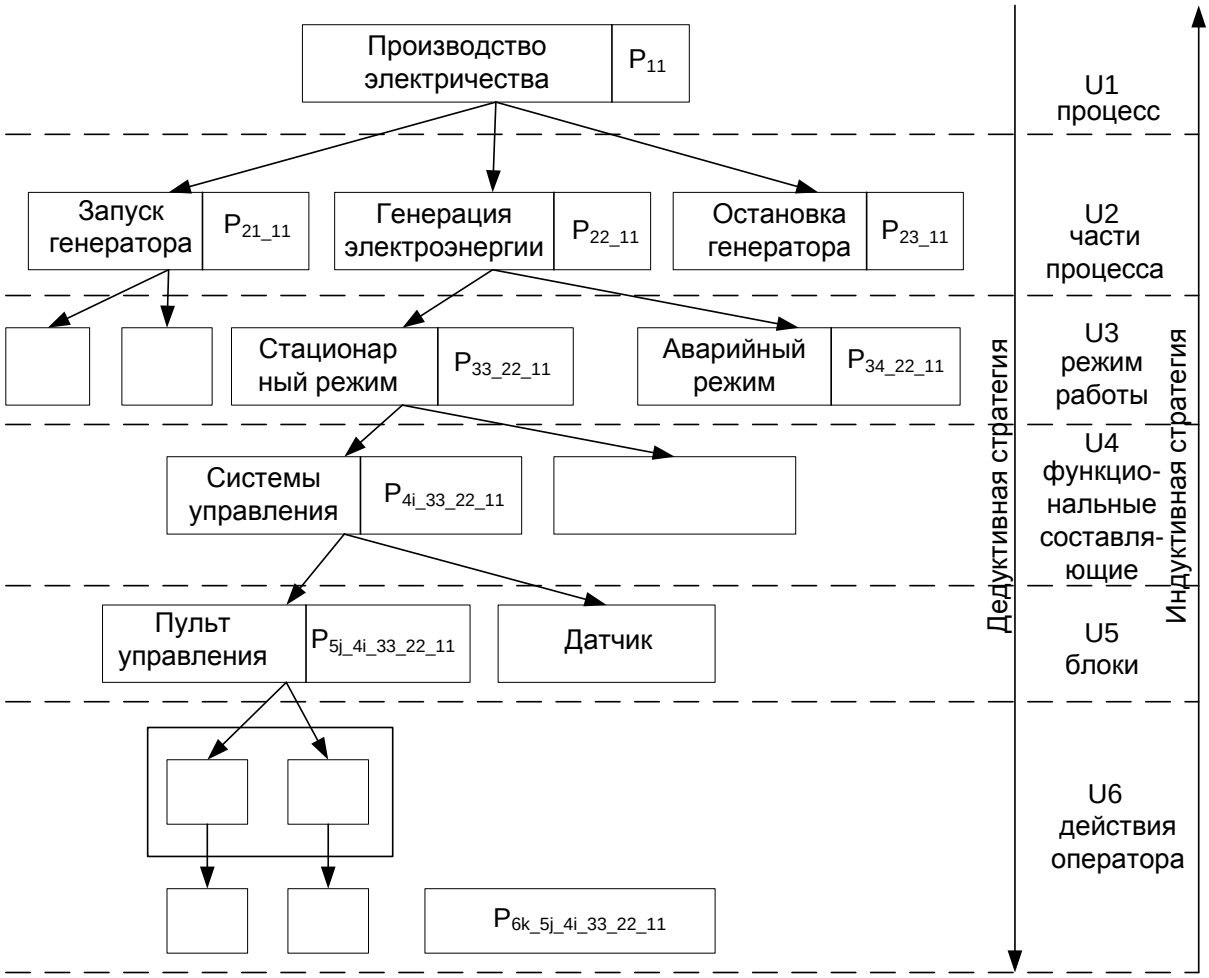


Рисунок 2.4 – Дуальная стратегия проектирования

3.2.2 Стратегии получения знаний

При формировании поля знаний ключевым вопросом является сам процесс получения знаний от эксперта и их преобразование инженером по знаниям. Для названия этого процесса в литературе по экспертным системам применяется несколько терминов [1] – приобретение, добыча, извлечение, получение, формирование знаний. В англоязычной специальной литературе используются, главным образом, два – *acquisition* (приобретение) и *elicitation* (выявление, извлечение, установление).

