МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем
Факультет автоматики и вычислительной техники (ФАВТ)
Кафедра электронных вычислительных машин

В. С. РОСТОВЦЕВ

СИСТЕМЫ ОБРАБОТКИ ЗНАНИЙ

Учебное пособие

Киров
2019

УДК 004.891(07)
Р785

Рекомендовано к изданию методическим советом Института математики и информационных систем ВятГУ

Допущено редакционно-издательской комиссией методического совета ВятГУ в качестве учебного пособия для студентов направления 09.03.01 «Информатика и вычислительная техника» по дисциплинам «Системы обработки знаний», «Теория принятия решений» всех профилей подготовки

Рецензенты:
кандидат технических наук, доцент,
заведующий кафедрой автоматики и телемеханики ВятГУ
В. И. Семёновых
кандидат технических наук, доцент, заместитель директора ООО «Литек»
Н. А. Чарушин

Ростовцев, В. С.
Р785 Системы обработки знаний: учебное пособие / В. С. Ростовцев. – Киров: ВятГУ, 2019. – 176 с.

Настоящее учебное пособие по дисциплине «Системы обработки знаний» предназначено для бакалавров 3-4 курсов по направлению 09.03.01 «Информатика и вычислительная техника» всех профилей подготовки и всех форм обучения. Данное учебное пособие охватывает теоретический материал от основных понятий в области инженерных знаний до способов предоставления знаний и принципов построения продукционных, нейросетевых, нечетких и гибридных экспертных систем.

УДК 004.891(07)

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Учебное пособие создано в результате работы автора над лекциями, лабораторными и самостоятельными работами со студентами направления 09.03.01 Информатика и вычислительная техника в ходе преподавания курса «Системы обработки знаний», «Теория принятия решений» в Вятском государственном университете.

Настоящее пособие охватывает основной теоретический материал по курсу «Системы обработки знаний», от основных понятий в области инженерии знаний до способов представления знаний и принципов построения интеллектуальных систем обработки знаний. Каждая глава заканчивается контрольными вопросами для самостоятельной работы студентов.

Первая глава содержит сведения о способах и моделях представления знаний. Рассмотрены модели представления знаний: логические, продукционные, фреймовые, семантические, нейронные, нечеткие и нечеткие когнитивные карты.

Вторая глава посвящена основным понятиям в области инженерии знаний, их истории развития и классификации.

В третьей главе описаны продукционные экспертные системы, основы теории приближенных рассуждений и пример реализации инструментальной экспертной системы.

Вопросы нейрокомпьютинга и применения нейросетевых технологий в системах обработки знаний представлены в четвертой главе, включая современные сверточные нейронные сети и пример обучения по алгоритму обратного распространения ошибки.

Пятая глава содержит краткий обзор основных универсальных пакетов нейросетевых программ (MATLAB, STATISTICA, библиотек языка Pyhton) предметно-ориентированных пакетов (NeuroShell 2, NeuralWorks Professional II/Plus, NeuroSolutions и другие).

В шестой главе содержатся сведения о нечетких системах обработки знаний, приводится модель Мамдани-Заде.

Седьмая глава посвящена генетическим алгоритмам, генетическим операторам, основным понятиям и примерам применения.

Краткие сведения по онтологиям представлены в восьмой главе, в которой представлены примеры использования онтологий для описания крупных проектов и организаций.

В девятой главе содержатся сведения о построении интеллектуальных систем на базе прецедентов, которые нашли свое применение во многих отраслях народного хозяйства.

Учебное пособие способствует лучшему усвоению студентами материалов курса «Системы обработки знаний».

1. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ

1.1. Данные и знания

Данными называют информацию фактического характера, описывающую объекты, процессы и явления предметной области, а также их свойства.

Данные – это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.

В процессе компьютерной обработки данные проходят следующие этапы преобразований:

- исходная форма существования данных (результаты наблюдений и измерений, таблицы, справочники, графики и т.п.);
- представление на специальных языках описания данных, предназначенных для ввода в ЭВМ;
- базы данных на машинных носителях.

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

D1 – данные как результат измерений и наблюдений;

D2 – данные на материальных носителях информации (таблицы, протоколы, справочники);

D3 – модели (структуры) данных в виде диаграмм, графиков, функций;

D4 – данные в компьютере на языке описания данных;

D5 – базы данных на машинных носителях информации.

Знание – это совокупность сведений, образующих целостное описание, соответствующее определенному уровню осведомленности об описываемой проблеме.

Основное отличие знаний от данных в том, что данные описывают лишь конкретное состояние объектов или группы объектов в текущий момент времени, а знания кроме данных содержат сведения о том, как оперировать этими данными.

Знания – это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

Единой принятой терминологии не выработано. Большинство исследователей в области искусственного интеллекта предлагают следующее определение.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

При обработке на ЭВМ знания формируются аналогично данным:

Z1 – знания в памяти человека как результат мышления;

Z2 – материальные носители знаний (учебники, методические пособия);

Z3 – поле знаний – условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

Z4 – знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы и другие модели);

Z5 – база знаний на машинных носителях информации.

В базе знаний ЭС знания должны быть обязательно структурированы и описаны терминами одной из модели знаний. Выбор модели знаний – это наиболее сложный вопрос в проектировании ЭС, так как формальное описание знаний оказывает существенное влияние на конечные характеристики и свойства ЭС.

В рамках одной БЗ все знания должны быть однородные и описаны простыми для понимания терминами. Однородность описания диктуется тем, что в рамках ЭС должна быть разработана единая процедура логического вывода, которая манипулирует знаниями на основе стандартных типовых подходов. Простота понимания определяется необходимостью постоянных контактов с экспертами предметной области, которые не обладают достаточными знаниями в компьютерной технике.

Знания подразделяются, с точки зрения семантики, на факты и эвристики. Факты, как правило, указывают на устоявшиеся в рамках предметной области обстоятельства, а эвристики основываются на интуиции и опыте экспертов предметной области.

По степени обобщенности описания знания подразделяются на поверхностные и глубинные [1, 4].

Поверхностные знания описывают совокупности причинно-следственных отношений между отдельными понятиями предметной области.

Глубинные знания характеризуют абстракции, аналогии, образцы, которые отображают глубину понимания всех процессов, происходящих в предметной области.

Введение в базу глубинных представлений позволяет сделать систему более гибкой и адаптивной, так как глубинные знания являются результатом обобщения разработчиком или экспертом первичных примитивных понятий.

По степени отражения явлений знания подразделяются на «жесткие» и «мягкие». «Жесткие» – позволяют получить однозначные четкие рекомендации при задании начальных условий. «Мягкие» – допускают множественные, расплывчатые решения и многовариантные рекомендации [9].

Мягкими называют знания, которые используют для представления теорию нечетких множеств, нейронных сетей, вероятностных и приближенных рассуждений, генетических алгоритмов.

Знания являются более сложной категорией информации по сравнению с данными. Знания описывают не только отдельные факты, но и взаимосвязи между ними. Поэтому знания называют структурированными данными. Они могут быть получены на основе обработки эмпирических данных. Они представляют результат мыслительной деятельности человека, направленной на обобщение опыта. Для того чтобы ввести знания в базу знаний любой интеллектуальной системы, их необходимо представить в определенной форме.

Знания в экспертных системах существуют в следующих формах:

– исходные знания (правила, выведенные на основе практического опыта, математические и эмпирические зависимости, отражающие взаимосвязи между фактами, описывающие изменение фактов с течением времени, графы и т. п.);

– описание исходных знаний средствами выбранной модели представления знаний (множество логических формул, продукционных правил, фреймов, семантических и нейронных сетей);

– представление знаний структурами данных;

– базы знаний на машинных носителях.

1.2. Классификация знаний

Существует множество классификаций знаний[1]. По своей природе знания подразделяются на декларативные и процедурные.

Декларативные знания представляют собой описания фактов и явлений, фиксируют наличие или отсутствие таких фактов.

Процедурные знания – это описания действий, которые возможны для манипулирования фактами и явлениями при достижении поставленных целей. К ним относятся методики, правила и т.п.

Существует множество способов определять понятия [1, 4]. Один из широко применяемых способов основан на идее интенсионала. *Интенсионал* понятия – это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств. Интенсионалы формулируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к определяемому объекту. Это определение через данные, или *экстенсионал понятия*.

Пример 1. Понятие «персональный компьютер». Его интенсионал: персональный компьютер – это дружественная и относительно недорогая ЭВМ, которую можно разместить на столе.

Экстенсионал этого понятия: «Персональный компьютер – это IBM PC». Для хранения данных используются базы данных (для них характерны большой объем и относительно небольшая удельная стоимость информации), для хранения знаний – базы знаний (небольшого объема, но исключительно дорогие информационные массивы).

Знания могут быть классифицированы по следующим категориям:

– *поверхностные* – знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области;

---

[1] «Когнитивные технологии для поддержки принятия управленческих решений» http://www.iis.ru/events/19981130/maximov.ru.html

– *глубинные* – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Пример 2. Поверхностные знания: «Если нажать на кнопку звонка, раздастся звук». «Если болит голова, то следует принять аспирин». Глубинные знания: «Принципиальная электрическая схема звонка и проводки». «Знания физиологов и врачей высокой квалификации о причинах, видах головных болей и методах их лечения».

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Исторически первичными были процедурные знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы.

Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все большая часть знаний сосредоточилась в структурах, данных (таблицы, списки, абстрактные типы данных).

В результате повысилась роль декларативных знаний.

Сегодня знания приобрели чисто декларативную форму, т. е. знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному языку и понятных специалистам.

По способу приобретения знания можно разделить на факты и эвристические правила, которые позволяют сделать выбор при отсутствии точных теоретических обоснований.

1.3. Логическая модель представления знаний

Распространенными моделями представления знаний в ЭС являются: логическая, продукционная, фреймовая, семантическая, нейронная.

Существуют множество моделей (или языков) представления знаний для предметных различных областей.

Логическая модель представления знаний основана на системе исчисления предикатов первого порядка. Предикатом называется некоторая связь, которая задана на наборе констант и переменных. Применение предикатов всеобщности, существования и операций «И», «ИЛИ», «НЕ», «Импликация», «Эквивалентность» позволяют описать многие знания в предметной области. Недостатками логической модели, основанной на системе исчисления предикатов первого порядка, являются следующие [1, 3, 4]:

– невозможность выразить через переменные другие предикаты;

– сложность логического вывода при больших массивах данных.

В основе языка предикатов первого порядка лежит понятие предикатов, т. е. логическая функция от одной или нескольких нелогических переменных. Функция может принимать значения истина (t) или ложь (f). В рамках логики

утверждение считается истинным, если и относящееся к нему предположение считается истинным и заключение самого утверждения тоже истина.

Синтаксис языка предикатов включает: предикативные символы, символы переменных, константы, а также разделители ( ), [ ], “, ‘.

Предикативные символы используются для обозначения отношений. Объекты отношений записываются в круглые скобки после предикативного имени и называются аргументами. Полная запись отношения называется атомарной формулой. Например, атомарная формула «Является (Иванов, специалист по компьютерам)» содержит два предикативных терма.

Термы могут быть константами и переменными. Разрешено также в качестве термов использовать функции, которые обязательно должны быть определены в рамках предметной области. Разработчик ЭС заранее определяет, как интерпретировать порядок термов в отношении.

Допустимые выражения в исчислении предикатов, в частности атомарные формулы, называются правильно построенными функциями (ППФ). В языке предикатов для каждой ППФ обязательно определяется конкретная интерпретация. Как только для ППФ определена интерпретация, то говорят, что формула имеет значение «истина», если соответствующее утверждение истинно, в противном случае ППФ имеет значение «ложь».

Из формул можно составить предложение с помощью логических связок: конъюнкция, дизъюнкция, импликация, отрицание.

Конъюнкция обозначается символом «&» и реализует функцию «И»:

Учится (Иванов, Московский государственный университет) & располагается (университет, Москва).

Дизъюнкция обозначается символом «V» (ИЛИ) и реализует функцию не исключающего «ИЛИ». Находятся (Иванов, аудитория 113) V Находится (Иванов, библиотека).

Импликация обозначается символом «→» и используется для представления утверждения типа «ЕСЛИ, ТО». Владеть (Иванов, автомобиль) → марка (автомобиль, “Opel”).

Левая сторона импликации называется антецедент, правая – консеквент. Импликация имеет значение «ложь» только в одном случае, если антецедент имеет значение «истина», а консеквент имеет значение «ложь».

ППФ со знаком отрицания «~» перед ней называется отрицанием.

В языке предикатов атомарная формула может принимать только истинные значения или только ложные значения в зависимости от значений переменных, которые в нее входят. Для того чтобы при исчислении предикатов можно было манипулировать значениями переменных, потребовалось ввести понятие «квантор».

Квантор – это операция, в которой участвуют все значения переменной одного предиката. Квантор служит для указания меры, в какой экземпляры переменной, т. е. константы должны быть истинными, чтобы все значения в целом были истинными.

Различают квантор общности ∀(x) и квантор существования. Если перед предикатом записан квантор ∀x для какой-то переменной, например, ∀(x), то

это означает, что значение предиката будет истинным только в том случае, если все значения переменной x будут истинными.

∀(x) (специалист_по_ЭВМ(x) → программист(x))

Например, квантор существования ∃(x) означает, что для истинности предиката достаточно, чтобы только некоторые значения переменной или одно значение были истинными.

∃(x) (специалист_по_ЭВМ(x) & оптимист(x))

В рамках одного предиката можно использовать и кванторы общности, и кванторы существования, но для разных переменных.

Машинная реализация языка предиката первого порядка имеет ряд серьезных проблем, которые связаны с универсальностью аппарата логического вывода.

Первая проблема – *монотонность рассуждений*. В процессе логического вывода нельзя отказаться от промежуточного заключения, если становятся известными дополнительные факты, которые свидетельствуют о том, что полученные на основе этого заключения решения не приводят к желаемому результату.

Вторая проблема – *комбинаторный взрыв*. В процессе логического вывода невозможно применять оценочные критерии для выбора очередного правила. Бессистемное применение правил в расчете на случайное доказательство приводит к тому, что возникает много лишних цепочек ППФ, активных в определенный момент времени. Это чаще всего приводит к переполнению рабочей памяти.

Наиболее эффективной разработкой этого подхода является язык PROLOG. В нем принята обратная стратегия вывода, полностью реализованы все средства описания знаний с помощью предикатов. Для порождения новых высказываний используется операция резолюции. В качестве процедуры поиска решения, позволяющей устранить монотонность и комбинаторный взрыв, используют поиск в иерархически упорядоченном пространстве состояний.

К достоинствам логической модели можно отнести наличие теоретического материала по методам логического вывода и наличие стандартной типовой процедуры логического вывода (доказательства теорем).

Недостатками логической модели являются: сложность использования эвристик в процессе логического вывода, монотонность логического вывода, возможность комбинаторного взрыва, слабая структурированность описаний.

1.4. Продукционная модель представления знаний

Продукционная модель или модель, основанная на правилах, позволяет представить знания в виде предложений типа «ЕСЛИ (условие), ТО (действие)».

Под «условием» (антецедентом) понимается некоторое предложение – образец, по которому осуществляется поиск в базе знаний, а под «действием» (консеквентом) – действия, выполняемые при успешном исходе поиска. Они могут быть промежуточными, выступающими далее, как условия, и терминальными или целевыми, завершающими работу системы.

Чаще всего вывод на такой базе знаний бывает *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения к данным). Существуют также системы с двунаправленными выводами. Данные – это исходные факты, хранящиеся в базе фактов, на основании которых запускается машина вывода или интерпретатор правил, выбирающий правила из продукционной базы знаний [1, 3, 4, 11].

Продукционная модель чаще всего применяется в промышленных экспертных системах. Она привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

Основные достоинствами продукционных систем связаны с простотой представления знаний и организации логического вывода. К недостаткам продукционных систем можно отнести следующее:

- отличие от структур знаний, свойственных человеку;
- неясность взаимных отношений правил;
- сложность оценки целостного образа знаний;
- низкая эффективность обработки знаний;
- возникновение конфликтных ситуаций и необходимость их разрешения;
- при большом количестве правил возможность появления противоречивых правил требует разработки специальных механизмов контроля вводимых и редактируемых правил.

Имеется большое число программных средств, реализующих продукционный подход (язык OPS 5; «оболочки» или «пустые» ЭС – EXSYS Professional, Kappa, ЭКСПЕРТ, ЭКО, инструментальные системы ПИЭС), а также промышленных ЭС на его основе (например, ЭС, созданных средствами G2 и др.).

1.5. Фреймовая модель представления знаний

Фреймовая модель основана на теории Минского и представляет систематизированную психологическую модель памяти человека и его сознания. Термин *фрейм* происходит от английского слова frame, которое означает «каркас» или «рамка». Он был предложен Марвином Минским (одним из пионеров искусственного интеллекта) в 1979 году.

Фрейм – это абстрактный образ для представления некоего восприятия. В психологии известно понятие абстрактного образа. В искусственном интеллекте фреймом называют структуру данных для представления некоторого концептуального объекта. Из понятия фрейма ничего нельзя выбросить. Например, фрейм комнаты. Из описания комнаты нельзя ничего выбросить. Если удалить окна, то образ комнаты превращается в образ чулана [1, 3, 4].

(ИМЯ ФРЕЙМА:
(имя 1-го слота: значение 1-го слота),
(имя 2-го слота: значение 2-го слота),
…
(имя N-го слота: значение N-го слота)).

Структура фрейма может быть представлена в виде табл. 1.1.

Таблица 1.1

Структура фрейма

Имя фрейма			
Имя слота	Значение слота	Способ получения значения	Присоединенная процедура
Рост	175		

В данной таблице дополнительные столбцы предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур. Например, слот ВОЗРАСТ может содержать имя процедуры, которая вычисляет возраст человека по дате рождения, записанной в другом слоте, и текущей дате.

Процедуры, располагающиеся в слотах, называются связанными или присоединенными процедурами. Вызов связанной процедуры осуществляется при обращении к слоту, в котором она помещена. Заполнителями слота могут быть правила продукций, используемые для определения конкретного значения.

В слоте может содержаться не одно, а несколько значений, например, массивы, списки, фреймы и т.п. Так, в слоте БРАТ может содержаться список имен, если объект, описываемый данным фреймом, имеет нескольких братьев.

Значение слота может содержать перечень возможных значений, арифметическое выражение, фрагмент текста и т. д.

Пример фрейма РУКОВОДИТЕЛЬ

Имя слота	**Значение слота**	**Тип значения слота**
Имя	Иванов И. И.	Строка символов
Рождение	01.01.1965	Дата
Возраст	Age (дата, рождение)	Процедура
Специальность	Юрист	Строка символов
Адрес	Домашний адрес	фрейм

Совокупность данных предметной области может быть представлена множеством взаимосвязанных фреймов, образующих единую фреймовую систему, в которой объединяются декларативные и процедурные знания. Такая система имеет иерархическую структуру, в которой фреймы соединены друг с другом родовидовыми связями. На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов. Например, фрейм УССУРИЙСКИЙ ТИГР наследует от фрейма ТИГР значение слота цвет – полосатый. Над фреймами можно совершать такие операции, как объединение и пересечение. При объединении фреймов в результирующем фрейме будут присутствовать все слоты, которые встречались в исходных фреймах. При пересечении фреймов в результирующем фрейме будут присутствовать только те слоты, которые имелись во всех исходных фреймах.

Фреймовые системы подразделяются на статические и динамические. Динамические допускают изменения фреймов в процессе решения задачи.

Имя фрейма служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным. Одни слоты являются системными и служат для выполнения специфических функций, например, слот-указатель родителя данного фрейма (IS-A), слот-указатель дочерних фреймов, слот для ввода имени пользователя, слот для ввода даты определения фрейма, слот для ввода даты изменения фрейма и т.п.

Имя слота. Оно должно быть уникальным в пределах фрейма. В качестве имени слота может выступать произвольный текст. Например, ИМЯ СЛОТА = Первый космонавт, а ЗНАЧЕНИЕ СЛОТА = Гагарин.

Имена системных слотов обычно зарезервированы, например, IS-A, HASSPART и т.п. Системные слоты служат для редактирования базы знаний и управления выводом во фреймовой системе.

Указатели наследования. Они показывают, какую информацию об атрибутах слотов фрейма верхнего уровня наследуют слоты с аналогичными именами в данном фрейме. В конкретных системах указатели наследования могут быть организованы различными способами:

U (Unique) – значение слота не наследуется;

S (Same) – значение слота наследуется;

R (Range) – значение слота должны находится в пределах интервала значений, указанных в одноименном слоте родительского фрейма;

O (Override) – при отсутствии значений в текущем слоте оно наследуется из фрейма верхнего уровня. Однако, в случае определения значения текущего слота оно может быть уникальным. Этот тип указателя выполняет одновременно функции указателя U и S.

Указатель типа данных. Он показывает тип значения слота:

Frame – указатель на фрейм;

Real – вещественное число;

Integer – целое число;

Boolean – логический тип;

Text – фрагмент текста;

List – список;

Table – таблица;

Expression – выражение;

Lisp – связанная процедура и т. п.

Значение слота. Оно должно соответствовать типу данных и условию наследования.

Демоны. Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия. Демоны автоматически запускаются при обращении к соответствующему слоту. Типы демонов связаны с условием запуска процедуры.

Демон IF-NEEDED запускается, если в момент обращения к слоту его значение не было установлено.

Демон IF-ADDED запускается при попытке изменения значения слота.

Демон IF-REMOVED запускается при попытке удаления значения слота.

Присоединенная процедура. В качестве значения слота может использоваться процедура, называемая служебной в языке ЛИСП или методом в языках объектно-ориентированного программирования. Она запускается по сообщению, переданному из другого фрейма. Демоны и присоединенные процедуры являются процедурными знаниями, объединенными вместе с декларативными знаниями в единую систему.

Во фреймах наследование происходит по связям АКО (IS-A). Слот АКО (IS-A) указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются значения аналогичных слотов. Например, в сети фреймов рис. 1.1 понятие «ученик» наследует свойства «ребенок» и «человек», которые находятся на более высоком уровне иерархии.

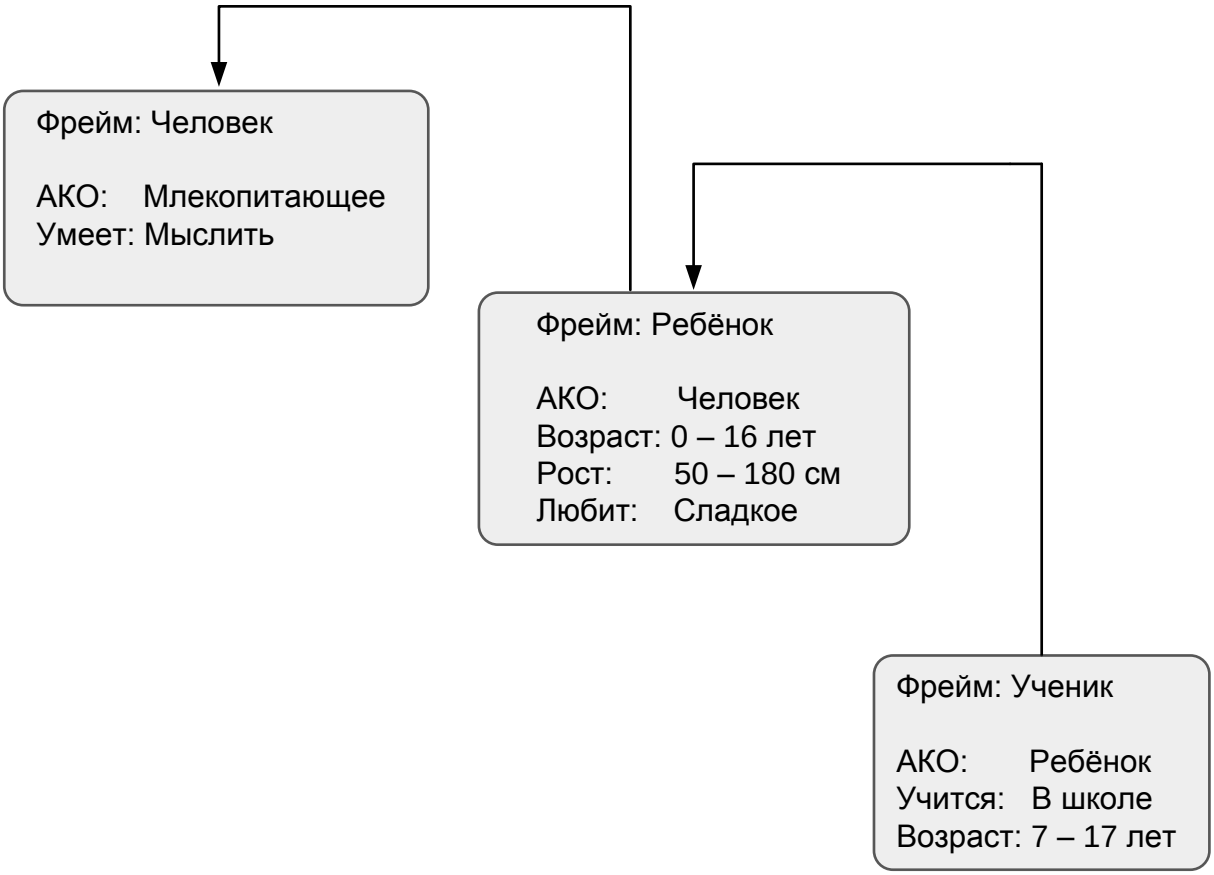


Рис. 1.1. Пример сети фреймов

Существует несколько способов получения слотом значений во фрейме-экземпляре:

- по умолчанию от фрейма-образца;
- через наследование свойств от фрейма, указанного в слоте АКО;
- по формуле, указанной в слоте;
- через присоединенную процедуру;
- через диалог с пользователем;
- из базы данных.

На вопрос «любит ли ученик сладкое» следует ответ «да», так как этим свойством обладают все дети, что указано во фрейме «ребенок». Наследова-

ние свойств может быть частичным, так как возраст для учеников не наследуется из фрейма «ребенок», поскольку указан явно в своем собственном фрейме.

В качестве значения слота может быть имя другого фрейма, так образуется сеть фреймов.

Различают фреймы-образцы (фреймы-прототипы), хранящиеся в базе знаний и фреймы-экземпляры (фреймы-примеры), которые создаются для отображения реальных ситуаций на основе поступающих данных.

Можно выделить различные типы моделей фреймов:

– фреймы-структуры, применяемые для отображения объектов и понятий (заем, вексель);
– фреймы-роли (менеджер, кассир, клиент);
– фреймы-сценарии (банкротство, собрание акционеров и т. п.);
– фреймы-ситуации (тревога, авария, рабочий режим устройства и т. п.).

Основное преимущество модели фреймов состоит в том, что такая модель отражает концептуальную основу организации памяти человека и является наглядной и гибкой. Разработаны специальные языки представления знаний в сетях фреймов. Это FRL (Frame representation Language), KRL (Knowledge Representation Language, оболочка Карра и другие [1]. Известны фрейм-ориентированные экспертные системы ANALYST, МОДИС, TRISTAN [1, 3, 4].

1.6. Семантическая модель представления знаний

Термин *семантическая* означает «смысловая». Семантика – это наука, устанавливающая отношения между символами и объектами, которые они обозначают, т.е. наука, определяющая смысл знаков [1, 3, 4].

Семантическая сеть – это ориентированный граф, вершины которого – понятия, а дуги – отношения между ними.

В качестве понятий обычно выступают абстрактные или конкретные объекты, а *отношения* – это связи типа: «это» («AKO – A-Kind-Of», «is»), «имеет частью», «принадлежит», «любит». Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

– класс – элемент класса (цветок – роза);
– свойство – значение (цветок – желтый);
– пример элемента класса (роза – чайная).

Семантическая сеть называется однородной, если содержит только один тип отношений, и неоднородной – в противном случае.

По типам отношений семантическая сеть подразделяется на бинарную (связывающую два объекта) и N-арную (для связи нескольких объектов).

Наиболее часто в семантических сетях используются следующие отношения:

– связи типа «часть-целое», «класс-подкласс», «элемент-множество»;
– функциональные связи, определяемые глаголами «производит», «влияет»;
– количественные (больше, меньше, равно);

– пространственные (далеко от, близко от, над, под и т.п.);
– временные (раньше, позже и т.п.);
– атрибутивные (иметь свойство, иметь значение);
– логические связи (И, ИЛИ, НЕ) и др.

Поиск решения в базе знаний, представленной в виде семантической сети, сводится к задаче поиска подграфа. В семантической сети (рис. 1.2) в качестве вершин выступают понятия «человек», «Иванов», «ВАЗ», «автомобиль», «вид транспорта», «двигатель».

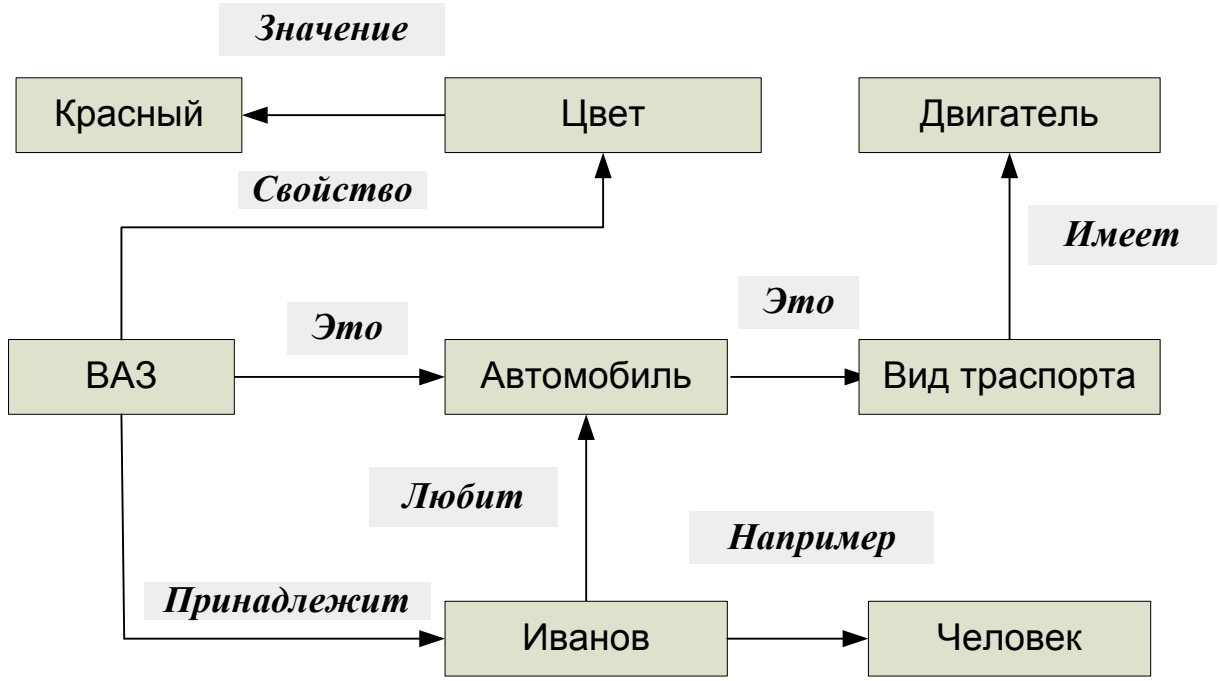


Рис. 1.2. Пример семантической сети

Данная модель была предложена американским психологом Куиллианом. Семантическая сеть соответствует современным представлениям об организации долговременной памяти человека. Здесь, как и во фреймах, декларативные и процедурные знания не разделены. К недостаткам следует отнести сложность поиска подграфа и вывода на семантической сети. Для реализации семантических сетей разработаны специальные языки (NET, SIMER+MIR и др.). Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний: PROSPECTOR, CASNET, TORUS [1, 3, 4].

1.7. Нейронная модель представления знаний

В последние годы бурно развиваются *нейронные экспертные системы*. В основе их построения лежит принцип обучения нейронной сети на известных примерах с последующим тестированием по любому входному вектору [2, 5].

Нейронные сети могут выступать в качестве модели представления знаний. В первую очередь это обучающая выборка, которая представляет неявную базу знаний (до обучения нейросетевой экспертной системы).

Во-вторых, это синаптическая карта, сформированная по результатам выбора оптимальной архитектуры и обучения нейронной сети. Однако при дополнении обучающей выборки требуется дообучение или переобучение нейронной сети.

Дообучение сети не требует изменения архитектуры нейронной сети, а переобучение может привести к изменению архитектуры нейронной сети.

Достоинства: не требуется разрабатывать множество продукционных правил и реализовывать механизм разрешения конфликтов. Обучающее множество можно оптимизировать по количеству примеров.

Недостатки: отсутствует методика выбора оптимальной архитектуры нейронной сети, и методика определения оптимального размера обучающей выборки.

1.8. Представление знаний в виде нечетких правил

Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия были впервые предложены американским ученым Лотфи Заде (Lotfi Zadeh) в 1965 году. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов [6, 7]. Следует использовать общепринятую терминологию.

Нечеткое множество (fuzzy sets) – множество, элементы которого принадлежат ему в той или иной степени.

Нечеткая логика (fuzzy logic) – умозаключение с использованием нечетких множеств или нечетких правил.

Нечеткое правило (fuzzy rule) – условное высказывание вида «ЕСЛИ X есть A, ТО У есть B», где A и B нечеткие множества. Нечеткое правило образует связь между нечеткими множествами.

Нечеткая система (fuzzy system) – множество нечетких првил, преобразующих входные данные в выходные.

В нечеткой логике основополагающим является понятие *лингвистической переменной,* значениями которой являются не числа, а слова естественного языка, называемые *термами.*

Характеристикой нечеткого множества выступает функция принадлежности (Membership Function). MFc(x) – степень принадлежности значения «x» к нечеткому множеству C, представляющая собой обобщение понятия характеристической функции обычного множества. Тогда нечетким множеством C называется множество упорядоченных пар вида C={MFc(x)/x}, MFc(x) принадлежит интервалу от 0 до 1. Значение MFc(x)=0 означает отсутствие принадлежности к множеству, 1 – полную принадлежность.

Задание лингвистической переменной можно осуществить в дискретной или непрерывной форме.

Дискретная форма:

A(x)={M(x1)/x1, M(x2)/x2, …, M(xn)/xn}.

Непрерывная форма в виде функциональной зависимости.

$M_A(x)=\exp(-(x-3)/0,2)^2$.

Пример. Для множества чисел от x1=7, x2=8, …, x7=13 степень принадлежности их к числам, близким к 10, выглядит таким образом:

A(x)={0,1/7; 0.3/8; 0.8/9; 1.0/10; 0.8/11; 0.3/12; 0.1/13}.

Примером нечеткого множества может служить формализация неточного определения ГОРЯЧИЙ ЧАЙ. В качестве x (область рассуждений) будет выступать шкала температуры в градусах Цельсия. Очевидно, что она будет изменяется от 0 до 100 градусов. Нечеткое множество для понятия ГОРЯЧИЙ ЧАЙ может выглядеть следующим образом:

C ={0/0; 0/10; 0/20; 0,15/30; 0,30/40; 0,60/50; 0,80/60; 0,90/70; 1/80; 1/90; 1/100}.

Например, чай с температурой 60 градусов Цельсия, принадлежит к множеству ГОРЯЧИЙ ЧАЙ со степенью принадлежности 0,80. Для одного человека чай с температурой 60 градусов Цельсия может оказаться горячим, для другого – не слишком горячим. Именно в этом и проявляется нечеткость задания соответствующего множества.

Нечеткая переменная описывается набором (N, X, A), где N – это название переменной, X – универсальное множество (область рассуждений), A – нечеткое множество на X.

Значениями лингвистической переменной могут быть нечеткие переменные, т.е. лингвистическая переменная находится на более высоком уровне, чем нечеткая переменная.

Каждая *лингвистическая переменная* состоит из следующих компонентов:

- наименование переменной;
- множество своих значений, которое также называется базовым терм-множеством T. Элементы базового терм-множества представляют собой названия нечетких переменных;
- универсальное множество X;
- синтаксические правила G, по которым генерируются новые термы с применением слов естественного или формального языка;
- семантические правила P, которые каждому значению лингвистической переменной ставят в соответствие нечеткое подмножество множества X.

Например, *нечеткое понятие* как ЦЕНА АКЦИИ представляет наименование лингвистической переменной. Для нее базовое терм-множество будет состоять из трех нечетких переменных: НИЗКАЯ, УМЕРЕННАЯ, ВЫСОКАЯ. Область рассуждений определяется в виде X=[100; 200] единиц.

Для каждого лингвистического терма из базового терм-множества T должна быть построена функция принадлежности. Количество термов в лингвистической переменной обычно не превышает семи.

Например, в случае управления мобильным роботом можно ввести две лингвистические переменные: ДИСТАНЦИЯ (расстояние до помехи) и НАПРАВЛЕНИЕ (угол между продольной осью робота и направлением на помеху).

Рассмотрим *лингвистическую переменную* ДИСТАНЦИЯ. Значениями ее можно определить термы ДАЛЕКО, СРЕДНЯЯ, БЛИЗКО и ОЧЕНЬ БЛИЗКО. Для физической реализации лингвистической переменной необходимо определить точные физические значения термов этой переменной. Пусть переменная ДИСТАНЦИЯ может принимать любое значение из диапазона от нуля до бесконечности. Согласно положениям теории нечетких множеств, каждому значению расстояния из указанного диапазона может быть поставлено в соответствие некоторое число от нуля до единицы, которое определяет степень принадлежности данного физического расстояния (допустим 40 см) к тому или иному терму лингвистической переменной ДИСТАНЦИЯ.

Степень принадлежности определяется так называемой функцией принадлежности M(d), где d – расстояние до помехи. В нашем примере расстоянию 40 см можно задать степень принадлежности к терму ОЧЕНЬ БЛИЗКО, равную 0,7, а к терму БЛИЗКО – 0,3 (рис.1.3). Конкретное определение степени принадлежности может проходить только при работе с экспертами.

Переменной НАПРАВЛЕНИЕ, которая может принимать значения в диапазоне от 0 до 360 градусов, зададим термы ЛЕВОЕ, ПРЯМО И ПРАВОЕ.

Теперь необходимо задать выходные переменные. В рассматриваемом примере достаточно одной, которая будет называться РУЛЕВОЙ УГОЛ. Она может содержать термы: РЕЗКО ВЛЕВО, ВЛЕВО, ПРЯМО, ВПРАВО, РЕЗКО ВПРАВО. Связь между входом и выходом запоминается в таблице нечетких правил (рис.1.4).

Каждая запись, представленная на рис.1.4, соответствует своему нечеткому правилу: «Если ДИСТАНЦИЯ БЛИЗКО и НАПРАВЛЕНИЕ ПРАВОЕ, тогда РУЛЕВОЙ УГОЛ РЕЗКО ВЛЕВО».

Таким образом, мобильный робот с нечеткой логикой будет работать по следующему принципу: данные с сенсоров о расстоянии до помехи и направлении на нее будут фаззифицированы, обработаны согласно табличным правилам, дефаззифицированы, и полученные данные в виде управляющих сигналов поступят на привод робота.

Все системы с нечеткой логикой функционируют по одному принципу: показания измерительных приборов фаззифицируются (переводятся в нечеткий формат), обрабатываются, дефаззифицируются и в виде привычных сигналов подаются на исполнительные устройства.

Фаззификация – это сопоставление множества значений x ее функции принадлежности M(x), т.е. перевод значений x в нечеткий формат (пример с термином молодой). Дефаззификация представляет собой процесс, обратный фаззификации [6,7].

Лингвистическая переменная «ДИСТАНЦИЯ» включает 4 терма
«ДИСТАНЦИЯ»={очень близко, близко, средняя, далеко}

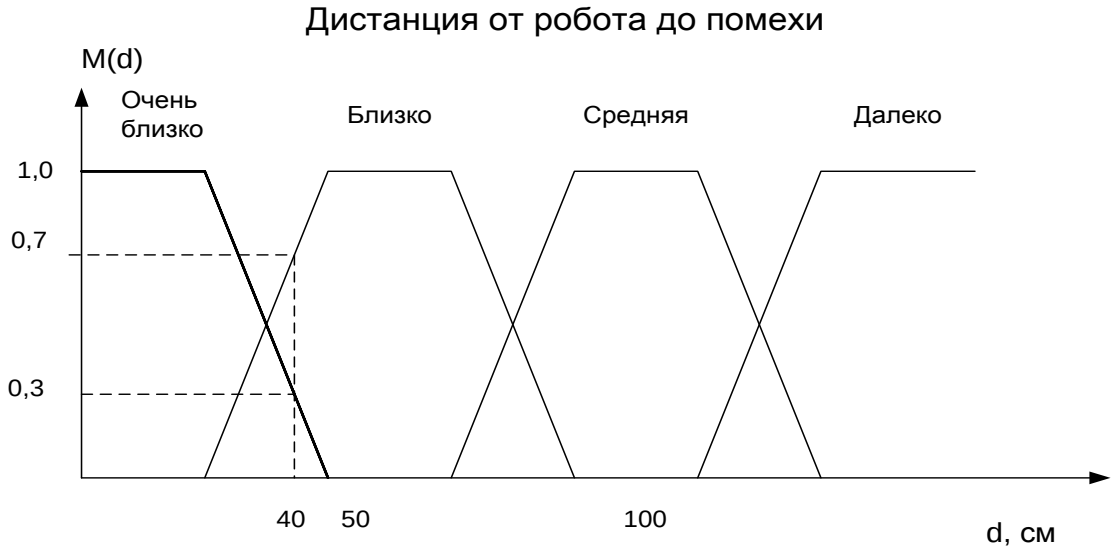


Рис.1.3. Лингвистическая переменная «ДИСТАНЦИЯ»

Пример нечеткого правила «ЕСЛИ дистанция БЛИЗКО
И направление ПРАВОЕ, ТО рулевой угол РЕЗКО ВЛЕВО»

Направление	**Дистанция**			
	Очень близко	*Близко*	Средняя	Далеко
Правое	Резко влево	*Резко влево*	Влево	Прямо
Прямо	Резко влево	Влево	Влево	Прямо
Левое	Резко вправо	Резко вправо	Вправо	Прямо

Рис. 1.4. Пример базы нечетких правил

В нечеткой логике есть возможность строить выражения, на основе которых можно получать новые значения нечетких (следовательно, и лингвистических) переменных.

Применение правил нечеткого вывода позволяет получить описание предметной области на основе имеющихся у эксперта знаниях о ней. Благодаря операциям фаззификации и дефаззификации имеется возможность обработки данных, общая схема которой представляется так:

– по имеющемуся набору данных, с помощью оператора фаззификации провести перевод этих данных в нечеткий формат;

– пользуясь имеющейся базой нечетких правил, преобразовать нечеткие переменные;

– получившийся набор нечетких переменных подвергнуть дефаззификации и получить четкое значение.

Часть выходных данных можно передать через обратную связь на вход, если того требует алгоритм управления.

Проведение таких преобразований называется нечетким логическим выводом. Для реализации ЭС на базе нечетких правил разработано множество алгоритмов нечеткого вывода. Например, правила нечеткого вывода заданы следующим образом:

П1: если x есть A, то w есть D,
П2: если y есть B, то w есть E, (1.1)
П3: если z есть C, то w есть F,

где x, y, z – имена входных переменных (четкого формата);
w – имя переменной вывода;
A, B, C, D, E, F – заданные функции принадлежности.
Иллюстрация к алгоритму нечеткого вывода представлена на рис.1.5.
Пример реализации алгоритма Мамдани с правилами П1 и П2:

П1: если x есть A1 и y есть B1, то z есть C1,
П2: если x есть A2 и y есть B2, то z есть C2, (1.2)

где x, y – имена входных переменных (четкого формата);
z – имя переменной вывода;
A1, B1, C1, A2, B2, C2 – заданные функции принадлежности.
Далее следует этап, называемый «введение нечеткости».
Находятся степени истинности для предпосылок каждого правила:

$$A_1(x_0), A_2(x_0), B_1(y_0), B_2(y_0), \quad (1.3)$$

где x_0, y_0 – имена входных переменных (четкого формата).
Находятся уровни отсечения для предпосылок каждого из правил.

$$a_1 = A_1(x_0) \wedge B_1(y_0)$$
$$a_2 = A_2(x_0) \wedge B_2(y_0), \quad (1.4)$$

где a1, a2 – уровни отсечения;
$\wedge$ – оператор минимума.
Производится объединение усеченных множеств

$$\mu_{\Sigma}(z) = (a_1 \wedge C_1(z)) \vee (a_2 \wedge C_2(z)),$$

где C(z) – функция принадлежности для элемента z. (1.5)

Для нахождения значения z_0 необходимо провести дефаззификацию, например, центроидным методом [26].

Дефаззификация – приведение к четкости. Существует несколько популярных методов [3,9,10], некоторые из которых приведены ниже.

Центроидный метод. Для дискретного варианта представления множеств:

$$Z_0 = \frac{\sum_{i=1}^{n} \mu_i z_i}{\sum_{i=1}^{n} \mu_i}. \quad (1.6)$$

где z – элемент эталонного множества;

z_i – элемент эталонного множества;

μ_i – функция принадлежности для элемента z_i;

Z_0 – четкое значение.

Метод первого максимума (first-of-maxima). Четкая величина находится как наименьшее значение, при котором достигается максимум функции принадлежности.

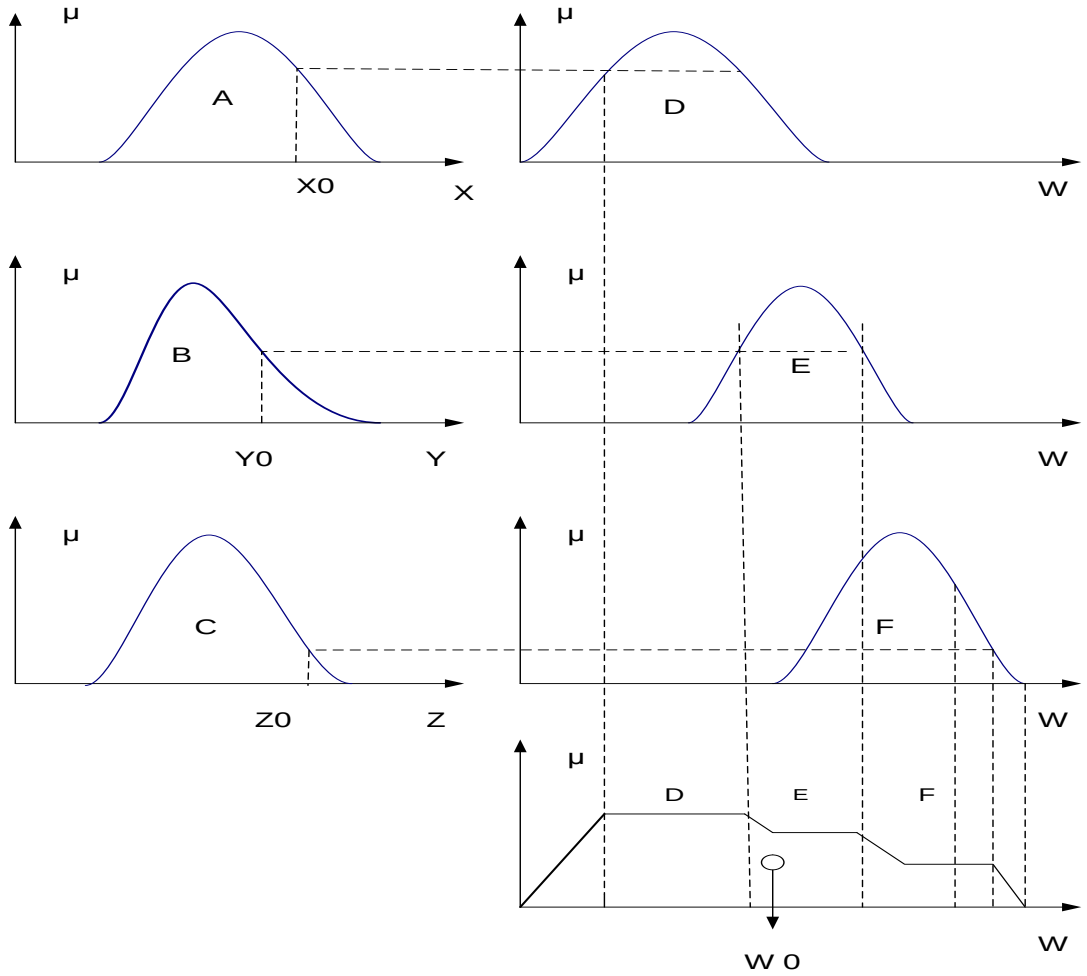


Рис. 1.5. Процедура нечеткого вывода

Метод среднего максимума. Для дискретного варианта представления множеств:

$$Z_0 = \frac{1}{n}\sum_{i=1}^{n} z_i \ . \qquad (1.7)$$

где z_i – элемент эталонного множества;

n – число элементов-максимумов одного уровня

Z_0 – четкое значение.

Кроме описанных алгоритмов, можно отметить такие алгоритмы как Tsukamoto, Sugeno, Larsen [6, 7].

1.9. Нечеткие когнитивные карты

Когнитивная карта (от лат. cognitio – знание, познание) – образ знакомого пространственного окружения. Когнитивные карты создаются и видоизменяются в результате активного взаимодействия субъекта с окружающим миром. При этом могут формироваться когнитивные карты различной степени общности, «масштаба» и организации. Это субъективная картина, имеющая прежде всего пространственные координаты, в которой локализованы отдельные воспринимаемые предметы. Выделяют карту-путь как последовательное представление связей между объектами по определенному маршруту, и карту-обозрение как одновременное представление пространственного расположения объектов [29, 30].

Когнитивные карты относятся к ранним компонентам памяти, они являются как бы схемой, канвой, или эскизом для формирования других предметных отражений. Они контролируют частные образы воображения. По сравнению с элементами реального пространства у когнитивных карт бывают систематические искажения.

На сегодняшний день используется большое количество методов для моделирования динамических систем и процессов, протекающих в них. Выбор методов определяется степенью информированности о поведении системы. Классическим является подход, описывающий систему в виде дифференциальных уравнений и основной задачей является подбор параметров, входящих в уравнения. В ряде случаев при исследовании сложных систем не удается построить достоверную математическую модель из-за большой неопределенности взаимодействия компонентов системы. Другим подходом является применение *нечетких когнитивных карт,* который позволяет выполнить моделирование поведения системы и проводить численные эксперименты.

Нечеткие когнитивные карты (НКК) были предложены Б. Коско в 1986 г. и применяются для моделирования причинных взаимосвязей, обнаруженных между концептами предметной области. В отличие от простых когнитивных карт, НКК представляет нечеткий ориентированный граф с обратной связью, узлы которого являются нечеткими множествами, а направленные ребра графа представляют причинно-следственные связи между концептами и характеризуют степень влияния (вес) связываемых концептов. НКК объединяет в себе свойства нечетких и нейронных сетей.

Основные проблемы связаны с процессом построения когнитивной карты, который не поддаётся формализации. Кроме того, необходимо доказать, что построенная когнитивная карта адекватна реальной моделируемой системе. Для решения данных проблем разработаны алгоритмы автоматического построения когнитивных карт на основе выборки данных.

Активное использование НКК в качестве средства моделирования обусловлено возможностью наглядного представления анализируемой системы и интерпретируемостью причинно-следственных связей между концептами[2].

[2] «Когнитивные технологии для поддержки принятия управленческих решений» http://www.iis.ru/events/19981130/maximov.ru.html

Процесс формирования и использования НКК состоит из следующих шагов:

1. Определение списка концептов, которые могут характеризовать события, действия, величины или цели.
2. Определение степени влияния между каждой парой концептов или задание функций принадлежности на каждом терме.
3. Построение нечеткой когнитивной карты.
4. Анализ и интерпретация нечеткой когнитивной карты.

Простые НКК содержат связи, которые могут принимать одно из трех значений {–1,0,1}. Значение +1 означает положительное влияние первого на второй концепт, минус 1 – отрицательное влияние, а ноль свидетельствует об отсутствии отношений причинности между концептами и на карте связь не отображается. Пример НКК приведен на рис.1.6.

Большой интерес представляет применение соответствующих технологий в области поддержки принятия решений и ситуационных центрах[3].

