РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

А.В. Суханов, З.В. Лященко

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Учебное пособие

Утверждено учебно-методическим советом университета

Ростов-на-Дону
2017

УДК 681.3(07) + 06

Рецензенты: начальник центра инновационных и интеллектуальных технологий С.М. Ковалев (Ростовский филиал ОАО «НИИАС»);
доктор технических наук, профессор М.А. Бутакова (РГУПС)

Суханов, А.В.

Интеллектуальные информационные системы: учеб. пособие / А.В. Суханов, З.В. Лященко; ФГБОУ ВО РГУПС. – Ростов н/Д, 2017. – 124 с.: ил. – Библиогр.: с. 122–123.

ISBN 978-5-88814-550-0

Настоящее пособие разработано в рамках дисциплин, связанных с искусственным интеллектом, преподаваемых на кафедре «Вычислительная техника и автоматизированные системы управления» ФГБОУ ВО «Ростовский государственный университет путей сообщения». Включает в себя основные понятия искусственного интеллекта, принципиальные отличия интеллектуальных от неинтеллектуальных систем, способы представления знаний, а также основные интеллектуальные методы и технологии, используемые при управлении информационными системами, в частности, искусственные нейронные сети, нечеткие и приближенные множества.

Учебное пособие предназначено для студентов высших учебных заведений по направлению подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.02 «Информационные системы и технологии» третьего и четвертого курсов обучения.

Одобрено к изданию кафедрой «Вычислительная техника и автоматизированные системы управления».

ISBN 978-5-88814-550-0

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Интеллект (от лат. *intellectus*) является характеристикой ума, рассудка, мыслительных способностей человека. Понятие «интеллект» в первую очередь связано с осознанной деятельностью. Соответственно, искусственный интеллект можно трактовать как использование мыслительных способностей машинами, их осознанный труд, способность принимать решения на основе опыта и рационального анализа внешних воздействий. В области искусственного интеллекта человек пытается не только понять природу интеллекта, но и создать искусственные сущности.

Идея создания искусственного интеллекта интересовала людей еще с древних времен. Еще Аристотель предлагал определить законы «Правильного мышления» и процессы неопровержимых рассуждений. В XIII веке Раймондом Луллием была совершена попытка реализации механизма, способного на основе логики получать новые истины на основе имеющихся понятий из различных областей деятельности – медицины, математики, права и др. Позднее, в XVII веке, Блезом Паскалем была разработана уже реальная машина, которая «производит эффект, который кажется более близким к мышлению по сравнению с любыми действиями животных». Первыми теоретическими работами в области искусственного интеллекта были работы Лейбница и Декарта, которые предложили универсальные языки классификации всех наук. Более близкой к нам по времени идеей является «аналитическая машина» Чарльза Бэббиджа, способная выполнять не просто арифметические операции, но и функционировать в соответствии с заранее подготовленными инструкциями.

Вышеперечисленные разработки относятся скорее к философскому аспекту, нежели к техническому. Действительное появление искусственного интеллекта как отдельной дисциплины относится к моменту появления ЭВМ, когда впервые возник вопрос: «могут ли машины мыслить?». Для ответа на него разработана целая дисциплина, которая в англоязычной литературе именуется как «*Computational intelligence*» (вычислительный интеллект), которая охватывает методы моделирования разумного вычисления:

- Искусственные нейронные сети;
- Нечеткая логика;
- Приближенные множества;
- Эволюционные алгоритмы;

и др.

Тематика настоящего пособия охватывает историю развития направлений искусственного интеллекта и основные понятия интеллектуальных систем, а также избранные популярные методы искусственного интеллекта, используемые при решении задач управления информационными

системами, начиная от традиционных методов, основанных на логике высказываний и логике предикатов, и заканчивая нейросетевыми методами и технологиями нечеткого вывода.

В **первой главе** изложена история возникновения дисциплины искусственного интеллекта, и представлены ключевые фигуры, стоявшие у истоков создания первых интеллектуальных систем. Глава содержит вводную информацию о терминах, целях и задачах искусственного интеллекта, а также раскрывает философские аспекты создания интеллектуальных систем. Во второй части главы рассматриваются основные понятия эвристических алгоритмов и экспертных систем, представлены прикладные интеллектуальные системы. В конце главы приводятся основные подходы к построению интеллектуальных систем, а также современный этап и перспективы развития искусственного интеллекта.

Во **второй главе** представлены базовые концепции, заложенные в период становления теории искусственного интеллекта для формализации взаимодействия между искусственными системами. Здесь речь идет о формальных системах, позволяющих исключить недостатки естественно-языкового общения при моделировании средств взаимодействия между людьми в современных информационных системах. В частности, приведено описание наиболее популярной формальной системы – исчисления предикатов первого порядка, – а также средства реализации человеческих рассуждений при использовании формальных систем – логики исчисления предикатов первого порядка. Рассмотрены основные понятия формальных систем и их конкретных интерпретаций – алгебраических. Даны простейшие примеры использования формальных систем.

В **третьей главе** представлено основополагающее определение знаний, которые отличают интеллектуальные системы от неинтеллектуальных. В главе приводятся базовые классификаторы знаний, а также свойства знаний, позволяющие отличить их от данных, с которыми работают традиционные информационные системы. Во второй части главы приводятся основные понятия экспертных систем, которые считаются наиболее наглядным типом интеллектуальных систем, основанных на знаниях.

Четвертая глава посвящена моделям представления знаний, на которые ссылаются классические работы по экспертным системам. В первой части главы представлено описание наиболее прикладной модели представления знаний – продукционной модели. Здесь приводятся основные характеристики модели, а также способы задания элементарных единиц представления знаний – продукций – и искусственных рассуждений на базе их использования. Во второй части изложены основные характеристики наглядных моделей представления знаний в виде сетей. Здесь описаны базовые понятия теории семантических сетей и систем фреймов, представлены примеры и раскрыты основные преимущества и недостатки таких моделей.

Начиная с **пятой главы** речь пойдет о современных методах искусственного интеллекта применительно к принятию решений при управлении информационными системами, содержащими НЕ-факторы. Пятая глава раскрывает теорию приближенных множеств, которая позволяет формализовать понятие неоднозначности и дает возможность оценки непохожести объектов. Здесь представлены метрики количественного описания степени детализации моделируемых объектов. В главе рассматриваются некоторые примеры, для которых использование средств теории приближенных множеств делает доступной дальнейшую обработку. В заключении главы представлены основные преимущества использования теории приближенных множеств для прикладных задач.

Шестая глава посвящена теме, являющейся фундаментальной в области искусственного ассоциативного запоминания. В главе описывается теория перцептронов, в простонародье именуемых искусственными нейронными сетями. В первой части главы представлена краткая справка о биологических нейронных сетях и об их элементарных частях – нейронах. Представлены первые идеи моделирования мозга, описаны простейшие алгоритмы обучения искусственных нейронных сетей. Во второй части главы представлены основные трудности, с которыми столкнулась данная теория, а также пути их решения. Приводится модель обучения нейронной сети, представленная Румельхартом, которая является базой современных прикладных нейросетевых систем. В главе также характеризуются основные преимущества искусственных нейронных сетей, в частности, свойство обобщения, которые позволяют их с полной уверенностью назвать интеллектуальными системами.

В **седьмой главе** изложена теория нечетких множеств, которая стала новой ветвью в развитии экспертных систем и искусственного мышления в целом. Глава описывает инструмент, позволяющий формализовать НЕ-фактор нечеткости и размытости информации, которая может быть получена информационной системой от субъективного источника. В главе представлены ключевые отличия теории нечетких множеств от классической теории множеств, раскрыты основные отличия меры нечеткости от меры вероятности. Описаны классические арифметические операции, расширенные на нечеткие величины, а также приведены характеристики специфических для нечетких множеств операции. В заключении главы представлен аппарат нечеткого вывода, который является в настоящее время популярным при построении экспертных систем, а также приведены примеры реализации нечеткого вывода.

ГЛАВА 1. ПОНЯТИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА. ИСТОРИЯ И НАПРАВЛЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

1.1 Возникновение искусственного интеллекта

Появление понятия «Искусственный интеллект» (ИИ) напрямую связано с началом третьей научно-технической революции (НТР), знаменующей замену интеллектуального труда человека машинным.

Стоит отметить, что первая научно-техническая революция характеризовалась переходом от физического труда человека к машинному, а также появлением автоматизированных средств передвижения, а именно железнодорожного транспорта в начале XIX века.

Для второй НТР, начавшейся в конце XIX века, характерной чертой является освоение электричества и появление двигателя внутреннего сгорания.

Началом третьей НТР, которая (по мнению авторов) продолжается до сих пор, принято считать появление первых электронно-вычислительных машин (ЭВМ) в середине XX века [1].

Ключевые фигуры, стоявшие у истоков создания ЭВМ и искусственного интеллекта – Джон Фон Нейман и Алан Тьюринг.

Джон Фон Нейман – профессор Принстонского института перспективных технологий в (США). Участвовал в разработке американской атомной бомбы. Основные результаты деятельности – разработка теоретической модели конечного автомата и идея управления ходом вычислений с помощью программы, хранимой в памяти машины.

Алан Тьюринг – великий английский математик, основатель информатики, дешифровщик (попросту, хакер), талантливый и веселый человек.

Алан Тьюринг является одним из основателей теории искусственного интеллекта, предложивший впервые абстрактную модель компьютера – машину Тьюринга. Такая машина является математической моделью, позволяющей раскрыть понятие алгоритма. Машина Тьюринга состоит из бесконечной ленты и каретки, которая движется вдоль этой ленты. При положении каретки на одной из ячеек ленты мы можем в зависимости от текущего значения, заданного на конечном алфавите, записывать новое значение и передвигать каретку влево или вправо.

Машина Тьюринга позволяет описать предельные возможности компьютеров, не вдаваясь в характеристики их производительности.

Вторым выдающимся вкладом Тьюринга в историю ИИ является создание так называемого теста Тьюринга (Рис. 1.1).

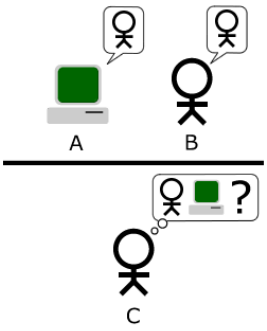


Рис. 1.1. Наглядное представление теста Тьюринга

Тест Тьюринга – тест вычислительной машины или программы на интеллектуальность. Тест заключается в следующем.

Эксперт *С* помещен в изолированную комнату. Компьютер *А* и человек *В* помещены за стенкой и отвечают на вопросы, задаваемые экспертом. Цель эксперимента – ввести эксперта в заблуждение, что ответы, данные компьютером – ответы человека. В этом случае будет сказано, что компьютер имеет интеллект и тест Тьюринга пройден.

Разработанный Тьюрингом тест на интеллектуальность вызвал бурю рассуждений и критики в научной среде [2]. Одним из известных критиков теории Тьюринга является американский философ Джон Сёрл, который является ярым сторонником того, что современные ЭВМ по своей сути хоть и оперируют символами по заданным правилам, но они не могут понять сути того, чем они оперируют и для чего, в связи с чем не обладают интеллектом. В подтверждение своей мысли он приводит эксперимент «Китайской комнаты» (Рис. 1.2). Предположим, что существует закрытая комната, в которой находится, допустим, кто-то из нас, кто говорит по-русски, но не знает китайского языка. Этому человеку дают карточки с китайскими иероглифами, с помощью которых он должен составить ответ на поставленный ему вопрос (естественно, вопрос на китайском языке). Наш герой не знает китайского, однако ему дают книгу с названием «Что делать, если судьба тебе подсунула карточки с китайскими символами?». Он составляет ответы согласно инструкциям и отвечает правильно. Но он делает это без единого проблеска интеллекта, так как все еще не понимает, о чем идет речь в этих карточках. Другие комментарии, возникшие в процессе интерпретации теста Тьюринга, возникли у английского ученого Роджера Пенроуза, который размышлял, приемлемо ли использовать машину, которая прошла тест Тьюринга, для удовлетворения собственных потребностей либо такое рабство заслуживает порицания? И таких философских обсуждений, особенно в конце прошлого века, породилось не мало.

Рис. 1.2. Наглядное представление мысленного эксперимента «Китайская комната»

Основным философским вопросом создания интеллектуальных систем в первую очередь является вопрос самой возможности создания таких систем. Отрицательный ответ на данный вопрос перечеркивает целесообразность всей дисциплины, поэтому будем считать его априорно положительным. Ведь, действительно, еще в библейских писаниях говорится, что Бог создал человека «по подобию и образу своему». Из этого становится очевидным, что человек может также создать что-то, что будет подобно ему. Другим неформальным доказательством возможности создания искусственного интеллекта является тот факт, что еще совсем недавно казалось, что вершиной интеллектуальной деятельности является игра в шашки и шахматы, распознавание зрительных и звуковых образов, разработка новых технических решений на базе синтеза существующих. В настоящее время все это решаемо на базе автоматического решения задачи оптимизации и даже не всегда называется искусственным интеллектом. Есть надежда, что в будущем моделирование мышления тоже станет не столь заоблачной целью.

С проблемой возможности воспроизведения мышления тесно связана проблема возможности самовоспроизведения, которое долгое время считалось прерогативой живых организмов. Этот факт можно опровергнуть, приведя такие доказательства, как самопроизвольный синтез слож-

ных молекул в результате слияния простых, рост кристаллов и др. Доказательством возможности воспроизведения в искусственном интеллекте является распространение компьютерных вирусов.

Другим философским вопросом искусственного интеллекта является цель создания. Существующие в настоящее время интеллектуальные системы являются плодом человеческой лени. Лени в плане желания человека экономить свою энергию для решения задач. Однако, что будет, если однажды будет воспроизведена интеллектуальная система, способная выполнять все, что умеет человек? Что будет, если интеллектуальная система помимо своих возможностей будет обладать волей, желаниями и характером? В этом случае она уже перестанет подчиняться, перестанет делать только то, на что укажет ей человек. В этом случае возникнет проблема безопасности. Тогда что станет с людьми? Какую роль будут они играть? Ответы на эти вопросы пока только попытались найти авторы скорее художественных, а не научных произведений. Например, у ученого и писателя-фантаста Айзека Азимова можно найти описание решения проблемы безопасности, принятое и учеными в области искусственного интеллекта. Оно заключается в определении трех так называемых законов робототехники:

1 Робот не может причинить вред человеку или своим бездействием допустить, чтобы человеку был причинен вред.

2 Робот должен повиноваться командам, которые ему дает человек, кроме тех случаев, когда эти команды противоречат первому закону.

3 Робот должен заботиться о своей безопасности, насколько это не противоречит первому и второму закону.

На первый взгляд подобные законы, при их полном соблюдении, должны обеспечить безопасность человечества. С другой стороны, эти законы сформулированы на человеческом языке и не допускают свой перевод в алгоритмическую форму. Сложно представить себе на машинном языке фразу «причинить вред» или «допустить вред». Также сложно представить, как будет пониматься искусственным интеллектом слово «вред». Ведь человек зачастую сам себе вредит: курит, пьет, теряет здоровье. В этом смысле искусственный интеллект может понять, что человек и является сам себе вредным.

Из вышесказанного следует, что разработанные три закона робототехники хотя и являются неплохим общим базисом для разработки интеллектуальных систем, однако требуют значительной конкретизации и дополнения, что на сегодняшний день до конца не осуществимо.

1.2 Цели и задачи искусственного интеллекта

Впервые понятие «искусственный интеллект» было предложено на конференции в Дартмутском колледже Джоном Маккарти в 1956 году [3].

В литературе встречаются различные определения понятия «Искусственный интеллект». В различных трактовках ИИ понимается как наука о машинах с человеческим интеллектом, область информатики, охватывающая компьютерные методы и технологии символьного вывода и представления знаний, а также как решение задачи познания с помощью компьютерной имитации. Общее определение искусственного интеллекта выглядит следующим образом:

Искусственный интеллект – это область компьютерных наук, которая занимается исследованием и автоматизацией разумного поведения.

Основной целью исследователей интеллектуальных систем является получение искусственных устройств с целенаправленным поведением и разумными вычислениями, схожими с мышлением.

Мышление по сути своей является деятельностью по решению интеллектуальных задач путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на основе опыта и при адаптации к обстоятельствам.

В определении мышления под термином «знания» предполагается не только та информация, которая поступает извне в мозг через органы чувств, но также ее восприятие, «целенаправленное преобразование». Другими словами, знания являются метаинформацией, пониманием и осознанием получаемых данных.

Задачи, решаемые в процессе мыслительной деятельности, названы интеллектуальными не просто так. Понятие «интеллектуальный» можно объяснить на примере описания алгоритма. Алгоритмом является точное описание действий по выполнению любой задачи из конкретного класса задач. Отыскание алгоритма требует рассуждений, которые могут быть построены только при наличии определенных знаний и компетентностей, что подразумевает интеллектуальный человеческий труд. Поэтому задачи, связанные с отысканием алгоритма решения, называются интеллектуальными, а важнейшим направлением в области искусственного интеллекта является разработка механизмов переноса компетентностей и знаний человека – обучение искусственных устройств. Что же касается задач с установленными алгоритмами, то, как отмечает известный специалист в области искусственного интеллекта Марвин Минский, «излишне приписывать им такое мистическое свойства, как "интеллектуальность"».

Таким образом, возникает еще одно определение интеллекта: *интеллект – универсальный мета-алгоритм.*

Предметом ИИ является изучение интеллектуальной деятельности человека, которая не подчиняется заранее известным законам.

Следующие понятия наиболее часто используются в теории искусственного интеллекта, в связи с чем их стоит раскрыть:

Система – множество элементов, образующих причинно-следственную связь.

Адаптивная система – это система, которая сохраняет свои свойства и работоспособность при непредвиденных изменениях окружающей среды путем подстройки своего поведения.

Интеллектуальная система (ИС) – адаптивная система, строящая решения на основе сложившейся на данный момент конкретной ситуации, зависящей от внешней среды.

1.3 Характеристики интеллектуальной системы

Как же понять, является ли система интеллектуальной или нет?

ИС должна уметь в наборе фактов распознавать существенные, ИС должна делать выводы на основе не только дедуктивного вывода (т.е. из набора имеющихся правил «ЕСЛИ-ТО»), но и с помощью аналогий и индукции (т.е. на базе анализа частных примеров). Кроме того, ИС должна обладать рефлексией (т.е. способностью к самоанализу).

Для примера представим компьютерную шахматную игру, которая является одновременно примером интеллектуальной и неинтеллектуальной системы. Ее можно считать интеллектуальной за исключением некоторых «НО». В случае, если игра делает одну и ту же ошибку в каждой партии, то она не является интеллектуальной, так как она не является адаптивной. Также игру нельзя считать интеллектуальной, если она действует на основе жесткого перебора правил.

Попытку интеллектуальной шахматной игры впервые предложил в 1954 году Аллен Ньюэлл. Два аналитика – Клифф Шоу и Герберт Саймон – предложили ему свою помощь. В 1956 году ими был создан язык программирования *IPL*-1 – первый символьный язык обработки списков. Вскоре ими была впервые предложена первая программа – *Logic Theorist* (или *LT*) – которая стала одним из первых достижений в программировании искусственного интеллекта и позволяла производить доказательства теорем в логике высказываний. Сама же программа для игры в шахматы, названная *NSS* (по первым буквам фамилий авторов), была завершена в 1957 году и была основана на *эвристическом поиске* (или поиске на основе выбора без достаточных теоретических объяснений). Алгоритм пытался уменьшить разность между оценками текущей ситуации и оценками цели.

Этой же группой (Ньюэлл, Шоу и Саймон) была изобретена программа *General Problem Solver*, позволяющая решать головоломки типа ханойской башни и вычислять интегралы. Процесс работы General Problem Solver также основывался на эвристиках.

Джон Маккарти (автор понятия «Искусственный интеллект) заинтересовали наработки его коллег, и в 1963 году им был разработан язык ЛИСП (или *List Processing*). Далее были предложены такие языки обработки символьной информации как Пролог и Рефал.

К концу 60-х годов прошлого века начали появляться первые игровые программы, системы для элементарного анализа текста и некоторых математических задач. Все они были основаны на эвристическом программировании, т.е. резком снижении перебираемых вариантов в сложных задачах на основе «здравого смысла». Однако позже было доказано, что у алгоритмов эвристического поиска существует предел**,** что при усложнении алгоритма и увеличении эвристик эффективность не повышается до бесконечности, а, главное, не допускается появление новых свойств и возможностей у ИС.

1.4 Предпосылки создания экспертных систем

Постепенно пришло понимание того, что всем ранее созданным программам не достает одного – знаний и опыта. Если программы при обработке информации будут обращаться к знаниям, то это приведет к более высокому качеству работы.

Это понимание в итоге привело к качественному скачку в 1970-х годах, в следствие чего появились *экспертные системы – прикладные системы, которые используют различные способы представления знаний для решения задач*. Основная идея экспертных систем заключается в интегрировании знаний эксперта в ИС и оперирования ими в процессе функционирования.

Прототипом ЭС является программа *DENDRAL*, написанная в первой половине 1960-х и позволяющая подсчитывать всевозможные конфигурации заданного множества атомов. Такая система оказалась чрезвычайно полезной при решении задач, для которых не существует аналитических методов.

В том же месте, что и *DENDRAL* (Стэндфордский университет), были разработаны две экспертные системы, вошедшие в историю ИИ как эталонные. Первая – PROSPECTOR – облегчила поиск полезных ископаемых и оценку их запасов. По сути, эта система является диалоговой системой, которая работает на основе правил, полученных от специалистов. Вторая – *MYCIN* – была спроектирована как система диагностики заболеваний. Основана *MYCIN* на знаниях о конкретных пациентах и их болезнях, также диагнозе и рекомендациях по лечению. Система позволила принимать решения при наличии неполной информации с расчетом уровня уверенности в поставленном диагнозе.

В настоящее время разработка экспертных систем превратилась в полноценную инженерную дисциплину**.** Среди современных отечественных известных экспертных систем можно выделить систему АТ-ТЕХНОЛОГИЯ**,** разработанную в институте МИФИ для обучения студентов и персонала. Подробнее об экспертных системах и представлению в них знаний описано в главе 3.

1.5 Прикладные интеллектуальные системы

Другим направлением развития ИС является робототехника. Понятие «Робот» впервые введено еще до появления ЭВМ и ИИ в произведении Карела Чапека «*Р.У.Р.*» [4]. Автор представляет картину неуважительного использования человекоподобных машин. История пьесы развивалась на фабрике, где использовались так называемые роботы (от старого чешского «*robota*» – подневольный труд) – подневольные, труд которых был направлен на замену человеческого труда. Через какое-то время количество роботов увеличивается, спрос на них растет, они начинают использоваться в военной промышленности. Еще через время у роботов начали проявляться проблески интеллекта. К концу пьесы автор показывает, как при появлении численного превосходства и интеллекта роботы начали истреблять человечество до полного уничтожения. Итогом произведения является появление разумных аналогов Адама и Евы, и начала новой эры.

Реальное же появление роботов относится к 50-м годам прошлого века, когда на заводе американской фирмы *General Motors* появились первые манипуляторы для сборки автомобилей. С тех пор началось динамичное развитие робототехники.

Робот – автоматическая система, выполняющая антропоморфные действия, направленные на замещение труда человека.

Одними из первых реализаций роботов были луноходы и марсоходы, изобретенные в конце 1960-х годов, которые могли передвигаться на основе манипуляций с Земли, а также осуществлять сбор грунта. В общепринятом понимании, т.е. как «машины с человекоподобным поведением», роботы появились в 1977 году. В это время был создан робот, умеющий подметать пол, стричь траву и готовить простую пищу. В то же время были созданы и робот, который поднимает и опускает предметы, и робот, который умеет прикуривать, а также множество других человекоподобных машин. Одним из наиболее выдающихся роботов того времени является робот Cubot, который мог собрать кубик Рубика менее чем за четыре минуты.

Хотя основы робототехники были заложены в 60-е годы, большого прогресса данная область не могла достигнуть долгие годы в связи с отсутствием материалов, технологий и вычислительных ресурсов.

В настоящее же время существует достаточное количество различных реализаций робототехники. Разработка роботов сегодня ассоциируется с Японией. К примеру, Honda демонстрирует все новые версии робота *ASIMO*. Создатели утверждают, что робот умеет говорить на двух языках, а также перемещается по лестнице и умеет разговаривать. Также японские разработчики (но уже из Sony) представили робота *AIBO*, который имеет вид щенка и может прыгать, бегать, вилять хвостом, а также демонстрировать собачьи эмоции. Два минуса современных роботов – они стоят дороже современных машин класса «Люкс», и руководство по их пользованию занимает как минимум 150 страниц.

Современных роботов по степени интеллектуальности можно классифицировать следующим образом:

1Роботы без обратной связи, выполняющие неоднократно одинаковые операции.

2Роботы с обратной связью, выполняющие разные операции.

3Обучаемые роботы. Обучение таких роботов производится оператором.

4Интеллектуальные роботы, способные находить нужные вещи, оценивать обстановку и принимать решения.

Исследователи интеллектуальности роботов и других прикладных систем ИИ задаются вопросом: насколько интеллектуальной должна быть система? Здесь сформировалось две гипотезы ИИ.

Слабая гипотеза ИИ. Интеллектуальная машина должна имитировать процесс человеческого познания, однако не может распознать психические состояния. Такая машина имеет все шансы на прохождение теста Тьюринга.

Сильная гипотеза ИИ. Интеллектуальная машина должна распознавать когнитивные психические состояния. Такой подход позволяет создать машину, способную осознать собственное существование и обладающую эмоциями.

Рождение робототехники выдвинуло задачи компьютерного зрения и распознавания изображений в число первоочередных. Теория компьютерного зрения начала зарождаться в 50–60-е годы прошлого века на основе результатов применения алгоритмов, обеспечивающих отнесение нового объекта к одному из заданных классов. Так как изображения того времени были малоинформативными, исследованиям подвергались в основном печатные символы.

В это время производились первые попытки моделирования нейронной деятельности человеческого мозга. Ученые попытались смоделировать поведение мозга, когда бесконечное множество состояний внешней среды отображается в конечное множество внутренних реакций. Первый успех здесь связан с разработкой психолога Фрэнка Розенблатта – перцептроном (или персептроном). Первый аппаратный вариант персептрона – *Mark 1* – был изготовлен в 1960 году и предназначался для распознавания образов, с чем безупречно справлялся. Однако посильные ему образы ограничивались печатными буквами.

Позднее, в 70-х, с увеличением разрешающей способности изображений начали появляться новые задачи распознавания образов, такие как навигационная задача, задача обработки изображений луны, обработка тепловизионных снимков и др. Все это привело к бурному развитию средств обработки изображений.

В 80–90-е годы стали появляться новые датчики обработки визуальной информации (приборы ночного видения, лазерные дальномеры и др.).

Появление новых компьютеров позволило реализовать обработку изображений даже на стационарных машинах, в связи с чем появляются приложения для домашнего использования на основе компьютерного зрения.

К настоящему моменту теория компьютерного зрения полностью сложилась как самостоятельный раздел, опирающийся на солидную научную и практическую базу знаний. Использующийся на сегодня инструмент *Deep learning* (глубинное обучение) позволяет практически также определять изображения, как это делают люди. Однако это практически применимо лишь в очень узких областях. К примеру, до сих пор нет алгоритма, который отличит на фото женщину от мужчины с высокой вероятностью. Кроме того, на сегодняшний день сложно представить такой алгоритм, который найдет нам все изображения «сложного описания», к примеру, где рыжий кот играет с фиолетовым мячиком на зеленом ковре (или тем более видео с таким содержанием), что объясняется нехваткой вычислительных ресурсов.

Другим направлением, рождение которого стало также необходимо при развитии интеллектуальных и робототехнических систем, стала обработка естественного языка, понимание речи и перевод текстов.

Началом работ по данной тематике следует считать 1954 год, когда в штабе корпорации *IBM* с помощью ЭВМ был произведен так называемый Джоджтаунский эксперимент. В ходе эксперимента был продемонстрирован автоматизированный перевод 60 фраз с русского на английский. Система включала всего 6 правил и 250 записей и была сугубо специализированной на органической химии. Этот эксперимент принес большой успех, что вскружило голову организаторам, которые провозгласили, что через каких-то 5 лет проблема машинного перевода будет решена. Однако идея была не доведена до конца и за 10 лет исследования прикрыли.

А прокололись авторы Джорджтаунского эксперимента в следующем. Они сказали, что достаточно создать большие хранилища словарей для перевода с одного языка на другой, разработать правила – и все! Однако это не явилось правдой. В 70-х годах было выяснено, что необходимо создать язык-посредник, облегчающий сопоставление фраз на различных языках. Во второй половине 70-х этот язык стал семантической моделью представления смысла переводимых текстов.

Задача смыслового представления является важной даже в случае, если необходим анализ текста в сугубо частной сфере. Так, для одной формальной предметной области и конкретных наборов предметов, обладающих недвусмысленными характеристиками и названиями, в 1971 году была создана программа *SHRDLU*.

Стоит отметить, что до сегодняшнего дня даже для английского языка (не говоря о русском) не существует программы, позволяющей адекватно понимать смысл фраз. В этой связи все еще остается множество открытых вопросов в этой сфере.

Автор эксперимента с Китайской комнатой Джон Сёрл однажды ответил на вопрос о важнейших достижениях искусственного интеллекта следующее [2]:

«Мне не настолько хорошо известны технологические достижения в этой области, однако меня всегда восхищали успехи в сфере автоматического понимания естественного языка».

Итак, современные, исследования в области преобразования речи и естественного языка охватывают следующие проблемы:

- синтез речи;
- понимание устной речи;
- понимание естественного языка;
- автоматический перевод.

В отдельную и не менее интересную область ИИ можно выделить машинное творчество.

В 1957 году американские исследователи Мэтьюз и Гутман посетили концерт малоизвестного пианиста. Концерт им не понравился, и они решили, что можно написать программу, играющую не хуже этого малоизвестного музыканта. В итоге под их началами выросло целое отделение анализа музыки в Стэндфордском институте.

Другим примером служат наработки Рудольфа Зарипова, который попытался сочинить музыку на вычислительной машине «Урал». Назвал их Зарипов «Уральские напевы». Для их сочинения он использовал случайные комбинации различных элементов музыкальной фактуры (форма, ритм и высота звука). И с его начал появилось множество алгоритмов обработки и синтеза музыки и различных звуков.

В 70-х годах были даже проведены эксперименты по сравнению человеческой и машинной музыки, в результате которых некоторые искусственные произведения оказались неотличимы от творчества человека. В 1975 году был проведен тест Тьюринга на искусственные музыкальные произведения, который в частных случаях имел успех.

Хотя это все в какой-то мере стало открытием в сфере искусственного интеллекта, данная теория столкнулась со множеством критик. К примеру, известный ученый Дмитрий Поспелов утверждает, что такой метод весьма стохастический, случайный, и никак не может быть назван творчеством; очевидно, что таким образом не будет создано что-то гениальное.

1.6 Архитектура интеллектуальных систем

Среди множества подходов к созданию искусственных интеллектуальных систем стоит выделить четыре основных группы подходов [5]:

Логический подход. Возникновение логического подхода обусловлено утверждением, что именно логическое мышление отличает человека от животных. Именно логика является первым этапом приближения машин к человеку. Основой для логического подхода служит логика предикатов

первого порядка, которая в свою очередь основана на Булевой алгебре и исчислении высказываний. Почти каждая компьютерная программа связана с логикой предикатов, так как при написании программного кода так или иначе требуется оператор «*if*».

Интеллектуальная система, построенная по логическому принципу, представляет собой машину, реализующую алгоритм доказательства теорем. При этом исходными данными являются аксиомы, а отношениями между ними – правила. Хотя современные вычислительные машины и основаны на представлении информации в виде булевых переменных, такой подход показал себя исчерпанным при моделировании выразительных способностей человека. Ведь мы используем не только двоичную информацию в виде «ДА/НЕТ», но еще и украшаем ее различными прилагательными (скорее да, возможно нет), а также можем придерживаться промежуточных значений (ни да, ни нет, не знаю). В этой связи продолжением логического подхода явилась нечеткая логика, основным отличием которой является возможность установки степени правдивости высказывания, что больше похоже на человеческое мышление.

Структурный подход. Структурный подход подразумевает попытки построения ИИ путем моделирования структуры человеческого мозга. Одной из первых успешных реализаций такого подхода является перцептрон, в котором в качестве структурной единицы выступает нейрон.

В настоящее время разработано множество структурных моделей интеллектуальных систем. Все они известны под общим названием «искусственная нейронная сеть». Эти модели отличаются по строению нейронов, по топологии связей между ними и по алгоритмам обучения. Наиболее популярными являются сети с обратным распространением ошибки, сети Хопфилда и стохастические нейронные сети. Одним из главных плюсов нейронных сетей является то свойство, которое позволяет им работать даже в условиях неполноты информации с возможностью выдачи неточных ответов. Хотя они и получили достаточное развитие в последние полвека, все же основным приложением структурных моделей на сегодняшний день является скорее не представление мышления, а распознавание образов. Однако встречаются работы, где описано успешное применение нейронных сетей при построении базы знаний интеллектуальной системы.

Эволюционный подход. Эволюционный подход по сути своей является попыткой использования теории Дарвина при построении интеллектуальных систем. Интеллектуальная система, построенная по такому принципу, является изменчивой, модифицируемой во времени. При построении эволюционной интеллектуальной системы необходимо правильно выбрать начальный вид модели и правила ее эволюционирования. Целью такой модели является оптимизация ее структуры, выбор лучшего пути развития на основе моделирования процесса естественного отбора.

Имитационный подход. Имитационная интеллектуальная система является моделью черного ящика, для которого известны входные и выходные воздействия, но не известна структура. При таком подходе нам не важно, как функционирует моделируемый объект, нам важно получить конкретные ответы на конкретные входные сигналы, чтобы интеллектуальная система копировала поведение исходного черного ящика. Таким образом, имитационный подход позволяет перенести в интеллектуальную систему другое свойство человека – способность копировать то, что делают другие, не вдаваясь в подробности, зачем это надо (как в эксперименте с китайской комнатой). Если такая затея будет когда-нибудь в полной мере реализована, вопрос увековечивания людей будет в буквальном смысле решен, ведь даже после смерти определенного человека его имитационная система будет жить и реагировать на внешние факторы так же, как это делал бы он.

1.7 Современный этап и перспективы развития искусственного интеллекта

На сегодняшний день тематика искусственного интеллекта охватывает огромный перечень научных направлений, начиная с таких задач общего характера, как обучение и восприятие, и заканчивая такими специальными задачами как игра в шахматы, доказательство теорем, сочинение поэтических произведений и диагностика заболеваний. Реализованы интеллектуальные системы (или, скорее, подсистемы), которые функционируют не хуже живых систем, при этом внутренняя структура естественных прототипов полностью до сих не раскрыта. Так, к примеру, искусственный глаз способен видеть в более широком диапазоне волн, при слабом освещении и без усталости. Устройства обработки голоса позволяют уловить девиацию голоса в 1–2 Гц, что позволяет уловить нервные колебания человека и с высокой точностью определить, лжет он или нет (даже в звукозаписи). Антиблокировочная система позволяет держать тормоза на грани заклинивания колес, что не удается сделать даже опытным водителям. В общем, при создании искусственного интеллекта человеком устраняются ограничения, которые присутствовали при создании природой естественного, так как алгоритмы живой природы стохастичны и не являются структурированными, что мешает достичь оптимума, дать наивысший эффект. При применении в «слабых местах» жестких машинных алгоритмов и систем управления создается возможность получения требуемой цели в заданной точности при использовании минимально времени обучения.

