Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный университе
т имени Франциска Скорины»

Н. Б. ОСИПЕНКО

Интеллектуальные информационные системы

ТЕКСТЫ ЛЕКЦИЙ
для студентов специальности 1 – 31 03 03 01
«Прикладная математика (научно-производственная деятельность)»

Гомель
УО «ГГУ им. Ф.Скорины»
2012

УДК 519.68 (075.8)
ББК 22.18я73
О 519

Рецензент:
кафедра математических проблем управления учреждения образования «Гомельский государственный университет имени Франциска Скорины»

Рекомендовано к изданию научно-методическим советом учреждения образования «Гомельский государственный университет имени Франциска Скорины»

Осипенко, Н. Б.

О 519 Интеллектуальные информационные системы : тексты лекций для студентов специальности 1 – 31 03 03 01 «Прикладная математика (научно-производственная деятельность)» / Н. Б. Осипенко; М – во образования РБ, Гомельский государственный университет им. Ф. Скорины. – Гомель : ГГУ им. Ф. Скорины, 2012. – 184с.

Тексты лекций ставят своей целью оказание помощи студентам в усвоении методов и программных средств аналитической обработки данных, позволяющих решать плохо формализованные задачи с помощью индуктивного вывода, нечеткой логики, нейросетевых и генетических и др. алгоритмов. Адресованы студентам специальности 1 – 31 03 03 01 «Прикладная математика (научно-производственная деятельность)».

УДК 519.68 (075.8)
ББК 22.18я73

Содержание

Введение

Целью текстов лекций является овладение студентами основами технологии применения теории и практики аналитической обработки данных при исследовании сложных систем из различных, в том числе, и плохо формализованных предметных областей. Специалист по прикладной математике должен владеть основами математического мышления, такими как индукция и дедукция, принципами математических рассуждений и математических доказательств, методами математической и прикладной статистики – интеллектуального анализа данных, математического моделирования и моделирования человеческих рассуждений; иметь представление о проблемах искусственного интеллекта, способах представления знаний и манипулирования ими (об инженерии знаний), об особенностях исследования сложных систем разного типа.

Тексты лекций посвящены теоретическим и организационно-методическим вопросам разработки и применения методов аналитической обработки данных и вопросам их применения при проектирования операций при исследовании сложных систем. Наиболее освоенные методы решения прикладных задач основаны на хорошо формализованных алгоритмах, полученных в результате построения математических моделей предметных областей. Чаще всего это трудоемкие расчеты по известным формулам либо простые последовательности действий, приводящие после многократного применения к желаемому результату. В практической деятельности многие задачи относятся к плохо формализованным, для которых неизвестны аналитические зависимости или цепочки действий, приводящих к результату без интеллектуального вмешательства человека. Методы решения плохо формализованных задач имеют дело с обработкой эмпирических данных. Поэтому для их решения требуется освоение способов организации хранения и выборки данных. К методам, позволяющим решать плохо формализованные задачи относятся индуктивный вывод, экспертные системы, нейросетевые алгоритмы, генетические алгоритмы, нечеткая логика, коллектив решающих правил и др.

Раздел 1 Принципы построения систем, ориентированных на анализ данных

Тема 1 Структура исследований в области искусственного интеллекта

1.1 Систематизация исследований в области искусственного интеллекта

1.2 Интеллектуальные информационно-поисковые системы

1.3 Интеллектуальные пакеты прикладных программ

1.4 Направления исследований в области искусственного интеллекта

1.1 Систематизация исследований в области искусственного интеллекта

Выделим в области искусственного интеллекта четыре группы исследований: имитация творческих процессов, интеллектуализация ЭВМ, новейшая технология решения задач, роботы.

В рамках исследований первой группы созданы программы, например, шахматной игры или сочиняющие музыку, доказательства теорем или имитации способностей человека к обучению. Результаты этих исследований носят фундаментальный характер. Они позволяют формировать методы и приемы искусственного интеллекта.

Основной целью второй группы исследований является построение ЭВМ, с которыми могут общаться неподготовленные пользователи. В рамках этих исследований созданы программные средства для систем общения, баз данных, систем планирования решения задач.

К третьей группе отнесены все исследования, направленные на создание новых принципов обработки информации и решения задач. Эти принципы используют характерные особенности методов искусственного интеллекта. Эти методы оперируют со знаниями так же, как это делают люди, когда выполняют творческую работу. В целом исследования этой группы ориентированы на планирование в АСУ, системы автоматизированного проектирования, оперативное управление и автоматизацию научных исследований.

Выделим четыре типа систем этой группы:

- интеллектуальные информационно-поисковые системы (ИПС);
- интеллектуальные пакеты прикладных программ (ИППП);
- расчетно-логические системы (РЛС);
- экспертные системы (ЭС).

Исторически первыми системами, нацеленными на автоматизацию интеллектуальных функций, связанных с деятельностью людей были ИПС. ИППП дают возможность конечному пользователю решать на ЭВМ задачи, давая их содержательные описания и определяя значения исходных данных без программирования процесса решения задачи. Создание развитых методов и инструментальных средств реализации интеллектуальных пакетов прикладных программ является одной из целей японского проекта ЭВМ пятого поколения. Обычно интеллектуальный пакет прикладных программ состоит из системы общения, работающей по «укороченной схеме», планировщика, функциональной семантической сети, являющейся системой представления процедурных знаний, и функционального наполнения в виде библиотек прикладных программ и/или их модулей. С помощью инструментальной системы и непроцедурного языка разработчик строит для конечного пользователя математическую модель предметной (проблемной) области в виде функциональной семантической сети. Такая модель и заложенные в систему методы работы с ней позволяют конечному пользователю на доступном ему входном (близком к естественному) языке решать задачи из своей предметной области по их описаниям и значениям исходных данных. При этом рабочая программа автоматически синтезируется планировщиком из набора программных модулей, т.е. функции программиста выполняются планировщиком.

РЛС являются развитием ИППП. РЛС ориентированы на группу людей, решающих общую задачу. Например, задачи планирования или проектирования. Крупные задачи такого типа разбиваются на иерархическую совокупность задач, решаемых отдельными группами пользователей. Для организации взаимодействия между пользователями одного и разных рангов РЛС должны иметь специальные средства обмена информацией, слежения за сохранением информационного контекста.