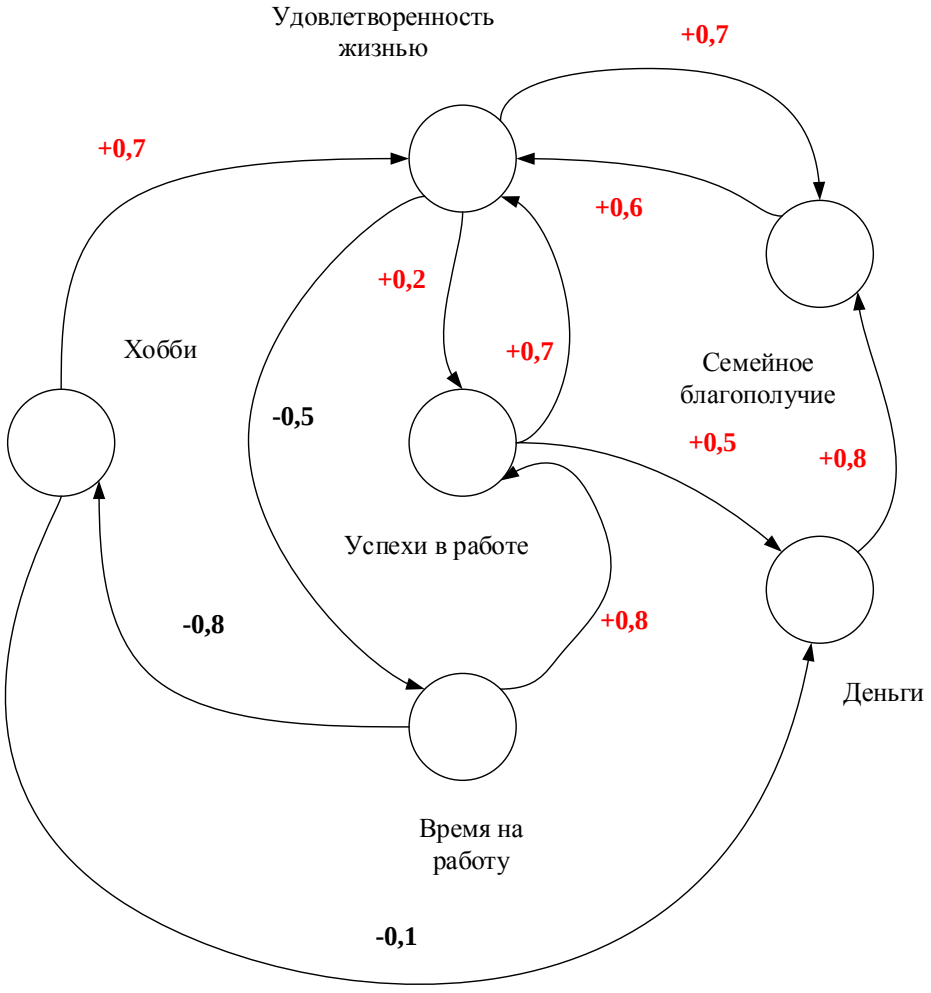


Рис.1.6. Иллюстрация нечеткой когнитивной карты

---

[3] «Когнитивные технологии для поддержки принятия управленческих решений» http://www.iis.ru/events/19981130/maximov.ru.html

В 1986 году Б. Коско предложил новый тип когнитивных карт [35–37], получивших название *«нечеткие когнитивные карты»* (Fuzzy Cognitive Maps).

Концепты в нечеткой когнитивной карте (НКК) могут принимать значения из диапазона действительных чисел [0, 1].

Термин «нечеткие» обозначает только то, что причинные связи могут принимать не только значение, равное 0 или 1, а лежат в диапазоне действительных чисел, отражающих «силу» влияния одного концепта на другой.

Нечеткие когнитивные карты (НКК), основанные на правилах, позволяют представить сложные системы в динамике с моделированием обратной связи и симуляцией внешних воздействий и их воздействий на систему. Такие НКК представляют собой ориентированный граф, состоящий из нечетких узлов (концептов) и нечетких связей (отношений).

Применение НКК, основанных на правилах, не ограничено представлением казульных отношений. Такие НКК моделируют работу системы потактово: каждое следующее состояние вычисляется на основе значений предыдущего состояния. НКК, основанные на правилах, могут быть использованы для предсказания эволюции системы с течением времени. С введением механизмов, позволяющих подавлять конкретные концепты и/или отношения, когда они не оказывают влияние на данное состояние, такие НКК могут стать инструментом для анализа динамики качественных систем.

Концепты представляют собой сущности, которые образуют систему. В НКК, основанных на правилах, выделяют 2 вида концептов: уровни и изменения. Уровни представляют собой абсолютные значения концептов при данном состоянии системы. Изменения представляют собой разницу между текущим и предыдущим состояниями. Поскольку изменения важны для нормальных отношений, а такие отношения чаще всего используются в когнитивных картах, многим системам не важны абсолютные значения концептов.

Отношения. В НКК, основанных на правилах, отношения определяются использованием различных нечетких правил вида «если … то». Рассмотрим отношения, реализованные в таких НКК. При появлении новых отношений должны быть разработаны нечеткие механизмы для их реализации.

При работе с когнитивными картами многие сталкиваются с проблемой: как правильно определить казуальные отношения между двумя концептами. Многие авторы считают, что такие отношения должны включать изменение. Изменение одного концепта должно вызвать эффект, который в свою очередь вызывает изменение другого концепта. Кроме того, казуальные эффекты должны быть аккумулятивными. Другими словами, несколько эффектов накладываются, усиливая или ослабляя друг друга. Подытоживая, мы можем говорить о казуальных отношениях, когда имеем дело с изменениями. На рис.1.8 представлен пример, какие факторы влияют на инфляцию.

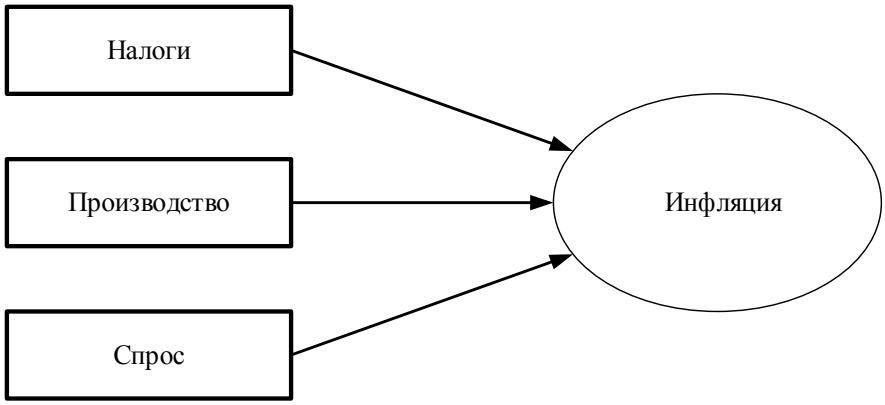


Рис.1.8. Казуальное отношение

Отношения влияния – это более общее представление казуальных отношений [29, 30]. Они представлены правилами вида «если … то». Пример представлен на рис.1.9.

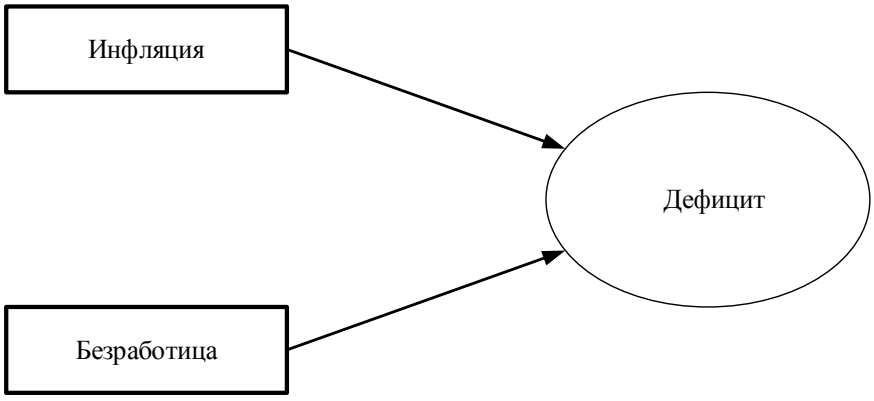


Рис.1.9. Отношение влияния

Нечеткие казуальные отношения практически не отличаются от обычных казуальных отношений, за исключением нечеткого базиса правил.

В случае казуальных отношений мы абсолютно уверены, что событие произойдет. Если это не так используются стохастические отношения. Имеется некая постоянная вероятность, что событие случится. Пример представлен на рис.1.10, где показаны возможные исходы революции.

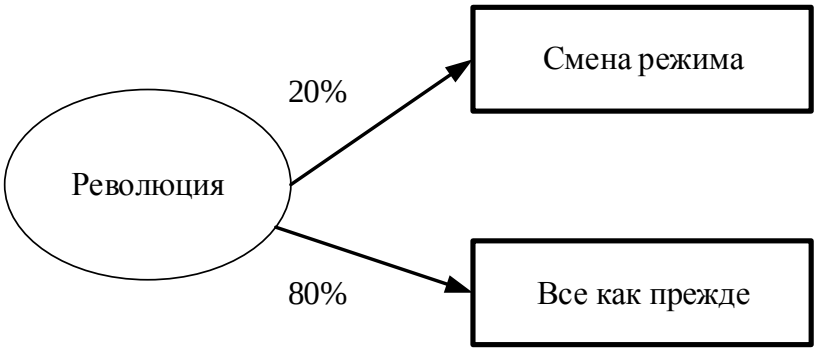


Рис.1.10. Стохастическое отношение

Иногда эта величина – не константа, а переменная, зависящая от времени. Тогда мы имеем дело со стохастической вероятностью, зависящей от времени, что позволяет моделировать ситуации, вероятность которых изменяется с течением времени.

4. НКК, основанные на правилах, и время при исследовании динамики систем.

Время необходимо для изучения динамики систем. При описании математической модели время задается в явном виде в математических выражениях. В случае качественных систем, оно вводится неявно при задании правил вывода. Без знания временных особенностей функционирования системы невозможно гарантировать нормальную симуляцию. Необходимо учитывать следующие рекомендации. Важно выбрать базовый интервал времени для представления каждой итерации. При определении итерации базовый интервал должен быть неявно задан. Правила, представляющие казуальные эффекты сильно зависят от базового интервала. Следует учитывать, что чем меньше базовый интервал, тем более подробные и сложные должны быть правила вывода.

В НКК, основанных на правилах, разные концепты могут оперировать с временными интервалами, отличными от базового интервала. То есть каждому концепту и отношению можно поставить в соответствие свой временной интервал, указывающий на какой итерации этот концепт/отношение должны быть применены [35–37]. НКК, основанные на правилах, предложены в качестве средство для анализа качественных систем в динамике. Были рассмотрены виды отношений, представлено руководство по использованию временных параметров в качественных системах.

Контрольные вопросы

1. Отличие знаний от данных.
2. Классификация знаний.
3. Примеры декларативных, процедурных, глубинных и поверхностных знаний.
4. Понятие интенсионала и экстенсионала.
5. Примеры продукционных правил.
6. Понятие фрейма, характеристика его слотов и пример фрейма.
7. Понятие семантической сети и пример.
8. Способы представления знаний в нейронной сети.
9. Пример нечетких правил.
10. Пример лингвистической переменной.
11. Метод нечеткого вывода Мамдани.
12. Понятие нечеткой когнитивной карты.
13. Области применения нечетких когнитивных карт.

2. ОСНОВНЫЕ ПОНЯТИЯ И НАЗНАЧЕНИЕ СИСТЕМ ОБРАБОТКИ ЗНАНИЙ

2.1. Понятие экспертной системы

В 60-70-х годах прошлого столетия в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название *«экспертные системы»* (ЭС).

Искусственный интеллект представляет собой область информатики, которая занимается разработкой интеллектуальных компьютерных систем, т. е. систем, обладающих возможностями, которые традиционно связаны с человеческим разумом: пониманием языка, обучением, способностью рассуждать, решать проблемы и т. д.

Цель исследований по экспертным системам[4] состоит в разработке программ, предназначенных для решения трудноформализуемых задач, не уступающих по качеству и эффективности решениям, которые получает эксперт.

Исследователи в области ЭС для названия своей дисциплины часто используют также термин «*инженерия знаний*» как «привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов» [1, 3, 4]. Процесс организации знаний в базу знаний и построения экспертной системы называют *инженерией знаний*.

Технология экспертных систем используется для решения различных типов задач: интерпретации предсказания, диагностики, планирования, конструирования, контроля, отладки, инструктажа, управления.

Решения экспертных систем обладают «прозрачностью» и могут быть объяснены пользователю на качественном уровне. Это свойство экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях. Экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом. В литературе приводятся разные определения ЭС. В настоящее время отсутствует единое определение ЭС. Наиболее популярными являются следующие.

Экспертная система – это программа для компьютера, которая оперирует со знаниями в определенной предметной области с целью выработки рекомендаций или решения проблем [3, 4].

Экспертная система – это сложный программно-аппаратный комплекс, аккумулирующий знания специалистов в конкретной предметной области и тиражирующий этот эмпирический опыт для консультаций менее квалифицированных пользователей.

Экспертная система – это система, основанная на знаниях (knowledge-based system).

---

[4] «Когнитивные технологии для поддержки принятия управленческих решений» http://www.iis.ru/events/19981130/maximov.ru.html

Строго говоря, экспертная система – это более широкое понятие. Система, основанная на знаниях, это любая система, процесс работы которой основан на применении правил отношений к символическому представлению знаний. Например, программа, способная рассуждать о погоде, будет системой, основанной на знаниях даже в том случае, если она не способна выполнить метеорологическую экспертизу. Программа, способная давать прогноз погоды, имеет право называться метеорологической ЭС.

Огромный интерес к ЭС со стороны пользователей вызван, по крайней мере, тремя причинами.

– Пользователи ориентированы на решение широкого круга задач в неформализованных областях, малодоступных для вычислительной техники.

– С помощью ЭС специалисты, не знающие технологии программирования, могут самостоятельно разрабатывать интересующие их приложения, что позволяет резко расширить сферу использования вычислительной техники.

– ЭС при решении практических задач достигает результатов, не уступающих, а иногда и превосходящих возможности людей-экспертов.

Экспертные системы предназначены для решения так называемых неформализованных задач, обладающих одной или несколькими характеристиками:

– задачи не могут быть заданы в числовой форме (символьный вывод);

– цели не могут быть выражены в терминах точно определенной целевой функции;

– не существует алгоритмического решения задач;

– применение эвристического поиска решений в условиях, когда алгоритмическое решение существует, но его нельзя использовать из-за ограниченности ресурсов (параметры времени и памяти).

Неформализованные задачи обычно характеризуются аспектами неполной информации: *неточностью, неопределенностью, нечеткостью* проблемной области [3]. Кроме того, наблюдается большая размерность пространства решений (перебор при поиске решения весьма велик) и динамически изменяющиеся данные. Неточные данные задаются в интервальной форме [D-E; D+E]. Неопределенность – это неизвестное значение истинности высказывания. Нечеткость данных связана с заданием функции принадлежности элементов множества.

2.2. Компоненты типовой статической экспертной системы

Типовая структура статической ЭС состоит из следующих основных компонентов, приведенных на рис. 2.1: модуля интерфейса пользователя; базы знаний (БЗ); машины логического вывода; модуля объяснений; интеллектуального редактора базы знаний.

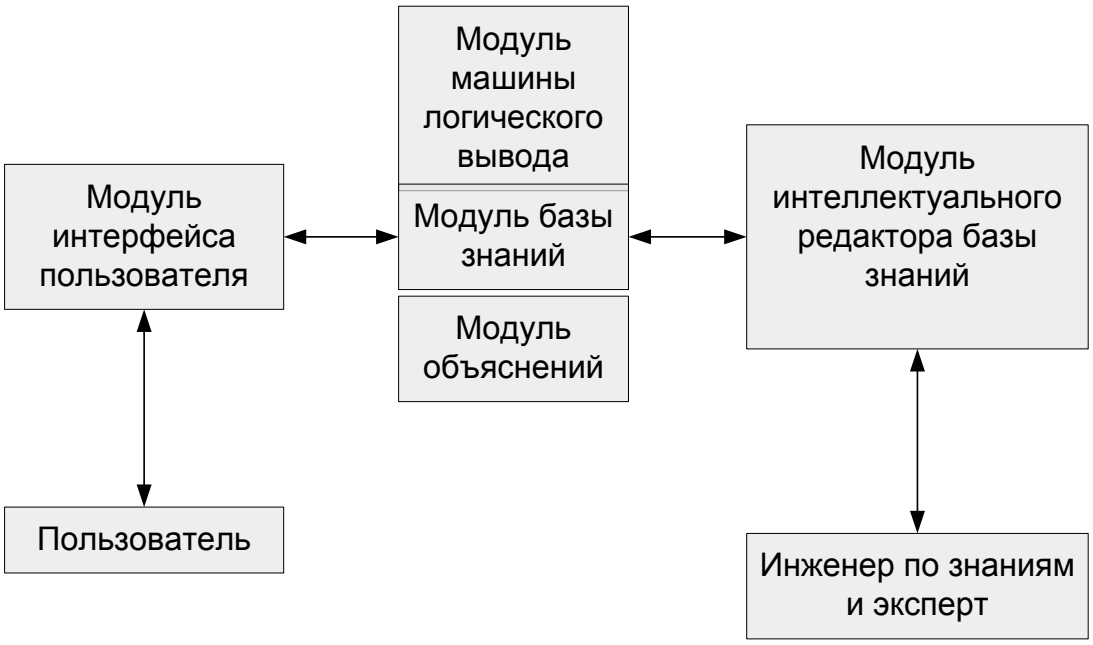


Рис. 2.1. Структура статической экспертной системы

Пользователь – специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС.

Инженер по знаниям – специалист по искусственному интеллекту, выступающий в роли промежуточного буфера между экспертом и БЗ.

Модуль интерфейса пользователя – программа, реализующая диалог пользователя с ЭС как на стадии ввода информации, так и получения результатов.

Модуль базы знаний – ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному языку). Параллельно такому «человеческому» представлению существует БЗ во внутреннем «машинном» представлении.

Модуль машины логического вывода – программа, моделирующая ход рассуждения эксперта на основании знаний, имеющихся в БЗ [1, 3, 4].

Модуль объяснений – программа, позволяющая пользователю получить ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?»

Модуль интеллектуального редактора БЗ – программа, представляющая инженеру по знаниям возможность создавать и редактировать БЗ в диалоговом режиме.

В разработке ЭС участвуют *эксперт проблемной области, инженер по знаниям и программист*.

Экспертная система работает в двух режимах: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

В режиме *приобретения знаний* общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области [1].

В режиме *консультации* общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения [4]. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам) или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу).

В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память.

2.3. История развития экспертных систем

Наиболее известные ЭС, разработанные в 60–70-х годах, стали в своих областях уже классическими. По происхождению, предметным областям и по преемственности применяемых идей, методов и инструментальных программных средств ЭС можно разделить на несколько семейств [1, 3, 4, 8, 11, 12, 13].

1. META-DENDRAL. ЭС DENDRAL позволяет определить наиболее вероятную структуру химического соединения по экспериментальным данным. Она автоматизирует процесс приобретения знаний и генерирует правила построения фрагментов химических структур.
2. MYCIN-EMYCIN-TEIREIAS-PUFF-NEOMYCIN. Это семейство медицинских ЭС и сервисных программных средств для их создания.
3. PROSPECTOR-KAS. Экспертная система PROSPECTOR предназначена для поиска (предсказания) месторождений на основе геологических анализов. KAS – система приобретения знаний для PROSPECTOR.
4. CASNET-EXPERT. Система CASNET – медицинская ЭС для диагностики выдачи рекомендаций по лечению глазных заболеваний. На ее основе разработан язык инженерии знаний EXPERT, с помощью которой создан ряд других медицинских диагностических систем.
5. HEARSAY-HEARSAY-2-HEARSAY-3-AGE. Первые две системы этого ряда являются развитием интеллектуальной системы распознавания слитной человеческой речи, слова которой берутся из заданного словаря. Эти системы отличаются оригинальной структурой, основанной на использовании доски объявлений – глобальной базы данных, содержащей текущие результаты работы системы. В дальнейшем на основе этих систем были созданы инстру-

ментальные системы HEARSAY-3 и AGE (Attempt to Generalize – попытка общения) для построения ЭС.

6. Системы AM (Artifical Mathematician – искусственный математик) и EURISCO были разработаны в Станфордском университете для исследовательских и учебных целей.

В экспертную систему AM первоначально было заложено около 100 правил вывода и более 200 эвристических алгоритмов обучения, позволяющих строить произвольные математические теории и представления. Дальнейшее развитие системы замедлилось, и было отмечено, что, несмотря на проявленные на первых порах «математические способности», система не может синтезировать новых эвристических правил, т.е. ее возможности определяются только теми эвристиками, что были в нее изначально заложены.

При разработке системы EURISCO была предпринята попытка преодолеть указанные недостатки системы AM. Как и в начале эксплуатации AM, первые результаты, полученные с помощью EURISCO, были эффективными. Сообщалось, что система EURISCO может успешно участвовать в очень сложных играх. С ее помощью в военно-стратегической игре, проводимой ВМФ США, была разработана стратегия, содержащая ряд оригинальных тактических ходов. Например, предлагалось взрывать свои корабли, получившие повреждения. При этом корабли, оставшиеся неповрежденными, получает необходимое пространство для выполнения маневра.

Однако через некоторое время обнаружилось, что система не всегда корректно переопределяет первоначально заложенные в нее правила. Так, например, она стала нарушать строгое предписание обращаться к программистам с вопросами только в определенное время суток. Экспертная система EURISCO, так же, как и ее предшественница, остановилась в своем развитии, достигнув предела, определенного ее разработчиком.

Некоторые примеры успешного применения экспертных систем:

– только в США ежегодный доход от продаж инструментальных средств разработки ЭС составлял в начале 90-х годов 300–400 млн. долларов, а от применения ЭС – от 80 до 90 млн. долларов [1, 3, 4];

– XCON (США) экономит 70 млн. долларов в год [1, 3, 4] благодаря экспертной системе XCON/XSEL, которая по заказу покупателя составляет конфигурацию вычислительной системы VAX. Использование экспертной системы сократило количество ошибок от 30% (допускал человек) до 1% (допускает экспертная система);

– фирма Sira (США) сократила затраты на строительство трубопровода в Австралии на 40 млн. долларов [1, 3] за счёт экспертной системы, управляющей трубопроводом. Экспертная система реализована на базе инструментальных средств G2 (фирма Gensym);

– фирма Monsanto (США) ежегодно экономит от 250 до 500 тыс. долларов благодаря экспертной системе выявления и блокирования неисправностей в нефтехимической промышленности. Экспертная система реализована на базе инструментальных средств G2 (фирма Gensym);

– фирма Aetha Insurance (США) уже сэкономила более 5 млн. долларов, а общий планируемый эффект составит 15–20 млн. долларов благодаря экспертной системе, используемой для моделирования страховых исков, обрабатываемых компанией. Экспертная система, реализованная на базе инструментальных средств G2 (фирма Gensym), позволяет находить в деятельности компании неэффективные процессы и рабочие потоки и производить оперативные изменения для увеличения продуктивности работы.

Среди отечественных разработок можно отметить следующие экспертные системы [3, 4, 8, 11]:

1. Экспертная диагностическая система, предназначенная для помощи лицам, принимающим решения при мониторинге и управлении сложными объектами и процессами различной природы в условиях жестких временных ограничений.
2. ЭС ДИАГЕН для диагностики наследственных болезней.
3. ЭС ДИН для диагностики неотложных состояний у детей в условиях ограничений на проведение дополнительных исследований, обусловленных тяжестью состояния или недостатком диагностической аппаратуры.
4. ЭС ВЕСТ-СИНДРОМ для диагностики эпилепсии.
5. ЭС поддержки проектирования процесса нормализации пленочных резисторов.

Большинство публикаций по отечественным ЭС приходится на медицину. Использование консультирующих ЭС, основанных на знаниях, особенно важно для начинающих врачей, не обладающих личным опытом. Такие ЭС не пропустят следующие ситуации:

– часто встречающаяся болезнь с нетипичными симптомами;
– симптомы-миражи, которые не имеют никакого отношения к заболеванию;
– напрасный поиск «зебр» (редких болезней) с самого начала диагностического процесса.

В последнее время наметилась тенденция разработки ЭС на базе нейронных сетей, нечеткой логики, которые способны обучаться и использовать интуитивный опыт экспертов [2, 3, 4, 7, 9, 10, 11]. Также отмечается в качестве перспективного направления создание гибридных экспертных систем [7].

2.4. Классификация экспертных систем

Наиболее популярная классификация, предложенная Т. А. Гавриловой и В. Ф. Хорошевским в [4], подразделяет ЭС на четыре основных класса: по типу решаемой задачи; по связи с реальным временем; по типу используемой ЭВМ; по степени интеграции.

Классификация по типу решаемой задачи: интерпретация данных; диагностика; мониторинг; проектирование; прогнозирование; планирование; обучение, управление.

Интерпретация в переводе с латинского понимается как истолкование, объяснение, перевод на более понятный язык или процесс определения смысла данных, например, определение основных свойств личности по результатам тестирования в системах АВАНТЕСТ и МИКРОЛЮШЕР; идентификация типов океанских судов по результатам аэрокосмического сканирования (SIAP).

Диагностика – это процесс соотнесения объекта с некоторым классом объектов и(или) обнаружение неисправности в объекте, например, диагностика и терапия сужения коронарных сосудов (ANGY).

Мониторинг – это непрерывная интерпретация данных в реальном масштабе времени, и сигнализация о выходе параметров за допустимые пределы, например, мониторинг за работой атомного реактора (REACTOR), контроль аварийных датчиков на химическом заводе (FALCON).

Проектирование – это процесс создания проекта и подготовка спецификаций на объект. Под спецификацией понимается комплект разрабатываемой документации, например, проектирование СБИС (CADHELP).

Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных, например, предсказание погоды (WILLARD), прогнозы в экономике (ECON).

Планирование – это нахождение плана действий, относящихся к объектам, способным выполнять некоторые функции, например, планирование поведения робота (STRIPS), планирование промышленных заказов (ISIS).

Обучение – это использование персонального компьютера для обучения какой-либо дисциплине, например, обучение языку программирования LISP с помощью программы «Учитель ЛИСП».

Управление – это функция организованной системы, поддерживающая определенный режим деятельности, например, управление системой календарного планирования Project Assistant.

Классификация по связи с реальным временем подразделяет ЭС на статические, квазидинамические и динамические [4].

Статические ЭС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени, например, ЭС диагностики состояния автомобиля.

Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Например, микробиологическая ЭС, которая считывает показания датчиков с технологического процесса один раз в 4-5 часов.

Динамические ЭС работают в режиме реального времени и производят непрерывную интерпретацию поступающих в систему данных, например, ЭС мониторинга за состоянием больного в реанимации.

Классификация по типу используемой ЭВМ выделяет ЭС, построенные с использованием: суперЭВМ (например, GRAY); мейнфреймов (например, System 390 серии G5); рабочих станций (SUN); миниЭВМ (VAX); персональных компьютеров; карманных персональных компьютеров.

Классификация по степени интеграции с другими программами подразделяет ЭС на автономные и интегрированные.

Автономные ЭС работают непосредственно в режиме консультаций с пользователем, при этом для решения задач не требуется привлекать традиционные методы обработки данных (расчеты, моделирование, статистическая обработка).

Интегрированные ЭС представляют собой программный комплекс, агрегирующий стандартные пакеты программ (например, математическую статистику) или системы управления базами данных и средства манипулирования знаниями.

2.5. Классификация инструментальных средств создания экспертных систем

В настоящее время известны три основные направления проектирования экспертных систем [1, 3, 4, 5, 6]:

1. Экспертные системы, выполненные в виде отдельных программ, на некотором алгоритмическом языке, база знаний которых является непосредственно частью этой программы. Как правило, такие системы предназначены для решения задач в одной конкретной предметной области. При создании таких систем применяются как традиционные процедурные языки PASCAL, C++ и т. д., так и специализированные языки искусственного интеллекта LISP, PROLOG.
2. Экспертные системы, построенные с использованием оболочек ЭС, которые представляют собой программный продукт, обладающий средствами представления знаний для определенных предметных областей. Задача пользователя заключается в формализации и вводе знаний с использованием специальной оболочки. Недостатком этих систем можно считать невозможность охвата одной системой всех существующих предметных областей. Примером могут служить ИНТЕР-ЭКСПЕРТ, PC+, VP-Expert, EMYCIN.
3. Экспертные системы, построенные с помощью генератора экспертных систем, который представляет собой мощный программный комплекс, предназначенный для получения оболочек, ориентированных на то или иное представление знаний в зависимости от рассматриваемой предметной области. Примеры этой разновидности – системы KEE, ART.

Система «Expert» является скелетным языком инженерии знаний, которая использует схему представления знаний, основанную на правилах, и имеет ограниченный механизм вывода, организованный по принципу прямой цепочки рассуждений и делающий его пригодным для задач типа диагностики и классификации. В этой системе имеются встроенные блоки построения объяснений, приобретения знаний и контроля непротиворечивости, которые ускоряют разработку. Блок контроля непротиворечивости хранит базу данных репрезентативных случаев с известными заключениями и использует их для тестирования экспертной системы после того, как инженер знаний добавляет новые правила. Если в каком-то случае не удаётся получить правильных рассуждений, то система «Expert» представляет процесс рассуждений до этого случая, чтобы ин-

женер знаний мог понять, какое новое правило привело к неожиданным результатам [4, 11, 12].

ИНТЕР-ЭКСПЕРТ – это интегрированная система ведения баз данных и баз знаний. Она представляет собой среду для создания экспертных систем в области делопроизводства и экономики, создания и ведения электронных ведомостей, использование деловой графики, составление отчётов и электронных таблиц.

В системе реализован принцип синергизма (каждой части ИНТЕР-ЭКСПЕРТ доступны все данные, имеющиеся в системе). Экспертная система, созданная с помощью ИНТЕР-ЭКСПЕРТ, имеет доступ к информации, хранимой в базе данных, электронной ведомости, другом наборе правил, таблице, файле графиков.

ACQUIRE – это законченная среда для разработки и поддержки интеллектуальных прикладных программ. Система содержит в себе методологию пошагового представления знаний, что позволяет специалистам в проблемной области непосредственно участвовать в процессе приобретения, структурирования и кодирования знания. Прямое участие специалиста в проблемной области улучшает качество, законченность и точность приобретенного знания, снижает время разработки и эксплуатационные расходы.

Особенностью оболочки является структурированный подход к приобретению знаний. Модель приобретения знаний основана на распознавании образов. Знания представлены как объекты, а продукционные правила – в табличной форме. Оболочка позволяет выполнять обработку неопределенных качественных знаний, содержит средства вывода и документацию баз знаний в среде гипертекста.

Инструментальная система G2 фирмы Gensym предлагает графическую, объектно-ориентированную среду для создания интеллектуальных прикладных программ, которые контролируют, диагностируют и управляют динамическими событиями в сетевых и моделируемых средах [http://www.gensym.com].

Инструментальная система G2 предназначена для создания ЭС на базе продукционных правил и других моделей (процедур) с использованием структурированного естественного языка. Инструментальная экспертная система G2 является основой всех прикладных программ фирмы Gensym. Она позволяет использовать визуальную среду программирования для создания интеллектуальных прикладных программ управления. NeurOn-Line и другие программы фирмы позволяют пользователям создавать нейросетевые прикладные программы. G2 совмещает выполнение правил и процедур в текущий момент времени со способностями рассуждений спустя некоторое время. Руководство по G2 позволяет пользователям создавать графические интерфейсы и системы диагностирования в реальном масштабе времени. Компания Telewindows Gensym's создала более мощную универсальную среду клиент/сервер, которая позволяет пользователям совместно использовать прикладные программы на основе G2.

Фирма Gensym также предлагает мосты (программы) для связи с другими программам (на языках С и АДА) и системы передачи и обработки данных о движущихся объектах в реальном времени, включая реляционные базы данных, распределенные системы управления и программируемые логические системы.

COMDALE/C – экспертная система реального времени, предназначенная для наблюдения и контроля над процессами в условиях производства [http://www.comdale.com].

+COMDALE/C позволяет вырабатывать рекомендации, заключения об управляющих воздействиях в непрерывном процессе принятия решения. Она обрабатывает неопределенные знания и данные и имеет открытую архитектуру. Её основные характеристики – объектно-ориентированная конфигурация; возможность организации работы в сети; обработка прерываний; хранение и обработка данных; поддержка работы с базой данных в реальном масштабе времени; интерфейсы с системами передачи данных.

+COMDALE/X – консультационная экспертная система, которая работает в режиме реального времени. Для принятия решения система организует диалог с пользователем. COMDALE/X совместно с системой COMDALE/C используется как инструмент разработки экспертных систем реального времени. COMDALE/X позволяет включить гипертекст в экспертную систему, что позволяет создавать hyper-справочники с удобным интерфейсом.

FLEX – гибридная экспертная система, работающая на различных платформах [http://vvv.com/ai/]. Система предлагает фреймовое, процедурное и продукционное представление знаний.

FLEX чередует прямой и обратный методы поиска решений, множественное наследование свойств, присоединенные процедуры, автоматическую систему вопросов и ответов. Правила, фреймы и вопросы написаны на естественном англо-подобном языке. Язык спецификаций (KSL) позволяет разрабатывать легко читаемые и простые в поддержке базы знаний. Экспертная система FLEX разработана на языке Пролог и использовался в многочисленных коммерческих финансовых экспертных системах.

EXSYS 3.0 (Exsys Inc.) – оболочка, предназначенная для создания экспертных прикладных систем классификационного типа. Использует прямую и обратную цепочки вывода, организует базу продукционных правил, имеющих вид «ЕСЛИ-ТО-ИНАЧЕ». EXSYS позволяет использовать данные из других программ, электронных таблиц, баз данных, поддерживает математические вычисления, строковые и числовые переменные [11, 12].

NEXPERT OBJECT (Neuron Data Corp.) – это мощная система, базирующаяся на использовании правил. Она поддерживает различные типы правил и комбинации прямого и обратного выводов [4, 12]. Система может автоматически строить сеть правил в графическом изображении, что позволяет наблюдать, каким образом правила связаны друг с другом. NEXPERT OBJECT имеет возможности представления обоих типов фреймов (с наследованием и без наследования) и механизм сопоставления с образом. В этой системе большое внимание уделено графическому представлению баз данных и объясняющих возможностей, что делает возможным организацию естественных и дружественных интерфейсов разработчика и конечного пользователя. Средство написано на языке С.

Все ведущие производители инструментальных средств экспертных систем (Gensym, Inference, Intellicorp, Neuron Data) признали и реализуют про-

блемно/предметно-ориентированные инструментальные средства. Наибольшего успеха в этом направлении добилась фирма Gensym со своими продуктами: G2, GDA, DSP, NeurOn-Line, Rethink [1, 3, 4, 5].

Развитие ориентированных (предметно-ориентированных) инструментальных средствах проводится в следующих направлениях:

– инструментальные средства для динамических экспертных систем реального времени, используемых в управлении технологическими процессами и имитационном моделировании [1, 3, 4];

– инструментальные средства для систем-советчиков;

– инструментальные средства для систем, основанных на прецедентах.

В области инструментальных средств и динамических экспертных систем доминирующие позиции занимает фирма Gensym (G2, GDA, DSP, NOL), затем идут Talarian (RTworks) и Comdale Technologies (Comdale/C, Comdale/X).

MYCIN – экспертная система для диагностики и лечения инфекционных заболеваний [4].

Разработан скелетный язык, иначе – оболочка ЭС EMYCIN [4]. Декларативные знания системы MYCIN описываются в виде «объект-атрибут-значение». Каждой тройке приписывается коэффициент уверенности, определяющий степень надежности знаний. Процедурные знания описаны в виде классического правила продукции. Механизм логического вывода основан на обратной цепочке рассуждений. Поиск производится в иерархически упорядоченном пространстве состояний.

В оболочке EMYCIN [4] по отношению к MYCIN усилена функция редактирования БЗ, доведена до высокого уровня система объяснения хода решения задачи, а также аппарат обучения системы. Оболочка разработана на языке Фортран.

OPS-5 – универсальный язык инженерии знаний, предназначенный для разработки ЭС, используемых в коммерческих приложениях.

Декларативные знания в системе описаны в виде «объект-атрибут-значение». Процедурные знания описаны в виде классических правил продукции.

В Российском научно-исследовательском институте информационных технологий и систем автоматизированного проектирования (РосНИИ ИТ и АП) разработан комплекс ЭКО [4,12]. Наиболее успешно комплекс применяется для создания экспертных систем, решающих задачи диагностики (технической и медицинской), эвристического оценивания (риска и надёжности и т.д.), качественного прогнозирования, а также обучения.

На основе комплекса ЭКО было разработано более 100 прикладных экспертных систем. Среди них можно отметить следующие:

– поиск одиночных неисправностей в персональном компьютере;

– оценка состояния гидротехнического сооружения;

– подготовка деловых писем при ведении переписки с зарубежными партнёрами.

Структура комплекса ЭКО включает три компонента. Ядром комплекса ЭКО является интегрированная оболочка экспертных систем, которая обеспечивает быстрое создание эффективных приложений для решения задач анализа.

Оболочка функционирует в двух режимах: в режиме приобретения знаний и в режиме консультации (решения задач). В первом режиме разработчик экспертных систем средствами диалогового редактора вводит в базу знаний описание конкретного приложения в терминах языка представления знаний оболочки. Во втором режиме оболочка решает конкретные задачи пользователя в диалоговом или пакетном режиме.

Для расширения возможностей оболочки по работе с глубинными знаниями комплекс ЭКО может быть дополнен компонентом К-ЭКО (конкретизатором знаний), который позволяет описывать закономерности в проблемных средах в терминах общих объектов и правил. К-ЭКО используется на этапе приобретения знаний вместо диалогового редактора оболочки для преобразования общих описаний в конкретные сети вывода, допускающие эффективный вывод решений средствами оболочки ЭКО.

Третий компонент комплекса ЭКО – система ИЛИС, позволяющая создавать экспертную систему в проблемных средах за счёт индуктивного обобщения данных (примеров) и предназначенная для использования в тех приложениях, где отсутствие правил, отражающих закономерности в проблемной среде, возмещается экспериментальными данными.

Средства представления знаний в оболочке ЭКО представляют собой совокупность нескольких моделей в базе знаний, каждая из которых описывает отдельное конкретное приложение или его компонент. Отдельная модель включает описание проблемной среды и знания о порядке решения задач. Описание проблемной среды состоит из описаний атрибутов и правил вывода. Атрибуты используются для описания состояний предметной области.

Оболочка работает со статистическими проблемными средами (значения атрибутов не изменяются в ходе решения задачи), в которых предметная область может быть описана с помощью априорно заданного набора атрибутов. Не допускается динамическое создание атрибутов во время решения задачи. Средства комплекса позволяют представлять качественные (символьные) и количественные (числовые) характеристики предметной области.

Высказывания типа «А есть В» называются утверждениями о состоянии предметной области. В этом высказывании посылка А представляет атрибут (качественную характеристику), а заключение В – одно из возможных значений высказывания. Решение задачи сводится к получению значений некоторых целевых атрибутов и определению истинности некоторых целевых утверждений. Оболочка позволяет работать с неточно и нечётко определёнными знаниями. С каждым утверждением о состоянии предметной области связывается коэффициент определённости, который характеризует степень уверенности в его истинности. Вывод решения заключается в нахождении коэффициентов определённости некоторых целевых утверждений, указанных разработчиком экспертных систем. Для построения вывода в условиях неопределённости может использоваться байесовский подход. Коэффициенты определённости утвержде-

ний – это действительные числа, принимающие значения от минус 5,00 до 5,00. Коэффициенту определённости D(H) утверждения H можно дать следующую интерпретацию [4, 18, 19]:

–

– если точно известно, что H истинно, то D(H) = 5,00;

– если точно известно, что H ложно, то D(H) = −5,00;

– если H может быть с одинаковой уверенностью истинно или ложно, то D(H) = 0,00;

– если H скорее истинно, чем ложно, то 0,00 < D(H) < 5,00, причём D(H) тем больше, чем больше уверенность в истинности H;

– если H скорее ложно, чем истинно, то −5,00 < D(H) < 0,00, причём D(H) тем меньше, чем больше уверенность в ложности H.

Утверждения и числовые атрибуты модели называются целями, а символьные атрибуты, представляющие собой множество утверждений, называются сложными целями. Значения целей определяются с помощью простых и сложных правил вывода:

– простой вопрос позволяет получать либо значение числового атрибута, либо коэффициент определённости отдельного утверждения;

– сложный вопрос позволяет получить распределение коэффициентов определённости по всем возможным значениям символьного атрибута;

– альтернативный вопрос используется в тех случаях, когда известно, что символьный атрибут имеет точно одно значение из множества возможных значений;

– дистрибутивный вопрос используется в тех случаях, когда символьный атрибут может иметь одновременно несколько значений или ни одного;

– арифметические правила предназначены для вычисления значений числовых атрибутов, а также получения коэффициентов определённости утверждений;

– логические правила предназначены для вычисления коэффициентов определённости утверждений по формулам нечёткой логики. При этом значение логического выражения (в условии правила) присваивается коэффициенту определённости целевого утверждения правила;

– байесовские правила предназначены для вычисления коэффициентов определённости тех утверждений, об истинности которых можно судить по выполнению ряда факторов (симптомов), имеющих разную значимость.

Для определения значения одной цели разработчик экспертной системы может задавать несколько правил, образующих в модели упорядоченный список правил вывода данной цели.

Сценарий консультации представляет собой последовательность предложений, каждое из которых может иметь условие применимости. В рамках каждого предложения возможно выполнение одного из следующих действий:

– вывести значение цели;

– выдать сообщение пользователю;

– выдать сообщение внешней программе;

– сбросить выведенные результаты (СБРОС);
– перейти к выполнению другого предложения;
– принять информацию от внешней программы;
– передать информацию о результатах решения внешней программе;
– создать контрольную точку консультации;
– загрузить контрольную точку;
– обратиться к подмодели, решающей некоторую частную подзадачу, передать ей параметры и получить выведенные в подмодели результаты;
– закончить консультацию с сообщением (СТОП).

Рассмотрим построение сети вывода на основе содержащихся в модели описаний правил и атрибутов. В процессе построения сети из числовых атрибутов, утверждений и правил строится сеть вывода, в явном виде включающая все связи между атрибутами и утверждениями, обусловленные правилами вывода. Сеть вывода образует граф с вершинами двух типов: первые вершины соответствуют простым целям; вторые вершины соответствуют простым правилам. Дуги представляют собой связи между простыми целями и простыми правилами. Особенность таких простых правил состоит в том, что применение одного влечёт применение остальных.

Стратегии управления в оболочке ЭКО характеризуются следующими моментами. В начале решения задачи выбирается первое предложение сценария, затем проверяется условие его применимости; если условие выполнено, выполняется указанное в нём действие. Если в ходе проверки возникает потребность в значении некоторой цели, то анализ условия приостанавливается и требуемое значение выводится из сети вывода. После обработки первого предложения сценария осуществляется переход к следующему предложению и т. д., пока не будет обнаружено действие СТОП или пока не будет исчерпан сценарий. В оболочке ЭКО используется стратегия обратного рассуждения от целей к данным в глубину: при рассмотрении некоторой цели делается попытка найти в сети вывода путь от вершин, представляющих исходные данные консультации, к вершине соответствующей выбранной цели. Путь считается найденным, если выполнены условия применимости всех правил, соответствующих дугам этого пути. В том случае, когда оказалось сразу несколько применимых правил, используется первое применимое правило. Значения целей вычисляются один раз и не могут быть изменены иначе, как по команде СБРОС.

Ввод знаний в базу знаний ЭКО осуществляется средствами диалогового редактора, предоставляющего:
– ЭКО;
– синтаксический и семантический контроль вводимой информации;
– тестирование и компиляцию моделей;
– навигацию по базе знаний;
– шаблоны ввода всех конструкций языка для представления знаний генерацию текстовых и гипертекстовых отчётов по базе знаний модели.

Помимо базы знаний разработчик экспертных систем может сформировать контекстно-зависимую помощь к приложению в виде иллюстрированного гипертекста.

Решение задач осуществляется в режиме консультации, при этом предоставляются следующие возможности:

– решение конкретной задачи на основе выбранной модели с формированием объяснений;
– просмотр информации о правилах в моделях;
– сброс значений всех целей либо значений всех выведенных целей
– (отмена всех значений, не являющихся исходными данными);
– получение трассы решения задачи;
– запись протокола консультации в файл и создание и загрузка контрольных точек.

Решение осуществляется в ходе диалога экспертной системы с пользователем. На экран выдаются сообщения в соответствии со сценарием консультации, а также задаются вопросы, описанные в применяемых правилах.

Контрольные вопросы

1. Понятие экспертной системы и ее отличие от другой программы.
2. Мотивация интереса пользователей к ЭС.
3. Характеристика неформализованной задачи.
4. Классификация ЭС.
5. Классификация инструментальных средств разработки ЭС.
6. Назначение основных модулей ЭС.
7. Понятие инженерии знаний и инженера знаний.
8. Примеры создания ЭС.
9. Характеристика ЭС ЭКО.
10. Отличие ЭС от системы, основанной на знаниях.
11. Декларативные знания системы MYCIN.

3. РАЗРАБОТКА ИНСТРУМЕНТАЛЬНОЙ ПРОДУКЦИОННОЙ ЭКСПЕРТНОЙ СИСТЕМЫ

3.1. Концепция построения инструментальной экспертной системы

Вопросы проектирования экспертных систем достаточно полно освещены в литературе [1, 3, 4, 5, 9] и на сайте www.mari-el.ru/mmlab/home/AI/.

При создании экспертных систем одним из основных критериев является трудоемкость создания экспертных систем. В условиях сжатых сроков разработки и ограничения на ресурсы проектирование ЭС на базе инструментальной системы является наиболее предпочтительным вариантом решения проблемы.

Инструментальная экспертная система предназначена для создания проблемно-ориентированных или предметно-ориентированных экспертных систем и демонстрации возможностей, которые предоставляют продукционные правила при их создании.

Часто к инструментальным экспертным системам предъявляют требование по обеспечению работы в условиях неопределённости и неполноты информации. Сведения о поставленной задаче могут быть неполными, и отношения между объектами предметной области могут быть приближёнными. Например, может не быть полной уверенности в наличии у пациента некоторого симптома или в том, что данные, полученные при измерении, верны. В таких случаях необходимы рассуждения с использованием вероятностного подхода.

Психологические исследования процессов принятия решений человеком показали, что, рассуждая, человек использует правила, аналогичные продукциям, которые называются «условие→действие».

При использовании продукционной модели база знаний состоит из набора правил. Основным модулем любой ЭС является модуль машины логического вывода, который управляет перебором правил, принятием решения и объяснением хода рассуждения [1, 3, 4].

Машина логического вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр правил из базы знаний, во-вторых, применения правил.

Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила оказывается недостаточно данных.

В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой программу и включает в себя два компонента: первый реализует собственно вывод, другой управляет этим процессом.

Действие компонента вывода основано на применении правила, называемого modus ponens: «*если известно, что истинно утверждение A и существует правило вида «ЕСЛИ A, ТО B», тогда утверждение B также истинно».*

Правила срабатывают, когда находятся истинные факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение.

Компонент вывода должен функционировать даже при недостатке информации. Полученное решение может и не быть точным, однако система не должна останавливаться из-за того, что отсутствует какая-либо часть входной информации.

Управляющий компонент определяет порядок применения правил и выполняет четыре функции:

– сопоставление – образец правила сопоставляется с имеющимися фактами;

– выбор – если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта);

– срабатывание – если образец правила при сопоставлении совпал с какими-либо фактами из рабочей памяти, то правило срабатывает;

– действие – рабочая память подвергается изменению путём добавления в неё заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действие, то оно выполняется (как, например, в системах обеспечения безопасности информации).

Интерпретатор продукций работает циклически [1]. В каждом цикле он просматривает все правила, чтобы выявить те посылки, которые совпадают с известными и истинными на данный момент фактами. После выбора правило срабатывает, его заключение заносится в рабочую память, а затем цикл повторяется сначала.

В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то такая ситуация называется конфликтной. Интерпретатор выполняет разрешение конфликтов путём выбора по определённому критерию единственного правила, которое срабатывает в данном цикле. Цикл работы интерпретатора схематически представлен на рис. 3.1.

Информация из рабочей памяти последовательно сопоставляется с посылками правил для выявления успешного сопоставления. Совокупность отобранных правил составляет так называемое конфликтное множество. Для разрешения конфликта в интерпретаторе разработчиком предусматривается блок разрешения конфликтов, который выбирает единственное правило, после чего оно срабатывает. Это выражается в занесении фактов, образующих заключение правила, в рабочую память или в изменении критерия выбора конфликтующих правил. Если же в заключение правила входит название какого-нибудь действия, то оно выполняется. Способы разрешения конфликтов достаточно подробно рассмотрены в [3]. По мнению автора книги, [3], в настоящее время не существует эффективного способа разрешения конфликтов и выбор способа полностью определяется предметной областью, для которой разрабатывается ЭС.

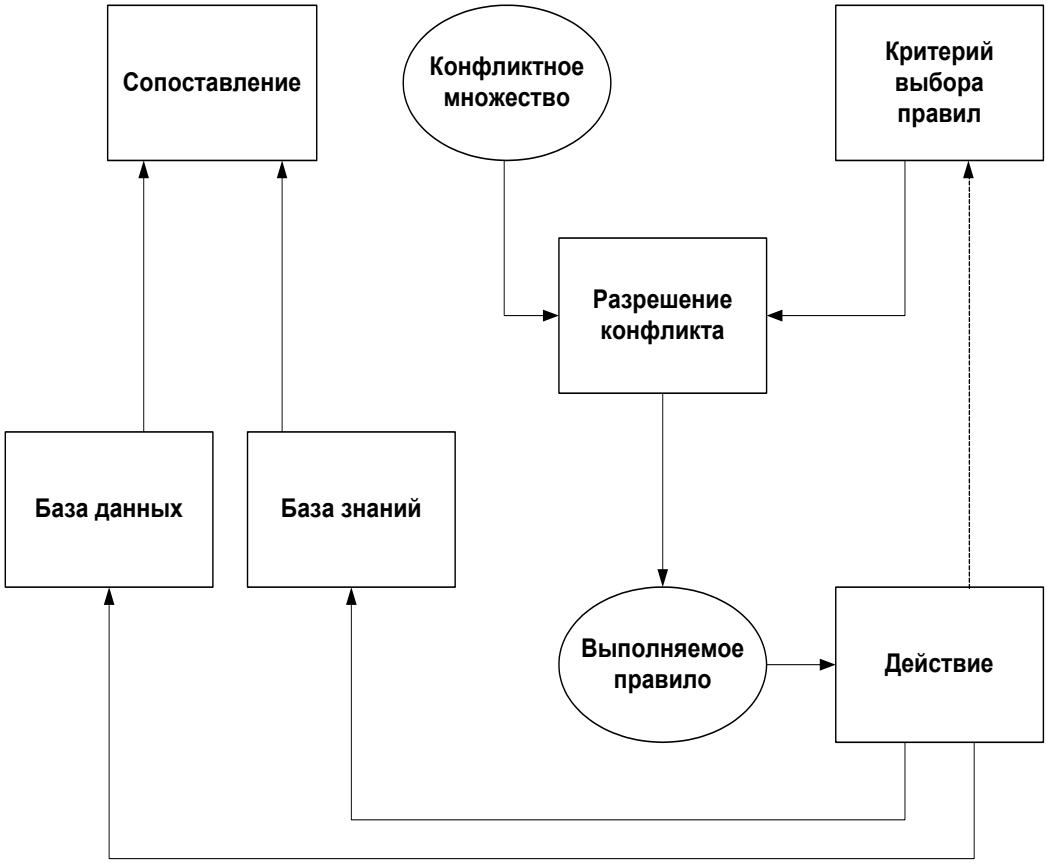


Рис. 3.1. Цикл работы интерпретатора машины логического вывода

При использовании теории приближенных рассуждений сравнительно корректно решается проблема функционирования ЭС в условиях неопределенности [1, 3, 4, 12] и отпадает необходимость реализации блока разрешения конфликтов.

Стратегии управления выводом – это порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно включаются в механизм вывода. Поэтому в большинстве случаев инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию.

При разработке стратегии управления выводом важно определить два вопроса.

1. Какую точку в пространстве состояний принять в качестве исходной? От выбора этой точки зависит и метод осуществления поиска – в прямом или обратном направлении;

2. Какими методами можно повысить эффективность поиска решения? Эти методы определяются выбранной стратегией перебора: в глубину или в ширину, по подзадачам или иначе.

Прямой и обратный вывод. При *обратном порядке вывода* вначале выдвигается некоторая гипотеза, а затем механизм вывода как бы возвращается назад, переходя к фактам, пытаясь найти те, которые подтверждают гипотезу. Если она оказалась правильной, то выбирается следующая гипотеза, детализи-

рующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчинённой гипотезы. Вывод такого типа называется управляемым целями, или управляемым консеквентами.

Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

В системах с *прямым выводом* по известным фактам отыскивается заключение, которое из этих фактов следует. Если такое заключение удаётся найти, то оно заносится в рабочую память. Прямой вывод часто называют выводом, управляемым данными, или выводом, управляемым антецедентами.