Термин "приобретение" знаний трактуется как способ построения базы знаний посредством диалога эксперта и специальной программы, в которую вложена структура поля знаний.

Термин "извлечение" знаний трактуется как процедура взаимодействия эксперта с источником знаний, в результате которой формируется структура поля знаний.

Проблемы обычно возникают на стадии извлечения знаний. Причиной этого являются следующие трудности:

- структура знаний не совпадает с методом их извлечения;
- неумение наладить контакт с экспертом;
- разнобой в терминологии;
- отсутствует целостная система – извлекаются фрагменты знаний;
- "картина мира" эксперта упрощена.

Термин "формирование" знаний закрепился за перспективной и развивающейся областью инженерии знаний, которая занимается разработкой моделей, методов и алгоритмов обучения. Она базируется на индуктивных методах формирования знаний, на автоматическом порождении гипотез, на обучающих выборках.

Таким образом, можно выделить три основные стратегии получения знаний при разработке экспертных систем [1]:

1 Применение ЭВМ и программного инструментария, то есть *приобретение* знаний.
2 Без использования вычислительной техники путем непосредственного контакта инженера по знаниям и эксперта, иначе *извлечение* знаний.
3 С использованием программ обучения и обучающей выборки с примерами принятия решений в предметной области, то есть *формирование* знаний.

3.2.3 Теоретические аспекты извлечения знаний

Основной проблемой инженерии знаний является процесс извлечения знаний. В этой процедуре выделяют три аспекта:

- А1 – психологический;
- А2 – лингвистический;
- А3 – гносеологический.

Из этих трех аспектов ведущим является психологический, поскольку он определяет успешность и эффективность общения (взаимодействия) эксперта с инженером по знаниям.

Общение, или коммуникация, – это процесс циркуляции информации и совместный поиск истины в новой, создаваемой этим процессом, информации.

Можно выделить четыре основных уровня общения:

1 Уровень манипулирования, когда один субъект рассматривает другого как средство или помеху своей деятельности.

2 Уровень "рефлексивной игры", когда один субъект учитывает "контропроект" другого субъекта, но не признает его ценности и стремится к реализации своего "проекта".

3 Уровень правового общения, когда субъекты признают право существования обоих проектов и пытаются согласовать их хотя бы внешне.

4 Уровень нравственного общения, когда субъекты внутренне принимают общий проект взаимной деятельности.

Четвертый высший уровень общения в информационном смысле наиболее эффективен и характеризует высокий профессионализм инженера по знаниям.

По данным П. П. Мицича [1] потери информации при разговорном общении характеризуются большими цифрами (рис. 2.5).

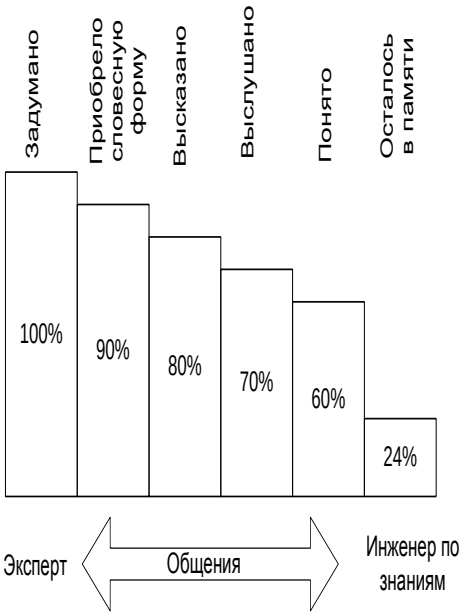


Рисунок 2.5 – Потери информации при разговорном общении

В процессе общения можно выделить такие структурные компоненты:

- участники общения (партнеры);
- средства общения (процедуры);
- предмет общения (знания).

Психологический аспект (A1). При извлечении знаний возникают три "слоя" психологических проблем:

A1 = {S11, S12, S13} = {контактный, процедурный, когнитивный}.