Что касается перспектив развития искусственного интеллекта, рассуждения по этому поводу разделяются на множество направлений. На сегодняшний день считается, что логическая память современных ЭВМ в том виде, в котором она есть, никогда не сможет достичь уровня памяти человека, являющейся ассоциативной. Процессы в нашем мозгу не поддаются

строгим вычислениям. Однако, как показывают исследования, вопрос построения искусственного интеллекта, аналогичного естественному, – вопрос времени. Ведь человеческий мозг в последние столетия почти не совершенствовался. В противовес этому, искусственный интеллект развивается невероятными темпами в последние 20 лет. Некоторые ученые придерживаются такой точки зрения, что рано или поздно искусственный интеллект сможет более успешно справляться с жизненными проблемами, чем чувствительный и ранимый представитель *homo sapiens*.

ГЛАВА 2. ФОРМАЛЬНЫЕ ЯЗЫКИ КАК МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

2.1 Понятие формального языка. Язык логики

Основным средством взаимодействия людей является язык. *Язык – знаковая система, которая используется для коммуникации и познания.*

Естественный язык обладает такими свойствами как неоднозначность, избыточность, а также возможность противоречивых описаний, что, безусловно является достоинством естественного языка, но создает непреодолимые проблемы при попытках его формализации в компьютерных системах. Другими словами, *естественный (разговорный) язык – знаковая система, которая возникла в стихийном порядке в течение долгого времени.*

В каждом естественном высказывании одно слово может иметь различный смысл в зависимости от контекста. Любое выражение может иметь фрагменты, явно не обозначаемые, но подразумеваемые и понятные собеседнику. В тексте возможно повторение слов для усиления акцента. Такое свойство как противоречивость описаний позволяет строить художественные, поэтические формы текста.

Отличительные черты естественного языка:

- многозначность слов;
- неточность определения многих слов;
- зависимость смысла от контекста;
- наличие синонимов;
- правила построения фраз не являются строгими, т.е. бывают исключения из правил.

Еще большая проблема состоит с языком мышления. Мало кто из нас может воспроизвести в точности фразу, которую услышал, допустим, от бабушки на лавочке сегодня утром во дворе, однако смысл этой фразы (если, конечно, она отложилась в памяти) передать мы в состоянии. Это свидетельствует о том, что люди обрабатывают слова, после чего формируют своего рода несловесное представление, которое и называется памятью. Другими словами, память человека ассоциативна.

К сожалению, такой способ общения, представления и запоминания до сих пор не реализован у искусственно мыслящих машин.

Для того, чтобы общаться с машинами, людям пришлось наложить на естественный язык ряд существенных ограничений. Ограничения эти должны, в частности, устранить многозначность языка, а также установить правила использования вспомогательных знаков. В случае наложения таких ограничений естественный язык признается искусственным. *Искусственный язык – язык, специально созданный людьми для определенных целей.*

Если к искусственному языку применить явно и строго сформулированную совокупность правил построения выражений, то он становится формальным (или формализованным) [6, 7].

Формальный язык – искусственный язык, характеризующийся наличием точных правил построения выражений, а также правил их понимания. Он строится в соответствии с четкими правилами, обеспечивая непротиворечивое, точное и компактное отображение свойств описываемого объекта, позволяя свести процесс познания к математическому вычислению.

Все языки, используемые при создании искусственного интеллекта, так или иначе базируются на языке логики.

Для понимания основы формальных языков важным является изучение логики предикатов первого порядка [8]. Эта логика, называемая также языком исчисления предикатов первого порядка, активно используется при автоматизации рассуждений, а также при планировании поведения и приобретении знаний.

Предикат – некоторое утверждение, функция вида $M^k \rightarrow \{0,1\}$*, которая принимает значение 1(истина) или 0(ложь).* M – множество определения переменных, k – количество переменных, называемое арностью предикатов.

Логика предикатов первого порядка по сравнению с логикой высказываний подразумевает использование кванторов всеобщности и существования.

Алфавит логики предикатов первого порядка включает:

1 $x, y, z, \ldots$ – индивидуальные переменные.
2 $P, Q, R, \ldots$ – предикатные символы (символы обозначения предикатов). Например, $P(x_1, \ldots, x_k)$ – k-арный (k-местный) предикат.
3 $f, g, h, \ldots$ - функциональные символы для обозначения функций вида $M^k \rightarrow$ M. Например, $f(x_1, \ldots, x_k)$ – k-арная (k-местная) функция.
4 $a, b, c, \ldots$ – константы (или 0-местные функции).
5 $\neg, \wedge, \vee, \rightarrow$ – логические операции «не», «и», «или», «следует».
6 $\forall, \exists$ – кванторы всеобщности и существования.
7 (,) – служебные символы.

При этом связки $\wedge, \vee$ выражаются через $\neg, \rightarrow$, а $\exists$ через $\forall$:

$A \wedge B = \neg(A \rightarrow \neg B)$

$A \vee B = \neg A \rightarrow B$

$\exists x\, A(x) = \neg \forall x\, \neg A(x)$

Определение терма и формул:

1 Символ для обозначения переменной или константы есть терм.
2 Если $t_1, \ldots, t_n$ – терм, то $f(t_1, \ldots, t_n)$ – терм.
3 Если $t_1, \ldots, t_n$ – терм, то $P(t_1, \ldots, t_n)$ – атомарная формула (литерал).
4 Атомарная формула является формулой.
5 A, B – формулы, то $(A \rightarrow B)$, $\neg A$, $\neg B$ – формулы.
6 Если A – формула, то $(\forall x)\, A$ – тоже формула.

7 Всякое слово является формулой тогда и только тогда, когда его можно показать в виде конечного числа применений 1–6.

Свободными переменными называются переменные без квантора.

Связанными переменными называются те, по которым стоит квантор.

Рассмотрим выражение

$\forall x\ x > y$.

Здесь x – связанная переменная, а y – свободная.

С этим выражением все ясно. Однако логика предикатов допускает такие формулы

$\forall x\ x > y \rightarrow \forall y\ y < x$,

где переменная y с одной стороны свободная, а с другой стороны – нет. Поэтому корректно говорить о связанных вхождениях, а не о связанных переменных в целом.

Замкнутая формула – формула, в которой отсутствуют свободно входящие переменные. При подстановке конкретных значений в такую формулу она превращается в предложение (которое точно является истинным или ложным). В случае незамкнутой формулы подстановка превращает ее в формулу, зависящую от связанных переменных.

Атомарная замкнутая формула называется фактом.

Если язык состоит только из предложений, то он – пропозициональный язык. Такой язык можно назвать логикой нулевого порядка или исчислением высказываний.

Аксиомы предикатов первого порядка, которые также являются аксиомами логики нулевого порядка:

1 $A \rightarrow (B \rightarrow A)$.

2 $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

3 $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$

Правила вывода в логике первого порядка:

a) *modus ponens* (правило отделения). Если A выводимо (или тождественно истинно, тавтология) и $A \rightarrow B$ выводимо, то B выводимо.

$$\frac{A, A \rightarrow B}{B}$$

или

$A \wedge (A \rightarrow B) \vdash B$ ($\vdash$ – знак выводимости)

b) Правило подстановки. В любую аксиому на место любой пропозициональной переменной можно подставить любое предложение с пропозициональными переменными, отличными от заменяемой.

$$\frac{P(A)}{P(B)}.$$

Пример вывода:

Доказательство закона тождества Аристотеля:

$P \rightarrow P$ (сам закон).

Подставим $P \rightarrow P$ вместо B (по правилу подстановки) в аксиому 1:

$A \to ((P \to P) \to A)$.

Из аксиомы 2 получим:

$(A \to ((P \to P) \to C)) \to ((A \to (P \to P)) \to (A \to C))$.

Так как первая часть является аксиомой (и, следовательно, выводима), и вся формула выводима, то и вторая часть выводима по правилу отделения:

$((A \to (P \to P)) \to (A \to C))$.

Подставим P вместо A и C:

$((P \to (P \to P)) \to (P \to P))$.

Первая часть снова является аксиомой. Таким образом, $P \to P$ также выводимо.

Исчисление предикатов добавляется еще двумя аксиомами и двумя правилами помимо тех, что используются в логике нулевого порядка:

4 Аксиома генерализации:

$$(\forall x)\,(A \to B) \to (A \to (\forall x)\,B),$$

где x – несвободная переменная в A.

5 Аксиома спецификации:

$$\forall t\, A(t) \to A(x),$$

где t – терм, а x не содержится в t в качестве свободной переменной.

c) правило обобщения:

$$\frac{F \to G(x)}{F \to (\forall x)G(x)},$$

где $G(x)$ содержит свободное вхождение x, а F не содержит.

d) Правило конкретизации:

$$\frac{G(x) \to F}{(\exists x)G(x) \to F},$$

где $G(x)$ содержит свободное вхождение x, а F не содержит.

Если к аксиомам добавить какую-нибудь предметную область (например, арифметику), то получим формальную теорию (например, формальную арифметику).

Как стало ясно из предыдущих пояснений, логика первого порядка отличается от логики высказываний тем, что в ней действуют кванторы всеобщности и существования на некотором множестве. Если действие кванторов распространяется уже на подмножества множества, то это уже логика второго порядка, на множества множеств – логика третьего порядка и т.д. Логики высоких порядков также принято называть сильными логиками.

2.2 Формальные системы

Обобщением понятия «исчисление» является формальная система [6].

Формальная система (исчисление) – совокупность абсолютно абстрактных объектов, в которой представлены правила оперирования цепочками символов в синтаксической трактовке без учета смыслового содержания.

Говорят, что формальная система задана, если заданы некоторый алфавит, множество правил, аксиом и правил вывода, т.е. *формальная система F определяется в виде четверки:*

$F=<T,P,A,П>$,

где T – алфавит (конечное множество символов);

P – множество правил, применение которых к элементам из T позволяет строить правильно построенные формулы;

A – множество аксиом;

П – конечное множество правил вывода.

Если среди аксиом имеются нелогические аксиомы (описывающие предметную область), то формальная система называется формальной теорией.

Выводом (или доказательством) f_m из множества формул $\{f_1, f_2, ..., f_k\}$ в формальной системе называется конечная последовательность правильно построенных формул $f_{k+1}, f_{k+2}, ..., f_{k+m}$, таких что каждая из формул последовательности либо является аксиомой, либо получена из предыдущих формул последовательности с использованием аксиом или правил вывода. Формула f_m в этом случае называется выводимой формулой (или теоремой) формальной системы F.

Запишем пример:

$T = \{a, b, c\}$.

Р – любая последовательность символов алфавита Т.

$A = \{aca\}$.

П = $\{\frac{X_1 c X_2}{b X_1 c X_2 b}\}$, где X_1 и X_2 – любые последовательности символов из T

Вывод формулы *bbacabb* будет следующим:

1 *aca*

2 *bacab*

3 *bbacabb*

bbacabb является теоремой, так как она выведена из аксиомы.

Такая формула как, допустим, *babbacab* не является теоремой, так как ее нельзя вывести из аксиомы.

Использование формальных систем при описании знаний в конкретной предметной области подразумевает использование алгебраической системы.

Алгебраическая система – это упорядоченная тройка:

$S = <M,G,R>$,

где M – основное множество;

G – множество n-местных функций из M^n в M;

R – семейство n-арных отношений на M.

n-арное отношение $R_i \in R$ на M – это некоторое подмножество $R_i \subseteq M^n$.

Если в определении алгебраической системы положить $G = \emptyset$, то такая алгебраическая система $S = <M, \emptyset, R>$ будет называться *моделью*, а если $R = \emptyset$ – то *алгеброй*.

Понятие алгебраической системы играет важную роль в случае использования формальных систем для представления знаний и манипулирования ими в конкретной предметной области. По своей сути алгебраическая система является результатом интерпретации формальной системы.

Интерпретацией формальной системы является модель языка $\Omega = <M, I>$, где M – непустое множество (область интерпретации), I – однозначное отображение, которое отображает формальную систему в алгебраическую, а именно:

1 Каждому константному символу ставит элемент M.

2 Каждому предикатному символу – отношение R.

3 Каждому функциональному символу – функцию G.

Формула является выполнимой в данной интерпретации, если существует такая подстановка констант, при которой она превращается в истинное высказывание.

Формула является истинной в данной интерпретации, если она истинна при любой подстановке констант.

Рассмотрим пример.

Пусть формальная система определена трехместным предикатом $P(x, y, z)$. Интерпретация ставит в соответствие этому символу трехместное отношение R = «СЕМЬЯ», заданное в виде таблицы (**Ошибка! Источник ссылки не найден.**).

Пример интерпретации формальной системы, заданной предикатом $P(x, y, z)$

Мать	**Отец**	**Ребенок**
Аня	Петя	Вася
Галя	Женя	Коля

Тогда универсумом является M = {Петя, Аня, Вася, Галя, Женя, Коля}. Формула $P(x, y, z)$ выводима только на тройках (Аня, Петя, Вася) и (Галя, Женя, Коля).

Отсюда следует, что $\forall x \forall y \forall z\ P(x, y, z)$ ложно, а $\exists x \exists y \exists z\ P(x, y, z)$ – истинно в построенной модели.

В завершение подытожим весь материал теоремой Гёделя о полноте, которая устанавливает взаимосвязь между выводимостью и истинностью формулы.

Произвольное предложение выводимо в исчислении предикатов первого порядка тогда и только тогда, когда оно истинно.

ГЛАВА 3. ОПРЕДЕЛЕНИЕ БАЗЫ ЗНАНИЙ. ЭКСПЕРТНЫЕ СИСТЕМЫ

3.1 Основные критерии знаний. Отличие знаний от данных

Параллельно с развитием структуры ЭВМ происходило развитие информационных структур для представления данных. Появились способы описания данных в виде: векторов, матриц, списочных структур, иерархических структур, структур, создаваемых программистом (абстрактных типов данных) [9, 10].

В настоящее время в языках программирования высокого уровня используются абстрактные типы данных, структура которых создается программистом. Появление баз данных знаменовало собой еще один шаг по пути организации работы с декларативной информацией. Напомним, что информация, с которой имеет дело ЭВМ, разделяется на *процедурную* и *декларативную.* Процедурная информация овеществлена в программах, которые выполняются в процессе решения задач, декларативная – в данных, с которыми эти программы работают.

При попытке объединения декларативной и процедурной информации возникает ряд трудностей [11]. По сути такое объединение является в некотором роде попыткой формализации мышления человека. Но как формализовать мышление, когда, во-первых, мы сами не всегда в силах описать алгоритм своих действий. Мы не знаем точный алгоритм распознавания текста, узнавания лица, выработки плана действий. Нам лишь известна некоторая часть этого алгоритма. Во-вторых, из-за физической ограниченности ЭВМ далеки от человеческого уровня компетентности: для того, чтобы машина выполнила какие-то действия, ей нужен конкретный алгоритм и программа.

По мере развития исследований в области интеллектуальных систем возникла концепция знаний, которая объединила в себе многие черты процедурной и декларативной информации.

Как было отмечено ранее, знания являются основой интеллектуальных систем. Одним из основных вопросов в теории искусственного интеллекта является объяснение соотношения и различия знаний и данных, а также описание особенностей их представления. Наглядное разделение данных и знаний представлено в Таблица 3.1, где проиллюстрированы некоторые виды их существования.

Как видно из таблицы, данные и знания имеют вполне схожую структуру, однако знания являются более сложной формой представления информации, в связи с чем в теории искусственного интеллекта трактовались как метаданные (данные о данных), хорошо структурированные данные и т.д.

Впервые характеристику знаниям интеллектуальной системы дал Д.А. Поспелов [12]:

1 Знания имеют более сложную структуру (метаданные).

2 Возможность задавать знания как экстенсионально (т.е. через набор конкретных фактов), так и интенсионально (т.е. через свойства). Данные задаются экстенсионально.

3 Знания обладают внутренней интерпретируемостью, что предусматривает связь информационной единицы с системой имен, т.е. наличие некоторой протоструктуры для информационной единицы.

4 Наличие рекурсивной структурированности знаний по принципу «матрешки», т.е. возможности представления одной информационной единицы в виде составных частей, которые в свою очередь также разложимы на части.

5 Знания взаимосвязаны. Здесь связи подразумевают наличие отношений между информационными единицами.

6 Наличие семантической метрики у знаний, т.е. возможности определять степень близости информационных единиц.

7 Знания обладают активностью, которая характеризуется возможностью ставить и достигать цели, формировать мотивы поведения, строить процедуры решения задач. Другими словами, знания имеют «дышащие» алгоритмы (*прим. авторов*).

8 Знания имеют функциональную целостность, т.е. возможность выбора желаемого результата, а также времени и средств его получения и анализа.

Таблица 3.1

Разделение понятия «Знания» и понятия «Данные»

ЗНАНИЯ	**ДАННЫЕ**
Начальное представление	
$Зн_1$ – знания в памяти человека	$Д_1$ – результат наблюдений над объектами в памяти человека
Материальное представление	
$Зн_2$ – материализованные знания (учебники, справочники и т.д.)	$Д_2$ – зафиксированные данные (графики, таблицы и т.д.)
Абстрактное представление	
$Зн_3$ – поле знаний (слабоформализованное описание $Зн_1$ и $Зн_2$)	$Д_3$ – модель данных (схема связей между объектами)
Машинное представление	
$Зн_4$ – знания на языках представления знаний (формализация $Зн_3$)	$Д_4$ – данные на языке описания данных
Машинное хранение	
$Зн_5$ – база знаний	$Д_5$ – база данных

Окончание табл. 3.1

Преобразование знаний	
$Зн_1$ – $Зн_3$ – $Зн_5$ (Знания – Поле знаний – База знаний)	$Д_1$ – $Д_3$ – $Д_5$ (Внешние данные – Логическая модель – Физические данные)

Вышеприведенные качественные свойства знаний на сегодня характерны скорее для знаний в аспекте человеческого мышления, нежели машинного [10].

В области искусственного интеллекта знания характеризуются как закономерности предметной области, полученные в результате практической деятельности и профессионального опыта, позволяющие ставить и решать задачи в этой области. Как показывает опыт, процесс формализации таких знаний обладает рядом специфических для знаний сложностей, обусловленных наличием НЕ-факторов [13]. В первую очередь, НЕ-факторы обусловлены нечеткостью и размытостью информации (большая температура, сильный ветер и т.п.), что затрудняет однозначную интерпретацию. Во-вторых, при обработке результатов измерений возникает неточность знаний, которые могут быть интерпретированы как полностью истинные или полностью ложные, что является некорректным. И, наконец, всегда существует элемент субъективности эксперта, формирующего базу знаний.

Все это положило начало развитию нечеткой логики и мягким вычислениям [14]. Эта область искусственного интеллекта будет описана в гл. 7.

1 С точки зрения классификации знаний можно выделить четыре классификации:

2 По глубине:

3 Поверхностные (совокупность эмпирических ассоциаций между понятиями предметной области);

4 Глубинные (образы, аналогии, отражающие взаимосвязь отдельных понятий предметной области).

5 По жесткости:

6 Жесткие (позволяющие получать однозначные четкие решения);

7 Мягкие (допускающие несколько решений или неоднозначное нечеткое решение).

8 По функциональности:

9 Знания о процессах решения задач;

10 Знания о языке общения;

11 Знания о способах представления информации.

12 По степени точности [11]:

Формализованные (образованные в виде строгих суждений – законов, формул и алгоритмов);

Неформализованные (отличающиеся конкретностью, субъективностью и приблизительностью).

Место хранения и представления знаний называется базой знаний. База знаний является ключевым звеном всех интеллектуальных систем. Однако база знаний – не единственное отличие интеллектуальных систем от неинтеллектуальных. Дело в том, что стиль программирования интеллектуальных систем не похож на таковой в традиционном понимании (Таблица 3.2).

Таблица 3.2

Отличительные особенности программирования интеллектуальных систем

Тип системы	**Характерные параметры**						
	Обработка	Метод решения	Задание алгоритма	Искомое решение	Связь управления и данных	Изменчивость	Степень достоверности ин-
Интеллектуальная	Символьная	Эвристический поиск	Неявное	Удовлетворительное	Перемешаны	Частая	Достоверная и недостоверная
Традиционная	Числовая	Строгий алгоритм	Точное	Оптимальное	Разделены	Редкая	Только достоверная

В интеллектуальных системах классическая модель ДАННЫЕ + АЛГОРИТМЫ = ПРОГРАММА заменяется на ЗНАНИЯ + ВЫВОДЫ = СИСТЕМА.

3.2 Экспертные системы

Рассмотрим один из наиболее популярных и наглядных примеров систем, использующих знания – экспертные системы [10].

Экспертные системы (ЭС) – интеллектуальные системы, основанные на применении и обработке знаний, требующих обращений к опыту

эксперта, накопленному в результате его профессиональной деятельности.

Есть еще и другое определение ЭС:

Экспертная система — это искусственная система, которая оперирует со знаниями в определенной предметной области с целью выработки рекомендаций или решения проблем.

Среди известных всем наглядных примеров экспертных систем служит игра Акинатор. Акинатор позволяет почти всегда угадывать известных персонажей за счет постоянно пополняемой базы знаний.

Экспертная система позволяет решить задачи, которые обладают одной из следующих характеристик:

1) задачи не могут быть представлены в числовой форме;

2) исходные данные обладают неточностью, неоднозначностью и противоречивостью;

3) цели нельзя точно выразить через функцию;

4) не существует однозначного алгоритма решения (алгоритм сам является решением);

5) алгоритмическое решение существует, но является многозатратным по времени и вычислительным ресурсам.

Другими словами, экспертная система позволяет оперировать слабоструктурированными знаниями.

Основными элементами ЭС считаются база знаний, логика вывода и интерфейс пользователя (Рис. 3.1).

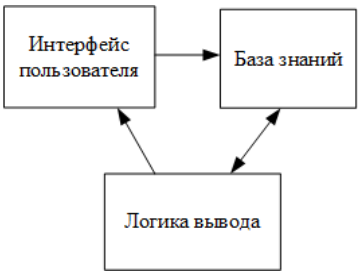


Рис. 3.1. Упрощенная структура экспертной системы

Знания в ЭС – это формализованная информация, которая используется в процессе логического вывода. Знания в ЭС бывают статическими, которые введены в систему на этапе проектирования, и динамические, которые можно охарактеризовать как опыт, или знания, полученные в процессе функционирования.

В общем смысле базу знаний составляют множества фактов и правил. Под фактами подразумеваются знания типа «*A* это *A*», они характерны для баз данных. Правила (продукции) представляют знания вида «ЕСЛИ-ТО». Правила позволяют вывести новые факты.

В частности, для характеристики базы знаний используют различные модели представления знаний. В Таблица 3 приведено расширенное описание базовых моделей представления знаний. При этом в качестве основы

экспертных (и других интеллектуальных) систем чаще всего используют продукционные, семантические и фреймовые модели.

Логика вывода – программный модуль, использующий базу знаний. Этот модуль позволяет применить различные алгоритмы вывода решения.

Стоит отметить, что, начиная с 2000-х годов большой популярностью пользуются так называемые скелетные ЭС, в которых реализована логика вывода, но база знаний пуста. В этом случае база знаний пополняется постепенно пользователем с помощью специальных редакторов в пользовательском интерфейсе.

Скелетные ЭС – системы с пустой базой данных, которая далее формируется пользователем с помощью специального интерфейса.

Таблица 3.3

Описание базовых моделей представления знаний

Разряд модели		Тип модели	Комментарий
Логические модели		Дедуктивная	Решаемая проблема записывается в виде утверждений некоторой формальной системы (например, исчисления предикатов первого порядка). Цель проблемы также записывается в виде утверждений, справедливость которых нужно установить или опровергнуть на основании аксиом и правил вывода
		Индуктивная	Здесь для получения общих выводов из наличия совокупности частных утверждений используется другой механизм, который может быть либо вероятностным, либо логическим в зависимости от специфики изучаемого явления
		Псевдофизические логики	Класс дедуктивных формальных систем, отличающихся тем, что в качестве пропозициональных переменных используются лингвистические переменные, основанные на нечетких множествах или порядковые шкалы
Эвристи-	Сетевые	Функциональные сети	Сети этого типа отражают некоторую декомпозицию определенной вычислительной или информационной процедуры, а дуги показывают функциональную связь между частями, возникающими в результате декомпозиции (например, блок-схема программы и т.п.)

Окончание табл. 3.3

		Сценарии	Однородные сети, в которых в качестве единственного отношения выступает отношение нестрогого порядка, семантика которого может быть различна (например, все возможные последовательности событий – это сетевой трафик и т.п.)
		Семантические сети	Ориентированный граф с помеченными дугами и состояниями (т.е. вершины сети могут иметь различную интерпретацию, а дуги-отношения принадлежат к различным типам, логическим, лингвистическим, теоретико-множественным, квантифицированным)
		Фреймы	Формализованная модель для отображения стереотипных, стандартных ситуаций, своеобразная реализация абстрактного мышления
	Продукционные	Продукции	Модель, основанная на правилах, позволяющих представить знания в виде правил типа «ЕСЛИ-ТО»

Проблематикой составления экспертных систем на сегодняшний день занимается целая дисциплина, которая называется инженерией знаний.

Инженерия знаний – дисциплина, занимающаяся разработкой экспертных систем, формированием баз знаний и выбором соответствующих алгоритмов вывода.

Специалисты в этой области занимаются поиском, структуризацией и обработкой знаний, а также выбором соответствующих алгоритмов вывода и разработкой интерфейса.

При разработке ЭС в первую очередь здесь стоит отметить проблему представления знаний. Для экспертных систем необходимым является обеспечение независимости продукций. При создании моделей представления знаний следует учитывать также такие факты, как однородность представления и простота понимания.

К проблеме представления знаний традиционно относят разработку формальных языков и программных структур для отображения когнитив-

ных структур (или структур человеческого сознания, отражающих представление личности об окружающей действительности). Также к этой проблеме относят исследования по составлению базовых структур (или концептов) знаний для различных объектов (графических или текстовых).

Принято выделять два уровня развития экспертных систем. ЭС первого поколения позволяют лишь повторить логический вывод эксперта. Однако таких систем недостаточно для того, чтобы ЭС выступила в роли полноценного помощника для решения интеллектуальных задач, в связи с чем разрабатываются системы второго поколения, которые позволяют проводить анализ нечисловых данных, отбрасывать и принимать гипотезы, оценивать достоверность фактов, самостоятельно пополнять базу знаний и т.п.

Экспертные системы на сегодня пользуются большой популярностью. Наибольшее распространение они получили при разработке интегральных микросхем и в автоматизации программирования. В настоящее время также ведутся разработки ЭС, позволяющих прогнозировать различные экономические и политические события, разрабатывать законопроекты, планировать распределение ресурсов.

ГЛАВА 4. МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

4.1 Продукционные модели представления знаний

Психологические исследования процессов мышления человека показали, что наиболее подходящей моделью представления рассуждений является модель, основанная на правилах (или продукционная модель). В этой связи правила зачастую являются одним из основных компонентов при построении систем искусственного интеллекта. Таким образом, при рассмотрении большей части существующих интеллектуальных системы окажется, что на самом общем уровне их можно представить в виде системы [15]:

$$PS = < F, P, I >,$$

где F – рабочая память, называемая также базой данных или базой фактов;

P – база правил, которую также называют базой знаний;

I – интерпретатор (решатель), содержащий стратегию управления (в соответствии с которой происходит выбор правил – вывод).

Множество правил позволяет представить знания в виде набора правил типа «ЕСЛИ-ТО», иначе называемых продукционными правилами (или продукциями). Продукции состоят из условия (антецедента) и следствия (консеквента).

Для примера в Таблица 4.1 представлены типичные правила различных видов экспертных систем, основанных на продукционных моделях.

Таблица 4.1

Прикладные продукционные модели знаний

Диагностическая ЭС *MYCIN*	ЕСЛИ: Окраска бактерий грамположительная И Морфология бактерий характерна для кокков И форма колоний – цепочки, ТО: Есть основание считать (0,7), что вид бактерий – стрептококк
ЭС проектирования конфигураций *XCON*	ЕСЛИ: Текущий контекст – подвод шины питания И Известно положение разъемов модуля И Существует свободное место для подвода шины питания И Существует шина питания, ТО: Подвести шину в свободное место

Окончание табл. 4.1

Планирующая ЭС *TATR*	ЕСЛИ: на аэродроме обнаружены под открытым небом самолеты И число их больше 0,25 X (X – общее число самолетов на аэродроме), ТО: Принять оценку ситуации на аэродроме равной ПРЕВОСХОДНО
Мониторинговая ЭС *REACTOR*	ЕСЛИ: Передача тепла от первичной системы охлаждения ко вторичной системе охлаждения недостаточна И расход воды в системе подпитки невелик ТО: Неисправность состоит в утечке подпитывающей воды

Правило P применяется к рабочей памяти F. Его можно интерпретировать как команду исполнительному органу, которая разворачивается в последовательность действий по отношению к рабочей памяти. В состав каждого правила входит некоторое условие, которому текущее состояние рабочей памяти может удовлетворять, либо не удовлетворять. В случае выполнения условия правило применяется, изменяя состояние рабочей памяти и, тем самым, отражая в ней изменения, произошедшие в модели мира при выполнении соответствующего правила. Применение правила также называется эффектом действия.

Формально правило может быть представлено тройкой (или кортежем) [6]

$$\Pi = < C, A, D >, \quad (1)$$

где C – множество условий правила;

A – множество добавляемых фактов;

D – множество удаляемых фактов.

Для представления конструкций правила в основном применяют язык исчисления предикатов первого порядка. Другими словами, каждое множество из представленного кортежа является множеством атомарных формул.

Таким образом, атомарные формулы из множеств кортежа П превращаются в факты в процессе применения соответствующих подстановок констант (m_1, m_2, …, m_n) вместо переменных (x_1, x_2, …, x_n) и проверки для каждой атомарной формулы $P(x_1, x_2, \ldots, x_n)$ из C принадлежности типа (m_1, m_2, …, m_n) $\in R(P)$, т.е. проверки его выполнимости, где $R(P)$ – интерпретирующее отображение (множество истинных отношений, сопоставленных предикатным символам). Из этого следует, что правило считается выпол-

нимым (применимым) в текущем состоянии, если каждая атомарная формула его множества условий истинна в этом состоянии (при этом также говорят, условие правила является выполнимым). При анализе цепочки правил принято говорить о применимости последовательности правил. В этом случае условие каждого правила последовательности является выполнимым в состоянии рабочей памяти, полученном в результате применения предыдущего правила.

После установления выполнимости правила значения, подставляемые на места свободных переменных во множестве условий *C*, также подставляются на место переменных во множествах добавляемых фактов *A* и множествах удаляемых фактов *D*, после чего либо добавляются в рабочую память, либо удаляются из нее.

Стоит отметить, что рабочая память должна быть согласована со множеством правил, т.е. для любого правила представлять интерпретацию в виде множества конкретных значений предметной области. Формально это звучит так:

$$\forall P(x, y, \ldots, z) \in C \cup A \cup D, \exists R(P) \subseteq M^n,$$

где $R(P)$ – интерпретирующее отображение;

M – множество конкретных значений предметной области.

Интерпретатор предназначен для организации процесса вывода заключения путем исполнения стратегии управления, основанной на *Modus* ponens (правило отделения) и в общем виде представляется последовательностью шагов (Рис. 4.1):

1 Выбрать очередное правило из множества правил;
2 Проверить выполнимость правила в текущем состоянии рабочей памяти;
3 Если условие правила выполнимо, поместить правило в конфликтное множество;
4 Если множество применимых правил исчерпано, выбрать какое-либо правило из конфликтного множества и применить его;
5 Перейти к шагу 1.

Отметим важное для продукционных моделей определение: *конфликтным множеством называется множество правил, выполнимых в некотором состоянии рабочей памяти.*

Критерием остановки алгоритма вывода, представленного выше, является пустое конфликтное множество или достижение целевого состояния.

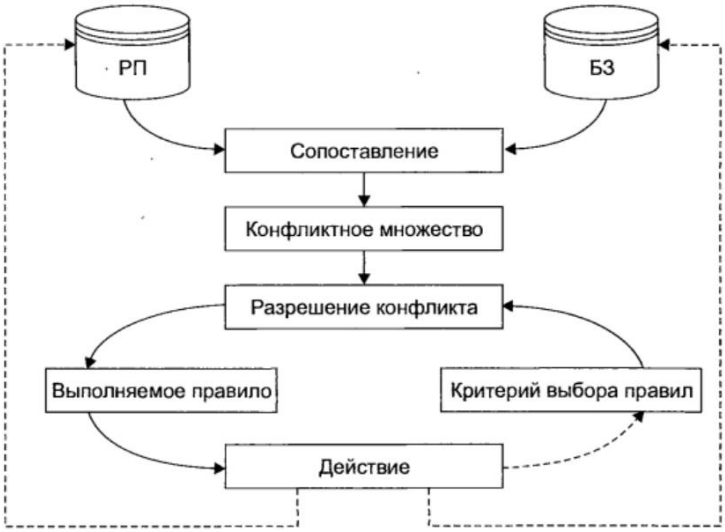


Рис. 4.1. Схематическое изображение процесса вывода в продукционных системах

Несложно заметить, что стратегия управления продукционной системы порождает недетерминированный процесс (т.е. с неизвестным заранее исходом), поскольку точно не устанавливает, как именно следует выбирать правило из множества продукций. Поэтому зачастую продукционной системе изначально недостаточно информации для вывода, и исследования в этой области привели к толчку в развитии так называемых поисковых алгоритмов (или алгоритмов оптимизации). Следуя таким алгоритмам, система продолжает проверку выполнимости правил до тех пор, пока не будет достигнут определенный критерий поиска. Популярными алгоритмами в этой сфере являются эвристики (напомним, что эвристики – это алгоритмы выбора без достаточных теоретических обоснований), которые позволяют значительно сократить область поиска нужного правила и избежать полный перебор вариантов.

Стратегия управления может реализовываться как в прямом (прямой вывод), так и в обратном (обратный вывод) порядке. В случае прямого вывода (вывода, управляемого данными) по известным фактам отыскивается применимое правило, консеквент которого заносится в рабочую память. Представленный выше алгоритм вывода является прямым.

При обратном выводе (выводе, управляемом целями) вначале выдвигается некоторое множество гипотез (целей), а затем в рабочей памяти осуществляется поиск таких фактов, которые могли бы их подтвердить или опровергнуть. Процесс поиска зачастую занимает значительное число шагов (с применением множества правил), при этом не исключается выдвижение новых гипотез. Такой вывод применяется только в случае заранее известного небольшого количества целей, так как его алгоритм несколько отличается от алгоритма прямого вывода и выглядит следующим образом:

1 Выбрать какую-либо (зачастую следующую по порядку) цель из множества целей.

2 Выбрать какое-либо правило из множества, где консеквентом является цель.

3 Проверить наличие антецедентов выбранного правила в рабочей памяти. Если все антецеденты присутствуют в рабочей памяти, добавить цель в рабочую память и удалить из множества целей. Перейти к шагу 1.

4 Если антецедент не присутствует в рабочей памяти, добавить его во множество целей.

5 Перейти к шагу 1.

Критерием остановки в данном случае является пустое множество целей.

4.2 Разрешение конфликтного множества в продукционных моделях

Важнейшей задачей для продукционных моделей с прямым выводом является правильный выбор оптимальных правил из конфликтного множества. Другими словами, необходимо разрешить конфликтное множество. Разрешение конфликтных множеств производится путем выбора тех или иных стратегий управления. Можно выделить два типа стратегий: *безвозвратный и пробный* [16]. В случае безвозвратной стратегии выбранное правило исполняется, после чего его невозможно пересмотреть в дальнейшем. Безвозвратные стратегии используются главным образом в случаях, когда последовательность правил не играет роли (например, при игре в пятнашки). Множество правил (обозначим его как $\{П_i\}$), представляющих такую последовательность, называются коммутативными и обладают следующими свойствами:

а) каждое из правил множества $\{П_i\}$, применимое к состоянию S^0 рабочей памяти, применимо также к состоянию S^1, полученному при выполнении любого правила множества $\{П_i\}$.