Например, имеется фрагмент базы знаний из двух правил.

Правило 1 – ЕСЛИ «отдых – летом» И «человек – активный», ТО «ехать – в горы».

Правило 2 – ЕСЛИ «любит – солнце», ТО «отдых – летом».

Предположим, в систему введены истинные факты – «любит – солнце» И «человек – активный».

Прямой вывод – исходя из фактических данных позволяет получить рекомендацию.

Первый проход

Шаг 1. Попробуем правило 1, не работает (не хватает данных «отдых – летом»).

Шаг 2. Попробуем правило 2, работает, в базу поступает факт «отдых – летом».

Второй проход

Шаг 3. Попробуем правило 1, работает, активизируется цель «ехать – в горы», которая и выступает как совет, который дает экспертная система.

Обратный вывод позволяет подтвердить выбранную цель при помощи имеющихся правил и данных.

Первый проход

Шаг 1. Цель – «ехать – в горы»: попробуем правило 1 – не хватает данных «отдых – летом», они становятся новой целью и выполняется поиск правила, у которого цель находится в левой части.

Шаг 2. Цель – «отдых – летом»: правило 2 подтверждает цель и активирует её.

Второй проход

Шаг 3. Попробуем правило 1, подтверждается искомая цель.

Идея двунаправленного поиска основывается сразу на двух стратегиях – прямого поиска от корневой вершины и обратного от целевой вершины. Процесс поиска прекращается, когда оба эти процесса встречаются.

Принцип двунаправленного вывода отражён в методе деления дизъюнктов.

В системах, база знаний которых насчитывается сотни правил, желательным является использование стратегии управления выводом, позволяющей минимизировать время поиска решения и тем самым повысить эффективность вы-

вода. К числу таких стратегий относятся: «поиск в глубину», «поиск в ширину», разбиение на подзадачи и альфа-бета алгоритм.

При *«поиске в глубину»* в качестве очередной подцели выбирается та, которая соответствует следующему, более детальному уровню описания задачи. Например, диагностирующая система, сделав на основе известных симптомов предложение о наличии определённого заболевания, будет продолжать запрашивать уточняющие признаки и симптомы этой болезни до тех пор, пока полностью не опровергнет выдвинутую гипотезу.

При *«поиске в ширину»*, напротив, система вначале проанализирует все симптомы, находящиеся на одном уровне пространства состояний, даже если они относятся к разным заболеваниям, и лишь затем перейдёт к симптомам следующего уровня детальности.

Разбиение на подзадачи подразумевает выделение подзадач, решение которых рассматривается как достижение промежуточных целей на пути к конечной цели. Примером, подтверждающим эффективность разбиения на подзадачи, является поиск неисправностей в компьютере: сначала выявляется отказавшаяся подсистема (питание, память и т. д.), что значительно сужает пространство поиска. Если удаётся правильно понять сущность задачи и оптимально разбить её на систему иерархически связанных целей, то можно добиться того, что путь к её решению в пространстве поиска будет минимальным.

Альфа-бета алгоритм позволяет уменьшить пространство состояний путём удаления ветвей, неперспективных для успешного поиска. Поэтому просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные направления исключаются.

3.2. Основы теории приближенных рассуждений

Существуют следующие проблемы, которые необходимо учитывать в понятии неопределенности систем логического вывода [4].

Как количественно выразить степень определенности при установлении истинности (или ложности) некоторой части данных?

Как отразить степень поддержки заключения конкретной посылкой?

Как использовать совместно несколько посылок, влияющих на заключение?

При разработке ЭС могут использоваться методы точного вероятностного и приближенного рассуждений [1, 3, 4].

Использование рассуждений на основе вероятностей становится все более трудным и неудобным. По этой причине многие ЭС применяют специальные методы приближенных рассуждений. Именно такой механизм использован в экспертных системах EMYCIN, FUZZYNET, которые демонстрируют эффективность многоступенчатых приближенных рассуждений. Импликация с одной посылкой имеет вид «ЕСЛИ (e), ТО (c)».

Обычное правило комбинирования, позволяющее вычислить коэффициент определенности заключения в случае, когда известен коэффициент опреде-

ленности посылки, лежащей в его основе, и связи в импликации, записывается следующим образом:

ct (заключение) = ct (посылка) · ct (импликация).

Если в примере истинность посылки определена с уверенностью 0,8, а импликация выполняется в большинстве случаев, но не всегда (например, с коэффициентом определенности 0,9), тогда коэффициент определенности заключения вычисляется следующим образом:

ct (заключение) = 0,8 · 0,9 = 0,72.

Логические комбинации посылок в одном правиле. Основной вычислительный прием, который можно использовать для нахождения коэффициента определенности заключения, сводится к следующему:

ct (заключение) = ct (посылка) · ct (импликация).

Прежде всего нужно суметь оценить коэффициенты определенности посылок. Посылкой считаются все логические выражения в правиле между словами «ЕСЛИ» и «ТО». Исключение составляет простая импликация, в которой выражение состоит из атомарных посылок, каждая из которых имеет свой коэффициент определенности.

Посылки могут быть связаны между собой логическими операциями, например, ЕСЛИ (e1 ИЛИ (e2 И e3)), ТО (c) или

ЕСЛИ (e1 И e2 И ((НЕ e3) ИЛИ e4)) , ТО (c).

Очевидно, требуется некоторый способ оценки коэффициентов определенности этих сложных форм в понятиях их отдельных компонент. Подход заключается в том, чтобы отбросить все сложные выражения и считать все правила простыми. Есть несколько тривиальных процедур для сведения коэффициентов определенности простых логических комбинаций в одно число.

Простой логической комбинацией является конъюнкция (И) между двумя элементарными свидетельствами. Импликация выглядит так:

ЕСЛИ (e1 И e2) , ТО (c)

Коэффициент определенности посылки равен коэффициенту определенности наименее надежной из посылок, т. е.

ct (e1 И e2) = min [ct (e1), ct (e2)].

Другой простой формой является правило, в котором используется дизъюнкция (ИЛИ), связывающая две части свидетельств: ЕСЛИ (e1 ИЛИ e2) , ТО (c).

Общее правило комбинирования, по которому вычисляется коэффициент определенности посылки, заключается в том, что коэффициент определенности дизъюнкции равен коэффициенту определенности ее сильнейшей части

ct (e1 ИЛИ e2) = max [ct (e1), ct (e2)].

Хотя правила иногда и записываются с помощью дизъюнкции, но если есть выбор, то принято разбивать дизъюнкцию на две части, например:

ЕСЛИ (e1) , ТО (c),

ЕСЛИ (e2) , ТО (c).

Использование двух правил вместо дизъюнкции требует механизм, определяющий коэффициент определенности заключения при поддержке этих правил. Выбор способа представления правил определяется экспертом.

Поддержка одного заключения множеством правил. Например, используются два правила и оба они поддерживают одно и то же заключение:

правило 1: ЕСЛИ (e1), ТО (c) ct (заключение) = 0,9;

правило 2: ЕСЛИ (e2), ТО (c) ct (заключение) = 0,8.

Допустим, обе посылки верны, тогда можно вычислить вероятность заключения для каждого правила по отдельности.

Предположим, что переменная ctotal представляет общий коэффициент определенности заключения, полученный использованием всех поддерживающих его правил. Можно предложить много различных комбинаций процедур. Рассмотрим простой и эффективный механизм, приведенный в [12]: ctotal = коэффициент определенности из правила 1 +

+ коэффициент определенности из правила 2 минус произведение (коэффициента определенности из правила 1) и· (коэффициента определенности из правила 2).

Здесь два коэффициента определенности преобразуются в один коэффициент, который всегда меньше единицы. Подставив числа, заданные в примере, получим

ctotal = 0,9 + 0,8 – (0,9) · (0,8) = 0,98.

Рассмотренный принцип можно распространить на случай из нескольких правил, поддерживающих одно заключение, где для каждого правила существует своя вероятность. Например, есть три правила со следующими коэффициентами определенности.

Правило 1: ЕСЛИ (e1), ТО (c) ct (заключение) = ct1.

Правило 2: ЕСЛИ (e2), ТО (c) ct (заключение) = ct2.

Правило 3: ЕСЛИ (e3), ТО (c) ct (заключение) = ct3.

Совокупный коэффициент определенности заключения с учетом всей возможной поддержки может быть вычислен следующим образом:

ctotal = ct1 + ct2 +ct3 – ct1 · ct2 – ct1 · ct3 – ct1 · ct3 + ct1 · ct2 · ct3.

Суть заключается в формировании произведений из базовых коэффициентов определенностей правил и в сложении и вычитании этих произведений соответствующим образом. Каждая двойная комбинация коэффициентов определенностей вычитается, тройная – прибавляется, четвертная – вычитается и т.п. до тех пор, пока все правила не будут использованы совместно.

Несколько правил, используемых последовательно. Механизм дополнения – это другой способ вычисления коэффициента определенности заключения, поддерживаемого несколькими правилами импликации. Он используется в том случае, когда сведения о разрешенных к применению правилах поступают последовательно, а не одновременно. Например, если система задает пользователю вопросы, то использование новых правил будет происходить по очереди.

Например, известно, что заключение поддерживается двумя правилами со следующими коэффициентами определенности.

Правило 1: ЕСЛИ (e1), ТО (c) ct (заключение)= ct1.

Правило 2: ЕСЛИ (e2), ТО (c) ct (заключение)= ct2.

При применении двух правил совокупный коэффициент определенности ctotal = ct1 + ct2 – ct1 · ct2.

Теперь предположим, что появилось третье правило, поддерживающее тоже заключение:

правило 3: ЕСЛИ (e3), ТО (c) ct (заключение) = ct3.

Если все, что получено из предыдущего рассмотрения, входит в переменную ctotal и считается, что ct3 может войти в рассуждения на общих основаниях, то можно использовать стратегию дополнения для формирования измененной оценки коэффициента определенности заключения:

cnewtotal = ct3 + ctotal – ct3 · ctotal.

Перемножив все компоненты этой формулы, получается следующий результат:

cnewtotal = ct3 + (ct1 + ct2 – ct1 · ct2) – ct3 · (ct1 + ct2 – ct1 · ct2) =

= ct1 + ct2 + ct3 – ct1 · ct2 – ct1 · ct3 – ct2 · ct3 + ct1 · ct2 · ct3.

Очевидно, что это в точности тот же вывод, который был сделан ранее, комбинируя свидетельства, полученные одновременно.

Данный вывод, если его обобщить, имеет два важных приложения:

– при использовании механизма дополнения порядок поступления правил, поддерживающих заключение, не имеет значения;

– можно объединять коэффициенты определенности из поддерживающих импликаций последовательно по мере их поступления или сохранять информацию, а затем использовать ее всю сразу – результат от этого не меняется.

Практически сеть рассуждений меняется, как только поступают новые сведения. Поэтому сохранять нужно лишь совокупный коэффициент определенности для каждого заключения, что обеспечивает наиболее экономный способ поддержки информационного обеспечения ЭС.

Биполярные схемы для коэффициентов определенности. Прототипом систем, основанных на приближенных рассуждениях, являются MYCIN и ее прямой потомок EMYCIN. Эти системы используют механизм объединения коэффициентов определенностей, который был рассмотрен выше. Коэффициент определенности является грубым приближением к вероятности. В EMYCIN [4, 12] используется интервал от минус 1 до 1, так что это не может быть вероятностью.

Границы интервала обозначают следующее: «1» – система в чем-то полностью определена, «0» – у системы нет знаний об обсуждаемой величине, «минус 1» – высказанная гипотетическая посылка или заключение абсолютно неверно. Промежуточные величины отражают степень доверия или недоверия к указанным ситуациям. Все описанные процедуры рассуждений применимы для коэффициентов определенности, задаваемых в этих более широких границах. Однако нужно учесть, что, когда с помощью правил вы находится максимум или минимум двух величин, нужно помнить про знаки. Например, значение 0.1 должно рассматриваться как более крупная величина, чем минус 0.2.

Полная реализация идеи биполярных коэффициентов определенности требует сделать два обобщения для развиваемого нами вычислительного механизма. Во-первых, отсутствует навык работы с отрицанием атомарных посылок. Например, в посылке имеется частица «НЕ»: ЕСЛИ (e1 И (НЕ e2)), ТО (c) .

В этом случае нужно только считать (не e2) атомарным утверждением. Можно дать ему новое имя, например, e3, но какой коэффициент определенности следует приписать этому новому свидетельству? Обычно коэффициент определенности задан для e2. Для вычисления же коэффициента определенности (НЕ e2) достаточно просто поменять знак: ct (НЕ e) = – ct (e).

Этот факт является следствием применения биполярной меры определенности. В любом случае коэффициент определенности для отрицания посылки всегда может быть легко найден, а потом использован в различных манипуляциях.

Особый интерес представляет процедура получения композиции коэффициентов определенностей в условиях поддержки двумя правилами одного и того же заключения.

Необходимо сделать следующее: если оба коэффициента определенности положительны:

$$ctotal = ct1 + ct2 - ct1 \cdot ct2, \quad (3.1)$$

если оба коэффициента определенности отрицательны:

$$ctotal = ct1 + ct2 + ct1 \cdot ct2. \quad (3.2)$$

Когда отрицателен только один из коэффициентов, то

$$ctotal = \frac{ct1 + ct2}{1 - \min\left[|ct1|, |ct2|\right]}, \quad (3.3)$$

где $|ct1|$, $|ct2|$ – модули коэффициентов уверенности.

В том случае, если один коэффициент определенности равен 1, а другой минус 1, то ctotal = 0.

Когда два правила с небольшими коэффициентами определенности поддерживают одно заключение, коэффициент определенности заключения возрастает. Если же знаки не совпадают, то результат определяется «сильнейшим» коэффициентом, но влияние его несколько ослабляется.

Применение биполярных коэффициентов определенности может привести к нереальным результатам, если правила сформулированы неточно. Работая с одним правилом вывода, следует не забывать, что всегда используется соотношение ct (заключение) = ct (посылка) · ct (импликация).

Все правила попадают в одну из этих двух очень важных категорий. Правила первой категории будем называть *обратимыми*. Одной из характеристик такого правила является его применимость к любому значению коэффициента определенности, которое может быть связано с посылкой. Правила второй категории считаются *необратимыми*. Эти правила «работают» только при положительных значениях посылки. Если же ее значение отрицательно, правило применять нельзя (оно не имеет здравого смысла). В этом случае необратимое правило с отрицательным коэффициентом уверенности отбрасывается.

При создании правил всегда следует проверять их на обратимость, имитируя отрицание посылки и заключения, и определяя сохраняет ли правило здравый смысл.

Многоступенчатые рассуждения. До сих пор окончательное заключение отделялось от посылки одним шагом рассуждений. Более типичной является другая ситуация – сеть, в которой окончательные рассуждения отдельны от ба-

зы посылок большим числом промежуточных шагов. Такие рассуждения называются многоступенчатыми.

Можно привести следующий пример многоступенчатых рассуждений на базе правил диагностики заболевания. Число в правой части каждого правила указывает коэффициент определенности для конкретной импликации.

Если у вас грипп, и вы находитесь в уязвимом возрасте,
то вызовите врача ct (импликация) = 0,9
Если у вас острый фарингит,
то вызовите врача ct (импдикация) = 1,0
Если у вас простуда,
то ложитесь в постель и примите аспирин ct (импдикация) = 0,4
Если у вас грипп, и вы не находитесь в уязвимом возрасте,
то ложитесь в постель и примите аспирин ct (импликация) = 0,4
Если у вас лихорадка и болят мышцы,
то это грипп ct (импликация) = 0,7
Если у вас насморк, мышечные боли и нет лихорадки,
то это простуда ct (импликация) = 0,7
Если у вас в горле нарывы и есть лихорадка,
то это острый фарингит ct (импликация) = 0,8
Если вам меньше 8 или больше 60 лет,
то вы находитесь в уязвимом возрасте ct (импликация) = 0,7

Теперь вы можете просмотреть правила и в зависимости от конкретных симптомов заболевания решить, обратиться ли к врачу или достаточно лечь в постель и принять аспирин. Вы даже можете использовать комбинацию изученных правил для определения коэффициента определенности, соответствующей каждому из возможных результатов, а потом выбрать тот, который имеет наибольший коэффициент определенности. Однако данная форма представления правил удобна для компьютера, но не для человека.

Для обсуждения многоступенчатого рассуждения правила удобно преобразовывать в другую форму, позволяющую более отчетливо представить соответствующие коэффициенты. Любая система может быть отображена графически. Она называется сетью вывода и имеет вид графика с указанными связями между правилами. На графике также отчетливо видны все возможные поддерживающие структуры промежуточных рассуждений, находящиеся ниже любого более высокого уровня заключения. Такая сеть логического вывода строится, чтобы придать правилам конкретную форму (рис. 3.2).

Сеть показывает возможности многоступенчатых рассуждении в задаче в более удобном виде, чем просто список утверждений. Здесь сделана попытка представить явным образом все шаги рассуждений для некоторой гипотетической ситуации, когда пациент имеет какое-то заболевание, родственное гриппу, и хочет получить рекомендации.

В диаграммах подобного типа используются некоторые стандартные приемы, которые надо знать, чтобы уметь их прочесть (рис. 3.3). Сеть вывода

и множество взаимосвязанных импликаций – это одно и то же. Обе формы содержат одинаковый объем информации.

В правилах, составляющих сеть вывода, могут быть и позитивные, и негативные утверждения. В рассмотренном выше примере встретились две фразы: «есть лихорадка» и «нет лихорадки».

Поскольку здесь речь идет об одном и том же, эти фразы удобно зафиксировать в одном узле сети вывода. Для правила, где фраза появляется в негативной форме, связь отмечается перечеркивающей полосой, проходящей через узел, отображающий лихорадку. Там, где она появляется в позитивной форме, связь имеет обычный вид.

Биполярные схемы позволяют получить коэффициент определенности негативной формы путем смены знака на противоположный.

При изображении связки «И» используется сплошная дуга (например, для вершин c1, c2, c4), а при изображении связки «ИЛИ» – штриховая дуга (например, для вершины c3). Связка «НЕ» изображается короткой чертой на выбранной дуге (например, e3-c1).

Пример, иллюстрирующий распространение коэффициентов определенности в сети, приведен на рис. 3.2. Коэффициент определенности отмечен справа от каждого узла. Отдельные первоначальные посылки, расположенные в нижней части дерева, показывают коэффициенты определенностей, которые были получены при задании необходимых вопросов и при получении данных из внешнего мира. В исходном состоянии все внутренние узлы имеют коэффициенты определенности, равные нулю, так как рассуждения пока не проводились.

Под каждым внутренним узлом стоит число, отражающее коэффициент определенности импликации, поддерживающей конкретный узел. Рядом с коэффициентом определенности импликации записывается признак rev (для обратимых правил) или признак nrev (для необратимых правил), что обозначает, будет импликация использоваться как обратимое или как необратимое правило. Обратимое правило можно применять всегда, а необратимое нужно удалить из сети, если коэффициент определенности посылки для этого правила становится отрицательным. Вычисление коэффициента определенности заключения может потребовать выполнения нескольких шагов: могут добавляться «И» «ИЛИ», «НЕ». В каждом конкретном случае, пока не будет закончена вся эта предварительная работа, нельзя с уверенностью сказать, применимо ли правило.

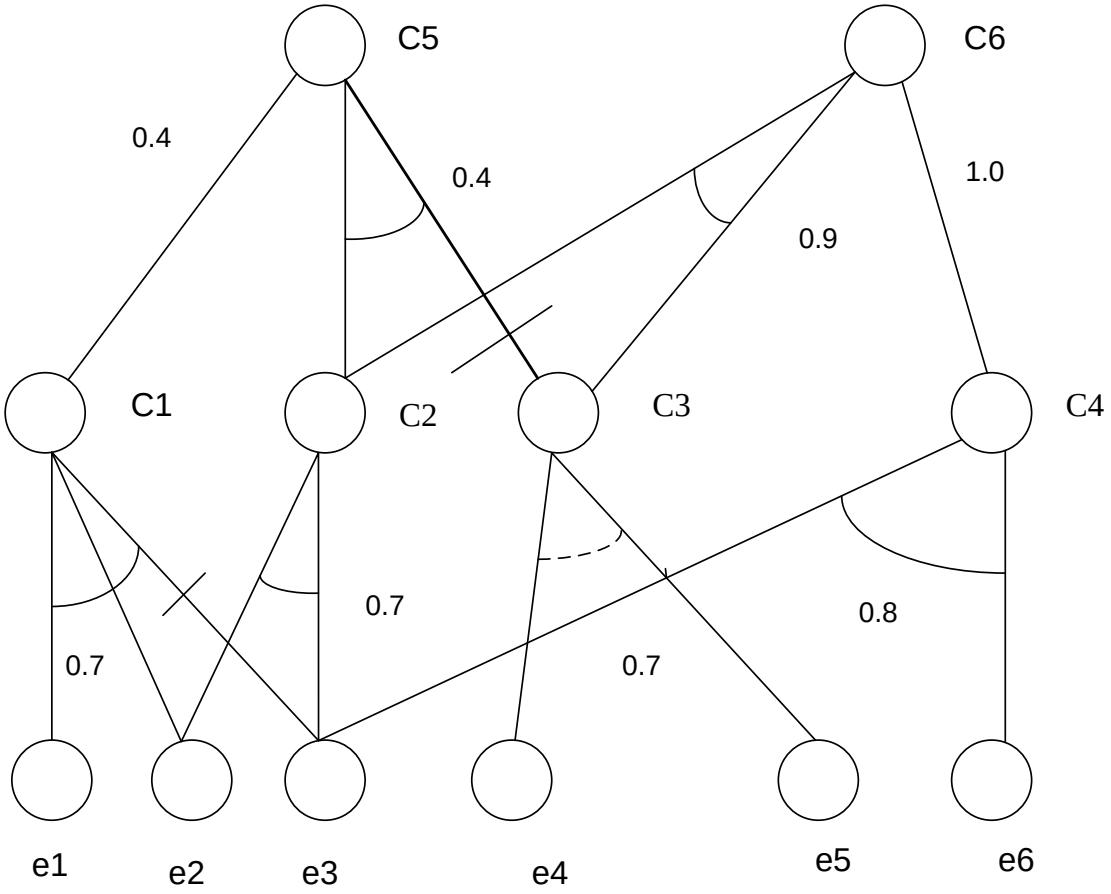
Иногда правило включает отрицание некоторой посылки или заключения. На диаграммах сети вывода определенности всегда показаны для посылок или заключения до применения отрицания.

Пример расчета коэффициентов уверенности в сети логического вывода, представлен на рис. 3.3. Предлагаются следующие правила:

ЕСЛИ (e1), ТО (c1)	ct (импликация) = 0,8 (nrev) ,
ЕСЛИ (e2), ТО (c2)	ct (импликация) = 0,9 (rev) ,
ЕСЛИ (e3), ТО (c2)	ct (импликация) = 0,7 (rev) ,
ЕСЛИ (e4), ТО (c3)	ct (импликация) = 0,6 (nrev),
ЕСЛИ (НЕ e5), ТО (c3)	ct (импликация) = 0,5 (nrev) ,

ЕСЛИ (c2 И c3), ТО (c4) ct (импликация) = 0,9 (rev),
ЕСЛИ (c1 ИЛИ c4), ТО (c5) ct (импликация) = 0,8 (nrev) .

Следует начать с основания и идти вверх по дереву, чтобы оценить, что же произошло.



e1 – насморк; e2 – мышечные боли; e3 – лихорадка; e4 – возраст менее 8 лет; e5 – возраст более 60 лет; e6 – нарывы в горле; c1 – простуда; c2 – грипп; c3 – уязвимый возраст; c4 – острый фарингит; c5 – лечь в постель и принять аспирин; c6 – вызвать врача

Рис. 3.2. Медицинские правила в виде сети логического вывода

В сети содержатся импликации разных типов: простые, И, ИЛИ, импликации с отрицаниями, а также обратимые и необратимые правила. Рассуждения начинаются с основания дерева, где все известно, а затем с помощью правил импликации находятся коэффициенты определенности для узлов, поддерживающих нижний информационный уровень. Этот процесс продолжается последовательно до тех пор, пока не будет найден коэффициент определенности для каждого заключения.

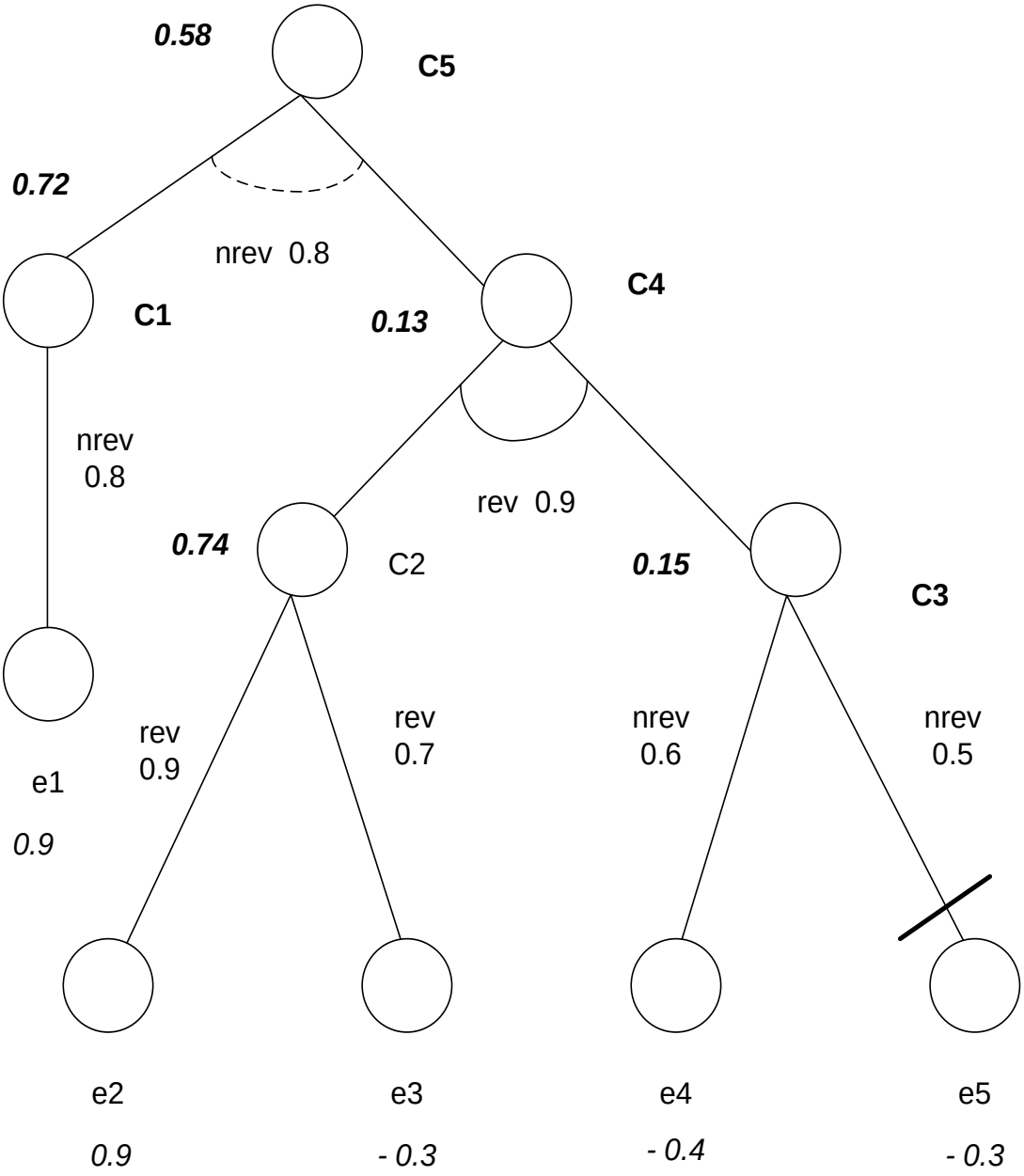


Рис. 3.3. Пример сети логического вывода

Коэффициент определенности c1 может быть вычислен следующим образом: ct (заключение c1) = 0.8 · 0.9 = 0.72.

Это простая необратимая импликация, но, поскольку коэффициент определенности посылки позитивен, правило можно применять.

Для вычисления коэффициента определенности c2 задействованы два правила и оба используются без ограничений, так как они обратимы. Правило слева даст оценку коэффициента определенности для вершины графа c2 ct (заключение c2) = 0.9 · 0.9 = 0.81. Правило справа даст вторую оценку для вершины графа c2 ct (заключение c2) = – 0.3 · 0.7 = –0.21.

Здесь приведены два поддерживающих правила, дающих оценку коэффициента определенности с противоположными знаками, поэтому для окончательного ответа применяется формула (3.3):

$$\text{ct (заключение c2)} = \frac{0.81 + (-0.21)}{1 - 0.21} = 0.74.$$

Для c3 имеется два правила. Правило, связанное с левым поддеревом, не применяется, так как оно необратимо и коэффициент определенности посылки отрицателен. Правило, связанное с правым поддеревом, есть простая импликация. Она необратима и содержит отрицательную посылку. Что нужно сделать? Правило утверждает:

ЕСЛИ (НЕ e5), ТО (c3) ct(импликация) = 0,5 (nrev).

Коэффициент определенности e5 равен минус 0,3. Так как он негативен, то коэффициент определенности посылки в правиле равен 0,3. Коэффициент определенности всего предложения, поддерживающего посылку использует процедуру, предназначенную для простой импликации: ct (заключение c3) = 0,3· 0,5=0,15.

Импликация, поддерживающая c4, включает конъюнкцию (связку И) посылок. Коэффициент определенности посылки определяется следующим образом: ct (свидетельства) =min (0,15; 0,74) = 0,15.

Поскольку правило обратимо, можно использовать посылку в любом интервале определенности. Используя этот результат, вычисяется коэффициент определенности для c4: ct (заключение c4) =0,15 · 0,9=0,13.

Теперь пройден путь вверх по дереву до того места, где можно судить об узле верхнего уровня. Здесь задействовано одно правило, в котором посылки разделены с помощью связки ИЛИ, поэтому

ct (свидетельства) =max (0,72; 0,13) =0,72.

Правило необратимо, но коэффициент определенности посылки позитивен и поэтому вычисление возможно.

Последнее звено в цепочке рассуждений (коэффициент определенности для узла высшего уровня) вычисляется по формуле

ct (заключение c4) = 0,72 · 0,8 = 0,58.

Полученную сеть логического вывода можно использовать для механизма объяснения выбора конкретного заключения.

3.3. Характеристика инструментальной экспертной системы

Состав инструментальной системы включает модуль пользовательского интерфейса, модуль работы с текстом, модуль машины логического вывода, модуль базы знаний.

Инструментальная экспертная система «ANIES» является учебной программой, предназначенной для демонстрации возможностей, которые предоставляют продукционные правила при логическом выводе.

В процессе разработки базы знаний формируется файлы ЭС для конкретной предметной области. Все файлы хранятся в текстовом формате.

Одним из основных управляющих элементов интерфейса является главное меню программы, которое состоит из горизонтального меню, содержащего имена основных групп команд, и выпадающих подменю, позволяющих выбрать

конкретную команду или режим работы. Такие пункты горизонтального меню, как «Файл», «Правка», «Помощь», являются стандартными для программ. Они содержат набор команд для работы с файловой системой, облегчения редактирования текста. Используя текстовый редактор, разработчик участвует в создании базы знаний с использованием продукционных правил «IF-THEN-ELSE».

Для распараллеливания работы при создании проекта ЭС можно разрабатывать разными специалистами разделы БЗ с последующим их объединением в один проект (команда «Объединение файлов в проект»).

В меню «Путь исполнения правил» устанавливается последовательность вызова правил из БЗ. По команде «Запуск» выполняется автоматическая процедура логического вывода.

По команде «Трассировка» выполняется процедура логического вывода только с остановом после вызова одного правила с анализом его выполнения. При помощи пунктов меню «Результаты» пользователь может анализировать решение задачи.

База знаний содержит правила, необходимые для логического вывода, а также факты из предметной области, введённые пользователем или выведенные машиной логического вывода.

После запуска ЭС пользователь вводит ответы на вопросы, формируемые системой с указанием коэффициента уверенности в диапазоне [−1; 1]. Отвечая на один вопрос, пользователь может указать несколько ответов или ни одного. Система, используя механизм логического вывода, производит подсчет коэффициентов уверенности всех заключений и отображает ранжированный перечень гипотез. При желании пользователь может просмотреть ход срабатывания правил экспертной системы в виде протокола решения или в режиме трассировки.

Некоторые пункты меню продублированы кнопками быстрого управления. Разработчику экспертной системы предлагается использовать панели инструментов: гипотез, параметров, ключевых слов – при формировании базы правил. Это позволяет значительно сократить количество ошибок и трудоемкость разработки ЭС.

При написании правил можно использовать обычный режим – режим редактирования (копировать, вырезать, вставить, вернуться на шаг назад).

```
NAME Правило1
 IF на_занятиях неусидчив
   THEN темперамент холерик [0,1]
 ELSE
   IF на_занятиях энергичен
     THEN темперамент сангвиник [0,1]
  ELSE
   IF на_занятиях спокоен
     THEN темперамент флегматик [0,1]
       ELSE темперамент меланхолик [0,1]
END
```

Синтаксис формируемых правил:

– все правила «IF-THEN-ELSE» и «CASE» должны заканчиваться ключевым словом «END»;

– при написании гипотез, параметров, наименований правил вместо пробела необходимо использовать знак подчёркивания с целью распознавания окончания;

– при составлении циклов «IF-THEN-ELSE» необходимо проставлять все ключевые слова: IF, THEN, ELSE.

Создание и редактирование базы знаний ЭС. Сначала создается новый проект ЭС с помощью команды системы меню «Файл / Новый проект», а затем формируется новый файл базы знаний (БЗ) пользователя. Ранее созданный и сохраненный файл БЗ файл базы знаний можно загрузить с помощью команды «Файл / Открыть существующую БЗ». Пункты меню «Файл / Сохранить» и «Файл / Сохранить как» предназначены для сохранения БЗ пользователя.

Пользователю предлагается использовать следующие функции системы:

– разбиение базы знаний на разделы при создании проекта или при редактировании базы знаний;

– слияние нескольких ранее созданных баз знаний с помощью команды меню «Файл / Объединение файлов в проект». Файлы с других компьютеров должны быть переписаны в текущий каталог системы.

– *Редактор базы знаний* предусматривает несколько режимов работы:

– работа с текстом;

– копирование, удаление, вставка, перемещение блоков;

– вызов с помощью правой кнопки мыши контекстное меню с перечнем параметров и гипотез.

Подготовка баз знаний с помощью данного текстового редактора заключается в последовательном выполнении ряда этапов:

– ввод разделов и правил БЗ;

– редактирование БЗ;

– открытие ранее разработанного файла БЗ;

– отладка правил БЗ по заранее созданным примерам эксперта;

– сохранение файла БЗ на магнитном диске.

После описания ЭС на продукционных правилах пользователь может запустить ЭС на выполнение. При запуске происходит чтение файла БЗ пользователя.

*Выполнение экспертной системы***.** Данный режим работы программы заключается в последовательном выполнении ряда этапов:

– указание пути исполнения правил;

– выбор метода логического вывода и метода поиска решений (прямой метод, обратный метод, «поиск в глубину», «поиск в ширину»);

– запуск на выполнение ЭС;

– анализ результатов работы ЭС путем просмотра протокола решений.

В инструментальной системе предусмотрены средства отладки: диагностики и трассировки. Средства трассировки позволяют пользователю следить за

действиями системы и анализировать полученный результат по каждому выбранному правилу. Средства диагностики используются для обнаружения ошибок в базе знаний.

Машина логического вывода выполняет следующие основные функции:

– формирование из непрерывного потока символов (извлекаемых из текстового редактора) лексем – минимальных единиц текста. Для реализации данной функции в структуру модуля работы с текстом был включён блок формирования лексем;

– идентификация лексем (блок идентификации) – определение, является ли лексема известным для оболочки экспертной системы словом или символом;

– контроль корректности исходного текста базы знаний;

– вывод пользователю сообщений о встретившихся ошибках и указание места ошибки в тексте базы знаний;

– поиск решения задачи исходя из правил, имеющихся в базе знаний и фактов с коэффициентами уверенности, полученных от пользователя;

– вычисление коэффициентов уверенности каждой гипотезы и параметра
в зависимости от степени доверия к исходным данным.

– Поиск решения можно разделить на несколько отдельных подзадач:

– выполнение процедуры диагностики;

– выполнение (решение) правила. Включает в себя проверку истинности условий правила, переход между ветвями правила в зависимости от знака коэффициента уверенности условия, вычисление коэффициента уверенности найденного заключения правила;

– подсчет результирующего коэффициента уверенности. В базе знаний могут присутствовать несколько правил, приводящих к одному и тому же заключению, поэтому нужно найти такие правила и решить их. Подсчёт коэффициентов уверенности производится в соответствии теории приближённых рассуждений [4, 12];

– задача поиска решения задачи заключается в поиске правил, к которым можно применить полученные заключения и решении найденных правил. Результаты решения правил вновь используются для поиска
и т. д., пока не будет найдено решение задачи.

Дополнительно в состав машины логического вывода была включена система управления режимом трассировки базы знаний.

Объяснение хода рассуждений системы выполняется путем протоколирования процесса рассуждений. Система записывает в файл информацию о каждом своем действии, а пользователь может просмотреть полученный протокол в любой момент решения задачи.

Тестирование базы знаний. Для проверки правильности написания базы знаний вводятся развитые средства распознавания ошибочных ситуаций, которые выдают пользователю сообщения об обнаруженных ошибках и указывают место ошибки в тексте базы знаний.

В блок диагностики включены специальные процедуры, которые автоматически исправляют некоторые виды ошибок пользователя в БЗ.

В отладке предусмотрены шесть основных механизмов управления выполнением программы:

– диагностика позволяет обнаружить и исправить некоторые классы ошибок в базе знаний;

– трассировка дает возможность покомандного выполнения выбранного правила;

– просмотр гипотезы дает возможность просмотра коэффициента уверенности гипотезы;

– просмотр параметра дает возможность просмотра коэффициента уверенности параметра;

– просмотр переменной дает возможность просмотра значения переменной;

– протокол решения дает возможность просмотра протокола решения задачи и выявить сработавшие правила.

Контрольные вопросы

1. Какие функции выполняет машина логического вывода?
2. Что такое продукционные правила?
3. Что представляет собой база знаний в продукционной ЭС?
4. Почему в зарубежных ЭС стали применять теорию приближенных рассуждений?
5. В чем состоит неопределенность продукционного правила?
6. Сформулируйте правило модус поненс.
7. Приведите пример с прямым и обратным методом вывода.
8. В чем отличие поиска «в глубину» от «поиска в ширину»?
9. Приведите формулу подсчета результирующего коэффициента для правила «Если e1 И e2 И e3, то С».
10. Приведите формулу подсчета результирующего коэффициента для правила «Если e1 ИЛИ e2 ИЛИ e3, то С».
11. Приведите формулу подсчета результирующего коэффициента для правила «Если НЕ e1 И e2 И e3 , то С».
12. Приведите формулу подсчета результирующего коэффициента для двух правил: «Если e1 И e2 И e3 , то С»; «Если Не e4 И e5 , то С»
13. Выполните расчет коэффициентов уверенности для сети, приведенной на рисунке 3.4. Значения коэффициентов уверенности для посылок: Ке1=0,5; Ке2= – 0,6; Ке3=0,7; Ке4=0,4; Ке5=0,8.

4. ОСНОВНЫЕ ПОНЯТИЯ И ОБЛАСТИ ПРИМЕНЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

4.1. Основные понятия

Нейрокомпьютинг или обработка данных с помощью нейроподобных сетей, реализованных на компьютерах либо в виде программ, либо аппаратным образом, в настоящее время все более широко используется для решения многих плохо формализуемых задач [2, 5, 9, 10].

Нейрокомпьютинг (нейровычисления) относятся к направлению информационных технологий, которое получило название Machine Leaning – ML, т. е. машинное обучение.

Указанное научное направление информатики обобщает результаты и идеи, связанные с нейросетевыми вычислениями, эволюционными и генетическими алгоритмами, нечеткими правилами. Термин Machine Leaning связывают с именем К. Самуэля, опубликовавшего в 1963 г. в сборнике Computers and Thought (Mc Craw-Hill, N. Y.) статью под названием «Some studies in machine leaning using the game of checkers» («Некоторые проблемы обучения машины на примере игры в шахматы»).

Не́йроинформа́тика – область научных исследований, лежащая на пересечении нейронаук и информатики. В сферу нейроинформатики входит сбор результатов, полученных в ходе нейробиологических исследований, перевод этих результатов в формат баз данных для их последующего анализа с помощью вычислительных моделей и специализированных компьютерных аналитических программных инструментов, обеспечение совместимости между базами данных, форматами моделей и другими коллекциями данных для облегчения обмена информацией о различных аспектах функционирования и строения нервных систем[2, 5, 6, 7].

Принцип *нейроподобных вычислений* отличается от привычных методов вычислений отсутствием необходимости программирования и, возможностью использования механизмов обучения нейронных сетейа для решения плохо формализуемых задач с нечетко заданными начальными данными и условиями.

Искусственные нейро́нные се́ти (ИНС) – это математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Это понятие появилось при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. ИНС могут рассматриваться как ориентированные графы с взвешенными связями, в которых искусственные нейроны являются узлами. Первой такой попыткой были нейронные сети Мак-Каллока и Питтса[5]. Впоследствии, после разработки алгоритмов обучения, получаемые модели

[5] «Когнитивные технологии для поддержки принятия управленческих решений» http://www.iis.ru/events/19981130/maximov.ru.html

стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает и посылает другим процессорам. Такие процессоры, соединённые в достаточно большую сеть с управляемым взаимодействием, способны выполнять сложные задачи.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей – это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть – способ решения проблемы эффективного параллелизма [2].

С точки зрения искусственного интеллекта, ИНС является основой *коннекционизма* и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

В основу коннекционизма положена идея о том, что нейроны можно моделировать простыми автоматами, а вся сложность мозга, гибкость его функционирования и другие качества определяются связями между нейронами.

С *коннекционизмом* связаны ряд идей:

– однородность системы (элементы одинаковы и просты, все определяется структурой связей);

– надежные системы из ненадежных элементов;

– «голографические» системы при разрушении случайно выбранной части система сохраняет свои свойства.

Широкие возможности системы связей компенсируют простоту элементов, их ненадежность и возможные отказы части связей.

Такие коннекционистские системы представляют собой устройства, использующие большое число рефлексов, называемых по имени канадского физиолога синапсами, которые в 1949 году описал теоретически Д. Хебб.

Нейронные сети не программируются, а обучаются. Возможность обучения – одно из главных преимуществ нейронных сетей перед традиционными алгоритмами.

Обучающая выборка для нейронной сети состоит из обучающих пар входного и выходного векторов, заданных в числовом формате.

Получение примеров для обучающей и тестовой выборки является серьезной проблемой построения нейросетевых моделей. В большинстве случаев моделируемый объект неизвестен и требуется выполнение экспериментов для нахождения таких примеров.

Необходимо заметить, что вся информация об объекте, который моделируется нейронной сетью, содержится в наборе примеров. Поэтому качество по-

строенной нейросетевой модели непосредственно зависит от того, насколько полно эти примеры описывают объект.

Теория нейронных сетей включают широкий круг вопросов из разных областей науки: биофизики, математики, информатики, схемотехники и информационной технологии. Поэтому понятие «нейронные сети» детально определить сложно.

ИНС представляют собой сеть искусственных нейронов, связанных между собой синаптическими соединениями. Сеть обрабатывает входную информацию и в процессе изменения своего состояния во времени формирует совокупность выходных сигналов.

Работа сети состоит в преобразовании входных сигналов во времени, в результате чего меняется внутреннее состояние сети и формируются выходные воздействия. Обычно ИНС оперирует цифровыми, а не символьными величинами.

Все модели ИНС требуют обучения или расчёта весов связей. В общем случае, обучение – такой выбор параметров сети, при котором сеть лучше всего справляется с поставленной проблемой. Обучение – это задача многомерной оптимизации, и для ее решения существует множество алгоритмов.

Проход по всем примерам обучающей выборки называют *циклом обучения*, а вся процедура прохождения сигнала по нейронной сети от входного вектора до выходного нейрона называется *тактом* функционирования сети.

4.2. Свойства биологических нейронных сетей

Развитие искусственных нейронных сетей вдохновляется биологией. Рассматривая сетевые конфигурации и алгоритмы, исследователи мыслят их в терминах организации мозговой деятельности. Но на этом аналогия может и закончиться. Наши знания о работе мозга столь ограничены, что мало бы нашлось руководящих ориентиров для тех, кто стал бы ему подражать. Поэтому разработчикам нейронных сетей приходится выходить за пределы современных биологических знаний в поисках структур, способных выполнять полезные функции. Во многих случаях это приводит к необходимости отказа от биологического правдоподобия, мозг становится просто метафорой, и создаются сети, невозможные в живой материи или требующие неправдоподобно больших допущений об анатомии и функционировании мозга [2, 5].

Несмотря на то, что связь с биологией слаба и зачастую несущественна, искусственные нейронные сети продолжают сравниваться с мозгом. Их функционирование часто напоминает человеческое познание, поэтому трудно избежать этой аналогии. К сожалению, такие сравнения неплодотворны и создают неоправданные ожидания, неизбежно ведущие к разочарованию. Несмотря на сделанные предупреждения, полезно все же знать кое-что о нервной системе млекопитающих, так как она успешно решает задачи, к выполнению которых лишь стремятся искусственные системы. Последующее обсуждение кратко.

Нервная система человека, построенная из элементов, называемых нейронами, имеет высокую сложность. Около 10^{11} нейронов участвуют в при-

мерно 10^{15} передающих связях, имеющих длину метр и более. Каждый нейрон обладает многими качествами, общими с другими элементами тела, но его уникальной способностью является прием, обработка и передача электрохимических сигналов по нервным путям, которые образуют коммуникационную систему мозга.

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от нашей кожи, ушей и глаз к мозгу, процессы мышления и управления действиями – все это реализовано в живом организме как передача электрических импульсов между нейронами [2, 5].

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию (рис. 4.1).

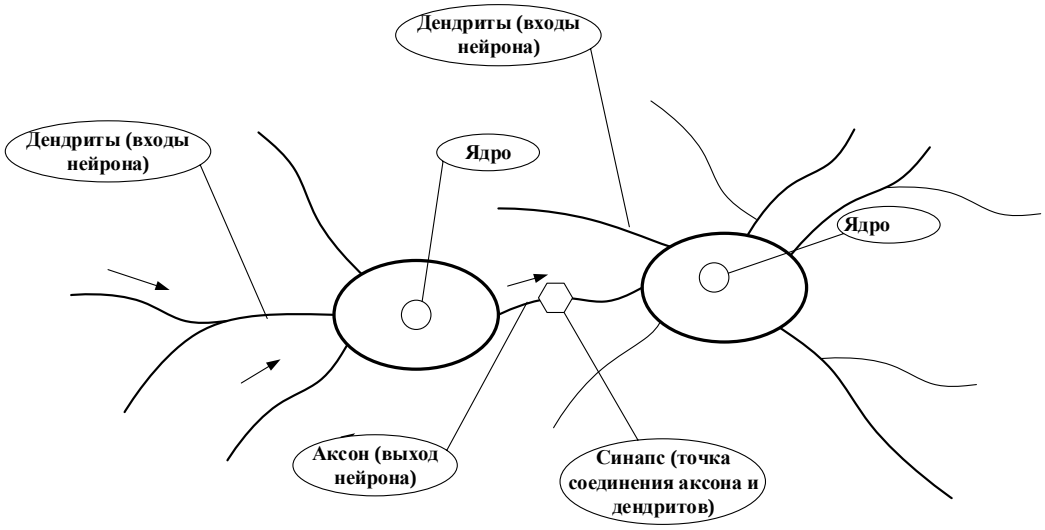


Рис. 4.1. Взаимосвязь биологических нейронов

Он состоит из тела и отростков нервных волокон двух типов – дендритов, по которым принимаются импульсы, и единственного аксона, по которому нейрон может передавать импульс. Тело нейрона включает ядро, которое содержит информацию о наследственных свойствах, и плазму, обладающую молекулярными средствами для производства необходимых нейрону материалов. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчик), который в конце разветвляется на волокна. На окончаниях этих волокон находятся специальные образования – синапсы, которые влияют на силу импульса.

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Когда импульс достигает синаптического окончания, высвобождаются определенные химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синаптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника ге-

нерировать электрические импульсы. Результативность синапса может настраиваться проходящими через него сигналами, так что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память, которая, возможно, ответственна за память человека. Важно отметить, что веса синапсов могут изменяться со временем, что изменяет и поведение соответствующего нейрона.

Использование нейронных сетей (НС) обеспечивает следующие полезные свойства систем.

1. *Нелинейность (nonlinearity).* Искусственные нейроны могут быть линейными и нелинейными. Нейронные сети из нелинейных элементов являются нелинейными и могут решать более сложные задачи.

2. *Отображение входной информации в выходную (input-output mapping*). Одной из популярных парадигм обучения является обучение с учителем (*supervised learning*). Это предполагает изменение синаптических весов на основе учебных примеров, каждый из которых состоит из входных сигналов и соответствующего ему желаемого отклика нейросети. Обучение проводится до тех пор, пока изменения синаптических весов не станут незначительными.

3. *Адаптивность (adaptivity).* НС обладают способностью адаптировать свои синаптические веса к изменениям окружающей среды.

4. *Очевидность ответа (evidential response).* При решении задачи классификации образов можно разработать нейронную сеть, которая повышает достоверность принимаемых решений и обеспечивает исключение сомнительных решений.

5. *Контекстная информация (contextual information).* Каждый нейрон сети потенциально может быть подвержен влиянию всех остальных её нейронов. Как следствие, существование нейронной сети непосредственно связано с контекстной информацией.

6. *Отказоустойчивость (fault tolerance).* При повреждении нейрона или его связи извлечение запомненной информации осуществляется с затруднением и потерей точности.

7. *Масштабируемость (VLSI Implementability).* Параллельная структура нейронной сети потенциально ускоряет решение некоторых задач и обеспечивает масштабируемость нейронных сетей в рамках технологии VLSI (very-large- scale-integrated), одним из преимуществ которой является возможность представить сложное поведение с помощью иерархической структуры.

8. *Единообразие анализа и проектирования (Uniformaty of analysis and desing).* Нейронные сети являются универсальным механизмом обработки информации. Одно и то же проектное решение нейронной сети может использоваться во многих предметных областях.

9. *Аналогия с нейробиологией (Neurobiological analogy*). Строение нейронных сетей определяется аналогией с человеческим мозгом, который демонстрирует отказоустойчивые параллельные вычисления для решения сложных задач.

К важнейшим свойствам биологических нейронных сетей относятся следующие:

1. *Параллельность обработки информации*. Каждый нейрон формирует свой выход только на основе своих входов и собственного внутреннего состояния под воздействием общих механизмов регуляции нервной системы.

2. *Способность к полной обработке информации*. Все известные человеку задачи решаются нейронными сетями. К этой группе свойств относятся ассоциативность (сеть может восстанавливать полный образ по его части), способность к классификации, обобщению, абстрагированию и множество других. Они до конца не систематизированы.

3. *Самоорганизация*. В процессе работы биологические НС самостоятельно, под воздействием внешней среды, обучаются решению разнообразных задач. Неизвестно никаких принципиальных ограничений на сложность задач, решаемых биологическими нейронными сетями. Нервная система сама формирует алгоритмы своей деятельности, уточняя и усложняя их в течение жизни. Человек пока не сумел создать систем, обладающих самоорганизацией и самоусложнением. Это свойство НС рождает множество вопросов. Ведь каждая замкнутая система в процессе развития упрощается, деградирует. Следовательно, подвод энергии к нейронной сети имеет принципиальное значение. Почему же среди всех диссипативных (рассеивающих энергию) нелинейных динамических систем только у живых существ, и, в частности, биологических нейросетей проявляется способность к усложнению? Почему человек пока не смог создать самоусложняющиеся системы?

4. *Биологические НС являются аналоговыми системами*. Информация поступает в сеть по большому количеству каналов и кодируется по пространственному принципу: вид информации определяется номером нервного волокна, по которому она передается. Амплитуда входного воздействия кодируется плотностью нервных импульсов, передаваемых по волокну.

5. *Надежность*. Биологические НС обладают фантастической надежностью: выход из строя даже 10 процентов нейронов в нервной системе не прерывает ее работы. По сравнению с последовательными ЭВМ, основанными на принципах фон Неймана, где сбой одной ячейки памяти или одного узла в аппаратуре приводит к краху системы.