ЭС используются при решении достаточно трудных для экспертов задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в рассматриваемой проблемной области. Достоинство применения экспертных систем заключается в возможности принятия решений в уникальных ситуациях, для которых алгоритм заранее не известен и формируется по исходным данным в виде цепочки рассуждений (правил принятия решений) из базы знаний. Причем решение задач предполагается осуществлять в условиях неполноты, недостоверности, многозначности исходной информации и качественных оценок процессов. ЭС является инструментом, усиливающим интеллектуальные способности эксперта, и может выполнять следующие роли: консультанта для неопытных или непрофессиональных пользователей; ассистента в связи с необходимостью анализа экспертом различных вариантов принятия решений; партнера эксперта по вопросам, относящимся к источникам знаний из смежных областей деятельности.

1.2 Интеллектуальные информационно-поисковые системы

Процесс поиска текстовой информации включает в себя следующие этапы:

1) формализация пользователем поискового запроса;

2) предварительный отбор тестовых документов, содержащих формальные признаки наличия интересующей информации;

3) анализ отобранных документов (лексический, морфологический, синтаксический, семантический);

4) оценка соответствия смыслового содержания найденной информации требованиям поискового запроса.

Данные этапы выполняются в полном объеме человеком при неавтоматизированном поиске, а эффективность их реализации определяется интеллектуальными способностями человека. Все вышеперечисленные этапы могут быть автоматизированы на основе использования систем искусственного интеллекта и экспертных систем.

Реализация полного лингвистического анализа текстовой информации предполагает решение следующих задач:

Лексический анализ заключается в разборе текстовой информации на отдельные абзацы, предложения, слова, определении национального языка изложения, типа предложения, выявлении типа лексических выражений (бранных, жаргонных слов) и т.д. Он не представляет существенной сложности для реализации.

Морфологический анализ сводится к автоматическому распознаванию частей речи каждого слова текста (каждому слову ставится в соответствие лексико-грамматический класс). Данная задача может быть выполнена для русского языка практически со стопроцентной точностью благодаря его развитой морфологии. В английском языке алгоритм, присваивающий каждому слову в тексте наиболее вероятный для данного слова лексико-грамматический класс (синтаксическую часть речи), работает с точностью около 90 %, что обусловлено лексической многозначностью английского языка.

Синтаксический анализ заключается в автоматическом выделении семантических элементов предложения – именных групп, терминологических целых, предикативных основ. Это позволяет повысить интеллектуальность процесса обработки тестовой информации на основе обеспечения работы с более обобщенными семантическими элементами.

Семантический анализ заключается в определении информативности текстовой информации и выделении информационно-логической основы текста. Проведение автоматизированного семантического анализа текста предполагает решение задачи выявления и оценки смыслового содержания текста. Данная задача является трудно формализуемой вследствие необхо-

димости создания совершенного аппарата экспертной оценки качества информации.

Реализация семантического анализа текстовой информации предполагает обязательное использование экспертных систем, систем искусственного интеллекта для выявления смыслового содержания информации. В настоящее время отсутствуют сложившиеся подходы к реализации задачи семантического анализа текстовой информации, что во многом обусловлено исключительной сложностью проблемы и недостаточно полной проработкой научного направления создания систем искусственного интеллекта. Поэтому ***существующие информационные технологии не обеспечивают эффективной реализации поисковых систем.***

Это обусловливает низкую адекватность найденной по запросу пользователя информации, то есть возврат системой большого объема малоинформативных документов. Проблема усугубляется низкой скоростью получения документов из Интернета, необходимостью просмотра пользователем всех найденных документов и оценки их информационного содержания в неавтоматизированном режиме, а также наличием специально создаваемых (вредоносных) информационных технологий, препятствующих эффективной реализации в поисковых системах автоматической оценки содержания найденных документов.

Существуют два основных класса информационно-поисковых систем:

1) поисковые системы;
2) поисковые каталоги.

Существует несколько категорий поиска: по ключевым словам; с булевой логикой объединения слов; по словосочетаниям; с учетом расстояния между словами; с учетом регистра; по семантике (концептуальный); по шаблону (подобию); по полям документа.

Поисковые системы обеспечивают автоматическую индексацию большого количества документов, но не обладают развитыми средствами искусственного интеллекта для экспертной оценки смыслового содержания информации. Этим обусловлена низкая релевантность ответа поисковых систем (релевантность – степень адекватности результатов поиска запросу пользователя).

Поисковые каталоги обеспечивают большую релевантность ответа за счет предварительной обработки документов редакторами в ручном режиме. Однако информационная полезность таких каталогов, как правило, ограничена небольшим количеством проиндексированных документов, большими затратами средств на поддержание актуальности базы проиндексированных документов и, следовательно, низкой оперативностью ее обновления.

Методический аппарат «интеллектуального поиска» текстовой информации позволяет реализовать автоматизацию всех этапов лингвистического анализа (лексического, морфологического, синтаксического и семантическо-

го). Данная технология соединяет преимущества автоматического индексирования документов в поисковых системах с экспертной обработкой их содержания в системах искусственного интеллекта.

Рисунок 1.2 – Схема технологии «интеллектуального поиска»

Реализация указанных функциональных возможностей достигается за счет:

1) углубленного лексического анализа текстовой информации, обеспечивающего подготовительную нормализацию обрабатываемого текста;

2) уникальной структуры морфологического словаря, включающего все морфологические и семантические характеристики слов, а также слова – синонимы и тематически связанные слова;

3) детального морфологического анализа, обеспечивающего определение частей речи с учетом семантики запроса пользователя и обрабатываемой текстовой информации;

4) поиска текстовой информации по синонимам и тематически связанным словам;

5) автоматизированного синтаксического анализа членов предложения и связей между ними;

6) отбора текстовой информации на основе семантического анализа ее соответствия запросу пользователя;

7) автоматической оценки релевантности предложений текстов запросу пользователя с обеспечением синтеза семантически полного ответа поисковой системы.

Новые качества интеллектуальной информационно-поисковой системы:

1) Обработка запроса пользователя, представленного на естественном языке.

2) Реализация диалога интеллектуальной поисковой системы с пользователем в ходе уточнения введенного им запроса и формирования ответа системы.

3) Возможность автоматического перевода запроса пользователя с естественного языка на формализованные языки запросов существующих поисковых систем.

4) Обеспечение поиска с учетом смыслового содержания многозначных слов.

5) Реализация поиска с учетом синонимов и тематически связанных слов.

6) Повышение релевантности результатов поиска запросу пользователя на основе учета семантики запроса и синтеза семантически полного ответа поисковой системы.

7) Обеспечение автоматической интегральной оценки семантического смысла проиндексированной текстовой информации.

Рассмотренные выше особенности построения технологии «интеллектуального поиска» и достигаемые за счет них новые качества поисковой системы обеспечивают существенное снижение «информационного шума» и значительное повышение оперативности формирования ответа системы, адекватного запросу пользователя.