В контактном слое существуют следующие параметры, влияющие на результаты извлечения знаний:

S11 = {s11_i} = {пол, возраст, личность, темперамент, мотивация}.

В процедурном слое выявлены следующие профессиональные параметры, влияющие на результаты извлечения знаний:

S12 = {s12_i} = {ситуация общения (место, время, продолжительность); оборудование (вспомогательные средства), профессиональные приемы (темп, стиль, методы)}.

Когнитивные науки (психология, нейрофизиология, эргономика, инженерия знаний) исследуют познавательные процессы человека с позиций возможности их моделирования.

Основными факторами, влияющими на когнитивную адекватность, будут:

S13 = {s13_i} = {когнитивный стиль (индивидуальные свойства человека при организации своей познавательной деятельности), семантическая репрезентативность поля знаний и концептуальной модели}.

Лингвистический аспект (А2). Этот аспект касается исследований языковых проблем. В инженерии знаний выделяют три слоя лингвистических проблем:

A2 = {S21, S22, S23} = {"общий код", понятийная структура, словарь пользователя}.

"Общий код" решает проблему языковых ножниц между профессиональной терминологией эксперта и обыденной речью инженера по знаниям:

S21 = {s21_i} = {общенаучная терминология, специальные понятия, элементы бытового языка, неологизмы, профессиональный сленг и др.}.

Понятийная структура предназначена для объединения отдельных терминов во фрагменты, фрагментов в сценарии и так далее, то есть для построения иерархической сети понятий.

Словарь пользователя необходим для создания "дружественного интерфейса" конечному пользователю, профессиональный уровень которого не может воспринимать специальный язык, используемый экспертом. Словарь пользователя составляется взамен непонятных конечному пользователю терминов специального языка.

Гносеологический аспект (А3). Гносеология – это раздел философии, связанный с теорией познания. Гносеологический аспект объединяет методологические проблемы получения знаний по цепочке: факт → эмпирический закон → теоретический закон. При этом основными критериями научности являются:

A3 = {S31, S32, S33} = {внутренняя согласованность, системность, объективность, историзм}.

3.2.4 Теоретические аспекты структурирования знаний

Стадия структурирования знаний является "узким местом" в разработке интеллектуальных систем. Методология структурирования близка к современной теории сложных систем, особенно к объектно-ориентированному анализу.

Объектно-ориентированный подход, возникший как технология программирования, основан на следующих понятиях:

- объекты, классы как объекты, связанные общностью структуры и свойств, а также классификации как средства упорядочения знаний;
- инкапсуляция (образование капсулы вокруг инородного тела) как средство ограничения доступа;
- методы определения функций и отношений.

Наряду с объектно-ориентированным подходом при проектировании сложных систем используется также структурный (системный) подход, основанный на идее алгоритмической декомпозиции.

Объединение двух этих подходов позволяет создать базисную парадигму объектно-структурного подхода к построению поля знаний. В настоящее время этот обобщенный подход доведен до технологии и программной реализации.

Основные постулаты этой парадигмы сводятся к следующему [1]:

1 Системная взаимосвязь между понятиями.
2 Абстрагирование понятий, выявление их существенных отличий.
3 Иерархия, то есть ранжирование системы абстракций.
4 Типизация – выделение классов понятий и их наследственных свойств.
5 Модульность – разбиение задачи на локальные подзадачи, создание горизонтальных и вертикальных сечений предметной области.
6 Наглядность и простота представления структуры.

На основе объектно-структурного подхода разработан алгоритм *объектно-структурного анализа* предметной области, который позволяет оптимизировать и упорядочить достаточно размытые процедуры структурирования знаний.