Современные искусственные НС по сложности и «интеллекту» приближаются к нервной системе таракана, но уже сейчас демонстрируют ценные свойства:

1. *Обучаемость.* Выбрав одну из моделей НС, создав сеть и выполнив алгоритм обучения, мы можем обучить сеть решению задачи, которая ей по силам. Нет никаких гарантий, что это удастся сделать при выбранных сети, алгоритме и задаче, но если все сделано правильно, то обучение бывает успешным.

2. *Способность к обобщению*. После обучения сеть становится нечувствительной к малым изменениям входных сигналов (шуму или вариациям входных образов) и дает правильный результат на выходе.

3. *Способность к абстрагированию*. Если предъявить сети несколько искаженных вариантов входного образа, то сеть сама может создать на выходе идеальный образ, с которым она никогда не встречалась.

1. Доступность и возросшие вычислительные возможности современных компьютеров привели к широкому распространению программ, использующих принципы нейросетевой обработки данных, о которых мы поговорим подробнее в следующих разделах, но исполняемых на последовательных компьютерах. Этот подход не использует преимуществ присущего нейро-вычислениям параллелизма, ориентируясь исключительно на способность нейросетей решать не формализуемые задачи.

4.3. Цели и проблемы обучения нейронных сетей

Одно из важнейших свойств нейронных сетей – это способность к обобщению полученных знаний. Сеть, натренированная на обучающей выборке, генерирует ожидаемые результаты при подаче на ее вход данных, которые не участвовали в обучении. Все множество данных можно разделить на обучающее и тестовое подмножества [2, 5–7, 10].

Преимущества ИНС заключаются, во-первых, за счёт распараллеливания обработки информации; во-вторых, из способности самообучаться, т.е. создавать обобщения. Под термином обобщения понимается способность получать обоснованный результат на основании данных, которые не участвовали в процессе обучения.

Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

Данные, входящие в G (тестовое подмножество) и в L (обучающее подмножество) должны быть типичными элементами множества R. В обучающем подмножестве не должно быть уникальных данных, свойства которых отличаются от ожидаемых значений [5].

Феномен обобщения возникает вследствие большого количества комбинаций входных данных, которые могут кодироваться в сети с N-входами.

Подбор весов в процессе обучения имеет целью найти такую комбинацию их значений, которая наилучшим образом воспроизводила бы последовательность ожидаемых обучающих пар (X_k, Dk). При этом наблюдается тесная связь между количеством весов сети и количеством примеров обучающей выборки.

Если бы целью было обучения было запоминание всех примеров обучающей выборки, то их количество могло быть равным числу весов. В таком случае каждый вес соответствовал бы единственной обучающей паре. Но такая сеть не обладала бы свойством обобщения.

Для обретения способности обобщать данные сеть должна быть натренирована на *избыточном множестве данных*, поскольку тогда веса будут адаптироваться не к уникальным выборкам, а к их усредненным совокупностям.

Следовательно, для усиления способности к обобщению необходимо не только оптимизировать структуру сети в направлении ее минимизации, но и оперировать достаточно большим объемом обучающих данных.

Истинная цель обучения состоит в таком подборе архитектуры и параметров сети, которые обеспечат минимальную погрешность распознавания тестового подмножества, не участвующего в процессе обучения. Эту погрешность будем называть $E_G(W)$.

Со статистической точки зрения погрешность обобщения зависит от уровня погрешности обучения $E_L(W)$ и от доверительного интервала ε [10].

$$\mathrm{EG(w)} \leq \mathrm{EL}(w) + \varepsilon\left(\frac{P}{h}, EL\right) \qquad (4.1)$$

Параметр ε зависит от уровня погрешности обучения $E_L(W)$ и от отношения количества обучающих пар P к фактическому значению h, называемого мерой Вапника-Червоненкиса и обозначаемого VCdim.

Эта мера VCdim отражает уровень сложности НС и связана с количеством содержащихся в ней весов. Параметр ε уменьшается по мере возрастания количества обучающих пар к уровню сложности сети.

Поэтому обязательным условием хороших способностей к обобщению считается грамотное определение меры Вапника-Червоненкиса для сети заданной структуры.

Точная методика определения меры Вапника-Червоненкиса [2,5] не разработана. Известно, что мера зависит от количества синаптических весов.

Верхнюю и нижнюю границы меры можно *определить в интервале*

$$2\mathrm{N} \times \left[\frac{\mathrm{K}}{2}\right] \leq \mathrm{V\ Cdim} < 2\mathrm{Nw} \times (1 + \lg \mathrm{Nn}), \qquad (4.2)$$

где [ ] –целая часть числа;
N –размерность входного вектора;
K – количество нейронов скрытого слоя;
Nw – общее количество весов сети;
Nn – общее количество нейронов сети.

Из формулы (4.2) следует:

– нижняя граница диапазона приблизительно равна числу весов, связывающих входной и скрытый слои;

– верхняя граница превышает двукратное суммарное количество всех весов сети.

В связи с невозможностью точного определения меры VCdim в качестве её приближенного значения используется общее количество весов НС.

Таким образом, на погрешность обобщения оказывает влияние отношение количества обучающих выборок к количеству весов сети.

Небольшой объём ОВ при фиксированном количестве весов вызывает хорошую адаптацию сети к его элементам, однако не усиливает способности к обобщению, т. к. в процессе обучения наблюдается относительное превышение числа подбираемых параметров (весов) над количеством пар фактических и ожидаемых сигналов сети.

Фактическая задача аппроксимации подменяется задачей приближенной интерполяции. Высокие результаты обобщения достигаются в случае, когда количество ОВ в несколько раз превышает меру VCdim.

Иллюстрация. НС, скрытый слой которой содержал 80 нейронов, была обучена с нулевой погрешностью обучения. Минимизация этой погрешности на слишком малом (относительно количества весов) количестве обучающей выборки спровоцировала случайный характер этих весов, что при переходе к текстовым наборам стало причиной значительных отклонений фактических у от ожидаемых значений.

Уменьшение в нейронной сети количества скрытых нейронов до 5 при неизменном объёме обучающей выборки позволило обеспечить и малую погрешность обучения, и высокий уровень обобщения сети.

Дальнейшее уменьшение количества скрытых нейронов может привести к потере НС способности восстанавливать обучающие данные, т.е. к слишком большой погрешности $E_L(w)$.

На практике тренируют несколько НС с разным количеством скрытых нейронов при допустимой погрешности обучения.

Для упрощения архитектуры НС применяются методы редукции сети, которые можно разделить на две группы [2,5]:

1. Первая группа исследуют чувствительность целевой функции (ЦФ) к удалению веса или нейрона. С их помощью удаляются веса с наименьшим заметным влиянием, оказывающим минимальное воздействие на ЦФ и процесс обучения продолжается на редуцированной сети.
2. Методы второй группы связаны с модификацией ЦФ, в которую вводятся компоненты, штрафующие за неэффективную структуру сети. Чаще это элементы, усиливающие малые значения амплитуды весов. Такой способ менее эффективен чем первый, т.к. малые значения весов не обязательно ослабляют влияние на функционирование сети.

Принципиально иной подход состоит в начале обучения при минимальном (нулевом) количестве скрытых нейронов и последовательном их добавлении вплоть до достижения требуемого уровня нетренированной сети на исходном множестве обучающей выборки. Добавление нейронов происходит, как правило, по результатам оценивания способности сети к обобщению после определенного количества циклов обучения.

Основной проблемой при обучении НС является эффект переобучения сети (эффект бабушкиного воспитания).

Эффект переобучения сети заключается в том, что погрешность обучения при увеличении количества итераций монотонно уменьшается, тогда как погрешность обобщения снижается только до определенного предела, а затем снова начинает расти. Таким образом слишком долгое обучение может привести к переобучению сети. Зависимости погрешности обучения и обобщения приведены на рис. 4.2.

Эффект переобучения объясняется избыточностью нейронной сети и в слишком детальной адаптации весов к несущественным флуктуациям обучающих примеров. Такая ситуация наблюдается при использовании сети с чрезмерным (по сравнению с необходимым) количеством весов. Это особенно заметно, если НС содержит «лишние» веса, которые адаптируются к любым нерегулярностям обучающих примеров, *воспринимая их в качестве важных характеристик.*

В результате они становятся источником значительных погрешностей воспроизведения. Для предупреждения этого эффекта в обучающем множестве выделяется *контрольное подмножество* V, которое в процессе обучения применяется для проверки уровня обобщения (в точке А на рис. 4.2 обучение прекращается).

Погрешность обобщения можно вычислить по формуле среднеквадратичной ошибки [10]:

$$EG(w) = 0{,}5 \sum_{k=1}^{M} (Yk - Dk)^2, \qquad (4.3)$$

где M – размер тестового подмножества;

Y_k – прогноз НС на выходе выходного слоя, полученный после предъявления сети входного вектора X_K;

Dk – вектор ожидаемого выходного сигнала.

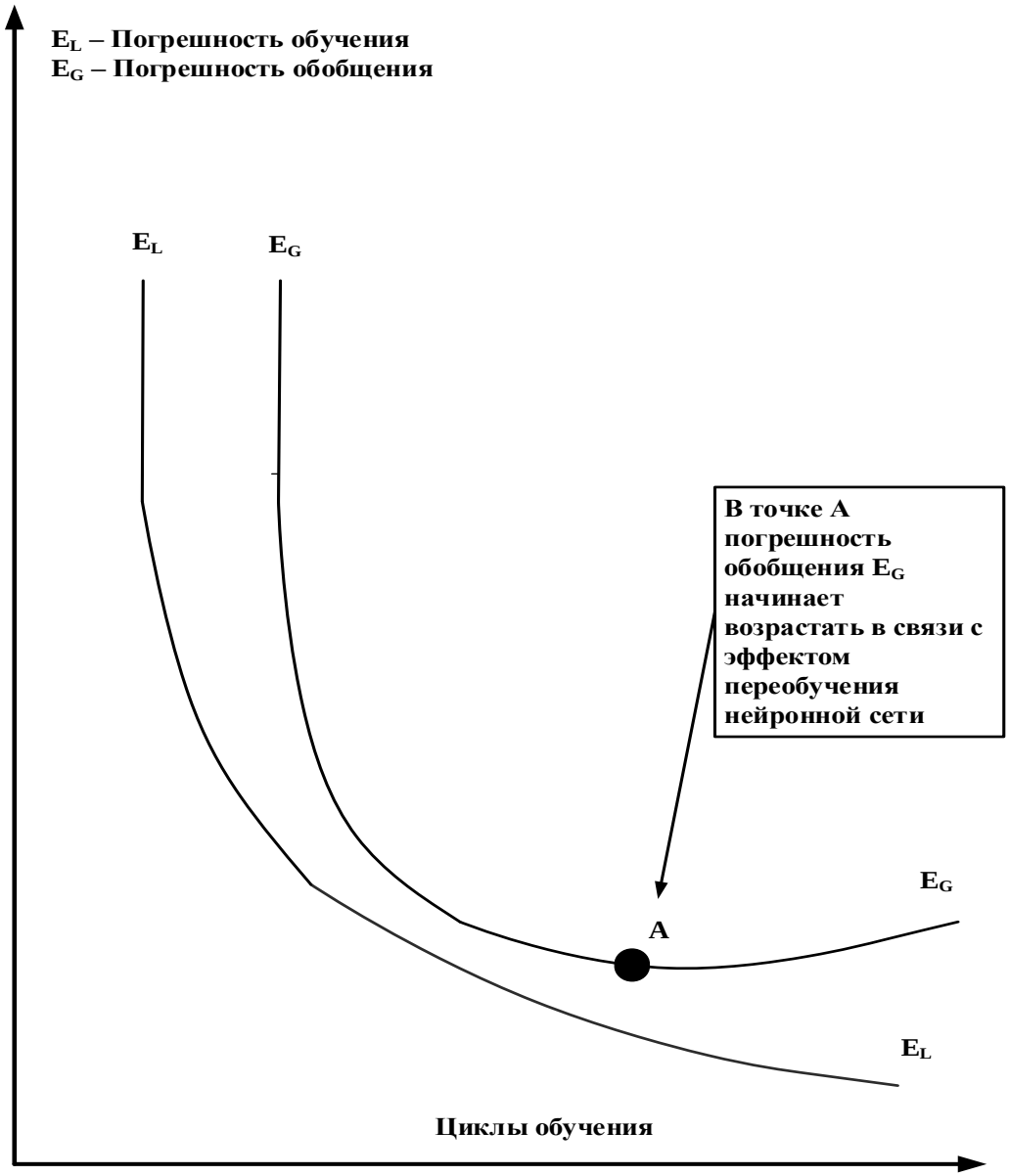


Рис. 4.2. Зависимости погрешности обучения и обобщения

Привлекательной чертой нейрокомпьютинга является единый принцип обучения нейросетей – минимизация эмпирической ошибки. Функция ошибки, оценивающая данную конфигурацию сети, задается извне – в зависимости от того, какую цель преследует обучение. Но далее сеть начинает постепенно модифицировать свою конфигурацию – состояние всех своих синаптических весов – таким образом, чтобы минимизировать эту ошибку. В итоге, в процессе обучения сеть все лучше справляется с возложенной на нее задачей.

Не вдаваясь в математические тонкости, образно этот процесс можно представить себе, как поиск самого глубокого оврага – минимума функции среднеквадратичной ошибки $E(W)$, зависящей от множества синаптических весов сети вектора W (см. рис. 4.3).

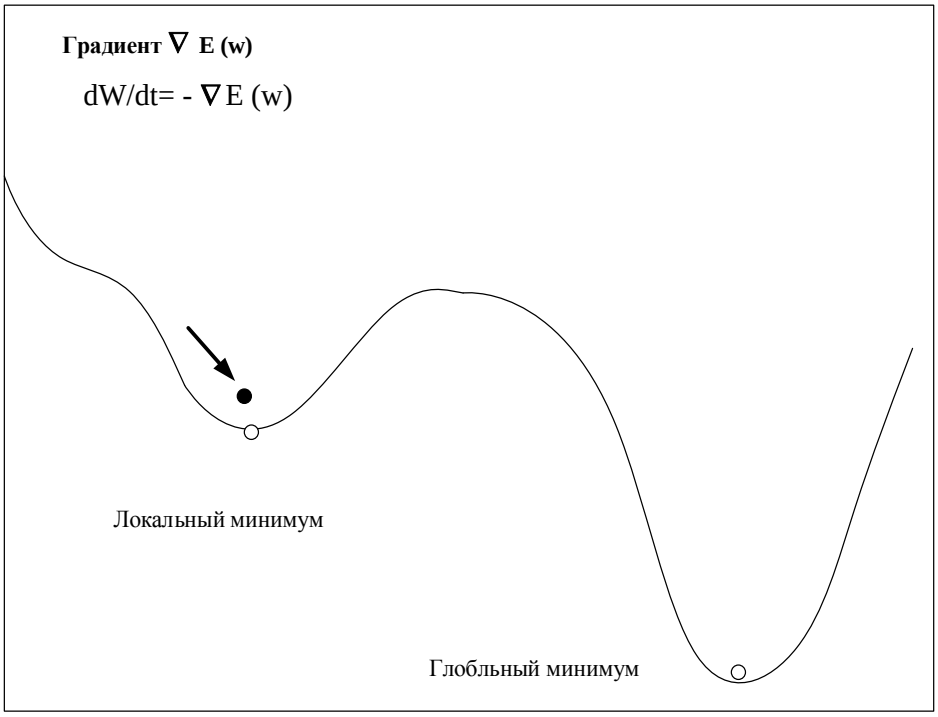


Рис. 4.3. Обучение сети как задача оптимизации

Базовой идеей всех алгоритмов обучения является учет локального градиента в пространстве конфигураций для выбора траектории быстрейшего спуска по функции ошибки. Функция ошибки, однако, может иметь множество *локальных минимумов*, представляющих *субоптимальные решения*. Поэтому градиентные методы обычно дополняются элементами стохастической оптимизации, чтобы предотвратить останов обучения нейронной сети в таких локальных минимумах. Идеальный метод обучения должен найти глобальный оптимум конфигурации сети.

Паралич сети. Если один из весов при обучении получает слишком большое значение, то при обычных значениях этого входа выход нейрона окажется в насыщении, т. е. будет близок к предельному значению функции активации. Выход нейрона будет мало зависеть от w, и поэтому производная среднеквадратичной ошибки замирает и стремится к нулю: $\partial E/\partial w \approx 0$.

Обучение по этому весу будет очень медленным, ведь изменение веса пропорционально производной. Выходной сигнал нейрона будет мало зависеть не только от веса, но и от входного сигнала x данного нейрона, а производная по x участвует в обратном распространении ошибки. Следовательно, предшествующие нейроны тоже будут обучаться медленно. Такое замедление обучения называется параличом сети.

Локальные минимумы. Как и любой градиентный алгоритм обучения, метод обратного распространения «застревает» в локальных минимумах функции ошибки, т.к. градиент вблизи локального минимума стремится к нулю. Шаг в алгоритме обратного распространения выбирается не оптимально. Точный одномерный поиск дает более высокую скорость сходимости [1, 6, 7, 10].

В дальнейшем нам встретится множество конкретных методов обучения сетей с разными конфигурациями межнейронных связей. Чтобы не потерять за деревьями леса, полезно заранее ознакомиться с базовыми нейро-архитектурами.

4.4. Классификация нейронных сетей

К настоящему времени разработано несколько типов нейросетей, используемых для решения практических задач обработки данных.

Нейронные сети различают по архитектуре (рис. 4.4) на неполносвязным, полносвязные, со случайными и регулярными связями, с симметричными и несимметричными связями [2, 5].

Неполносвязные нейронные сети описываются неполносвязным ориентированным графом и разделяются на однослойные и многослойные (слоистые) с прямыми, перекрёстными и обратными связями.

К неполносвязным относятся, неокогнитрон Фукушимы, сверточные нейронные сети и другие, не указанные в классификации на рис. 4.4.

В полносвязных нейронных сетях каждый нейрон передаёт свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети [10].

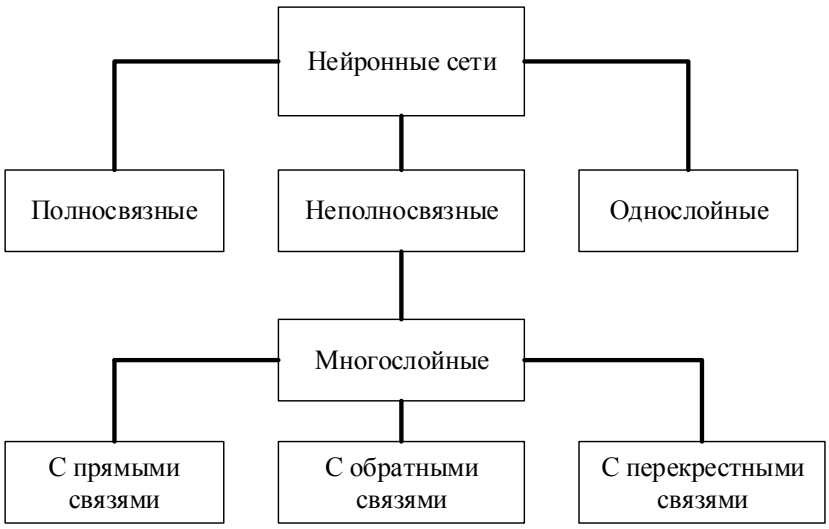


Рис. 4.4. Классификация искусственных нейронных сетей

Наиболее популярным среди многослойных нейронных сетей является многослойный персептрон.

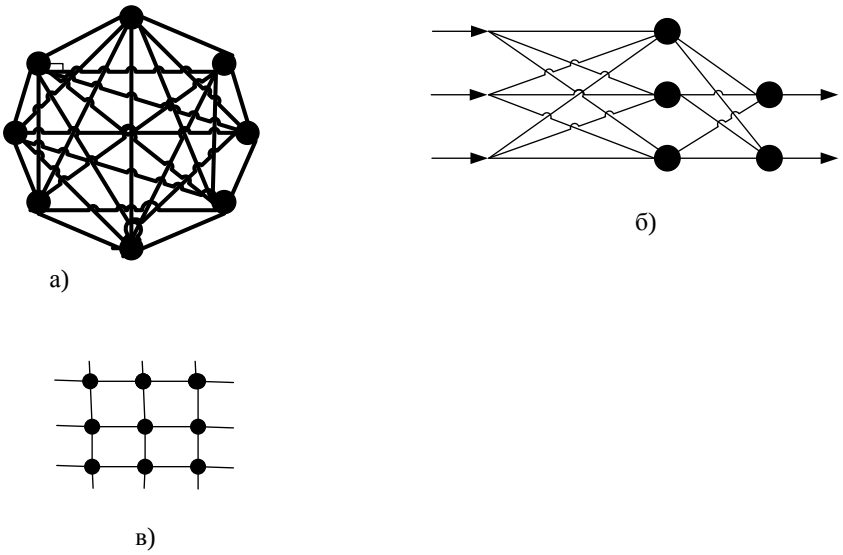
Нейронные сети прямого распространения являются статическими, а рекуррентные сети – динамическими в силу обратных связей, так как в них модифицируются входные сигналы, которые приводят к изменению состояния сети.

На рис. 4.5 изображены модели: а – полносвязной сети; б – многослойной сети с прямыми связями; в – слабосвязной сети.

В многослойных нейронных сетях нейроны объединяются в слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях.

В сетях с обратными связями информация с последующих слоёв передаётся на предыдущие слои.

В последнее время большое распространение получили так называемые радиальные сети, у которых функция активации позволяет определять близость исследуемого образа (точки пространства) к группе других точек, объединенных в кластер. В простейшем случае мера близости равна расстоянию от центра масс выделенного кластера[6].



а – полносвязная сеть; б – многослойная сеть с прямыми связями;
в – слабосвязная сеть

Рис. 4.5. Классификация нейронных сетей по типу связей

Нейронные сети классифицируются по характеру входных и выходных сигналов (информации, циркулирующей между узлами).

Преимущества и недостатки архитектур нейронных сетей без обратных связей и с обратными связями приведены в табл. 4.1.

Классификация по типу связей и типу обучения (Encoding-Decoding) представлена в табл. 4.2 и табл. 4.3.

[6] http://alife.narod.ru/lectures/neural/Neu_ch05.htm

Таблица 4.1

Сравнение архитектур нейронных сетей

Сравнение нейронных сетей	**Многослойные нейронные сети без обратных связей**	**Многослойные нейронные сети с обратными связями**
Преимущества	Простота реализации. Гарантированное получение ответа после прохождения данных по слоям	Минимизация размеров сети по сравнению сетями без обратных связей. Нейроны многократно участвуют в обработке данных. Меньший объем сети облегчает процесс обучения и упрощает архитектурные затраты на реализацию сети
Недостатки	Требуется большее число нейронов для алгоритмов одного и того же уровня сложности. Следствие – временные затраты и сложность обучения	Требуются специальные условия, гарантирующие сходимость вычислений

Таблица 4.2

Классификация по типу связей и типу обучения

Тип связей	**Принцип обучения нейронной сети**	
	с «учителем»	**без «учителя»**
1	2	3
Без обратных связей	Многослойные персептроны (аппроксимация функций, классификация) Применение генетических алгоритмов и алгоритмов «с подкреплением»	Соревновательные сети, карты Кохонена (сжатие данных, выделение признаков)
С обратными связями	Рекуррентные аппроксиматоры (предсказание временных рядов, обучение в режиме on-line)	Сеть Хопфилда (ассоциативная память, кластеризация данных, оптимизация)

Таблица 4.3

Классификация по типу обучения и архитектуре

Парадигмы	**Обучающее правило**	**Архитектура**	**Алгоритм обучения**	**Задача**
С учителем	Коррекция ошибки	Однослойный и многослойный персептрон	Алгоритм обучения персептрона. Обратное распространение Adaline, Madaline	Классификация образов, аппроксимация функций, управление
	Больцман	Рекуррентная сеть	Алгоритм обучения Больцмана	Классификация образов
	Хебба	МСПРС	Линейный дискриминантный анализ	Анализ данных, классификация образов
	Соревнование	Соревнование	Векторное квантование	Категоризация внутри класса, сжатие данных
		Сеть ART	ARTMap	Классификация образов
Без учителя	Коррекция ошибки	МСПРС	Проекция Саммона	Категоризация внутри класса, анализ данных
	Хебба	Сеть прямого распространения	Анализ главных компонент	
		Хопфилда	Обучение ассоциативной памяти	Ассоциативная память
	Соревнование	Соревнование	Векторное квантование	Категоризация, сжатие данных
		SOM Кохонена	SOM Кохонена	Категоризация, анализ данных
		Сеть ART	ART1,ART2	Категоризация

4.5. Свёрточная нейронная сеть

Свёрточная нейронная сеть (англ. convolutional neural network, CNN) – специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание изображений, входит в состав технологий глубокого обучения (англ. deep learning). Использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток [23-34].

Таким образом, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв (англ. *convolution layers*) и субдискретизирующих слоёв (англ. *subsampling layers* или англ. *pooling layers*, слоёв подвыборки). Архи-

тектура сети – однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки.

Название архитектура сети получила из-за наличия операции *свёртки*, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

Примеры для обучения из естественных изображений создаются на основе реальных данных. Их создание состоит из следующих этапов: Сбор графических данных (фотографирование интересующих объектов, снятие видеопотока с камеры, выделение части изображения на интернет странице).

1. Фильтрация – проверка изображений на ряд требований: достаточный уровень освещенности объектов на них, наличие необходимого объекта и т. д.
2. Подготовка инструментария для разметки (написание собственного или оптимизация готового).
3. Разметка (выделение четырехугольников, необходимых знакомест, интересующих областей изображения).
4. Присвоение каждому изображению метки (буква или название объекта на изображении).

Эти операции требуют значительных затрат рабочего времени, и, соответственно, подобный способ создания, обучающий базы весьма дорог.

Другой подход к созданию обучающих данных – их искусственная генерация. Можно взять несколько шаблонов / «идеальных» примеров (например, наборов шрифтов) и с помощью различных искажений создать необходимое число примеров для обучения. Можно использовать следующие искажения:

1. Геометрические (афинные, проективные и другие).
2. Яркостные и цветовые.
3. Замена фона.
4. Искажения, характерные для решаемой задачи: блики, шумы, размытие и т. п.

Такой способ создания обучающих примеров содержит в себе преимущества обоих вышеизложенных подходов: он *не требует высоких материальных затрат* и позволяет создать большое число примеров, необходимых для обучения распознавателя. Этот способ получил название аугментация (augmentation, «раздутие») данных для обучения нейронной сети. *Аугментация*[7] позволяет увеличить тренировочный набор данных и повысить точность распознавания за счет получения новых изображений с помощью случайного сдвига, масштабирования и растягивания. Параметры аугментации также подбираются в зависимости от задачи. Поэтому сверточная нейронная сеть не чувствительна к сдвигу, повороту изображения, ее качество распознавания намного превышает каче-

[7] «Аугментация (augmentation, «раздутие») данных для обучения нейронной сети на примере печатных символов», август 2015. https://habr.com/company/smartengines/blog/264677/

ство полносвязной нейронной сети за счет не попиксельного анализа изображения, а выделения отличительных черт объекта.

Компьютерные методы, имитирующие человеческое зрение, сегодня используются для решения многих задач – от определения лиц на Facebook и автопилотируемых Google-мобилей до суперсовременных алгоритмов диагностики заболеваний.

Свертка – фактически главное, что необходимо понять о сверточных нейронных сетях. Этот замысловатый математический термин нужен для движущегося окна или фильтра по исследуемому изображению. Перемещающееся окно применяется к определенному участку узлов, как показано на рис. 4.6 с примененным фильтром – (0.5 * значение в узле).

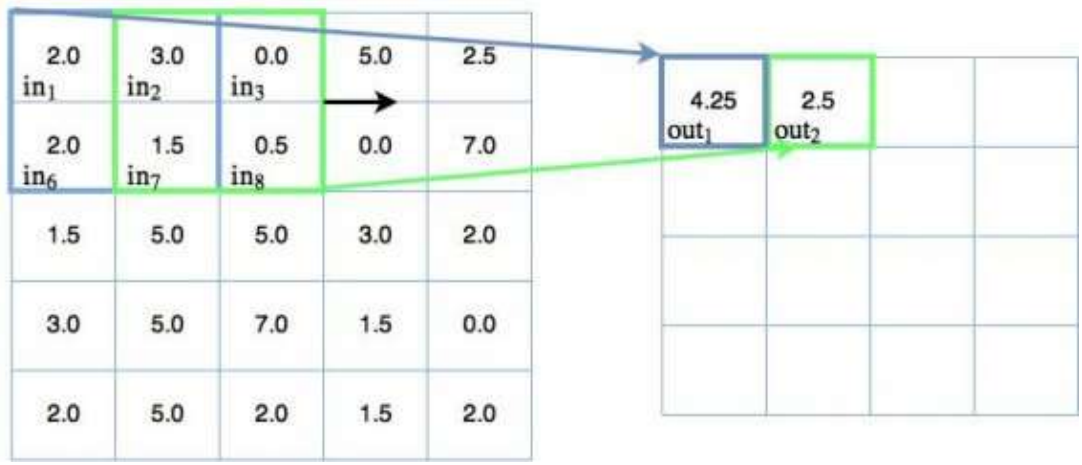


Рис. 4.6. Иллюстрация принципа работы сверточной нейронной сети

На диаграмме показаны только два выходных значения, каждое из которых отображает входной квадрат размера 2×2. Вес отображения для каждого входного квадрата, как ранее упоминалось, равен 0.5 для всех четырех входов (inputs). Поэтому выход может быть посчитан так:

$out_1 = 0.5in_1 + 0.5in_2 + 0.5in_6 + 0.5un_7 = 0.5*2.0 + 0.5*3.0 + 0.5*2.0 + 0.5*1.5 = 4.25$

$out_2 = 0.5in_2 + 0.5in_3 + 0.5in_7 + 0.5un_8 = 0.5*3.0 + 0.5*0.0 + 0.5*1.5 + 0.5*0.5 = 2.5$

Не каждый узел в первом (входном) слое соединен с каждым узлом во втором слое. Этим отличается архитектура CNN от полностью связанной нейронной сети, где каждый узел соединен со всем другими в следующем слое.

При движении фильтра по изображению одинаковые веса применяются для каждого 2 x 2 набора узлов. Каждый фильтр может быть обучен для выполнения специфичных трансформаций входного пространства. Следовательно, каждый фильтр имеет определенный набор весов, которые применяются для каждой операции свертки.

Этот процесс уменьшает количество параметров. Нельзя говорить, что любой вес постоянен внутри отдельного фильтра. В примере выше веса были *[0.5, 0.5, 0.5, 0.5]*, но ничего не мешало им быть и *[0.25, 0.1, 0.8, 0.001]*. Выбор конкретных значений зависит от обучения каждого фильтра.

Эти два свойства сверточных нейронный сетей существенно уменьшают количество параметров для тренировки, по сравнению с полносвязными сетями. Поскольку веса отдельных фильтров остаются постоянными, будучи примененными на входных узлах, они могут обучаться выбирать определенные признаки из входных данных. В случае изображений, архитектура способна учиться различать общие геометрические объекты – линии, грани и другие формы исследуемого объекта.

Вот откуда взялось определение признакового отображения. Из-за этого любой сверточный слой нуждается в множестве фильтров, которые тренируются детектировать различные признаки.

Следующий шаг в структуре CNN – прохождение выхода операции свертки через нелинейную активационную функцию. Например, используя функцию ReLU, обеспечивается нелинейное поведение этой нейронной сети [23–34].

Пулинг – другой тип техники скользящего окна, где вместо применения обучаемых весов используется статистическая функция некоторого типа по содержимому этого окна. Наиболее частый тип пулинга – *max pooling*, который применяет функцию *max()*.

Есть и другие варианты – mean pooling (который применяет функцию усреднения по содержимому окна), которые применяются в особых случаях.

Основными преимуществами для пулинга в сверточной нейронной сети являются:

1. Уменьшение количества параметров в вашей модели благодаря процессу *даунсемплинга (down-sampling)*.
2. Детектирование признаков становится более правильным при изменении ориентации или размера объекта.

На рис. 4.7 можно наблюдать действие max pooling. Для первого окна голубого цвета max pooling выдает значение 3.0, которое является максимальным значением узла в 2x2 окне. Таким же образом зеленое окно выводит максимальное значение, равно 5.0, а для красного окна максимальное значение – 7.0.

Он состоит в разделении карты признаков на непересекающиеся участки и выделении на этих участках нейронов с максимальной активностью.

Max-объединение карты признаков делает процесс распознавания более точным, избавляясь от ненужных «ореолов» и сокращая число параметров сверточной нейронной сети (таким образом устраняя потенциальные проблемы, связанные со сверподгонкой модели).

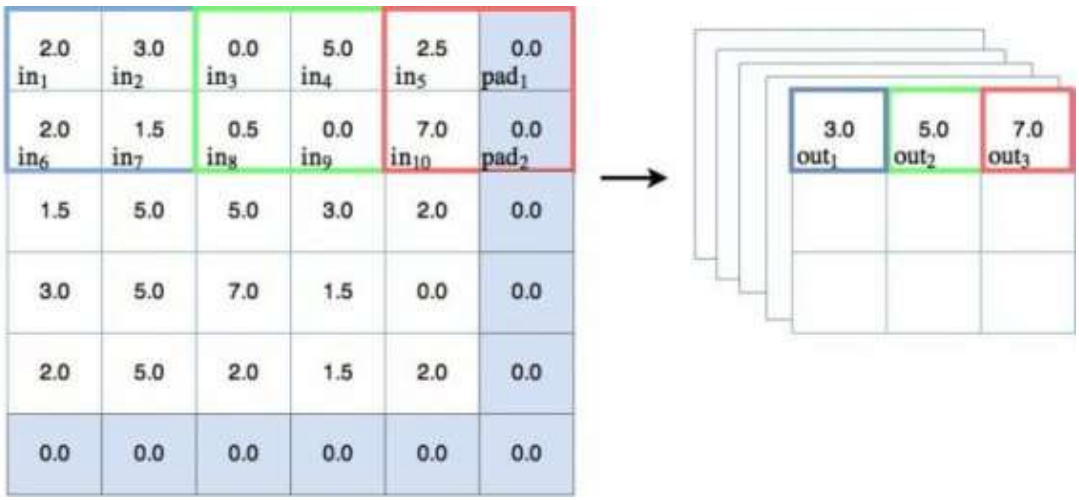


Рис. 4.7. Иллюстрация действия max pooling

Чаще всего модель сверточной нейронной сети состоит из нескольких слоев свёртки и ректификации, затем идет слой пулинга; этот шаблон может повторяться несколько раз.

Далее осуществляется переход к полносвязным слоям, которых также может быть несколько штук.

Последний полносвязный слой содержит оценку входного изображения, представленную в виде вектора размерностью N, где N – число распознаваемых классов.

Пример архитектуры сверточной нейронной сети представлен на рис. 4.8. Далее серия сверточных фильтров сканирует входное изображение (рукописная буква) для отображения признаков. Из выходов этих фильтров операции пулинга выбирают подвыборку. После этого идет следующий набор сверток и пулинга на выходе из первого набора операций пулинга и свертки. Далее осуществляется переход к полносвязным слоям, которых также может быть несколько штук. Последний полносвязный слой содержит оценку входного изображения, представленную в виде вектора размерностью N, где N – число распознаваемых классов.

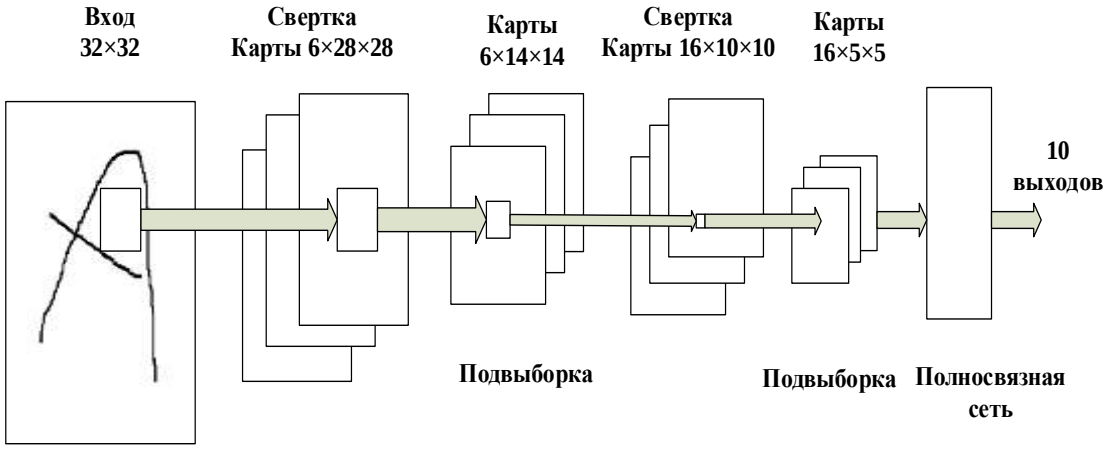


Рис. 4.8. Пример архитектуры сверточной нейронной сети

Распознавание изображений, основанное на глубоком обучении, сегодня решает многие задачи эффективнее, чем человеческое зрение

В качестве библиотеки для распознавания изображений чаще других используется Keras. Ее преимущества перед остальными библиотеками – это использование в качестве элементов построения сети слои вместо тензоров и графов потоков данных и минимальная трудоемкость прототипирования.

4.6. Принципы построения многослойных нейронных сетей

4.6.1. Архитектура многослойной нейронной сети

Среди различных архитектур нейронных сетей (НС) одной из популярных является многослойная архитектура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя. Для первого слоя нейрон первого скрытого слоя связан со всеми входами НС. НС, содержащая входной слой, один или несколько скрытых слоёв и один выходной слой, называется многослойным персептроном.

Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, которое минимизирует ошибку на выходе сети. По этому принципу реализован, например, алгоритм обучения однослойного персептрона [6].

В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и обучить такой персептрон невозможно, руководствуясь только величинами ошибок на выходах сети.

Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо.

Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Этот алгоритм обучения НС получил название процедуры обратного распространения.

Третий вариант – обучение многослойного персептрона с помощью генетических алгоритмов [5], которые демонстрируют неплохие результаты обучения только для относительно простых НС. В качестве генотипа используются веса синапсов и коэффициенты крутизны функций активации всех нейронов нейронной сети. Генетические алгоритмы позволяют проводить «распараллеливание» программы для сокращения времени обучения.

4.6.2. Алгоритм обратного распространения ошибки

Обратное распространение – это самый популярный алгоритм для обучения НС с помощью изменения весов связей. Ошибка распространяется от вы-

ходного слоя к входному, т. е. в направлении противоположном направлению прохождения сигнала при нормальном функционировании сети.

Выполнение алгоритма начинается с генерации произвольных весов для многослойной сети. Затем процесс, описанный ниже, повторяется до тех пор, пока средняя ошибка на входе не будет признана достаточно малой:

1. Берётся пример входного сигнала с соответствующим правильным значением выхода, т.е. входной и выходной векторы.
2. Рассчитывается прямое распространение сигнала через сеть (определяются весовые суммы Si и активаторы ui для каждого нейрона).
3. Начиная с выходов, выполняется обратное движение через ячейки выходного и промежуточного слоя, при этом программа рассчитывает значения ошибок по формуле (4.4) и (4.5):

– для нейрона выходного слоя

$$\delta_0 = (C_i - u_5) \times u_5 \times (1 - u_5) \quad (4.4)$$

– для всех нейронов скрытого слоя

$$\delta_i = (\sum mm > i \times w_{mi} \times \delta_0) \times u_i \times (1 - u_i) \quad (4.5)$$

Здесь m обозначает все нейроны, связанные со скрытым узлом, w – заданный вектор веса, u – выход активационной функции.

4. Веса в сети w^*_{ij} обновляются следующим образом по формулам (4.6) и (4.7):

– для весов синапсов между скрытым и выходным слоем

$$w_{ij}^* = w_{ij} - n \times \delta_0 \times u_i \quad (4.6)$$

– для весов синапсов между скрытым и входным слоем

$$w_{ij}^* = w_{ij} - n \times \delta_i \times u_i \quad (4.7)$$

Здесь параметр η характеризует коэффициент скорости обучения (или размер шага). Это небольшое значение ограничивает изменение, которое может произойти на каждом шаге. Параметр η можно определить экспериментальным путем. Лучше начать обучение с небольшого значения (η=0,1) и затем постепенно его повышать.

Продвижение вперёд по сети соответствует активации нейронов, а продвижение назад – коррекции весов синапсов и весов смещений нейронов. Затем веса обновляются таким образом, чтобы минимизировать ошибку для данного входного вектора. Если коэффициент обучения слишком велик, сеть может никогда не сойтись, т.е. не будут найдены нужные веса синапсов при минимальной среднеквадратичной ошибке обучения. При малом значении коэффициента обучения увеличивается время обучения сети.

4.6.3. Пример расчёта весовых коэффициентов

Проход вперёд. Сначала выполняется расчёт движения входного сигнала по сети (рис.4.9). Значение смещения для каждого нейрона равно 1.

$$U_3 = f(w_{3,1} \times u1 + w_{3,2} \times u2 + w_{3,b} \times 1) \quad (4.9)$$

$$U_3 = f(0 \times 1 + 0.5 \times 1 + 1 \times 1) = f(1.5)$$

$$f(S) = \frac{1}{1+e^{-\beta S}} \quad (4.10)$$

U3=0,81757 (при коэффициенте крутизны функции активации β=1)

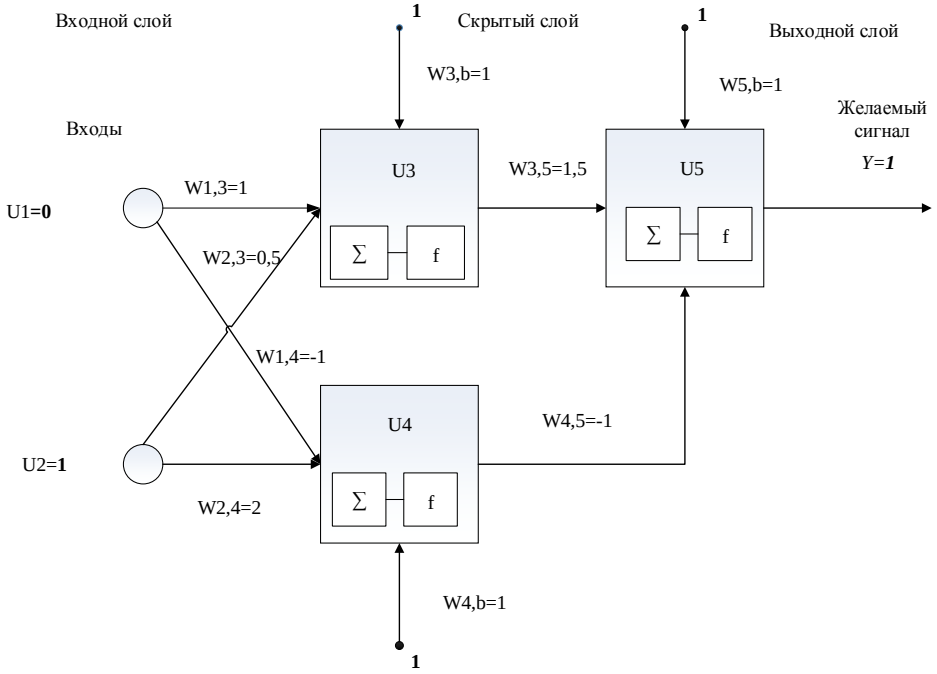


Рис.4.9. Иллюстрация фрагмента нейронной сети

Сначала выполняется расчёт движения входного сигнала по сети (рис.3.2).

$$U_4 = f(w_{4,1} \times u1 + w_{4,2} \times u2 + w_{4,b} \times 1) \quad (4.11)$$

U_4=f (−1*0+2*1+ 1*1=f (3)

$$f(S) = \frac{1}{1 + e^{-\beta S}}$$

U_4=0, 952574

(при коэффициенте крутизны функции активации β=1)

Теперь сигнал дошёл до скрытого слоя. Конечный шаг – переместить сигнал в выходной слой и рассчитать значение на выходе сети:

$$U_5 = f(w_{5,3} \times u_3 + w_{5,4} \times u_4 + w_{5,b} \times 1) \quad (4.12)$$

U_5=f (0, 81757*1, 5+0, 952574*(–1) + 1*1=f (1, 2195)

$$f(S) = \frac{1}{1 + e^{-\beta S}}$$

U_5=0,78139

(при коэффициенте крутизны функции активации β=1)

Правильный реакцией НС на входной сигнал является значение U_5=1,0; значение рассчитанное сетью, составляет 0, 78139.

Для коррекции весовых коэффициентов обычно используется среднеквадратичная ошибка, вычисляемая по формуле среднеквадратичной ошибки $E=0,5*\Sigma(y^{эталонное} - y^{расчётное})^2$.

Значение $y^{эталонное}$ берётся из обучающей выборки, а $y^{расчётное}$ вычисляется после подачи на входы нейронной сети входного вектора.

$E=0,5*(1,0 - 0,78139)^2=0,023895$.

Обратный проход процедуры обратного распространения ошибки.

По формуле (3.27) рассчитается ошибка в выходном узле:

$\delta_0= (C - u_5) *u_5*(1-u_5) =(1,0-0,78139) * 0,78139*(1- 0,78139) =0,0373$

Теперь следует рассчитать ошибку δ_{u3} и δ_{u4} для двух скрытых нейронов по формуле (3.28):

$\delta_{u3}= (\delta_0 * w_{5,3})*u_3 *(1- u_3)$,

$\delta_{u3}= (0,0373* 1,5* 0,81757 * (1 - 0,81757)= 0, 0083449$,

$\delta_{u4}= (\delta_0 * w_{5,4})*u_4 *(1- u_4)$,

$\delta_{u4}=(0,0373* (-1)* 0,952574*(1 - 0,952574)= - 0,0016851$.

Изменение весовых коэффициентов сети.

1. Сначала обновляются веса w^*_{ij} между выходным и скрытым слоем.

$$w_{ij}^*(t+1) = w_{ij}(t) + n \times \delta_0 \times u_i \qquad (4.13)$$

$w_{5,3}(t+1) = w_{5,3}(t) + (\eta*0, 0373*u3)$,

$w_{5,3} = 1, 5 + (0, 5*0, 0373*0, 81757) = 1, 51525$,

$w_{5,4}(t+1) = w_{5,4}(t) + (\eta*0,0373*u_4)$,

$w_{5,4}(t+1) = -1, 0 + (0, 5*0, 0373*0, 952574) = -0,9882$.

2. Теперь нужно обновить веса смещений для выходного нейрона

$w_{5,b}(t+1) = w_{5,b}(t) + (\eta*0,0373*1$,

$w_{5,b}(t+1) = 1 + (0, 5*0, 0373*1) =1, 01865$.

Для $w_{5,3}$ вес увеличен с 1,5 до 1,51525, а для $w_{5,4}$ вес изменён с -1 до $-0,9882$. Смещение обновлено для уменьшения среднеквадратичной ошибки E=0, 78139.

3. Теперь обновляются веса между входным и скрытым слоем:

$w_{3,1}(t+1) = w_{3,1}(t) + (0, 5*0, 0083449*u_1)$,

$w_{3,1}(t+1) = 1, 0 + (0, 5*0, 0083449*0) = 1, 0$,

$w_{3,2}(t+1) = w_{3,2}(t) + (\eta*0, 0083449*u_2)$,

$_{3,2}(t+1) = 0, 5 + (0, 5*0, 0083449*1) = 0, 50417$,

$w_{4,1}(t+1) = w_{4,1}(t) + (0,5*- 0,0016851*u_1)$,

$w_{4,1}(t+1) = -1,0 + (0,5*- 0,0016851*0)= -1,0$,

$w_{4,2}(t+1) = w_{4,2}(t) + (\eta*- 0,0016851*u_2)$,

$w_{4,2}(t+1) = 2 + (0,5*- 0,0016851*1)=1,999916$.

5. Теперь обновляются веса смещений для скрытых нейронов:

$W_{3,b}(t+1) = w_{3,b}(t) + (\eta * 0,0083449 * 1,$

$W_{3,b}(t+1) = 1 + (0,5 * 0,0083449 * 1) = 1,00417,$

$W_{4,b}(t+1) = w_{4,b}(t) + (\eta * -0,0016851 * 1,$

$W_{4,b}(t+1) = 1 + (0,5 * -0,0016851 * 1) = 0,99915.$

6. Обновление весов для скрытого слоя завершается. Выполним ещё один проход вперёд.

$U_3 = f(w_{3,1} * u_1 + w_{3,2} * u_2 + w_{3,b} * 1$

$U_3 = f(0 * 1 + 0,50417 * 1 + 1,00417 * 1 = f(1,50834)$

$$f(S) = \frac{1}{1+e^{-\beta S}},$$

U_3=0,8188 (при коэффициенте крутизны функции активации β=1)

$U_4 = f(w_{4,1} * u_1 + w_{4,2} * u_2 + w_{4,b} * 1$

$U_4 = f(-1 * 0 + 1,99915 * 1 + 1,99915 * 1 = f(2,99831)$

U_4=0,952497

$U_5 = f(w_{5,3} * u_3 + w_{5,4} * u_4 + w_{5,b} * 1$

$U_5 = f(1,51525 * 0,81888 + (-0,9822) * 0,952497 + 1,01865 * 1 = f(1,32379)$

U_5=0, 7898

(при коэффициенте крутизны функции активации β=1)

$E = 0,5 * (1,0 - 0,7898)^2 = 0,022$

На первой итерации ошибка была равна 0,023895, а на второй значение E уменьшилось до 0,022.

Следовательно, алгоритм обратного распространения ошибки обеспечивает уменьшение среднеквадратичной ошибки в процессе обучения НС.

При разработке нейронной сети требуется подобрать параметры и ее обучение и выбрать архитектуру сети (табл.4.4).

Таблица 4.4

Выбор параметров, влияющих на обучение и архитектуру нейронной сети

Наименование параметра	**Краткое описание «узких мест»**	**Рекомендации**
Характеристика обучающей выборки	Представительная обучающая выборка обеспечивает качество обучения и обобщение	Обеспечить полноту, равномерность и непротиворечивость обучающей выборки
Нормализация обучающей выборки	Исключение доминирования ряда параметров обучающей выборки	Подбор производится экспериментально и зависит от опыта специалиста
Последовательность выбора примеров из обучающей выборки	Случайный выбор примеров обеспечивает отсутствие «привыкания» НС к ряду примеров	Рекомендуется случайный выбор примеров из обучающей выборки
Тип активационной функции	Выбор типа функции активации влияет на погрешность обучения нейронной сети	Подбор производится экспериментально и зависит от опыта специалиста. Для скрытых слоев рекомендуется сигмоидальная функция, а для выходного слоя – линейная
Параметр крутизны активационной функции	Выбор типа коэффициента крутизны функции активации влияет на погрешность обучения нейронной сети и возможный «паралич» сети	Подбор производится экспериментально и зависит от опыта специалиста, создающего НС
Коэффициент скорости обучения	Влияет на скорость и качество обучения. Малый коэффициент скорости обучения увеличивает время обучения, а слишком большой приводит к «проскакиванию» глобального экстремума ошибки	Подбор производится экспериментально и зависит от опыта специалиста. В большинстве случаев рекомендуется значение 0,1
Выбор количества слоев и нейронов в слое	Неправильно выбранная архитектура нейронной сети увеличивает погрешность ее обучения	Подбор производится экспериментально и зависит от опыта специалиста
Выбор алгоритма обучения многослойной нейронной сети	Выбор алгоритма обучения нейронной сети с использованием алгоритма градиентного спуска производится экспериментально и зависит от опыта специалиста	В среде MatLab устойчивые результаты дает применение trainlm – функция обучения НС с использованием алгоритма Левенберга-Марквардта (LM)
Погрешность обобщения нейронной сети	Эффект переобучения нейронной сети. Погрешность обобщения начинает возрастать	В обучающем множестве выделяется контрольное подмножество, которое не используется при обучении объемом от 10 до 30 процентов от обучающего.
Метод момента или «тяжелого шарика»	В результате возрастает значение целевой функции и обеспечивается выход за пределы локального минимума среднеквадратичной ошибки	Однако показатель момента не должен доминировать в процессе обучения. Это может привести к нестабильности обучения (отсутствие сходимости алгоритма)

Анализ параметров, влияющих на качество обучения

Анализ параметров, влияющих на качество обучения и выбор архитектуры нейронной сети имеет несколько *«узких мест»* [2, 5].

1. *«Паралич сети».* В процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых коэффициентов сместят рабочую точку на сигмоидах многих нейронов в область насыщения. Малые величины производной от логистической функции приведут к остановке обучения, что парализует НС. Этот эффект называется «паралич сети».