б) состояние, удовлетворяющее целевое состояние, будет переводиться любым выполнимым правилом из $\{П_i\}$ в состояние, также удовлетворяющее целевое состояние (под словом «удовлетворяет целевому» понимается, что множество фактов целевого состояния является подмножеством фактов удовлетворяющего состояния).

в) в любой применимой последовательности правил перемена мест правил не приводит к изменению результирующего состояния.

Из этих трех свойств вытекает также то свойство, что если существует последовательность правил из множества $\{П_i\}$ такая, что переводит начальное состояние рабочей памяти в целевое, то любая другая последовательность правил из множества $\{П_i\}$ может быть продолжена таким образом, что будет переводить рабочую память из начального состояния в состояние, которое удовлетворяет целевое состояние.

Пробная стратегия резервирует возможность возврата к исходному состоянию с использованием альтернативного правила. Такая стратегия

управления применима в случаях, когда применение неудачного правила блокирует дальнейшие вычисления (например, при решении известной задачи с перевозкой на соседний берег волка, козы и капусты, см. ниже). Среди пробных режимов различают режим с возвратом, где запоминается точка возможного возврата, и режим поиска на графе, где происходит выполнение сразу нескольких правил и запоминание цепочек с дальнейшим анализом полученной графовой структуры. Рассмотрим эти два режима подробнее.

Стратегия с возвращениями основана на том, что запоминает состояние, в котором формируется конфликтное множество правил. Из этого конфликтного множества выбирается некоторое правило *П* и применяется. К нему применяется следующее применимое правило и т.д. Если этот путь не ведет к решению, то все сделанные шаги «забываются» и вместо правила *П* выбирается другое правило.

Стратегии с возвращениями могут использоваться независимо от того, насколько полной информацией для выбора мы обладаем. Если такая информация вообще отсутствует, то правила выбираются в соответствии с произвольной схемой. Однако в таком случае потребуется гораздо больше вычислительных затрат, что естественным образом негативно сказывается на практической применимости стратегии с возвращениями. Поэтому при разработке стратегии с возвращениями следует выбрать один из следующих способов выбора правила из конфликтного множества:

1) определить функцию оценки примененного правила, значение которой позволит установить степень «актуальности» выбора правила.
2) ввести меру близости целевого и полученного состояния.
3) определить «запрещенные» состояния (в которые не следует переходить).

Существуют и другие способы выбора правила, однако по мнению авторов перечисленные методики являются наиболее популярными.

Стратегию управления с поиском на графе можно рассматривать как средство нахождения пути на графе от вершины, являющейся исходным состоянием рабочей памяти, к вершине, являющейся целевым состоянием рабочей памяти. Классическим наглядным примером такой стратегии является стратегия решения задачи о волке, козе и капусте (Рис. 4.2). Сама задача звучит так:

«Крестьянину нужно перевезти через реку волка, козу и капусту. Но лодка такова, что в ней может поместиться только крестьянин, а с ним или один волк, или одна коза, или одна капуста. Но если оставить волка с козой, то волк съест козу, а если оставить козу с капустой, то коза съест капусту».

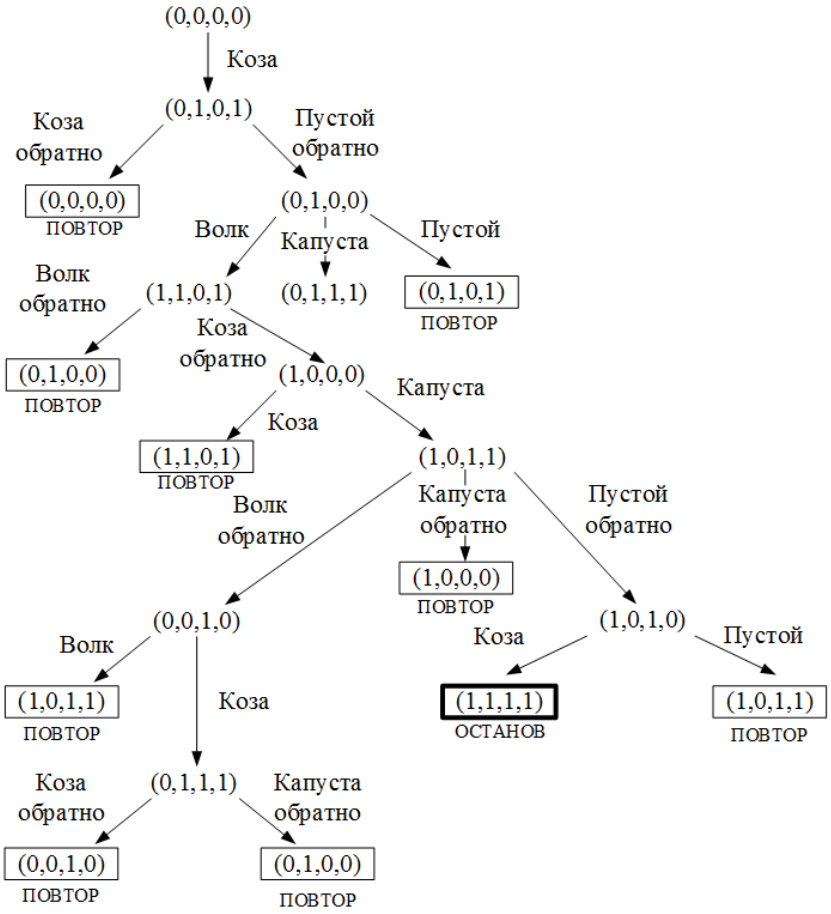


Рис. 4.2. Решение задачи о волке, козе и капусте с помощью графа (последовательность (*a*, *b*, *c*, *d*) указывает на факт нахождения волка, козы, капусты и крестьянина, соответственно на нужном берегу)

Чтобы исключить полный перебор всех ветвей, целесообразным является каждой дуге графа (и/или вершине) приписать положительное число $c(i,j)$ – ее стоимость (вес). Такая стоимость может равняться сумме стоимостей предыдущих дуг, сумме стоимостей предыдущих вершин или предопределена частным алгоритмом построения стратегии. Задача поиска на графе простейшего типа сводится к поиску пути (скорее всего с минимальной стоимостью) между вершиной *s*, соответствующей начальному состоянию, и вершиной *t*, удовлетворяющему целевое состояние. В большинстве случаев необходимо найти путь l_i от вершины *s* до вершины $t_i \in \{t_n\}$, который является кратчайшим.

4.3 Примеры использования и преимущества продукционных систем

Рассмотрим пример прямого вывода для решения задачи построения одной башни из *n* одинаковых кубиков, лежащих на земле. Пусть у нас имеются два правила:

$П_1$:

Если первый кубик лежит на земле и не находится под кубиком, то поднять его и поставить на второй кубик, не находящийся под кубиком.

$П_2$:

Если первый кубик лежит на земле и не находится под кубиком и если второй кубик не находится под кубиком и лежит на третьем кубике, то поднять первый кубик и поставить на второй кубик.

Для формализации задачи обозначим множество кубиков как $M = \{m_1, m_2, \ldots, m_n\}$ введем три предикатных символа:

On – двуместный предикатный символ «находиться на»;

Em – одноместный предикатный символ «не находиться под кубиком»;

Er – одноместный предикатный символ «находиться на земле».

Тогда правила могут быть представлены по формуле (1) как:

$П_1 = <C_1, A_1, D_1>$,

где$C_1 = \{\text{Er}(x), \text{Em}(x), \text{Em}(y)\}$,

$A_1 = \{\text{On}(x,y)\}$,

$D_1 = \{\text{Er}(x), \text{Em}(y)\}$.

$П_2 = <C_2, A_2, D_2>$,

где$C_2 = \{\text{Er}(x), \text{Em}(x), \text{Em}(y), \text{On}(y,z)\}$,

$A_2 = \{\text{On}(x,y)\}$,

$D_2 = \{\text{Er}(x), \text{Em}(y)\}$.

Наглядно представим начальное (Таблица 4.2) и целевое (табл. 4.3) состояние рабочей памяти в виде таблиц интерпретирующего отображения предикатных символов (так как кубики равнозначны, порядок нумерации элементов множества *M* не имеет значения).

Таблица 4.2

Начальное состояние рабочей памяти для «мира кубиков»

R(On)	$\emptyset$
R(Em)	$\{m_1\}, \{m_2\}, \ldots, \{m_n\}$
R(Er)	$\{m_1\}, \{m_2\}, \ldots, \{m_n\}$

Таблица 4.3

Целевое состояние рабочей памяти для «мира кубиков»

R(On)	$\{m_2, m_1\}, \{m_3, m_2\}, \ldots, \{m_n, m_{n-1}\}$
R(Em)	$\{m_n\}$
R(Er)	$\{m_1\}$

Применим стратегию управления прямого вывода:

Шаг 1. Выбираем $П_1$.

Шаг 2. Для проверки выполнимости условия C_1 подставим m_2 и m_1 вместо *x* и *y* в атомарных формулах. Условие C_1 выполнимо для текущего состояния, так как предикаты $Er(m_2)$, $Em(m_2)$, $Em(m_1)$ истинны.

Шаг 3. Помещаем $П_1$ в конфликтное множество.

Шаг 4. $П_2$ не является применимым, так как $R(\text{On}) = \emptyset$. Применяя $П_1$, переведем рабочую память в состояние, отображения которого представлены в Таблица .

Шаг 5. Переходим к шагу 1.

И т.д.

Таблица 4.4

Состояние рабочей памяти для «мира кубиков» после первого прохода

R(On)	$\{m_2, m_1\}$
R(Em)	$\{m_2\}, \{m_3\}, \ldots, \{m_n\}$
R(Er)	$\{m_1\}, \{m_3\} \ldots, \{m_n\}$

Отметим, что на втором цикле конфликтное множество будет состоять из двух правил, поэтому требуется уточнить словосочетание «какое-либо правило» на четвертом шаге. Тут целесообразным будет использовать критерий большей информативности. Иначе говоря, выбор должен пасть на то правило, которое содержит больше атомарных формул в условии С, т.е. $П_2$. Выбор такой эвристики приведет к выбору второго правила и на всех последующих шагах вплоть до достижения целевого состояния.

Стоит обратить особое внимание, что алгоритм строительства башни из кубиков изначально задан не был, в связи с чем можно с полной уверенностью назвать описанную выше систему интеллектуальной.

Рассмотрим далее алгоритм обратного вывода на примере задачи покупки легкового автомобиля [17], где база правил имеет вид:

$П_1$. Если у человека много денег и ему нужен автомобиль, то нужно ехать в автосалон.

$П_2$. Если у человека мало денег и ему нужен автомобиль, то нужно ехать на рынок подержанных автомобилей.

$П_3$. Если имеется время и нужно ехать в автосалон, то необходимо определиться с маркой покупаемой машины.

$П_4$. Если имеется время и нужно ехать на рынок, то необходимо определиться с пробегом покупаемой машины.

$П_5$. Если человек приехал в автосалон и определился с маркой, то нужно произвести покупку нового автомобиля.

$П_6$. Если человек приехал на авторынок и определился с пробегом, то нужно произвести покупку б/у автомобиля.

Обозначим множество термов как $M = \{m_i\}$ (здесь уже порядок имеет смысл в отличие от предыдущего примера), где:

m_1 – деньги,

m_2 – автомобиль,

m_3 – автосалон,
m_4 – рынок,
m_5 – время,
m_6 – марка,
m_7 – пробег,
m_8 – новый автомобиль,
m_9 – автомобиль с пробегом.
Введем одноместные предикатные символы:
Mn – много;
Lt – мало;
Nd – нужен;
Gt – ехать в (на);
Hs –имеется;
Df – определиться;
Buy – купить.
Формально правила могут быть представлены и как в предыдущем примере, однако нагляднее в данном случае будет использовать другую принятую форму представления продукций:

П$_1$. *IF Mn*(m_1) *AND Nd*(m_2) *THEN Gt*(m_3).
П$_2$. *IF Lt*(m_1) *AND Nd*(m_2) *THEN Gt*(m_4).
П$_3$. *IF Hs*(m_5) *AND Gt*(m_3) *THEN Df*(m_6).
П$_4$. *IF Hs*(m_5) *AND Gt*(m_4) *THEN Df* (m_7).
П$_5$. *IF Gt*(m_3) *AND Df*(m_6) *THEN Buy*(m_8).
П$_6$. *IF Gt*(m_4) *AND Df*(m_7) *THEN Buy*(m_9).

Пусть выдвинута гипотеза «Покупка б/у авто» (или Buy(m_9)) и заданы начальные условия «Мало денег», «Нужен автомобиль», «Имеется время» (Таблица 4.5).

Таблица 4.5

Состояние рабочей памяти для ситуации покупки автомобиля при частной выдвинутой гипотезе

R(Mn)	∅
R(Lt)	$\{m_1\}$
R(Nd)	$\{m_2\}$
R(Gt)	∅
R(Hs)	$\{m_5\}$
R(Df)	∅
R(Buy)	∅

Применим стратегию управления обратного вывода:
Шаг 1. Выбираем *Buy*(m_9)
Шаг 2. Выбираем *П*$_6$.
Шаг 3. *Gt*(m_4) и *Df*(m_7) отсутствуют в рабочей памяти.

Шаг 4. Добавляем $Gt(m_4)$ и $Df(m_7)$ во множество целей.

Шаг 5. Переходим к шагу 1.

Шаг 1. Выбираем $Gt(m_4)$.

Шаг 2. Выбираем $П_2$.

Шаг 3. Антецеденты из $П_2$ присутствуют в рабочей памяти. Добавляем $Gt(m_4)$ в рабочую память и удаляем из множества целей (Таблица 4.6). Переходим к шагу 1.

Шаг 1. Выбираем $Df(m_7)$.

Шаг 2. Выбираем $П_4$.

Шаг 3. Антецеденты из $П_4$ присутствуют в рабочей памяти. Добавляем $Df(m_7)$ в рабочую память и удаляем из множества целей (табл. 4.7). Переходим к шагу 1.

Шаг 1. Выбираем $Buy(m_9)$

Шаг 2. Выбираем $П_6$.

Шаг 3. Антецеденты из $П_6$ присутствуют в рабочей памяти. Добавляем $Buy(m_9)$ в рабочую память и удаляем из множества целей (табл. 4.8). Критерий останова достигнут.

Таким образом, основная гипотеза подтвердилась достоверными фактами.

Таблица 4.6

Состояние рабочей памяти для ситуации покупки автомобиля после первого прохода

R(Mn)	$\emptyset$
R(Lt)	$\{m_1\}$
R(Nd)	$\{m_2\}$
R(Gt)	$\{m_4\}$
R(Hs)	$\{m_5\}$
R(Df)	$\emptyset$
R(Buy)	$\emptyset$

Таблица 4.7

Состояние рабочей памяти для ситуации покупки автомобиля после второго прохода

R(Mn)	$\emptyset$
R(Lt)	$\{m_1\}$
R(Nd)	$\{m_2\}$
R(Gt)	$\{m_4\}$
R(Hs)	$\{m_5\}$
R(Df)	$\{m_7\}$
R(Buy)	$\emptyset$

Таблица 4.8

Терминальное состояние рабочей памяти для ситуации покупки автомобиля

R(Mn)	$\emptyset$
R(Lt)	$\{m_1\}$
R(Nd)	$\{m_2\}$
R(Gt)	$\{m_4\}$
R(Hs)	$\{m_5\}$
R(Df)	$\{m_7\}$
R(Buy)	$\{m_9\}$

Перечислим далее достоинства и недостатки продукционной модели знаний.

При разработке небольших систем, состоящих из нескольких десятков правил, проявляются в основном положительные стороны систем продукций, однако при увеличении объема знаний более заметными становятся слабые стороны.

Основные достоинства продукционных систем связаны с простотой представления знаний и организации логического вывода.

К недостаткам систем продукций можно отнести следующие:

- отличие от структур знаний, свойственных человеку;
- неясность взаимных отношений правил;
- сложность оценки целостного образа знаний;
- низкая эффективность обработки знаний.

Несмотря на это, на сегодняшний день продукционные модели из-за своей наглядности, модульности, легкости внесения изменений, а также простоты вывода являются наиболее популярными при разработке промышленных систем обработки знаний (более 80 % [10]).

4.4 Семантические сети

Наряду с продукционными моделями популярным инструментом представления знаний в интеллектуальных системах являются сетевые модели.

Сетевые модели подразделяются на множество различных видов, среди которых наиболее популярными являются семантические сети и системы фреймов. Толчок к развитию семантических сетей дало развитие сети Интернет. Основным достоинством, резко отличающим семантические сети от продукционных правил, является возможность отражения синтаксиса и семантики в одном представлении благодаря использованию не только обозначений элементов, но и самих элементов, а также связей, сопоставляющих элементам их интерпретации. Иначе говоря, семантические сети позволяют представить наиболее общую картину знаний [18].

Само понятие «семантика» характеризует науку об отношениях между символами и объектами, которые эти символы обозначают.

Понятие семантической сети возникло в 1966 году в работах Росса Квиллиана [6]. Трактовок существует достаточное множество [10], но суть остается прежней:

Семантическая сеть – это ориентированный граф, составленный из множества вершин, характеризующих понятия, и дуг, раскрывающих взаимосвязи между понятиями.

Простейшая форма семантической сети может быть представлена в виде реализации высказывания «Евгений дал Марии книгу» и «Евгений и Мария являются людьми» [19] (Рис. 4.3).

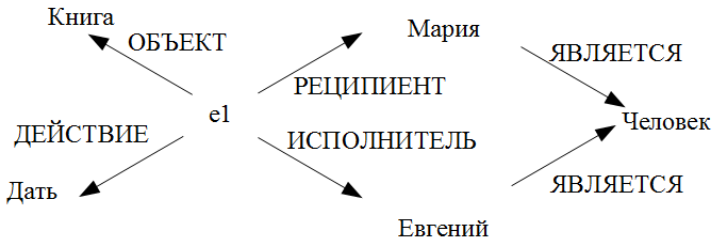


Рис. 4.3. Пример семантической сети

Выражение «*e*1» в данной сети представляет собой событие «передача книги», где исполнителем является Евгений, принимающим лицом (реципиентом) является Мария, а объектом – книга.

Если попытаться выразить вышеприведенную сеть в виде предикатов, она примет следующую форму:

ОБЪЕКТ (*e*1, книга);

ДЕЙСТВИЕ (*e*1, передача);

ИСПОЛНИТЕЛЬ (*e*1, Евгений);

РЕЦИПИЕНТ (*e*1, Мария);

ПРИНАДЛЕЖНОСТЬ (Евгений, Человек);

ПРИНАДЛЕЖНОСТЬ (Мария, Человек).

Таким образом, в этом случае названия дуг можно охарактеризовать предикатными символами, а актанты (то есть участники ситуации, аргументы) – это вершины. В общем, в семантических сетях выделяют три вида основных вершин [10, 15]:

1 Вершины-ситуации (состояния, процессы и пр.), выражаемые предикатами;

2 Вершины-понятия (абстрактные и физические);

3 Вершины-характеристики.

В качестве дуг могут выступать:

1 Теоретико-множественные отношения (элемент-множество, часть-целое, множество-подмножество и т.п.);

2 Логические отношения (И, ИЛИ, НЕ);

3 Квантифицированные отношения (∀, ∃);

4 Лингвистические отношения (ДЕЙСТВИЕ, ИСПОЛНИТЕЛЬ, ЯВЛЯЕТСЯ и т.д.).

Кроме простых предикатов, семантические сети представимы в так называемой клаузальной форме [19]. Так, высказывание «Евгений дает книгу всем, кто ему нравится» представимо в виде клаузы «заключение» ← «условие»:

ДАТЬ (Евгений, книга, *x*) ← НРАВИТСЯ (Евгений, *x*).

Символ «←» характеризует логическую связку «ЕСЛИ». В общем случае клаузы содержат несколько условий и несколько альтернативных заключений. Например, клауза

ДОВЕРЯЕТ(Евгений, *x*), РОДСТВЕННИК(Евгений, *x*) ← ДАТЬ(Евгений, *y, x*), НУЖЕН(Евгений, *y*)

Означает, что ЕСЛИ Евгений отдает вещь, которая ему нужна, другому человеку, то этот человек является либо родственником Евгения, либо Евгений доверяет ему.

Иначе говоря, дизъюнкция альтернатив заключений, которая следует из конъюнкции условий. При этом количество и дизъюнктов, и конъюнктов может варьироваться от 0 до бесконечности. Клауза без условий является безусловным утверждением, а клауза без заключений является опровержением. При отсутствии и тех и других клауза является тождественно ложной. На основе вышесказанного сформируем формальное определение клаузы:

Клауза – это выражение вида

$A_1, A_2, \ldots, A_n \leftarrow B_1, B_2, \ldots, B_m,$

где$A_1, A_2, \ldots, A_n$ – заключения $(n \geq 0)$, выраженные в предикатной форме (атомарные формулы),

$B_1, B_2, \ldots, B_m$ – условия (посылки) $(m \geq 0)$, выраженные в предикатной форме (атомарные формулы).

Если $m = 0$ и клауза содержит k переменных $x_1, \ldots, x_k$, то

$\forall x_1, \forall x_2, \ldots, \forall x_k \, (A_1 \vee A_2 \vee \ldots \vee A_n)$.

Если $n = 0$ и клауза содержит k переменных $x_1, \ldots, x_k$, то

$\neg\exists x_1, \neg\exists x_2, \ldots, \neg\exists x_k \, (B_1 \wedge B_2 \wedge \ldots \wedge B_m)$.

Если $n = m = 0$, то клауза является тождественно ложной.

Если клауза содержит не более одной атомарной формулы в заключении, то она называется клаузой Хорна или Хорновской клаузой.

Семантические сети в зависимости от свободности переменных в клаузе могут быть как простыми и расширенными [6].

Простая семантическая сеть рассматривается как форма записи утверждений клаузальной логики без свободных переменных. Например, клауза $P(a, b) \leftarrow$ в семантической сети обозначается как дуга, помеченная меткой P и направленная из a в b (Рис. 4.4).

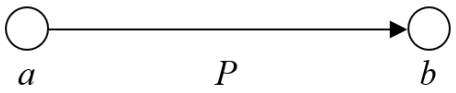


Рис. 4.4. Простая семантическая сеть

Представленная на Рис. 4.3 семантическая сеть также является простой.

Расширенная семантическая сеть позволяет формировать клаузы из утверждений, представленных простыми сетями (Рис. 4.5).

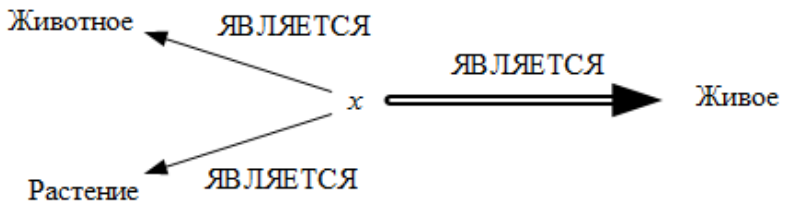


Рис. 4.5. Расширенная семантическая сеть

На Рис. 4.5 представлена клауза, представляющая выражение «Любое живое существо является либо животным, либо растением».

Расширенные семантические сети позволяют представлять функциональные термы в виде единичных вершин. Например, Рис. 4.6 представляет нам клаузу: «для того, чтобы что-то дать, надо что-то взять взамен». При этом атомарные формулы, соответствующие условиям клауз, описываются двойными дугами, а заключения – одинарными.

Рис. 4.6. Представление функциональных термов в семантической сети

При наличии множества клауз возникает проблема соотношения дуг и клауз. Такая проблема решается двумя способами:

1) Каждой дуге присваивается номер соответствующей клаузы.

2) Сеть разделяется на подсети, каждая из которых представляет отдельную клаузу.

Для примера представим расширенную семантическую сеть на Рис. 4.7, содержащую простую семантическую сеть из Рис. 4.3, где клаузы, содержащие более одной атомарной формулы, объединены в подсети.

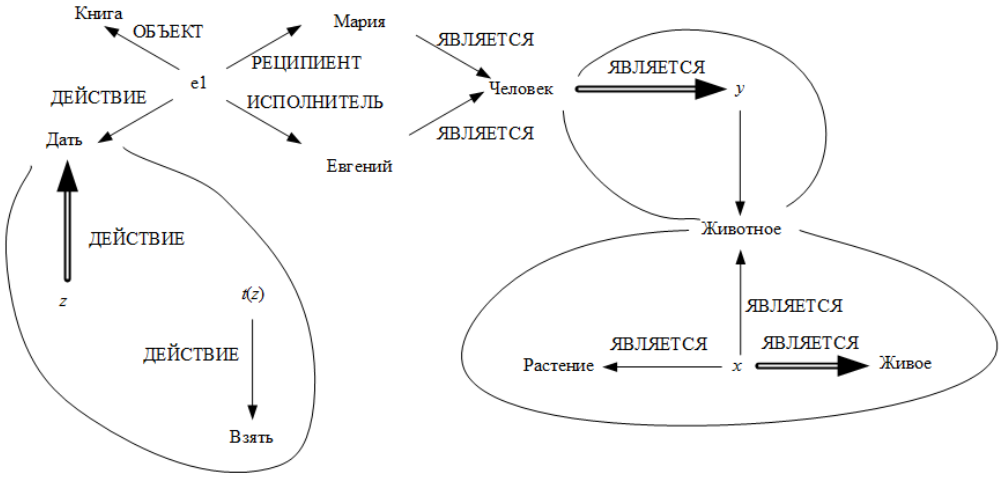


Рис. 4.7. Пример расширенной семантической сети

Помимо изобразительных возможностей, семантическая сеть обладает и другими достоинствами [6]. Основным серьезным обстоятельством, отличающим семантическую сеть, является тот факт, что информация об индивидах представляется в единственном месте – в одной вершине. Это позволяет сократить время поиска при выводе. Второй характерной положительной чертой семантической сети является то, что она одновременно представляет интерпретацию и синтаксиса, и семантики клаузы. Это обеспечивает возможность взаимодействия методов синтаксического и семантического анализа, что позволяет сделать вывод более эффективным.

Наибольшее распространение семантические сети получили при обработке естественно-языковых текстов. В качестве примера можно привести отечественную систему ПОЭТ, которая воспринимает вопросительные предложения русского языка [20] и дает ответы на них благодаря использованию семантической сети для разбора предложения.

Несмотря не достоинства семантических сетей имеет место и наличие недостатков, среди которых [5]:

– громоздкость, сложность построения и изменения;

– потребность в разнообразных процедурах обработки, связанная с разнообразием типов дуг и вершин.

4.5 Системы фреймов

Системы фреймов фактически представляют собой семантическую сеть, у которой одна часть всегда заполнена, а вторая содержит незаполненные подструктуры-слоты [10, 21]. Понятие «Фрейм» по своей сути обозначает структуру данных для отображения стереотипной (стандартной) ситуации (Рис. 4.8).

Сдача экзамена в вузе	
Сдающий	{студент, аспирант, абитуриент, группа}, ФИО
Принимающие	{лектор, ассистент, комиссия}, ФИО
Дисциплина	название дисциплины
Результат	оценка
Место/время	{расписание сессии}

Рис. 4.8. Пример фрейма представления ситуации «сдача экзамена»

Фрейм является своеобразной реализацией абстрактного мышления. Например, произнесение вслух слова «комната» порождает у слушателей какой-то образ комнаты. Отдельные части представленной комнаты у отдельного слушателя могут быть и разными, однако все представляют именно комнату. В теории фреймов именно это представление или образ комнаты называется фреймом комнаты. Фрейм комнаты содержит общие понятия (атрибуты) – названия слотов (например, «площадь», «наличие мебели», «дверь» и т.д.), и содержит значения этих атрибутов – значения слотов (например, «15 кв.м.», «диван, стул, шкаф», «зеленая дверь» и т.д.). В общем случае фреймы описываются в виде следующей конструкции:

$$F = \langle N, \{a_i, b_i, P_i\} \rangle,$$

где N – название фрейма;

a_i – название i-го слота;

b_i – значение i-го слота;

P_i – процедура i-го слота (может не присваиваться).

Другое описание фрейма может быть представлено в виде следующих элементов:

<фрейм> ::= <имя фрейма> :{<тело фрейма>};

<тело фрейма> ::= <множество слотов>;

<множество слотов> ::= <слот> | <слот>, <множество слотов>;

<слот> ::= <имя слота> : <тип слота>, <значение слота>;

<тело слота> ::= <данные>;

<тип слота> ::= <декларативный> | <наследование> | <процедурный>;

<декларативный> ::= <значение> ;

<наследование> ::= <AKO> | <ISA>;

<процедурный> ::= <IF-NEEDED> | <IF-ADDED> | <IF-REMOVED>;

и др.

Важно, что в качестве значения слота может использоваться название другого фрейма, что обеспечивает связь между ними и образование сети фреймов.

Выделяют фреймы-прототипы и фреймы-экземпляры. Фреймы-прототипы хранятся в базе знаний и определяют связи вида «*ISA*» (*is a*, является представителем) и «*AKO*» (*a kind of*, является частью или свойством)

с другими фреймами. Другими словами, фрейм-прототип является абстрактным представлением ситуаций одного класса. Фрейм из Рис. 4.*8* является фреймом-прототипом. Фрейм, на который не ссылаются другие фреймы, называется элементарным фреймом-прототипом. Фреймы-экземпляры создаются на основе подстановки значений конкретных ситуаций в слоты фрейма-прототипа.

С каждым слотом, содержащим данные (декларативный слот или слот наследования), может быть связан определенный процедурный слот-демон, который запускается при выполнении некоторого условия. Демоны имеют структуру, схожую со структурой правил продукционных моделей (условие-множество добавляемых фактов-множество удаляемых фактов). При этом типовыми условиями являются *IF-NEEDED* (если в момент обращение к слоту его свойство не было задано), *IF-ADDED* (если значение слота изменено), *IF-REMOVED* (если значение слота удалено). Кроме демонов существуют и другие процедурные слоты, которые запускаются только по запросу и являются в основном безусловными (процедурные слоты-слуги).

Важнейшим качеством, отличающим фреймы от других моделей представления знаний, является то, что фреймы позволяют моделировать ситуации при отсутствии некоторых деталей (например, при использовании значений слотов «по умолчанию»). Для большего понимания на Рис. 4.9 представлена иерархия понятия отчета.

Финансовый отчет	
ISA	ОТЧЕТ
Автор	Руководитель проекта (по умолчанию)
Тема	ТЕМА
Дата	13.10.2005 (по умолчанию)
Объем	2 (по умаолчанию)
Добавление даты	IF-NEEDED (Дата) → Дата(30.10.2005)
Удаление отчета	IF-REMOVED (this) → уведомить автора

ОТЧЕТ	
Автор	String
Тема	ТЕМА
Дата	10.03.1979 (по умолчанию)
Объем	{1, 2,…, 30}

ТЕМА	
AKO	ОТЧЕТ
Название	Новая тема (по умолчанию)
Новизна	Актуальная (по умолчанию)

Финансовый отчет №3	
ISA	Финансовый отчет
Автор	Иванов
Тема	ТЕМА
Дата	23.02.2017
Объем	3

Технический отчет	
ISA	ОТЧЕТ
Автор	Участники проекта (по умолчанию)
Тема	ТЕМА
Дата	10.03.1979 (по умолчанию)
Объем	2

Рис. 4.9. Сеть фреймов, иллюстрирующих понятие «отчет»

Следует отметить, что, несмотря на информативность фреймов, их построение является довольно трудоемким, а также сильно зависит от конкретной реализации и от языка реализации. В этой связи сети фреймов являются менее популярными, чем продукционные системы.

ГЛАВА 5. ПРИБЛИЖЕННЫЕ МНОЖЕСТВА

5.1 Основные понятия

В окружающем нас мире невозможно найти два абсолютно идентичных объекта [2]. При сравнении двух, пусть даже с виду похожих объекта, всегда можно найти тот атрибут, которым они различаются. Для этого всего лишь надо провести более детальное рассмотрение объектов. Если же описание предметной области будет менее детальным, то множество объектов, являвшихся разными, станут неразличимыми. Например, четыре человека родились в 1985, 1986, 1993, 1994 годах. При сравнении их по году рождения все они будут отличаться. Однако, если взять в качестве атрибутов десятки лет, то эти люди станут попарно неразличимы. В случае, если атрибутом является век, то все они будут неразличимы. В качестве другого примера можно взять высоту зданий. Если рассматривать трехмерное пространство, то все здания будут различны (при рассмотрении их высоты с бесконечно высокой точностью). При использовании двумерного пространства (например, снимка из космоса), высоты зданий уже будут одинаковы (а точнее равны нулю).

На базе вышесказанного можно выделить две задачи, возникающие при представлении окружающих нас сущностей. Первая задача связана с ограничением набора атрибутов снизу так, что различимость объектов будет удовлетворять условиям рассматриваемой предметной области. Вторая задача обусловлена ограничением набора атрибутов сверху так, что этот набор будет адекватным в рассматриваемой предметной области. Решением этих задач занимается теория приближенных множеств, разработанная Здзиславом Павлаком в 80-х годах XX века [22].

Приближенные множества являются своеобразной попыткой описания НЕ-фактора неоднозначности с точки рассмотрения сходных (неразличимых) объектов.

С точки зрения теории приближенных множеств информация об объектах предметной области представляется в виде информационной системы [2]:

$$SI = <U, Q, V, f>, \qquad (1)$$

где U – множество объектов,

$Q = \{q_i\}$ – множество атрибутов (свойств),

$V = \bigcup_{q \in Q} V_q$ – множество всевозможных значений этих атрибутов (V_q – множество всевозможных значений атрибута q),

f – информационная функция, которая сопоставляет каждому объекту множество значений его атрибутов (f: $U \times Q \rightarrow V$). Для получения атрибута q объекта x используют запись вида $v^x_q = f(x,q)$.

При использовании приближенных множеств в качестве инструмента принятия решений множество атрибутов разделяют на два подмножества, называемых множеством атрибутов-условий (или свойств) C и

множеством атрибутов-решений (или классов) D (т.е. $Q = C \times D$). В этом случае информационная система (1) будет называться таблицей решений и может быть представлена в виде множества решающих правил $\{П_i\}$, где $П_i$ является правилом вида

«ЕСЛИ $c_1=v_{c1}^i$ И $c_2=v_{c2}^i$ И…И $c_n=v_{cn}^i$ ТО $d_1=v_{d1}^i$ И $d_2=v_{d2}^i$ И…И $d_m=v_{dm}^i$»,

определяющим i-й объект системы.

Для примера рассмотрим магазин по торговле автомобилями, в котором в настоящий момент находится 10 машин. Пространство решений U состоит из 10 объектов, что можно записать в виде:

$$U = \{x_1, x_2, \ldots, x_{10}\}.$$

Владелец магазина отмечает четыре атрибута в своих документах, о которых чаще всего спрашивают клиенты. Это количество дверей q_1, мощность двигателя q_2, цвет q_3 и марка q_4:

$$Q = \{q_1, q_2, q_3, q_4\}.$$

Множество значений каждого атрибута представляется в виде:

$$V_{q1} = \{2, 3, 4\},$$
$$V_{q2} = \{60, 100, 200\},$$
$$V_{q3} = \{\text{Черный, Белый, Красный}\},$$
$$V_{q4} = \{\text{Лада, Киа, Тойота}\}.$$

На основании своих знаний владелец записывает значения атрибутов у себя в журнале в виде таблицы (Таблица 5.1).