В объектно-структурном анализе подразумевается дезагрегация программного обеспечения на восемь (как правило) страт или слоев (табл. 2.1). Здесь каждый слой отражает назначение знаний и их содержательное наполнение

Таблица 2.1 – Стратификация знаний предметной области

Номер слоя	Назначение	Содержание
s_1	ЗАЧЕМ-знания	Стратегический анализ: назначение и функции системы
s_2	КТО-знания	Организационный анализ: коллектив разработчиков системы
s_3	ЧТО-знания	Концептуальный анализ: понятийная структура
s_4	КАК-знания	Функциональный анализ: гипотезы и модели принятия решений
s_5	ГДЕ-знания	Пространственный анализ: окружение, оборудование, коммуникации
s_6	КОГДА-знания	Временной анализ: временные параметры и ограничения
s_7	ПОЧЕМУ-знания	Причинно-следственный анализ: формирование подсистемы объяснений
s_8	СКОЛЬКО-знания	Экономический анализ: ресурсы, затраты, прибыль, окупаемость

Алгоритм объектно-структурного анализа предусматривает создание матрицы понятий предметной области. Этот алгоритм можно представить двумя составляющими:

- A_1. Глобальный анализ (вертикальный) по всем строкам таблицы 2.1.
- A_2. Анализ страт (горизонтальный), в результате которого строятся многоуровневые структуры по каждой страте (область, проблема, задача, подзадача и т. д.).

В результате объектно-структурного анализа образуется подготовленная для программирования модель базы знаний.

3.3 Объектно-ориентированное программирование знаний

3.3.1 Технологии разработки программного обеспечения

Базисными понятиями в создании программного обеспечения являются *модель программы и модель системы* [1].

Обычно для создания модели применяются так называемые WorkBench-системы (выражение переводится как "станок для производства программного обеспечения").

Одним из подходов к разработке программного обеспечения является *методология DSSD* (Data Structured System Development), в основе кото-

рой лежит логическое соответствие между эвристической структурой программ и данными, которые обрабатываются этими программами. В DSSD применяются две концепции представления – диаграммы входов, которые дает представление о структуре системы, и линейные диаграммы, представляющие функциональное развитие системы.

Следующим подходом к созданию программ является *логическое моделирование*. Логическая модель проектируется в следующей последовательности:

1) с помощью диаграмм потока данных DFD (Data Flow Diagram) описывается природа предметной области;

2) выделяются списки элементов данных в информационных узлах;

3) производится проверка соответствия DFD структуре данных;

4) полученная информация сводится в двумерные таблицы и нормализуется;

5) диаграммы потока данных корректируются с учетом нормализации;

6) полученные модели разбиваются на процедурные единицы;

7) создается прототип системы на целевом языке.

Анализ литературных источников последних лет показывает, что новой ветвью в технологии программирования стала CASE-технология (Computer Aided Software Engineering).

Функциональные возможности CASE-технологии представлены на рисунке 2.6.

Рисунок 2.6 – Функциональные возможности CASE-технологии

Таким образом, CASE-технология поддерживает все основные этапы процесса создания программных систем.

3.3.2 Языки представления знаний для систем искусственного интеллекта

В процессе создания и внедрения интеллектуальных систем общения, систем машинного перевода, интеллектуальных систем синтеза программ, а также экспертных систем возникла проблема применения существовавших инструментальных средств, которые не имели достаточно развитых систем представления знаний.

Стиль программирования систем искусственного интеллекта существенно отличается от стиля программирования с применением обычных алгоритмических языков. При этом почти для каждой области искусственного интеллекта необходим собственный стиль программирования. В таблице 2.2 приведены некоторые отличия между обычными системами и системами искусственного интеллекта (ИИ).

Таблица 2.2 – Отличия систем ИИ от обычных программных систем [1]

Характеристика программирования	Программирование традиционное	Программирование в системах ИИ
Тип обработки	Числовая	Символьная
Методы	Алгоритм	Эвристический поиск
Задание шагов решения	Точное	Неявное
Искомое решение	Оптимальное	Удовлетворительное
Управление и данные	Разделены	Перемешаны
Знания	Точные	Неточные
Модификация	Редкая	Частая

Язык KRL (Knowledge Representation Language) был создан в 70-х годах XX века [4]. При создании языка была поставлена цель: построить систему, в которой знания были бы структурированы так, чтобы все они рассматривались не изолированно, а в совокупности с другими объектами-прототипами. Основанием для выбора такого подхода было то, что свойства объекта должны зависеть от цели решаемой задачи. Например, в задаче "сыграть музыкальную пьесу" наибольший интерес будут вызывать такие свойства музыкального инструмента, как качество звучания и настройка,

а в задаче "транспортировать музыкальный инструмент" более важны та-

кие свойства, как габариты и вес.