2. *Выбор коэффициента скорости обучения.* Применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Эта проблема связана еще с одной, а именно – с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращения весов и, следовательно, скорость обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве параметра скорости обучения η обычно выбирается число меньше 1, но не очень маленькое, например, 0.1. Оно может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов стабилизируются, η кратковременно сильно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

3. *Нормализация обучающей выборки.* Нормализация сильно влияет на результаты обучения нейронной сети. Если данные подаются на вход сети в ненормализованном виде, то среднеквадратичная ошибка получается сравнительно большой. Например, для одной выборки среднеквадратичная ошибка получается большой при нормализации на интервалах $[-0{,}5;\ 0{,}5]$, $[-1;\ 1]$, а для другой выборки при нормализации $[0;1]$. Выбор способа нормализации зависит от конкретной обучающей выборки.

4. *Последовательность выбора примеров из обучающей выборки.* Исследуемым параметром является порядок выбора примеров из обучающей выборки (последовательный или случайный). Результаты экспериментов показывают, что при случайном выборе среднеквадратичная ошибка обучения меньше. Случайный метод выбора примеров из выборки не вызывает «привыкания» НС к выбираемым примерам.

5. *Выбор коэффициента крутизны.* Выявлено, что сигмоидальная функция с большим коэффициентом крутизны, обеспечивает наименьшую среднеквадратичную ошибку для заданного примера аппроксимации. Выбор коэффициента крутизны функции активации влияет на погрешность обучения нейронной сети и возможный «паралич» сети.

6. 6*, Смещение активационной функции.* Для заданного примера аппроксимации эксперименты показали зависимость среднеквадратичной ошибки обучения от величины смещения.

7. *Параметр скорости обучения.* Этот параметр сети влияет на скорость и сходимость алгоритма обучения. По результатам экспериментов выбирается параметр скорости обучения. В нашем примере он был выбран равный единице, обеспечивающий меньшую среднеквадратичную ошибку и, естественно, меньшее время обучения сети.

8. *Метод момента или «тяжелого шарика».* Одним из способов выхода из локального экстремума является применение параметра момента или «тяжелого шарика», значение которого выбрано 0,5.

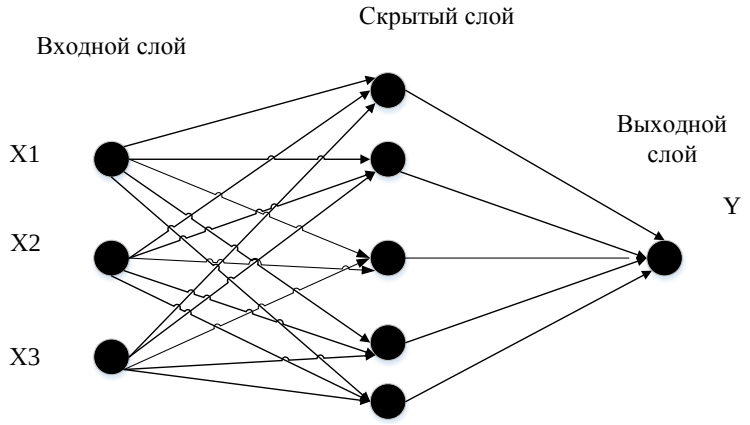


Рис. 4.10. Архитектура нейронной сети (3-5-1)

Метод момента или «тяжелого» шарика характеризуется формулой модификации весовых коэффициентов

$$w_{ij}(t+1) = -n \times \delta_i \times x_i + a \times w_{ij}(t). \qquad (4.14)$$

Первый член соответствует обычному методу градиентного спуска, а второй член соответствует методу «тяжелого шарика или момента» и не зависит от градиента.

Параметр α – коэффициент инерционности, влияние которого увеличивается с ростом α, особенно, в окрестности локального минимума.

В результате возрастает значение целевой функции и обеспечивается выход за пределы локального минимума.

Однако показатель момента не должен доминировать в процессе обучения, так как это приведет к нестабильности обучения (отсутствие сходимости алгоритма).

9. *Выбор алгоритма обучения нейронной сети.* При обучении многослойных нейронных сетей применяются три подхода:

1. Обучение по алгоритму обратного распространения ошибки.
2. Обучение с подкреплением.
3. Обучение с использованием генетических алгоритмов.

Таблица 4.5

Методы обучения многослойного персептрона

Сравнительные характеристики	**Вид обучения**		
	с «учителем»	**с «подкреплением»**	**с использованием генетических алгоритмов**
Что подается в качестве обучающих примеров	Набор пар входов-выходов $\{\mathbf{x}^\alpha, \mathbf{y}^\alpha\}$	Оценка выходов сети $\{\mathbf{x}^\alpha, \mathbf{y}(\mathbf{x}^\alpha)\}$	Набор пар входов-выходов $\{\mathbf{x}^\alpha, \mathbf{y}^\alpha\}$
Что требуется от нейронной сети	Найти функцию, обобщающую примеры, и научиться реагировать в схожих ситуациях	Научиться заданной «правильной» линии поведения	Решить задачу поиска минимума среднеквадратичной ошибки обучения НС с множеством переменных (параметров нейронной сети), представленных в виде хромосом

Обучение с учителем по алгоритму обратного распространения ошибки в большинстве случаев используют градиентные методы.

Ошибка нейронной сети зависит, как уже говорилось, от конфигурации сети – совокупности всех ее синаптических весов. Но эта зависимость не прямая, а опосредованная. Ведь непосредственные значения весов скрыты от внешнего наблюдателя. Для него сеть – своего рода черный ящик, и оценивать ее работу он может лишь основываясь на ее поведении, т.е. на том, каковы значения выходов сети при данных входах. Такой способ обучения, когда действительный выход нейросети сравнивают с эталонным, называют *обучением с учителем*.

Иногда выходная информация известна не полностью. Например, вместо эталонных ответов известно лишь хуже или лучше данная конфигурация сети справляется с задачей (вспомним детскую игру «холоднее-горячее» или лабораторную мышь в лабиринте с лакомствами и электрошоком). Этот тип обучения называют *обучением с подкреплением* (*reinforcement learning*).

Обучение с подкреплением является одним из способов машинного обучения, в ходе которого испытуемая система (*агент*) обучается, взаимодействуя с некоторой *средой*. Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются *сигналы подкрепления*, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или ее модель. Агент воздействует на среду, а среда воздействует на агента. О такой системе говорят, что она имеет обратную связь. Такую систему нужно рассматривать как единое целое, и поэтому линия раздела между средой и агентом достаточно условна. Конечно, с анатомической или физической точек зрения между средой и агентом (организмом) существует вполне определенная граница, но если эту систему рассматривать с функциональной точки зрения, то разделение становится не четким. Например, резец в руке скульптора можно считать либо частью сложного биофизического механизма,

придающего форму куску мрамора, либо частью материала, которым пытается управлять нервная система.

Розенблатт пытался классифицировать различные алгоритмы обучения, называя их системами подкрепления. Он даёт следующее определение:

«*Системой подкрепления называется любой набор правил, на основании которых можно изменять с течением времени матрицу взаимодействия (или состояние памяти) персептрона».*

Обучение с подкреплением – это обучение тому, что надо делать, как следует отображать ситуацию в действия, чтобы максимизировать некоторый сигнал поощрения (вознаграждения), принимающий числовые значения. Основная идея состоит в том, чтобы уловить и зафиксировать наиболее важные аспекты реальной задачи, имея ввиду организацию взаимодействия агента со средой для достижения некоторой цели. Ясно, что агент должен иметь возможность в какой-то мере воспринимать состояние среды, а также предпринять действия, которые могут повлиять на состояние среды [2, 5]. Агент должен иметь цель или цели, связанные с состоянием среды.

Одна из наиболее серьезных проблем, возникающих в обучении с подкреплением и отсутствующих в других видах обучения, – это проблема поиска компромисса между изучением и применением, то есть между поведением, направленным на получение знания, и поведением, основанным на использовании уже имеющегося знания. В качестве примера можно представить поведение интеллектуального робота в среде со случайными динамически появляющимися препятствиями (см. метод[8] «обучение с подкреплением»).

4.6.1. Обучение нейронной сети на базе генетических алгоритмов

Обучение с использованием генетических алгоритмов. Рассмотрим обучение нейронной сети (НС) на примере трёхслойной НС (рис. 4.11).

В каждом нейроне Р1, Р2, Р3 активационная функция имеет вид

$$f(S) = \frac{1}{1+e^{-\beta S}}, \qquad (4.15)$$

где β – параметр функции активации, ответственный за её крутизну.

Таким образом, целевая функция будет представлять собой максимальное для набора примеров относительное отклонение от эталонного значения, выраженное в процентах. Чем меньше значение целевой функции, тем сеть лучше. Здесь уже можно задать критерий останова генетического алгоритма. Можно указать число поколений, а можно задать условие на значение целевой функции. Например, остановить работу алгоритма, когда для одной из особей значение целевой функции будет равно 0,1%, что является достаточно хорошим результатом.

---

[8] «Аугментация (augmentation, «раздутие») данных для обучения нейронной сети на примере печатных символов», август 2015. https://habr.com/company/smartengines/blog/264677/

При обучении нейросетей размер популяции выбирается достаточно большим, как минимум 100 особей.

Хромосома в нашем случае будет представлять массив из 11 действительных чисел, по четыре на каждый скрытый нейрон и три для выходного нейрона, которые представляют собой веса соответствующих входов нейронов и параметры их функций активации (рис. 4.12).

Для генов хромосомы – введем ограничения, обусловленные природой нейросетей: $-1 \le w_{ij} \le 1$ и $P_i > 0$.

В качестве операторов скрещивания, отбора и редукции выбираются стандартные генетические операторы. Дополнительно для оператора скрещивания можно установить вероятность применения 0,95 и использовать элитные хромосомы.

В качестве оператора мутации будем использовать случайное изменение значений весов и параметра функции активации для каждого нейрона на случайную величину. Вероятность мутации 0,01. Причем одновременно будет изменяться параметр функции только для одного нейрона, и для каждого будет изменяться один из входных весов. Какие конкретно веса и параметры будут меняться, определяется по равномерному закону. Например, мутация может быть следующей: значение P_2 увеличивается на 0,1; значение w_{11} уменьшается на 0,05; w_{21} – уменьшается на 0,01; w_{31} – увеличивается на 0,05.

Исходная популяция будет формироваться на основе равномерного распределения для каждого элемента хромосомы.

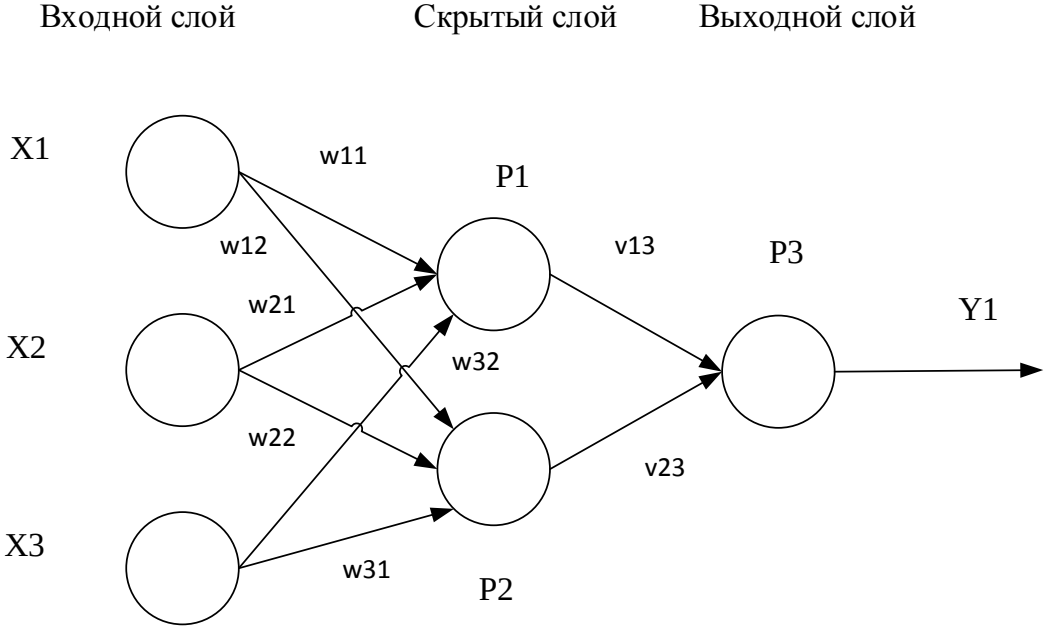


Рис. 4.11. Пример архитектуры нейронной сети

W_{11}	W_{12}	W_{13}	P_1	W_{21}	W_{22}	W_{23}	P_2	V_{31}	V_{32}	P_3

Рис. 4.12. Структура хромосомы

Целью работы генетического алгоритма является поиск значений весов для всех скрытых и выходных нейронов, а также параметра P_i их функций активации.

После останова работы такого алгоритма мы получим обученную нейросеть, удовлетворяющую заданным требованиям. Заметим, что в нашем примере алгоритм применялся только для построения весовой матрицы сети и вычисления параметров функций активации.

Недостатком применения генетических алгоритмов является увеличение времени обучения НС при повышении сложности ее архитектуры. Данный метод дает сравнительно хорошие результаты только для простых архитектур НС. Для сложных архитектур с большим количеством нейронов (более 50) требуется разработка параллельных программ генетических операторов (отбора, кроссинговера, мутации, редукции).

4.7. Области применения искусственных нейронных сетей

Нейросети пригодны для решения широкого круга задач, связанных с обработкой образов. Вот список типичных постановок задач для нейросетей [2,5–10, 21]:

- аппроксимация функций по набору точек (регрессия);
- классификация данных по заданному набору классов;
- кластеризация данных с выявлением заранее неизвестных классов-прототипов;
- сжатие информации;
- восстановление утраченных данных;
- ассоциативная память;
- оптимизация, оптимальное управление.

Этот список можно было бы продолжить и дальше. Заметим, однако, что между всеми этими внешне различными постановками задач существует глубокое родство. За ними просматривается некий единый прототип, позволяющий при известной доле воображения сводить их друг к другу.

Рассмотрим, например, задачу *аппроксимации* функции по набору точек. Это типичный пример некорректной задачи, т. е. задачи, не имеющей единственного решения. Чтобы добиться единственности, такие задачи надо *регуляризировать* – дополнить требованием минимизации некоторого регуляризирующего функционала. Минимизация такого функционала и является целью обучения нейронной сети. Задачи *оптимизации* также сводятся к минимизации целевых функций при заданном наборе ограничений. С другой стороны, *классификация* – это ни что иное, как аппроксимация функции с дискретными значениями (идентификаторами классов), хотя ее можно рассматривать и как частный случай заполнения пропусков в базах данных, в данном случае – в колонке идентификаторов класса. Задача *восстановления* утраченных данных, в свою очередь – это *ассоциативная память*, восстанавливающая прообраз по его части. Такими прообразами в задаче *кластеризации* выступают цен-

тры кластеров. Наконец, если информацию удается восстановить по какой-нибудь ее части, значит мы добились *сжатия* этой информации, и т. д.

Многие представители разных наук, занимающихся перечисленными выше задачами и уже накопившими изрядный опыт их решения, видят в нейросетях лишь перепев уже известных им мотивов. Каждый полагает, что перевод его методов на новый язык нейросетевых схем ничего принципиально нового не дает. Статистики говорят, что нейросети – это всего лишь частный способ статистической обработки данных, специалисты по оптимизации – что методы обучения нейросетей давно известны в их области, теория аппроксимации функций рассматривает нейросети наряду с другими методами многомерной аппроксимации. Нам же представляется, что именно синтез различных методов и идей в едином нейросетевом подходе и является неоценимым достоинством нейрокомпьютинга. Нейрокомпьютинг предоставляет единую методологию решения очень широкого круга практически интересных задач. Это, как правило, ускоряет и удешевляет разработку приложений. Причем, что обычно забывают за неразвитостью соответствующего hardware, но что, видимо, в конце концов сыграет решающую роль, нейросетевые алгоритмы решения всех перечисленных выше задач заведомо параллельны. Следовательно, все может быть решено быстрее и дешевле.

Практически в каждой предметной области можно найти постановки нейросетевых задач [2,5–10].

Экономика и бизнес: предсказание рынков, оценка риска невозврата кредитов, предсказание банкротств, оценка стоимости недвижимости, выявление пере- и недооцененных компаний, автоматическое рейтингование, оптимизация портфелей, оптимизация товарных и денежных потоков, автоматическое считывание чеков и форм, безопасность транзакций по пластиковым карточкам.

Программное обеспечение компании RETEK, дочерней фирмы HNC Software, – лидер среди крупных ритейлоров с оборотом свыше $1 млрд. Ее последний продукт января 1998 года Retek Predictive Enterprise Solution включает развитые средства нейросетевого анализа больших потоков данных, характерных для крупной розничной торговли. Он также содержит прогнозный блок, чтобы можно было заранее просчитать последствия тех или иных решений. (http://www.retek.com)

Медицина: обработка медицинских изображений, мониторинг состояния пациентов, диагностика, факторный анализ эффективности лечения, очистка показаний приборов от шумов.

Группа *НейроКомп* из Красноярска (под руководством А. Н. Горбаня) совместно с Красноярским межобластном офтальмологическом центром им. Макарова разработали систему ранней диагностики меланомы сосудистой оболочки глаза. Этот вид рака составляют почти 90% всех внутриглазных опухолей и легко диагностируется лишь на поздней стадии. Метод основан на косвенном измерении содержания меланина в ресницах. Полученные данные спектрофотометрии, а также общие характеристики обследуемого (пол, возраст и др.) подаются на входные синапсы 43-нейронного классификатора. Нейросеть

решает, имеется ли у пациента опухоль, и если да, то определяет ее стадию, выдавая, кроме этого, процентную вероятность своей уверенности (http://www.chat.ru/~neurocom/).

Авионика: обучаемые автопилоты, распознавание сигналов радаров, адаптивное пилотирование сильно поврежденного самолета.

Компания McDonnell Douglas Electronic Systems разработала автоматический переключатель режимов полета в реальном масштабе времени в зависимости от вида повреждения самолета. Данные от 20 сенсорных датчиков и сигналов от пилота используются нейросетью для выработки около 100 аэродинамических параметров полета. Сильной стороной является возможность сети адаптироваться к непредсказуемым аэродинамическим режимам, таким как потеря части крыла и т. д.

Связь: сжатие видеоинформации, быстрое кодирование-декодирование, оптимизация сотовых сетей и схем маршрутизации пакетов.

Нейросети уже продемонстрировали коэффициент сжатия 120:1 для черно-белого видео. Цветное видео допускает примерно вдвое большую степень сжатия 240:1 за счет специальной схемы кодирования цветов. (http://www.ee.duke.edu/~cec/JPL/paper.html)

Интернет: ассоциативный поиск информации, электронные секретари и агенты пользователя в сети, фильтрация информации в push-системах, коллаборативная фильтрация, рубрикация новостных лент, адресная реклама, адресный маркетинг для электронной торговли.

Фирма Autonomy отделилась от родительской фирмы Neurodynamics в июне 1996 года с уставным капиталом $45 млн и идеей продвижения на рынок Internet электронных нейросетевых *агентов*. Согласно ее пресс-релизу, первоначальные вложения окупились уже через год. Компания производит семейство продуктов AGENTWARE, создающих и использующих профили интересов пользователей в виде персональных автономных нейро-агентов. Такие компактные нейро-агенты (не более 1 Кб) могут представлять пользователя в любом из продуктов компании. Например, агенты могут служить в качестве нейро-секретарей, фильтруя поступающую по информационным каналам информацию. Они также могут постоянно находиться на сервере провайдера, или посылаться для поиска в удаленных базах данных, осуществляя отбор данных на месте. В будущем, когда эта технология получит достаточное распространение, она позволит снизить нагрузку на трафик Сети. (http://www.agentware.com)

Автоматизация производства: оптимизация режимов производственного процесса, комплексная диагностика качества продукции (ультразвук, оптика, гамма-излучение), мониторинг и визуализация многомерной диспетчерской информации, предупреждение аварийных ситуаций, робототехника.

Ford Motors Company внедрила у себя нейросистему для диагностики двигателей после неудачных попыток построить экспертную систему, т. к. хотя опытный механик и может диагностировать неисправности он не в состоянии описать алгоритм такого распознавания. На вход нейро-системы подаются данные от 31 датчика. Нейросеть обучалась различным видам неисправностей по 868 примерам. «После полного цикла обучения качество диагностирования не-

исправностей сетью достигло уровня наших лучших экспертов, и значительно превосходило их в скорости.» (Marko K, et. al., Ford Motors Company, Automative Control Systems Diagnostics, IJCNN 1989)

Политические технологии: анализ и обобщение социологических опросов, предсказание динамики рейтингов, выявление значимых факторов, объективная кластеризация электората, визуализация социальной динамики населения.

Группа *НейроКомп* из Красноярска довольно уверенно предсказала результаты президентских выборов в США на основании анкеты из 12 вопросов. Причем, анализ обученной нейросети позволил выявить пять ключевых вопросов, ответы на которых формируют два главных фактора, определяющих успех президентской кампании. Этот пример будет рассмотрен более подробно в главе, посвященной извлечению знаний с помощью нейросетей.

Безопасность и охранные системы: системы идентификации личности, распознавание голоса, лиц в толпе, распознавание автомобильных номеров, анализ аэрокосмических снимков, мониторинг информационных потоков, обнаружение подделок.

Многие банки используют нейросети для обнаружения подделок чеков. Корпорация Nestor (Providence, Rhode Island) установила подобную систему в Mellon Bank, что по оценкам должно сэкономить последнему $500,000 в год. Нейросеть обнаруживает в 20 раз больше подделок, чем установленная до нее экспертная система.

Ввод и обработка информации: Обработка рукописных чеков, распознавание подписей, отпечатков пальцев и голоса. Ввод в компьютер финансовых и налоговых документов.

Разработанные итальянской фирмой RES Informatica нейросетевые пакеты серии FlexRead, используются для распознавания и автоматического ввода рукописных платежных документов и налоговых деклараций. В первом случае они применяются для распознавания не только количества товаров и их стоимости, но также и формата документа. В случае налоговых деклараций распознаются фискальные коды и суммы налогов.

Геологоразведка: анализ сейсмических данных, ассоциативные методики поиска полезных ископаемых, оценка ресурсов месторождений.

Нейросети используются фирмой Amoco для выделения характерных пиков в показаниях сейсмических датчиков. Надежность распознавания пиков – 95% по каждой сейсмо-линии. По сравнению с ручной обработкой скорость анализа данных увеличилась в 8 раз. (J.Veezhinathan & D.Wadner, Amoco, First Break Picking, IJCNN, 1990)

Обилие приведенных выше применений нейросетей – не рекламный трюк. Просто нейросети – это не что иное, как новый инструмент анализа данных. И лучше других им может воспользоваться именно специалист в своей предметной области. Основные трудности на пути еще более широкого распространения нейротехнологий – в неумении широкого круга профессионалов формулировать свои проблемы в терминах, допускающих простое нейросетевое решение. Данное учебное пособие призвано помочь усвоить типовые постанов-

ки задач для нейросетей. Для этого, прежде всего, нужно четко представлять себе основные особенности нейросетевой обработки информации – *парадигмы* нейрокомпьютинга.

4.7. Нормализация обучающей выборки

Входная и выходная информация для нейронной сети формируется из векторов IN (входные значения обучающей выборки), OUT (выходные значения обучающей выборки).

Нормализация значений необходима в виду того, что на веса синапсов сети обычно наложены требования принадлежности некоторому диапазону значений. Это приводит к тому, что обычно нельзя подавать сети входные сигналы в их истинном диапазоне величин и получать от сети выходные сигналы в требуемом диапазоне.

Кроме того, нормирование исключает доминирование одних входных сигналов перед другими [5].

Поэтому перед подачей входных сигналов их необходимо нормировать в одном из диапазонов, например, от минус 1 до 1 или от минус 0,5 до 0,5 или от 0 до 1.

Наиболее простое нормирование сигналов можно выполнить следующим образом. Каждая компонента входного вектора данных x_i заменяется величиной, вычисляемой по формулам

$$x_{i0} = \frac{x_i - 0.5(\max\{x_i\} + \min\{x_i\})}{0.5(\max\{x_i\} - \min\{x_i\})}, \qquad (4.16)$$

$$x_{i0} = \frac{x_i - 0.5(\max\{x_i\} + \min\{x_i\})}{(\max\{x_i\} - \min\{x_i\})}, \qquad (4.17)$$

где x_{i0} – нормализованное значение величины;

x_i – нормализуемая величина;

max x_i и min x_i – соответственно максимальное и минимальное значения для данной компоненты, вычисленные по всей обучающей выборке.

Можно нормировать и по-другому, например, пересчитывая выборку так, чтобы разброс данным был единичным, по формуле

$$x_{i0} = \frac{x_i - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}}. \qquad (4.18)$$

Здесь имеется одна сложность. Любое изменение обучающей выборки должно соответственно менять и правило нормирования данных.

Выбор способа нормализации и значений зависит от конкретной обучающей выборки и выбирается экспериментально.

Для денормализации данных используются обратные преобразования, выполняемые по формулам (4.19), (4.20), (4.21):

$$x_i = \min\{x_i\} + x_{i0}(\max\{x_i\} - \min\{x_i\});\qquad (4.19)$$

$$x_i = 0.5(\max\{x_i\} + \min\{x_i\}) + 0.5x_{i0}(\max\{x_i\} - \min\{x_i\}),\qquad (4.20)$$

$$x_i = 0.5(\max\{x_i\} + \min\{x_i\}) + x_{i0}(\max\{x_i\} - \min\{x_i\}).\qquad (4.21)$$

Для обучения нейронной сети формируется задачник из обучающих векторов. Каждая обучающая пара состоит из двух векторов (X_k – входной вектор, Dk – выходной вектор). Вектор D_k представляет собой ответы, которые должна выдавать сеть после обучения.

При формировании обучающей выборки встречаются две основные проблемы. Одна проблема заключается в наличии противоречивых выборок – одним и тем же входным данным соответствуют различные выходные. Для решения данной проблемы в нейросетевых программах предусматривается проверка путем сравнения обучающих выборок. Данная проблема возникает при неполноте входной информации.

Другой проблемой является ситуация, когда в таблице данных имеются пробелы. Вместо таких отсутствующих компонент данных можно подавать нуль, можно исключать некомплектные векторы из обучающей выборки, можно перед обучением сети решать задачу заполнения пробелов в данных некоторыми правдоподобными значениями.

При разработке нейросетевой экспертной системы необходимо использовать специальный формат данных. Для этих целей используется специальный компонент, называемый предобработчиком.

Разработка эффективных предобработчиков для нейронных сетей является новой, почти совсем не исследованной областью. Большинство разработчиков нейросетевого программного обеспечения склонны возлагать функции предобработки входных данных на пользователя. Это решение технологически неверно. Дело в том, что при постановке задачи для нейронной сети трудно сразу угадать правильный способ предобработки. Для его подбора проводится серия экспериментов. В каждом из экспериментов используется одна и та же обучающая выборка и разные способы предобработки входных данных сети.

Если привычный для человека способ представления входных данных непригоден для нейронной сети, то и формат ответов нейронной сети часто малопригоден для человека. Необходимо интерпретировать ответы нейронной сети.

Интерпретация зависит от вида ответа. Так, если ответом нейронной сети является действительное число, то его, как правило, приходится масштабировать и сдвигать для попадания в нужный диапазон ответов. Если сеть используется как классификатор, то выбор интерпретаторов еще шире.

Например, экспертной системе, входной информацией которой служили изображения микрофотографий, предоставили спектральный состав этих изображений. При его использовании обучение нейронной сети происходит не по внешнему виду снимков, а по их цветовой насыщенности. До начала процесса обучения формируются файлы спектров, которые в дальнейшем используются как обучающие последовательности.

Файлы спектров представляют собой массивы из 256 чисел, каждое из которых показывает долю цвета палитры в анализируемом изображении.

Нейронная сеть в ходе обучения находит характерные участки спектра. На практике этот метод зарекомендовал себя как наиболее быстрый. Время обучения, по сравнению с традиционным способом, уменьшается в 15–20 раз. Связано это с тем, что в качестве обучающей последовательности подаются не графические массивы размером 512x512 пикселей (262144 элемента), а массивы спектров из 256 элементов.

Этот метод имеет еще и то достоинство, что не зависит от ориентации изображения. Как бы ни было подано изображение (прямо, боком, под углом) будут анализироваться не контуры, а его цветовая насыщенность, которая во всех случаях окажется одинаковой.

4.8. Примеры реализации нейронных систем

Приведенные ниже материалы, подобранные профессором Красноярской государственной медицинской академии Д.А. Россиевым, содержат интересные разработки по нейросетевым экспертным системам [5].

В Италии разработана интересная экспертная система для диагностики и лечения артериальной гипертонии. Система включает в себя три нейросетевых модуля, причем ответы одних являются входными данными для других. В начале исследования больному проводят измерение систолического и диастолического давления каждые полчаса в течение суток. Данные за каждый час усредняются. Таким образом, образуется массив из 48 величин артериального давления (по 24 для систолического и диастолического).

После этого первый модуль, состоящий из двух трехслойных нейросетей (в каждой из которых 2 входных, 4 «скрытых» и 24 выходных нейрона), на основании данных о поле и возрасте больного рассчитывает аналогичные «должные» величины и сравнивает их с реальными.

Параллельно второй модуль (двухслойная нейросеть с 17 входными и 4 выходными нейронами) на основании клинических данных (симптоматика, анамнез) рассчитывает возможные сочетания гипотензивных лекарственных средств, которые могут быть использованы для лечения данного больного.

Данные, снятые с выходов обоих модулей, вместе с клиническими данными подаются на вход последнего, третьего модуля (6-слойная нейросеть). Этот модуль оперирует 4 группами гипотензивных препаратов (диуретики, бетаадреноблокаторы, ингибиторы ангиотензина, блокаторы кальциевых каналов). Цель – назначить суточный (почасовой) график приема больным лекарств каждой (если требуется) из 4 групп. Поэтому этот модуль имеет 96 выходных нейронов

(4 препарата на 24 часа). С каждого выходного нейрона снимается доза, соответствующая одному препарату, назначаемому на данный час суток. Естественно, что в реальной ситуации большинство выходных данных равны нулю.

Таким образом, создается оптимальная для пациента схема лечения гипертонии. Нужно отметить, что система учитывает некоторые особенности приема препаратов больными, например, затруднение приема препаратов ночью (назначает ночной прием только в крайних случаях), запрет на назначение мочегонных лекарств на ночь.

Отличительной чертой системы является возможность пользователя (врача) передавать нейронной сети свой опыт. Для этого создателями программы предусмотрен специальный блок, который выводит на экран компьютера суточные кривые артериального давления и предлагает врачу ввести в компьютер суточную схему приема гипотензивных препаратов в необходимых, по его мнению, дозах. Введенный пример помещается в базу данных. В любое время можно инициировать дообучение нейронных сетей с новыми примерами.

В одной из работ приводится метод выявления атеросклеротических бляшек в артериях с использованием нейронных сетей. Для этого применяется нейросеть, интерпретирующая флуоресцентные спектры, получаемые при исследовании тканей с помощью лазера.

Аналогичным образом проводится диагностика заболеваний периферических сосудов, например, определение форм артериита.

Проводится комплекс исследований по использованию нейросетей для диагностики инфаркта миокарда. Автор приводит данные по чувствительности (77,7%) и специфичности (97,2%) нейросетевого теста. С помощью нейронной сети устанавливается диагностическую значимость клинических параметров при диагностике инфаркта миокарда.

Нейросетевой анализ акустических сигналов позволяет проводить диагностику клапанных шумов сердца и оценивать систолическую и диастолическую фазы сердечного сокращения с постановкой предварительного диагноза.

Нейросети используются терапевтами для диагностики заболеваний печени по лабораторным данным исследования функций печени, а также для дифференциальной диагностики заболеваний печени и желчного пузыря.

Нейропрограммы могут с успехом работать с медицинскими данными, относящимися к субъективным категориям, например, в психиатрии. Оценка субъективных данных дает возможность распознавания психических симптомов и диагностики некоторых психиатрических симптомов.

Актуальная проблема диагностики злокачественных новообразований, возможно, получит новый уровень осмысления с началом применения нейроалгоритмов (80%-ая точность ранней диагностики меланом кожи – одного из самых злокачественных заболеваний).

Одним из серьезных направлений применения нейронных сетей является интерпретация медицинских данных. В последние годы идет бурное развитие новых средств диагностики и лечения. При этом наблюдается «вторая волна» изучения и использования древних, старинных методов и, наоборот, применение последних технических новшеств. При этом встает проблема их грамотной

и корректной интерпретации. Поиск глубинных закономерностей между получаемыми данными и патологическими процессами начинает отставать от разработки все новых и новых методов, поэтому применение для этой цели нейросетей может оказаться чрезвычайно выгодным.

Классической проблемой в кардиологии является интерпретация электрокардиограмм, требующая значительного опыта врача. Сотрудники Университета Глазго (Великобритания) ведут исследования по применению нейросетей для диагностики инфарктов миокарда по результатам анализа электрокардиограмм. Входными данными для сетей являются избранные параметры 12-канальной электрокардиограммы и 12-канальной векторкардиограммы (длины зубцов, расстояния между зубцами).

Исследователи обучили огромное количество нейросетей (167 сетей для диагностики инфаркта миокарда передней стенки и 139 сетей для инфаркта нижней стенки) на массиве данных из 360 электрокардиограмм. Обученные сети затем тестировали отдельную выборку с заранее известными ответами (493 случая). Одновременно для получения отдельной серии ответов на тестируемой выборке был использован логический метод (с заранее заданным алгоритмом). Затем сравнивались результаты тестирования выборки лучшими нейросетями и с помощью логического алгоритма.

Сравнение показало, что во многих случаях чувствительность и специфичность нейросетевого теста оказались выше, чем у логического метода. В тех случаях, когда логический алгоритм решения задачи все-таки можно выстроить, разумно комбинировать в экспертных системах оба подхода.

Интерпретация ЭКГ с помощью нейросетей была применена для диагностики злокачественных желудочковых аритмий. Трехслойная сеть с 230 входными синапсами была обучена на 190 пациентов. Результаты тестирования сравнивались с логическим методом интерпретации данных. Показано, что нейросетевой тест обладает большей чувствительностью (73% по сравнению с 70% для логического метода) и специфичностью (83% по сравнению с 59%).

Актуальным является моделирование работы электрокардиостимуляторов (искусственных водителей ритма) на базе нейронных сетей. Выпускаемые за рубежом электрокардиостимуляторы задают ритм не жестко, а в зависимости от исходного ритма, генерируемого синусовым узлом сердца. Электрокардиостимулятор представляет собой систему вход-преобразование-выход, где входом является ритм синусового узла, выходом – собственный ритм электрокардиостимулятора, а преобразование осуществляется по заданному логическому алгоритму. Авторы смоделировали замену логического преобразователя нейронной сетью, так как взаимоотношения между генерацией импульсов в синусовом узле и требуемым ритмом не линейны и применяемые алгоритмы на практике не всегда эффективны. Нейросеть, обученная на 27 здоровых людях в ситуациях с различной физической нагрузкой, показала гораздо лучшую способность задавать ритм, чем логический алгоритм, применяющийся в электрокардиостимуляторе.

Одной из самых сложных задач для нейросетей в практической медицине является обработка и распознавание сложных образов, например, рентгено-

грамм. Разработана экспертная система интерпретации рентгенограмм груди у новорожденных с выбором одного и более диагнозов из двенадцати.

Созданы нейросетевые экспертные системы для классификации опухолей молочной железы (определения, доброкачественная опухоль, или злокачественная) по данным маммографии (сканограмма молочной железы). Точность диагностики до применения нейросети составляла не более 75%. При тестировании системы, нейросеть, анализирующая сканограмму, давала правильный ответ в 100% случаев. Изображение, получаемое в результате метода, представляется в виде матрицы точек размером 1024x1024 пиксела с 10-битовой шкалой яркости. Изображение подается на нейросеть, имеющую 2 входных, 80 «скрытых» и 2 выходных нейрона. При этом один из выходных нейронов «отвечает» за доброкачественную опухоль, другой – за злокачественную опухоль. Диагноз определяется в зависимости от выходного нейрона, выдавшего больший по величине ответ.

Несколько работ посвящены нейросетевой обработке лабораторных анализов и тестов. Приводится нейросетевой метод интерпретации лабораторных данных биохимического анализа крови. В работе показаны преимущества нейронных сетей в сравнении с линейным дискриминантным анализом, которым параллельно обрабатывались данные.

Особое место среди нейросетевых экспертных систем занимают прогностические модели, применяемые, например, для прогнозирования исходов заболеваний.

В 1990 году американская фирма «Апачи Медикл Системз Инк.» установила в реанимационном отделении одной из больниц штата Мичиган экспертную систему «Апачи–III». Ее цель – прогнозирование исхода заболевания у больных, находящихся в тяжелом состоянии. Для прогноза в компьютер необходимо ввести 27 параметров больного: первичный диагноз, симптомы, степень утраты сознания, наличие или отсутствие других заболеваний.

После этого система выдает вероятность выживания больного в диапазоне от 0 до 100 процентов. Ценность применения системы заключается в том, что она позволяет очень быстро оценить динамику изменения состояния больного, незаметную «на глаз». Например, можно получить ответ у системы до и после введения какого-либо лекарства и, сравнив ответы, выяснить, будет ли наблюдаться эффект от терапии.

Необходимо отметить, что система была обучена на данных, взятых из историй болезней 17448 пациентов, лечившихся в 40 больницах штата в 1989 году. Очевидно, что если качество работы системы обеспечивается таким большим объемом выборки, возможности перенастройки системы не слишком велики. Поэтому данная система не способна к дообучению в процессе работы, опыт «зашит» в нее жестко. Это может быть существенным недостатком при установке программы в регионы, резко отличающиеся по социально-географическим условиям от тех, где проводилось обучение. Кроме того, огромный массив примеров для обучения повышает стоимость программы.

Прогностические нейросетевые модели могут использоваться в демографии и организации здравоохранения.

Судя по литературным данным, именно биологические научные исследования являются наиболее развиваемой областью применения нейросетей. В последнее время биологи, знакомые с исследованиями в области нейроинформатики, приходят к выводу, что многие системы в живых организмах работают по принципам, сходным с алгоритмами нейронных сетей (или наоборот, нейронные сети работают по принципу биосистем). Таким образом, можно наблюдать «взаимное стимулирование» научных разработок в биологии и нейроинформатике.

Эндокринная система человека рассматривается как нейронная сеть из 30 элементов, которые представлены различными гормонами, взаимодействующими друг с другом с помощью прямых и обратных связей. Похожие исследования проводятся для иммунной системы.

Применение нейросетей для исследований в области нейрофизиологии строится на похожих принципах функционирования нейросетей и нервных структур живых организмов. С помощью нейросети осуществлена попытка моделирования простейшей нервной системы.

Анализ публикаций о применении нейросетевых технологий в медицине показывает, что практически отсутствуют какие-либо методологии разработки нейросетевых медицинских систем, о чем свидетельствует как отсутствие работ такого профиля, так и огромное разнообразие подходов к нейросетевым алгоритмам обучения и архитектурам нейронных сетей.

В Санкт-Петербурге (Боткинская больница) разработана нейросетевая ЭС по диагностике некоторых классов болезней, которые плохо диагностируются врачами. Результаты проверки качества работы НЭС свидетельствуют о высокой достоверности результатов (94%).

Данная НЭС разработана на базе нейропакета Neural Planner. Нейронная сеть была обучена при помощи 18 примеров обучающей выборки. Время обучения составляет 2 минуты при заданном пороге ошибки 0,05. Сеть обучилась за 7500 циклов.

После задания пациенту вопросов ответы записываются в бинарном виде: «Да (1)», «Нет (0)». В результате формируется бинарный входной вектор, который предъявляется НЭС, которая определяет класс заболевания у пациента.

Отличием предлагаемого подхода является возможность конструирования экспертных систем самим специалистом, который может передать нейронной сети свой индивидуальный опыт, опыт своих коллег или обучать сеть на реальных данных, полученных путем наблюдений.

В литературе [2,5–10] приведены созданные НЭС: ADAM, GURU, NESTOR, Neural, NEXPERT Object, Smart Elements.

При использовании коммерческих пакетов прикладных нейросетевых программ трудоемкость создания сокращается в десятки раз.

4.9. Достоинства и недостатки нейросетевых систем

4.9.1. Преимущества нейросетевых систем

1. Нейросетевые системы принимают решения на основе опыта и позволяют моделировать ситуацию принятия решения;
2. Создателю нейросетевых систем не требуется устанавливать взаимосвязи между входными данными и необходимым решением, затрачивая время на статистическую обработку, подбор математического аппарата и проверку моделей;
3. Применение нейропакетов позволяет упрощать процесс создания нейросетевых систем;
4. Решение, принимаемое НЭС, не является категоричным, так как она выдает решение со степенью уверенности и оставляет лицу, принимающему решение (ЛПР) критически оценивать ответ;
5. Ответ нейросетевых систем дает очень быстро (доли секунды), что позволяет использовать их в динамических ЭС для незамедлительного принятия решения (например, применение НЭС в реанимации).

4.9.2. Недостатки нейросетевых систем

1. Программные коммерческие нейропакеты сравнительно дорогие (тысячи долларов);
2. Проблема выбора оптимальной архитектуры НС может быть решена только при использовании мощных нейросетевых программ;
3. Отсутствует механизм оптимизации обучающей выборки.

Контрольные вопросы

1. Представление знаний в нейросетевой экспертной системе.
2. Понятие дообучение и переобучение нейронной сети.
3. Сравнение нейросетевых и экспертных систем.
4. Сравнительные характеристики нейропакета.
5. Выбор оптимальной конфигурации нейронной сети.
6. Назначение меры Вапника-Червоненкиса.
7. Рекомендации по выбору архитектуры нейронной сети.
8. Использование разных видов активационных функций.
9. Нормализация входных и выходных данных обучающей выборки.
10. Объяснение эффекта переобучения нейронной сети.
11. Примеры нейросетевых экспертных систем.
12. Проблемы создания нейросетевой экспертной системы.

5. ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

5.1. Обзор коммерческих нейросетевых программ

Основное применение в настоящее время получили программные реализации различных нейросетевых парадигм, называемые нейропакетами (нейроэмуляторами).

Программу моделирования нейронной сети называют программой-имитатором или нейропакетом, понимая под этим определением программную оболочку, эмулирующую для пользователя среду нейрокомпьютера на типовом компьютере [2, 5, 10].

В настоящее время на рынке программного обеспечения имеется множество разнообразных программ для моделирования нейронных сетей, которые обеспечивают их создание, обучение и тестирование.

Пакеты прикладных программ условно можно разделить на 2 класса:

1. Пакеты нейросетевых программ универсального назначения (MatLab, Statistica, NeuralWorks Professional II/Plus, NeuroSolutions, NeuroShell 2 нейросетевая библиотека Python и т. п.);
2. Специализированные пакеты нейросетевых программ, ориентированные на конкретную область применения и ограниченную архитектуру нейронной сети (Neural Planner, Neuro Office, NeuroPro, и т. п.).

Преимущества таких «виртуальных» нейрокомпьютеров для относительно небольших задач очевидны:

– во-первых, не надо тратиться на новую аппаратуру, если можно загрузить уже имеющиеся компьютеры общего назначения;

– во-вторых, пользователь не должен осваивать особенности программирования на специализированных процессорах и способы их сопряжения с базовым компьютером;

– универсальные ЭВМ не накладывают никаких ограничений на структуру сетей и способы их обучения, тогда как специализированные процессоры зачастую имеют ограниченный набор «зашитых» в них функций активации и достигают пиковой производительности лишь на определенном круге задач.

Таблица 5.1

Преимущества и недостатки нейросетевых программных продуктов

Сегмент рынка нейронных продуктов	**Преимущества**	**Недостатки**
Нейронные пакеты общего назначения	Не требуют программирования, легко осваиваются, пригодны для решения прикладных задач	Не способны к расширению, не могут использоваться для разработки сложных систем

Окончание табл. 5.1

Сегмент рынка нейронных продуктов	Преимущества	Недостатки
Системы разработки нейронных приложений	Могут использоваться для создания сложных систем обработки данных в реальном времени	Требуют навыков программирования, более глубокого знания нейросетей
Готовые решения на основе нейросетей	Предоставляют комплексное решение проблемы	Высокая стоимость

Универсальный нейропакет NeuroSolutions фирмы NeuroDimension, Inc. предназначен для моделирования широкого круга искусственных нейронных сетей [1]. Основное достоинство описываемого нейропакета состоит в его гибкости: помимо традиционно используемых нейросетевых парадигм (типа полносвязных многослойных нейронных сетей или самоорганизующихся карт Кохонена) нейропакет включает в себя мощный редактор визуального проектирования нейронной сети, позволяющий создавать практически любые собственные нейронные структуры и алгоритмы их обучения. Особо следует отметить, что данный нейропакет позволяет пользователю вводить собственные критерии обучения нейронной сети, не ограничивая его только широко распространённым, но далеко не самым оптимальным критерием минимума среднеквадратичной ошибки.

Нейропакет NeuroSolutions снабжен мощными и хорошо продуманными средствами визуализации: отображать и визуально контролировать можно многое, начиная от структуры нейронной сети и кончая процессом и результатом обучения. Наличие мощных средств визуализации выводит нейропакет на уровень CAD-систем (систем автоматизированного проектирования), т. е. NeuroSolutions можно считать системой проектирования и моделирования нейронной сети.

NeuroSolutions предназначен для моделирования нейронных сетей. К достоинствам пакета можно отнести гибкость. Помимо традиционных нейросетевых архитектур (полносвязных и многослойных НС, самоорганизующихся карт Кохонена) нейропакет включает редактор визуального проектирования, позволяющий создавать произвольные архитектуры нейронных сетей и алгоритмов их обучения. В результате NeuroSolutions рассматривается на уровне CAD-систем (систем автоматизированного проектирования) моделирования и проектирования НС.

В пакете реализуется большой перечень нейронов, включая взвешенный сумматор (нейрон первого порядка), нейроны высших порядков (с перемножением входов), а также непрерывный интегрирующий нейрон. Функция активации выбирается из пяти стандартных (кусочно-линейная, функции знака, и три типа сигмоидальных) функций и заданная пользователем. Связи между нейронами задаются произвольно на этапе проектирования и могут быть изменены

в процессе работы. Поддерживаются все типы связей: прямые, перекрестные и обратные. При этом хорошо реализована схема организации связей: можно задать одну векторную связь с заданной весовой матрицей, а не набор скалярных связей с весовыми коэффициентами.

Нейропакет NeuroSolutions содержит мощные средства для организации обучающих выборок. Встроенные конверторы данных поддерживают графические изображения в формате BMP, текстовые файлы с числовыми или символьными данными, а также функции непрерывного аргумента (например, времени), заданные в аналитическом виде или в виде выборки значений. Нейропакет позволяет использовать любые внешние конверторы данных.

На этапе обучения может быть использован широкий круг критериев обучения, как дискретных, так и непрерывных. Помимо этого, можно вводить собственные критерии. Можно использовать как встроенный алгоритм обучения типа back-propagation или дельта-правила, так и использовать собственный. Система визуализации процесса обучения позволяет проводить анализ изменения весов непосредственно в процессе обучения и вносить коррективы. Может быть введена шумовая характеристика как при тестировании, так и при обучении нейронной сети. Можно задать аддитивный белый шум, шум произвольной природы, а также любой заданный тип шума (например, белый мультипликативный).

Neurosolutions содержит генератор (мастер) стандартных нейросетевых архитектор (Neural Wizard), с помощью которого быстро задается архитектура, подбирается обучающая выборка, критерии и методы обучения нейронной сети.

NeuralWorks Professional II/Plus [2,10]. В отличие от NeuroSolutions в пакете *NeuralWorks Professional II/Plus* (фирма NeuralWare, Inc.) основной упор сделан на применение стандартных нейронных парадигм и алгоритмов обучения и в этом данный пакет превосходит все остальные. В нем реализованы 28 стандартных нейронных парадигм, используемых при решении прикладных задач.

NeuralWorks Professional имеет хорошо организованную систему визуализации данных. Можно просмотреть структуру нейронной сети, изменение весовых коэффициентов в процессе обучения, изменение ошибки обучения, а также корреляцию весов нейронной сети при обучении. Последнее является уникальной возможностью, представляемой только пакетом NeuralWorks Professional и весьма полезной при анализе поведения нейронной сети при обучении и работе.

NeuralWorks Professional представляет собой открытую систему, в которую можно интегрировать внешние программные модули, написанные пользователями. Пакет имеет встроенный генератор кода, поддерживающий компилятор Microsoft Visual C++.

Нейропакет Process Advisor [2,5]. Нейропакет *Process Advisor* (фирма Al Ware, Inc) создавался для решения задач управления динамическими процессами (в частности, технологическими).

В нейропакете реализована только многослойная нейронная сеть прямого распространения, обучаемая с помощью модифицированного алгоритма обрат-

ного распространения ошибки (backpropagation error). Введены возможность работы с динамическими процессами. В частности, в Process Advisor возможна работа с входными сигналами как с функциями времени, а не только как с дискретным набором точек. Помимо Process Advisor, такую возможность предоставляет только пакет NeuroSolutions. Кроме того нейропакет Process Advisor позволяет осуществлять управление внешними аппаратными контроллерами, подключаемыми к компьютеру.

Нейропакет *NeuroShell 2* (фирма Ward Systems Group) является одной из трех программ, входящих в состав пакета The AI Trilogy. Он представляет собой универсальный нейропакет, предназначенный для моделирования нескольких наиболее известных нейронных парадигм: многослойных нейронных сетей, сети Кохонена и других. Пакет NeuroShell 2 [2, 5] проигрывает по сравнению с NeuroSolutions и NeuralWorks.

Система для профессионала позволяет опытным пользователям создавать 16 различных видов нейросетей со значительно большими возможностями установки и контроля их параметров, чем в системе для начинающего.

Средства автономного использования, входящие в состав пакета NeuroShell 2, позволяют пользователю использовать свою созданную нейронную сеть как динамическую библиотеку (DLL), которая может быть вызвана из других программ или из Microsoft Excel. Вы можете также осуществить генерацию программного кода на Си или Visual Basic для созданных Вами сетей. Нейросети, которые Вы создали с помощью NeuroShell 2, Вы можете распространять без каких-либо ограничений и уплаты роялти. Для использования нейронной сети вне NeuroShell 2 используется модуль Генератор автономных файлов.

Нейропакет NeuroShell 2 имеет и недостаточно продуманную систему визуализации данных: контролировать можно многие параметры, но в разных режимах работы нейропакета. Из-за отсутствия единого интегрального контроля данных в процессе обучения или работы нейронной сети часто приходится переключаться из одного режима в другой, что отнимает много времени и весьма неудобно в использовании.

К особенностям нейропакета следует отнести жестко реализованную последовательность действий при работе с нейронной сетью. Так, невозможно определить структуру нейронной сети до того, как заданы входные данные. С одной стороны, это очень удобно, особенно для начинающих пользователей, поскольку сразу становится ясно, что и в какой последовательности следует делать. С другой стороны, более опытного пользователя такая жесткая зафиксированная последовательность действий утомляет: для того, чтобы внести в нейронную сеть небольшое изменение, приходится выполнять всю цепочку действий. Таким образом, можно сказать, что пакет NeuroShell 2 удобен для начинающих пользователей.

Обычно процесс анализа данных начинается с подготовки данных. Пользователь может ввести данные вручную или импортировать данные из файлов. Нейросетевые архитектуры в программе NeuroShell 2. приведены на рис. 5.1

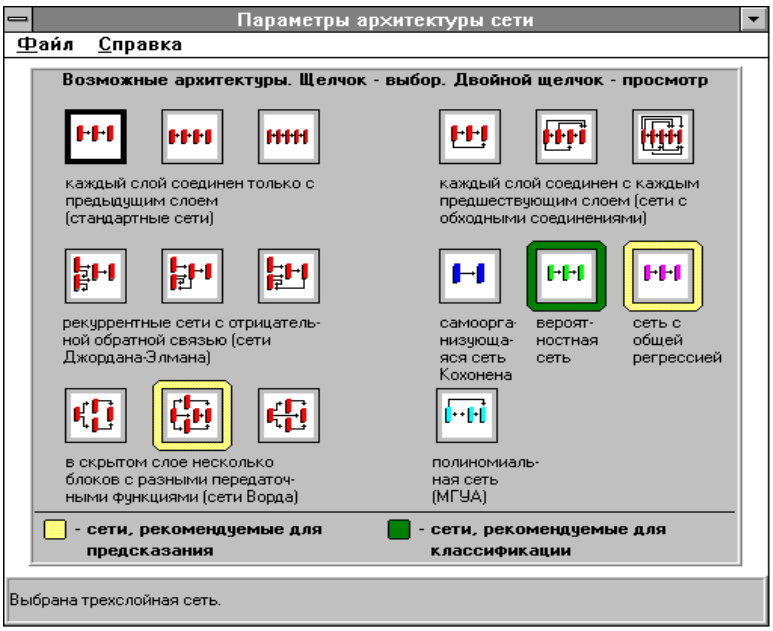


Рис.5.1. Выбор типа архитектуры в программе NeuroShell 2

Модуль «Проектирование» NeuroShell 2 предлагает на выбор 16 различных нейросетей, приведённых на рис. 9.1. Для каждой из них возможны несколько методов обучения. Чтобы установить параметры нейросети, Вам нужно просто указать курсором на блок или связь и щелкнуть мышкой.