Таблица 5.1

Сведения, записанные владельцем автосалона

Объект, U	Количество дверей, q_1	Мощность двигателя, q_2	Цвет, q_3	Марка, q_4
x_1	2	60	Белый	Лада
x_2	2	100	Черный	Киа
x_3	2	200	Черный	Тойота
x_4	2	200	Красный	Тойота
x_5	2	200	Красный	Лада
x_6	3	100	Красный	Лада
x_7	3	100	Красный	Лада
x_8	3	200	Черный	Тойота
x_9	4	100	Белый	Киа
x_{10}	4	100	Белый	Киа

Предположим, что на основании знаний владельца нам необходимо составить экспертную систему, которая по первым трем атрибутам позволяет определить марку автомобиля. В этом случае множество атрибутов будет разделено следующим образом:

$$C = \{q_1, q_2, q_3\}, D = \{q_4\}.$$

Можно считать, что $c_i = q_i$ ($i \in [1,3]$), а $d_1 = q_4$.

На основании определения таблицы решений содержимое табл. 5.1 представимо в форме продукционных правил:

$П_1$: ЕСЛИ $c_1 = 2$ И $c_2 = 60$ И $c_3 =$ Белый ТО $d_1 =$ Лада

…

$П_{10}$: ЕСЛИ $c_1 = 5$ И $c_2 = 100$ И $c_3 =$ Белый ТО $d_1 =$ Киа

При описании объектов, имеющих схожее описание (несколько равных атрибутов), в теории приближенных множеств используют понятие *P*-неразличимости:

Если некоторые объекты x_i, $x_j \in U$ характеризуются одинаковыми значениями свойств $q \in P$ ($P \subseteq Q$), т.е. $\forall q \in P$, $f(x_i,q) = f(x_j,q)$, то эти объекты называются *P*-неразличимыми (связаны отношением *P*-неразличимости). Отношение *P*-неразличимости определяется в пространстве $U \times U$ и записывается как $\boldsymbol{P}$. В математической форме определение отношения *P*-неразличимости записывается как:

$$x_i P x_j \Leftrightarrow \forall q \in P, \quad f(x_i, q) = f(x_j, q),$$

где $x_i, x_j \in U$; $P \subseteq Q$.

Множество объектов, связанных отношением $\boldsymbol{P}$, называется классом абстракции отношения *P*-неразличимости. Для каждого x_i существует ровно одно такое множество, обозначаемое как $[x_i]_P$, т.е.

$$[x_i]_P = \{x \in U : x_i P x\}.$$

Семейство всех классов абстракции отношения $\boldsymbol{P}$ обозначается как P^*.

Для примера с машинами классы абстракции для отношения *C*-неразличимости будут выглядеть как:

$$[x_1]_C = \{x_1\},$$
$$[x_2]_C = \{x_2\},$$
$$[x_3]_C = \{x_3\},$$
$$[x_4]_C = [x_5]_C = \{x_4, x_5\},$$
$$[x_6]_C = [x_7]_C = \{x_6, x_7\},$$
$$[x_8]_C = \{x_8\},$$
$$[x_9]_C = [x_{10}]_C = \{x_9, x_{10}\}.$$

Можно утверждать, что пары x_4, x_5, а также x_6, x_7 и x_9, x_{10} являются *C*-неразличимыми. Семейством C^* классов абстракции будет множество:

$$C^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4, x_5\},\{x_6, x_7\},\{x_8\},\{x_9, x_{10}\}\}.$$

5.2 Аппроксимация множества

В пространстве *U* могут существовать несколько подмножеств *X*, характеризующих отдельные классы объектов (в роли идентификаторов

классов могут выступать атрибуты-решения). При этом из-за ограниченности доступных атрибутов принадлежность некоторых объектов может быть неоднозначной. Иллюстративно это обстоятельство может быть представлено в виде рис. 5.10.

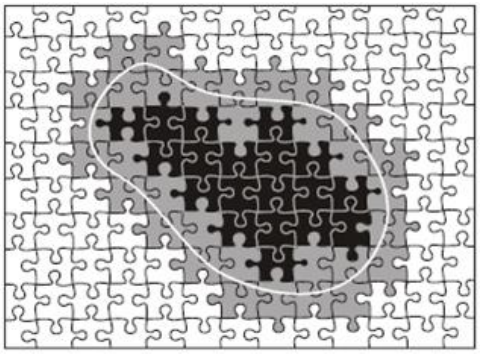


Рис. 5.10. Иллюстрация приближенного множества

Такая неоднозначность относится к одному из НЕ-факторов и формализуется путем введения понятий верхней и нижней аппроксимаций. Другими словами, *неоднозначность принадлежности объектов некоторого пространства U может быть задана в виде приближенного множества*.

Итак, P -нижней аппроксимацией множества $X \subseteq U$ называется множество $\underline{P}X$, о принадлежности элементов которого к X можно говорить с полной уверенностью (черная область на рис. 5.7). $\underline{P}X$ описывается как

$$\underline{P}X = \left\{ x \in U : [x]_P \subseteq X \right\}.$$

Нижнюю аппроксимацию также называют положительной областью множества X (иногда обозначая ее $Pos_p(X)$)

Рассмотрим определение нижней аппроксимации на примере с автомобилями. Обозначим подмножества пространства U по атрибуту-решению q_4, в результате чего получим три подмножества:

$$X_L = \{x_1, x_5, x_6, x_7\};$$

$$X_K = \{x_2, x_9, x_{10}\};$$

$$X_T = \{x_3, x_4, x_8\}.$$

C -нижними аппроксимациями этих подмножеств будут следующие множества:

$$\underline{C}X_L = \{x_1, x_6, x_7\};$$

$$\underline{C}X_K = \{x_2, x_9, x_{10}\};$$

$$\underline{C}X_T = \{x_3, x_8\}.$$

Несложно заметить, что, хотя объект x_4 и принадлежит подмножеству X_L, класс его абстракции содержит объект x_5, который не принадлежит этому подмножеству. Аналогичное утверждение можно сделать относительно x_5 и X_T.

P -верхней аппроксимацией множества $X \subseteq U$ называется множество $\overline{PX}$, о принадлежности элементов которого к X нельзя говорить с полной уверенностью (серая и черная область на рис. 5.10). $\overline{PX}$ описывается как

$$\overline{PX} = \left\{ x \in U : [x]_P \cap X \neq \varnothing \right\}.$$

Обратимся снова к примеру с автомобилями. C -верхними аппроксимациями подмножеств пространства U будут следующие множества:

$$\overline{CX}_L = \{x_1, x_4, x_5, x_6, x_7\};$$

$$\overline{CX}_K = \{x_2, x_9, x_{10}\};$$

$$\overline{CX}_T = \{x_3, x_4, x_5, x_8\}.$$

Говорят, что множество является P -точным, если его верхняя и нижняя аппроксимации совпадают и P -приближенным – в противном случае.

На основе данных о нижней и верхней аппроксимации можно вычислить граничную (серая область на рис. 5.10) и отрицательную область (белая область на Рис. 5.*10*). P -граничная область определяется как:

$$Bn_P(X) = \overline{PX} \setminus \underline{PX}.$$

P -отрицательная область является множеством тех элементов, которые однозначно не принадлежат множеству X, и определяется как:

$$Neg_P(X) = U \setminus \overline{PX}.$$

В нашем примере граничными и отрицательными областями будут являться следующие множества:

$$Bn_C(X_L) = Bn_C(X_T) = \{x_4, x_5\};$$

$$Bn_C(X_K) = \varnothing;$$

$$Neg_C(X_L) = \{x_2, x_3, x_8, x_9, x_{10}\};$$

$$Neg_C(X_K) = \{x_1, x_3, x_4, x_5, x_6, x_7, x_8\};$$

$$Neg_C(X_T) = \{x_1, x_2, x_6, x_7, x_9, x_{10}\}.$$

Для оценки «степени приближенности» рассматриваемого пространства объектов вводят 3 меры:

1 Мера точности аппроксимации (P -точность) множества:

$$\mu_P(X) = \frac{\mathrm{card}\left(\underline{PX}\right)}{\mathrm{card}\left(\overline{PX}\right)},$$

где $\mathrm{card}(\cdot)$ обозначает меру множества (в частности, его мощность).

2 Мера точности аппроксимации (P -точность) семейства множеств:

$$\beta_P(X) = \frac{\mathrm{card}\left(\cup \underline{PX}_i\right)}{\mathrm{card}\left(\cup \overline{PX}_i\right)} = \frac{\sum_i \mathrm{card}\left(\underline{PX}_i\right)}{\sum_i \mathrm{card}\left(\overline{PX}_i\right)},$$

где X_i – множество из некоторого семейства множеств X ($X=\{X_i\}$) пространства U ($X \subseteq U$).

3 Мера качества аппроксимации (P -качество) семейства множеств:

$$\gamma_P(X) = \frac{\text{card}\left(\cup \underline{P}X_i\right)}{\text{card}(U)}.$$

Вычислим меры приближенности для примера с автомобилями:

$\mu_C(X_L) = \frac{3}{5} = 0,6$,

$\mu_P(X_K) = \frac{3}{3} = 1$,

$\mu_P(X_T) = \frac{2}{4} = 0,5$,

$\beta_C(X) = \frac{3+3+2}{5+3+4} = \frac{2}{3}$,

$\gamma_C(X) = \frac{3+3+2}{10} = 0,8$.

5.2 Анализ таблиц решений

Важным качеством использования приближенных множеств для представления пространства объектов является возможность анализа взаимозависимости атрибутов информационной системы. Благодаря этому качеству можно объективно решить две задачи представления сущностей, представленные в начале настоящей главы, путем задания оптимального набора атрибутов объекта для его однозначного описания.

Степень зависимости множества атрибутов P_2 от множества атрибутов P_1 ($P_1, P_2 \subseteq Q$) можно определить через меру качества аппроксимации

$$k = \gamma_{P_1}(P_2^*).$$

Если $k = 1$, то зависимость является полной. В обратном случае ($k < 1$) зависимость считается неполной.

Если найти степень зависимости решающих атрибутов D от условных атрибутов C, можно сделать вывод об адекватности построения таблицы решений. Говорят, если $\gamma_C(D) = 1$, то таблица является хорошо определенной (а ее правила детерминированными), т.е. из равенства условных атрибутов следует равенство ее решающих атрибутов. В противном случае таблица является плохо определенной, а пара правил, для которых левые части равны, а правые различны, называются недетерминированными.

В математической форме хорошо определенная таблица решений представляется следующим образом:

$$\forall \Pi_i, \Pi_j \left(i, j \in [1, N], i \neq j\right): \forall c \in C\ f(x_i, c) = f(x_j, c) \rightarrow \forall d \in D\ f(x_i, d) = f(x_j, d),$$

где Π_i, Π_j – пара различных правил, представляющих таблицу решений,

N – количество объектов информационной системы.

Аналогично, плохо определенная таблица представляется как:

$$\exists \Pi_i, \Pi_j \left(i, j \in [1, N], i \neq j\right): \forall c \in C\ f(x_i, c) = f(x_j, c) \rightarrow \exists d \in D\ f(x_i, d) \neq f(x_j, d).$$

Плохо определенную таблицу решений можно «подкорректировать» двумя способами:

1 Исключить недетерминированные правила.

2 Расширить множество условных атрибутов.

Исследуем степень зависимости решающих атрибутов D от условных атрибутов C для примера с автомобилями.

Семейство классов абстракции отношения D-неразличимости будет содержать объединение множеств X_L, X_K, X_T (по определению этих множеств), т.е.:

$$D^* = \{X_L, X_K, X_T\}.$$

Поэтому C-нижняя аппроксимация множества D^* будет представлена в виде объединения $\underline{C}X_L$, $\underline{C}X_K$, $\underline{C}X_T$:

$$D^* = \{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\},$$

что позволяет определить степень зависимости решающих атрибутов от условных:

$$k = \gamma_C(D^*) = \frac{card(D^*)}{card(U)} = \frac{8}{10}.$$

Полученное значение свидетельствует о том, что таблица решений плохо определена, т.е. на основании множества условных атрибутов не всегда можно сделать однозначный вывод о принадлежности объектов к множествам X_L, X_K, X_T, представляющим классы абстракции отношения D-неразличимости.

При исключении объектов, представленных недетерминированными правилами (x_4 и x_5) легко показать, что таблица станет хорошо определенной:

$$U = U / \{x_4, x_5\} = \{x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}\} = D^*.$$

Однако такой способ в нашем примере не является рациональным. Ведь владелец автосалона не сможет убрать два автомобиля из продажи. Поэтому необходимо применить способ расширения множества атрибутов-условий (таб. 5.2).

Таблица 5.2

Хорошо определенная таблица сведений владельца автосалона

Объект, U	Количество дверей, c_1	Мощность двигателя, c_2	Цвет, c_3	Топливо, c_4	Обшивка, c_5	Молдинги, c_6	Марка, d_1
x_1	2	60	Белый	Бензин	Ткань	Сталь	Лада
x_2	2	100	Черный	Дизтопливо	Ткань	Сталь	Киа
x_3	2	200	Черный	Бензин	Кожа	Алюминий	Тойота

Окончание табл. 5.2

x_4	2	200	Крас-ный	Бензин	Кожа	Алю-миний	Тойота
x_5	2	200	Крас-ный	Бензин	Ткань	Сталь	Лада
x_6	3	100	Крас-ный	Диз-топ-ливо	Кожа	Сталь	Лада
x_7	3	100	Крас-ный	Газ	Ткань	Сталь	Лада
x_8	3	200	Чер-ный	Бензин	Кожа	Алю-миний	Тойота
x_9	4	100	Белый	Газ	Ткань	Сталь	Киа
x_{10}	4	100	Белый	Диз-топ-ливо	Ткань	Алю-миний	Киа

В расширенной таблице решений семейство классов абстракции отношения С-неразличимости примет вид:

$$C^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\}\},$$

Т

о есть каждый объект будет иметь индивидуальный набор значений атрибутов.

C-нижние и *C*-верхние аппроксимации семейства множеств D^* будут совпадать:

$$\underline{C}X_L = \overline{C}X_L = \{x_1, x_5, x_6, x_7\},$$

$$\underline{C}X_K = \overline{C}X_K = \{x_2, x_9, x_{10}\},$$

$$\underline{C}X_T = \overline{C}X_T = \{x_3, x_4, x_8\}.$$

Несложно заметить, что в такой ситуации C-точность и C-качество аппроксимации семейства множеств D^* будут равны единице, что свидетельствует о хорошо определенной таблице решений.

Создание хорошо определенной таблицы решений относится к решению задачи ограничения набора атрибутов снизу. Для решения второй задачи необходимо ввести понятие излишнего атрибута-условия.

Атрибут $c \in C$ является излишним, если для $C_1 = C\backslash\{c\}$ выполняется равенство $C_1^* = C^*$. В противном случае атрибут называется неустранимым.

Множество неустранимых атрибутов из *C* называется ядром *C*:

$$CORE(C) = \{c \in C : C'^* \neq C^*, C' = C \setminus \{c\}\}.$$

Вычислим неустранимые элементы и ядро из табл. 5.2. Проверим выполнимость равенства $C_1^* = C^*$ ($C_1 = C\backslash\{c\}$) для каждого условного атрибута c_i (i = [1;6]):

$$(C\setminus\{c_1\})^* = \{\{x_1\},\{x_2\},\{x_3,x_8\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_9\},\{x_{10}\}\} \neq C^*;$$
$$(C\setminus\{c_2\})^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\}\} = C^*;$$
$$(C\setminus\{c_3\})^* = \{\{x_1\},\{x_2\},\{x_3,x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\}\} \neq C^*;$$
$$(C\setminus\{c_4\})^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\}\} = C^*;$$
$$(C\setminus\{c_5\})^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\}\} = C^*;$$
$$(C\setminus\{c_6\})^* = \{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\}\} = C^*;$$

Следовательно,

$$CORE(C) = \{c_1, c_3\}.$$

Для выявления степени «неустранимости» атрибута-условия c (или множества атрибутов-условий C_1) относительно атрибутов-решений используют нормализованный коэффициент существенности:

$$\sigma_{(C,D)}(C_1) = \frac{\gamma_C(D^*) - \gamma_{C_2}(D^*)}{\gamma_C(D^*)},$$

где $C_2 = C\setminus C_1$.

Для примера с автомобилями при $C_1 = \{c_1\}$ получаем $\sigma_{(C,D)}(\{c_1\}) = 1$. Следовательно, c_1 для рассматриваемой системы является несущественным и его исключение не приведет к ухудшению качества аппроксимации.

Нормализованный коэффициент существенности используют также как меру погрешности приближения при необходимости сокращения множества условных атрибутов на множество C'. При $\sigma_{(C,D)}(C') = 0$ можно исключить C' из множества атрибутов-условий без ущерба аппроксимации семейства множеств D^*.

Подводя итог по теории приближенных множеств, стоит отметить основные преимущества ее использования:

1 Приближенные множества позволяют формализовать понятие неоднозначности, являющейся одним из НЕ-факторов;

2 Приближенные множества помогают в проектировании базы знаний экспертной системы в условиях недетерминированных пар правил;

3 Приближенные множества позволяют определить набор атрибутов, являющихся ключевыми для однозначного определения классовой принадлежности;

4 Приближенные множества позволяют оценить погрешность приближения пространства объектов при исключении одного или нескольких атрибутов из рассмотрения.

В настоящее время известно множество теоретических и практических случаев использования теории приближенных множеств в качестве инструментария искусственного интеллекта [23]. Для интересующихся читателей имеется бесплатное для некоммерческого использования ПО на базе использования теории приближенных множеств [24].

ГЛАВА 6. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

6.1 Зарождение теории искусственных нейронных сетей

Как известно, наиболее популярной архитектурой компьютера является Неймановская архитектура. Однако наряду с ней популярность набирает еще одна схема. Речь идет о нейросетевых и нейрокомпьютерных технологиях.

Искусственные нейронные сети (ИНС) – одно из направлений компьютерной индустрии, основанное на принципах функционирования искусственных устройств по образу и подобию человеческого мозга [1]. Поэтому, прежде чем рассматривать теорию ИНС необходимо привести основные сведения о принципах организации и функционирования мозга. В табл. 6.1 представлено сравнение свойств современного компьютера (машины фон Неймана) и человеческого мозга, доказывающее актуальность направления ИНС и превосходства их над традиционными вычислительными машинами.

Таблица 6.1

Сравнение традиционного компьютера и человеческого мозга

Параметр	**Машина фон Неймана**	**Человеческий мозг**
Процессор	Сложный	Простой
	Высокоскоростной	Низкоскоростной
	Один или несколько	Большое количество
Память	Отделена от процессора	Интегрирована в процессор
	Локализованная	Распределенная
	Адресация не по содержанию	Адресация по содержанию
Вычисления	Централизованные	Распределенные
	Последовательные	Параллельные
	По хранимым программам	По самообучающимся программам
Надежность	Высокая уязвимость	Живучесть
Среда функционирования	Строго определенная	Плохо определенная
	Строго ограниченная	Без ограничений

Мозг – самая сложная и взаимосвязанная система, самая крупная и функционально важная часть центральной нервной системы. В течение многих лет ученые пытаются узнать структуру и способ функционирования мозга [2]. До настоящего времени мозг остается захватывающей, но не до конца решенной загадкой. Мозг состоит из серого и белого вещества. Серое вещество представляет собой совокупность тел нейронов (или сом),

а белое – нервные волокна, которые соединяют. Помимо тела нейрон включает дендриты и аксон.

Нейрон получает информацию через свои дендриты, аккумулирует ее в соме и передает их дальше через аксон, который разветвляется на синапсы – нервные нити, которые соединяют нейроны между собой (рис. 6.1).

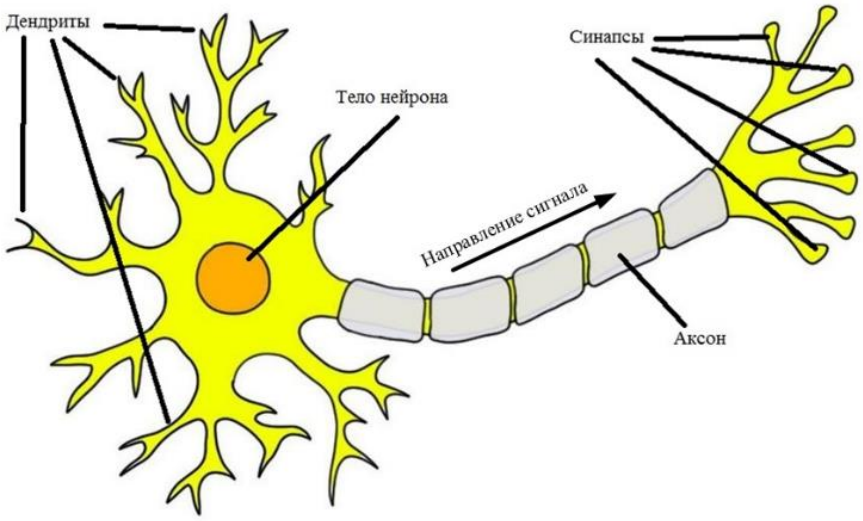


Рис. 6.1. Модель нейрона нервной системы человека

Наш мозг содержит около 10^{11} нейронов, каждый из которых связан с тысячами или десятками тысяч других нейронов. Таким образом, биологическая нейронная сеть содержит до 10^{15} взаимосвязей.

Каждый нейрон может существовать в двух состояниях – возбужденном (активном) и невозбужденном (неактивном). В возбужденное состояние нейрон переходит под действием электрических сигналов, поступающих к нему от других нейронов через синапсы, когда эти сигналы становятся достаточно большими (или больше некоторого порога). В процессе возбуждения нейрона синапсы выделяют вещество (нейромедиатор), которое способствует возбуждению или затормаживает его.

В возбужденном состоянии нейрон сам посылает электрический сигнал, который либо идет к следующему нейрону, либо ведет к активации какого-либо действия (сокращению мышцы, появлению зрительного образа, восприятию звука, и т.д.).

Известно, что количество нейронов с возрастом у человека не меняется. Примерно одинаковое количество нейронов содержат мозг ученого, политического деятеля и спортсмена. Отличие состоит в силе синаптических связей, т.е. в величине проводимостей нервных волокон. В связи с этим существует гипотеза о том, что вся информация, которая содержится в мозге, закодирована в виде сил синаптических связей. Получается, мозг человека представим в виде матрицы значений сил синаптических связей. Процесс обучения человека состоит в непрерывной корректировке содержимого этой матрицы.

6.2 Простейшие перцептроны и способы их обучения

Первой работой, явившейся теоретическим фундаментом сегодняшних интеллектуальных устройств, которые позволяют формализовать структуру и функциональность человеческого мозга, является опубликованная в 1943 году статья Уоррена Мак-Калока и Уолтера Питтса [25]. Ее авторы выдвинули гипотезу математического нейрона – устройства, которое моделирует нейрон мозга человека. Математический нейрон подобно биологическому имеет несколько входов и один выход (рис. 6.2).

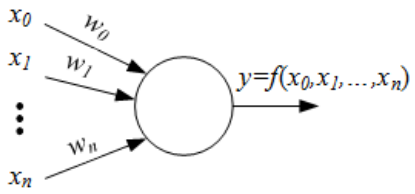


Рис. 6.2. Модель математического нейрона

На входы математического нейрона поступают входные воздействия x_i, которые суммируются, умножаясь на некоторые весовые коэффициенты w_i. Таким образом, в теле нейрона формируется взвешенная сумма:

$$u = \sum_{i=0}^{n} w_i x_i \ . \qquad (1)$$

Выходной сигнал нейрона y принимает одно из двух значений – нуль или единицу, которые формируются при вычислении пороговой функции (функции Хевисайда) [26]:

$$y = f(u) = \begin{cases} 1, \text{ если } u \geq 0 \\ 0, \text{ если } u < 0 \end{cases}. \qquad (2)$$

Таким образом, математический нейрон, как и его биологический прототип, существует в двух состояниях. Логическую функцию $f(u)$ принято называть функцией активации нейрона. Графически функция активации представлена на рис. 6.3.

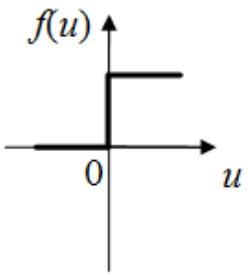


Рис. 6.3. Ступенчатая функция активации нейрона

Весовые коэффициенты w_i ($i = 1,2,..n$, n – количество входных сигналов) имитируют нейромедиаторы (силу синаптических связей), а весовой коэффициент w_0 представляет собой порог (обозначаемый также «-θ»), который указывает на «достаточность» входных сигналов x_j, о которой было

упомянуто при описании биологического нейрона. Принято считать, что в математическом нейроне всегда $x_0 = 1$. Таким образом, фактически, функция активации (2) математического нейрона представляется в виде:

$$f(S) = \begin{cases} 1, & \text{если } S \geq \theta \\ 0, & \text{если } S < \theta \end{cases}, \qquad (3)$$

где $S = \sum_{i=1}^{n} w_i x_i$ – взвешенная сумма сигналов, поступивших по синаптическим связям от других нейронов;

$\theta = -w_0$ – порог чувствительности нейрона[1].

Графически это представлено на рис. 6.4.

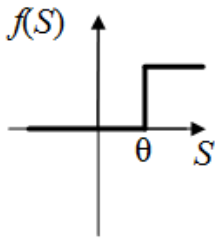


Рис. 6.4. Ступенчатая функция активации нейрона с иллюстрацией порогового значения

Таким образом, модель нейрона Мак-Калока-Питтса представляет собой линейный пороговый классификатор (рис. 6.5).

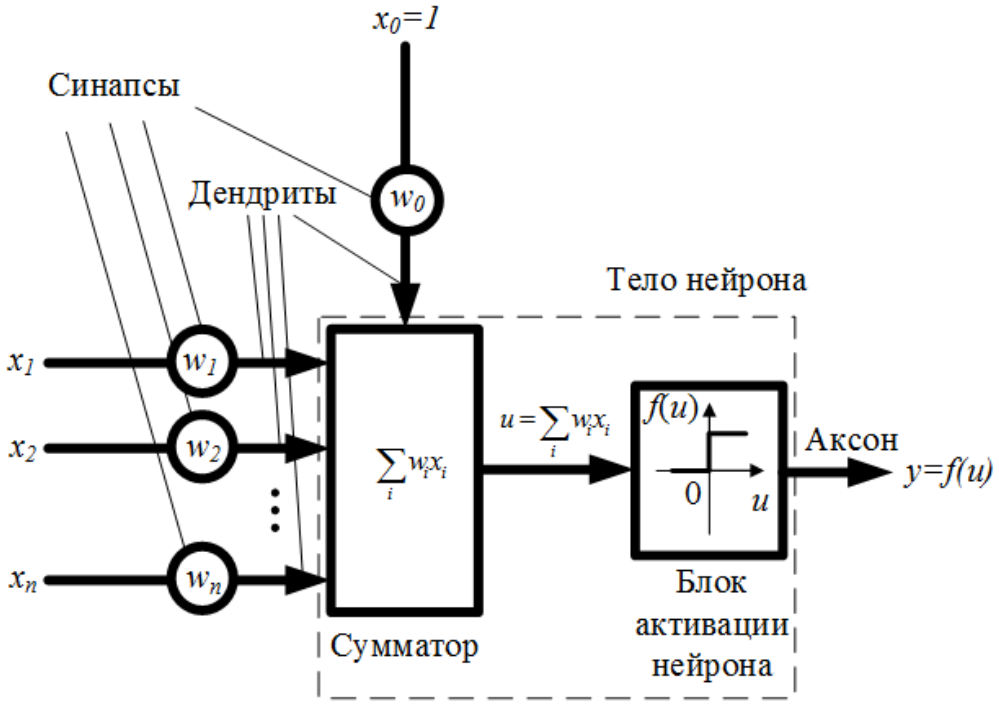
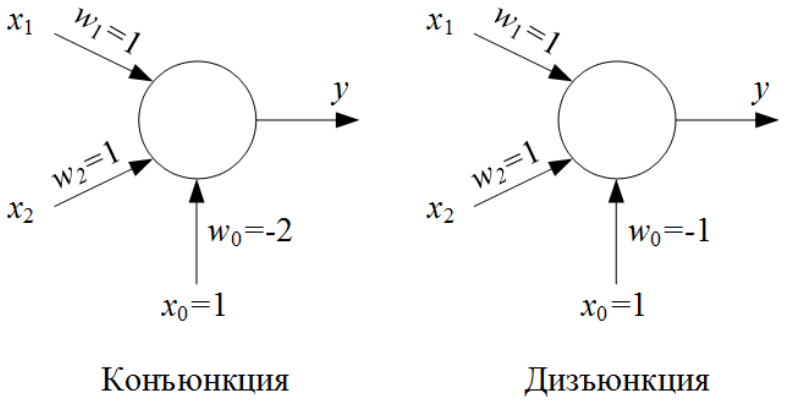


Рис. 6.5. Расширенная схема нейрона Маккалока-Питтса

[1] $u = S - \theta$ прим. автора

Одними из первых реализаций математического нейрона были логические функции. Так, математический нейрон, имеющий два входа, позволяет реализовать операцию конъюнкции или дизъюнкции (рис. 6.6).



Конъюнкция Дизъюнкция

Рис. 6.6. Пример реализации логических операций с помощью математического нейрона

Помимо описания структуры математического нейрона, а также реализаций логических функций, Мак-Калок и Питтс выдвинули гипотезу о том, что сеть из таких нейронов может обучаться, распознавать образы, обобщать информацию, т.е. он способна выполнять функции человеческого мозга. Идея Мак-Калока-Питтса была усовершенствована Дональдом Хэббом и материализована Френком Розенблаттом, в результате чего получился эвристический алгоритм обучения нейрона, основанный на принципах нейрофизиологии, согласно которым при тренировке связь между нейронами усиливается при запоминании информации и ослабевает при ошибках или редком использовании данной связки нейронов. Этот алгоритм сперва был реализован в виде программы, а затем и в виде электронного устройства, моделирующего человеческий глаз. Это устройство, названное перцептроном (или персептроном), позволило решить задачу распознавания букв латинского алфавита.

Рассмотрим эвристический алгоритм обучения на примере задачи классификации цифр на четные и нечетные. Представим себе матрицу из 12 фотоэлементов, на которую накладывается карточка с изображением цифры (рис. 6.7). Если на фотоэлемент попадает какой-либо фрагмент цифры, то данный фотоэлемент вырабатывает сигнал в виде двоичной единицы, в противном случае – нуль. Согласно модели Мак-Калока-Питтса полученные сигналы умножаются на весовые коэффициенты (которые изначально задаются генератором случайных чисел) и складываются друг с другом и со случайно заданным коэффициентом сдвига w_0. Цель обучения состоит в том, чтобы выходной сигнал был равен единице, если на карточке изображена четная цифра, и нулю, если цифра нечетная.

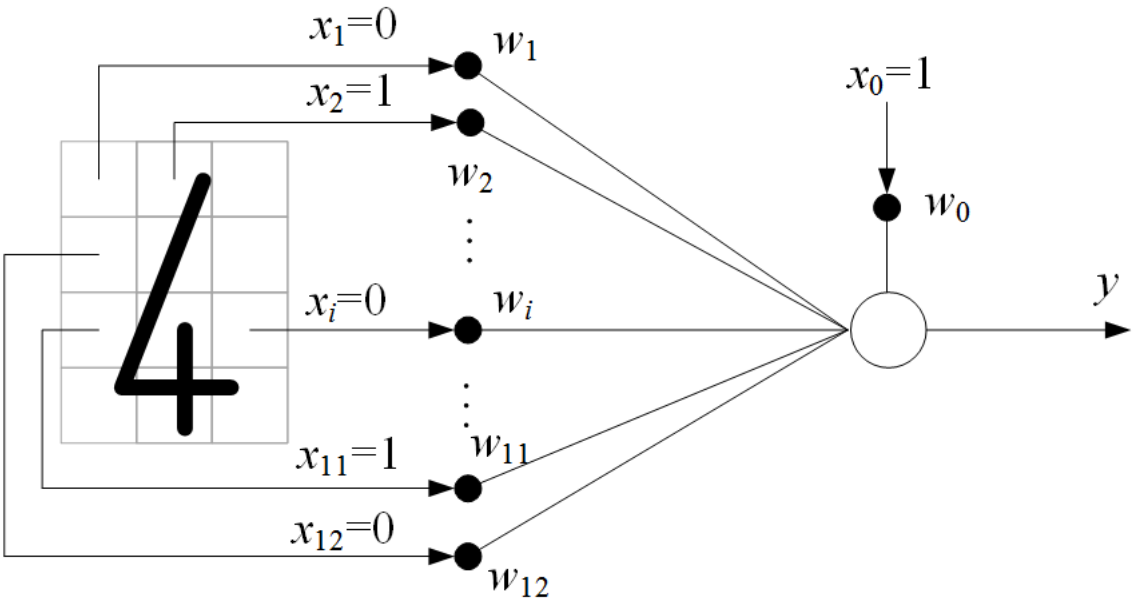


Рис. 6.7. Перцептрон, классифицирующий цифры на четные и нечетные

Эта цель достигается путем корректировки весовых коэффициентов и порогового значения, что и является обучением нейрона. Если, к примеру, на вход перцептрона предъявлена карточка с цифрой 4 и выходной сигнал, полученный после подстановки случайных коэффициентов, оказался равным единице, то корректировать вес не нужно, так как реакция нейрона верная. Однако, как и происходит чаще всего, если $y = 0$, то следует увеличить веса единичных входов. Аналогично с нечетными цифрами: если $y = 1$, то следует уменьшить веса. Последние два предложения называются первым и вторым правилом Хэбба и формулируются в общем смысле как:

1 Если сигнал нейрона неверен и равен нулю, то необходимо увеличить веса тех входов, на которые была подана единица.

2 Если сигнал нейрона неверен и равен нулю, то необходимо увеличить веса тех входов, на которые была подана единица.

Правила Хэбба напоминают естественный процесс обучения методом поощрения-наказания. Поэтому, как и в естественном смысле, так и в искусственном, процесс обучения является итерационным – за конечное число итераций (попыток, эпох) может привести к нужной цели. Итерационный алгоритм корректировки весовых коэффициентов для задачи распознавания четных цифр представляется как:

1 Подать очередной образ и вычислить выход перцептрона y.

2 Если выход правильный, то перейти на шаг 1.

3 Если выход неправильный, то применить правила Хэбба. В самом простейшем исполнении корректировка веса здесь может быть $w_i = w_i + 1$ для первого правила Хэбба и $w_i = w_i - 1$ для второго правила Хэбба.

4 Перейти на шаг 1 или завершить процесс обучения.

Правила Хэбба можно представить в более общей форме. Если через *y'* обозначить требуемое значение выходного сигнала, то на каждой итерации можно рассчитывать разницу между желаемым результатом и реальным значением *y* в виде ошибки классификации:

$$\varepsilon = y' - y . \quad (4)$$

Тогда случай ε = 0 соответствует шагу 2, случай ε > 0 – шагу 3 с выполнением условия первого правила Хэбба, а случай ε < 0 – шагу 3 с выполнением условия второго правила Хэбба.

При сохранении идеи Хэбба корректировка весов в таком случае выполняется по формуле:

$$w_i(t+1) = w_i(t) + \Delta w_i , \quad (5)$$

где $w_i(t+1)$ – новое значение весового коэффициента w_i;

$w_i(t)$ – прежнее значение весового коэффициента w_i;

Δw_i – значение корректировки для весового коэффициента w_i, вычисляемое по формуле:

$$\Delta w_i = \varepsilon x_i . \quad (6)$$

Изменение ε от w_i при определенных фиксированных значениях остальных весов w_j ($j \in [0,n]$, $j \neq i$) в общем смысле может быть наглядно представлено в виде рис. 6.8.