В основе процедурных свойств языка KRL лежат наиболее распространенные методы программирования, которые предполагают подключение процедур общего вида к классам объектов данных. Благодаря этому подклассы наследуют как процедуры, так и данные своего суперкласса.

Наследование процедур позволяет программировать в терминах *родовых операций*, а детали конкретизируются по-разному для объектов разных классов. При этом вырабатывается естественный стиль суждений – глобальные задачи, не требующие детализации, решаются на уровне общих процедур, а локальные задачи – с активизированием многочисленных процедур детализации.

Язык KRL способствовал развитию исследований в области теории представления знаний и созданию более совершенных систем.

Объектно-ориентированный стиль программирования идеально подходит для решения проблем, требующих детального представления объектов реального мира и динамических отношений между ними. В таких программах сложная система представляется структурами, инкапсулирующими и данные, и функции. В результате формируется модель реального мира, в которой каждый объект обладает собственным набором функций (операций).

Таким образом, вместо представления объекта пассивным набором данных объектно-ориентированная система позволяет объекту взаимодействовать с другими объектами, обмениваться с ними сообщениями. Интерфейс между объектами определяется соответствующими протоколами обмена, содержащими перечень функций. Помимо функций интерфейса объекты имеют собственные функции *родовых операций*, применимых к данному объекту.

Эти функциональные возможности были реализованы в языках LOOPS и FLAVORS, специально разработанных для задач искусственного интеллекта [4].

Отличительной чертой языка LOOPS является возможность задания *метаклассов,* членами которых являются другие классы. Это позволило создавать экземпляры классов, каким-то образом наследующих отдельные свойства метакласса, причем у каждого класса возможна своя модификация.

Язык FLAVORS поддерживает множественное наследование, то есть позволяет объектам иметь нескольких родителей. При этом графическое изображение отношений между разными объектами становится похожим не на дерево связей, а на решетку.

Примером множественного наследования может служить отображение окон в системе Windows. Окна как объекты образуют несколько под-

классов класса "окно" – окна с рамкой, без рамки, с заголовком, без заголовка и т. д. Если потребуется создать окно с рамкой и с заголовком, то новый класс будет наследником класса "окно" и независимым "близким родственником" уже существующих классов "окно с рамкой" и "окно с заголовком".

Механизмы наследственности реализуются и в других языках программирования – CLOS, CLIPS, Micro-PLANNER, OPS5, PROLOG, C++.

В языке C++ множественное наследование реализуется в виде виртуальных функций. Виртуальная функция, объявленная в классе X, это функция, которая будет перегружена (переопределена) в классе, производном от X. Ключевое слово *virtual* говорит компилятору, что программный код функции будет уточнен в производных классах. В связи с этим при применении языка C++ виртуальная функция должна иметь квалификатор *virtual* во всех классах иерархии до тех пор, пока в некотором производном классе не будет представлена ее конкретная реализация.

В чисто иерархической структуре классов, когда каждый производный класс имеет единственного "родителя", передача методов по наследству выполняется с помощью квалификаторов наследования *public* и *private*.

Таким образом, при применении языка C++ для создания экспертных систем необходимо продумывать организацию передачи свойств и поведения от классов родителей к производным классам с учетом особенностей механизма наследственности.

3.4 Интеллектуальные системы в среде Интернет

3.4.1 Системы интеллектуального поиска Autonomy и Webcompass

В системе поиска и обработки Интернет-ресурсов наметилась тенденция к использованию средств искусственного интеллекта [1].

Во второй половине 90-х годов были созданы первые версии агентных поисковых систем *Autonomy* (1998 г.) и *Webcompass* (1999 г.). Они были разработаны с целью обеспечить пользователя интегрированными средствами поиска отвечающей его интересам (релевантной) информации в сети Интернет. Однако принятые в этих системах проектные решения различны.

Первое различие состоит в том, что они ориентированы на разные категории пользователей.