Таблица 1.1 – Сравнительный анализ основных параметров технологии

Характеристики систем	Поисковая система Яndex	Интеллектуальная информационно – поисковая систем
Реализуемые этапы лингвистического анализа	лексический, морфологический, синтаксический (частично)	лексический, морфологический, синтаксический, семантический
Основные разделы морфологического словаря	основы слов, морфологические формы слов	основы слов, морфологические формы слов, синонимы слов, тематические слова, семантика слов
Типы запроса пользователя	ключевые слова, формализованный язык запросов (иногда)	запрос на естественном языке, ключевые слова
Обработка текстов на национальных языка	русский, английский	русский, любой иностранный (в перспективе)
Диалог системы с пользователем при вводе запрос	Отсутствует	уточнение сформированного перечня ключевых слов; уточнение семантики многозначных слов, уточнение семантики ответа системы
Формы ответа систем	упорядоченный перечень ссылок на тексты, содержащие ключевые слова	упорядоченный перечень ссылок на тексты, содержащие ключевые слова; абзацы текста, содержащие ключе-

		вые слова; восстановленный проиндексированный текст; семантически синтезированный ответ интеллектуальной информационно – поисковой системы

1.3 Интеллектуальные пакеты прикладных программ

Интеллектуальные пакеты прикладных программ дают возможность конечному пользователю решать на ЭВМ задачи, давая их содержательные описания и определяя значения исходных данных без программирования процесса решения задачи. Разработка пакетов прикладных программ для непрограммирующих пользователей была начата в середине 60-х годов. Одними из первых в этой области были созданы пакеты ФИХАР, АССА, СИРИУС, позднее – системы ПРИЗ, СПОРА и МАВР [11, стр. 234].

Общая схема работы интеллектуального пакета прикладных программ. Обычно интеллектуальный пакет прикладных программ состоит из системы общения, работающей по «укороченной схеме», планировщика, функциональной семантической сети, являющейся системой представления процедурных знаний, и функционального наполнения в виде библиотек прикладных программ и/или их модулей. Модульный принцип формирования наполнения программных комплексов сформулирован и развит школами российских академиков А. А. Самарского и Н.Н. Яненко.

В упомянутом пакете ПРИЗ с помощью инструментальной системы того же названия ПРИЗ и непроцедурного языка УТОПИСТ разработчик строит для конечного пользователя математическую модель предметной (проблемной) области в виде функциональной семантической сети. Такая модель и заложенные в систему методы работы с ней позволяют конечному пользователю на доступном ему входном языке решать задачи из своей предметной области по их описаниям и значениям исходных данных. При этом рабочая программа автоматически синтезируется планировщиком из набора программных модулей, т.е. функции программиста выполняются планировщиком. Заметим, что, освоив язык УТОПИСТ, конечный пользователь сам сможет формировать функциональную семантическую сеть предметной области.

В системе МАВР математическая модель в виде функциональной семантической сети строится автоматически по описанию проблемы на языке конечного пользователя. Функциональная семантическая сеть – это двудольный граф с двумя типами вершин: вершины-параметры (подлежащие вычислению или задаваемые) и вершины-отношения, определяющие функциональные отношения между параметрами.

Проиллюстрируем особенности работы планировщика по синтезу рабочих программ из заранее заготовленных программных модулей для семантической сети такой предметной области, как преподавание математики в шко-

ле для решения задач с треугольником. При построении сети использованы известные тригонометрические формулы. Сеть представляет собой двудольный неориентированный граф с двумя типами вершин: вершины-параметры треугольника: стороны треугольника и углы треугольника, а также вершины-отношения между параметрами. Теперь можно приступить к решению какой-либо задачи. Пусть конечный пользователь сформулировал задачу: *«Определить площадь треугольника S по стороне c и прилегающим к ней углам* α *и* β «. Задание поступает в систему: проверяется корректность постановки задачи (т. е. выполнение условия $\alpha + \beta < \pi$), определяется минимальная замкнутая система отношений, позволяющая решить задачу и соответственно из каких программных модулей образуется цепочка рабочей программы. Таким образом, происходит преобразование неориентированного двудольного графа отношений в ориентированный граф решения задачи и соответственно синтез рабочей программы решения задачи из цепочки программных модулей.

1.4 Направления исследований в области искусственного интеллекта

Термин «искусственный интеллект» ИИ [5] (AI – artificial intelligence) был предложен в 1956г. на семинаре с аналогичным названием в Дартмутском колледже (США). Семинар был посвящен разработке логических (не вычислительных) задач. В переводе на русский язык слово intelligence означает «умение рассуждать разумно», а не интеллект, для которого есть английское слово intellect.

После признания ИИ самостоятельным научным направлением в нем обозначились два направления нейрокибернетика и кибернетика «черного ящика».

Нейрокибернетика ориентирована на программно-аппаратное моделирование структур, подобных мозгу, на основании идеи, что единственным существом, способным мыслить, является человек, поэтому любое «мыслящее» устройство должно каким-то образом воспроизводить его структуру.

Кибернетика «черного ящика» основана на идее, что не важно, как устроено «мыслящее» устройство, а то, что на заданные входные воздействия оно реагирует так же, как человеческий мозг. В середине XX века велись интенсивные поиски моделей и алгоритмов человеческого мышления и разработка первых программ на их основе. Представители гуманитарных наук (философы, психологи, лингвисты) не смогли предложить таких алгоритмов. Кибернетики создали свои модели и подходы, среди них можно упомянуть: модель лабиринтного поиска, эвристические алгоритмы, метод резолюций, обратный вывод Маслова, представление знаний – фреймовое и семантическими сетями, языки ЛИСП и

ПРОЛОГ, экспертные системы и др. В странах СНГ это: алгоритм КОРА Бонгарда М.М., язык РЕФАЛ – в МГУ, ситуационное управление – под руководством Поспелова Д.А.

Тема 2 Классификация интеллектуальных информационных систем

2.1 Признаки интеллектуальности информационных систем
2.2 Системы с интеллектуальным интерфейсом
2.3 Самообучающиеся системы
2.4 Систематизация самообучающихся систем

2.1 Признаки интеллектуальности информационных систем

Для интеллектуальных информационных систем, ориентированных на генерацию алгоритмов решения задач, характерны следующие признаки:

- развитые коммуникативные способности;
- умение решать сложные плохо формализуемые задачи;
- способность к самообучению.