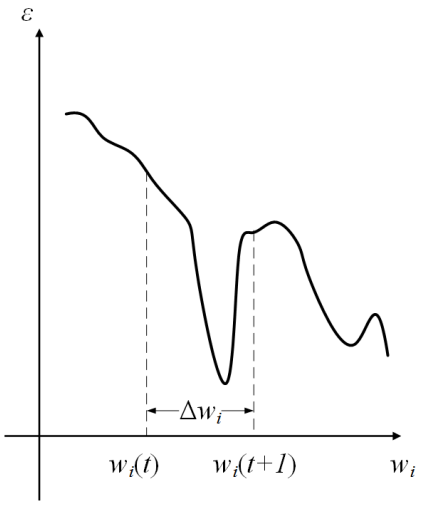


Рис. 6.8. Частная зависимость ошибки классификации от весового коэффициента

Естественным образом, достижение поставленной цели предполагает минимизацию ε. Здесь очевидно, что такое значение Δw_i слишком велико, чтобы «заскочить» в глобальный минимум. С целью устранения этого недостатка принято использовать так называемый коэффициент скорости обучения (или скорость обучения, коэффициент обучения) $\alpha \in [0,1]$, с помощью которого можно управлять величиной коррекции весов:

$$\Delta w_i = \alpha \varepsilon x_i . \quad (7)$$

Алгоритм обучения нейронов с помощью формул (4), (5), (7) известен под названием «дельта-правило». Практически все алгоритмы обучения перцептронов базируются на использовании дельта-правила.

Рис.6.9 иллюстрирует схему перцептрона, предназначенного для распознавания букв русского алфавита. В отличие от перцептрона, приведенного в предыдущем примере, такой перцептрон содержит 33 нейрона, что соответствует количеству букв в русском алфавите. Полагается, что значение y_1 на выходе первого нейрона должно быть равным единице при поступлении буквы «А» на вход и нулю для остальных букв. Значение y_2 на выходе второго нейрона должно быть равным единице при поступлении буквы «Б» на вход и нулю для остальных букв. И так далее до буквы «Я» для тридцать третьего нейрона.

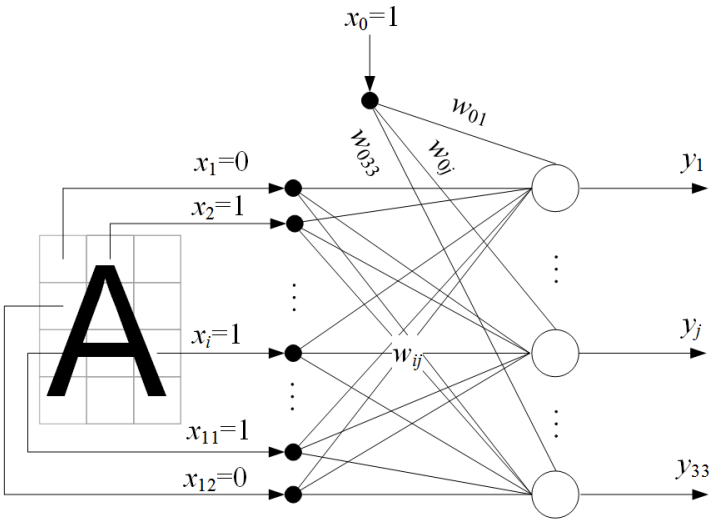


Рис. 6.9. Перцептрон, классифицирующий буквы русского алфавита

Алгоритм обучения данного перцептрона выглядит следующим образом:

1 Генератором случайных чисел всем весовым коэффициентам w_{ij} присваиваются некоторые (малые) значения.

2 На вход перцептрона предъявляется очередной образ буквы, что заставляет систему фотоэлементов выработать входной вектор $X=\{x_i\}$ (i = [1,12]).

3 Каждый нейрон выполняет взвешенное суммирование входных сигналов (1):

$$u = \sum_{i=0}^{12} w_i x_i \,,$$

и вырабатывает выходной сигнал по пороговой формуле (2).

4 Для каждого j-го нейрона вычисляется ε_j по формуле (4).

5 Для каждого весового коэффициента $w_{ij}(t+1)$ производится корректировка относительно $w_{ij}(t)$ (7):

$$\Delta w_{ij} = \alpha \varepsilon_j x_i \,.$$

6 Повторение шагов 2–5 необходимое количество раз.

6.3 Перцептроны без скрытых слоев

Вышеописанные перцептроны могут быть расширены и на больший круг задач, если вместо бинарных входных сигналов подавать аналоговые, т.е. имеющие непрерывные значения. Такое предположение выдвинули и доказали Бернард Уидроу и Тед Хофф. Они же разработали устройство на мемисторах, которое реализовывало математический нейрон, и назвали его адалайн (*ADALINE* – adaptive linear element). Нейронная сеть из адалайнов они назвали мадалайном (*Many ADALINE*).

Уидроу и Хофф предложили использовать непрерывную сигмоидальную (логистическую) функцию активации вместо ступенчатой (рис. 6.10):

$$f(u) = \frac{1}{1+e^{-u}}. \qquad (8)$$

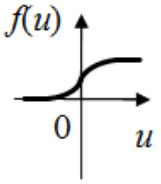


Рис. 6.10. Логистическая функция активации

Это позволило использовать значения сигналов непрерывном интервале [-∞;+∞] и выходов на непрерывном интервале [0;1]. Если существует необходимость обработки выходов в интервале [-1;1], то используется биполярная сигмоида – гиперболический тангенс:

$$f(u) = \frac{e^{2u}-1}{e^{2u}+1}. \qquad (9)$$

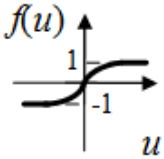


Рис. 6.11. Биполярная сигмоидальная функция активации

Помимо функции активации, они также предложили в качестве оценки разницы между желаемым и реальным значением выхода нейрона использовать суммарную среднеквадратическую ошибку по всем выходам перцептрона:

$$\varepsilon = \frac{1}{2}\sum (y' - y)^2. \qquad (10)$$

Таким образом, задача обучения нейронной сети сводится к минимизации среднеквадратической ошибки (ее также называют среднеквадратической ошибкой обучения), т.е. к решению оптимизационной задачи.

Наиболее простым, но не оптимальным методом здесь является случайный перебор весов w_{ij} с последующим вычислением y и сравнением с идеальным значением. Однако более эффективным в теории искусственных нейронных сетей принято считать метод градиентного спуска, согласно которому изменение весового коэффициента производится в сторону, противоположную градиенту поверхности, описываемой функцией $\varepsilon(\{w_{ij}\})$ [27]:

$$\Delta w_{ij} = -\alpha \frac{\partial \varepsilon}{\partial w_{ij}}, \tag{11}$$

где α – коэффициент обучения.

Среднеквадратическая ошибка является сложной функцией, зависящей, в первую очередь, от выходных сигналов перцептрона, поэтому по правилу дифференцирования сложных функций:

$$\frac{\partial \varepsilon}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial y_j} \cdot \frac{\partial y_j}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{ij}}. \tag{12}$$

При этом согласно (10):

$$\frac{\partial \varepsilon}{\partial y_j} = -(y'_j - y_j), \tag{13}$$

Согласно (8):

$$\frac{\partial y_j}{\partial u_j} = f(u)(1 - f(u)) \tag{14}$$

Согласно (1):

$$\frac{\partial u_j}{\partial w_{ij}} = x_i. \tag{15}$$

Подставив (12), (13), (14) и (15) в (11) получим:

$$\Delta w_{ij} = \alpha(y'_j - y_j) y_j (1 - y_j) x_i. \tag{16}$$

Формула (16) справедлива для логистической функции. В общем случае для обучения однослойного перцептрона используется формула:

$$\Delta w_{ij} = \alpha(y'_j - y_j) \frac{\partial f(u_j)}{\partial u_j} x_i, \tag{17}$$

или

$$\Delta w_{ij} = \alpha \delta_j x_i. \tag{18}$$

Этот алгоритм называют обобщенным дельта-правилом, преимущество которого состоит в обработке не только бинарных, но и непрерывных данных.

Для того, чтобы улучшить сходимость обучения (т.е. ускорить процесс нахождения локального минимума), используют так называемый момент обучения (или зависимость текущей поправки от предыдущей) μ:

$$\Delta w_{ij}(t+1) = -\alpha \delta_j x_i + \mu \Delta w_{ij}(t). \tag{19}$$

Наглядно преимущества использования коэффициентов обучения представлены на рис. 6.12.

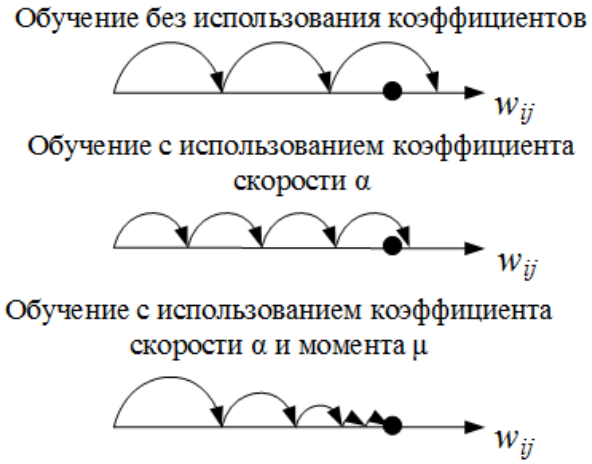


Рис.6.12. Преимущество использования коэффициентов обучения

Стоит отметить, что сигмоидальная функция активации сильно зависит от разброса значений входных сигналов. Если сигнал x_1 будет изменяться в пределах [-0.1; 0.1], а x_2 – в пределах [100; 10000], то в процессе обучения их влияние на выходной сигнал может стать неравнозначным и в результате произойдет «паралич» обучения нейронной сети. Поэтому практикой принято использовать предварительную нормализацию входных сигналов:

$$x_i = \frac{x_i - x_{i(\min)}}{x_{i(\max)} - x_{i(\min)}}, \qquad (20)$$

где $x_{i(\max)}$ – максимально возможное значение сигнала x_i,

$x_{i(\min)}$ – минимально возможное значение сигнала x_i.

До 70-х годов прошлого века появилось множество реализаций перцептрона на основе фундаментальных работ Маккалока и Питтса, Розенблатта и Хэбба. Это был колоссальный прорыв в моделировании человеческого мозга. Казалось, что ключ к сильному искусственному интеллекту был найден и его расшифровка – вопрос времени. Перцептроны использовались для решения задач диагностики, прогнозирования и анализа данных в целом. Между тем, возможно, из-за неправильной интерпретации работ Розенблатта (который уже представлял перцептрон не как один слой нейронов, а как полноценную нейронную сеть в современном понимании – с тремя слоями: входным, скрытым и выходным), либо из-за ограниченных технических возможностей того времени, оказалось, что перцептрон не мог решить некоторые задачи, причем эти задачи внешне не отличались от тех, с которыми перцептрон ранее успешно справлялся. Возникла необходимость более детального исследования перцептрона и создания более глубокой теоретической базы искусственных нейронных сетей.

6.4 Перцептроны, позволяющие решить линейно неразделимые задачи. Перцептрон Румельхарта

Успехом в этих исследованиях принято считать труд Марвина Минского [1], где математически строго доказывается неспособность использования однослойных перцептронов для решения многих простых задач, которые названы впоследствии «линейно неразделимыми». Наиболее наглядным примером, иллюстрирующим этот факт, является реализация логической операции «*XOR*» (Исключающее ИЛИ, сложение по модулю «2»), которая принимает значение «1», когда только один из входных аргументов является истинным, и «0» в остальных случаях.

Задача здесь состоит в обучении нейрона с двумя входами вычислять функцию истинности «$x_1 XOR x_2$» (табл. 6.2).

Таблица 6.2

Таблица истинности функции XOR

x_1	x_2	x_1XORx_2
0	0	0
1	0	1
0	1	1
1	1	0

При прохождении взвешенных сигналов через сумматор в теле нейрона (Рис.рис. 6.13), получаем преобразование (1):

$$u = w_1 x_1 + w_2 x_2 + w_0 . \qquad (21)$$

Если считать коэффициенты w_1, w_2, w_0 константами, то можно рассматривать уравнение (21) в виде прямой на плоскости Ox_1x_2 (рис. 6.13).

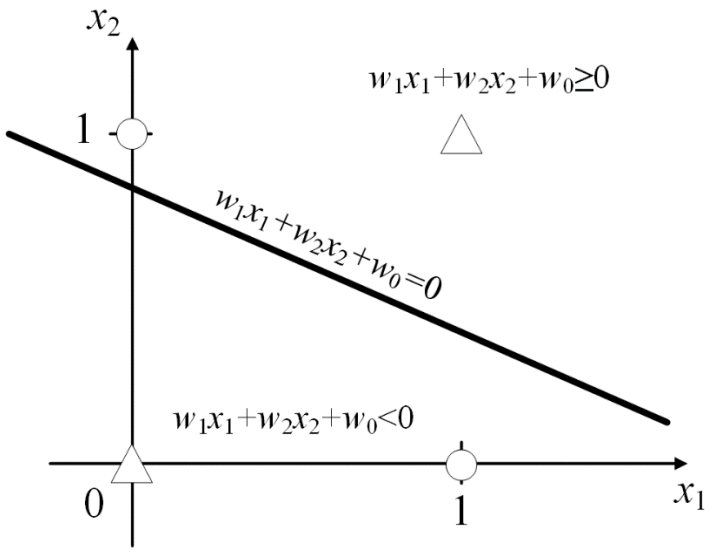


Рис. 6.13. Иллюстрация невозможности представления функции XOR однослойным перцептроном

Так как значения x_1 и x_2 являются бинарными, то целесообразно использовать ступенчатую функцию активации (2). Тогда зависимость $y(u) = y(x_1, x_2)$ примет вид поверхности, изображенной на рис. 6.14. Другими словами, выходное значение нейрона будет равно «1», если точка находится выше либо на прямой $u = 0$, и «0» – в случае попадания точки ниже прямой $u = 0$.

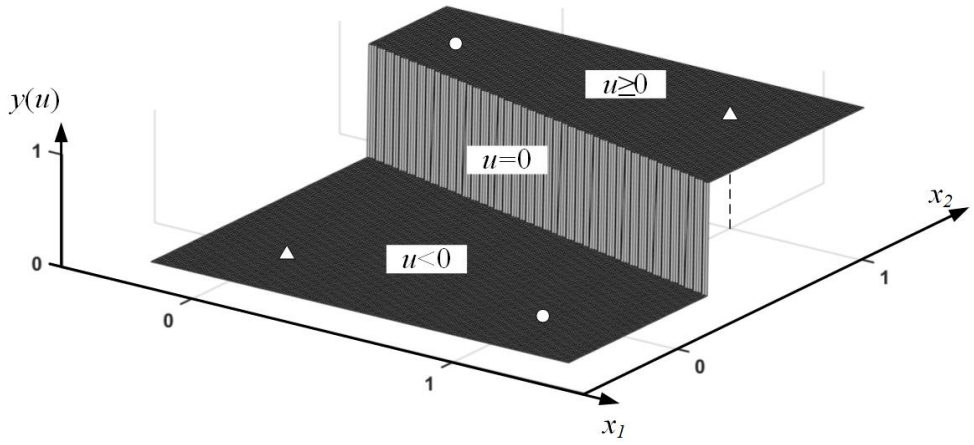


Рис. 6.14. Трехмерное представление выходного сигнала нейрона с двумя входами и ступенчатой функцией активации для реализации функции XOR

Согласно рис. 6.13 и таблице истинности функции *XOR* (табл. 6.2) для правильной реализации необходимо, чтобы точки с координатами (0;0) и (1;1) должны лежать ниже прямой $u = 0$, а точки с координатами (0;1) и (1;0) – выше, либо на ней, что, очевидно, невозможно. Это означает, что какие бы веса не подставлялись, рассмотренный перцептрон не способен воспроизвести операцию *XOR*. Для решения этой задачи необходимо задать как минимум две прямые разделения классов (в связи с чем задача и называется «линейно неразделимой»).

Таких проблем, как *XOR,* существует немало. Многие из них описаны Минским в [27]. Для их решения используют многослойный перцептрон, содержащий кроме выходного слоя нейронов так называемые скрытые слои (исследования многослойного перцептрона продолжаются до сих пор в областях *Deep Learning* и *Convolutional Neural Networks*). Приведем пример многослойного перцептрона, содержащего один скрытый слой и позволяющего реализовать операцию *XOR* (рис. 6.14, *а*). Для понимания читателя представлен также рис 6.14, *б*, который является идентичным представлением такого перцептрона, принятым при иллюстрации искусственных нейронных сетей.

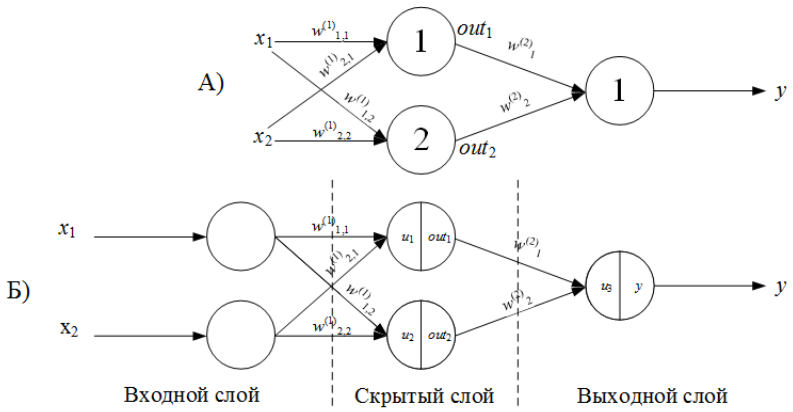


Рис. 6.15. Многослойный перцептрон, реализующий функцию *XOR*

Работа такого перцептрона строится следующим образом (представлен один из возможных вариантов набора весов):

$$u_1 = 0.5x_1 - 0.5x_2 - 0.2;$$

$$out_1 = f(u_1) = \begin{cases} 1, & \text{если } u_1 \geq 0 \\ 0, & \text{если } u_1 < 0 \end{cases};$$

$$u_2 = -0.2x_1 + 0.2x_2 - 0.05;$$

$$out_2 = f(u_2) = \begin{cases} 1, & \text{если } u_2 \geq 0 \\ 0, & \text{если } u_2 < 0 \end{cases};$$

$$u_3 = 0.3out_1 + 0.5out_2 - 0.1;$$

$$y = f(u_3) = \begin{cases} 1, & \text{если } u_3 \geq 0 \\ 0, & \text{если } u_3 < 0 \end{cases}.$$

После работ Минского пришло понимание, что добавление нового слоя существенно расширяет класс задач, решаемых с помощью перцептронов, благодаря приведению линейно неразделимых входных данных к линейно разделимым на выходе. Однако появилась новая проблема, заключающаяся в корректировке весов скрытого слоя. Классические правила Хэбба и дельта-правило годились для корректировки параметров нейронов выходного слоя, тогда как вопрос о настройке параметров внутренних нейронных слоев оставался открытым.

Эффективный алгоритм обучения многослойных перцептронов стал известен только в 1986 году благодаря работе Румельхарта, Хилтона и Вильямса [28] (в связи с чем многослойный перцептрон также называют перцептроном Румельхарата). Интересно, что представленный ими алгоритм, называемый алгоритмом обратного распространения ошибки (*back propagation*), был предложен и ранее в работах Паркера и Ле-Кана, а также изложен в диссертации Вербосом [1], однако остался незамеченным в широкой публике.

Рассмотрим алгоритм обратного распространения ошибки для случая обучения искусственной нейронной сети с одним скрытым слоем, имеющей *N* нейронов во входном слое, *L* нейронов в скрытом слое и *M* нейронов в выходном слое.

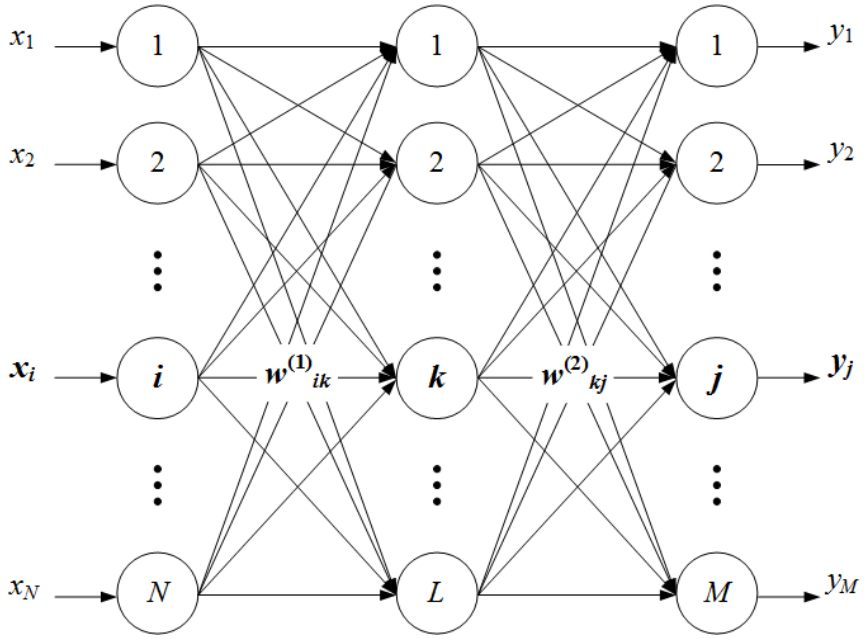


Рис. 6.16. Перцептрон Румельхарта

Алгоритм корректировки выходных весов $w^{(2)}_{kj}$ прежний:

$$w^{(2)}_{kj}(t+1) = w^{(2)}_{kj}(t) + \Delta w^{(2)}_{kj}(t+1); \quad (22)$$

$$\Delta w^{(2)}_{kj}(t+1) = \alpha \delta^{(2)}_{j} out_k + \mu \Delta w^{(2)}_{kj}(t); \quad (23)$$

$$\delta^{(2)}_{j} = (y'_j - y_j)\frac{\partial y_j}{\partial u_j}. \quad (24)$$

Напомним, что формула (24) для логистической функции принимает вид:

$$\delta^{(2)}_{j} = y_j(y'_j - y_j)(1 - y_j). \quad (25)$$

Веса скрытого слоя $w^{(1)}_{ik}$, в принципе, вычисляются также:

$$w^{(1)}_{ik}(t+1) = w^{(1)}_{ik}(t) + \Delta w^{(1)}_{ik}(t+1); \quad (26)$$

$$\Delta w^{(1)}_{ik}(t+1) = \alpha \delta^{(1)}_{k} x_i + \mu \Delta w^{(1)}_{ik}(t); \quad (27)$$

$$\delta^{(1)}_{k} = (out'_k - out_k)\frac{\partial out_k}{\partial u_k}. \quad (28)$$

Здесь все аргументы известны, неясным лишь остается вопрос о вычислении ошибки нейрона $(out'_k - out_k)$, так как желаемое значение выхода скрытых нейронов *out'*$_k$ неизвестно. Идея Румельхарта и др. состояла в том, чтобы в качестве этой ошибки использовать суммарные ошибки выходных нейронов, умноженные на веса соответствующих синапсов:

$$(out'_k - out_k) = \sum_{j=1}^{M} \delta^{(2)}_{j} w_{kj}, \quad (29)$$

то есть формула (28) примет вид:

$$\delta^{(1)}_{k} = \left(\sum_{j=1}^{M} \delta^{(2)}_{j} w_{kj} \right) \frac{\partial out_k}{\partial u_k}, \quad (30)$$

или в случае логистической функции:

$$\delta^{(1)}{}_k = \left(\sum_{j=1}^{M} \delta^{(2)}{}_j w_{kj} \right) out_k (1 - out_k) . \quad (31)$$

Таким образом, алгоритм обратного распространения ошибки для перцептрона, содержащего 1 скрытый слой, будет следующий:

1 *Инициализация данных*

Всем весовым коэффициентам $w^{(1)}{}_{ik}$ и $w^{(2)}{}_{kj}$ ($i \in [0,N]$, $k \in [0,L]$, $j \in [0,M]$) присваиваются случайные значения (обычно в интервале [-0.1;0.1]). Устанавливаются критерии остановки обучения: максимальное число итераций t_{max} и допустимая ошибка обучения $\varepsilon_{ср(доп)}$ (среднеквадратическая ошибка, усредненная по всем обучающим примерам).

2 *Прямой проход*

Подаем очередной (обычно случайно выбранный) пример $X = \{x_1, x_2, ..., x_N\}$, класс которого заранее известен и равен j (т.е. желаемые выходной сигнал $y'_j = 1$, а остальные желаемые выходные сигналы равны 0), из обучающей выборки на вход нейронной сети. Вычисляются выходные сигналы всех нейронов скрытого слоя по формуле (8) или по формуле (9):

$$out_k = f\left(\sum_{i=0}^{N} w_{ik} x_i \right), \quad (32)$$

И выходные сигналы всех нейронов выходного слоя по формуле (8) или по формуле (9):

$$y_j = f\left(\sum_{k=0}^{L} w_{kj} out_k \right). \quad (33)$$

3 *Обратный проход*

Вычисляем веса относительно полученной разницы между желаемыми выходными сигналами y'_j и полученными реальными значениями y_j по формулам (22) и (26).

4 *Проверка критериев останова*

Вычисляем среднеквадратическую ошибку обучения:

$$\varepsilon_{ср} = \frac{1}{Q} \sum_{q=1}^{Q} \sum_{j=1}^{M} \left(y'_j(q) - y_j(q) \right)^2 , \quad (34)$$

где Q – количество примеров в обучающей выборке;

$y'_j(q)$ и $y_j(q)$ – эталонное значение и реальное значение j-го выхода сети для примера q, соответственно.

Если выполняется один из критериев: $\varepsilon_{ср} < \varepsilon_{ср(доп)}$ или $t > t_{max}$, то обучение останавливается. В противном случае переходим к п. 2.

Данный алгоритм применим и к нейронной сети с бóльшим количеством скрытых слоев. Тогда формула (29) справедлива с учетом нейрона последующего слоя. Однако большинство решаемых в настоящее время задач ограничиваются использованием искусственных нейронных сетей с одним скрытым слоем.

Теперь представим себе, что нейронную сеть с одним выходом ($y'_{А}=1, y'_{не\text{-}А}=0$) обучили распознавать букву «А» (рис. 6.17). После обучения нам необходимо классифицировать по принципу «А»-«не-А» изображение буквы немного видоизмененного шрифта (рис. 6.18). Если шрифт не слишком отличается от того, что использовался при обучении, наш перцептрон выдаст правильное заключение при правильной настройке.

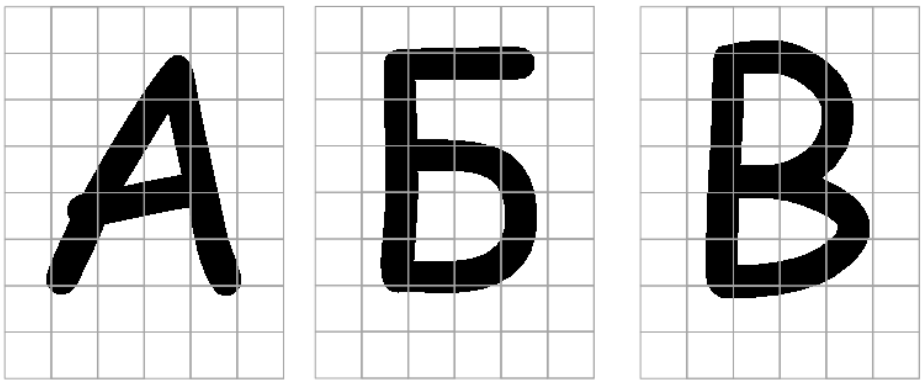


Рис. 6.17. Пример изображений из обучающей выборки

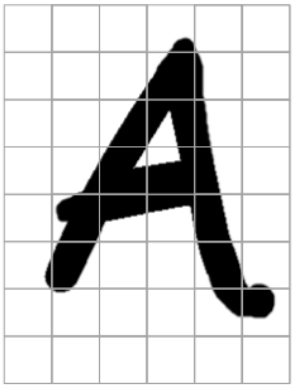


Рис. 6.18. Пример изображения «А» из тестовой выборки

Свойство перцептронов правильно реагировать на входные образы, которых не было в обучающей выборке, называется свойством обобщения. При этом важную роль играет количество нейронов скрытого слоя, благодаря которому задается значения параметра «не слишком отличается». Если количество нейронов слишком большое, то нейронная сеть может оказаться «переобученной» (или «зубрилой»), в следствие чего будет угадывать только примеры, которые присутствовали при обучении. В противном случае, если количество нейронов скрытого слоя слишком мало, то мы рискуем «недообучить» нейронную сеть, которая будет не в силах определить различия между разноклассовыми примерами. Нахождение компромиссного значения числа скрытых нейронов – задача скорее интуитивная. Решение ее – выбор числа нейронов в интервале между количество входных и выходных нейронов.

Однако на практике иногда используется значение «золотой середины»:

$$L = \frac{Q}{k\left(N+M\right)}, \tag{35}$$

где k – случайно выбранное значение (в интервале [2,10]).

Таким образом, искусственная нейронная сеть, описанная Румельхартом, является мощнейшим инструментом в определении нелинейной зависимости между структурой данных и их классами, что привлекает все новый интерес специалистов в области ИИ, в частности, при решении задачи распознавания образов. Представленные схемы нейронных сетей являются далеко не единственными в своем роде. Все описанные выше перцептроны относятся к полносвязным нейронным сетям прямого распространения, обучающихся с учителем. Заинтересованный читатель может также обратить внимание на изучение других популярных типов нейронных сетей: радиально-базисных, рекуррентных и самообучающихся, примеры которых представлены в [1].

ГЛАВА 7. ОСНОВЫ ТЕОРИИ НЕЧЕТКИХ МНОЖЕСТВ

7.1 Мера возможности и мера вероятности

Развитие ЭВМ порождает стремление решать все более сложные практические задачи, что ведет к потребности формализации противоречивой и неточной информации, недоступной в количественной форме [29]. Недоступность может быть по различным причинам – из-за несовершенства измерительных устройств или вследствие того, что единственным источником сведений является человек.

Зачастую при анализе сложной и многомерной системы ее упрощенная модель обеспечивает более понятную информацию, чем детальная и более точная модель. Таким образом, по мере возрастания сложности системы способность формализации точных сведений о ее поведении падает до некоторого порога, за которым точность и смысл становятся взаимоисключающими. «Хорошая модель» системы должна реализовывать компромисс между избытком точности и избытком неточности. В результате первого мы получим противоречивость и двусмысленность фактов, во втором случае мы будем иметь малую информативность.

Для количественного описания неполноты окружающей нас информации разработана целая теория, называемая теорией нечетких множеств. Понятие «нечеткое множество» предложено в 1965 году Лотфи Заде как видоизменение традиционного множества, имеющее градации в отношении принадлежности. Количественное описание этих градаций представляют меру возможности (или нечеткости) появления того или иного события. Долгое время возникали споры по вопросу: «Является ли теория нечетких множеств лженаукой, а мера возможности всего лишь разновидностью меры вероятности?». Ответ на вопрос можно получить, если рассмотреть нечеткость и вероятность подробнее.

Мера вероятности *P* имеет две интерпретации. В первой интерпретации вероятность рассматривается как отношение числа благоприятных исходов к общему числу исходов. Вторая интерпретация, субъективистская, представляет вероятность как число, пропорциональное сумме, которую субъект должен заплатить, если высказывание, являющееся по его утверждению истинным, окажется ложным. Другими словами, мера вероятности события *A* характеризует степень того, что обратное событие «не-А» не произойдет, т.е. с точки зрения теории вероятности:

$$A \cap \overline{A} = \varnothing .$$

Вероятность появления двух независимых событий обладает свойством аддитивности:

$$P(A \cup B) = P(A) + P(B) .$$

Мера возможности *П* оценивает степень уверенности субъекта о возможности появления события, не исключая того факта, что противоположное событие также возможно, т.е. с точки зрения теории возможностей:

$$A \cap \overline{A} \neq \varnothing .$$

Возможность появления двух независимых событий не является аддитивной мерой и определяется наилучшим событием:

$$\Pi(A \cup B) = \max\left(\Pi(A), \Pi(B)\right).$$

Другим фактом, указывающим на различие нечеткости и вероятности, является различие НЕ-факторов, изучаемых в рамках двух теорий.

Теория вероятности имеет дело со стохастической неопределенностью, имеющей место в ситуациях, когда некоторое хорошо описанное событие может произойти, а может и не произойти [30]. При этом степень неопределенности с течением времени меняется.

Рассмотрим высказывание: «Вероятность того, что при бросании монеты выпадет орел, равна 0,5».

После бросания монеты неопределенность исчезнет, так как монета окажется в одном из двух состояний. Таким образом, рассматриваемое высказывание имеет смысл только по отношению к событию в будущем.

Рассмотрим другое высказывание: «Вероятность того, что завтра пойдет дождь, равна 80 %».

В этом высказывании предполагается определенность в описании «пойдет дождь». Однако при подробном рассмотрении станет очевидно, что это событие плохо определено: не ясно, будет ли идти дождь целый день или 80 % продолжительности суток? Будет ли мелкий дождь или ждать ливня? Таким образом, здесь мы сталкиваемся с другим типом неопределенности, который содержательно отличается от стохастического. Эта неопределенность связана с лингвистическим описанием ситуации, она относится скорее к качественному описанию точности события, а не к количественной оценке того, произойдет ли это событие в будущем или нет.

Сложность качественной оценки проявляется в форме лингвистической неопределенности описания картины мира.

Этот тип неопределенности связан с неточностью и неоднозначностью языкового описания, с которым мы сталкиваемся в повседневной жизни. Достаточно рассмотреть фразы «высокий человек», «горячие пирожки», «устойчивая валюта» и т.п., чтобы понять, что дать им точные количественные определения вряд ли удастся.

Действительно, люди различного роста будут иметь различное мнение, является ли человека высоким или нет. Более того, если считать человека выше 180 см высоким, то будет ли человека с ростом 179,999 см низким или нет?

Рассмотрим высказывание: «Вероятно, мы будем иметь успешный финансовый год».

Это высказывание имеет существенные отличия от ранее рассмотренных. Во-первых, само событие «успешный финансовый год» не определено. Для некоторых компаний это означает, что им удастся избежать кризиса, для других – превышение прибыли за предшествующий год. Даже

для отдельно взятой компании сложно предположить количественное значение прибыли, определяющее год как успешный. Подобные понятия принято называть «субъективными категориями».

Другая особенность высказывания состоит в определении вероятности. Оно задается качественно (в отличие от предыдущих высказываний). Следовательно, выражение вероятности также относится к субъективной категории.

Несмотря на субъективность, качественные высказывания играют немалую роль в процессе повседневного принятия решений. Высказывания, не имеющие количественного содержания, успешно используются для комплексных оценок. В некоторых случаях лингвистическая неопределенность позволяет придать дополнительную гибкость высказыванию.

Чтобы адекватно использовать полезную информацию субъективных высказываний в решении технических проблем, необходимо было разработать математическую модель. Именно с этой целью и была основана теория нечетких множеств.

Объектом исследований теории нечетких множеств является проблема формализации субъективного рассуждения людей в искусственных системах. Люди умеют рассуждать приблизительно, современные компьютеры этой возможностью не обладают. При общении людей не возникает концептуальных проблем при интерпретации фраз типа «высокий человек» и «высокооплачиваемая работа», поскольку эти фразы передают семантически понятную информацию.

В то же время компьютеры не в силах понять эти фразы в исходном виде, для их интерпретации необходимо конкретное значение высоты или заработной платы, которое сравнивается с заданным пороговым значением для вынесения вердикта относительно «высоты». Основным достоинством теории нечетких множеств является использование лингвистических переменных вместо количественных, а нечеткую логику вместо бинарной для формализации субъективных категорий. Теория нечетких множеств осуществляет попытку включения опыта и интуиции отдельного человека при рассмотрении сложных технических систем.