Система *Autonomy* представляет собой совокупность программных

агентов для интеллектуального поиска и обработки информации, которые организованы скорее для новичков, чем для предметных специалистов. Пользователю предлагается парадигма "антропоморфного" общения и игровой подход для решения достаточно сложных задач.

Система *Webcompass* архитектурно тоже состоит из агентно-ориентированных компонентов. Но ориентирована эта система, прежде всего, на предметных специалистов, которые могут сформировать структурное описание своих интересов. Коммуникационный центр *Webcompass* предлагает пользователю парадигму многооконного интерфейса, характерную для современных приложений, и систему структурных редакторов для спецификации предметной области, поисковых запросов и управляющей информации.

Второе различие между системами *Autonomy* и *Webcompass* заключается в подходе к описанию предметной области поиска.

В первых версиях *Autonomy* применялась технология нейросетей с решением задач распознавания образов и обработки сигналов. При этом система формировала представления о том, какие должны быть релевантные документы на дальнейшем этапе поиска информации.

В системе *Webcompass* предметная область описывается с использованием таксономии (расположения по порядку) понятий, связанных между собой отношениями типа *is a, part of, has part, is a kind of* и других.

Третье различие между системами состоит в использовании разных средств спецификации запросов.

В системе *Autonomy* запрос на поиск представляется на естественном языке. Система анализирует текст и автоматически извлекает из него смысловое содержание, которое помещается в специальный конфигурационный файл. При этом внутреннее представление запроса фиксируется нейросетью.

Запрос в системе *Webcompass* формируется пользователем прямым описанием предметной области в виде таксономии понятий (ключевых слов и выражений). Для формирования запроса необходимо только промаркировать интересующие пользователя темы. На основании этих пометок система сама формирует запрос на поиск релевантной информации.

Недостатком таких систем является слабая обучаемость агентов.

3.4.2 Система MARRI

Система *MARRI* была создана в 1999 г. [1]. Для решения задач поиска релевантной информации система использует знания, представленные в виде онтологии, которая понимается как множество концептов и связей

между ними.

В основе системы заложено предположение разработчиков, что релевантные тексты состоят из значимых для предметной области предложений, которые содержат фрагменты, в какой-то степени соответствующие онтологии предметной области. Таким образом, задача сводится к тестированию информации.

Суть онтологического теста заключается в следующем. Сначала осуществляется морфологический и синтаксический анализ предложений текста, полученного от агентов сети, и строится его синтаксическое дерево. Затем определяется тип предложения и тип речевого акта, который это предложение отражает. Для дальнейшего анализа отбираются только утвердительные предложения, в которых присутствуют значимые для предметной области слова. В результате этого онтологический тест производит наложение анализируемого текста на онтологию предметной области.

Архитектура системы *MARRI* (рис. 2.7) представляет собой сеть из четырех агентов – агента пользователя (User Agent), агента-брокера (Broker Agent), агента сети (Connection Agent), и агента обработки текста (Text Processing Agent).