Коммуникативные способности ИИС характеризуют способ взаимодействия (интерфейса) конечного пользователя с системой, в частности, возможность формулирования произвольного запроса в диалоге с ИИС на языке, максимально приближенном к естественному.

Сложные плохо формализуемые задачи – это задачи, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных данных и знаний.

Способность к самообучению – это возможность автоматического извлечения знаний для решения задач из накопленного опыта конкретных ситуаций.

В различных ИИС перечисленные признаки интеллектуальности развиты в неодинаковой степени и редко, когда все признаки реализуются одновременно. Условно каждому из признаков интеллектуальности соответствует свой класс ИИС:

- системы с интеллектуальным интерфейсом;
- экспертные системы;
- самообучающиеся системы.

2.2 Системы с интеллектуальным интерфейсом

Интеллектуальные базы данных отличаются от обычных баз данных возможностью выборки по запросу необходимой информации, которая может

явно не храниться, а выводиться из имеющейся в базе данных. Примером такого запроса может быть следующий: «Вывести список товаров, цена которых выше среднеотраслевой». Для выполнения запроса необходимо сначала проведение статистического расчета среднеотраслевой цены по всей базе данных, а уже после этого собственно отбор данных.

Во всех запросах такого типа требуется осуществить поиск по условию, которое должно быть доопределено в ходе решения задачи. Интеллектуальная система без помощи пользователя по структуре базы данных сама строит путь доступа к файлам данных. Формулирование запроса осуществляется в диалоге с пользователем, последовательность шагов которого выполняется в максимально удобной для пользователя форме. Запрос к базе данных может формулироваться и с помощью естественно-языкового интерфейса.

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний. Для этого необходимо решать задачи морфологического, синтаксического и семантического анализа и синтеза высказываний на естественном языке. Так, морфологический анализ предполагает распознавание и проверку правильности написания слов по словарям, синтаксический контроль – разложение входных сообщений на отдельные компоненты (определение структуры) с проверкой соответствия грамматическим правилам внутреннего представления знаний и выявления недостающих частей и, наконец, семантический анализ – установление смысловой правильности синтаксических конструкций. Синтез высказываний решает обратную задачу преобразования внутреннего представления информации в естественно-языковое.

Естественно-языковый интерфейс используется для:

- доступа к интеллектуальным базам данных;
- контекстного поиска документальной текстовой информации;
- голосового ввода команд в системах управления;
- машинного перевода с иностранных языков.

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы отличаются возможностью более сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Таким образом, механизм поиска работает, прежде всего, с базой знаний ключевых слов, а уже затем непосредственно с текстом. В более широком плане сказанное распространяется и на поиск мультимедийной информации, включающей помимо текстовой и цифровой информации графические, аудио и видео- образы.

Системы контекстной помощи можно рассматривать, как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В отличие от обычных систем помощи, навязывающих пользователю схему поиска требуемой информации, в системах контекстной помощи пользователь опи-

сывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам документации (например, технической документации по эксплуатации товаров).

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Графические образы в наглядном и интегрированном виде описывают множество параметров изучаемой ситуации. Например, состояние сложного управляемого объекта отображается в виде человеческого лица, на котором каждая черта отвечает за какой-либо параметр, а общее выражение лица дает интегрированную характеристику ситуации.

Системы когнитивной графики широко используются также в обучающих и тренажерных системах на основе использования принципов виртуальной реальности, когда графические образы моделируют ситуации, в которых обучаемому необходимо принимать решения и выполнять определенные действия. Одно из направления интеллектуального анализа данных (DM) – визуализация (когнитивная компьютерная графика). Средства визуализации могут выступить в качестве дополнительных инструментальных средств порождения зависимостей и как самостоятельный набор инструментов визуального анализа (помогает выдвигать нетривиальные гипотезы).

Интерактивная компьютерная графика **(**ИКГ) является важнейшим элементом современной информационной технологии и широко используется сегодня в системах автоматизированного проектирования сложных технических объектов; диагностики и управления технологическими процессами; автоматизации научных исследований, в обучающих системах и т. д. ИКГ выполняет две диалектически взаимосвязанные, но при определенных условиях различные функции: иллюстративную и когнитивную. Иллюстративная функция ИКГ, более традиционная и привычная, обеспечивает визуальную адекватность графического образа своему оригиналу, т. е. визуальную узнаваемость этого оригинала. Очевидно, что для такой «узнаваемости», как минимум, необходимо, чтобы этот оригинал существовал объективно, «вне и независимо от нашего сознания». Вторая, когнитивная, функция ИКГ позволяет при определенных условиях и требованиях изображать в наглядной графической форме внутреннее содержание, идею, суть изображаемого оригинала, которым, в частности, может служить любое абстрактное научное понятие, гипотеза или теория, которые, очевидно, уже невозможно «пощупать и потрогать руками». Хорошо известно также, что удачный рисунок иногда не только позволяет сделать наглядной и понятной суть сложного вопроса, но нередко способен подсказать принципиально новое соображение, идею, гипотезу, которые без такого рисунка просто, что называется, не приходят в голову. Именно этот важный аспект использования ИКГ для активизации образного мышления исследователей и конструкторов подчеркивается сегодня во многих работах.

2.3 Самообучающиеся системы

В основе самообучающихся систем лежат методы автоматической классификации примеров ситуаций реальной практики (обучения на примерах). Примеры реальных ситуаций накапливаются за некоторый исторический период и составляют *обучающую выборку*. Эти примеры описываются множеством признаков классификации. Причем обучающая выборка может быть:

1) *«с учителем»,* когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций (классообразующего признака);

2) *«без учителя»,* когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

В результате обучения системы автоматически строятся обобщенные правила или функции, определяющие принадлежность ситуаций классам, которыми обученная система пользуется при интерпретации новых возникающих ситуаций. Таким образом, автоматически формируется база знаний, используемая при решении задач классификации и прогнозирования. Эта база знаний периодически автоматически корректируется по мере накопления опыта реальных ситуаций, что позволяет сократить затраты на ее создание и обновление.

Общие недостатки, свойственные всем самообучающимся системам, заключаются в следующем: возможна неполнота и/или зашумленность (избыточность) обучающей выборки и, как следствие, относительная адекватность базы знаний возникающим проблемам; возникают проблемы, связанные с плохой смысловой ясностью зависимостей признаков и, как следствие, неспособность объяснения пользователям получаемых результатов; ограничения в размерности признакового пространства вызывают неглубокое описание проблемной области и узкую направленность применения.

2.4 Систематизация самообучающихся систем

Индуктивные системы. Обобщение примеров по принципу от частного к общему сводится к выявлению подмножеств примеров, относящихся к одним и тем же подклассам, и определению для них значимых признаков.