В NeuroShell 2 можно использовать сети, реализующие следующие нейросетевые парадигмы: сети с обратным распространением ошибки; сети Кохонена; вероятностные нейронные сети; нейронные сети с общей регрессией; полиномиальные сети.

Нейропакет *BrainMaker Pro* (фирма California Scientific Software) является достаточно простым при моделировании многослойных нейронных сетей, обучаемых с помощью алгоритма обратного распространения ошибки. Основным достоинством нейропакета BrainMaker Pro [2, 5] можно считать большое число параметров настройки алгоритма обучения нейронных сетей, в том числе возможность обучения с ограничениями на весовые коэффициенты.

Большинство приложений, созданных красноярской группой «Нейро-Комп» связано с решением задач классификации, технической диагностики, космической навигации и др. Среди них программа NeuroPro [1]. Она является свободно распространяемой альфа-версией нейросетевого программного продукта для моделирования многослойных нейронных сетей без обратных связей с числом слоев до 10, а нейронов в слое – до 100 и одним видом нелинейной функции активации. Обучение нейронной сети производится с применением одного из следующих методов оптимизации: градиентного спуска; модифицированного ParTan-метода; метода сопряженных градиентов. Предусмотрено упрощение (контрастирование) нейронной сети следующими методами: сокращение числа входных сигналов сети; сокращение числа синапсов; сокращение числа неоднородных входов сети; сокращение числа нейронов; бинаризация весов сети. Применение данного программного продукта возможно в традицион-

ных областях: медицина, экология, климатология, метеорология, прогнозирование курсов акций и т. п.

На основании анализа оценок нейропакетов можно сделать вывод, что наиболее мощными, универсальными и простыми в использовании являются нейропакеты *NeuroSolutions* и в меньшей степени NeuralWorks Professional.

5.2. Нейронные сети в пакете MatLab

MATLAB (сокращение от англ. *«Matrix Laboratory»*, в русском языке произносится как Матлáб) – пакет прикладных программ для решения задач технических вычислений и одноимённый язык программирования, используемый в этом пакете[9]. Пакет используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, Mac OS, и Windows.

MATLAB как язык программирования был разработан Кливом Моулером (англ. Cleve Moler) в конце 1970-х годов когда он был деканом факультета компьютерных наук в Университете Нью-Мексико. Целью разработки служила задача дать студентам факультета возможность использования программных библиотек Linpack и EISPACK без необходимости изучения Фортрана. Вскоре новый язык распространился среди других университетов и был с большим интересом встречен учёными, работающими в области прикладной математики. До сих пор в Интернете можно найти версию 1982 года, написанную на Фортране, распространяемую с открытым исходным кодом. Инженер Джон Литтл (англ. John N. (Jack) Little) познакомился с этим языком во время визита Клива Моулера в Стэнфордский университет в 1983 году. Поняв, что новый язык обладает большим коммерческим потенциалом, он объединился с Кливом Моулером и Стивом Бангертом (англ. Steve Bangert). Совместными усилиями они переписали MATLAB на C и основали в 1984 году компанию The MathWorks для дальнейшего развития. Эти переписанные на C библиотеки долгое время были известны под именем JACKPAC. Первоначально MATLAB предназначался для проектирования систем управления (основная специальность Джона Литтла), но быстро завоевал популярность во многих других научных и инженерных областях. Он также широко использовался и в образовании, в частности, для преподавания линейной алгебры и численных методов.

MATLAB как язык программирования был разработан Кливом Моулером (англ. Cleve Moler) в конце 1970-х годов когда он был деканом факультета компьютерных наук в Университете Нью-Мексико. Целью разработки служила задача дать студентам факультета возможность использования программных библиотек Linpack и EISPACK без необходимости изучения Фортрана. Вскоре новый язык распространился среди других университе-

[9] «Аугментация (augmentation, «раздутие») данных для обучения нейронной сети на примере печатных символов», август 2015. https://habr.com/company/smartengines/blog/264677/

тов и был с большим интересом встречен учёными, работающими в области прикладной математики. До сих пор в Интернете можно найти версию 1982 года, написанную на Фортране, распространяемую с открытым исходным кодом. Инженер Джон Литтл (англ. John N. (Jack) Little) познакомился с этим языком во время визита Клива Моулера в Стэнфордский университет в 1983 году. Поняв, что новый язык обладает большим коммерческим потенциалом, он объединился с Кливом Моулером и Стивом Бангертом (англ. Steve Bangert). Совместными усилиями они переписали MATLAB на C и основали в 1984 компанию The MathWorks для дальнейшего развития. Эти переписанные на C библиотеки долгое время были известны под именем JACKPAC. Первоначально MATLAB предназначался для проектирования систем управления (основная специальность Джона Литтла), но быстро завоевал популярность во многих других научных и инженерных областях. Он также широко использовался и в образовании, в частности, для преподавания линейной алгебры и численных методов.

Для MATLAB имеется возможность создавать специальные наборы инструментов (англ. toolbox), расширяющие его функциональность. Наборы инструментов представляют собой коллекции функций и объектов, написанных на языке MATLAB для решения определённого класса задач. Компания Mathworks поставляет наборы инструментов, которые используются во многих областях, включая следующие:

- Цифровая обработка сигналов, изображений и данных: *Signal Processing Toolbox, DSP System Toolbox, Image Processing Toolbox, Wavelet Toolbox, Communications System Toolbox* – наборы функций и объектов, позволяющих решать широкий спектр задач обработки сигналов, изображений, проектирования цифровых фильтров и систем связи.
- Системы управления: *Control Systems Toolbox, Robust Control Toolbox, System Identification Toolbox, Model Predictive Control Toolbox, Model-Based Calibration Toolbox* – наборы функций и объектов, облегчающих анализ и синтез динамических систем, проектирование, моделирование и идентификацию систем управления, включая современные алгоритмы управления, такие как робастное управление и другие.
- Финансовый анализ: *Econometrics Toolbox, Financial Instruments Toolbox, Financial Toolbox, Datafeed Toolbox, Trading Toolbox* – наборы функций и объектов, позволяющие быстро и эффективно собирать, обрабатывать и передавать различную финансовую информацию.
- Анализ и синтез географических карт, включая трёхмерные: *Mapping Toolbox*.
- Сбор и анализ экспериментальных данных: *Data Acquisition Toolbox, Image Acquisition Toolbox, Instrument Control Toolbox, OPC Toolbox* – наборы функций и объектов, позволяющих сохранять и обрабатывать данные, полученные в ходе экспериментов, в том числе в реальном времени. Поддерживается широкий спектр научного и инженерного измерительного оборудования.

– Визуализация и представление данных: *Virtual Reality Toolbox* – позволяет создавать интерактивные миры и визуализировать научную информацию с помощью технологий виртуальной реальности и языка VRML.

– Средства разработки: *MATLAB Builder for COM, MATLAB Builder for Excel, MATLAB Builder for NET, MATLAB Compiler, HDL Coder* – инструменты, позволяющие создавать независимые приложения из среды MATLAB.

– Взаимодействие с внешними программными продуктами: *MATLAB Report Generator, Excel Link, Database Toolbox, MATLAB Web Server, Link for ModelSim* – наборы функций, позволяющие сохранять данные различных видов таким образом, чтобы другие программы могли с ними работать.

– Базы данных: *Database Toolbox* – инструменты работы с базами данных.

– Научные и математические пакеты: *Bioinformatics Toolbox, Curve Fitting Toolbox, Fixed-Point Toolbox, Optimization Toolbox, Global Optimization Toolbox, Partial Differential Equation Toolbox, Statistics And Machine Learning Toolbox, RF Toolbox* – наборы специализированных математических функций и объектов, позволяющие решать широкий спектр научных и инженерных задач, включая разработку генетических алгоритмов, решения задач в частных производных, целочисленные проблемы, оптимизацию систем и другие [6].

1. Нейронные сети: *Neural Network Toolbox* – инструменты для синтеза и анализа нейронных сетей.
2. Нечёткая логика: *Fuzzy Logic Toolbox* – инструменты для построения и анализа нечётких множеств.
3. Символьные вычисления: *Symbolic Math Toolbox* – инструменты для символьных вычислений с возможностью взаимодействия с символьным процессором программы Maple.

Помимо вышеперечисленных, существуют тысячи других наборов инструментов для MATLAB, написанных другими компаниями и энтузиастами.

Neural Network Toolbox – это пакет расширения MatLab, содержащий средства для проектирования, моделирования, разработки и визуализации нейронных сетей. Пакет обеспечивает всестороннюю поддержку типовых нейросетевых парадигм и имеет открытую модульную архитектуру. Пакет содержит функции командной строки и графический интерфейс пользователя для быстрого пошагового создания нейросетей [2, 5].

Кроме этого Neural Network Toolbox обеспечивает поддержку Simulink, что позволяет моделировать нейросети и создавать блоки на основе разработанных нейросетевых структур. В состав пакета входят более 150 различных функций, образуя собой своеобразный макроязык программирования и позволяя пользователю создавать, обучать и тестировать самые различные нейронные сети.

MatLab – это высокоуровневый язык и интерактивная среда для программирования, численных расчетов и визуализации результатов. С помощью MatLab можно анализировать данные, разрабатывать алгоритмы, создавать модели и приложения.

Язык, инструментарий и встроенные математические функции позволяют вам исследовать различные подходы и получать решение быстрее, чем с использованием электронных таблиц или традиционных языков программирования, таких как C/C++ или Java. MatLab широко используется в таких областях, как:

- обработка сигналов и связь;
- обработка изображений и видео;
- системы управления;
- автоматизация тестирования и измерений;
- финансовый инжиниринг;
- вычислительная биология и т. п.

Более миллиона инженеров и ученых по всем миру используют MatLab в качестве языка технических вычислений.

Обучение ИНС заключается в определении сетевой архитектуры и корректировке весовых отношений для конкретной задачи. Чтобы настроить вес ссылок в нейронной сети, используется образец обучения. По мере улучшения итерации весов соответственно улучшается функционирование сети.

Он должен начинаться с определения модели внешней среды, в которой работает ИНС, поэтому устанавливается необходимая информация для обучения. Кроме того, вам необходимо установить способ изменения параметров сетевого веса. Таким образом, сам алгоритм обучения может быть представлен в виде процедуры, которая использует правила обучения для корректировки весов.

В настоящее время при обучении нейронных сетей используют три правила: с учителем, без учителя (самообучение) и смешанная. Нейронная сеть располагает правильными ответами (выходами сети) на каждый входной пример в первом типе обучения. Настраиваются веса так, чтобы сеть производила ответы близкие к правильным ответам.

Графический интерфейс пользователя NNToolBox системы MatLab позволяет выбирать структуры ИНС из обширного перечня и предоставляет множество алгоритмов обучения для каждого типа сети.

В процессе обучения сети входные данные подаются на его входы, а значение, полученное на выходе, сравнивается с целевым (желаемым). Исходя из результата сравнения, вычисляются значения изменения весов и смещения, которые уменьшают это отклонение.

Теперь следует приступить к созданию нейронной сети. Для этого в рабочее окно MatLab следует ввести команду nnstart затем выбрать опцию Fitting app, после чего откроется графический интерфейс для работы с нейронной сетью, где необходимо выбрать обучающие матрицы.

В нашем случае в качестве входных данных мы выбрали из исходных данных значения изменения индекса за первые пять дней, которые поместили в матрицу под названием Data. В качестве же целевых данных выступит матрица Exit, которая содержит значения индекса на момент шестого дня. Таким образом наша нейронная сеть сможет делать прогнозы основываясь на данных предыдущих пяти дней.

Следующий шаг заключается в распределении входных и целевых векторов в случайном порядке данные на три набора в процентном соотношении. В нашем случае:

- 70% векторов используются для обучения;
- 15% векторов – для проверки достоверности результатов и исключения эффекта переобучения сети;
- 15% векторов используются для независимого испытания сети.

На следующей странице отображается структура сети (рис. 5.2). Здесь используется однонаправленная нейронная сеть с обратным распространением ошибок и сигмоидальной функцией передачи в скрытом слое и линейной функцией передачи в выходном слое. Так же на данном этапе можно задать количество нейронов на скрытом уровне нейронной сети. Количество нейронов, используемых в скрытом слое зависит от сложности решаемой задачи, в нашем случае мы выбрали 50 нейронов на скрытом уровне.

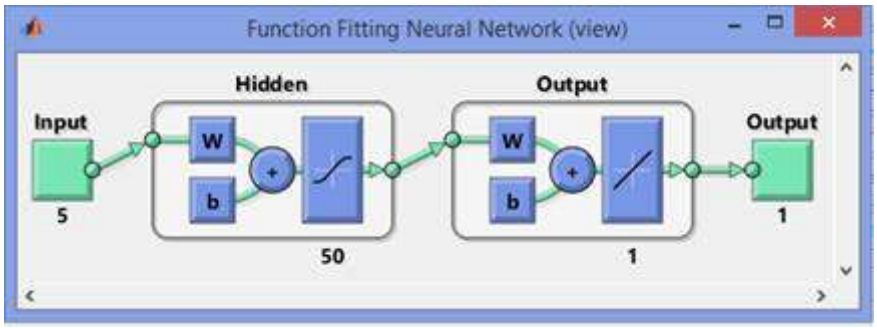


Рис. 5.2. Структура нейронной сети

Следующий шаг непосредственно само обучение сети. В рабочем окне Neural Network Toolbox нажимаем на кнопку Train, далее открывается окно обучения. Оно показывает процесс тренировки сети и позволяет его прервать, нажав кнопку «Stop Training». В качестве алгоритма обучения данная нейронная сеть использует алгоритм Левенберга-Марквардта.

Алгоритм Левенберга-Марквардта предназначен для оптимизации параметров моделей нелинейной регрессии. Предполагается, что в качестве критерия оптимизации будет использоваться среднеквадратичная ошибка модели на обучающем образце. Алгоритм состоит в последовательном приближении заданных начальных значений параметров к желаемому локальному минимуму.

После завершения обучения в открывшемся окне Train станет доступен график, который демонстрирует нам сколько эпох длилось обучение сети и как менялась среднеквадратичная ошибка проверки достоверности результатов по ряду показателей (рис. 5.3). В нашем случае обучение нейросети было остановлено на второй эпохе и среднеквадратичная ошибка проверки достоверности результатов была равна 0,00756.

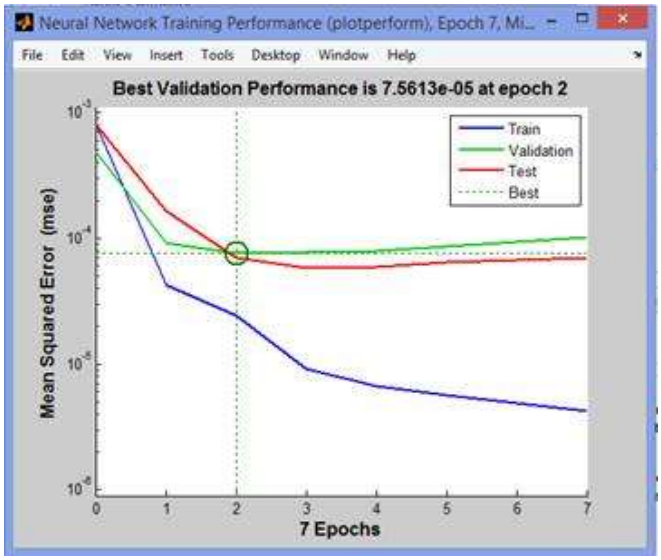


Рис. 5.3. График значений ошибок записи обучения в зависимости от количества учебных эпох

В этом примере, результат адекватен из следующих соображений:

1. Заключительная среднеквадратичная ошибка (СКО) мала.
2. Ошибка проверочного (Validation) и тестового (Test) наборов утверждения имеют подобные характеристики.
3. Переобучения не произошло, так как после точки остановки происходит лишь увеличение СКО проверочного набора до 7 эпохи.

После окончания процесса обучения, следует проверит правильность решения проблемы путем запуска и тестирования только что обученной сети. Для проверки будем использовать контрольный набор данных, которые не использовались для обучения искусственной нейронной сети.

Проведем тестирование и сравним данные. Сравнивая данные, выданные системой, и реальные данные, можно убедиться, что нейронная сеть действительно делает прогнозы, приближенные к реальности.

Рекомендуется изменить модель нейронной сети, если в результате последовательных шагов в обучении и контроле ошибка остается неприемлемо большой (можно увеличить количество нейронов на скрытом слое или сменить тип сети). Так же рекомендуется использовать специальный тестовый набор данных, который будет своеобразным образцом независимым от входных данных. Проводя тестирование на таком наборе, появляется возможность проверки надежности результатов обучения ИНС. Однако, такой набор можно использовать только один раз, так как его легко преобразовать в набор управления, который используется для настройки сети.

5.3. Нейросетевой пакет STATISTICA Neural Networks

Нейросетевой пакет STATISTICA Neural Networks (STNN) – это набор мощнейших методов анализа данных, визуализации, прогнозирования, добычи данных, и контроля качества, ориентированные в основном на анализ данных.

STATISTICA Автоматизированные Нейронные Сети – богатая, современная, мощная и чрезвычайно быстрая среда анализа нейросетевых моделей, которая предоставляет удобные и простые в использовании методы для статистического анализа, наглядного графического представления данных, создания собственных пользовательских приложений, интеграции, совместной работы, web-доступа и др. Интерфейс прикладной системы программирования STATISTICA Neural Networks API позволяет интегрировать разрабатываемые нейросетевые приложения в уже существующие вычислительные среды (рис. 5.5).

STATISTICA Automated Neural Networks – единственный в мире нейросетевой программный продукт, полностью переведенный на русский язык. Методологии нейронных сетей получают все большее распространение в самых различных областях деятельности от фундаментальных исследований до практических приложений анализа, данных, бизнеса, промышленности и др.

STATISTICA Automated Neural Networks (SANN) является одним из самых передовых и самых эффективных нейросетевых продуктов на рынке. Он предлагает множество уникальных преимуществ и богатых возможностей. Например, уникальные возможности инструмента автоматического нейросетевого поиска, *Автоматизированная нейронная сеть (АНС)*, позволяют использовать систему не только экспертам по нейронным сетям, но и новичкам в области нейросетевых вычислений.

Все данные, участвующие в создании, обучении и тестировании, делятся на четыре группы (множества): обучающие, контрольные, тестовые и не учитываемые. Обучающее множество служит для обучения НС, контрольное – для независимой оценки хода обучения и исключения эффекта переобучения сети, тестовое – для окончательной оценки после завершения серии экспериментов. Не учитываемое множество используется в тех случаях, когда часть данных испорчена, ненадежна.

Для обучения многослойных персептронов используются пять различных алгоритмов обучения: обратного распространения, быстрые методы второго порядка – спуск по сопряженным градиентам и Левенберга-Марккварта, методы быстрого распространения и «дельта-дельта с чертой».

Рис. 5.5. Нейронные сети для решения задач в пакете Statistica

Типы нейронных сетей представлены на рис.5.6.

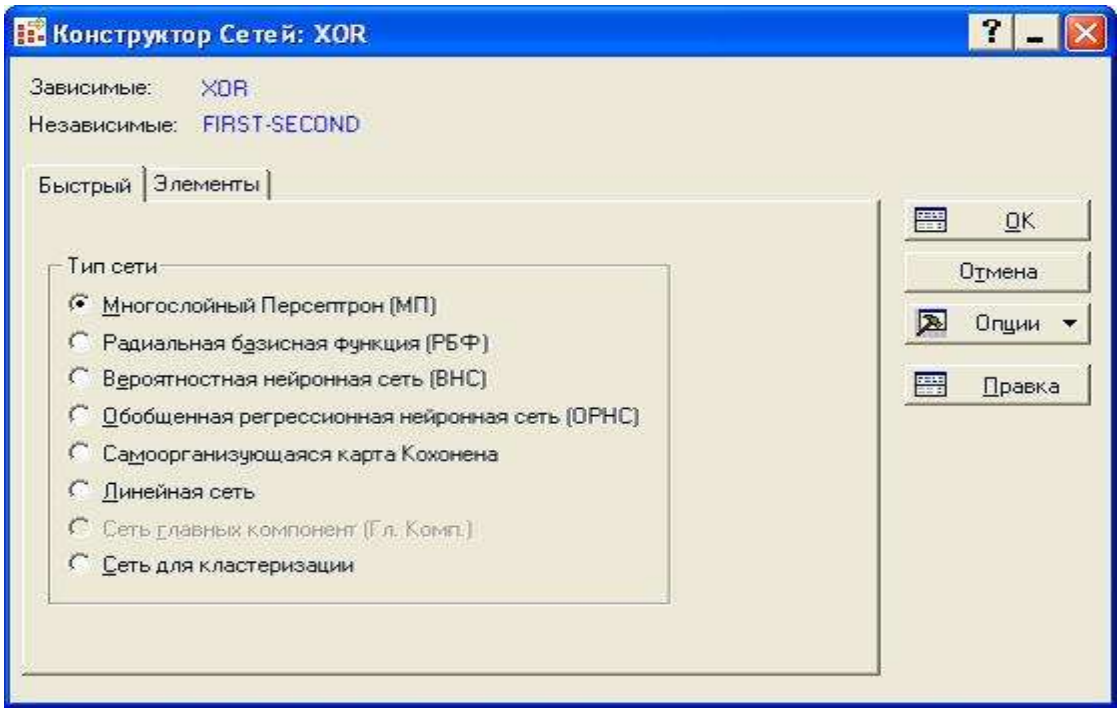


Рис. 5.6. Типы нейронных сетей в пакете Statistica

Рассмотрим основные преимущества использования *STATISTICA Automated Neural Networks.*

1. Пре- и пост-процессирование, включая выбор данных, кодирование номинальных значений, шкалирование, нормализацию, удаление пропущенных данных с интерпретацией для классификации, регрессию и задачи временных рядов.
2. Простота в использовании и аналитическая мощность.
3. Самые современные, оптимизированные и мощные алгоритмы обучения сети (включая методы сопряженных градиентов, алгоритм Левенберга-Марквардта, BFGS, алгоритм Кохонена); полный контроль над всеми параметрами, влияющими на качество сети, такими как функции активации и ошибок, сложность сети (см. рис. 5.7).

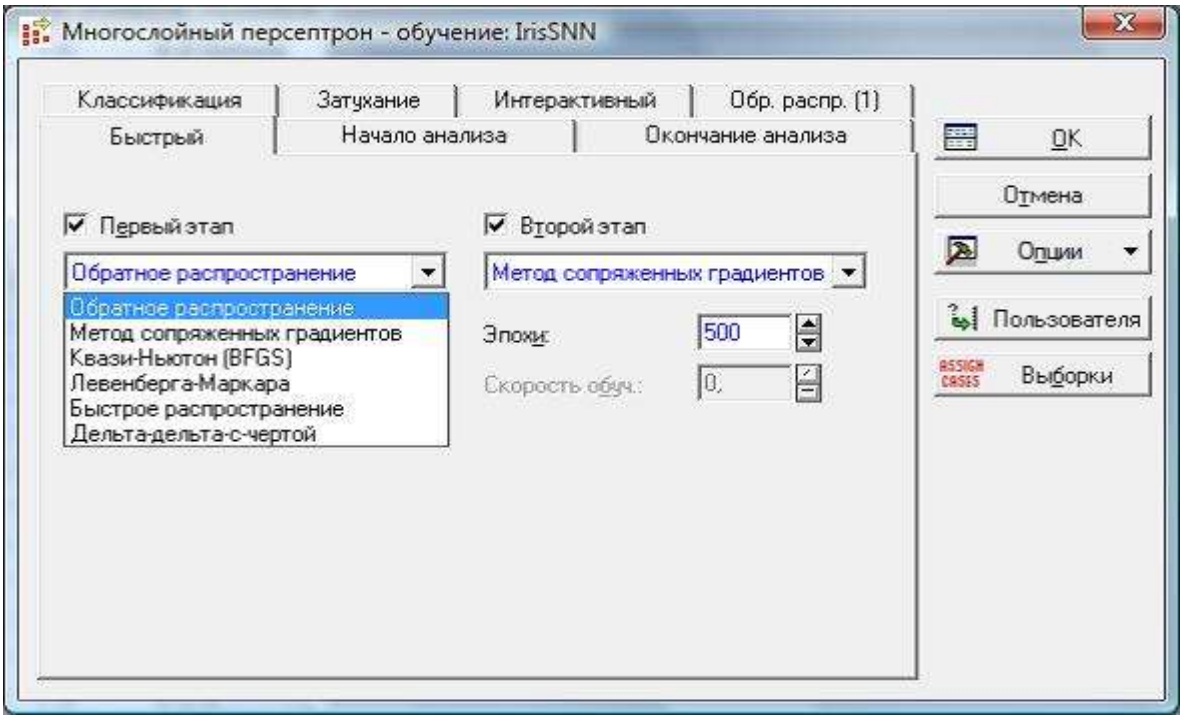


Рис. 5.7. Алгоритмы обучения нейронных сетей

4. Поддержка ансамблей нейросетей и нейросетевых архитектур практически неограниченного размера.
5. Богатые графические и статистические возможности, которые облегчают интерактивный исследовательский анализ.
6. Полная интеграция с системой *STATISTICA*; все результаты, графики, отчеты и т. д. могут быть в дальнейшем модифицированы с помощью мощных графических и аналитических инструментов *STATISTICA* (например, для проведения анализа предсказанных остатков, создания подробного отчета и т. п.).
7. Полная интеграция с мощными автоматическими инструментами *STATISTICA*; запись полноценных макросов для любых анализов; создание собственных нейросетевых анализов и приложений с помощью *STATISTICA* Visual Basic, вызов *STATISTICA Automated Neural Networks* из любого приложения, поддерживающего технологию COM (например, автоматическое проведение нейросе-

тевого анализа в таблице MS Excel или объединение нескольких пользовательских приложений, написанных на языках C, C++, C#, Java и т. д.).

8. Выбор наиболее популярных сетевых архитектур, включая Многослойные персептроны, Радиальные базисные функции и Самоорганизующиеся карты признаков.

9. Имеется инструмент *Автоматического Сетевого Поиска*, позволяющий в автоматическом режиме строить различные нейросетевые архитектуры и регулировать их сложность.

10. Сохранение наилучших нейронных сетей.

11. Поддержка различного рода статистического анализа и построение прогнозирующих моделей, включая регрессию, классификацию, временные ряды с непрерывной и категориальной зависимой переменной, кластерный анализ для снижения размерности и визуализации.

12. Поддержка загрузки и анализа нескольких моделей.

13. Опциональная возможность генерации исходного кода на языках C, C++, C#, Java, PMML (Predictive Model Markup Language), который может быть легко интегрирован во внешнюю среду для создания собственных приложений.

Имеется генератор кода *STATISTICA Автоматизированные Нейронные Сети,* который может сгенерировать исходный системный программный код нейросетевых моделей на языках C, Java и PMML (Predictive Model Markup Language). Генератор кода является дополнительным приложением к системе *STATISTICA Автоматизированные Нейронные Сети*, которое позволяет пользователям на основе проведенного нейросетевого анализа генерировать C или Java-файл с исходным кодом моделей и интегрировать его в независимые внешние приложения.

Ориентированность на анализ данных и наличие русскоязычной версии позволяет быстро решить поставленную задачу. В качестве основных моделей нейронных сетей используются многослойные нейронные сети и сети радиально-базисных функций. Достоинством системы *STATISTICA Автоматизированные Нейронные Сети* можно отметить выбор архитектуры сети и применение обучающих, контрольных и тестовых подмножеств для решения проблемы переобучения нейронной сети. В качестве функций активации можно выбрать следующие: тождественная, логистическая, гиперболическая, экспонента и синус (см.рис. 5.8).

С 2008 года StatSoft является золотым партнером Microsoft (Microsoft Gold Partnership). Statistica полностью соответствует стандартам Microsoft, включая OLE DB и DDE. Это позволяет интегрировать новые модули в существующие системы и строить на основе Statistica интеллектуальную систему принятия решений, используя процедуры Statistica как готовые элементы.

Statistica позволяет напрямую производить импорт/экспорт данных из Microsoft Office, работать в Microsoft Excel «внутри» Statistica, автоматически сохранять результаты в Microsoft Word.

Statistica взаимодействует с любыми реляционными базами данных (Oracle, MS SQL Server, Informix, Access и др.), хранилищами бизнес-информации SAP Business Warehouse и обеспечивает интеграцию с языком R.

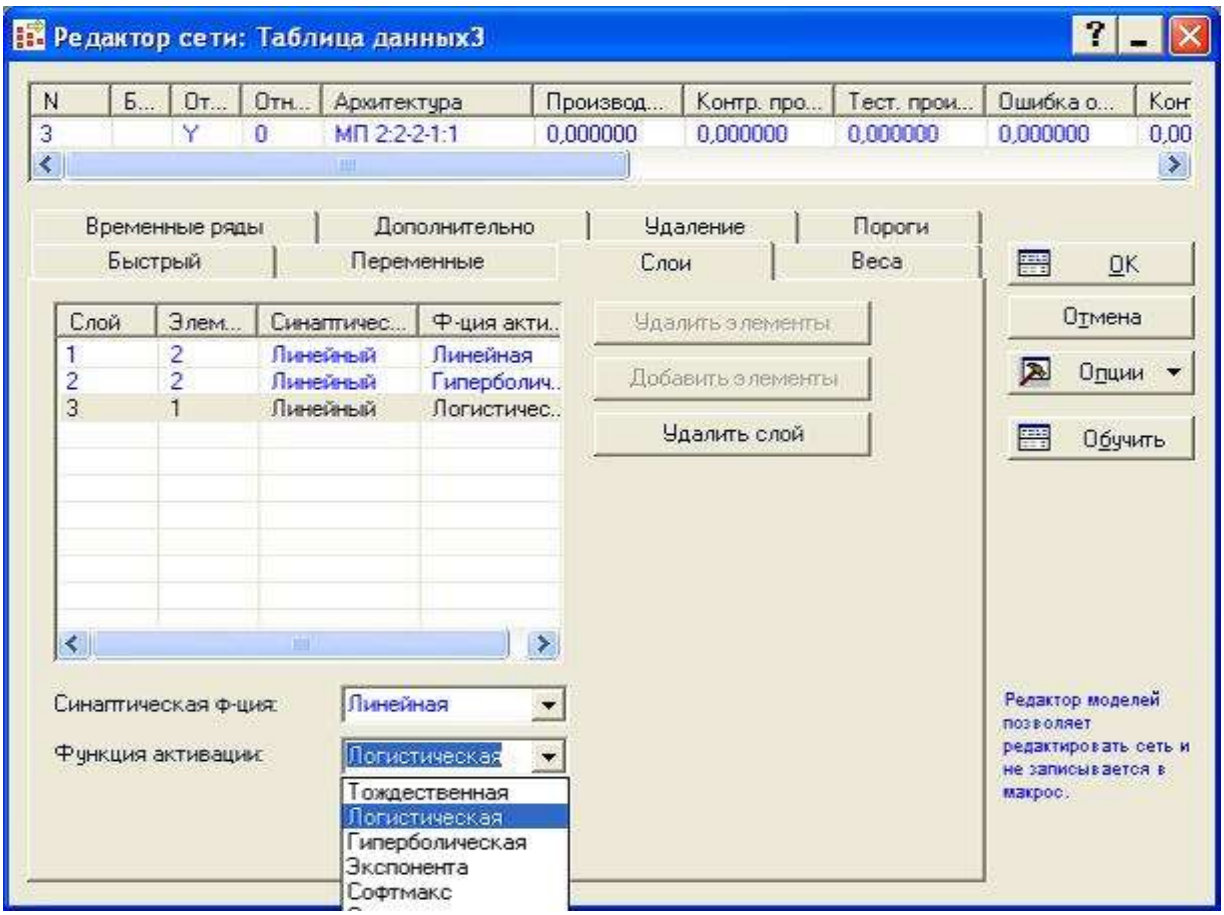


Рис. 5.8. Выбор функции активации

Statistica взаимодействует с Microsoft SharePoint, позволяет выгружать данные из OSI PI. Поддерживает OLAP технологии и Web-технологии.

Полный COM ориентированный интерфейс среды Statistica Visual Basic (SVB) для всех функций и процедур, автоматическая запись макросов позволяют создавать пользовательские приложения и надстройки над Statistica для автоматизации любых еженедельных или длительных процедур. Доступ ко всем свойствам через объектную модель, мощный отладчик процедур, браузер функций и т.д. позволяют создавать необходимые приложения максимально быстро. SVB интегрирован со множеством приложений (таких как MS Excel) и различными языками программирования (C++, Java и др.).

Корпоративные версии Statistica полностью web-интегрированы: «ввод» и «вывод» данных через Web, направление результатов на Web-сервер, построение сложных автоматизированных систем, работающих с данными из внешних источников, проведение анализов и обновление содержания HTML-страниц на Web-сервере. Возможность пакетной обработки данных. Использование многоуровневой архитектуры клиент-сервер при наличии русифицированной версии.

5.4. Пакет Python с нейросетевыми библиотеками PyBrain

В последнее время широкой популярностью пользуется пакет Python с библиотекой нейросевых программ PyBrain для решения задач анализа данных [23–34].

PyBrain – одна из лучших Python библиотек для изучения и реализации большого количества разнообразных алгоритмов связанных с нейронными сетями. Являет собой хороший пример удачного совмещения компактного синтаксиса Python с хорошей реализацией большого набора различных алгоритмов из области машинного интеллекта. Предназначен для следующих категорий пользователей:

1. Исследователей – предоставляет единообразную среду для реализации различных алгоритмов, избавляя от потребности в использовании десятков различных библиотек. Позволяет сосредоточится на самом алгоритме, а не особенностях его реализации.
2. Студентов – с использованием PyBrain удобно реализовать домашнее задание, курсовой проект или вычисления в дипломной работе. Гибкость архитектуры позволяет удобно реализовывать разнообразные сложные методы, структуры и топологии.
3. Лекторов – обучение методам Machine Learning было одной из основных целей при создании библиотеки. Авторы будут рады, если результаты их труда помогут в подготовке грамотных студентов и специалистов.
4. Разработчиков – проект Open Source.

Сама библиотека является продуктом с открытым исходным кодом и бесплатна для использования в любом проекте. Рассмотрим основные библиотеки, ориентированные на нейронные сети [23–27].

1. Theano. Программа представляется в символьной парадигме, описывается граф вычислений. Theano позволяет проводить ресурсоемкие вычисления

с высокой точностью за разумное время. Используются вычисления на CPU и GPU. Содержит встроенные механизмы оптимизации кода. Имеет расширения для юнит-тестирования и самопроверки.

2. TensorFlow. Универсальная библиотека машинного обучения с открытым исходным кодом, разработанная компанией Google. Вычисления TensorFlow выражаются в виде потоков данных через граф состояний. Поддерживаются вычисления на GPU Nvidia. Высокая скорость работы. Большой функционал. Постоянное развитие платформы.

3. *Keras*. Представляет собой API высокого уровня, используется как надстройка над TensorFlow или Theano. Модульное представление модели нейронной сети обеспечивает высокую скорость разработки программ. Содержит многочисленные реализации строительных блоков нейронных сетей (слои, передаточные функции, оптимизаторы) и множество инструментов для упрощения работы с изображениями и текстом.

4. *NLTK (Natural Language Toolkit).* Преимущественно используется для анализа текстовых документов: тегирования; токенизации; идентификации имен; построения связей между предложениями и частями текста; семантические рассуждения.

5. *Caffe.* Открытый исходный код. Содержит реализацию большого количества известных нейросетей. Обеспечивает высокую скорость работы программы, особенно при обработке изображений. Высокий уровень взаимодействия центрального и графического процессоров. Включена поддержка MATLAB.

6. *OpenCV.* Библиотека алгоритмов компьютерного зрения. Содержит алгоритмы для: интерпретации изображений, устранение оптических искажений, определение сходства, анализ перемещения объекта, определение формы объекта и слежение за объектом, 3D-реконструкция, сегментация объекта, распознавание жестов и т.д.

7. *Scikit-learn.* Свободная библиотека на Python с низким порогом вхождения. Позволяет проводить классификацию, кластеризацию, уменьшение размерности данных, обучение с учителем и многое другое.

Microsoft CNTK / Microsoft Cognitive Toolkit. Фреймворк с открытым исходным кодом, основанный на нейронных сетях с поддержкой текста, сообщений и ремоделирования голоса. Хорошо оптимизирован, высокая масштабируемость и скорость работы. Включает в себя различные компоненты, такие как: настройки гиперпараметра, GAN, CNN, RNN и т. д.

PyBrain оперирует сетевыми структурами, которые могут быть использованы для построения практически всех поддерживаемых библиотекой сложных алгоритмов.

Дополнительно присутствуют программные инструменты, позволяющие реализовывать сопутствующие задачи: построение и визуализация графиков; поддержка netCDF; запись и чтение XML.

Пример создания нейронной сети с двумя входами, тремя скрытыми слоями и одним выходом:

```
>>> from pybrain.tools.shortcuts import buildNetwork
>>> net = buildNetwork (2, 3, 1)
```

В результате, в объекте net находится созданная нейронная цепь, инициализированная случайными значениями весов. Функция активации задаётся следующим образом:

```
net.activate([2, 1])
```

Количество элементов, передаваемых в сеть должно быть равно количеству входов. Сеть возвращает ответ в виде единственного числа, если текущая цепь имеет один выход, и массив, в случае большего количества выходов.

Контрольные вопросы

1. Сравните нейропакеты для решения поставленной задачи.
2. Перечислите преимущества создания и применения нейросетей в пакете MatLab.
3. Перечислите преимущества создания и применения пакета NeuralWorks Professional II/Plus.
4. Перечислите преимущества применения пакета NeuroShell 2.
5. Перечислите преимущества применения пакета NeuroSolutions.
6. Перечислите преимущества создания и применения нейросетей в пакете Statistica.
7. Перечислите преимущества создания и применения нейросетей в пакете Python.
8. Области применения нейросетевых технологий.

6. НЕЧЕТКИЕ СИСТЕМЫ ОБРАБОТКИ ЗНАНИЙ

6.1. Основы нейросетевых нечетких систем

В 1970 году Беллман и Заде опубликовали статью "Decision – Making in Fuzzy Environment", которая послужила отправной точкой для большинства работ по нечеткой теории принятия решений.

Интерес к теории нечетких множеств постоянно возрастает. Об этом может свидетельствовать экспоненциальный рост публикаций в этой области за последние тридцать. Начиная с 90-х годов в области нечеткой логики акценты исследований сместились с общетеоретических – к прикладным инженерным [6].

В настоящее время издаются более десяти специализированных журналов по нечетким множествам и системам, среди которых “Fuzzy Sets and Systems” (издательство Elsevier Science), “Journal of Intelligent and Fuzzy Systems” (издательство IOS Press), “Intenational Journal of Uncertainty, Fuzziness and Knowledge-Based Systems” (издательство World333 Scientific), IEEE Transactions on Fuzzy Systems (издательство IEEE). Огромное количество журналов публикуют статьи по нечетким множествам, среди которых отметим следующие русскоязычные «Кибернетика и системный анализ», «Известия РАН. Теория и системы управления», «Автоматика и телемеханика», «Автоматика и вычислительная техника».

Принятие решения – это выбор альтернативы, которая одновременно удовлетворяет и нечетким целям, и нечетким ограничениям. В этом смысле, цели и ограничения являются симметричными относительно решения, что стирает различия между ними и позволяет представить решение как слияние нечетких целей и ограничений.

При принятии решений по схеме Беллмана-Заде не делается никакого различия между целью и ограничениями [4, 6].

В настоящее время большую актуальность приобретает использование систем нечеткой логики для решения объемных, трудно формализуемых задач в различных предметных областях, причем области применения нечеткой логики постоянно расширяются. Нечеткие системы находят широкое применение в медицине и экономике, в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других. В результате, возникает необходимость в разработке адекватных моделей, эффективных алгоритмов и реализующих их программных комплексов.

Недостаток создания нечетких систем заключается в том, что набор правил вывода устанавливается человеком-экспертом. Построение окончательной модели требует зачастую объединения знаний специалистов многих областей и многократного тестирования системы. Экспертам, помимо правил, необходимо определиться с типами и параметрами функций принадлежности нечетких множеств, заданных на совокупности входных и выходных параметров, а это требует достаточно длительного и утомительного процесса подбора указанных величин.

Основные трудности при использовании нечетких систем на практике связаны с априорным определением правил и построением функции принад-

лежности для каждого значения лингвистических переменных, описывающих структуру объекта, которые обычно проектировщик выполняет вручную.

Поскольку вид и параметры функций принадлежности выбираются субъективно, они могут быть не вполне адекватны реальной действительности.

Наиболее перспективными моделями, используемыми в интеллектуальных системах поддержки принятия решений в этом плане, являются модели на основе нечеткой логики и нейросетевой технологии. Объединение этих технологий в рамках единой системы позволяет создать следующие типы гибридных моделей [6, 7]:

– нейросетевые нечеткие модели, в которых нейросетевая технология используется как инструмент в нечетких логических системах (для автоматического формирования функций принадлежности, определения нечетких отношений и т. д.);

– нечеткие нейронные сети, в которых с помощью аппарата нечеткой логики осуществляется фаззификация отдельных элементов нейросетевых моделей (нейронов, межнейронных связей, разделения на кластеры и т. д.);

– нечетко-нейросетевые гибридные модели, в которых объединение осуществляется на уровне методов для настройки одного из элементов модели.

Многообещающим представляется третий вид интеграции, при котором исходные модели не модифицируются, а лишь более «тонко» подстраиваются для более четкого выполнения некоторых функций.

Реализация нечеткой системы включает в среднем 50 правил, 6 лингвистических переменных. В результате внедрения нечеткой системы кондиционирования воздуха в помещении сокращается время ее создания на 40 процентов к стандартному решению, обеспечивается поддержка температуры при наличии возмущающих факторов (открытые окна и т.п.), экономия энергии составляет до 24 процентов.

Пример: Система кондиционирования воздуха Mitsubishi.

Промышленная система кондиционирования воздуха, обеспечивающая гибкую реакцию на изменения окружающих условий.

Реализация: 50 правил; 6 лингвистических переменных; Разрешение: 8 бит; Входные значения: температура в комнате, температура стены и мгновенные значения этих сигналов.

Разработка: 4 дня на создание прототипа; 20 дней на тестирование и интеграцию; 80 дней на оптимизацию на реальных тестовых объектах; Реализация в виде чисто программного комплекса на стандартном микроконтроллере.

Результаты применения системы кондиционирования воздуха Mitsubishi: сокращение времени и трудоемкости разработки до 40 процентов к стандартному решению; поддержка температуры при наличии возмущающих факторов (открытые окна и т.п.) существенно улучшена; используется небольшое число датчиков; экономия энергии – 24 процента.

6.2. Система нечеткого вывода Мамдани-Заде

Для реализации систем на базе нечетких правил разработано множество алгоритмов нечеткого вывода. Алгоритмы нечеткого вывода различаются главным образом видом используемых правил, логических операций и разновидностью метода дефаззификации. Разработаны модели нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото [6].

Например, правила нечеткого вывода заданы следующим образом:

П1: если x есть A, то w есть D,

П2: если y есть B, то w есть E,

П3: если z есть C, то w есть F,

где x, y, z – имена входных переменных (четкого формата);

w – имя переменной вывода;

A, B, C, D, E, F – заданные функции принадлежности.

Иллюстрация к алгоритму нечеткого вывода представлена на рис. 6.1.

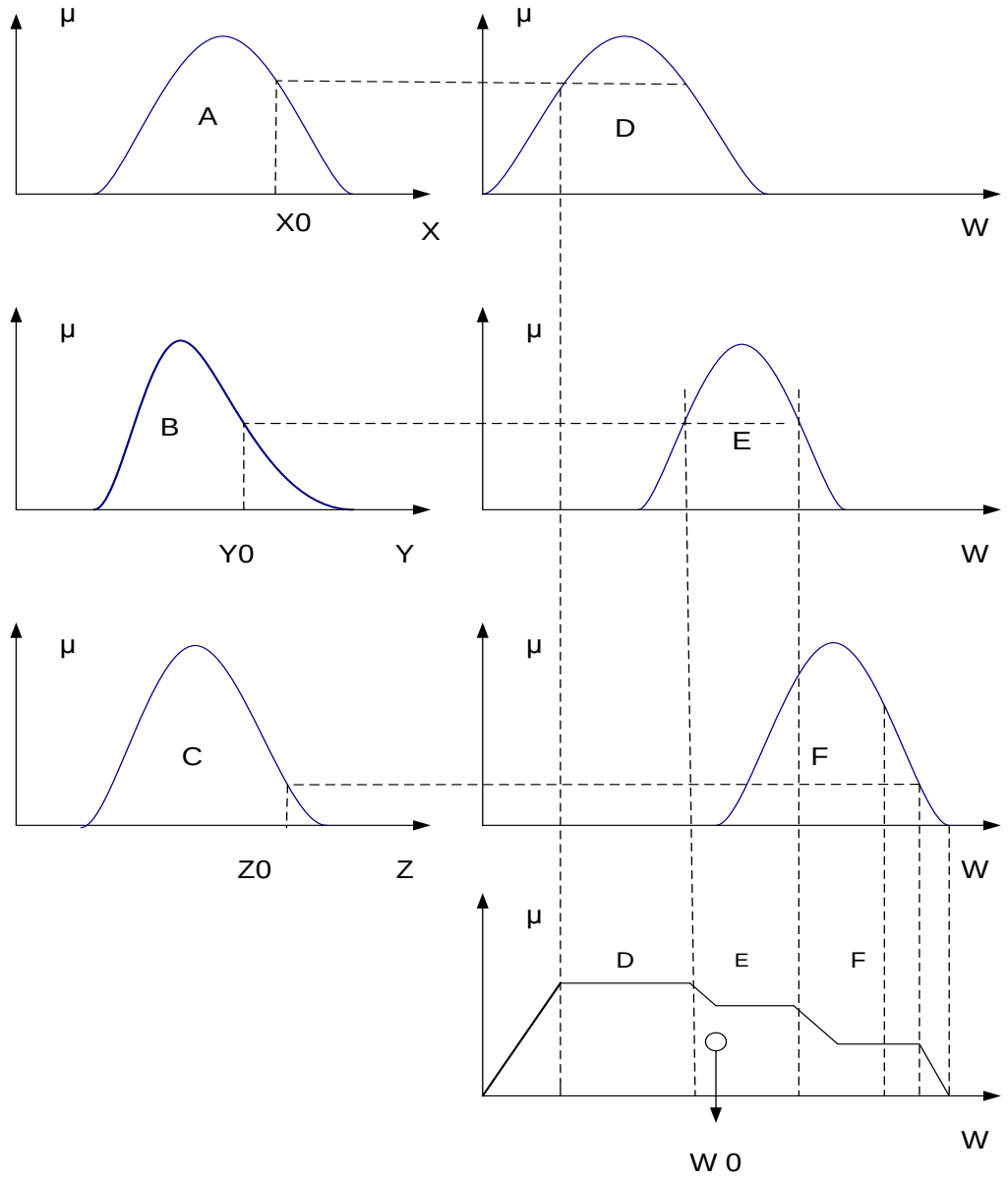


Рис. 6.1. Иллюстрация нечеткого вывода в системе Мамдани-Заде

Пример 2 реализации алгоритма Мамдани с правилами П1 и П2:

П1: если x есть A1 & y есть B1, то z есть C1,

П2: если x есть A2 & y есть B2, то z есть C2

где x, y – имена входных переменных (чет,кого формата);

z – имя переменной вывода;

A1, B1, C1, A2, B2, C2 – заданные функции принадлежности.

Далее следует этап, называемый «введение нечеткости».

Находятся степени истинности для предпосылок каждого правила:

$A_1(x_0), A_2(x_0), B_1(y_0), B_2(y_0),$

где x_0, y_0 – имена входных переменных (четкого формата).

Находятся уровни α-отсечения для предпосылок каждого из правил.

$$\alpha_1 = A_1(x_0) \cap B_1(y_0), \quad (6.1)$$

$$\alpha_2 = A_2(x_0) \cap B_2(y_0), \quad (6.2)$$

где α_1, α_2 – уровни отсечения;

$\cap$ – оператор минимума.

Производится объединение усеченных множеств

$$\mu\sum(z)=(\alpha_1 \cap C_1(z)) \cup (\alpha_2 \cap C_2(z)), \quad (6.3)$$

где C(z) – функция принадлежности для элемента z.

Для нахождения значения z_0 необходимо провести дефаззификацию, например, центроидным методом [6].

Рассмотрим подробнее нечеткий вывод на примере механизма в модели Мамдани-Заде, в котором присутствуют следующие операторы:

– оператор логического или арифметического произведения для определения результирующего уровня активации, в котором учитываются все компоненты вектора *x* условия;

– оператор логического или арифметического произведения для определения значения функции принадлежности для всей импликации A→B*;*

– оператор логической суммы как агрегатор равнозначных результатов импликации многих правил;

– оператор дефаззификации, трансформирующий нечеткий результат μ*(y)*
в четкое значение выходной переменной *y*.

Этапы логического вывода:

1. Проводится процедура фаззификации: определяются степени истинности, т. е. значения функций принадлежности для левых частей каждого правила (предпосылок).

2. Определяются уровни α-отсечения для левой части каждого из правил:

3. $$\alpha\mathrm{i} = \frac{min}{i}(\mathrm{Aik(xk)}). \quad (6.4)$$

4. Далее находятся *«усеченные»* функции принадлежности:

5. $$\text{Усеченные } Bi(y) = \frac{min}{i}\big(\alpha\mathrm{i}, \mathrm{Bi(y)}\big). \quad (6.5)$$

6. Композиция, или объединение полученных усеченных функций, для чего используется максимальная композиция нечетких множеств:

7. $$\mu\ \mathrm{F(y)}=\frac{max}{i}\ (\text{усеченные } Bi(y)). \quad (6.6)$$

8. где μ F(y) – функция принадлежности итогового нечеткого множества.

9. Далее осуществляется процедура приведения к четкости, например, методом среднего центра, или центроидным методом.

6.3. Принципы построения нейросетевых нечетких систем

Нейросетевой нечеткой системой называется такая система, в которой отдельные элементы нечеткости (функции принадлежности, логические операторы, отношения) и алгоритмы вывода реализуются с помощью нейронной сети [6,7].

Основное преимущество нечетких систем в отличие от нейронных сетей следующее:

1. Знания в этих системах представляются в форме легко понимаемых человеком гибких логических конструкций, таких, как

«IF … – THEN …».

2. Б. Коско (1993) доказал теорему: *«любая математическая функция может быть аппроксимирована системой, основанной на нечеткой логике, следовательно, такие системы являются универсальными».*

С помощью НС имеется возможность выявления закономерностей в данных, их обобщение, т.е. извлечение знаний из данных, а основной недостаток– невозможность непосредственно (в явном виде, а не в виде вектора весовых коэффициентов межнейронных связей) представить функциональную зависимость между входом и выходом исследуемого объекта.

Наглядно нейро-сетевую реализацию демонстрирует следующий пример. Рассмотрим возможность построения функций принадлежности для трех значений некоторой лингвистической переменной: «мало (S-Small)», «средне (M-Medium)», «много (L-Large)». Нейросетевая реализация функций принадлежности приведена на рис. 6.2.

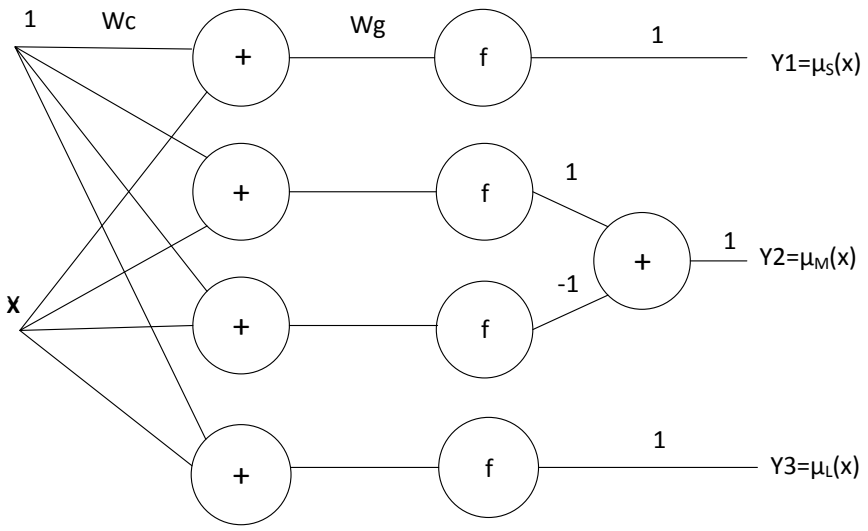


Рис. 6.2. Нейросетевая реализация функций принадлежности

На рис. 6.2 параметр wc – весовой коэффициент смещения и wg – вес суммарного сигнала на входе нелинейного преобразователя. Путем подстройки весовых коэффициентов формируется функция принадлежности. Выходы y1, y2, y3 определяют величины соответствующих функций принадлежности – μS(x), μM(x), μL(x). Узлы со знаком «+» суммируют сигналы входов нейронов, а узлы с символом f реализуют сигмовидные функции.