Рассмотренные выше примеры иллюстрируют тот факт, что стохастическая и лингвистическая неопределенность имеют различный характер. Стохастическая неопределенность имеет дело с неопределенностью того, произойдет ли событие или нет, а теория вероятностей позволяет дать этому количественную оценку. Лингвистическая неопределенность связана с неточностью описания самой ситуации независимо от времени ее рассмотрения. Теория вероятностей здесь непригодна, так как представления о субъективных категориях не согласуются с ее аксиомами.

Приведем еще один наглядный пример. Представим себе ситуацию, когда путник после длительного путешествия, испытывая чувство жажды, находит две бутылки с неизвестной жидкостью внутри.

Предположим, что степень принадлежности первой бутылки (бутылки А) к жидкостям пригодным для питья, равна 0,9. Известно также, что вероятность того, что во второй бутылке (бутылке Б) вероятность пригодности для питья содержимого равна 0,9. Какую в этом случае путнику выбрать бутылку?

Анализируя информацию о содержимом бутылки А, можно предположить, что в ней находится не совсем пригодная жидкость для питья, к примеру, болотная вода. В то же время в ней не может находиться жидкость, не пригодная для питья, так как в этом случае степень принадлежности была бы 0.

Анализируя информацию о содержимом бутылки Б, естественно предположить о частотной интерпретации содержимого. В этом случае резонно предположить, что в 9 случаях из 10 в бутылке Б может быть вода и путник утолит жажду. Однако эта информация не позволяет ответить на вопрос, что произойдет с путником в 10 случае. Значит, в одном случае из десяти в этой бутылке может находиться нечто, что совсем непригодно для питья, например, соляная кислота.

Вывод напрашивается очевидный – необходимо выбрать бутылку А.

При изменении информативности ситуации, например, в случае, если путнику станет известно, что в бутылке А содержится джин-тоник, а в бутылке Б – уксус, степень принадлежности бутылки А останется неизменной, в то время как вероятность пригодности для бутылки Б станет равной нулю.

7.2 Основные определения теории нечетких множеств

Нечеткое множество представляет собой совокупность элементов произвольной природы, относительно которых нельзя с полной уверенностью утверждать – принадлежит ли тот или иной элемент рассматриваемой совокупности данному множеству или нет.

Формально нечеткое множество A определяется как множество упорядоченных пар вида:

$$< x, \mu_A(x) >, \quad (1)$$

где x – элемент некоторого универсального множества X;

$\mu_A(x)$ – функция принадлежности, которая ставит в соответствие каждому из элементов $x \in X$ некоторое действительное число из интервала [0, 1], т.е. определяется в форме отображения:

$$\mu_A(x) : X \to [0,1]. \quad (2)$$

При этом $\mu_A(x) = 1$ означает, что элемент x определенно принадлежит нечеткому множеству A, а $\mu_A(x) = 0$ – x определенно не принадлежит нечеткому множеству A.

Нечеткое множество записывается в виде[2]:

$$A = \{\langle x_1, \mu_A(x_1)\rangle, \langle x_2, \mu_A(x_2)\rangle, ..., \langle x_n, \mu_A(x_n)\rangle\}. \quad (3)$$

При описании нечетких множеств применяют понятия из классической теории множеств. Например, используются такие понятия как «пустое нечеткое множество» и «универсум».

Пустое нечеткое множество ∅ определяется как множество, функция принадлежности которого тождественно равна нулю для всех без исключения элементов: $\mu_{\varnothing} = 0$. Что касается универсума X, то функция его принадлежности тождественно равна единице для всех без исключения элементов: $\mu_X = 1$.

Одним из основных понятий, используемых при описании нечетких множеств, является понятие носителя нечеткого множества.

Носителем нечеткого множества A является обычное множество A, которое содержит только те элементы универсума, для которых значения функции принадлежности соответствующего нечеткого множества отличны от нуля:

$$A = \{x \in X \mid \mu_A(x) > 0\} \qquad \forall x \in X .$$

Очевидно, пустое нечеткое множество имеет пустой носитель, а носитель универсума совпадает с универсумом.

В зависимости от количества элементов в нечетком множестве по аналогии с обычными множествами можно определить конечные и бесконечные нечеткие множества.

Нечеткое множество называется конечным, если носитель является конечным множеством. В этом случае принято говорить, что мощность нечеткого множества конечна и численно равна количеству элементов его носителя, а само множество записывается в форме (3).

Бесконечное нечеткое множество имеет носитель, не являющийся конечным множеством. В этом случае удобнее формально представлять нечеткое множество в виде аналитической записи:

$$A = \{\langle x, \mu_A(x)\rangle\},$$

или

$$A = \{x, \mu_A(x)\}, \quad (4)$$

где $\mu_A(x)$ – некоторая функция, заданная аналитически в форме математического выражения $f(x)$ или графически в форме некоторой кривой.

Обычно субъективные высказывания задаются в виде нормального нечеткого множества, т.е.:

---

[2] В литературе встречаются и такие обозначения как $A = \{\langle \mu_A(x_1), x_1\rangle, \langle \mu_A(x_2), x_2\rangle, ..., \langle \mu_A(x_n), x_n\rangle\}$ и $A = \{x_1/\mu_A(x_1) + x_2/\mu_A(x_2) + ... + x_n/\mu_A(x_n)\}$ («+» обозначает не арифметическую сумму, а теоретико-множественное объединение). Часто для наглядности нечеткие множество обозначаются знаком «~», т.е. как $\tilde{A}$

$$\exists x \in X : \mu_A(x) = 1. \quad (5)$$

Для наглядной иллюстрации вышеописанных понятий рассмотрим пример нечеткого множества.

Предположим, необходимо построить некоторое нечеткое множество, содержательно описывающее выходные дни семидневной недели (также говорят «терм «выходные дни недели»»). В терминологии классических множеств ситуация тривиальная, а именно, дни недели с понедельника по пятницу являются рабочими, а суббота и воскресенье – выходными. Таким образом, классическое множество выходных дней A состоит из двух элементов: $A=\{$*суббота, воскресенье*$\}$.

При определении соответствующего нечеткого множества A можно оценить субъективность отношения к различным дням недели, рассматривая их с точки зрения психологии возможного отдыха. Тут, что касается дней с понедельника по четверг, то отношение к ним вряд ли изменится. А вот пятница, особенно вечер, для многих ассоциируется с полноценным отдыхом и высокой степенью положительных эмоций. Суббота является, безусловно выходным днем, а воскресенье является пограничным временем. К вечеру настроение падает в связи с наступающей рабочей неделей. Таким образом, нечеткое множество A описывается как:

$$A = \begin{Bmatrix} \langle \textit{понедельник}, 0 \rangle, \langle \textit{вторник}, 0 \rangle, \langle \textit{среда}, 0 \rangle, \langle \textit{четверг}, 0 \rangle, \\ \langle \textit{пятница}, 0.5 \rangle, \langle \textit{суббота}, 1 \rangle, \langle \textit{воскресенье}, 0.8 \rangle \end{Bmatrix}.$$

Представим графически нечеткое множество A в виде координатной плоскости, где абсциссой является значение элемента универсума, а ординатой – соответствующее ей значение функции принадлежности (рис. 7.1).

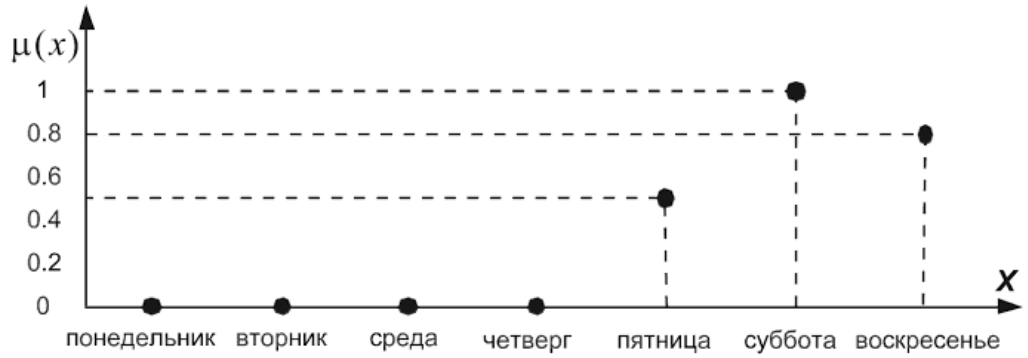


Рис. 7.1. Графическое представление конечного нечеткого множества, описывающего выходные дни недели

Рассмотрим этот пример на случай представления множества как бесконечного. Это можно сделать благодаря тому факту, что наше ощущение «выходного дня» меняется с течением суток. Тогда функция принадлежности может быть задана аналитически (рис. 7.2).

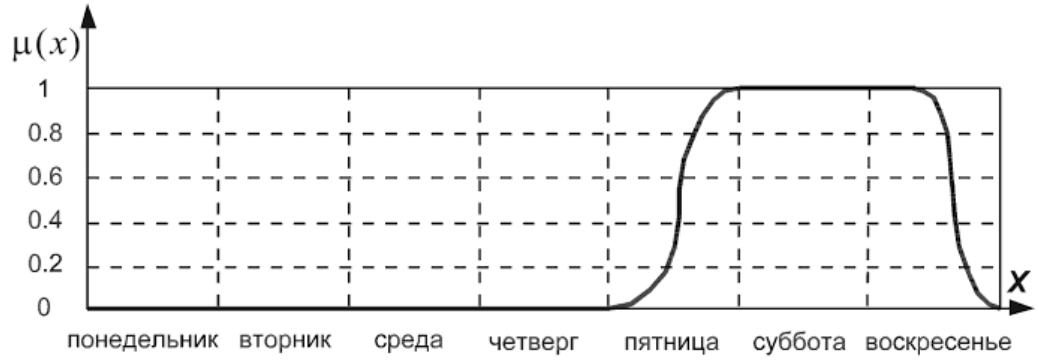


Рис. 7.2. Графическое представление бесконечного нечеткого множества, описывающего выходные дни недели

Для сравнения представим обычное множество в аналитической форме (рис. 7.3).

Рис. 7.3. Графическое представление обычного множества, описывающего выходные дни недели

Помимо носителя нечеткого множества вводятся такие базовые понятия как «высота нечеткого множества», «ядро нечеткого множества» и «границы нечеткого множества».

Величина $h_A = sup\{\mu_A(x)\}$ называется высотой нечеткого множества. При этом условие (5) может и не выполняться. В таком случае множество будет являться субнормальным.

Примером субнормального множества может служить множество *B*, представляющее терм «Большое действительное число, с функцией принадлежности, заданной математическим выражением (рис. 7.4):

$$\mu_B(x) = \begin{cases} 0 & \text{если } x \in [0;1) \\ \dfrac{x-1}{x} & \text{если } x \in R_+ \setminus [0;1) \end{cases}$$

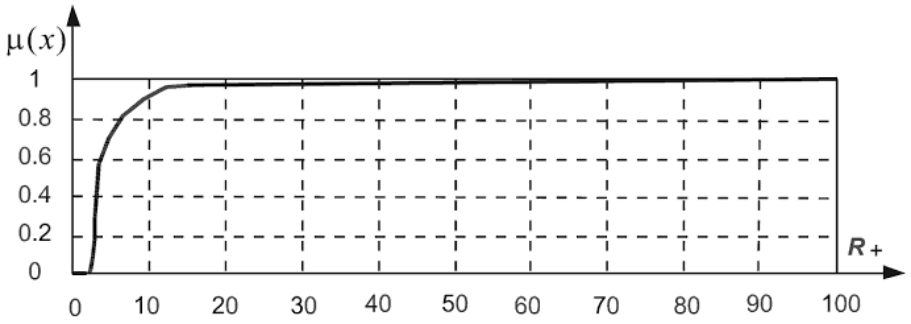


Рис. 7.4. График функции принадлежности бесконечного нечеткого множества «большое действительное число»

Высота этого множества равна 1, хотя в нем отсутствуют элементы, степень принадлежности которых равна 1.

Ядром нечеткого множества $\tilde{A}$ является обычное множество A, элементы которого удовлетворяют условию:

$$A = \{x \in X \mid \mu_A(x) = 1\}. \quad (6)$$

Границами нечеткого множества называются такие элементы универсума, для которых значения функции принадлежности отличны от 0 и 1:

$$A = \{x \in X \mid \mu_A(x) \in (0,1)\}. \quad (7)$$

В общем случае понятия нечетких множеств можно проиллюстрировать графически следующим образом (рис. 7.5).

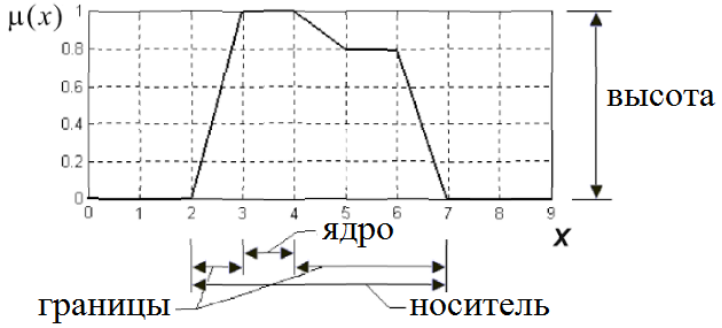


Рис. 7.5. Иллюстрация базовых понятий теории нечетких множеств

7.3 Способы задания нечетких множеств

Формальное определение нечеткого множества не накладывает никаких ограничений на выбор конкретной функции принадлежности для его представления. Однако на практике сформировался устойчивый набор допустимых функций принадлежности, которые допускают наиболее удобное представление. Рассмотрим наиболее популярные из них.

Кусочно-линейные функции принадлежности. Такие функции принадлежности состоят из отрезков прямых линий, образуя непрерывную

или кусочно-линейную форму. Наиболее популярными являются треугольная и трапециевидная функции принадлежности (рис. 7.6).

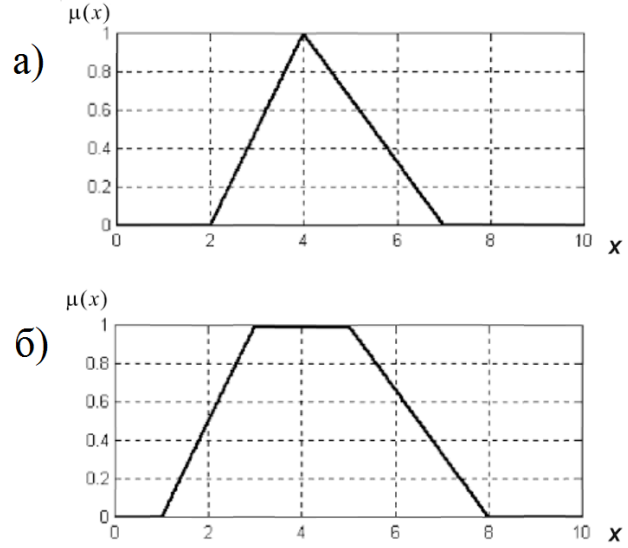


Рис. 7.6. Треугольная (а) и трапециевидная (б) функции принадлежности

Первая из этих функций принадлежности может быть задана аналитически следующим образом:

$$\mu_{\triangle}(x)=\begin{cases}0, & \text{если } x \le a \\ \dfrac{x-a}{b-a}, & \text{если } a < x \le b \\ \dfrac{c-x}{c-b}, & \text{если } b < x \le c \\ 0, & \text{если } x > c\end{cases}, \qquad (8)$$

где $a, b, c \in \mathbb{R}$ – параметры функции, связанные отношением $a \le b \le c$.

Для функции принадлежности, изображенной на Рис. 6, а), значения этих параметров равны соответственно, 2, 4 и 7. Несложно заметить, что параметры a и c характеризуют основание треугольника, а b – его вершину.

Трапециевидная функция принадлежности может быть задана аналитически следующим образом:

$$\mu_{\mathrm{T}}(x)=\begin{cases}0, & \text{если } x \le a \\ \dfrac{x-a}{b-a}, & \text{если } a < x < b \\ 1, & \text{если } b \le x \le c \\ \dfrac{d-x}{d-c}, & \text{если } c < x < d \\ 0, & \text{если } x \ge d\end{cases}, \qquad (9)$$

где $a, b, c, d \in \mathbb{R}$ – параметры функции, связанные отношением $a \le b \le c \le d$.

Для функции принадлежности, изображенной на рис 7.6,*б*), значения этих параметров равны соответственно, 1, 3, 5 и 8. Параметры *a* и *d* характеризуют нижнее основание трапеции, а *c* и *d* – верхнее основание.

Сигмоидальная функция принадлежности. Сигмоидальная функция принадлежности является наиболее популярной функцией семейства сплайн-функций и задается следующим выражением:

$$\mu_S(x) = \frac{1}{1+e^{-a(x-b)}}, \tag{10}$$

где $a, b \in \mathbb{R}$ – параметры функции.

При этом в случае $a > 0$ функция является *S*-образной сплайн-функцией, а в случае $a < 0$ – *Z*-образной (рис. 7.7).

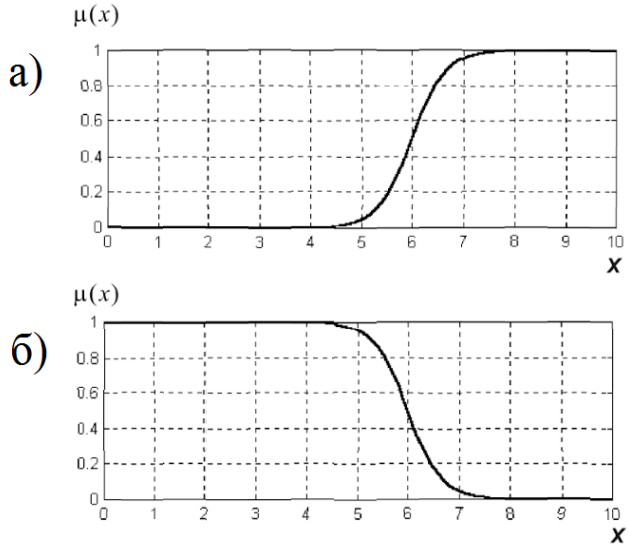


Рис. 7.7. *S*-образная (а) и *Z*-образная (б) сигмоидальные функции принадлежности

Для *S*-образной функции на рис. 7.7, *а*) значения параметров равны 3 и 6, соответственно, а для *Z*-образной функции на рис. 7.7, *б*) – равны -3 и 6.

П-образная функция принадлежности. Еще одной популярной функцией принадлежности является *П*-образная функция принадлежности (рис. 7.8, *а*):

$$\mu_{П}(x) = e^{\frac{-(x-a)^2}{2b^2}}, \tag{11}$$

где $a, b \in \mathbb{R}$ – параметры функции.

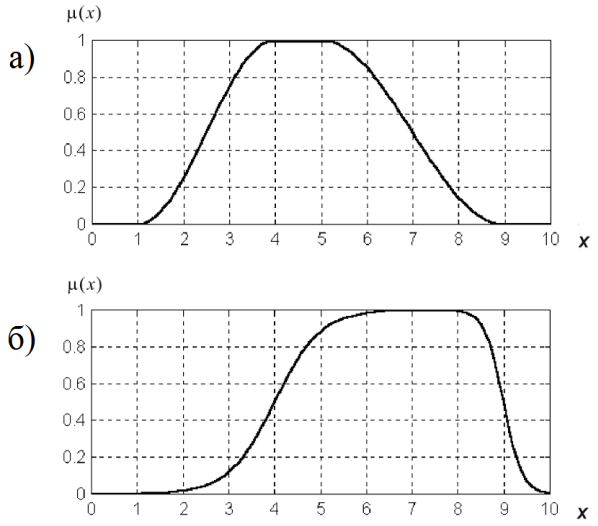


Рис. 7.8. *П*-образные функции принадлежности

На рис. 7.8, *а* функция принадлежности принимает значения параметров 2 и 4, соответственно.

Также П-образная функция может быть задана в виде арифметического произведения *S*-образной и *Z*-образной функции (рис. 7.8, *б*):

$$\mu_{П}(x) = f_S(x,a,b) \cdot f_Z(x,c,d), \qquad (12)$$

где $f_S(\cdot)$ и $f_Z(\cdot)$ – *S*-образная и *Z*-образная функции, соответственно, заданные формулой (10).

На рис. 7.8, *б* функция принадлежности принимает значения параметров 1, 4, 5 и 9, соответственно.

7.4 Операции над нечеткими множествами

Пересечением двух нечетких множеств *A* и *B* является третье нечеткое множество $C=A \cap B$, функция принадлежности которого определяется как:

$$\forall x \in X : \mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}. \qquad (13)$$

Другими словами, *C* – наибольшее нечеткое множество, которое содержится одновременно во множестве *A* и *B* (рис. 7.9).

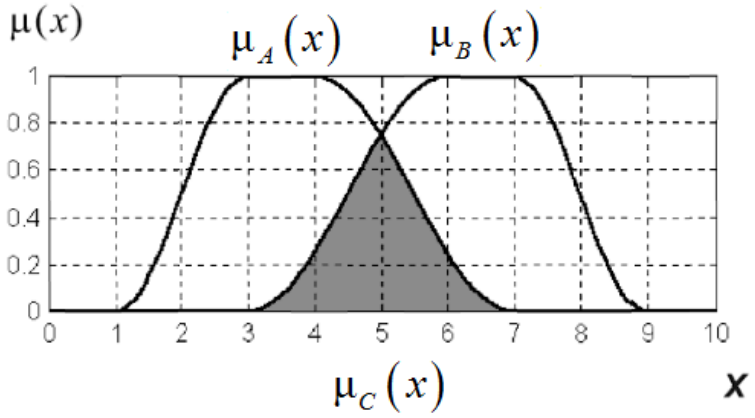


Рис. 7.9. Графическое представление операции пересечения нечетких множеств

В качестве примера рассмотрим нечеткое множество *C* «Небольшое натуральное число, приближенно равное трем», которое в свою очередь представляет пересечение множеств *A* «Небольшое натуральное число» и *B* «Натуральное число, приближенно равное трем».

$$A=\{\langle 1,1\rangle,\langle 2,1\rangle,\langle 3,0.9\rangle,\langle 4,0.8\rangle,\langle 5,0.6\rangle,\langle 6,0.5\rangle,\langle 7,0.4\rangle,\langle 8,0.2\rangle,\langle 9,0.1\rangle\},$$

$$B=\{\langle 1,0.3\rangle,\langle 2,0.6\rangle,\langle 3,1\rangle,\langle 4,0.7\rangle,\langle 5,0.4\rangle,\langle 6,0.2\rangle,\langle 7,0\rangle,\langle 8,0\rangle,\langle 9,0\rangle\},$$

$$C=A\cap B=\{\langle 1,0.3\rangle,\langle 2,0.6\rangle,\langle 3,0.9\rangle,\langle 4,0.7\rangle,\langle 5,0.4\rangle,\langle 6,0.2\rangle,\langle 7,0\rangle,\langle 8,0\rangle,\langle 9,0\rangle\}.$$

Объединением двух нечетких множеств *A* и *B* является третье нечеткое множество $C=A \cup B$, функция принадлежности которого определяется как:

$$\forall x \in X : \mu_C(x) = \max\{\mu_A(x), \mu_B(x)\}. \qquad (14)$$

Другими словами, *C* – наименьшее нечеткое множество, которое содержит (или доминирует) множества *A* и *B* (рис. 7.10).

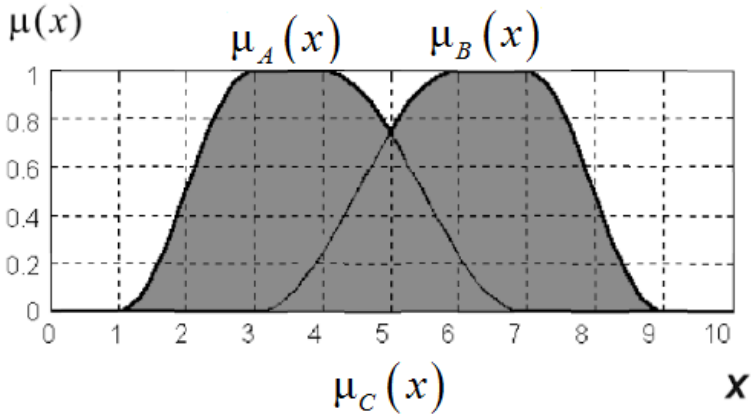


Рис. 7.10. Графическое представление операции объединения нечетких множеств

В качестве примера рассмотрим нечеткое множество *C* «Натуральное число, приближенно равное трем или пяти», которое в свою очередь представляет объединение множеств *A* «Натуральное число, приближенно равное пяти» и *B* «Натуральное число, приближенно равное трем».

$$A=\{\langle 1,0\rangle,\langle 2,0.1\rangle,\langle 3,0.4\rangle,\langle 4,0.7\rangle,\langle 5,1\rangle,\langle 6,0.8\rangle,\langle 7,0.4\rangle,\langle 8,0.2\rangle,\langle 9,0.1\rangle\},$$

$$B=\{\langle 1,0.3\rangle,\langle 2,0.6\rangle,\langle 3,1\rangle,\langle 4,0.7\rangle,\langle 5,0.4\rangle,\langle 6,0.2\rangle,\langle 7,0\rangle,\langle 8,0\rangle,\langle 9,0\rangle\},$$

$$C=A\cup B=\{\langle 1,0.3\rangle,\langle 2,0.6\rangle,\langle 3,1\rangle,\langle 4,0.7\rangle,\langle 5,1\rangle,\langle 6,0.8\rangle,\langle 7,0.4\rangle,\langle 8,0.2\rangle,\langle 9,0.1\rangle\}.$$

Вышерассмотренные операции пересечения и объединения по своей сути являются частными случаями данных операций над нечеткими множествами. Корректнее их стоит называть алгебраическим пересечением и объединением, соответственно. В теории нечетких множеств пересечение нечетких множеств в общем виде представляет *T*-норму (*t*-норму), а объединение – *t*-конорму (*s*-норму) [30]. *T*-норма и *T*-конорма могут представляться алгебраическими (13, 14), граничными, драстическими и другими операциями.

Разностью двух нечетких множеств *A* и *B* является третье нечеткое множество *C*=*A* \ *B*, функция принадлежности которого определяется как (рис. 7.11):

$$\forall x\in X:\mu_C(x)=\max\{\mu_A(x)-\mu_B(x),0\}. \qquad (15)$$

Другими словами, *C* – наименьшее нечеткое множество, которое содержит (или доминирует) множества *A* и *B* (рис. 7.11).

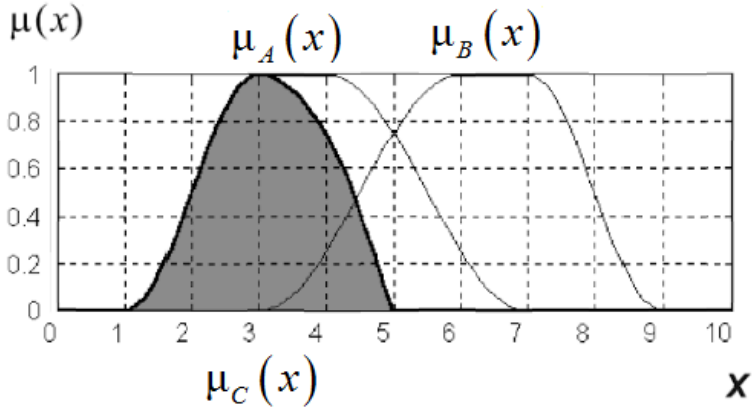


Рис. 7.11. Графическое представление операции разности нечетких множеств

В качестве примера рассмотрим нечеткое множество *C* «Небольшое натуральное число, не являющееся приближенно равным трем», которое в свою очередь представляет разность множеств *A* «Небольшое натуральное число» и *B* «Натуральное число, приближенно равное трем».

$$A=\{\langle 1,1\rangle,\langle 2,1\rangle,\langle 3,0.9\rangle,\langle 4,0.8\rangle,\langle 5,0.6\rangle,\langle 6,0.5\rangle,\langle 7,0.4\rangle,\langle 8,0.2\rangle,\langle 9,0.1\rangle\},$$

$$B = \{\langle 1, 0.3\rangle, \langle 2, 0.6\rangle, \langle 3, 1\rangle, \langle 4, 0.7\rangle, \langle 5, 0.4\rangle, \langle 6, 0.2\rangle, \langle 7, 0\rangle, \langle 8, 0\rangle, \langle 9, 0\rangle\},$$

$$C = A \setminus B = \{\langle 1, 0.7\rangle, \langle 2, 0.4\rangle, \langle 3, 0\rangle, \langle 4, 0.1\rangle, \langle 5, 0.2\rangle, \langle 6, 0.3\rangle, \langle 7, 0.4\rangle, \langle 8, 0.2\rangle, \langle 9, 0.1\rangle\}.$$

Следует отметить унарную операцию дополнения нечеткого множества. Дополнением нечеткого множества A является нечеткое множество $\overline{A}$, функция принадлежности которого определяется как (рис. 7.12):

$$\forall x \in X : \mu_{\overline{A}}(x) = 1 - \mu_A(x). \quad (16)$$

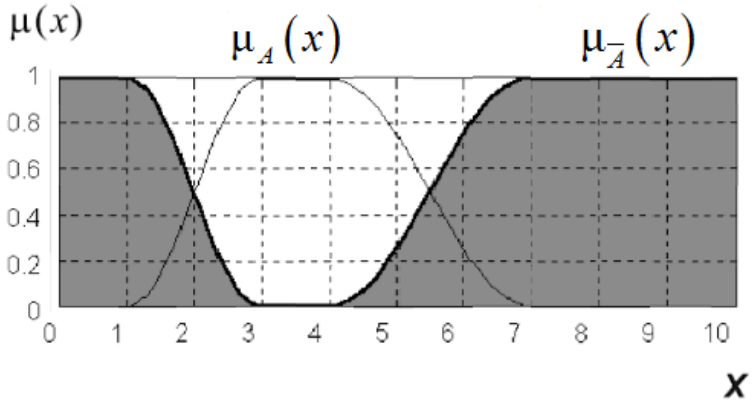


Рис. 7.12. Графическое представление операции дополнения нечеткого множества

В качестве примера рассмотрим нечеткое множество $\overline{A}$ «Натуральное число, не равное приближенно трем», которое в свою очередь представляет дополнение множества *A* «Натуральное число, приближенно равное трем».

$$A = \{\langle 1, 0.3\rangle, \langle 2, 0.6\rangle, \langle 3, 1\rangle, \langle 4, 0.7\rangle, \langle 5, 0.4\rangle, \langle 6, 0.2\rangle, \langle 7, 0\rangle, \langle 8, 0\rangle, \langle 9, 0\rangle\},$$

$$\overline{A} = \{\langle 1, 0.7\rangle, \langle 2, 0.4\rangle, \langle 3, 0\rangle, \langle 4, 0.3\rangle, \langle 5, 0.6\rangle, \langle 6, 0.8\rangle, \langle 7, 1\rangle, \langle 8, 1\rangle, \langle 9, 1\rangle\}.$$

Существуют и альтернативные представления вышеуказанных операций (а также их расширенные формы). Заинтересованный читатель может найти их в [30].

Наряду с понятием нечеткого множества фундаментальным понятием является «нечеткое отношение». Оно является важным при представлении субъективных высказываний, содержащих 2 и более аргументов, например, «*x* приблизительно равно *y*».

Нечеткое *N*-арное отношение *R*, заданное на множествах X_1, X_2, …, X_N определяется как нечеткое множество, которое определено на декартовом произведении этих множеств:

$$\forall x_1 \in X_1, \forall x_2 \in X_2, ..., \forall x_N \in X_N : R = \left\{\left\langle (x_1, x_2, ..., x_N), \mu_R(x_1, x_2, ..., x_N)\right\rangle\right\}, \quad (17)$$

где $\mu_R(\cdot)$ – функция принадлежности, которая каждой последовательности x_1, x_2, …, x_N приписывает вещественное число из интервала [0,1], интерпретируемое как сила связи между элементами последовательности:

$\mu_R : X_1 \times X_2 \times ... \times X_N \to [0,1]$.

Нечеткое отношение, содержащее 2 элемента, как и в случае классических множеств, принято называть бинарным, 3 элемента – тернарным.

Для примера раскроем вышеупомянутое высказывание *"x* приблизительно равно *y"*. Данное высказывание представляет собой бинарное отношение, определенное на декартовом произведении множеств *X* и *Y*. Допустим, $X = \{3, 4, 5\}$, а $Y=\{4, 5, 6\}$. Отношение *R* в таком случае можно определить как:

$$R = \begin{Bmatrix} \langle (3,4), 0.8 \rangle, \langle (3,5), 0.6 \rangle, \langle (3,6), 0.4 \rangle, \\ \langle (4,4), 1 \rangle, \langle (4,5), 0.8 \rangle, \langle (4,6), 0.6 \rangle, \\ \langle (5,4), 0.8 \rangle, \langle (5,5), 1 \rangle, \langle (5,6), 0.8 \rangle \end{Bmatrix}.$$

Кроме вышеприведенной формы списка с явным перечислением, нечеткие отношения могут быть представлены:

- Аналитически путем использования математического выражения для соответствующей функции принадлежности:

$$\mu_R(x,y) = \begin{cases} 1.0, & \text{если } x = y \\ 0.8, & \text{если } |x-y| = 1 \\ 0.6, & \text{если } |x-y| = 2 \\ 0.4, & \text{если } |x-y| = 3 \end{cases};$$

- Графически в форме некоторой поверхности или совокупности отдельных точек в трехмерном пространстве (рис. 7.13);

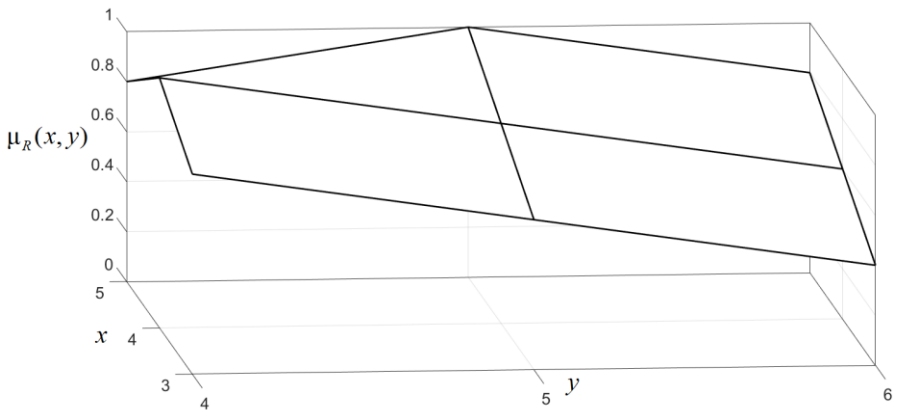


Рис. 7.13. Графическое представление нечеткого отношения

- В виде матрицы нечеткого отношения:

$$M_R = \begin{bmatrix} 0.8 & 0.6 & 0.4 \\ 1 & 0.8 & 0.6 \\ 0.8 & 1 & 0.8 \end{bmatrix};$$

- В форме нечеткого графа (рис. 7.14):

$G = \langle V, E, \mu_G \rangle$,

где $V = \{v\}$ – множество вершин нечеткого графа, условно представляющих элементы универсума;

$E = \{e\}$ – множество дуг нечеткого графа, условно представляющих отношения между элементами универсума;

μ_G – функция принадлежности дуг данному нечеткому графу ($\mu_G : E \to [0,1]$).