Процесс классификации примеров осуществляется следующим образом:

1) выбирается признак классификации из множества заданных (либо последовательно, либо по какому-либо правилу, например, в соответствии с максимальным числом получаемых подмножеств примеров);

2) по значению выбранного признака множество примеров разбивается на подмножества;

3) выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному подклассу;

4) если какое-то подмножество примеров принадлежит одному подклассу, т.е. у всех примеров подмножества совпадает значение классообразующего признака, то процесс классификации заканчивается (при этом остальные признаки классификации не рассматриваются);

5) для подмножеств примеров с несовпадающим значением классообразующего признака процесс классификации продолжается, начиная с пункта 1. (Каждое подмножество примеров становится классифицируемым множеством).

Примерами инструментальных средств, поддерживающих индуктивный вывод знаний, являются 1st Class (Programs in Motion), Rulemaster (Radian Corp.), ИЛИС (ArgusSoft), KAD (ИПС Переяславль-Залесский).

Системы, основанные на прецедентах (Case-based reasoning). В этих системах база знаний содержит описания не обобщенных ситуаций, а собственно сами ситуации или прецеденты. Тогда поиск решения проблемы сводится к поиску по аналогии (абдуктивному выводу от частного к частному):

1) получение подробной информации о текущей проблеме;

2) сопоставление полученной информации со значениями признаков прецедентов из базы знаний;

3) выбор прецедента из базы знаний, наиболее близкого к рассматриваемой проблеме;

4) в случае необходимости выполняется адаптация выбранного прецедента к текущей проблеме;

5) проверка корректности каждого полученного решения;

6) занесение детальной информации о полученном решении в базу знаний.

Так же как и для индуктивных систем, прецеденты описываются множеством признаков, по которым строятся индексы быстрого поиска. Но в отличие от индуктивных систем допускается нечеткий поиск с получением множества допустимых альтернатив, каждая из которых оценивается некоторым коэффициентом уверенности. Далее наиболее подходящие решения адаптируются по специальным алгоритмам к реальным ситуациям. Обучение системы сводится к запоминанию каждой новой обработанной ситуации с принятыми решениями в базе прецедентов.

Системы, основанные на прецедентах, применяются как системы распространения знаний с расширенными возможностями или как в системах контекстной помощи .

Информационные хранилища (Data Warehouse). В отличие от интеллектуальной базы данных информационное хранилище представляет собой хранилище извлеченной значимой информации из оперативной базы данных, которое предназначено для оперативного анализа данных (реализации OLAP

– технологии). Извлечение знаний из баз данных осуществляется регулярно, например, ежедневно.

Типичными задачами оперативного ситуационного анализа являются: определение профиля потребителей конкретного товара; предсказание изменений ситуации на рынке; анализ зависимостей признаков ситуаций (корреляционный анализ) и др.

Для извлечения значимой информации из баз данных используются специальные методы (Data Mining или Knowledge Discovery), основанные или на применении многомерных статистических таблиц, или индуктивных методов построения деревьев решений, или нейронных сетей. Формулирование запроса осуществляется в результате применения интеллектуального интерфейса, позволяющего в диалоге гибко определять значимые признаки анализа.

Применение информационных хранилищ на практике все в большей степени демонстрирует необходимость интеграции интеллектуальных и традиционных информационных технологий, комбинированное использование различных методов представления и вывода знаний, усложнение архитектуры информационных систем.

Тема 3 Опыт создания интеллектуализированного программного обеспечения по многомерному статистическому анализу

3.1 Экспертные системы

3.2 Проблемы и опыт создания интеллектуализированного программного обеспечения по многомерному статистическому анализу

3.3 Интеллектуальные возможности статистической экспертной системы и основные вопросы, возникающие при ее создании

3.4 Схема эволюции систем анализа данных и систем поддержки принятия решений

3.1 Экспертные системы

Экспертные системы используются во многих областях, среди которых лидирует сегмент приложений в бизнесе. Экспертная система является инструментом, усиливающим интеллектуальные способности эксперта, и может выполнять следующие роли: **консультанта** для неопытных или непрофессиональных пользователей; **ассистента** в связи с необходимостью анализа экспертом различных вариантов принятия решений; **партнера** эксперта по вопросам, относящимся к источникам знаний из смежных областей деятельности.

Архитектура экспертной системы включает в себя два основных компонента: базу знаний (хранилище единиц знаний) и программный инструмент доступа и обработки знаний, состоящий из механизмов вывода заключений (решения), приобретения знаний, объяснения получаемых результатов и интеллектуального интерфейса. Причем центральным компонентом экспертной системы является база знаний, которая выступает по отношению к другим компонентам как содержательная подсистема, составляющая основную ценность. «Know-how» базы знаний хорошей экспертной системы оценивается в *сотни тысяч долларов,* в то время как программный инструментарий – *в тысячи или десятки тысяч долларов.*

База знаний – это совокупность единиц знаний, которые представляют собой формализованное с помощью некоторого метода представления знаний отражение объектов проблемной области и их взаимосвязей, действий над объектами и, возможно, неопределенностей, с которыми эти действия осуществляются.

В качестве методов представления знаний чаще всего используются либо правила, либо объекты (фреймы), либо их комбинация.

Механизм объяснения. В процессе или по результатам решения задачи пользователь может запросить объяснение или обоснование хода решения. С этой целью ЭС должна предоставить соответствующий механизм объяснения. Объяснительные способности ЭС определяются возможностью механизма вывода запоминать путь решения задачи. Тогда на вопросы пользователя «Как?» и «Почему?» получено решение или запрошены те или иные данные система всегда может выдать цепочку рассуждений до требуемой контрольной точки, сопровождая выдачу объяснения заранее подготовленными комментариями. В случае отсутствия решения задач объяснение должно выдаваться пользователю автоматически. Полезно иметь возможность и гипотетического объяснения решения задачи, когда система отвечает на вопросы, что будет в том или ином случае.

Однако, не всегда пользователя может интересовать полный вывод решения, содержащий множество ненужных деталей. В этом случае система должна уметь выбирать из цепочки только ключевые моменты с учетом их важности и уровня знаний пользователя. Для этого в базе знаний необходимо поддерживать модель знаний и намерений пользователя. Если же пользователь продолжает не понимать полученный ответ, то система должна быть способна в диалоге на основе поддерживаемой модели проблемных знаний обучать пользователя тем или иным фрагментам знаний, т.е. раскрывать более подробно отдельные понятия и зависимости, если даже эти детали непосредственно в выводе не использовались.