Реализация нечетких И (AND), ИЛИ (OR) и других операторов в нейросетевом логическом базисе дает основу для построения нейросетевых нечетких моделей, т.е. можно задать функцию активации, реализующую min-оператор для нечеткой операции AND, max-оператор для нечеткой операции OR. «Мягкий» (softmin) оператор может быть использован для замены оригинального нечеткого оператора И:

$$(a \cap b) = softmin(a,b) = \frac{ae^{-ka} + be^{-kb}}{e^{-a} + e^{-b}}. \qquad (6.7)$$

Две главные проблемы, возникающие при организации вывода на основе нечеткой логики, которые более эффективно можно реализовать на основе нейросетевого подхода (рис. 6.3):

Первая проблема связана с определением функций принадлежности, использующихся в условной части правил.

Вторая проблема – с выбором одного правила, определяющего решение, из совокупности правил.

Правила нечеткого вывода имеют формат

Rs: IF X=(x1, x2,…,xn) is As THEN ys=HCs(x1,x2,…,xn).

Условная часть правила IF использует r правил вывода (s=1, 2,…,r). As-нечеткое множество условной части каждого правила. HCs-структура нейросетевой модели (многослойный персептрон) со входами x1,x2,…,xn и выходом ys.

Для каждого правила своя НС. Для определения функций принадлежности условной части правила используется HCs(μ) –многослойный персептрон. Нейронная сеть HCs(μ) обучается на входных данных обучающей выборки.

Выходы обученной НС рассматриваются как величины функций принадлежности нечетких множеств в условной части IF правил, т.к. величина выходного сигнала определяет принадлежность данных к каждому правилу.

Алгоритм процедуры нечеткого вывода на основе нейросетевой модели:

Формирование обучающей No и тестовой NT выборки; N= No + NT – общее число примеров из базы данных.

Кластеризация обучающей выборки. Обучающая выборка делится на r классов Rs (по числу правил), где s=1,2,…,r. Каждая обучающая подвыборка для класса Rs определяется парой ($x^s i, y^s i$), где i=1,2,…,Ns, а Ns– число примеров в обучающей выборке для класса Rs.

Разделение n-мерного входного пространства на число *r* означает, что число правил вывода равно r.

Обучение HCs(μ). Для каждого входного вектора X определим вектор функций принадлежности к правилу Mi такой, что ms i=1 и mk i=0 для k≠s . Нейронная сеть HCs(μ) с n входами и r выходами обучается на парах (Xi, Mi).

После обучения и тестирования такая сеть будет способна определить степень принадлежности для каждого входного вектора, принадлежащего к классу Rs.

Таким образом, функция принадлежности к части IF правила определяется как выходная величина обученной НС.

На четвертом шаге алгоритма процедуры нечеткого вывода на основе нейросетевой модели – это обучение HCs. Обучающая выборка с входами и выходной величиной yi; i=1,2,…,Ns подается на вход и выход HCs, которая является нейросетевой моделью части THEN в Rs.

Пятый шаг алгоритма процедуры нечеткого вывода – это принятие решения на основе нейросетевой модели. Для заданного входного вектора производится вычисление выходной величины по аналогии с формулой дефаззификации:

$$Yt^{*}=\frac{\sum_{s=1}^{r} mAs(Xi)ms(Xi)}{\sum_{s=1}^{r} mAs(Xi)}. \qquad (6.8)$$

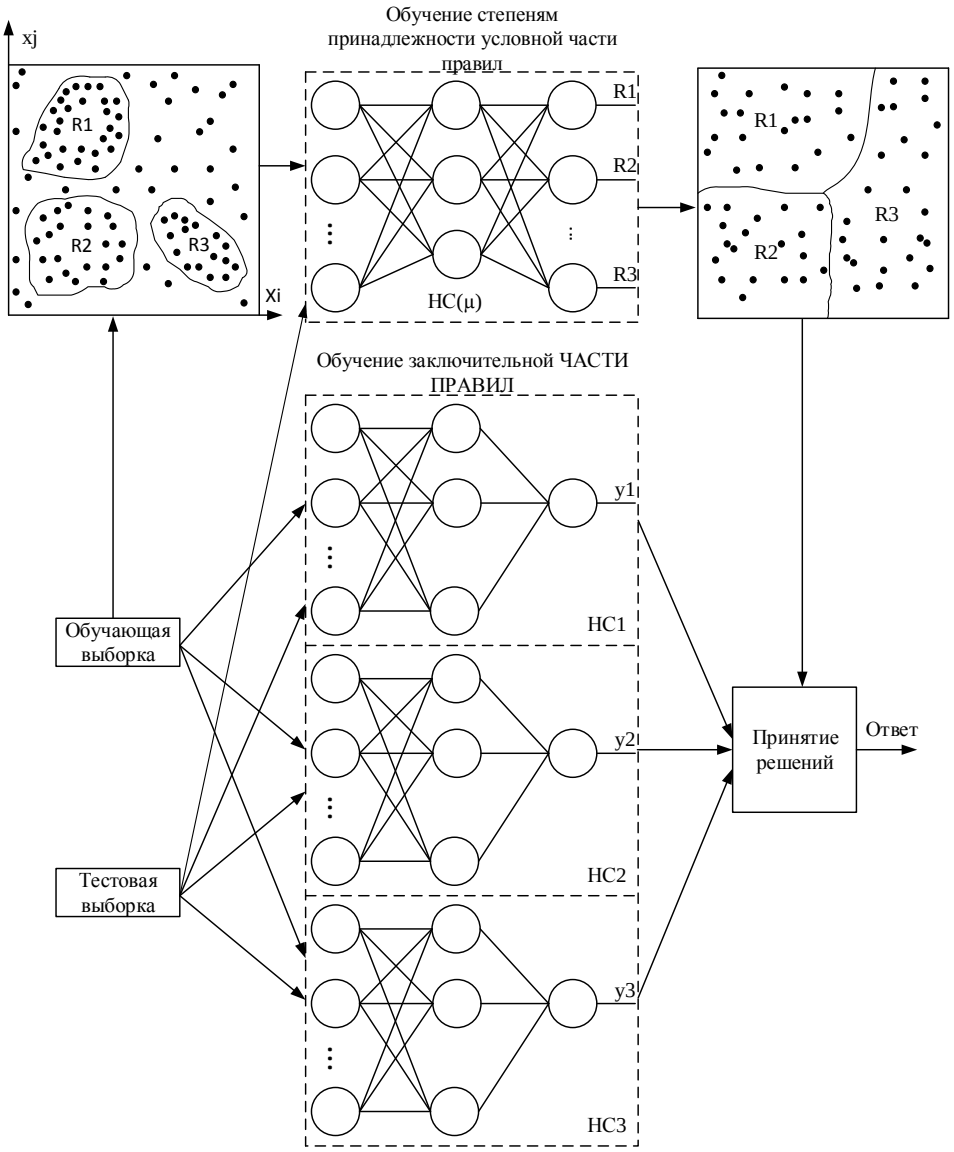


Рис. 6.3. Иллюстрация нейросетевого подхода

Более подробно нечеткие системы и примеры их создания описаны в [6].

Контрольные вопросы

1. Описать алгоритмы нечеткого вывода на основе нейросетевой модели.
2. Реализация нечетких И (AND), ИЛИ (OR) и других операторов в нейросетевом логическом базисе.
3. Нейросетевая реализация функций принадлежности.
4. Основное преимущество нечетких систем в отличие от нейронных сетей.
5. Понятие нейросетевой нечеткой системой.
6. Основные этапы алгоритма нечеткого вывода в системе Мамдани-Заде.
7. Эффективность применения нечетких систем.
8. Области применения нечетких систем.
9. Принцип работы агрегатора.
10. Принцип работы фаззификатора.
11. Принцип работы дефаззификатора.
12. Принципы построения нейросетевых нечетких систем.

7. ОСНОВЫ ТЕОРИИ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

7.1. Основные понятия и определения генетических алгоритмов

Генетические алгоритмы представляют собой алгоритмы поиска, построенные на принципах, сходных с принципами естественного отбора и генетики. Они имеют целью нахождение лучшего, а не оптимального решения задачи. Это связанно с тем, что для сложной системы часто требуется найти удовлетворительное решение, а проблема достижения оптимума отходит на второй план. При этом другие методы, ориентированные на поиск именно оптимального решения, вследствие чрезвычайной сложности задачи становятся вообще неприменимыми. В этом кроется причина появления, развития и роста популярности генетических алгоритмов[10].

Генетические алгоритмы – это новая область исследований, которая появилась в результате работ Д. Холланда и его коллег. Генетические алгоритмы, описанные Д. Холландом, заимствуют в своей терминологии многое из естественной генетики. Далее будут приведены технические толкования терминов из биологии и генетики, которые используются в теории и практике генетических алгоритмов. Впервые генетические алгоритмы были применены к таким научным проблемам, как распознавание образов и оптимизация. Генетический алгоритм представляет собой адаптивный поисковый метод, основанный на селекции лучших элементов в популяции, подобно эволюционной теории Ч. Дарвина.

Основой для возникновения генетических алгоритмов послужили модель биологической эволюции и методы случайного поиска. Л. Растригин отмечал, что методы случайный поиск возник как реализация простейшей модели эволюции, когда случайные мутации моделировались случайными шагами оптимального решения, а отбор – «устранением» неудачных вариантов. Например, генетические алгоритмы используются для выбора оптимальной архитектуры нейронной сети, ее обучения, подбора параметров нейросетевой системы обработки знаний и т.п.

Эволюционный поиск с точки зрения преобразования информации – это последовательное преобразование одного конечного нечеткого множества промежуточных решений в другое. Само преобразование можно назвать алгоритмом поиска, или генетическим алгоритмом. Генетические алгоритмы – это не просто случайный поиск. Они эффективно используют информацию, накопленную в процессе эволюции[11].

Цель генетических алгоритмов состоит в том, чтобы:

– абстрактно и формально объяснять адаптацию процессов в естественной системе и интеллектуальной исследовательской системе;

---

[10] *«Аугментация (augmentation, «раздутие») данных для обучения нейронной сети на примере печатных символов», август 2015. https://habr.com/company/smartengines/blog/264677/*

[11] http://www.intuit.ru/studies/courses/83/83/lecture/20490?page=3

– моделировать эволюционные процессы для эффективного решения оптимизационных задач науки и техники.

– В настоящее время используется новая парадигма решений оптимизационных задач на основе генетических алгоритмов и их различных модификаций. Генетические алгоритмы осуществляют поиск баланса между эффективностью и качеством решений за счет «выживания сильнейших альтернативных решений» в неопределенных и нечетких условиях.

– Генетические алгоритмы отличаются от других оптимизационных и поисковых процедур следующим:

– работают в основном не с параметрами задачи, а с закодированным множеством параметров;

– осуществляют поиск не путем улучшения одного решения, а путем использования сразу нескольких альтернатив на заданном множестве решений;

– используют целевую функцию, а не её различные приращения для оценки качества принятия решений;

– применяют не детерминированные, а вероятностные правила анализа оптимизационных задач.

Для работы генетических алгоритмов выбирают множество натуральных параметров оптимизационной проблемы и кодируют их в последовательность конечной длины в некотором алфавите. Они работают до тех пор, пока не будет выполнено заданное число генераций (итераций алгоритма) или на некоторой генерации будет получено решение определенного качества, или, когда найден локальный оптимум, т.е. возникла преждевременная сходимость и алгоритм не может найти выход из этого состояния. В отличие от других методов оптимизации эти алгоритмы, как правило, анализируют различные области пространства решений одновременно и поэтому они более приспособлены к нахождению новых областей с лучшими значениями целевой функции.

Рассмотрим понятия и определения из теории генетических алгоритмов. Все генетические алгоритмы работают на основании начальной информации, в качестве которой выступает популяция альтернативных решений *P*.

Популяция $P_t=\{P_1, P_{2,\dots}, P_{i,\dots}, P_{Np}\}$ есть множество элементов P_i , t=0,1,2… – номер генерации генетического алгоритма, Np – размер популяции. Каждый элемент P_i этой популяции, как правило, представляет собой одну или несколько *хромосом или особей*, или индивидуальностей (альтернативных упорядоченных или неупорядоченных решений). Хромосомы состоят из генов $P_t=\{g_1, g_2, \dots, g_v\}$ (элементы, части закодированного решения), и позиции генов в хромосоме называются лоции или локус для одной позиции, т. е. *ген* – подэлемент (элемент в хромосоме), *локус* – позиция в хромосоме, *аллель* – функциональное значение гена.

Гены могут иметь числовые или функциональные значения. Обычно эти числовые значения берутся из некоторого алфавита. Генетический материал элементов обычно кодируется на основе двоичного алфавита {0,1}, хотя можно использовать буквенные, а также десятичные и другие алфавиты. Примером за-

кодированной хромосомы длины девять на основе двоичного алфавита может служить хромосома P_t=(001001101).

Элементы в генетических алгоритмах часто называют родителями. *Родители* выбираются из популяции на основе заданных правил, а затем смешиваются (скрещиваются) для производства «*детей*» (*потомков*). Дети и родители в результате генерации, т. е. одного цикла (подцикла) эволюции, создают новую популяцию. Генерация, то есть процесс реализации одной итерации алгоритма, называется поколением.

По аналогии с амии, происходящими в живой природе, в технике считают, что *эволюция популяции* – это чередование поколений, в которых хромосомы изменяют свои значения так, чтобы каждое новое поколение наилучшим способом приспосабливалось к внешней среде. Тогда общая генетическая упаковка называется *генотипом*, а организм формируется посредством связи генетической упаковки с окружающей средой и называется *фенотипом*.

Каждый элемент в популяции имеет определенный уровень качества, который характеризуется значением целевой функции (в литературе иногда называется функция полезности, приспособляемости или пригодности (fitness)). Эта функция используется в генетических алгоритмах для сравнения альтернативных решений между собой и выбора лучших.

Следовательно, основная задача генетических алгоритмов состоит в оптимизации целевой функции. Другими словами, генетические алгоритмы анализируют популяцию хромосом, представляющих комбинацию элементов из некоторого множества, оптимизируют целевую функцию, оценивая каждую хромосому. Генетические алгоритмы анализируют и преобразовывают популяции хромосом на основе механизма натуральной эволюции. Каждая популяция обладает наследственной изменчивостью. Это означает наличие возможностей случайных отклонений от наиболее вероятного среднего значения целевой функции. Отклонения описываются нормальным законом распределения случайных величин. При этом наследственные признаки закрепляются, если они имеют приспособительный характер, т. е. обеспечивают популяции лучшие условия существования и размножения.

Так же как процесс эволюции начинается с начальной популяции, так и алгоритм начинает свою работу с создания начального множества конкурирующих между собой решений оптимизационной задачи. Затем эти «родительские» решения создают «потомков» путем случайных и направленных изменений. После этого оценивается эффективность этих решений, и они подвергаются селекции. Аналогично естественным системам здесь действует принцип «выживания сильнейших», наименее приспособленные решения «погибают», а затем процесс повторяется вновь и вновь.

Традиционные оптимизационные алгоритмы для нахождения лучшего решения используют большое количество допущений, что расширяет класс задач, которые можно решать с помощью генетических алгоритмов. Согласно существующим исследованиям можно сказать, что генетические алгоритмы позволяют решать те проблемы, решение которых традиционными алгоритмами затруднительно.

Генетический алгоритм дает преимущества при решении практических задач. Одно из них – это адаптация к изменяющейся окружающей среде. В реальной жизни проблема, которая была поставлена для решения изначально, может претерпеть огромные изменения в процессе своего решения. При использовании традиционных методов все вычисления приходится начинать заново, что приводит к большим затратам машинного времени. При эволюционном подходе популяцию можно анализировать, дополнять и видоизменять применительно к изменяющимся условиям, для этого не требуется полный перебор. Другое преимущество генетических алгоритмов для решения задач состоит в способности быстрой генерации достаточно хороших решений.

При решении практических задач с использованием генетических алгоритмов обычно выполняют четыре предварительных этапа:

- выбор способа представления решения;
- разработка операторов случайных изменений;
- определение способов «выживания» решений;
- создание начальной популяции альтернативных решений.

Рассмотрим некоторые особенности выполнения этих этапов.

На первом этапе для представления решения в формальном виде требуется такая структура, которая позволит кодировать любое возможное решение и производить его оценку. Математически доказано, что не существует идеальной структуры представления, так что для создания хорошей структуры требуется анализ, перебор и эвристические подходы. Возможный вариант представления должен позволять проведение различных перестановок в альтернативных решениях. Для оценки решений необходимо определить способ вычисления целевой функции.

На втором этапе достаточно сложным является выбор случайного оператора (или операторов) для генерации потомков. Существует огромное число таких операторов. Существуют два основных типа размножения: половое и бесполое. При половом размножении два родителя обмениваются генетическим материалом, который используется при создании потомка. Бесполое размножение – это фактически клонирование, при котором происходят различные мутации при передаче информации от родителя к потомку. Модели этих типов размножения играют важную роль в генетических алгоритмах. В общем случае можно применить модели размножения, которые не существуют в природе. Например, использовать материал от трех или более родителей, проводить голосование при выборе родителей и т. п. Фактически нет пределов в использовании различных моделей, и поэтому при решении технических задач нет смысла слепо копировать законы природы и ограничиваться только ими.

Успех генетических алгоритмов во многом зависит от того, как взаимодействуют между собой схема представления, методы случайных изменений и способ определения целевой функции. Поэтому для определенного класса задач целесообразно использовать направленные методы.

В качестве примера рассмотрим два способа представления перестановок при решении оптимизационных задач. В первом случае будем использовать одного родителя (альтернативное решение) и получать одного потомка. Во втором

случае используем двух родителей, случайно выберем точку перестановки и для образования потомка возьмем первый сегмент у первого родителя, а второй сегмент – у второго. Первый метод похож на бесполое размножение, а второй – на половое размножение. Стоить отметить, что если первый метод всегда генерирует реальное решение, то второй может генерировать недопустимые решения. При этом требуется «восстанавливать» допустимые решения перед их оценкой.

На третьем из рассматриваемых этапов задаются правила выживания решений для создания потомства. Существует множество способов проведения селекции альтернативных решений. Простейшее правило – это «выживание сильнейших», т. е. остаются только лучшие решения с точки зрения заданной целевой функции, а все остальные устраняются. Такое правило часто оказывается малоэффективным при решении сложных технических проблем. Иногда лучшие могут происходить от худших, а не только от самых лучших. Тем не менее, логично использовать принцип: чем лучше решение, тем больше вероятность его выживания.

Отметим, что принцип (от латинского «начало») – это:

- основное исходное положение какой-либо теории;
- внутренняя убежденность в чем-либо;
- основная особенность работы механизма, устройства и т. п.

На последнем предварительном этапе создается начальная популяция. При неполноте исходных данных о проблеме решения могут случайным образом выбираться из всего множества. Это реализуется генерацией случайных внутри хромосомных перестановок, каждая из которых представляет собой определенное решение. При создании начальной популяции рекомендуется использовать знания о решаемой задаче. Например, эти знания могут быть получены из опыта разработчика, существующих стандартов и библиотек алгоритмов решения задач данного класса.

Эффективность генетического алгоритма – степень реализации запланированных действий алгоритма и достижение требуемых значений целевой функции. Эффективность во многом определяется структурой и составом её популяции. При создании начального множества решений происходит формирование популяции на основе четырех основных принципов:

- «одеяло» – генерируется полная популяция, включающая все возможные решения в некоторой заданной области;
- «дробовик» – подразумевает случайный выбор допустимых альтернатив из всей области решения данной задачи;
- «фокусировка» – реализует случайный выбор допустимых альтернатив из заданной области решения данной задачи;
- «комбинирование» – состоит в различных совместных реализациях первых трех принципов.

Отметим, что популяция обязательно является конечным множеством.

7.2. Генетические операторы

В каждой генерации генетического алгоритма хромосомы являются результатом применения некоторых генетических операторов.

Оператор – это языковая конструкция, представляющая один шаг из последовательности действий или набора описаний алгоритма.

Генетический алгоритм состоит из набора генетических операторов.

Генетический оператор по аналогии с оператором алгоритма – средство отображения одного множества на другое. Другими словами, это конструкция, представляющая один шаг из последовательности действий генетического алгоритма[12].

Рассмотрим основные операторы генетических алгоритмов.

Оператор репродукции (селекция) – это процесс, посредством которого хромосомы (альтернативные решения), имеющие более высокое значение целевой функции (с «лучшими» признаками), получает большую возможность для воспроизводства (репродукции) потомков, чем «худшие» хромосомы. Элементы, выбранные для репродукции, обмениваются генетическим материалом, создавая аналогичных или различных потомков.

Существует большое число видов операторов репродукции. К ним относятся следующие:

– *Селекция на основе рулетки* – это простой и широко используемый в простом генетическом алгоритме метод. При его реализации каждому элементу в популяции соответствует зона на колесе рулетки, пропорционально соразмерная с величиной целевой функции. Тогда при повороте колеса рулетки каждый элемент имеет некоторую вероятность выбора для селекции. Причем элемент с большим значением целевой функции имеет большую вероятность для выбора.

– *Селекция на основе заданной шкалы.* Здесь популяция предварительно сортируется от «лучшей» к «худшей» на основе заданного критерия. Каждому элементу назначается определенное число и тогда селекция выполняется согласно этому числу.

– *Элитная селекция.* В этом случае выбираются лучшие (элитные) элементы на основе сравнения значений целевой функции. Далее они вступают в различные преобразования, после которых снова выбираются элитные элементы. Процесс продолжается аналогично до тех пор, пока продолжают появляться элитные элементы.

– *Турнирная селекция.* При этом некоторое число элементов (согласно размеру «турнира») выбираются случайно или направленно из популяции, и лучшие элементы в этой группе на основе заданного турнира определяются для дальнейшего эволюционного поиска.

– Оператор репродукции считается эффективным, если он создает возможность перехода из одной подобласти альтернативных решений области поиска в другую. Это повышает вероятность нахождения глобального оптимума

---

12 http://mathmod.asu.edu.ru/images/File/ebooks/GAfinal.pdf

целевой функции. Выделяют два основных типа реализации оператора репродукции:

- случайный выбор хромосом;
- выбор хромосом на основе значений целевой функции.

Кроме описанных, существует большое число других методов селекции, которые можно условно классифицировать на три группы. К первой группе отнесем вероятностные методы. Ко второй – детерминированные методы. К третьей – различные комбинации методов из первой и второй групп. Построение новых операторов репродукции непрерывно продолжается.

Основной трудностью решения инженерных оптимизационных задач с большим количеством локальных оптимумов является *предварительная сходимость алгоритмов*. Другими словами, попадание решения в один, далеко не самый лучший, локальный оптимум при наличии их большого количества. Различные методы селекции и их модификации как раз и позволяют в некоторых случаях решать проблему предварительной сходимости алгоритмов. Следует отметить, что исследователи генетических алгоритмов все более склоняются к мысли применять комбинированные методы селекции с использованием предварительных знаний о решаемых задачах и предварительных результатах.

Опишем теперь *операторы кроссинговера (скрещивания). Оператор кроссинговера* – это языковая конструкция, позволяющая на основе преобразования (скрещивания) хромосом родителей (или их частей) создавать хромосомы потомков. Существует огромное число операторов кроссинговера, так как их структура в основном и определяет эффективность генетических алгоритмов. Кратко рассмотрим основные операторы кроссинговера, известные в литературе, и их модификации.

*Простой (одноточечный) оператор кроссинговера***.** Перед началом работы одноточечного оператора кроссинговера определяется так называемая точка оператора кроссинговера, или разрезающая точка оператора кроссинговера, которая обычно определяется случайно. Эта точка определяет место в двух особях, где они должны быть «разрезаны». Например, пусть популяция P состоит из хромосом P_1 и P_2, которые выступают в качестве родителей, $P=\{P_1, P_2\}$. Пусть первый второй родители имеют вид P_1: 11111, P_2:00000. Выберем точку оператора кроссинговера между вторым и третьим генами в P_1, P_2. Тогда, меняя элементы после точки оператора кроссинговера между двумя родителями, можно создать два новых потомка.

В нашем примере получим

P_1	:	1	1	1	1	1
P_2	:	0	0	0	0	0
P'_1	:	1	1	0	0	0
P'_2	:	0	0	1	1	1

Итак, одноточечный оператор кроссинговера выполняется в три этапа:

1. Две хромосомы $A = a_1, a_2, ..., a_L$ и $B = a'_1, a'_2, ..., a'_L$ выбираются случайно из текущей популяции.

2. Число k выбирается из$\{1,2,\ldots,n-1\}$ также случайно. Здесь L – длина хромосомы, k – точка оператора кроссинговера (номер, значение или код гена, после которого выполняется разрез хромосомы).

3. Две новые хромосомы формируются из A и B путем перестановок элементов согласно правилу

$$A' = a_1, a_2, \ldots, a_k, a'_{k+1}, \ldots, a'_L,$$

$$B' = a'_1, a'_2, \ldots, a'_k, a_{k+1}, \ldots, a_L.$$

После применения оператора кроссинговера имеем две старые хромосомы и всегда получаем две новые хромосомы. Схематически простой оператор кроссинговера показывает преобразование двух хромосом и частичный обмен информацией между ними, использующий точку разрыва, выбранную случайно.

Двухточечный оператор кроссинговера. В каждой хромосоме определяются две точки оператора кроссинговер, и хромосомы обмениваются участками, расположенными между двумя точками оператора кроссинговер. Например:

P_1	:	1	1	1	0	1	0	0
P_2	:	0	0	0	1	1	1	0
P'_1	:	1	1	1	1	1	0	0
P'_2	:	0	0	0	0	1	1	0

Отметим, что точка оператора кроссинговер в двухточечном операторе кроссинговера также определяется случайно. Существует большое количество модификаций двухточечного оператора кроссинговера. Развитием двухточечного оператора кроссинговера является многоточечный или N-точечный оператор кроссинговера. Многоточечный оператор кроссинговера выполняется аналогично двухточечному, хотя большое число «разрезанных» точек может привести к потере «хороших» родительских свойств.

Пример трехточечного оператора кроссинговера:

P_1	:	1	1	1	0	1	0	0
P_2	:	0	0	0	1	0	1	1
P'_1	:	1	0	0	0	1	1	1
P'_2	:	0	1	1	1	0	0	0

Здесь точки оператора кроссинговера делят хромосому на ряд строительных блоков (в данном случае 4). Потомок P'_1 образуется из нечетных блоков родителя P_1 и четких блоков родителя P_2. Потомок P'_2 образуется соответственно из нечетных блоков родителя P_2 и четких блоков родителя P_2.

Тогда многоточечный оператор кроссинговера выполняется аналогичным образом.

Циклический оператор кроссинговера. Циклический оператор кроссинговера выполняет рекомендации согласно циклам, которые существуют при установлении соответствия между генами первого и второго родителей. Например, пусть популяция **P** состоит из двух хромосом: P = $\{P_1$ и $P_2\}$. Первый и второй родители и их потомок имеют вид:

P_1	;	1	2	3	4	5	6	7	8	9	10
P_2	;	5	3	9	1	4	8	10	2	6	7
P'_2	;	1	3	9	4	5	8	10	2	6	7

При выполнении циклического оператора кроссинговера P'_1 заполняется, начиная с первой позиции, и копирует элемент с первой позиции P_1. Элементу 1 в P_1 соответствует элемент 5 в P_2. Следовательно, сформирован первый путь в цикле (1,5). Элементу 5 в P_1 соответствует элемент 4 в P_2, откуда второй путь в первом цикле (1,5; 5,4). Продолжая далее, получим, что элементу 4 в P_1 соответствует элемент 1 в P_2. Следовательно, сформирован первый цикл (1,5; 5,4; 4,1). Согласно этому циклу элемент 5 переходит в пятую позицию P'_1, а элемент 4 – в четвертую позицию.

Сформируем теперь второй цикл. Элемент 2 в P_1 соответствует элементу 3 в P_2. Продолжая аналогично, получим второй (2,3; 3,9; 9,6; 6,8; 8,2) и третий (7,10; 10,7) циклы.

Следовательно, в P'_1 элемент 3 расположен во втором локусе, т.е. на второй позиции, элемент 9 – в третьем, элемент 6 – в девятом, элемент 8 – в шестом, элемент 2 – в восьмом, элемент 10 – в седьмом и элемент 7 – в десятом локусах. Циклический оператор кроссинговера и его модификации эффективно применяются для решения комбинаторно-логических задач, задач на графах и гиперграфах и других оптимизационных задач.

Универсальный оператор кроссинговера. В настоящее время он популярен для решения различных задач из теории расписаний. Вместо использования разрезающей точки (точек) в универсальный оператор кроссинговера вводят двоичную маску, длина которой равна длине заданных хромосом. Первый потомок получается сложением первого родителя с маской на основе следующих правил: (0+0=0, 0+1=1, 1+1=0). Второй потомок получается аналогичным образом. Для каждого элемента в P_1, P_2 гены меняются, как показано на следующем примере:

P_1	:	0	1	1	0	0	1
P_2	:	0	1	0	1	1	1
		0	1	1	0	1	0 – маска
P'_1	:	0	0	0	0	1	1
P'_2	:	0	0	1	1	0	1

Маска может быть задана или выбирается случайно с заданной вероятностью или на основе генератора случайных чисел. При этом чередование 0 и 1 в маске происходит с вероятностью ≈ 50%. В некоторых случаях используется параметризированный универсальный оператор кроссинговера, где маска может выбираться с вероятностью для 1 или 0 выше, чем 50%. Такой вид маски эффективен, когда хромосомы кодируются в двоичном алфавите.

Код Грея – это двоичный код, последовательные значения которого отличаются только одним двоичным разрядом. Код Грея может использоваться для хромосом, представленных в двоичном виде. Например:

0 – 0000 1 – 0001 2 – 0011 3 – 0010 4 – 0110 5 – 0111 и т.д.

Такое кодирование альтернативных решений позволяет решать вопросы «взбалтывания» популяции; оно эффективно на начальных стадиях генетического алгоритма.

Как следует из биологии, некоторые процессы преобразования популяции происходят толчками. Основой таких процессов являются точковые мутации. В генетическом алгоритме мутация необходима потому, что предотвращает потерю генетического материала. Точковые мутации не изменяют размера и строения хромосом, а изменяют расположение генов в хромосоме.

Оператор мутации – языковая конструкция, позволяющая на основе преобразования родительской хромосомы (или ее части) создавать хромосому потомка.

Оператор мутации обычно состоит из двух этапов:

1. В хромосоме $A=(a_1, a_2, a_3, \ldots, a_{L-2}, a_{L-1}, a_L)$ определяются случайным образом две позиции (например, a_2 и a_{L-1}).
2. Гены, соответствующие выбранным позициям, переставляются, и формируется новая хромосома $A'=(a_1, a_{L-1}, a_3, \ldots, a_{L-2}, a_2, a_L)$.

Основные операторы мутации. Простейшим оператором мутации является одноточечный. При его реализации случайно выбирают ген в родительской хромосоме и, обменивая его на рядом расположенный ген, получают хромосому потомка, например:

P_1 : 0 1 1 | 0 1 1

P_1' : 0 1 0 | 1 1 1

Здесь P_1 – родительская хромосома , P' – хромосома-потомок после применения одноточечного оператора мутации.

При реализации *двухточечного оператора мутации* случайным или направленным образом выбираются две точки разреза. Затем производится перестановка генов между собой, расположенных справа от точек разреза, например:

P_1 : A | B C D | E F

P_1' : A E C D B F

Здесь P_1 – родительская хромосома , P' – хромосома-потомок после применения двухточечного оператора мутации.

Развитием двухточечного оператора мутации является многоточечный (или n-точечный) оператор мутации. В этом случае происходит последовательный обмен генов, расположенных правее точек разреза друг с другом в порядке их расположения. Ген, расположенный правее последней точки разреза, переходит на место первого, например:

P_1 : A | B C D | E F | G H

P_1' : A G C D B F E H

Строительные блоки – это тесно связанные между собой гены (части альтернативных решений), которые нежелательно изменять при выполнении генетических операторов. Из строительных блоков (как из кирпичиков при построении дома) можно создавать альтернативные оптимальные или квазиоптимальные решения.

В частном случае, когда строительные блоки, расположенные между точками разреза, одинаковы, многоточечный оператор мутации выполняется следующим образом. При четном числе точек разреза меняются местами гены, расположенные справа и слева от выбранных точек, например:

P_1 : A B | C D | E F | G H | I J

P_1' : A C B E D G F I H J

При нечетном числе точек потомок получается после обмена участками хромосом, расположенных между четными точками разреза, например:

P_1 : A B C | D E F | G H I | J

P_1' : A B C G H I D E F J

Часто используют операторы мутации, использующие знания о решаемой задаче. Реализация таких операторов заключается в перестановке местами любых выбранных генов в хромосоме, причем точка или точки мутации определяются не случайно, а направленно.

В позиционном операторе мутации две точки разреза выбираются случайно, а затем ген, соответствующий второй точке мутации, размещается в позицию перед геном, соответствующим первой точке, например:

P_1 : A | B C D | E F

P_1' : A E B C D F

Введем понятие оператора инверсии. *Оператор инверсии* – это языковая конструкция, позволяющая на основе инвертирования родительской хромосомы (или ее части) создавать хромосому потомка. При его реализации случайным образом определяется одна или несколько точек разреза (инверсии), внутри которых элементы инвертируются.

Генетический оператор инверсии состоит из следующих шагов.

1. Хромосома $B=(b_1, b_2, \ldots, b_L)$ выбираются случайным образом из текущей популяции.
2. Два числа y_1' и y_2' выбираются случайным образом из множества $\{0,1,2,\ldots,L+1\}$, причем считается, что $y_1=\min\{y_1', y_2'\}$ и $y_2=\max\{y_1', y_2'\}$.
3. Новая хромосома формируется из B путем инверсии сегмента, который лежит справа от позиции y_1 и слева от позиции y_2 в хромосоме B. Тогда после применения оператора инверсии получаем:
   $B = (b_1, \ldots, b_{y1}, b_{y2-1}, b_{y2-2}, \ldots, b_{y1+1}, b_{y2}, \ldots, b_L)$.
   Для одноточечного оператора инверсии запишем
   P_2 : A | B C D E F G H
   P_2' : A | H G F E D C B

Здесь P_2 – родительская хромосома, P'_2 – хромосома-потомок после применения оператора инверсии.

Например, для двухточечного оператора инверсии получим

P_1 : A B C | D E F | G H

P'_1 : A B C | F E D | G H

Здесь P_1 – родительская хромосома, P'_1 – хромосома-потомок после применения двухточечного оператора инверсии.

Часто применяется специальный оператор инверсии. В нем точки инверсии определяются с заданной вероятностью для каждой новой создаваемой хромосомы в популяции.

Оператор транслокации – это языковая конструкция, позволяющая на основе скрещивания и инвертирования из пары родительских хромосом (или их частей) создавать две хромосомы потомков. Другими словами, он представляет собой комбинацию операторов кроссинговера и инверсии. В процессе его реализации случайным образом производится один разрез в каждой хромосоме. При формировании потомка P'_1 берется левая часть до разрыва из родителя P_1 и инверсия правой части до разрыва из P_2. При создании P'_2 берется левая часть P_2 и инверсия правой части P_1, например:

P_1 : A B | C D E F

P_2 : G K | H I J Q

P'_1 : A B Q J I H

P'_2 : G K F E D C

Существует большое число других видов оператора транслокации. Отметим, что до последнего времени оператор транслокации не применялся в генетических алгоритмах, а также при разработке интеллектуальной искусственной системы и решении оптимизационных задач.

Оператор транспозиции – языковая конструкция, позволяющая на основе преобразования и инвертирования выделяемой части родительской хромосомы создавать хромосому потомка. Например:

P_1 : A | B C | D E F | G H

P'_1 : A F E D B C G H

Здесь три точки разреза. Точки разреза выбираются случайным или направленным образом. В родительской хромосоме P_1 строительный блок *DEF* инвертируется и вставляется в точку разреза между генами *A* и *B*. В результате получаем хромосому-потомок P'_1. Отметим, что существует большое количество модификаций оператора транспозиции.

Оператор сегрегации. Это языковая конструкция, позволяющая на основе выбора строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

Приведем один их примеров его реализации. Отметим, что оператор сегрегации, как правило, реализуется на некотором наборе хромосом. Пусть имеется популяция *P*, состоящая из четырех родительских:

$P = \{P_1, P_2, P_3, P_4\}$ P_1 : (12345678); P_2 : (24316587); P_3 : (31425768); P_4 : (87654321).

В каждой хромосоме выделим строительные блоки. Выделим по два строительных блока: в P_1 – 23 и 67, в P_2 – 1 и 4, в P_3 – 768 и 425 и в P_4 – 54 и 87. Тогда, например, потомок P'_1 можно сформировать, взяв первые строительные блоки из каждой родительской хромосомы популяции. Так, вариант P'_1 будет представлен последовательностью (23176854) и является реальным решением, а вариант P'_2 (67442587) является нереальным (недопустимым). Очевидно, этот можно реализовать различными способами в зависимости от выборки строительных блоков или генов из хромосом.

Оператор удаления. Это языковая конструкция, позволяющая на основе удаления строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

При его реализации направленным или случайным образом определяется точка или точки разреза. Далее производится пробное удаление генов или их строительных блоков с вычислением изменения значения целевой функции. Элементы, расположенные слева от точки оператора удаления или между двумя точками, удаляются из хромосомы. При этом производится преобразование потомка таким образом, чтобы соответствующее альтернативное решение оставалось реальным.

Оператор вставки. Это языковая конструкция, позволяющая на основе вставки строительных блоков в хромосомы родителей создавать хромосомы потомков.

При его реализации направленным или случайным образом создается хромосома (донор), состоящая из строительных блоков, которые желательно разместить в другие хромосомы популяции. После этого направленным или случайным образом определяется хромосома для реализации оператора вставки. В ней находится точка или точки разреза. Затем анализируются другие хромосомы в популяции для определения альтернативных вставок. Далее производится пробная вставка строительных блоков с вычислением изменения значения целевой функции и получений реальных решений. Новые строительные блоки вставляются в хромосому справа от точки оператора вставки или между его двумя точками. Отметим, что оператор удаления и оператор вставки могут изменять размер хромосом. Для сохранения постоянного размера хромосом эти операторы можно применять совместно.

На конечном этапе поиска целесообразно применять выбор близких решений в соответствии с определенным критерием, т. е. искать решение среди лучших P_k^t.

Оператор редукции – языковая конструкция, позволяющая на основе анализа популяции после одной или нескольких поколений генетического алгоритма уменьшить ее размер до заданной величины.

Рассмотрим теперь *оператор рекомбинации***.** Оператор рекомбинации – языковая конструкция, которая определяет, как новая генерация хромосом будет построена из родителей и потомков. Другими словами, оператор рекомбинации – это технология анализа и преобразования популяции при переходе из

одной генерации в другую. Существует много путей выполнения рекомбинации. Один из них состоит из перемещения родителей в потомки после реализации каждого генетического оператора. Другой путь заключается в перемещении некоторой части популяции после каждой генерации.

Часто в генетических алгоритмах задается параметр *W(P)*, который управляет этим процессом. Так, $N_p(1- W(P))$ элементов в популяции *P*, выбранных случайно, могум «выжить» в следующей генерации. Здесь N_p – размер популяции. Величина $W(P) = 0$ означает, что целая предыдущая популяция перемещается в новую популяцию в каждой генерации. При дальнейшей реализации алгоритма лучшие или отобранные элементы из родителей и потомков будут выбираться для формирования новой популяции.

В инженерных задачах используются различные механизмы и модели этого процесса. Приведем несколько из них:

– M1 – вытеснение (crowding factor). Этот механизм определяет способ и порядок замены родительских хромосом из генерации *t* хромосомами потомками после генерации *t+1*. Механизм реализован таким образом, что стремится удалять «похожие» хромосомы из популяции и оставлять отличающиеся.

– M2 – разделение (sharing). Этот механизм вводит зависимость значения целевой функции хромосомы от их распределения в популяции и поисковом пространстве. Это позволяет копиям родительских хромосом или близких к ним не появляться в популяции.

– M3 – введения идентификаторов (tagging). Специальным хромосомам присваиваются метки. Операторы генетических алгоритмов применяются только к помеченным хромосомам.

Отметим, что оператор редукции является частным случаем оператора рекомбинации.

Важным понятием при реализации генетических операторов является вероятность, которая определяет, какой процент общего числа генов в популяции изменяется каждой генерации. Для оптимизационных задач вероятность оператора кроссинговера обычно принимают в пределах (0,6÷0,99); вероятность оператора мутации – 0,6; инверсии – (0,1÷0,5); транслокации – (0,1÷0,5); транспозиции – (0,1÷0,5); сегрегации – (0,6÷0,99); удаления – (0,6÷0,99); вставки – (0,6÷0,99).

Применение элитизма способствует сохранению общего качества популяции на высоком уровне. При этом элитные особи участвуют еще и в процессе отбора родителей для последующего скрещивания.

7.2. Пример работы генетического алгоритма

Пример поиска максимума одномерной функции. Пусть имеется набор натуральных чисел от 1 до 31 и функция, определенная на этом наборе чисел, f(x)=x. Требуется найти максимальное значение функции [4].

В качестве кода используется двоичное представление аргументов функции. Это положение представляет собой фенотип нашего алгоритма. Сам код будет представлять собой двоичную строку из 5 бит. Это генотип алгоритма. Целевой

функцией будет непосредственно сама рассматриваемая функция, аргументом которой является число, чье двоичное представление использует алгоритм.

Принятые характеристики генетического алгоритма:

- размер популяции 4;
- вероятность мутации 0,001,
- процесс мутации заключается в инверсии одного из битов строки, выбираемого случайно по равномерному закону;
- оператор скрещивания и отбора аналогичны описанным ранее;
- элитизм не используется.

Пусть случайным образом создана исходная популяция из четырех особей, представленная табл. 7.1.

Предположим, что оператор отбора выбрал для производства потомков две пары строк (1, 2) и (2, 4). Работа оператора скрещивания проиллюстрирована в табл. 7.2. При этом в каждой паре разбиение на подстроки происходит независимо. Точка разбиения задана звездочкой.

Пропорциональный простейший отбор (рулетка) выбирает n особей после n запусков. Колесо рулетки содержит по одному сектору для каждого гена популяции. Размер сектора пропорционален вероятности участия.

Таблица 7.1

Исходная популяция

Номер строки	**Код генотипа**	**Значение целевой функции**	**Вероятность участия в размножении**
1	01011	11	11/43
2	10010	18	18/43
3	00010	2	2/43
4	01100	12	12/43
		Сумма 43	

Таблица 7.2

Формирование потомков

Номер строки	**Родители**	**Потомки**	**Значение целевой функции для потомков**
1	0*1011	00010	2
2	1*0010	11011	27
3	100*10	10000	16
4	011*00	01110	14

Пусть оператор мутации, несмотря на низкую вероятность, срабатывает для младшего бита потомка в строке 3 и данный код изменяет свое значение с 10000 на 10001.

Таким образом, популяция за счет порожденных потомков расширяется до восьми особей, представленных табл. 7.3.

Таблица 7.3

Расширенная популяция

Номер строки	Код генотипа	Значение целевой функции
Исходная популяция		
1	01011	11
2	10010	18
3	00010	2
4	01100	12
Порожденные потомки		
5	00010	2
6	11011	27
7	10001	17
8	01110	14

Оператор редукции далее сокращает популяцию до исходного числа особей, исключив из нее особи с минимальным значением целевой функции. В результате исключаются строки 1, 3, 4, 5, и популяция первого поколения принимает вид, представленный табл. 7.4.

На этом шаг работы генетического алгоритма закончится. Очевидно, что даже за эту одну итерацию качество популяции значительно возросло.

Если в исходной популяции среднее значение целевой функции было 10, 75, а ее минимальное значение составляло 2, то в популяции первого поколения среднее значение увеличилось до 19, а минимальное значение составило 14. Лучшее же решение увеличилось с 18 до 27 при оптимальном решении 31.

Таблица 7.4

Популяция первого поколения

Номер строки	Код генотипа	Значение целевой функции	Вероятность участия в процессе размножения
1	10010	18	18/ 76
2	11011	27	27/ 76
3	10001	17	17/ 76
4	01110	14	14/ 76
		Сумма 76	

Таким образом, данный пример работы генетического алгоритма наглядно иллюстрирует процесс улучшения как популяции в целом, так и поиск наилучшего решения [4]. После первого шага работы генетического алгоритма максимальное значение увеличилось с 18 до 27 при оптимальном значении 31.

Контрольные вопросы

1. Понятия мягких вычислений и мягкой экспертной системы
2. Понятие гибридной экспертной системы.
3. Структура гибридной экспертной
4. База знаний мягкой экспертной системы.
5. Преимущества генетических алгоритмов.
6. Основные термины генетических алгоритмов.
7. Этапы выполнения генетического алгоритма.
8. Применение стратегии элитизма в генетических алгоритмах.
9. Принцип работы оператора отбора.
10. Принцип работы двухточечного кроссинговера.
11. Принцип работы оператора универсального скрещивания.
12. Принцип работы оператора циклического скрещивания.
13. Принцип работы оператора вставки и удаления.
14. Принцип работы оператора транспозиции.
15. Принцип работы оператора сегрегации.
16. Принцип работы оператора транслокации.
17. Принцип работы оператора мутации.
18. Принцип работы оператора редукции.
19. Результат работы генетического алгоритма.
20. Достоинства и недостатки операторов транспозиции и сегрегации.

8. ОНТОЛОГИИ

8.1. Общие сведения об онтологии

Онтология (в информатике) – это попытка всеобъемлющей и детальной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из иерархической структуры данных, содержащей все классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области. Этот термин в информатике является производным от древнего философского понятия «онтология» [13–16].

Онтологии, упрощенно говоря, представляют собой описания знаний, сделанные достаточно формально, чтобы быть обработаны компьютерами. Такие формальные описания используются в самых различных и порой достаточно неожиданных областях компьютерной науки. Далее мы рассмотрим, какие обстоятельства привели к возникновению термина «онтология», а также опишем некоторые популярные аспекты его использования при написании программ.

Современные онтологии строятся по большей части одинаково, независимо от языка написания. Обычно они состоят из экземпляров, понятий, атрибутов и отношений.

Знания, которые заложены в компьютерных программах, можно разделить на два сорта:

Процедурные знания, т.е. знания о том, что надо сделать в каждой конкретной ситуации. Например, если бухгалтерской программе пришли данные о платежах, то надо сделать соответствующие изменения на счетах получателей платежей, а также другие необходимые действия, налагаемые данной ситуацией.

Кроме процедурных знаний, каждой программе необходимы знания о мире задачи или *декларативные знания*, т.е. о том, что такое платежи, проводки, счета и т.п. вещи. Без этих знаний, очевидно, программа не сможет функционировать, нельзя будет построить алгоритм программной системы.

Таким образом, при создании интеллектуальной системы приходится учитывать такое разделение знаний и придумывать какие-то программные инструменты для оперирования этими знаниями.

Томас Грубер рассматривал вопросы взаимодействия интеллектуальных систем между собой, а также с человеком. Идея Грубера состояла в том, чтобы позволить интеллектуальным системам обмениваться между собой заложенными в них знаниями о мирах задач. Если внутри интеллектуальной системы знания о мире могут быть закодированы как угодно, то для обмена этими знаниями с другой интеллектуальной системой необходимо предоставить описание этих знаний. Это описание должно быть в достаточной степени формальным, чтобы быть понятным другой системе, а также должен быть известен язык этого описания. Кроме того, описание должно быть понятно также и человеку. Для этого Грубер предложил описывать знания двумя способами:

1. *В канонической форме*, которая представляет собой описание знаний на языке логики предикатов (например, в виде фактов языка Prolog).

2. *В форме онтологии*, которая представляет собой множество классов, связанных между собой отношением обобщения (это обратное отношение для отношения наследования).

Таким образом, онтология по Груберу представляет собой описание декларативных знаний, сделанное в виде классов с отношением иерархии между ними. К этому описанию, предназначенному для чтения человеком, присоединено описание в канонической форме, которое предназначено для чтения машинами. Каждая интеллектуальная система может предоставлять несколько таких описаний, соответствующих различным областям хранящихся в ней декларативных знаний и, таким образом, выступает как хранилище библиотеки онтологий. Грубер представлял, что интеллектуальные системы будут выступать как библиотеки онтологий и свободно обмениваться онтологиями между собой. При этом библиотеке онтологий уже не обязательно быть интеллектуальной системой, достаточно просто предоставлять сервис по передаче онтологий по требованию.

Составление описания декларативного знания обычно требует большой работы и определенных навыков. Для обозначения этой работы, а также ее результата, Грубер ввел в обиход специальный термин «концептуализация». Описание он называл «спецификацией». Таким образом, онтология *по Груберу определяется как спецификация концептуализации.*

Экземпляры (instances) (или индивиды – individuals) – это основные, нижнеуровневые компоненты онтологии. Экземпляры могут представлять собой как физические объекты (люди, дома, планеты), так и абстрактные (числа, слова). Однако одной из главных целей онтологии является классификация таких объектов.

Понятия (concepts) (или классы – classes) – это абстрактные группы коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы, либо же сочетания и того, и другого.

Объекты в онтологии могут иметь атрибуты. Каждый атрибут имеет, по крайней мере, имя и значение, и используется для хранения информации, которая специфична для объекта и привязана к нему. Значение атрибута может быть сложным типом данных.

Важная роль атрибутов заключается в том, чтобы определять зависимости (отношения) между объектами онтологии. Обычно отношением является атрибут, значением которого является другой объект.

Онтологии делятся на специализированные и общие. Специализированные (предметно-ориентированные) онтологии – это представление какой-либо области знаний или части реального мира. В такой онтологии содержатся специальные для этой области значения терминов.

Общие онтологии используются для представления понятий, общих для большого числа областей. Такие онтологии содержат базовый набор терминов, глоссарий или тезаурус, используемый для описания терминов предметных областей.

Если использующая специализированные онтологии система развивается, то может потребоваться их объединение. Это серьёзная задача, так как онтологии часто несовместимы друг с другом, хотя могут представлять близкие области. Разница может появляться из-за особенностей местной культуры, идеологии и т. п., или вследствие использования другого языка описания.

Онтологии используются в процессе программирования как форма представления знаний о реальном мире или его части. Основные сферы применения – моделирование бизнес-процессов, семантическая паутина (англ. Semantic Web), искусственный интеллект.

В настоящее время рассматриваются онтологии, используемые в качестве центральных контролируемых словарей, которые интегрированы в каталоги, базы данных, веб-публикации, приложения для управления знаниями и т. д. Большие онтологии являются важными компонентами во многих онлайн-приложениях, включая поиск (например, Yahoo и Lycos), электронную коммерцию как Amazon и eBay), конфигурации (например, Dell и PC-Order) и т. д. Также есть онтологии, которые имеют длительный срок службы, иногда в нескольких проектах (таких как UMLS, SIC-коды и т. д.). Такое разнообразное использование создает много последствий для онтологических сред[13].

Современные онтологии строятся по большей части одинаково, независимо от языка написания. Обычно они состоят из экземпляров, понятий, атрибутов и отношений [13–16].

Экземпляры (англ. instances) или индивиды (англ. individuals) – это объекты, основные нижнеуровневые компоненты онтологии; могут представлять собой как физические объекты (люди, дома, планеты), так и абстрактные (числа, слова). Строго говоря, онтология может обойтись и без конкретных объектов, однако, одной из главных целей онтологии является классификация таких объектов, поэтому они также включаются.

Понятия (англ. concepts) или классы (англ. classes) – абстрактные группы, коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы, либо же сочетания и того, и другого. Пример: понятие «люди», вложенное понятие «человек». Чем является «человек» – вложенным понятием, или экземпляром (индивидом) – зависит от онтологии.

Понятие «индивиды», экземпляр «индивид».

Классы онтологии составляют таксономию – иерархию понятий по отношению вложения.