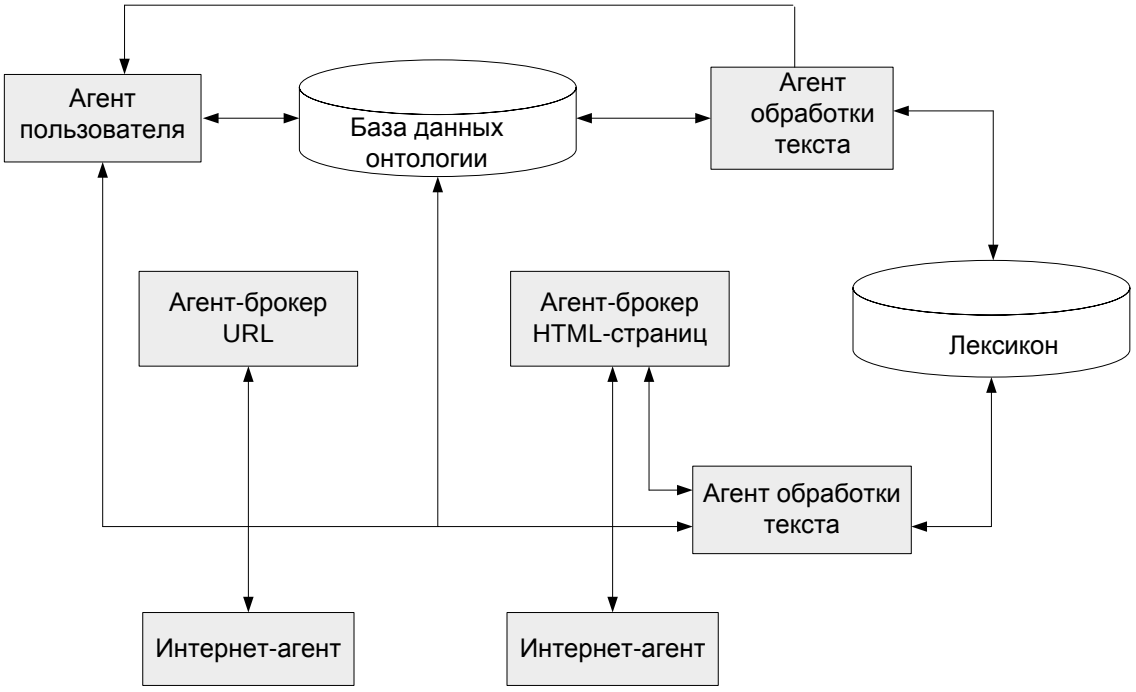


Рисунок 2.7 – Архитектура системы MARRI

Каждый из агентов обладает следующими свойствами:

- имеет автономную Java-программу с собственным сетевым адресом (URL);

- взаимодействует с другими агентами на языке ACL (Agent Communication Language) с HTTP-протоколом;
- является потребителем и поставщиком информации;
- взаимодействует с программными компонентами (базами данных, Web-броузерами и др.);
- обладает специальными знаниями и возможностями для выработки решения о релевантности информации определенной предметной области.

Агент пользователя выступает ассистентом при формулировке запросов и предоставляет результаты поимка в виде списка релевантных URL или Web-страниц. При выборе пользователем предметной области интерфейсный агент запрашивает соответствующую онтологию из базы данных онтологии.

Задачей агента-брокера является подключение к заданной URL Web-странице, её считывание и анализ.

Целью агента обработки текста является семантический анализ Web-страниц для проверки их релевантности на базе соответствующей онтологии. Результат обработки текста представляется в виде синтаксического дерева, которое должно отождествляться с определенным фрагментом онтологии.

В настоящее время система *MARRI* реализована для двух онтологий – область электронной коммерции и область Интернет-безопасности.

ЛИТЕРАТУРА

1 **Гаврилова, Т. А.** Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб. : Питер. 2001. – 384 с.

2 Системы управления гибким автоматизированным производством : учеб. пособие / под общ. ред. д-ра техн. наук, проф. А. А. Краснопрошиной. – К. : Вища шк. Головное изд-во, 1987. – 383 с.

3. **Медведев, В. С.** Нейронные сети.MATLAB 6 / В. С. Медведев, В. Г. Потёмкин, под общей ред. к.т.н. В. Г. Потёмкина. – М. : ДИАЛОГ-МИФИ, 2002. – 496 с.

4. **Джексон, Питер**. Введение в экспертные системы. : учеб. пос. : пер. с англ. / Питер Джексон. – М. : Вильямс, 2001. – 624 с.

Навчальне видання

КОМП'ЮТЕРНІ СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Конспект лекцій

(для студентів спеціальності
"Компьютерная инженерия»)

(*Російською мовою*)

Укладач СЕРДЮК Олександр Олександрович

Редактор О. М. Болкова

Комп'ютерна верстка О. П. Ордіна

119/2007. Підп. до друку . Формат 60 х 84/16.
Папір офсетний. Ум. друк. арк. 2,79. Обл.-вид. арк. 2,18.
Тираж прим. Зам. №

Видавець і виготівник
«Донбаська державна машинобудівна академія»
84313, м. Краматорськ, вул. Шкадінова, 72.
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру
серія ДК №1633 від 24.12.03.