Механизм приобретения знаний. База знаний отражает знания *экспертов* (специалистов) в данной проблемной области о действиях в различных ситуациях или процессах решения характерных задач. Выявлением подобных

знаний и последующим их представлением в базе знаний занимаются специалисты, называемые *инженерами знаний*. Для ввода знаний в базу и их последующего обновления ЭС должна обладать механизмом приобретения знаний. В простейшем случае это интеллектуальный редактор, который позволяет вводить единицы знаний в базу и проводить их синтаксический и семантический контроль, например, на непротиворечивость, в более сложных случаях извлекать знания путем специальных сценариев интервьюирования экспертов, или из вводимых примеров реальных ситуаций, как в случае индуктивного вывода, или из текстов, или из опыта работы самой интеллектуальной системы.

1.2 Проблемы и опыт создания интеллектуализированного программного обеспечения по многомерному статистическому анализу

Что такое «интеллектуализация программного обеспечения» и почему она нужна в прикладной статистике.

Как известно, конечной целью общей программы разработки ЭВМ *пятого поколения* является создание компьютеров, в которых будет реализован такой резкий скачок их интеллектуальных возможностей, в результате чего машина сможет непосредственно «понимать» задачу, поставленную перед ней непрофессиональным пользователем на естественном языке, т. е. с помощью речи, чертежей, схем, графиков и т.п.

В этой общей программе можно выделить четыре основных направления разработок:

1) развитие *элементной базы* (в частности, уже сегодня реально решение задачи достижения плотности «упаковки» порядка нескольких тысяч вентилей на одном кристалле);

2) разработка *новой архитектуры* (и в первую очередь архитектуры с многими параллельными потоками команд и обрабатываемых данных, предусматривающей, в частности, использование спецпроцессоров);

3) совершенствование *программной технологии* (и в частности, разработка языков высокого уровня для параллельной обработки данных);

4) *интеллектуализация,* т. е. оснащение ЭВМ системой решения задач и логического мышления, обеспечивающей способность машины к самообучению, ассоциативной обработке информации и получению логических выводов, что в конечном счете позволит резко повысить уровень «дружелюбия» машины по отношению к пользователю.

Именно в русле ключевых задач пятого направления лежат проблемы разного уровня интеллектуализации прикладного (проблемно- и методоориентированного) программного обеспечения (ППО). *Экспертные систе-*

мы принято относить к одной из основных форм высшего уровня интеллектуализации. Их создание связано в первую очередь с разработкой методов и средств формализации и ввода знаний в компьютерные системы (круг этих вопросов составляет содержание специальной дисциплины – так называемой «инженерии знаний») и манипулирования введенными знаниями.

Таким образом, проблематику, связанную с разработкой экспертных систем, можно отнести кругу ключевых вопросов решения общей программы создания ЭВМ пятого поколения. Однако следует подчеркнуть разницу в уровне дружелюбия, характеризующем экспертную систему и ЭВМ пятого поколения: услугами последней смогут пользоваться лица, не имеющие опыта работы с ЭВМ, в то время как для работы с экспертной системой все-таки должна быть определенная профессиональная подготовка.

В дополнение к сказанному необходимо остановиться на еще одном факторе, стимулирующем развитие работ в области создания именно *статистических экспертных систем* (СЭС).

Дело в том, что бурно возрастающие объемы информации, требующие грамотной статистической обработки, и почти столь же интенсивно растущее количество промышленного (и коммерчески распространяемого) статистического программного обеспечения (СПО). в основном в виде специализированных пакетов, находятся в явном дисбалансе с относительно медленно растущей численностью квалифицированных специалистов в области прикладной статистики. Это общая тенденция, но в странах СНГ она проявляется особенно остро.

В результате катастрофически нарастающее число лиц, не являющихся специалистами в области статистического анализа данных, использует СПО независимо от того, получили ли они одобрение специалистов по прикладной статистике и нужно ли это для успешного решения стоящих перед ним задач. Это в свою очередь является причиной развития опасного процесса роста доли неквалифицированного, порой безграмотно-спекулятивного использования СПО, что приводит к дискредитации аппарата прикладной статистики, наносит вред делу.

Распространение опыта специалистов по прикладной статистике в виде СЭС, нацеленных на подсказки и машинное ассистирование, в первую очередь в области предмодельного (разведочного) анализа данных, выбора подходящих моделей и нужной последовательности применяемых методов, интерпретации промежуточных и конечных результатов статистического анализа позволит в какой-то мере ослабить развитие упомянутого опасного процесса роста неквалифицированного использования СПО и смягчить причину этого процесса-дисбаланса между потребностью в квалифицированных специалистах по прикладной статистике и их фактическим наличием.

И наконец, *о социальном аспекте проблемы* создания СЭС. В этой связи следует упомянуть о наличии (в рядах специалистов по прикладной стати-

стике) определенной доли скептиков и даже явных противников, которые считают, что СЭС снижают потребность в знаниях живых специалистов, в какой-то мере заменяют и вытесняют их, выступают в качестве их конкурентов; следовательно, необходимо устраниться от участия в работах по созданию СЭС.

В действительности СЭС позволяет существенно повысить лишь средний, так сказать «ширпотребовский», уровень использования статистических методов анализа данных. Им в настоящее время обладает выросшая в последние десятилетия целая армия особого рода пользователей – «смежников», которые, как правило, «понемногу» ориентируются и в предметной области, в рамках которой решаются соответствующие статистические задачи (в экономике, социологии, медицине, геологии, технике и т.д.), и в инструментарии прикладной статистики, не являясь профессионалами ни там, ни здесь. Вот для этой армии работников кондиционные СЭС действительно представляют угрозу, так как при наличии хороших СЭС этих работников с пользой для дела целесообразно заменить специалистами-профессионалами соответствующих предметных областей.

Что касается профессионалов-статистиков, то создание и распространение СЭС лишь позволит высвободить часть их рабочего времени, отводимого для выполнения функций специалиста средней квалификации (в основном рутинного характера), и переключить его на решение задач более высокого профессионального уровня. Если к этому добавить продуманную систему экономического стимулирования работ профессионалов-статистиков в области создания СЭС, то их заинтересованность в развитии этих работ станет не только профессионально-органичной, но и активной.