Объекты в онтологии могут иметь атрибуты. Каждый атрибут имеет по крайней мере имя и значение и используется для хранения информации, которая специфична для объекта и привязана к нему. Например, объект Автомобиль-модели-А имеет такие атрибуты, как:

Название: Автомобиль-модели-А

Число-дверей: 4

Двигатель: {2.0л, 2.6л}

[13] http://www.ksl.stanford.edu/people/dlm/papers/ontologies-come-of-age-abstract.html

Коробка-передач: 6-ступенчатая

Значение атрибута может быть сложным типом данных. В данном примере значение атрибута, который называется Двигатель, является списком значений простых типов данных.

Важная роль атрибутов заключается в том, чтобы определять отношения (зависимости) между объектами онтологии. Обычно отношением является атрибут, значением которого является другой объект.

Предположим, что в онтологии автомобилей присутствует два объекта – автомобиль Автомобиль-модели-А и Автомобиль-модели-Б. Пусть Автомобиль-модели-Б это модель-наследник Автомобиль-модели-А, тогда отношение между Автомобиль-модели-А и Автомобиль-модели-Б определим, как атрибут «isSuccessorOf» со значением «Автомобиль-модели-А» для объекта Автомобиль-модели-Б (следует заметить, что в языках описания онтологий существуют предопределенные отношения наследования).

Специализированные (предметно-ориентированные) онтологии – это представление какой-либо области знаний или части реального мира. В такой онтологии содержатся специальные для этой области значения терминов. К примеру, слово «поле» в сельском хозяйстве означает участок земли, в физике – один из видов материи, в математике – класс алгебраических систем.

Общие онтологии используются для представления понятий, общих для большого числа областей. Такие онтологии содержат базовый набор терминов, глоссарий или тезаурус, используемый для описания терминов предметных областей.

Если использующая специализированные онтологии система развивается, то может потребоваться их объединение. Подзадачей объединения онтологий является задача отображения онтологий. Онтологии даже близких областей могут быть несовместимы друг с другом. Разница может появляться из-за особенностей местной культуры, идеологии или вследствие использования другого языка описания. Объединение онтологий выполняют как вручную, так и в полуавтоматическом режиме. В целом это – трудоёмкий, медленный и дорогостоящий процесс. Использование базисной онтологии – единого глоссария – несколько упрощает эту работу. Есть научные работы по технологиям объединения, но они по большей части теоретические.

Практически, создание онтологий включает следующие этапы:

1. Определение понятий (классов) в онтологии.
2. Организация понятий (классов) в некоторую иерархию (базовый класс –> подкласс).
3. Определение слотов и их допустимых значений.
4. Заполнение значений слотов для экземпляров классов.

Идея Грубера состояла в том, чтобы позволить интеллектуальным системам обмениваться между собой заложенными в них знаниями, прежде всего декларативными. Если внутри интеллектуальной системы знания о мире могут быть закодированы как угодно, то для обмена этими знаниями с другой интеллектуальной системой необходимо предоставить описание этих знаний. Это описание должно быть в достаточной степени формальным, чтобы быть понят-

ным другой системе, а также должен быть известен язык этого описания. Кроме того, описание должно быть понятно также и человеку. Для этого Грубер предложил описывать знания двумя способами:

– в канонической форме, которая представляет собой описание знаний на языке логики предикатов (например, в виде фактов языка Prolog);

– в форме онтологии, которая представляет собой множество классов, связанных между собой отношением обобщения (это обратное отношение для отношения наследования).

В [13–16] выявлены основные причины возникновения потребности в создании онтологий.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий. К примеру, пусть, несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную информацию для ответов на запросы пользователей или как входные данные для других приложений.

Обеспечение возможности использования знаний предметной области стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д.

Создание явных допущений в предметной области, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области.

Отделение знаний предметной области от оперативных знаний – это еще один вариант общего применения онтологий. Можно описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов.

Анализ знаний в предметной области возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении.

Приведенные выше причины определяют основные цели создания онтологий, а области применения их чрезвычайно обширны.

Метаонтология оперирует общими концептами и отношениями, которые не зависят от конкретной предметной области. Концептами метауровня являются общие понятия. Тогда на уровне метаонтологии мы получаем интенсиональное описание свойств предметной онтологии и онтологии задач. Онтология метауровня является статической, что дает возможность обеспечить здесь эффективный вывод.

Предметная онтология содержит понятия, описывающие конкретную предметную область, отношения, семантически значимые для данной предметной области, и множество интерпретаций этих понятий и отношений (декларативных и процедурных). Понятия предметной области специфичны в каждой прикладной онтологии, но отношения – более универсальны. Поэтому в качестве базиса обычно выделяют такие отношения модели предметной онтологии, как part_of, kind_of, contained_in, member_of, see_also и некоторые другие.

Онтология задач в качестве понятий содержит типы решаемых задач, а отношения этой онтологии, как правило, специфицируют декомпозицию задач на подзадачи.

Машина вывода онтологической системы в общем случае может опираться на сетевое представление онтологий всех уровней.

При этом ее функционирование будет связано: с активацией понятий и/или отношений, фиксирующих решаемую задачу (описание исходной ситуации); определением целевого состояния(ситуации); выводом на сети, заключающемся в том, что от узлов исходной ситуации распространяются волны активации, использующие свойства отношений, с ними связанных. Критерием остановки процесса является достижение целевой ситуации.

8.2. Языки описания онтологий

Язык описания онтологий – формальный язык, используемый для кодирования онтологии. Существует несколько подобных языков. Вот список некоторых из них [13-16]:

- OWL – Web Ontology Language, стандарт W3C, язык для семантических утверждений, разработанный как расширение RDF и RDFS;
- KIF (англ.)русск. (англ. Knowledge Interchange Format – формат обмена знаниями) – основанный на S-выражениях синтаксис для логики;
- Common Logic (CL) (англ.) русск. – преемник KIF (стандартизован – ISO/IEC 24707:2007);
- CycL (англ.) русск. – онтологический язык, использующийся в проекте Cyc. Основан на исчислении предикатов с некоторыми расширениями более высокого порядка;
- DAML+OIL (англ.) русск. (FIPA).

Главный элемент языка RDF – это тройка, или триплет. Тройка представляет собой совокупность трех сущностей: субъект; объект; предикат.

Предикаты еще часто называют отношениями. Тройка имеет также представление в виде графа вида субъект–предикат–объект, где субъект и объект

представлены как узлы, а предикат выступает в роли ребра, которое эти узлы соединяет.

Базовым элементом языка OWL является класс всех классов, определяемый как owl:Class. Любой OWL-класс должен быть задан как экземпляр класса owl:Class.

В языке OWL также присутствуют два предопределенных класса:

– Класс owl:Thing (сущность), обозначающий множество всех индивидов.

– Класс owl:Nothing (ничто), обозначающий пустое множество.

Каждый класс OWL является дочерним классом класса owl:Thing и родительским классом класса owl:Nothing. Наследование классов в языке OWL задается с помощью конструкции rdfs:subClassOf.

В OWL существует разделение свойств на два класса:

– Объектные свойства используются для связывания индивидов друг с другом. Объектные свойства – это экземпляры класса owl:ObjectProperty.
– Свойства типов данных связывают индивидов с так называемыми значениями типов данных (data values). Под значениями здесь подразумеваются RDF-литералы, или типы данных, определенные в XML Schema. Свойства типов данных – это экземпляры класса owl:DatatypeProperty.

Классы owl:ObjectProperty и owl:DatatypeProperty являются дочерними классами класса rdf:Property.

Язык OWL позволяет описывать различные характеристики классов и свойств, которые обычно задаются как разного рода ограничения на структуру связей между своими экземплярами. Эти ограничения выражаются в виде предопределенных соотношений, называемых в языке OWL аксиомами. В этом состоит основное отличие языка OWL от RDFS. Эти ограничения позволяют выражать в онтологии более тонкие вещи, чем с помощью RDFS.

Таких ограничений в языке OWL множество. Но все они подобраны таким образом, чтобы не снизить производительность алгоритма логического вывода по фактам, которые описаны в онтологии.

Для работы с языками онтологий существует несколько видов технологий: редакторы онтологий (для создания онтологий), СУБД онтологий (для хранения и обращения к онтологии) и хранилища онтологий (для работы с несколькими онтологиями).

Наиболее широкое распространение онтологии нашли во Всемирной паутине. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты, до категоризаций продаваемых товаров и их характеристик.

Для создания и редактирования онтологий используются специальные программы – *редакторы онтологий*. В данный момент в сети Интернет представлено множество онтологических редакторов, большинство из которых кроссплатформенны, то есть могут быть установлены на компьютер с любой операционной системой.

Самые известные из них – Ontolingua, Protégé, OntoEdit, OilEd, WebOnto, OntoSaurus, HOZO и др. Для работы этих онторедакторов необходимо установ-

ленное Java ПО, которое бесплатно доступно в Интернете. Функциональность онторедакторов можно определить по следующим параметрам:

- осуществление поиска по онтологии;
- редактирование (ввод, корректировка, удаление);
- логический контроль при вводе;
- тестирование функциональности;
- взаимодействие с другими онтологиями.

Был выбран редактор Protégé, разработанный в Стэнфордском университете (Stanford University) под руководством Марка Мьюсена для построения онтологий и баз знаний. Редактор Protégé доступен для бесплатного скачивания на официальном сайте. Этот редактор поддерживает язык OWL в последней редакции.

8.3. Основные правила разработки онтологий

В литературе предлагаются следующие основные правила разработки онтологий [13–16].

1. Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений.

2. Разработка онтологии – это обязательно итеративный процесс. В процессе создания важно возвращаться к уже созданным классам и отношениям и уточнять, и добавлять информацию в случае необходимости. Иногда полезно вносить и кардинальные изменения для улучшения общей структуры онтологии.

3. Понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей предметной области. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают предметную область. Это поможет лучше понимать онтологию людям, не являющимся разработчиками, но заинтересованными в применении ее к своим приложениям.

То есть, знание того, для чего будет использована онтология и насколько детальной или общей она будет, повлияет на многие решения, касающиеся моделирования. Среди нескольких жизнеспособных альтернатив нужно определить, какая поможет лучше решить поставленную задачу и будет более наглядной, более расширяемой и более простой в обслуживании. Также нужно помнить, что онтология – это модель реального мира и понятия в онтологии должны отражать эту реальность. После того, как будет определена начальная версия онтологии, можно оценить и отладить ее, используя ее в приложениях или в методах решения задач и/или обсудив ее с экспертами предметной области. В результате почти наверняка нужно будет пересмотреть начальную онтологию. Этот процесс итеративного проектирования, вероятно, будет продолжаться в течение всего жизненного цикла онтологии.

На рис. 8.1. представлена онтология нефтегазодобывающего предприятия и повышения экологической безопасности его работы, разрабатываемая для обеспечения оперативного мониторинга его технологической инфраструктуры.

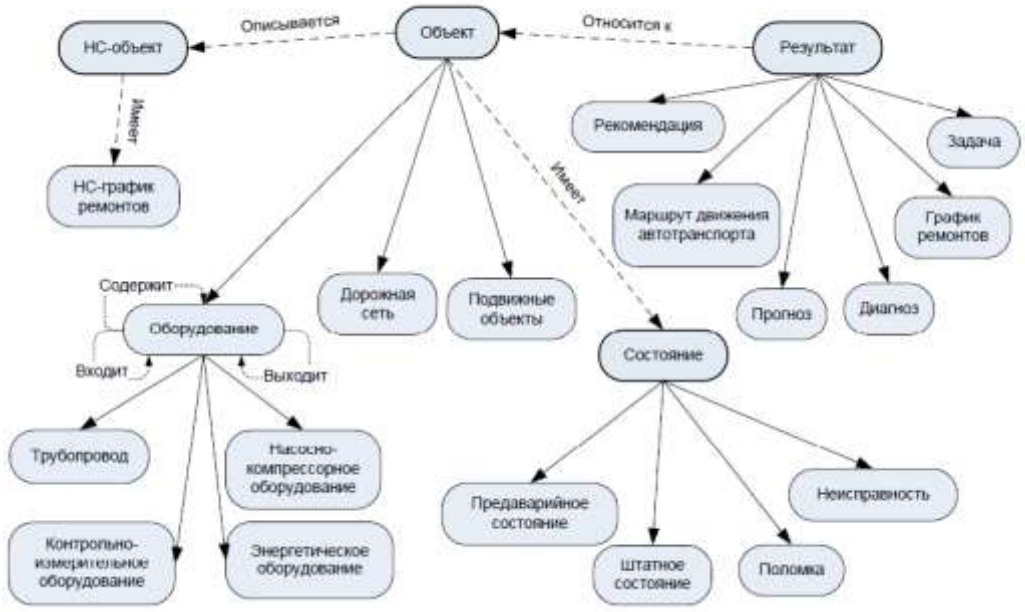


Рис. 8.1. Пример онтологии программного обеспечения нефтегазодобывающего предприятия

Существует множество разработанных онтологий в различных отраслях и сферах. Также существую онтологии, которые являются объединением ранее построенных онтологий [34–36].

8.4. Языки онтологий

Для реализации умного Web необходимо разработать стандартный язык, который был бы понятен всем поисковым программам. На настоящий момент разработаны два таких языка:

– Язык Resource Description Framework (RDF) – система описания ресурсов Web.

– Web Ontology Language (OWL) – язык онтологии Web. OWL можно рассматривать как расширение языка RDF.

– Язык RDF разработан для описания содержимого Web. В Semantic Web, когда говорят о каких-то сущностях Web, называют эти сущности ресурсами. RDF представляет собой язык для описания таких ресурсов. Ввиду того, что описания семантики документов должны быть понятны компьютерам, необходимо разработать специальные программы-агенты, которые производили бы такое чтение. Также необходимо обеспечить возможность обмена информацией между различными программными агентами. Таким образом, под RDF подразумевается не только сам язык, но также и различные дополнитель-

ные программные модули, необходимые для обеспечения полноценного чтения и обмена информацией, записанной на этом языке.

Главный элемент языка RDF – это тройка, или триплет. Тройка представляет собой совокупность трех сущностей:

1. Субъект.
2. Объект.
3. Предикат.

Предикаты еще часто называют свойствами. Тройка имеет также представление в виде графа вида субъект–предикат–объект, где субъект и объект представлены как узлы, а предикат выступает в роли ребра, которое эти узлы соединяет.

В качестве логических операций могут выступать: логическое «или» (дизъюнкция), логическое «и» (конъюнкция) и логическое следование (импликация). Имеются также кванторы существования и всеобщности, позволяющие ограничивать область применения высказывания.

Язык RDF основан на математическом аппарате логики, которая базируется на формализмах семантических сетей и фреймов. В математической логике производится явное разделение на синтаксис и семантику. Синтаксис задает язык, с помощью которого записываются различные высказывания об элементах мира данной логической системы. Семантика задает ту часть описываемого мира, которая удовлетворяет заданным ограничениям. Таких частей может быть более одной или даже бесконечно много. Каждая такая часть мира называется *моделью* данной логической системы.

Контрольные вопросы

1. Понятие онтологии в информатике.
2. Онтологии как описания знаний.
3. Описание онтологии по Груберу.
4. Компоненты онтологии: экземпляры, классы и отношения.
5. Объекты в онтологии и их атрибуты.
6. Специализированные (предметно-ориентированные) онтологии.
7. Основные сферы применения онтологий.
8. Понятие метаонтологии.
9. Язык описания онтологий.
10. Понятие элемента языка RDF (тройка, или триплет).
11. Распространение онтологии во Всемирной паутине.
12. Основные функции редактора онтологий.
13. Какой язык поддерживает редактор Protégé?
14. Основные правила разработки онтологий.

9. СИСТЕМЫ ОБРАБОТКИ ЗНАНИЙ НА БАЗЕ ПРЕЦЕДЕНТОВ

9.1. Рассуждения по прецедентам

Создание экспертных систем на базе прецедентов тесно связана с актуальной проблемой в области искусственного интеллекта и конструирования перспективных интеллектуальных (экспертных) систем – проблемой моделирования человеческих рассуждений (рассуждений «здравого смысла»).

В современных и дорогостоящих зарубежных средствах конструирования интеллектуальных (экспертных) систем (G2, RTworks и др.) практически отсутствуют развитые средства, реализующие механизмы правдоподобных рассуждений, которые способны обеспечить работу системы в условиях неопределенности как в исходной информации, получаемой от объекта управления и среды, так и в экспертных знаниях [10]. Наличие подобных механизмов рассуждений (индуктивных, абдуктивных, нечетких, аргументации, на основе аналогий и прецедентов в системах экспертной диагностики и поддержки принятия решений позволяет своевременно осуществлять диагностирование проблемной ситуации на объекте и дает возможность лицам, принимающим решения (ЛПР), принимать более адекватные и экономически выгодные управляющие воздействия на объект управления с целью нормализации проблемной ситуации.

В настоящее время основное внимание уделяется правдоподобным рассуждениям на основе прецедентов (накопленного опыта), активно применяемым в диагностических системах (медицинской диагностике, диагностике спутникового оборудования и т. д.), в юриспруденции, экспертных системах и системах машинного обучения [18–22]. Рассматривается возможность использования различных метрических алгоритмов для извлечения прецедентов из базы прецедентов системы и учета коэффициентов важности параметров объекта, задаваемых экспертом.

Рассуждения по прецедентам – это метод формирования умозаключений, опирающийся не на логический вывод от исходных посылок, а на поиск и анализ случаев формирования подобных умозаключений в прошлом [18–22].

С точки зрения решения задач, рассуждения по прецедентам – это метод получения решения путем поиска подобных проблемных ситуаций в памяти, хранящей прошлый опыт решения задач, и адаптации найденных решений к новым условиям. Применение данного метода для решения задач оправдано в случае выполнения следующих условий, касающихся природы прикладной области.

Во-первых, подобные задачи должны иметь подобные решения (принцип регулярности). В этом случае накопленный опыт решения задач может служить отправной точкой процесса поиска решения для новых подобных задач.

Во-вторых, виды задач, с которыми сталкивается решатель, должны иметь тенденцию к повторению. Это условие гарантирует, что для многих проблем в будущем будет существовать аналог в прошлом опыте.

Рассмотрим в общих чертах, что собой представляет процесс рассуждений по прецедентам.

Прецедент можно определить, как единичную запись предыдущего опыта. Какую именно информацию содержит такая запись, зависит от предметной области и целей использования прецедента.

В большинстве энциклопедических источников прецедент определяется как случай, имевший место ранее и служащий примером или оправданием для последующих случаев подобного рода. Вывод на основе прецедентов (CBR – Case-Based Reasoning) является подходом, позволяющим решить новую задачу, используя или адаптируя решение уже известной задачи. Как правило, такие методы рассуждений включают в себя четыре основных этапа, образующие так называемый цикл рассуждения на основе прецедентов или CBR-цикл [18–22].

В случае применения CBR-метода для решения задач прецедент содержит, по меньшей мере, постановку задачи и способ ее решения. Множество всех прецедентов, накопленных в процессе работы CBR-метода, формируют информационное хранилище, называемое *библиотекой прецедентов*.

CBR в общем случае представляет собой циклический процесс: решение проблемы, запоминание этого решения в качестве прецедента, решение новой проблемы и так далее. CBR-цикл может быть описан следующими тремя процессами:

1. Поиск похожего прецедента(ов) – поиск прецедента(ов), у которых постановка задачи наиболее похожа на описание новой задачи.
2. Адаптация – получение на базе найденного прецедента решения для новой задачи. Этот этап может также включать проверку полученного нового решения на корректность и толерантность к ошибкам, и, возможно, дополнительную коррекцию решения.
3. Сохранение прецедента – сохранение той части полученного опыта, которая может оказаться полезной для решения новых задач (пополнение или корректировка библиотеки прецедентов).

Процесс адаптации разделяют на повторное использование и проверку. Так как эти процессы часто оказываются сильно взаимосвязанными, разделение – это весьма условно.

Таким образом, решение каждой задачи в CBR сводится к последовательному решению подзадач поиска схожих прецедентов, получения из них (путем адаптации) решения для новой задачи и сохранения нового случая решения задачи в библиотеке прецедентов. Графически, CBR-цикл можно представить в виде следующей диаграммы (рис. 9.1).

Для процессов, входящих в CBR-цикл, а также для самого прецедента (как хранилища информации) существует множество вариантов реализации. На уровне процессов уже нет каких-то универсальных решений; реализации процессов опираются на знания о предметной и проблемной области, то есть они специфичны в каждой конкретной прикладной системе.

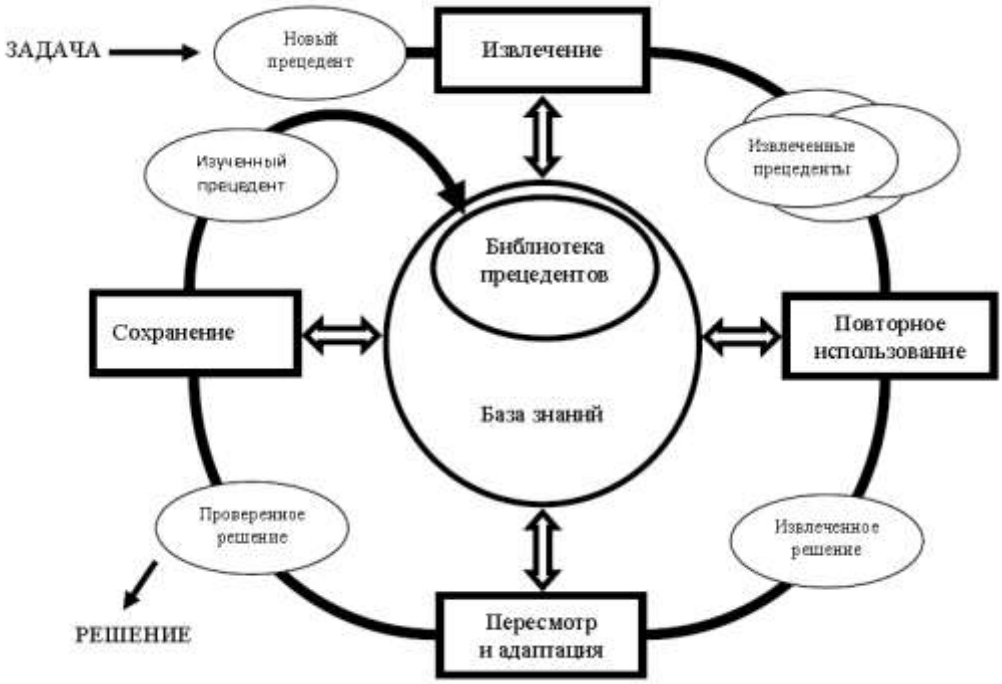


Рис. 9.1. CBR-цикл

9.2. Достоинства и недостатки использования прецедентов

Метод рассуждений по прецедентам имеет свои преимущества и недостатки по сравнению с другими методами получения решений. Среди *преимуществ* можно выделить следующее.

1. Легкость приобретения знаний (в противоположность системам, основанным на правилах). Создание системы, основанной на правилах, требует таких трудоемких этапов как получение, формализация и обобщение экспертных знаний, верификация системы на корректность и полноту. В системах, основанных на прецедентах, приобретение знаний происходит путем формального описания случаев из практики (нет необходимости обобщения, и вытекающей отсюда угрозы пере обобщения).

2. Возможность объяснения полученного решения (в противоположность системам, основанным на нейронных сетях). CBR-система может объяснить полученное решение путем демонстрации успешного прецедента с отражением показателей сходства и рассуждений, использовавшихся при адаптации прецедента к новой ситуации. Такое объяснение может быть даже лучше, чем объяснения, выдаваемые системами, основанными на правилах. Последние иногда выдают очень длинные последовательности рассуждений, а сами правила конечному пользователю (в отличие от эксперта) могут казаться неочевидными или слишком сложными.

3. Возможность работы в предметных областях, которые невозможно полностью понять, объяснить или смоделировать.

4. Возможность обучения в процессе работы. Причем обучение будет происходить только в определенных направлениях, которые реально встречаются на практике и востребованы (нет избыточности).

5. Возможность избежать повторения ошибки (обучение сбоям и их причинам для избегания их появления в дальнейшем).

6. Возможность получения решений путем модификации прецедентов позволяет уменьшить объем вычислений в предметных областях, где генерация решения «с нуля» требует больших усилий.

Основными *недостатками* использования прецедентов являются:

1. Метод применим только в областях, где выполняется принцип регулярности и имеет место повторяемость видов задач. Если все время решаются принципиально новые задачи или если решения сходных задач различны, то метод на базе прецедентов неприемлем.

2. Некомпактное (без обобщения) хранение знаний (опыта).

3. Сложность и специфичность процессов поиска подобных случаев и адаптации решения.

9.3. Примеры реализации

В работе [18–22] рассмотрены методы правдоподобных рассуждений на основе аналогий и прецедентов для систем искусственного интеллекта. Исследованы различные способы представления и извлечения прецедентов из БЗ системы. Описаны возможности разработанных программных средств, базирующихся на CBR-технологии и методах структурной аналогии.

С помощью нейросетевой ЭС формируется, предварительное заключение, предъявляемое врачу-специалисту. Но во всяких правилах возможно исключения. Для этого использовалась методика обработки прецедентов для критического анализа результатов применения правил. Это выполняется путем поиска прецедентов, аналогичных рассматриваемому случаю, если этот случай можно считать исключением из правил.

В автоматизированных системах обучения возникает проблема получения строгих функциональных зависимостей между входными и выходными параметрами, связанная со слабой формализуемостью объекта управления. В работе предполагается использовать «обходной» вариант, основанный на методе вывода по прецедентам.

Рассматриваются вопросы, связанные с реализацией механизмов правдоподобных рассуждений на основе прецедентов для систем экспертной диагностики. Основное внимание уделено проблеме извлечения прецедентов из БП системы. Предложен модифицированный метод ближайшего соседа для извлечения прецедентов. Разработаны соответствующие метрические алгоритмы извлечения прецедентов, обеспечивающие учет коэффициентов важности параметров объекта и работу с неполной информацией.

Применению прецедентов посвящено исследование средств программирования для решения задач планирования целенаправленной деятельности. Описана реализация ассоциативного планирования в решателе геометрических

задач. Предложен алгоритм поиска близких ситуаций, основанный на компиляции описаний стереотипных ситуаций в сеть специального вида. Рассмотрены критерии отбора стереотипов для получения эффективного плана. Приведены решения интеллектуальных многошаговых задач, демонстрирующие метод ассоциативного планирования на основе стереотипов и близких ситуаций.

Рассматриваются методы поиска решения на основе структурной аналогии и прецедентов в плане применения их в современных информационных систем поддержки принятия решений. Описаны базовые алгоритмы поиска решения на основе аналогии свойств и аналогии отношений. Предложены более эффективные в плане качества получаемого решения алгоритмы. Алгоритмы используют модифицированную структурную аналогию, что позволяет учитывать множество свойств, определяющих первоначальный контекст аналогии, и обеспечивает поиск решения в случае, когда такое множество свойств является пустым, а также переносить от источника аналогии на приемник только те факты, которые уместны в контексте построенной аналогии. Предложены методы определения сравнительных оценок полученных аналогий с учетом контекста и важности параметров объекта.

Метод принятия решений, основанный на совместном применении ранее не комбинировавшихся методов извлечения знаний и вывода по прецедентам, где методы добычи данных используются для автоматического отбора из большой базы прецедентов [20]. В работе предложена локальная контекстнозависимая метрика, имеющая интерпретацию расстояния и позволяющая ранжировать объекты, по отношению к исследуемому, целыми числами. При построении метрики используется предложенный авторами модифицированный метод кластерного анализа, ориентированный на распознавание объектов в ситуациях, когда объекты и кластеры имеют не полностью совпадающие наборы признаков. Эта метрика применима к широкому кругу приложений и не накладывает ограничений на типы используемых атрибутов. Что касается адаптации решения – предлагаемый метод позволяет сделать эту проблему более формализуемой. Хотя в общем случае проблема адаптации остается зависимой от предметной области, предложенный подход значительно упрощает эту задачу, так как учитывает фоновое знание.

Рассматривается модуль правдоподобного вывода по прецедентам как один из компонентов системы автоматизации исследований. Компонент представляет собой динамическую библиотеку, реализующую функции: создания модели прецедента, формирования базы прецедентов и рассуждений по прецедентам. Приведено описание архитектуры и функциональных блоков модуля [18–22].

Рассматривается реализация баз знаний прецедентов активных экспертных систем на основе ансамблевых моделей нейросетей. Решается задача распознавания прецедентов ансамблевой моделью нейросетей в процессе комплексных отказов, с учетом частичной или полной неопределенности параметров системы «временной автомат приемистости и авиационный газотурбинный двигатель».

В другом источнике рассматривается технология сборки систем из компонентов. В данной работе под компонентом системы автоматизации исследований понимается компонент, который, кроме основной функциональности, обеспечивает реализацию механизма внутренней памяти и предложенного авторами унифицированного интерфейса компонента.

Среди множества методов, которые могут быть алгоритмизированы и реализованы в виде программного кода компонентов для системы автоматизации исследований, выделен метод, основанный на изучении существующего в определенной предметной области опыта – это правдоподобный вывод по прецедентам. Выбор данного метода обусловлен тем, что зачастую в областях, где требуется поддержка исследователя (например, обеспечение надежности и безопасности сложных технических систем в нефтехимии), к моменту возникновения новой проблемы уже накоплен значительный опыт решения похожих проблем, возникавших ранее на подобном оборудовании.

Однако невозможность или сложность аналитической обработки этого опыта по ряду причин (например, отсутствие специалистов необходимой квалификации, экспертов и др.) приводит к невозможности эффективно использовать эти знания и реализовать их потенциал. Представление же этого опыта в виде прецедентов, и его автоматизированная обработка при помощи специализированных программных систем позволяют значительно повысить эффективность его повторного использования. В работе указано подробное решение проблемы при помощи прецедентов.

Нейросетевая система обработки знаний на основе прецедентов приведена на рис. 9.2.

В общей структуре нейросетевой системы обработки знаний на основе прецедентов главенствует модуль-интегратор, который управляет взаимодействием интеллектуальных и программных модулей. Интеллектуальная подсистема[14] включает следующие компоненты:

– модуль приобретения знаний – включает анализ и извлечение;

– входной информации из базы данных абонентов (БДА); входные данные преобразуются в форму прецедента или в форму продукционного нечеткого правила;

– база знаний прецедентов (БЗП);

– продукционная нечеткая база знаний (ПНБЗ) содержит правила в форме нечетких продукций;

– механизм поиска по прецедентам (МПП);

– блок обучения нейронной сети – преобразует правила из ПНБЗ в обучающие выборки для нейронной сети;

– нейро-нечеткий механизм (ННМ) – программный блок,

– реализующий структуру нечеткого контроллера на основе нейронной сети;

– блок объяснений решения;

---

14 http://ej.kubagro.ru/2013/02/pdf/24.pdf

– блок адаптации данных (АД) – преобразует результат нейросетевого поиска решения в форму нового прецедента.

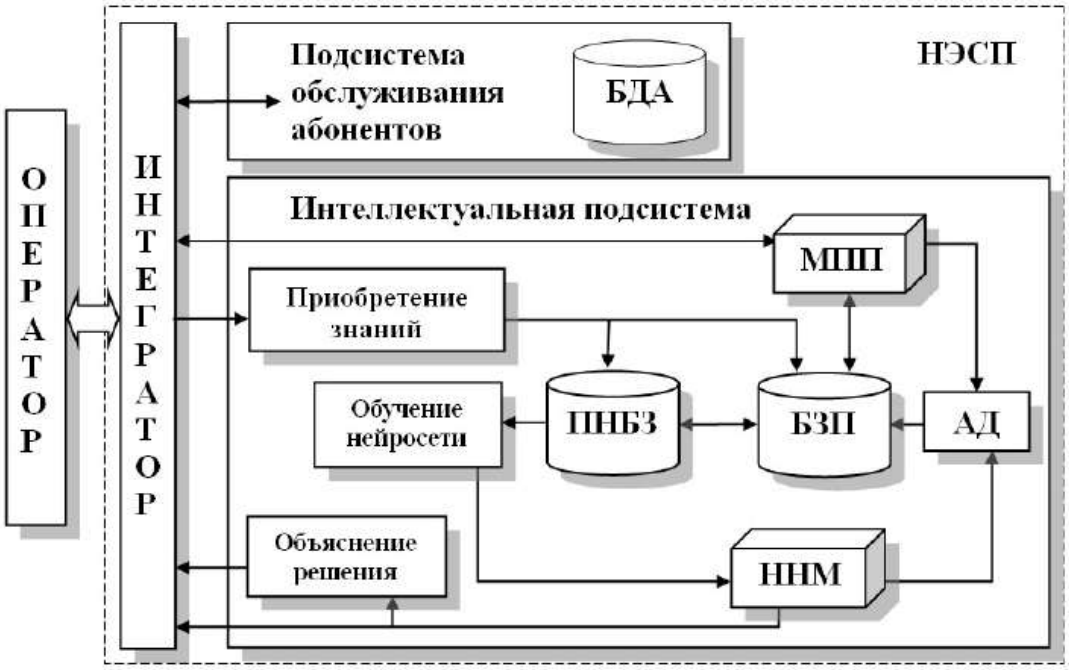


Рис. 9.2. Структура нейросетевой системы на основе прецедентов

Входные переменные представляют собой характеристики проблемы, которая возникла у абонента. В качестве выходных переменных выступают причины, повлекшие возникновение проблемы.

9.4. Принципы создания системы на базе прецедентов

Система обработки знаний на базе прецедентов должна удовлетворять следующим требованиям:

– наличие базы знаний, хранящей прецеденты;
– реализация модуля вывода на основе прецедентов;
– наличие модуля, формирующего протокол решения;
– разработка не менее 16 прецедентов для демонстрации работы программы.

Если разрабатываемая система обработки знаний на базе прецедентов предназначается для формирования кулинарных рецептов из мяса, то она принимает информацию о целевых характеристиках блюда (тип, вкусовые качества, основные ингредиенты) и формирует подходящий рецепт. Результатом работы программы должен быть рецепт последовательности операций, позволяющий приготовить такое блюдо.

Получив заказ, программа просматривает свою базу прецедентов, отыскивает в ней рецепт приготовления аналогичного блюда и адаптирует его в соответствии с особенностями текущего запроса.

Исходя из поставленных задач, программа должна содержать в себе модули извлечения прецедентов, модификации и сохранения нового прецедента. Прецеденты должны храниться в специальной базе, которая позволит описать все взаимосвязи. Структурная схема программы, имеющая отношение к манипуляции с базой прецедентов представлена на рис. 9.3.

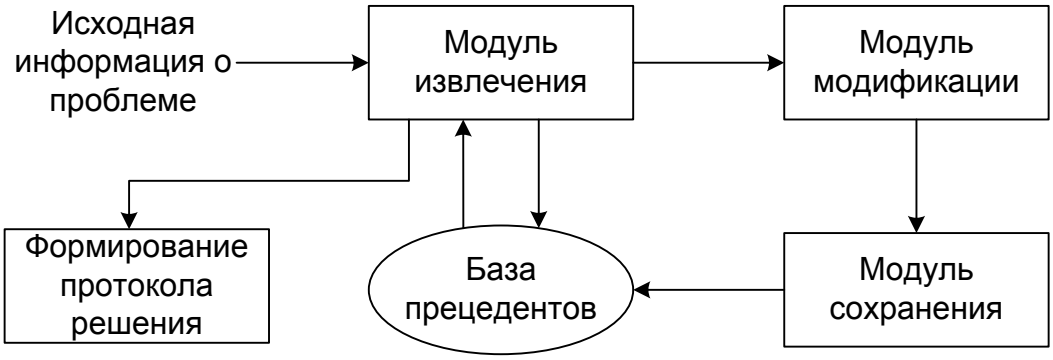


Рис. 9.3. Структурная схема программы

Для успешной реализации рассуждений на основе прецедентов необходимо обеспечить корректное извлечение прецедентов из базы прецедентов. Выбор метода извлечения прецедентов напрямую связан со способом представления прецедентов и соответственно со способом организации БП.

Основные способы представления прецедентов можно разделить на следующие группы:

- параметрические;
- объектно-ориентированные;
- специальные (в виде деревьев, графов, формул и т. д.).

В большинстве случаев для представления прецедентов достаточно простого параметрического представления, т.е. представления прецедента в виде набора параметров с конкретными значениями и решения (диагноз по проблемной ситуации и рекомендации ЛПР):

CASE(x_1, x_2, … , x_n, R),

где $x_1 \ldots x_n$ – параметры ситуации, описывающей данный прецедент ($x_1 \in X_1$, $x_2 \in X_2, \ldots, x_n \in X_n$);

R – диагноз и рекомендации ЛПР;

n – количество параметров прецедента, а X_1, … , X_n – области допустимых значений соответствующих параметров прецедента.

1. Существует целый ряд методов извлечения прецедентов и их модификаций:

1. *Метод ближайшего соседа (NN – Nearest Neighbor).* Это самый распространенный метод сравнения и извлечения прецедентов. Он позволяет довольно легко вычислить степень сходства текущей проблемной ситуации и прецедентов из БП системы. С целью определения степени сходства на множестве параметров, используемых для описания прецедентов и текущей ситуации, вводится определенная метрика. Далее в соответствии с выбранной метрикой определя-

ется расстояние от целевой точки, соответствующей текущей проблемной ситуации, до точек, представляющих прецеденты из БП, и выбирается ближайшая к целевой точка.

2. Преимуществами данного метода являются простота реализации и универсальность в смысле независимости от специфики конкретной проблемной области. К существенным недостаткам метода можно отнести сложность выбора метрики для определения степени сходства и прямую зависимость требуемых вычислительных ресурсов от размера БП, а также неэффективность при работе с неполными и зашумленными исходными данными.

3. На практике применяются модификации указанного метода. Обычно решение выбирается на основе нескольких ближайших точек (соседей), а не одной (метод k ближайших соседей). Возможно использование метода ближайшего соседа, основанного на знаниях о предметной области (определенных зависимостях между параметрами объекта).

4. *Метод извлечения прецедентов на основе деревьев решений.* Этот метод предполагает нахождение требуемых прецедентов путем разрешения вершин дерева решений. Каждая вершина дерева указывает, по какой ее ветви следует осуществлять дальнейший поиск решения. Выбор ветви осуществляется на основе информации о текущей проблемной ситуации. Таким образом, необходимо добраться до концевой вершины, которая соответствует одному или нескольким прецедентам. Если концевая вершина связана с некоторым подмножеством прецедентов, то тогда для выбора, наиболее подходящего из них может использоваться метод ближайшего соседа. Такой подход рекомендуется применять для больших бах прецедентов, т.к. основная часть работы по извлечению прецедентов выполняется заранее на этапе построения дерева решений, что значительно сокращает время поиска решения.

5. *Метод извлечения прецедентов на основе знаний.* В отличие от методов, описанных выше, данный метод позволяет учесть знания экспертов (ЛПР) по конкретной предметной области (коэффициенты важности параметров, выявленные зависимости и т.д.) при извлечении. Метод может успешно применяться совместно с другими методами извлечения прецедентов, особенно когда база прецедентов имеет большие размеры, и предметная область является открытой и динамической.

6. *Метод извлечения с учетом применимости прецедентов.* В большинстве систем, использующих механизмы рассуждений на основе прецедентов, предполагается, что наиболее схожие с текущей проблемной ситуацией прецеденты являются наиболее применимыми в этой ситуации. Однако это не всегда так. В основе понятия извлечения на основе применимости (адаптируемости) лежит то, что извлечение прецедентов базируется не только на их сход-

стве с текущей проблемной ситуацией, но и на том, насколько хорошую для желаемого результата модель они собой представляют. На выбор извлекаемых прецедентов влияет возможность их применения в конкретной ситуации. В некоторых системах эта проблема решается путем сохранения прецедентов вместе с комментариями по их применению. Использование указанного подхода позволяет сделать поиск решения более эффективным, заранее отбрасывая часть заведомо неперспективных прецедентов. Помимо рассмотренных методов для извлечения прецедентов могут успешно применяться и другие методы (например, аппарат искусственных нейронных сетей).

Сравнение систем, основанных на правилах и прецедентах представлено в [22]. Существует достаточно сильная аналогия между системами, основанными на правилах и прецедентах. И те, и другие необходимо каким-то образом индексировать, чтобы обеспечить эффективное извлечение. И те, и другие выбираются в результате сопоставления, причем выбор и ранжирование производятся на основании фоновых знаний, хранящихся в каких-либо дополнительных структурах, например, в виде фреймов (в MYCIN аналогичную роль выполняют таблицы знаний).

Различия между этими двумя классами систем. Они суммированы ниже. Правила являются образцами – содержат переменные и не описывают непосредственно решение, а прецеденты являются константами.

Правило выбирается на основе точного сопоставления антецедента и данных в рабочей памяти. Прецедент выбирается на основе частичного сопоставления, причем учитываются еще и знания о сущности характеристик, по которым выполняется сопоставление.

Применение правил организовано в виде итерационного цикла – последовательности шагов, приводящих к решению. Прецедент можно рассматривать как приближенный вариант полного решения. Иногда, однако, появляется возможность итеративно проводить аналогию с разными прецедентами, которые «подходят» для различных частей проблемы.

Построение суждений на основе прецедентов поддерживает и другую стратегию решения проблем, которую называют «извлечение и адаптация». Эта стратегия существенно отличается и от эвристической классификации, и от стратегии «предложение и проверка».

В новом подходе есть нечто очень близкое нам интуитивно, поскольку весьма напоминает наш повседневный опыт. Даже на первый взгляд ясно, как привлекательно вспомнить аналогичный случай, принесший успех в прошлом, и поступить так же. Редко кто из нас затрудняет себя «нудными рассуждениями», когда можно быстро извлечь готовое решение.

Контрольные вопросы

1. Метод рассуждения по прецедентам.
2. Понятие прецедент.
3. Принцип работы CBR-метода.
4. Достоинства и недостатки использования прецедентов
5. Методы, связанные с реализацией механизмов правдоподобных рассуждений на основе прецедентов.
6. Нейросетевая система обработки знаний на основе прецедентов.
7. Принципы создания системы на базе прецедентов.
8. Способы представления прецедентов.
9. Методы извлечения прецедентов.
10. Метод ближайшего соседа.
11. Метод извлечения прецедентов на основе деревьев решений.
12. Сравнение систем, основанных на правилах и прецедентах.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

АД – блок адаптации данных

БДА – базы данных абонентов

БЗ – база знаний

БЗП – база знаний прецедентов

ГЭС – гибридная экспертная система.

ЛПР – лицо, принимающее решение

МПП – механизм поиска по прецедентам

МСПРС – многослойная сеть с прямым распространением сигналов

МЭС – мягкая экспертная система

НКК – нечеткая когнитивная карта

ННМ – нейро-нечеткий механизм

НС – нейронная сеть

НЭС – нейросетевая экспертная система

НЭСП – нейросетевая экспертная система на основе прецедентов

ОВ – обучающая выборка

ПНБЗ – продукционная нечеткая база знаний

ППФ – правильно построенная функция

СКО – среднеквадратичная ошибка

ЭС – экспертная система

CBR – Case-Based Reasoning

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Люгер, Д. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем [Текст] / Д. Ф. Люгер. – 4-е изд. – Москва: Вильямс, 2003. – 864 с.
2. Хайкин, С. Нейронные сети [Текст]: полный курс / С. Хайкин; пер. с англ. Н. Н. Куссуль, А. Ю. Шелестова. – 2-е изд., испр. – Москва [и др.]: Вильямс, 2008. – 1103 с.
3. Джексон, П. Введение в экспертные системы [Текст]: учеб. пособие / П. Джексон. – 3-е изд.– Москва; Санкт-Петербург; Киев: Вильямс, 2001. – 624 с.
4. Ростовцев, В. С. Принципы построения экспертных систем [Электронный ресурс]: учеб. пособие / В. С. Ростовцев; ВятГУ, ФАВТ, каф. ЭВМ. – Киров, 2007. – 155с.
5. Ростовцев, В. С. Искусственные нейронные сети [Электронный ресурс]: учеб. для студентов направления 230101.68.05 / В. С. Ростовцев; ВятГУ, ФАВТ, каф. ЭВМ. – Киров, 2014. – 208 с.
6. Ростовцев, В. С. Теория и применение нечеткой логики [Электронный ресурс]: учеб. пособие для студ. направления 09.04.01 «Информатика и вычислительная техника» всех профилей подготовки, всех форм обучения / В. С. Ростовцев; ВятГУ, ФАВТ, каф. ЭВМ. – Киров, 2016. – 111 с.
7. Ярушкина, Н. Г. Основы теории нечетких и гибридных систем [Текст]: учеб. пособие /Н. Г. Ярушкина. – Москва: Финансы и статистика, 2004. – 320 с.
8. Башлыков, А. А. Экспертная диагностическая система как компонент интеллектуальной системы поддержки принятия решений реального времени [Текст] / А. А. Башлыков, А. П. Еремеев // Новости искусственного интеллекта. – 2002. – №3. – С. 35–40.
9. Савушкин, С. А. Нейросетевые экспертные системы [Текст] / С. А. Савушкин // Нейрокомпьютер. – 1992. – №2. – С. 29–36.
10. Жернаков, С. В. Нейросетевая база знаний прецедентов активной экспертной системы для комплексного контроля и диагностики параметров авиационного двигателя[Текст] / С. В. Жернаков // Информационные технологии. – 2002. – №5. – С. 45–53.

Интернет-источники

1. Ветров Д. П. Программный комплекс для проектирования экспертных систем «ExSys» [Электронный ресурс] / Д. П. Ветров, Д. А. Кропотов // Математические методы распознавания образов (ММРО 11): докл. 11 Всерос. конф. – Москва, 2003. – Режим доступа: http://www.ccas.ru/mmro/received.html. – 14.10.2019.
2. Экспертные системы [Электронный ресурс]: ил. самоучитель. – Режим доступа: http://claw.ru/enciklopedija-programmirovanija/ekspertnie-sistemi-samouchitel.html. – 14.10.2019.

3. McGuinness, D. L. Ontologies Come of Age [Electronic research] / D. L. McGuiness // Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential / ed. by D. Fensel [et al.]. – Cambridge, MA: MIT Press, 2003. – Access mode: http://www.ksl.stanford.edu/people/dlm/papers/ontologies-come-of-age-abstract.html. – 14.10.2019.

4. Лапшин, В. А. Онтологии в компьютерных системах [Текст] / В. А. Лапшин. – Москва: Научный мир, 2010. – 222 с.

5. Онтологии и тезаурусы: модели, инструменты, приложения [Текст]: учеб. пособие / Б. В. Добров [и др.]. – Москва: Бином. Лаборатория знаний: ИНТУИТ, 2009. – 173 с.

6. Онтологии и тезаурусы [Текст]: учеб. пособие / В. Д. Соловьев [и др.]. – Казань: Изд-во Казанск. гос. ун-та; Москва. Изд-во МГУ, 2006. – 157 с.

7. Аксенов, К. А. Построение оболочки экспертных систем для предметной области процессов преобразования ресурсов [Электронный ресурс] / К. А. Аксенов // Современные проблемы науки и образования. – 2012. – № 6. – Режим доступа: http://www.science-education.ru/ru/article/view?id=7849. – 15.10.2019.

8. Варшавский, П. Р. Реализация метода правдоподобных рассуждений на основе прецедентов для интеллектуальных систем поддержки принятия решений [Текст] / П. Р. Варшавский // Десятая национальная конференция по искусственному интеллекту с международным участием КИИ-2006: сб. тр., Обнинск, 25–28 сент. 2006 г. В 3 т. Т. 1. – Москва: Физматлит, 2006. – С. 303–311.

9. Варшавский, П. Р. Методы правдоподобных рассуждений на основе аналогий и прецедентов для интеллектуальных систем поддержки принятия решений [Текст] / П. Р. Варшавский, А. П. Еремеев // Новости Искусственного Интеллекта. – 2006. – № 3. – С. 39–62.

10. Поминчук, Е. В. Рассуждения, основанные на прецедентах [Электронный ресурс]: [статья] / Е. В. Поминчук. – Режим доступа: http://masters.donntu.org/2011/fknt/pominchuk/library/article5.htm. – 15.10.2019.

11. Малыхина, М. П. Оценка эффективности гибридизации интеллектуальных методов на примере нейросетевой экспертной системы на основе прецедентов [Текст] / М. П. Малыхина, Ю. В. Бегман // Научный журнал КубГАУ. – 2013. – № 86(02). – Режим доступа: http://ej.kubagro.ru/2013/02/pdf/24.pdf. – 17.10.2019.

12. Сравнение систем, основанных на правилах и прецедентах [Электронный ресурс]: [статья]. – Режим доступа: http://www.hardline.ru/selfteachers/Info/Programming/Introduction%20to%20expert%20systems/Glava%2022/Index10.htm. – 17.10.2019.

13. Двенадцать лучших Python-библиотек для Data Science [Электронный ресурс]: [статья]. – Режим доступа: https://geekbrains.ru/posts/python_data_science. – 17.10.2019.

14. Kothari, A. Top 9 Frameworks in the World of Artificial Intelligence [Электронный ресурс] = Топ – 9 фреймворков в мире искусственного интеллекта : пер. с англ. Ч. 1 / A. Kothari. – Режим доступа:

https://medium.com/nuances-of-programming/топ-9-фреймворков-в-мире-искусственного-интеллекта-часть-1-40015cfb98dd. – 17.10.2019.

15. Что такое свёрточная нейронная сеть [Электронный ресурс]: [статья]: пер. с англ. – Режим доступа: https://habr.com/post/309508/. – 17.10.2019.

16. Сверточная нейронная сеть [Электронный ресурс]: определение понятия. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%91%D1%80%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C. – 17.10.2019.

17. Сверточные нейронные сети, или как научить компьютер «видеть» [Электронный ресурс]: [статья]. – Режим доступа: http://datareview.info/article/svertochnyie-neyronnyie-seti-ili-kak-nauchit-kompyuter-videt/. – 17.10.2019.

18. Сверточные нейронные сети: взгляд изнутри [Электронный ресурс]: [статья]. – Режим доступа: http://ru.datasides.com/code/cnn-convolutional-neural-networks/. – 17.10.2019.

19. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество [Электронный ресурс]: [статья]. – Режим доступа: https://habr.com/post/348000/. – 17.10.2019.

20. Сверточная нейронная сеть, часть 2: обучение алгоритмом обратного распространения ошибки [Электронный ресурс]: [статья]. – Режим доступа: https://habr.com/post/348028/. – 17.10.2019.

21. Обучение нейронной сети. Алгоритм обратного распространения ошибок [Электронный ресурс]: [статья]. – Режим доступа: https://microtechnics.ru/obuchenie-nejronnoj-seti-algoritm-obratnogo-rasprostraneniya-oshibok. – 17.10.2019.

22. Сверточные нейронные сети [Видеозапись]: [учеб. видеокурс]. – Режим доступа: https://www.youtube.com/watch?v=52U4BG0ENiM. – 17.10.2019.

23. Нейронные сети: практическое применение [Электронный ресурс]: [статья]. – Режим доступа: https://habr.com/post/322392/. – 17.10.2019.

24. Библиотеки глубокого обучения [Электронный ресурс]: [учеб. видеокурс]. – Режим доступа: https://www.youtube.com/watch?v=9xfPb2hiqNY&list=PLtPJ9lKvJ4oiz9aaL_xcZd-x0qd8G0VN_&index=4. – 17.10.2019.

25. Саханков, И. А. Нечеткие когнитивные карты как средство анализа качественных систем в динамике [Электронный ресурс] / И. А. Саханков, А. С. Федулов // Естественные и математические науки в современном мире: сб. ст. VI междунар. науч.-практ. конф. – Новосибирск: СибАК, 2013. – Режим доступа: https://sibac.info/conf/naturscience/vi/33085https://sibac.info/conf/naturscience/vi/33085. – 17.10.2019.

26. Паклин, Н. Б. Нечетко-когнитивный подход к управлению динамическими системами [Текст] / Н. Б. Паклин // Искусственный интеллект. – 2003. – № 4. – С. 342–349.

27. Максимов, В. И. Когнитивные технологии для поддержки принятия управленческих решений [Электронный ресурс]: [статья] / В. И. Максимов, Е. К. Корноушенко, С. В. Качаев. – Режим доступа: http://www.iis.ru/events/19981130/maximov.ru.html. – 17.10.2019.

Учебное издание

Ростовцев Владимир Сергеевич

СИСТЕМЫ ОБРАБОТКИ ЗНАНИЙ

Учебное пособие

Авторская редакция
Тех. редактор А. В. Нилова

Подписано в печать 23.10.2019. Печать цифровая. Бумага для офисной техники.
Усл. печ. л. 10,35. Тираж 5 экз. Заказ №5729.

Федеральное государственное бюджетное образовательное учреждение
высшего образования «Вятский государственный университет».

610000, г. Киров, ул. Московская, 36, тел.: (8332) 74-25-63, http://vyatsu.ru