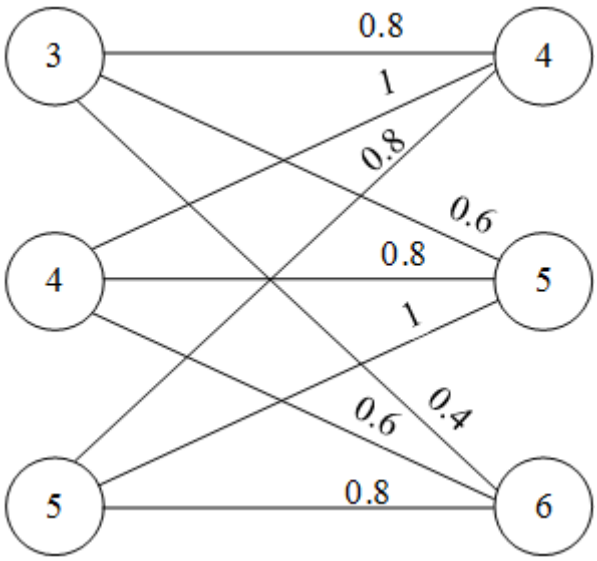


Рис. 7.14. Графовое представление нечеткого отношения

Для нечетких отношений справедливы те же операции, что и для нечетких множеств. В то же время, для нечетких отношений используются специфические операции, среди которых важную роль играет операция композиции нечетких отношений. Физически, композицией нечетких отношений является вычисление отношения между *x* и *z*, когда известны отношения между *x* и *y* и между *y* и *z*. Формально, композиция – нечеткое отношение $Q \otimes R$, заданное на декартовом произведении $X \times Z$ (при этом Q задано на $X \times Y$, а R – на $Y \times Z$), имеющее функцию принадлежности:

$$\forall (x,z) \in X \times Z : \mu_{Q \otimes R}(x,z) = \max_{y \in Y} \left\{ \min \left\{ \mu_Q(x,y), \mu_R(y,z) \right\} \right\}.$$

Такое выражение иногда называют максиминной сверткой нечетких множеств и обозначают как $Q \circ R$.

Для пояснения понятия композиции рассмотрим типовой пример, связанный с консалтингом в области выбора профессии для получения соответствующей специальности. Допустим, нам известны два отношения – $R = \{\langle (x,y), \mu_R(x,y) \rangle\}$ и $Q = \{\langle (y,z), \mu_Q(y,z) \rangle\}$. Первое обозначает психофизиологическое профилирование специальностей (табл. 7.1), второе – психофи-

зиологическое профилирование кандидатов (табл.7.2). При этом $x \in X$ – специальность, $y \in Y$ – психофизиологическая характеристика, $z \in Z$ – идентификатор (фамилия) кандидата. На основе имеющейся информации необходимо представить, насколько хорошо каждый кандидат подходит к той или иной специальности.

Таблица 7.1

Профилирование специальностей

	Быстрота и гибкость, y_1	Умение быстро принимать решения, y_2	Устойчивость и концентрация внимания, y_3	Зрительная память, y_4	Быстрота реакции, y_5
Менеджер, x_1	0.9	0.9	0.8	0.4	0.5
Программист, x_2	0.8	0.5	0.9	0.3	0.1
Водитель, x_3	0.3	0.9	0.6	0.5	0.9
Секретарь, x_4	0.5	0.4	0.5	0.5	0.2
Переводчик, x_5	0.7	0.8	0.8	0.2	0.6
	Двигательная память, y_6	Физическая выносливость, y_7	Координация движения, y_8	Эмоциональная устойчивость, y_9	Ответственность, y_{10}
Менеджер, x_1	0.3	0.6	0.2	0.9	0.8
Программист, x_2	0.2	0.2	0.2	0.5	0.5
Водитель, x_3	0.8	0.9	0.8	0.6	0.3
Секретарь, x_4	0.2	0.3	0.3	0.9	0.8
Переводчик, x_5	0.2	0.2	0.3	0.3	0.2

Таблица 7.2

Профилирование кандидатов

	Петров, z_1	Иванов, z_2	Сидоров, z_3	Васильева, z_4	Григорьева, z_5
Быстрота и гибкость, y_1	0.9	0.8	0.7	0.9	1
Умение быстро принимать решения, y_2	0.6	0.4	0.8	0.5	0.6
Устойчивость и концентрация внимания, y_3	0.5	0.2	0.3	0.8	0.7
Зрительная память, y_4	0.5	0.9	0.5	0.8	0.4
Быстрота реакции, y_5	1	0.6	0.5	0.7	0.4
Двигательная память, y_6	0.4	0.5	1	0,7	0.8
Физическая выносливость, y_7	0.5	0.8	0.9	0.5	0.4
Координация движения, y_8	0.5	0.6	0.7	0.6	0.5
Эмоциональная устойчивость, y_9	0.8	1	0.2	0.5	0.6
Ответственность, y_{10}	0.3	0.5	0.9	0.6	0.8

Матрицы исходных нечетких отношений имеют следующий вид:

$$M_S = \begin{bmatrix} 0.9 & 0.9 & 0.8 & 0.4 & 0.5 & 0.3 & 0.6 & 0.2 & 0.9 & 0.8 \\ 0.8 & 0.5 & 0.9 & 0.3 & 0.1 & 0.2 & 0.2 & 0.2 & 0.5 & 0.5 \\ 0.3 & 0.9 & 0.6 & 0.5 & 0.9 & 0.8 & 0.9 & 0.8 & 0.6 & 0.3 \\ 0.5 & 0.4 & 0.5 & 0.5 & 0.2 & 0.2 & 0.3 & 0.3 & 0.9 & 0.8 \\ 0.7 & 0.8 & 0.8 & 0.2 & 0.6 & 0.2 & 0.2 & 0.3 & 0.3 & 0.2 \end{bmatrix};$$

$$M_Q = \begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.9 & 1 \\ 0.6 & 0.4 & 0.8 & 0.5 & 0.6 \\ 0.5 & 0.2 & 0.3 & 0.8 & 0.7 \\ 0.5 & 0.9 & 0.5 & 0.8 & 0.4 \\ 1 & 0.6 & 0.5 & 0.7 & 0.4 \\ 0.4 & 0.5 & 1 & 0.7 & 0.8 \\ 0.5 & 0.8 & 0.9 & 0.5 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.6 & 0.5 \\ 0.8 & 1 & 0.2 & 0.5 & 0.6 \\ 0.3 & 0.5 & 0.9 & 0.6 & 0.8 \end{bmatrix}.$$

Таким образом, к примеру, результат нечеткой композиции для характеристики Петрова как менеджера будет следующим:

$$\mu_{Q\otimes R}(x_1, z_1) = \max \begin{Bmatrix} \min\{\mu_Q(x_1, y_1), \mu_R(y_1, z_1)\}, \min\{\mu_Q(x_1, y_2), \mu_R(y_2, z_1)\}, ..., \\ \min\{\mu_Q(x_1, y_{10}), \mu_R(y_{10}, z_1)\} \end{Bmatrix} =$$

$$= \max \begin{Bmatrix} \min\{0.9, 0.9\}, \min\{0.9, 0.6\}, \min\{0.8, 0.5\}, \min\{0.4, 0.5\}, \min\{0.5, 1\}, \\ \min\{0.3, 0.4\}, \min\{0.6, 0.5\}, \min\{0.2, 0.5\}, \min\{0.9, 0.8\}, \min\{0.8, 0.3\} \end{Bmatrix} = 0.9$$

.

Результаты остальных композиций представляются в виде матрицы:

$$M_{S\circ Q} = \begin{bmatrix} 0.9 & 0.9 & 0.8 & 0.9 & 0.9 \\ 0.8 & 0.8 & 0.7 & 0.8 & 0.8 \\ 0.9 & 0.8 & 0.9 & 0.7 & 0.8 \\ 0.8 & 0.9 & 0.8 & 0.6 & 0.8 \\ 0.7 & 0.7 & 0.8 & 0.8 & 0.7 \end{bmatrix}.$$

Для наглядности результаты также представлены в табл. 7.3.

Таблица 7.3

Характеристика кандидатов по специальностям

	Петров, z_1	Иванов, z_2	Сидоров, z_3	Васильева, z_4	Григорьева, z_5
Менеджер, x_1	0.9	0.9	0.8	0.9	0.9
Программист, x_2	0.8	0.8	0.7	0.8	0.8
Водитель, x_3	0.9	0.8	0.9	0.7	0.8
Секретарь, x_4	0.8	0.9	0.8	0.6	0.8
Переводчик, x_5	0.7	0.7	0.8	0.8	0.7

7.5 Нечеткие величины

Как и в традиционном случае, обработка информации в системе, функционирующей на базе нечетких множеств, основана на количественном представлении входных и выходных данных. Такое представление связано с рассмотрением специальных нечетких множеств, называемых нечеткими величинами. Отличие таких нечетких множеств состоит в том, что оно задано только на множестве действительных чисел (элемент нечеткого множества всегда имеет количественное значение).

Формально, нечеткая величина – это такое нечеткое множество *A*, для которого функция принадлежности есть отображение

$$\mu_A(x): \square \to [0,1]. \qquad (18)$$

Среди нечетких величин наибольший интерес представляет нечеткий интервал и нечеткое число.

Нечетким интервалом называется нечеткая величина *A* с выпуклой функцией принадлежности, т.е. для которой справедливо условие:

$$\forall x \;\; a < x < b : \mu_A(x) \geq \min\{\mu_A(a), \mu_A(b)\}. \qquad (19)$$

Нечетким числом называется нечеткий интервал *A* с унимодальной функцией принадлежности, т.е. функция принадлежности которого содержит только одно максимальное значение.

Для нечетких чисел помимо нечетких отношений могут быть использованы и простейшие бинарные арифметические операции $f(x, y)$, такие как сложение, вычитание, умножение, деление. Функция принадлежности результата арифметической операции представляется как:

$$\mu_C(z) = \sup_{z=f(x,y)} \{\min\{\mu_A(x), \mu_B(y)\}\}, \qquad (20)$$

где $A =< x, \mu_A(x) >$, и $B =< x, \mu_B(y) >$ – аргументы арифметической операции;

$C =< z, \mu_C(z) >$ – результат арифметической операции.

В качестве примера рассмотрим выполнение операции сложения двух чисел *I*, приближенно равных единице и представленных в виде множества:

$$I = \{< 0, 0.2 >, < 1, 1.0 >, < 2, 0.2 >\}.$$

Множество *I*+*I* вычисляется на основе формулы (20):

$$\mu_{I+I}(x+y) = \sup_{x+y} \{\min\{\mu_I(x), \mu_I(y)\}\}.$$

То есть для $x+y = 2$ результат будет следующим:

$$\mu_{I+I}(2) = \max\{\min\{\mu_I(0), \mu_I(2)\}, \min\{\mu_I(1), \mu_I(1)\}, \min\{\mu_I(2), \mu_I(0)\}\} =$$
$$= \max\{\min\{0.2, 0.2\}, \min\{1,1\}, \min\{0.2, 0.2\}\} = 1,$$

а итоговое множество будет иметь вид:

$$I + I = \{< 0, 0.2 >, < 1, 0.2 >, < 2, 1 >, < 3, 0.2 >, < 4, 0.2 >\}$$

Вышеуказанные понятия нечеткого интервала и нечеткого числа являются довольно общими и в этом виде не применяются. На практике удобно использовать аналитические формы представления нечетких интервалов и чисел.

Все аналитические функции представления нечетких величин аппроксимируют так называемыми функциями (*L*-*R*)-типа.

Для нечеткого числа функция (*L*-*R*)-типа записывается в виде:

$$\mu_A(x) = \begin{cases} L\left(\dfrac{a-x}{\alpha}\right) & \text{если } x \le a \\ R\left(\dfrac{x-a}{\beta}\right) & \text{если } x > a \end{cases}, \qquad (21)$$

где $L(\cdot)$ и $R(\cdot)$ – произвольные функции типа $f : R \to [0,1]$,

$\alpha>0$ и $\beta>0$ – коэффициенты нечеткости,

a – мода нечеткого числа.

Нечеткие интервалы представимы следующей функцией (*L*-*R*)-типа:

$$\mu_A(x) = \begin{cases} L\left(\dfrac{a-x}{\alpha}\right) & \text{если } x \le a \\ 1 & \text{если } a<x<b, \\ R\left(\dfrac{x-b}{\beta}\right) & \text{если } x \ge b \end{cases} \qquad (22)$$

где a и b – верхняя и нижняя модальные значения (ядро нечеткого множества).

Примерами популярных *L* или *R* функций являются следующие:

$$f(u) = e^{-|u|^p}, \qquad (23)$$

$$f(u) = \frac{1}{1+|u|^p}, \qquad (24)$$

где $p>0$ – некоторый параметр.

Графики этих функций представлены на рис. 7.15.

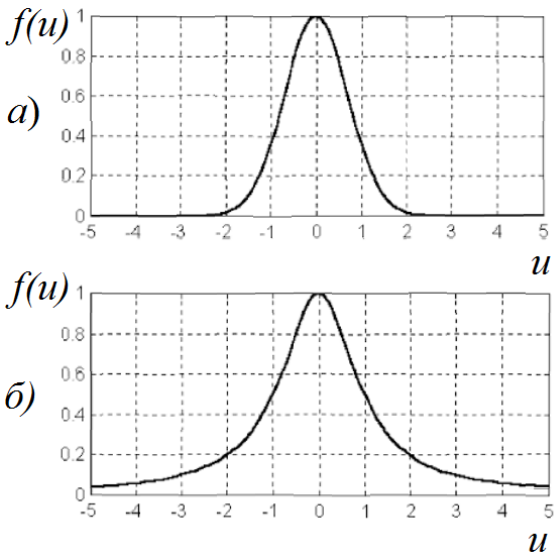


Рис. 7.15. Графики функций, представленных в виде (23) (а) и (24) (б), для значения параметра $p = 2$.

Из определения нечеткого числа (21) и нечеткого интервала (22) удобно обозначать первое как $A_{LR} = < a, \alpha, \beta >_{LR}$, а второй как $A_{LR} = < a, b, \alpha, \beta >_{LR}$.

Такое представление делает возможным уточнение арифметических операций.

Операция сложения нечетких чисел A и B (L-R)-типа обозначается через $C = < a, \alpha, \beta >_{LR}$, где:

$$a = a_1 + a_2,\ \alpha = \alpha_1 + \alpha_2,\ \beta = \beta_1 + \beta_2. \qquad (25)$$

Операция вычитания нечетких чисел A и B (L-R)-типа обозначается через $C = <a, \alpha, \beta>_{LR}$, где:

$$a = a_1 - a_2,\ \alpha = \alpha_1 + \alpha_2,\ \beta = \beta_1 + \beta_2. \qquad (26)$$

Другими словами, сложение и умножение нечетких чисел заключается в соответствующих операциях между их модами и расширении «размытости» функции принадлежности путем сложения коэффициентов нечеткости.

Представление операции умножения и деления для нечетких чисел зависит от знаков мод множителей.

Умножение нечетких чисел A и B (L-R)-типа, для которых $a_1>0$ и $a_2>0$ обозначается через $C = < a, \alpha, \beta >_{LR}$, где:

$$a = a_1 a_2,\ \alpha = a_2\alpha_1 + a_1\alpha_2,\ \beta = a_2\beta_1 + a_1\beta_2. \qquad (27)$$

Умножение нечетких чисел A и B (L-R)-типа, для которых $a_1<0$ и $a_2>0$ обозначается через $C = < a, \alpha, \beta >_{LR}$, где:

$$a = a_1 a_2,\ \alpha = a_2\alpha_1 - a_1\beta_2,\ \beta = a_2\beta_1 - a_1\alpha_2. \qquad (28)$$

Умножение нечетких чисел A и B (L-R)-типа, для которых $a_1<0$ и $a_2<0$ обозначается через $C = < a, \alpha, \beta >_{LR}$, где:

$$a = a_1 a_2\,,\ \alpha = |\, a_2\beta_1 + a_1\beta_2\,|\,,\ \beta = |\, a_2\alpha_1 + a_1\alpha_2\,|\,. \qquad (29)$$

Другими словами, умножение нечетких чисел заключается в умножении их мод и расширения «размытости» путем попарного умножения коэффициентов нечеткости чисел |A| и |B| (рис. 7.16).

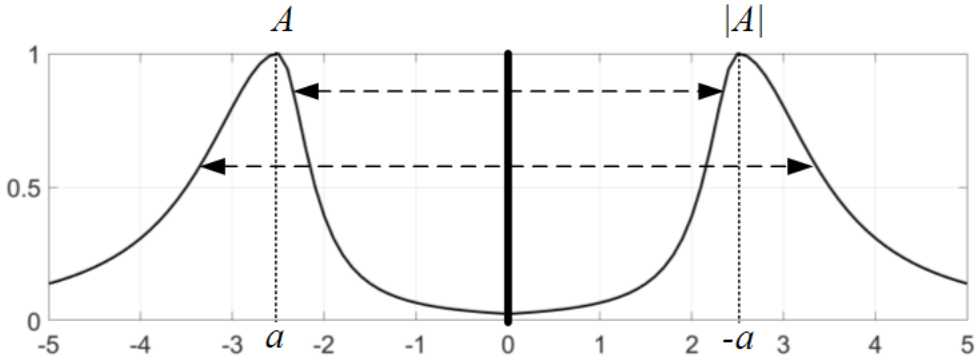


Рис. 7.16. Пояснение операции |A|

Деление нечетких чисел A и B (L-R)-типа, для которых $a_1 > 0$ и $a_2 > 0$ обозначается через $C = < a, \alpha, \beta >_{LR}$, где:

$$a = a_1 / a_2\,,\ \alpha = \left(a_1\beta_2 + a_2\alpha_1\right)/ a_2^{\,2}\,,\ \beta = \left(a_1\alpha_2 + a_2\beta_1\right)/ a_2^{\,2}\,. \qquad (30)$$

Деление в случае $a_1 < 0$ и $a_2 > 0$, $a_1 < 0$ и $a_2 < 0$ определяется аналогично при использовании абсолютных значений |A| и |B|, как в случае умножения.

В качестве примера рассмотрим арифметические операции между нечеткими числами «нечеткая тройка» $A=<3,2,2>$ и «нечеткая двойка» $B=<2,1,1>$. При этом функция принадлежности представляется в виде (23) со значением параметра $p = 2$.

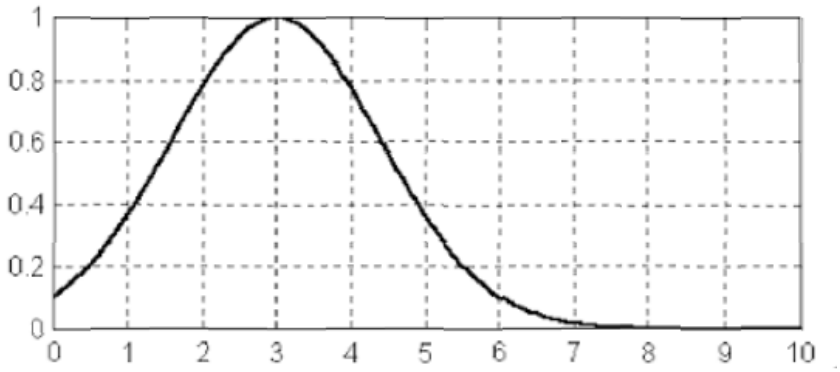


Рис. 7.17. График нечеткого числа «нечеткая тройка»

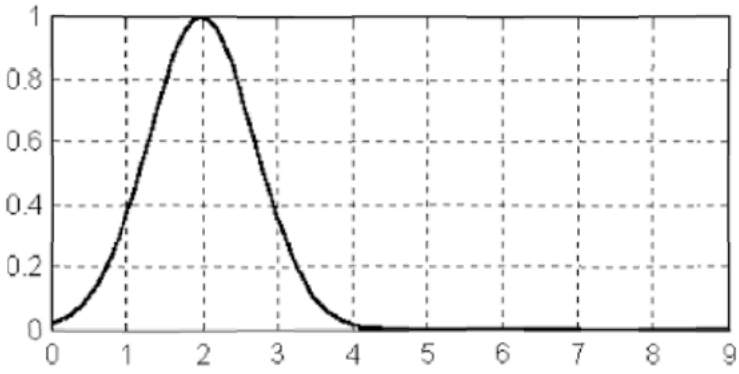


Рис. 7.18. График нечеткого числа «нечеткая двойка»

Результаты выполнения арифметических операций представлены на рис. 7.19, 7.20, 7.21 и рис 7.22.

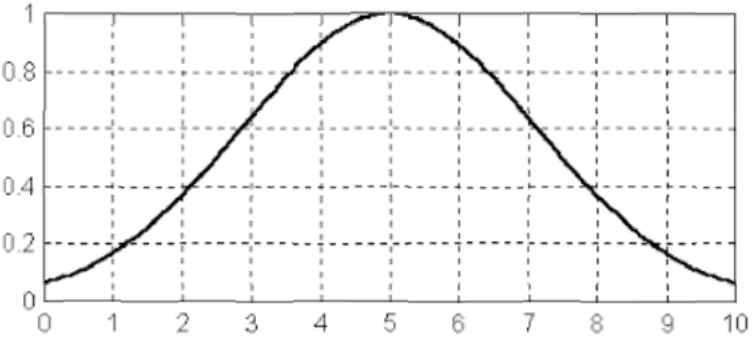


Рис. 7.19. График нечеткого числа «нечеткая пятерка»

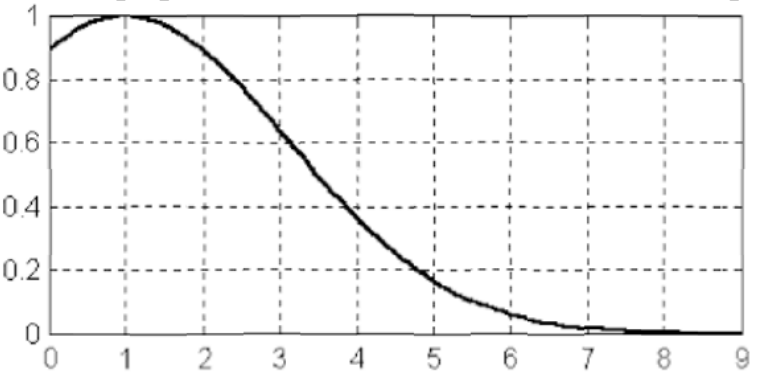


Рис. 7.20. График нечеткого числа «нечеткая единица»

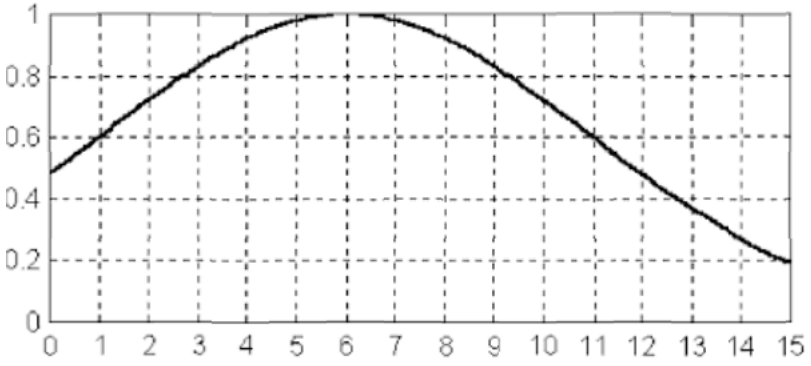


Рис. 7.21. График нечеткого числа «нечеткая шестерка»

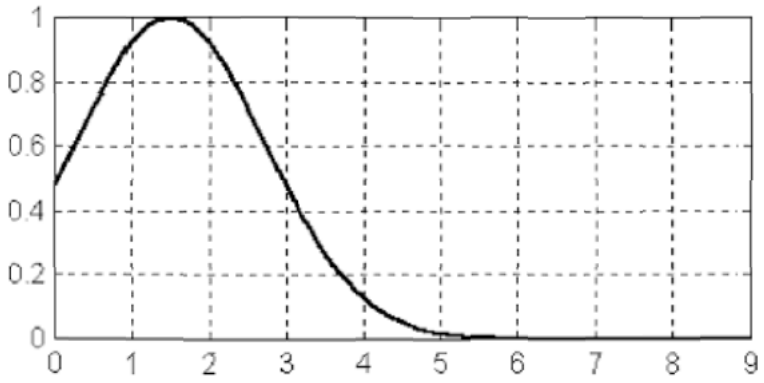


Рис. 7.22. График нечеткого числа «нечеткая дробь 2/3»

Наибольший практический интерес нашли (*L*-*R*)-функции, представленные в линейной форме, а именно треугольной для нечеткого числа и трапециевидной для нечеткого интервала.

Треугольное нечеткое число (ТНЧ) – нечеткое число $<a,\alpha,\beta>$, функция принадлежности которого представлена треугольной функцией (8). При этом $\alpha = b - a,\ \beta = c - b$.

Трапециевидный нечеткий интервал (ТНИ) – нормальный нечеткий интервал $<a,b,\alpha,\beta>_T$, функция принадлежности которого представлена трапециевидной функцией (9). При этом $\alpha = b - a,\ \beta = d - c$.

В качестве примера ТНЧ можно взять «нечеткую тройку» <3,1,2> (рис 7.23). Примером ТНИ служит «нечеткий интервал от 4 до 6» <4,6,2,1> (рис 7.24).

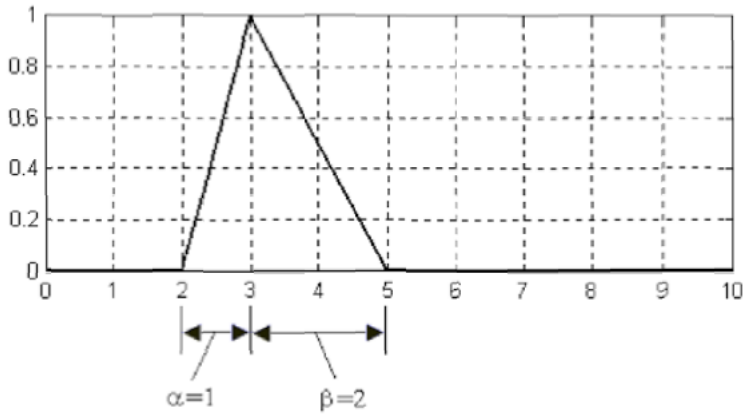


Рис. 7.23. График ТНЧ «нечеткая тройка»

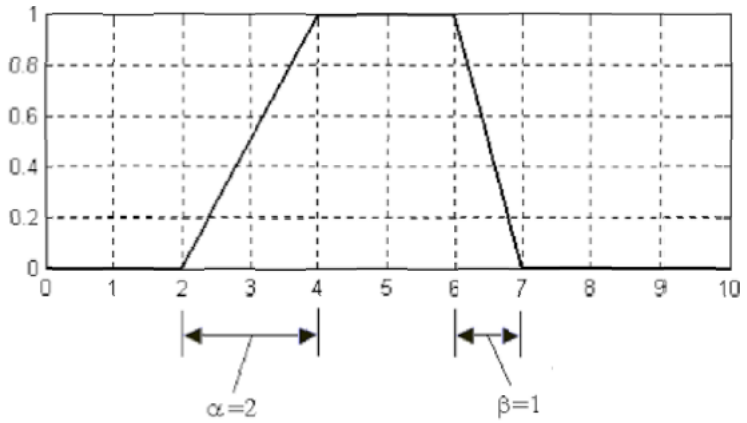


Рис. 7.24. График ТНИ «нечеткий интервал от 4 до 6»

Для оперирования ТНЧ и ТНИ используют те же формулы, что и для (*L*-*R*)-функций общего типа.

7.6 Лингвистическая переменная

Для использования нечетких множеств в прикладных системах необходимым является формализация отдельных переменных. Одной из таких формализаций является понятия нечеткой и лингвистической переменной, которая часто используется в нечетком управлении для представления входных и выходных данных управляемой системы.

Нечеткая переменная определяется как кортеж

$$\langle \alpha, X, A \rangle, \tag{31}$$

где α – название нечеткой переменной (ее качественный смысл);

X – область определения нечеткой переменной;

A – нечеткое множество, описывающее возможные значения переменной A.

В качестве примера нечеткой переменной можно привести нечеткое множество, которое характеризует «горячий кофе»:

$$\langle Горячий\ кофе, \{x \mid 0°C < x < 100°C\}, \{x, \mu(x)\} \rangle.$$

Лингвистическая переменная является обобщением нечеткой переменной на случай множественных качественных характеристик:

$$\langle \beta, T, X, G, M \rangle, \tag{32}$$

где β – название лингвистической переменной;

T – множество значений (или терм-множество), которые может принимать лингвистическая переменная, в виде названий нечетких переменных;

X – область определения лингвистической переменной;

G – множество синтаксических процедур, которые могут быть использованы для оперирования лингвистической переменной и получения новой информации на ее основе;

M – множество семантических процедур, которые формализуют множество T и множество G.

В качестве примера лингвистической переменной можно привести субъективную оценку температуры кофе в виде (32), где:

β – температура (степень «горячести-холодности») кофе;

$T = \{"холодный\ кофе","теплый\ кофе","горячий\ кофе"\}$;

$X = [0,100]$;

G – «И», «ИЛИ», «НЕ», «ОЧЕНЬ», «СЛЕГКА» и т.д.;

M – формализация описания нечетких переменных «холодный кофе», «теплый кофе», «горячий кофе», а также синтаксических связей типа $G(\cdot)$.

Процедуры задания нечетких переменных удобнее всего представлять графически на одном рисунке (рис. 7.25).

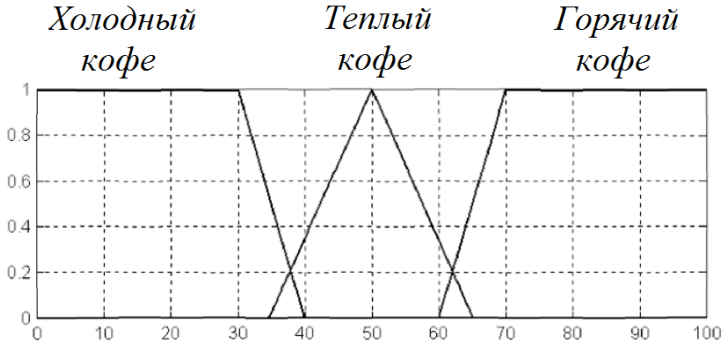


Рис. 7.25. Графики функций принадлежности нечетких множеств для представления субъективной информации о температуре кофе

Процедуры задания нечетких переменных удобнее всего представлять графически на одном рисунке.

В случае представления термов лингвистической переменной в виде (L-R)-функций имеются некоторые рекомендации для задания их формы относительно семантики терма (табл. 7.4).

*Таблица 7.4*Рекомендации по представлению термов лингвистической переменной

Терм лингвистической переменной	**(L-R)-представление**
Средний, около, приблизительно	$<a, \alpha, \beta>$; $\alpha, \beta < \infty$
Малый, низкий	$<a, \alpha, \beta>$; $\alpha = \infty, \beta < \infty$

Окончание табл. 8

Большой, высокий	$<a, \alpha, \beta>$; $\alpha < \infty$, $\beta = \infty$
Приблизительно в интервале (a,b)	$<a, b, \alpha, \beta>$; $\alpha, \beta < \infty$
Не выше b	$<a, b, \alpha, \beta>$; $\alpha = \infty$, $\beta < \infty$
Не меньше a	$<a, b, \alpha, \beta>$; $\alpha < \infty$, $\beta = \infty$

7.7 Нечеткая логика

Классическая логика, расширенная на нечеткие множества, называется нечеткой. Она предназначена для формализации субъективных человеческих суждений, которые более адекватно позволяют описывать ситуации с неопределенностью. В отличие от классической, нечеткая логика позволяет оценить степень истинности того или иного суждения.

Исходным понятием нечеткой логики является понятие элементарного нечеткого высказывания. Таким высказыванием является повествовательное предложение, выражающее законченную мысль, относительно истинности которой можно судить только с некоторой степенью уверенности. Главным отличием элементарного нечеткого высказывания от элементарного высказывания является следующий факт. Множество значений истинности элементарных высказываний состоит из двух элементов: {«истина», ложь»} ({0,1}). В нечеткой логике степень истинности элементарного нечеткого высказывания принимает значение из замкнутого интервала [0,1], причем 0 и 1 являются предельными значениями степени истинности со значениями «ложь» и «истина», соответственно.

Примерами нечетких высказываний являются следующие:

«Остап Бендер имеет довольно высокий рост»;

«Возможно, завтра пойдет дождь»;

«3 – малое число».

Нечеткие высказывания и операции с ними объединяются в нечеткую логику высказываний, обобщающую классическую логику высказываний (логику предикатов нулевого порядка). Для обобщения логики предикатов первого порядка вводится понятие «нечеткий предикат».

Нечеткий предикат, в первую очередь, представляет собой описание нечеткого отношение. А именно, k-местный нечеткий предикат $P\left(x_1, x_2, ..., x_N\right)$ отражает функцию принадлежности k-местного нечеткого отношения $R = \left\{\left\langle (x_1, x_2, ..., x_N), \mu_R(x_1, x_2, ..., x_N)\right\rangle\right\}$.

Как и в случае классической логики, так и в случае нечеткой, основными операциями над высказываниями являются логическое отрицание, конъюнкция, дизъюнкция и импликация. При этом отрицание нечеткого

высказывания A ($\overline{A}$, «НЕ-A») выполняется как операция дополнения соответствующего значения истинности (16). В случае конъюнкции ($A \wedge B$ «A И B») и дизъюнкции ($A \vee B$, «A ИЛИ B») нечетких высказываний A и $Б$ по аналогии с нечеткими множествами используют Т-норму и Т-конорму. Что касается нечеткой импликации ($A \to B$, «из A *следует* B»), здесь может быть использовано несколько формул, предложенных Заде, Мамдани, Лукасевичем, Гогеном и др. [30]. Наиболее практически используемая импликация – импликация Мамдани (или нечеткая импликация минимума корреляции):

$$\mathrm{T}(A \to B) = \min\left\{\mathrm{T}(A), T(B)\right\}, \qquad (33)$$

где $\mathrm{T}: U \to [0,1]$ – отображение истинности элементарного высказывания U.

Кроме вышеперечисленных, популярной операцией в нечеткой логике является нечеткая эквивалентность ($A \equiv B$, «A эквивалентно B»). Истинность такого высказывания определяется как:

$$\mathrm{T}(A \equiv B) = \min\left\{\max\left\{\mathrm{T}\left(\overline{A}\right), T(B)\right\}, \max\left\{\mathrm{T}(A), T\left(\overline{B}\right)\right\}\right\}. \qquad (34)$$

Центральное место в нечеткой логике естественным образом занимает процесс нечеткого вывода. Нечеткий вывод представляет собой некоторую процедуру (или механизм) получения нечетких заключений на основе нечетких условий или предпосылок. Достигнутые успехи в применении систем, базирующихся на нечетком выводе, послужили основой становления теории нечетких множеств как прикладной науки.

Основой механизма нечеткого вывода являются нечеткие продукционные правила (нечеткие продукции) вида:

ЕСЛИ A ТО B

При этом высказывания A и B принимают следующие формы:

1 «β есть α», где β – наименование лингвистической переменной, а α – ее значение, которому соответствует терм из терм-множества T лингвистической переменной β.

2 «β есть ∇α», где ∇ - модификатор вида «ОЧЕНЬ», «БОЛЕЕ-МЕНЕЕ», «МНОГО БОЛЬШЕ» и др., полученный с использованием процедур G и M лингвистической переменной β.

3 Составные высказывания, полученные из высказываний вида 1 и 2 и операций «И», «ИЛИ».

Стоит отметить, что заключение нечеткой продукции B в основном принимает форму 1 или 2.