1.3 Интеллектуальные возможности статистической экспертной системы и основные вопросы, возникающие при ее создании

Создатели большинства известных к настоящему времени статистических экспертных систем ставили перед собой задачу обеспечить пользователю СЭС машинное ассистирование по следующему кругу вопросов:

1) подсказки по существующим литературным, методическим и программным материалам, относящимся к специфике решаемой задачи;

2) советы в выработке адекватных исходных допущений о природе обрабатываемых данных и в выборе общего вида модели;

3) предложение «меню» подходящих методов статистической обработки с пояснением (в случае запроса пользователя) их сущности, особенностей, сфер применимости;

4) подсказки в построении технологической цепочки статистических процедур и алгоритмов, из которых должна состоять основная обрабатывающая (счетная) программа, и ее автоматическая реализация на ЭВМ;

5) помощь в проведении осмысления и интерпретации промежуточных и конечных результатов статистического анализа и (в случае необходимости) в выработке корректирующих управляющих команд к проведению дальнейшего статистического анализа;

6) помощь в выборе форм представления результатов проведенного статистического анализа.

Основной круг пользователей, на который рассчитаны подобные СЭС, это прикладные статистики и математики разного уровня квалификации, а также специалисты предметных областей (экономисты, социологи, медики, инженеры и т. д.), обладающие вероятностно-статистической подготовкой в объеме экономического или технического вуза.

В процессе создания СЭС разработчикам приходится последовательно анализировать вопросы (и уточнять их решение). На какого именно пользователя (предметная область, уровень квалификации) ориентирована создаваемая статистическая экспертная система, каковы конечные прикладные цели разработки и требования к уровню ее интеллектуализации? Какова структура функционального наполнения и сценария диалога СЭС? Какова главная концептуальная направленность (базовый методологический принцип) создаваемого машинного ассистирование (консультации в выборе и реализации используемых статистических методов, помощь в выборе стратегии статистического исследования и т.д.)? Какие именно технические средства целесообразно привлечь для реализации создаваемой СЭС? Какие типовые и оригинальные программные средства и алгоритмические языки необходимы для создания СЭС? Какие средства интеллектуального ассистирования и интерактивного режима необходимы для построения СЭС? В какой мере возможно использование существующих, а в какой – необходима разработка новых методов и средств формализации и ввода знаний в компьютерные системы, манипулирования введенными знаниями? Как проводить апостериорную оценку уровня интеллектуализации созданной СЭС?

3.4 Схема эволюции систем анализа данных и систем поддержки принятия решений

В числе основных характерных особенностей задач нового типа в компьютерном анализе данных можно назвать следующие:

1) объект исследования характеризуется **большими** объемами данных, требуется анализ в ограниченное время;

2) формальная модель объекта отсутствует (нет полного и непротиворечивого аналитического описания);

3) необходимо уметь выделять параметры, определяющие поведение (оптимизируемость и управляемость) в тех или иных ситуациях;

4) необходимо уметь обобщать имеющую информацию, выделяя неявно представленные зависимости (то есть те эмпирические правила, которые

позволяют оптимизировать и предсказывать поведение модели в новых обстоятельствах).

Особенностью новой парадигмы компьютерной обработки данных и знаний является использование: средств поддержки хранения больших пополняющихся объемов информации; развитых средств представления знаний и компьютерных моделей рассуждений; средств компьютерной аппроксимации психологических аспектов умственной деятельности (когнитивная графика и другие средства визуализации, формализация эвристических способов решения задач, формализация поиска релевантного знания в процессе рассуждения)э

К середине 90-х годов появилась технология Хранилищ информации (***Data WareHouse***) DWH и интеллектуального анализа данных (***Data Mining and Knowlelge Discovery in Databases***) DM: **DWH & DM.**

DWH – **Предметноориентированный и интегрированный** (объединяющий значения различных параметров), неизменяемый и поддерживающий хронологию НД, специфическим образом организованный для целей поддержки принятия решения (*Bill Inmon).*

DM – управляемый данными процесс (data driven) извлечения зависимостей из больших БД. В этом процессе центральное место занимает автоматическое порождение характеризующих анализируемые данные: моделей правил, функциональных зависимостей. Затем они предъявляются пользователю для оценки «интересности», релевантности и полезности для целей процесса Data Mining.

Схема эволюции систем анализа данных и систем поддержки принятия решений с учетом роста объемов данных, усложнения и интеллектуализации средств анализа данных (АД), ориентации на фактор РВ может быть охарактеризована следующими основными шагами.

1) **Технология баз данных (БД).** БД – специальная форма организации данных, поддерживаемая СУБД для поиска нужного значения параметра в системе формализованных отношений.

2) **Технология OLTP (*OnLine Transact Processing)*.** Стандарт промышленных СУБД, не способных быстро извлекать нужную информацию в режиме РВ был вытеснен с рынка информационных технологий.

3) **Технология OLAP (*OnLine Analysis Processing)*.** Усложнение средств АД в процессе принятия решений потребовало усовершенствований в технологиях накопления и обработки данных («расчеты по заранее заданным формулам»).

4) **Технология DWH & DM.** Наряду с задачами OLAP-обработки **поиск** всех релевантных **данным** и **целям** их обработки **функциональных зависимостей**. Характерна взаимная согласованность технологий накопления данных (представления данных и знаний, эффективного хранения, поиска и

доставки) и автоматического извлечения из них полезных зависимостей (моделей, правил, функциональных отношений).

Тема 4 Системы поддержки принятия решения и хранилища данных

4.1 Развитие систем поддержки принятия решения и хранилища данных

4.2 Принципы организации данных в системах поддержки принятия решения

4.3 Концепция хранилищ данных

4.4 Свойства хранилищ данных

4.1 Развитие систем поддержки принятия решения и хранилища данных

К середине 80-х годов 20 в. в развитых странах мира завершился первый этап оснащения бизнеса и органов государственного управления средствами вычислительной техники. Военные ведомства и крупные корпорации установили распределенные вычислительные системы, состоявшие из мощных мейнфреймов. С появлением персональных компьютеров ЭВМ стали доступны множеству средних фирм и организаций. Исторически эти системы в первую очередь реализовывали потребности в операционной обработке данных – они обслуживали информационные архивы, телефонные сети, системы резервирования билетов, сбора метеоданных и др. Использование мощных средств вычислительной техники позволило накапливать большие объемы информации: документы, сведения о банковских операциях, клиентах, предоставленных услугах. Однако период хранения этой информации был относительно невелик – сохранялись только данные за текущий календарный период.