Системы, основанные на механизме нечеткого вывода, (или просто системы нечеткого вывода) предназначены для преобразования значений входных переменных, поступающих в систему в процессе управления, в выходные переменные на основе использования нечетких продукций на основе следующего алгоритма:

1 Формирование базы правил;

2 Фаззификация входных переменных;

3 Агрегирование подусловий;

4 Активизация подзаключений;

5 Аккумулирование заключений;

6 Дефаззификация выходных переменных.

Ниже рассмотрим основные особенности каждого из этих этапов.

Формирование базы правил.

База правил представляет собой конечное множество нечетких продукций:

$П_1$: ЕСЛИ «Условие 1» ТО «Заключение 1»

$П_2$: ЕСЛИ «Условие 2» ТО «Заключение 2»

…

$П_N$: Если «Условие N» ТО «Заключение N».

Лингвистические переменные, используемые в высказываниях условий правил, называют входными лингвистическими переменными, а в высказываниях заключений – выходными лингвистическими переменными.

Формирование базы правил заключается в определении множества входных и выходных лингвистических переменных, а также в составлении базы правил на основе этих переменных. Формирование базы правил зависит от специфики решаемой задачи и используемого механизма нечеткого вывода. Основные виды механизмов нечеткого вывода будут рассмотрены позже.

Фаззификация входных переменных.

Под фаззификацией (введение нечеткости) понимается процесс нахождения значений функций принадлежности нечетких термов для конкретных входных данных. Целью фаззификации является установление соответствия между конкретным значением входной переменной *a* системы нечеткого вывода и значением функции принадлежности $\mu_\alpha(a)$ соответствующего ей терма α входной лингвистической переменной β. Значение $b=\mu_\alpha(a)$ и является результатом фаззификации подусловия «β есть α».

Этап фаззификации считается законченным, когда будут найдены все значения функции принадлежности для каждого из подусловий всех нечетких продукций базы правил.

Для иллюстрации выполнения этого этапа рассмотрим пример процесса фаззификации трех нечетких высказываний α_1 = «скорость автомобиля малая», α_2 = «скорость автомобиля средняя», α_3 = «скорость автомобиля высокая» (рис. 7.26). Предположим, что скорость автомобиля *a* = 55 км/ч. Тогда фаззификация дает следующие значения: $\mu_{\alpha_1}(a)=0, \mu_{\alpha_2}(a)\approx 0.67, \mu_{\alpha_3}(a)=0$.

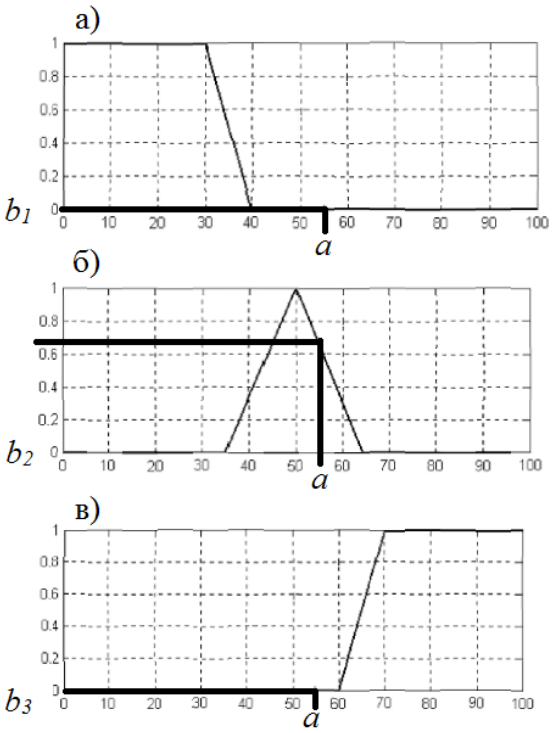


Рис. 7.26. Фаззификация подусловий для термов лингвистической переменной «скорость автомобиля»

Агрегирование подусловий

Агрегирование подусловий представляет собой процедуру определения истинности условий по каждому из правил системы нечеткого вывода.

Допустим, для каждого *j*-го подусловия *i*-го правила системы нечеткого вывода произведена фаззификация, т.е. известно множество $B_i = \{b_j\}$. Если нечеткая продукция представлена высказыванием типа 1 или 2, то данное множество содержит только один элемент, и степень истинности *i*-го правила b_i' равна значению этого элемента.

Если же условие содержит составные высказывания, то необходимо применить *T*-норму в случае связки «И» или *T*-конорму в случае связки «ИЛИ».

Этап агрегирования считается законченным, когда степени истинности всех правил найдены.

Для иллюстрации выполнения этого этапа рассмотрим пример процесса агрегирования двух нечетких высказываний (рис. 7.27):

«скорость автомобиля средняя» И «кофе горячий»;

«скорость автомобиля средняя» ИЛИ «кофе горячий».

Предположим, что скорость автомобиля $a_1 = 55$ км/ч, а температура кофе $a_2 = 70$ °С. Тогда агрегирование первого высказывания с использованием *T*-нормы вида (13) дает в результате $b' = 0.67$ (рис. 7.27,*а*). Агрегирование второго высказывания с использованием *T*-конормы вида (14) дает в результате $b' = 0.8$ (рис. 7.27,*б*).

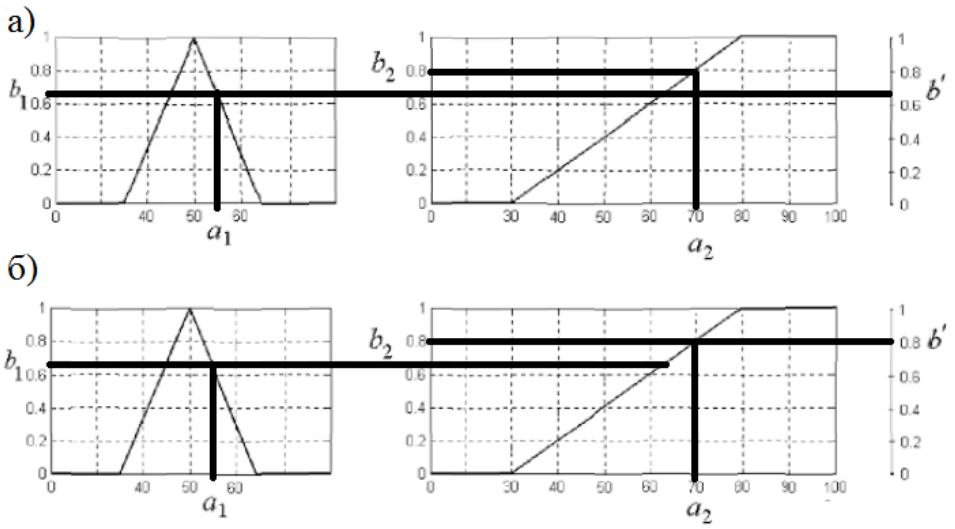


Рис. 7.27. Агрегация подусловий для высказываний со связкой «И» (*а*) и со связкой «ИЛИ» (*б*)

Активизация подзаключений

Активизация подзаключений представляет собой процесс нахождения функции принадлежности каждого из подзаключений нечетких продукций на основе полученного значения истинности условия.

Активизация может выполняться несколькими способами (min-активизация, prod-активизация и average-активизация) [30]. Наиболее популярным способом является min-активизация, основанная на методе импликации Мамдани (33):

$$\mu'(y) = \min\{F \cdot b', \mu(y)\}, \qquad (35)$$

Где $\mu'(y)$ – результирующая функция принадлежности для терма, представляющего подзаключение правила;

$F \in [0,1]$ – весовой коэффициент правила (в основном $F = 1$);

$\mu(y)$ – исходный вид функции принадлежности для терма, представляющего подзаключение правила.

Процесс активизации считается законченным, когда для каждого подзаключения будут определены функции принадлежности представляющего терма.

Для иллюстрации процесса активизации рассмотрим пример процесса активизации заключения для следующего правила:

ЕСЛИ «скорость автомобиля средняя» ТО «кофе горячий»

Предположим, что текущая скорость автомобиля $a = 55$ км/ч, а $F = 1$. Тогда результат min-активизации будет реализован функцией принадлежности, график которой отмечен темным цветом на рис. 7.28.

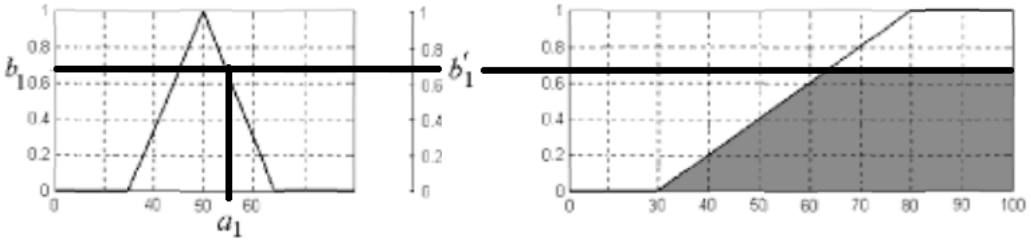


Рис. 7.28. Результат min-активизации заключения нечеткой продукции

Аккумулирование заключений

Аккумулирование (или аккумуляция) представляет собой процесс нахождения функции принадлежности для каждой из выходных лингвистических переменных путем объединения функций принадлежности подзаключений, полученных в результате этапа активизации. Объединение функций принадлежности производится путем использования операции *T*-конормы.

Для иллюстрации выполнения этого этапа рассмотрим пример аккумуляции для трех множеств, полученных в результате активизации для выходной лингвистической переменной «скорость автомобиля» (рис. 7.29). Функция принадлежности, полученная в результате аккумуляции, представлена на рис. 7.30.

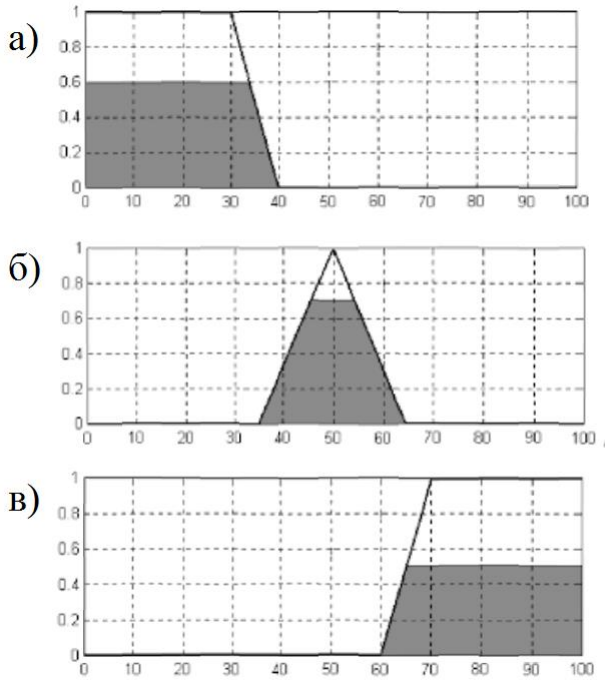


Рис. 7.29. Функции принадлежности (закрашены темным), полученные в результате активизации термов лингвистической переменной «скорость автомобиля»

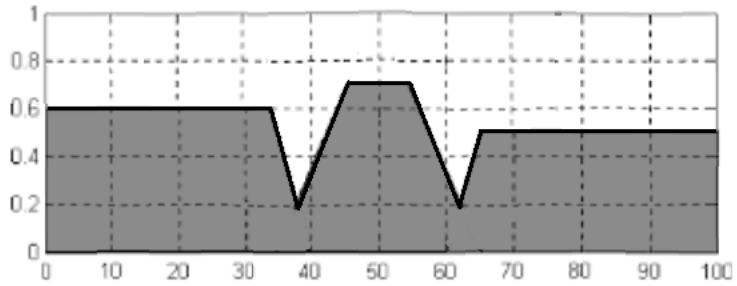


Рис. 7.30. Функции принадлежности, полученные в результате аккумуляции подзаключений, представленных на рис. 7.29

Дефаззификация выходных переменных

Дефаззификация представляет собой процесс нахождения обычного (не нечеткого) значения для каждой из выходных лингвистических переменных. Цель дефаззификации состоит в том, чтобы, используя процесс нечеткого вывода, можно было оперировать выходными значениями в других системах и устройствах. Выходное значение находится как оптимальное (или центральное) значение функции принадлежности, полученной на этапе аккумулирования заключений.

Этап дефаззификации считается законченным, когда найдены количественные значения всех выходных лингвистических переменных.

Для выполнения расчетов на этапе дефаззификации могут быть использованы различные формулы, которые получили название методов дефаззификации. Наиболее популярные методы – метод центра тяжести и метод центра площади.

Метод центра тяжести заключается в нахождении абсциссы центра фигуры, ограниченной результирующей функцией принадлежности (рис. 7.31):

$$y = \frac{\int_{\min}^{\max} x \cdot \mu(x) dx}{\int_{\min}^{\max} \mu(x) dx}. \qquad (36)$$

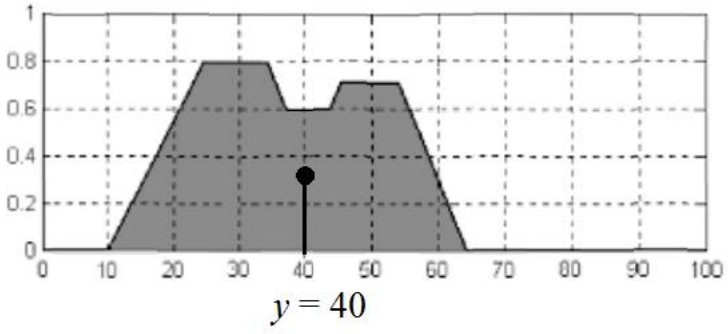


Рис. 7.31. Иллюстрация нахождения центра тяжести

В случае, если функция принадлежности задана в виде одноточечных множеств, то центр тяжести рассчитывается по формуле:

$$y = \frac{\sum_{i=1}^{n} x_i \cdot \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}. \qquad (37)$$

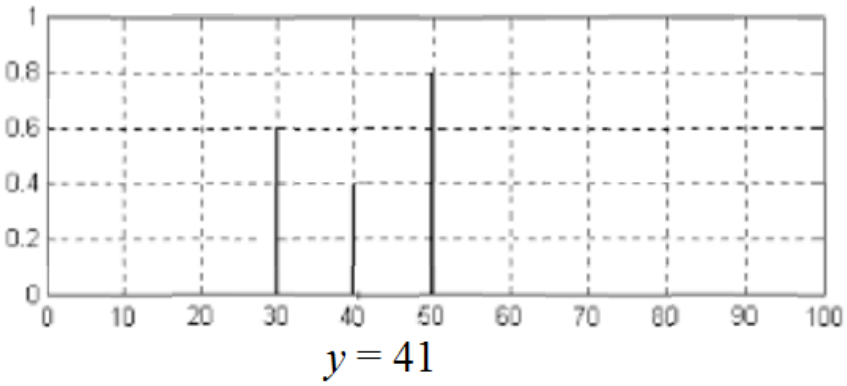


Рис. 7.32. Иллюстрация нахождения центра тяжести в случае задания функции принадлежности в виде одноточечных множеств

Метод центра площади заключается в нахождении абсциссы точки, в которой фигура, ограниченная результирующей функцией принадлежности, делится на две равные части (рис. 7.33):

$$\int_{\min}^{y} \mu(x)dx = \int_{y}^{\max} \mu(x)dx. \qquad (38)$$

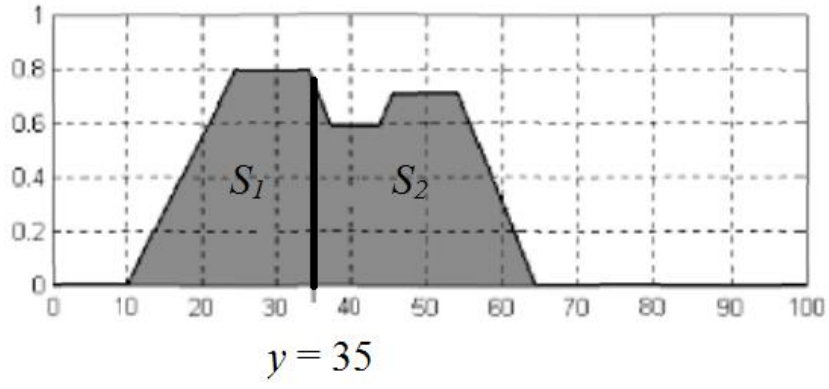


Рис. 7.33. Иллюстрация нахождения центра площади ($S_1 = S_2$)

Описанные выше этапы нечеткого вывода могут быть реализованы неоднозначным образом, поскольку включают себя отдельные параметры, которые надо специфицировать. Набор конкретных вариантов для пара-

метров каждого из этапов определяет алгоритм, который однозначно реализует нечеткий вывод в прикладных системах. Наиболее популярными алгоритмами являются алгоритм Мамдани и алгоритм Такаги-Сугено (или просто Сугено).

Алгоритм Мамдани

Алгоритм Ибрагима Мамдани является одним из первых, который нашел применение в системах нечеткого вывода. Он был предложен в 1975 году для управления паровым двигателем и по своей сути уточняет вышеперечисленные этапы:

1 Особенности формирования базы правил нечеткого вывода совпадают с рассмотренными выше при описании данного этапа.

2 Фаззификация входных переменных производится также аналогичным способом с вышеописанным.

3 Агрегирование подусловий производится с помощью min-операции (13) для конъюнкции и max-операции для дизъюнкции (14).

4 Активизация подзаключений осуществляется в виде min-активизации (35).

5 Аккумуляция заключений осуществляется с помощью max-операции (14).

6 Дефаззификация традиционно осуществляется в форме (36).

Представим далее пример нечеткого вывода с помощью алгоритма Мамдани. Допустим, нам необходимо запрограммировать робота корректировать скорость своего движения на основе дальности его от препятствия. Пусть имеется база правил вида:

ЕСЛИ «β_1 есть α_i» И «β_2 есть α_j» ТО «β_3 есть α_k»,

где β_1 – входная лингвистическая переменная «расстояние до препятствия» (Рис. 34);

β_2 – входная лингвистическая переменная «угол относительно препятствия» (рис. 7.35);

β_3 – выходная лингвистическая переменная «скорость» (рис. 7.36);

$\alpha_i \in$ {«Близко», «Далеко», «Очень далеко»} – терм лингвистической β_1;

$\alpha_j \in$ {«Малый», «Средний», «Большой»} – терм лингвистической переменной β_2;

$\alpha_k \in$ {«Очень медленная», «Медленная», «Быстрая», «Очень быстрая», «Наивысшая»} – терм лингвистической переменной β_3;

i, j, k – порядковые номера термов соответствующих лингвистических переменных.

Таблица 7.5

База правил Мамдани системы нечеткого вывода для управления скоростью робота

	Близко	**Далеко**	**Очень далеко**
Малый	Очень медленная	Медленная	Быстрая
Средний	Медленная	Быстрая	Очень быстрая
Большой	Быстрая	Очень быстрая	Наивысшая

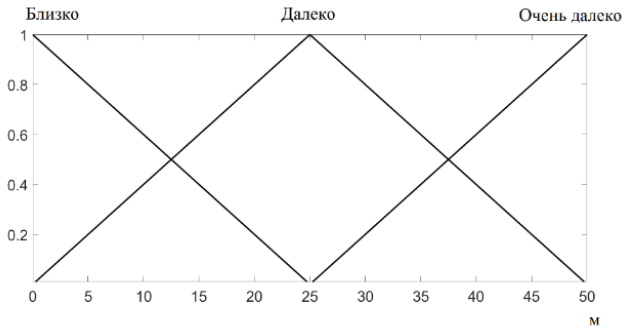


Рис. 7.34. Изображение функций принадлежности для термов лингвистической переменной «расстояние до препятствия»

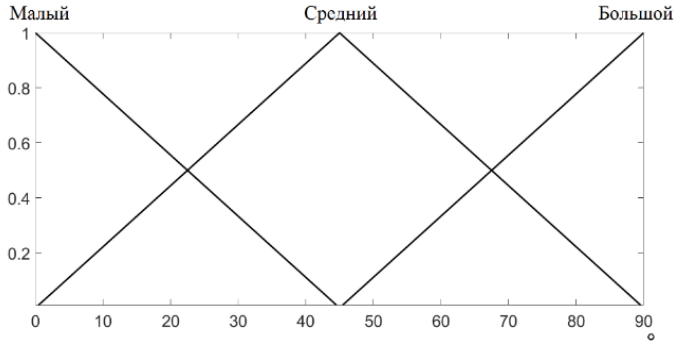


Рис. 7.35. Изображение функций принадлежности для термов лингвистической переменной «угол относительно препятствия»

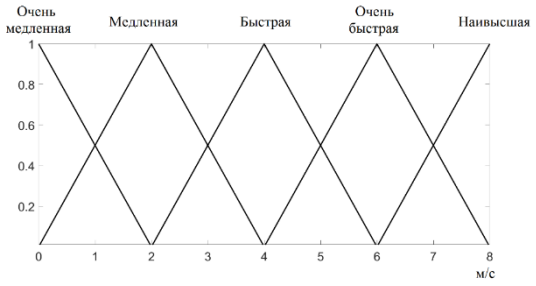


Рис. 7.36. Изображение функций принадлежности для термов лингвистической переменной «скорость»

Результат выполнения нечеткого вывода для случая, когда расстояние до препятствия равно 9 метров, а угол равен 55°, представлен на рис. 7.37. В этом случае на основе нечеткого вывода скорость будет равна 3,38 м/с.

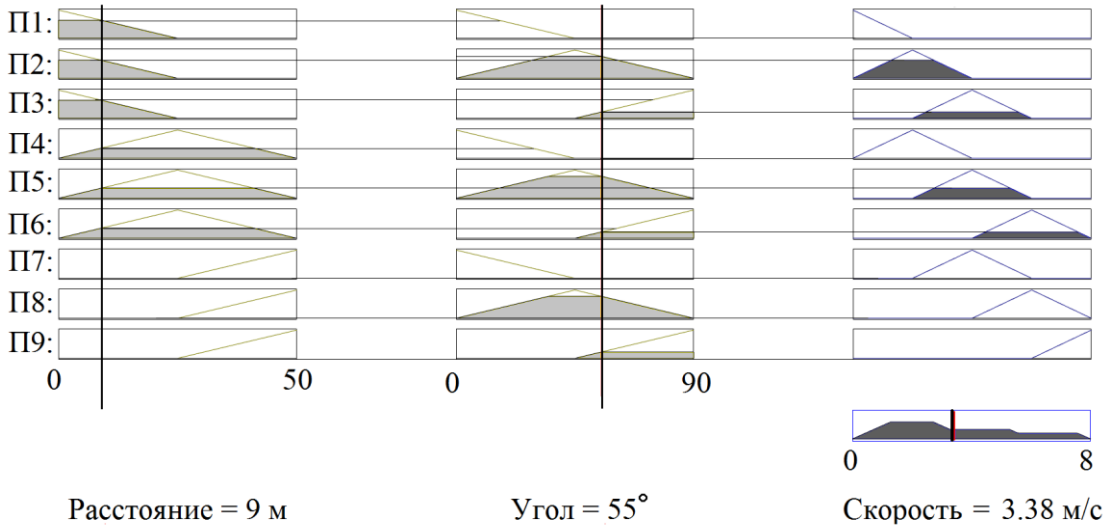


Рис. 7.37. Результат нечеткого вывода по алгоритму Мамдани для системы управления скоростью робота при конкретных значениях входных переменных

Алгоритм Сугено

Алгоритм Такаги и Сугено определяется следующим образом:

1 Нечеткие продукции в базе правил представляются в виде:

ЕСЛИ «β_1 есть α_1» И «β_2 есть α_2» И…И «β_n есть α_n» ТО «$w = f(a_1, a_2, \ldots, a_n)$»,

где a_i – реальное количественное значение входной переменной, подходящей под i-ый терм;

w – действительное значение выходной переменной, определяемое как вещественная функция от входных переменных.

2 Фаззификация входных переменных производится также аналогичным способом с вышеописанным.

3 Агрегирование подусловий производится с помощью min-операции (13) для конъюнкции и max-операции для дизъюнкции (14).

4 Активизация подзаключений осуществляется в форме вычисления одноточечных нечетких множеств, определенных в точке $f(a_1, a_2, \ldots, a_n)$, и функцией принадлежности в виде степени истинности, полученной на этапе агрегирования.

5 Результатом аккумуляции заключений является семейство одноточечных нечетких множеств.

6 Дефаззификация выходных переменных производится в форме (37).

Представим далее пример нечеткого вывода с помощью алгоритма Сугено для ранее показанного вычисления скорости робота. В случае алгоритма Сугено нечеткая продукция базы правил системы будет иметь вид:

«ЕСЛИ «β_1 есть α_i» И «β_2 есть α_j» ТО $v = x_n$,

где v – переменная скорости, принимающая вещественное значение;

$x_n \in [0;8]$ – значение переменной скорости для n-го правила.

Таблица 7.6

База правил Сугено системы нечеткого вывода для управления скоростью робота

	Близко	**Далеко**	**Очень далеко**
Малый	0	2	4
Средний	2	4	6
Большой	4	6	8

Результат выполнения нечеткого вывода для случая, когда расстояние до препятствия равно 9 метров, а угол равен 55°, представлен на рис. 7.38.

В этом случае на основе нечеткого вывода скорость будет равна 3,64 м/с.

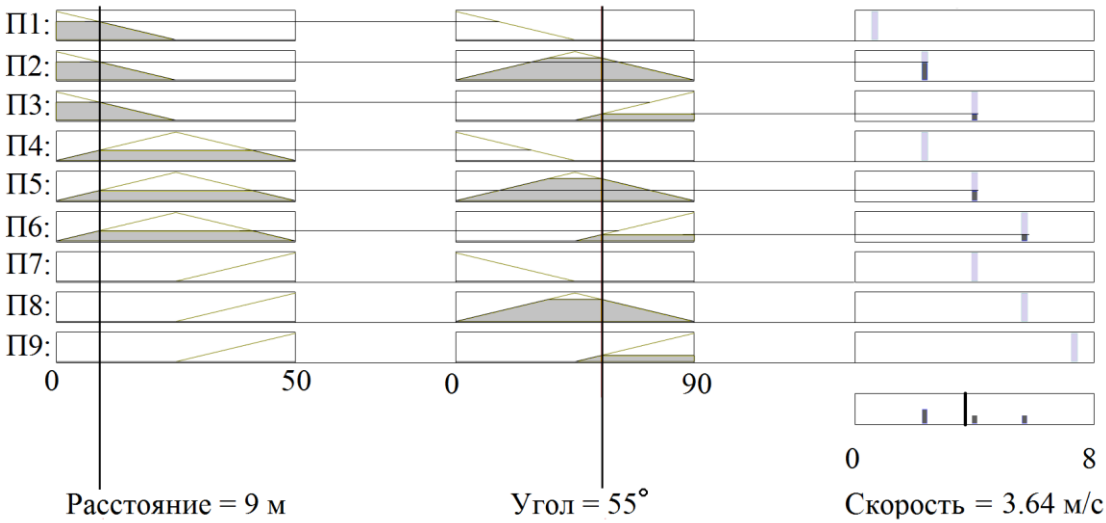


Рис. 7.38. Результат нечеткого вывода по алгоритму Сугено для системы управления скоростью робота при конкретных значениях входных переменных

Таким образом, теория нечетких множеств является одним из наиболее перспективных направлений в области интеллектуального принятия решений в информационных системах, когда присутствует необходимость субъективного вмешательства эксперта. Нечеткие множества и нечеткая логика, в частности, являются особенно полезными, когда повышается сложность моделируемой системы, что затрудняет или даже исключает применение точных количественных методов и подходов. Эти аспекты естественным образом рождают повышающийся с каждым годом интерес в применении нечетких методов управления, охватывающих такие области, как проектирование промышленных роботов и бытовых электроприборов, управление доменными печами и движением поездов метро, автоматическое распознавание речи и изображений.

ЗАКЛЮЧЕНИЕ

В настоящем пособии рассмотрены некоторые базовые концепции и принципы, лежащие в основе интеллектуальных информационных систем. Основная часть книги посвящена прикладным методам искусственного интеллекта. Естественно, представленные методы не являются единственными методами искусственного интеллекта, так как в настоящее время дисциплина искусственного интеллекта – быстро развивающаяся область знаний, в которой с каждым днем появляются все более актуальные задачи и принципиально новые технологии.

Студентам и аспирантам, обучающимся по дисциплинам, связанным с искусственным интеллектом и управлением информационными системами в целом, представленный материал послужит неплохой базой для освоения основ интеллектуальных систем. Данное пособие позволит понять отличительные признаки интеллектуальных систем, позволяющие провести грань между ними и автоматическими системами управления, а также узнать ключевые аспекты понятия «знания» и его связи с понятием «данные».

Для дальнейшего ознакомления с дисциплиной искусственного интеллекта читателям следует ознакомиться с другими теориями искусственного интеллекта, такими как генетические алгоритмы и эволюционные вычисления, роевые методы и биоинспирированные алгоритмы, когнитивное моделирование и др. Также желающим продвинуться в области интеллектуального управления следует более подробно ознакомиться с современными нейросетевыми средствами, такими как глубинное обучение и сверточные нейронные сети, а также изучить фундаментальные работы, основанные на идеях, отличных от предложенных Маккалоком и Питтсом, Хэббом, Румельхартом и др. Здесь имеются ввиду работы таких известных ученых, как Хопфилд, Кохонен и др.

В пособии также не раскрыт вопрос о программировании интеллектуальных систем. Заинтересованные в этой сфере могут изучить материалы по языкам *LISP* и *Prolog*. Кроме того, интересной и одной из наиболее применимых на практике является свободно распространяемая библиотека *OpenCV*, использующая средства искусственного интеллекта для компьютерного зрения.

Авторы выражают благодарность своим коллегам, оказавшим ценные консультации в ходе написания и публикации работы: профессору Сергею Михайловичу Ковалеву, доценту Валерию Борисовичу Тарасову, профессору Александру Витальевичу Боженюку, доценту Майе Викторовне Сухановой.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 **Ясницкий, Л.Н.** Введение в искусственный интеллект / Л.Н. Ясницкий. – М. : Академия, 2008.

2 **Рутковский, Л.** Методы искусственного интеллекта / Л. Рутковский; пер. с польск. И.Д. Рудинского. – М. : Горячая линия-Телеком, 2010. – 520 с.

3 **Рассел, С.** Искусственный интеллект. Современный подход : пер. с англ. / С. Рассел, П. Норвиг. – 2-е изд. – М. : Издательский дом «Вильямс», 2006. – 1408 с.

4 **Мацкевич, В.В.** Занимательная анатомия роботов / В.В. Мацкевич. – 2-е изд., перераб. и доп. – М. : Радио и связь, 1988. – 128 с.

5 **Боженюк, А.В.** Интеллектуальные Интернет-технологии : учебник / А.В. Боженюк, Э.М. Котов, А.А. Целых. – Таганрог : Изд-во ТТИ ЮФУ, 2007. – 376 с.

6 **Осипов, Г.С.** Лекции по искусственному интеллекту / Г.С. Осипов. – М.: Книжный дом «ЛИБРОКОМ», 2014. – 272 с.

7 **Иванов, В.М.** Интеллектуальные системы : учеб. пособие / В.М. Иванов. – Екатеринбург : Изд-во Урал. ун-та, 2015. – 92 с.

8 **Кейслер, Г.** Теория моделей : пер. с англ. / Г. Кейслер, Ч.Ч. Чэн. – М. : Издательство «Мир», 1977. – 614 с.

9 **Гаскаров, Д.В.** Интеллектуальные информационные системы : учебник для вузов / Д.В. Гаскаров. – М. : Высш. шк., 2003. – 431 с.

10 **Рыбина, Г.В.** Основы построения интеллектуальных систем : учеб. пособие / Г.В. Рыбина. – М. : Финансы и статистика; ИНФРА-М, 2010. – 432 с.

11 **Советов, Б.Я.** Интеллектуальные системы и технологии / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. – М. : Издательский центр «Академия», 2013.

12 **Поспелов, Д.А.** Ситуационное управление: теория и практика / Д.А. Поспелов. – М. : Наука, 1986. – 288 с.

13 **Нариньяни, А.С.** Недоопределенные модели и операции с недоопределенными значениями / А.С. Нариньяни. – Препринт ВЦ СО АН СССР, N 400, 1982.

14 **Заде, Л.** Понятие лингвистической переменной и его применение к принятию приближенных решений / Л. Заде. – М. : Мир, 1976.

15 **Александров, Е.А.** Основы теории эвристических решений. Подход к изучению естественного и построению искусственного интеллекта / Е.А. Александров. – М. : Радио и связь, 1975.

16 **Лорьер, Ж.-Л.** Системы искусственного интеллекта / Ж.-Л. Лорьер. – М. : Мир, 1991.

17 **Валькман, Ю.Р.** Интеллектуальные технологии исследовательского проектирования. Формальные системы и семиотические модели / Ю.Р. Валькман. – Киев : Port-Royal, 1998.

18 **Спицын, В.Г.** Представление знаний в информационных системах : учеб. пособие / В.Г. Спицын, Ю.Р. Цой. – Томск : Изд-во Томского политехнического университета, 2008. – 152 с.

19 **Deliyanni, A.** Logic and semantic networks / A. Deliyanni, R.A. Kowalski // Communications of the ACM. – 1979. – Т. 22. – №. 3. – С. 184–192.

20 **Попов, Э.В.** Общение с ЭВМ на естественном языке / Э.В. Попов. – М. : Наука, 1982. – 360 с.

21 **Минский, М.** Структура для представления знания / М. Минский // Психология машинного зрения; под ред. П. Уинстона. – М. : Мир, 1978.

22 **Pawlak, Z.** Rough sets / Z. Pawlak // Proc. On Int. Journal of information and Computer Science. – 1982. – Т. 2. – № 341.

23 **Zhang, Q.** A Survey on Rough Set Theory and Its Applications / Q. Zhang, Q. Xie, G. Wang // CAAI Transactions on Intelligence Technology. – 2016.

24 Rosetta. Инструмент для обработки данных на базе теории приближенных множеств [Электронный ресурс]. – Режим доступа: http://bioinf.icm.uu.se/rosetta/index.php

25 McCulloch, W.S., Pitts, W.A. Logical Calculus of Ideas Immanent in Nervous Activity // Bull. Mathematical Biophysics, 1943. – Т. 5.

26 Воронцов, К.В. Курс лекций: Линейные методы классификации [Электронный ресурс] / К.В. Воронцов. – 2009. – Режим доступа: http://www.machinelearning.ru/wiki/images/6/68/Voron-ML-Lin.pdf.

27 Нейронные сети для начинающих [Электронный ресурс]. –Режим доступа: https://habrahabr.ru/post/313216/

28 **Rummelhart, D.E.** Learning internal representations by error propagation. In McClelland et al / D.E. Rummelhart, G.E. Hilton, R.J. Williams. – 1986.

29 **Дюбуа, Д.** Теория возможностей. Приложения к представлению знаний в информатике : пер. с фр. / Д. Дюбуа, А. Прад. – М.: Радио и связь, 1990. – 288 с.

30 **Леоненков, А.В.** Нечеткое моделирование в среде MATLAB и fuzzyTECH / А.В. Леоненков. – СПб. : БХВ-Петербург, 2014. – 736 с.

Учебное издание

Суханов Андрей Валерьевич
Лященко Зоя Владимировна

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Редактор Т.М. Чеснокова
Техническое редактирование и корректура Т.М. Чесноковой

Подписано в печать 05.07.17. Формат 60×84/16.
Бумага офсетная. Ризография. Усл. печ. л. 7,21.
Тираж . Изд. № 101. Заказ

Редакционно-издательский центр ФГБОУ ВО РГУПС.

Адрес университета: 344038, Ростов н/Д, пл. Ростовского Стрелкового Полка Народного Ополчения, д. 2.

ISBN 978-5-88814-550-0