Вскоре возникло понимание, что сбор данных – не самоцель и накопленные информационные массивы могут быть полезны. Системы операционной обработки способны выполнять тривиальный анализ данных – вычислять максимальные, минимальные и средние значения атрибутов и рассчитаны на быстрое обслуживание относительно простых запросов большого числа пользователей. Но из накопленных данных можно почерпнуть намного более глубокие сведения как о функционировании организации, которая обслуживается информационной системой, так и о сфере ее деятельности. В информационных массивах можно попытаться выявить скрытые, на первый взгляд, закономерности и вывести из них правила, которым подчиняется предметная область информационной системы. Впоследствии эти правила

можно использовать для стратегического планирования, принятия решений и прогнозирования их последствий.

Осознание пользы накапливаемой информации и возможности использовать ее для решения аналитических задач привело к появлению нового класса вычислительных систем – ***систем поддержки принятия решений (СППР)***, ориентированных на аналитическую обработку данных. Под *системой поддержки принятия решений* понимают человеко-машинный вычислительный комплекс, ориентированный на анализ данных и обеспечивающий получение информации, необходимой для разработки решений в сфере управления. Следует заметить, что аналитические системы существовали и ранее, но именно возможность обработки больших объемов накапливаемых данных дала новый толчок их развитию и приходу на рынок. Также этому способствовали снижение стоимости высокопроизводительных компьютеров и расходов на хранение больших объемов данных, развитие математических методов обработки информации. К числу задач, которые традиционно решают системы поддержки принятия решений, относятся: *оценка альтернатив решений, прогнозирование, классификация, кластеризация, выявление ассоциаций, моделирование процессов предметной области* и др.

Для получения интересующей их информации лица, принимающие решения (ЛПР), или аналитики обращаются к СППР с запросами. Эти запросы в большинстве случаев более сложные, чем те, которые применяются в системах операционной обработки данных. Например, в OLTP-системе банка запрос может сводиться к получению сведений о сумме на счету конкретного клиента. В аналитической системе запрос может быть таким: «Найти среднее значение промежутка времени между выставлением счета и оплатой его клиентом в текущем и прошедшем году отдельно для разных групп клиентов.»

В большинстве случаев сложный аналитический запрос невозможно сформулировать в терминах языка SQL, поэтому для получения информации приходится применять специализированные языки, ориентированные на аналитическую обработку данных. К их числу можно отнести, например, язык Express 4GL фирмы Orac1e. Также для выполнения аналитических запросов могут быть использованы приложения, написанные специально для решения тех или иных аналитических задач.

4.2 Принципы организации данных в СППР

Для того чтобы можно было извлекать полезную информацию из данных, СППР должны быть организованы особым, отличным от принятого в OLTP-системах образом. Связано это со следующими факторами. Во-первых, для выполнения аналитических запросов необходима обработка больших информационных массивов. Чем выше степень нормализации базы данных, и чем больше в ней таблиц, тем медленнее выполняется анализ. Происходит

это прежде всего потому, что увеличивается число операций соединения отношений. В системах обработки транзакций нормализация таблиц БД позволяет устранить избыточность данных, уменьшив тем самым объем действий, необходимых при обновлении информации. Поэтому в нормализованных БД нет необходимости менять одни и те же значения в различных отношениях. В аналитических системах данные практически не обновляются – в системе производится лишь их накопление и чтение. Поэтому проблема нормализации БД в них не столь актуальна, как в системах обработки транзакций. Во-вторых, выполнение некоторых аналитических запросов, например, анализ тенденций и прогнозирование, требует хронологической упорядоченности данных. Реляционная модель не предполагает существования порядка записей в таблице. В-третьих, данные, используемые для целей анализа, как правило, отличаются от данных систем обработки транзакций. При обслуживании аналитических запросов чаще используются не детальные, а обобщенные (агрегированные) данные. Так, например, для прогнозирования объема продаж сети универмагов будет излишним иметь информацию о каждой сделанной покупке, достаточно знать значение прогнозируемой величины за несколько предыдущих лет.

Принципы, лежащие в основе систем поддержки принятия решений, не позволяют эффективно обрабатывать транзакции, поэтому данные, применяемые для анализа, стали выделять в отдельные базы данных. Впоследствии эти базы данных стали называть ***хранилищами данных (ХД)*** или ***информационными хранилищами***. В литературе используется также англоязычный термин «Data Warehouse».

Отцом концепции использования хранилищ данных в аналитических системах считают Bill Inmon, технического директора компании Prism Solutions. В начале 90-х годов он опубликовал ряд работ, которые стали отправной точкой для последующих исследований в области аналитических систем. Большое влияние на разработку концепции хранилищ данных оказала также американская корпорация IBM.

4.3 Концепция хранилищ данных

Концепция хранилищ данных – это концепция подготовки данных для последующего анализа. Она предполагает выполнение следующих положений: интеграции и согласования данных из различных источников: традиционных систем операционной обработки данных, информации из внутренних и внешних по отношению к организации электронных архивов; разделения наборов данных, используемых системами обработки транзакций и системами поддержки принятия решений.

В работе «Создание хранилища данных» («Building the Data Warehouse») [4] Билл Инмон определил *хранилище данных* как «предметно-ориентированный, интегрированный, неизменяемый и поддерживающий

хронологию набор данных, предназначенный для обеспечения принятия управленческих решений». Позднее подробнее будут рассмотрены черты ХД, указанные Инмоном. А пока попытаемся уяснить схему функционирования СППР, основанной на концепции хранилища данных, проведя аналогии с процессами производства и реализации промышленной продукции.

Производство и реализация товаров имеют много общего с анализом данных: на предприятии из сырья получается готовая продукция, которая затем доставляется потребителю; в процессе анализа из накопленных данных добывается и предоставляется полезная специалистам информация, используемая для разработки решений.

Максимально упрощенно процесс производства и реализации промышленных товаров может быть описан следующим образом. Любая продукция, прежде чем быть доставленной потребителю, должна быть изготовлена. Этим занимаются заводы. Произведенная продукция отправляется на склад, откуда поступает в магазины. Именно там она находит своего потребителя.

Подобная схема обработки и снабжения справедлива и для аналитической системы. Исходные данные для анализа производятся системами операционной обработки, поступают из электронных архивов и от поставщиков информации, например, онлайновых информационных агентств. Эти источники слабо связаны между собой, поэтому и данные, которые они предоставляют, имеют различную структуру и форматы представления. Необходимо произвести согласование данных разных источников, чтобы ими было удобно оперировать при анализе. Это подразумевает приведение их к единому формату, устранение дублирующихся и некорректных значений.

Подготовленные данные загружаются в хранилище. Пользователи-аналитики осуществляют доступ к нему через клиентские приложения. Эти приложения могут осуществлять трансляцию запросов потребителей информации либо производить аналитическую обработку данных хранилища. В отличие от систем операционной обработки данных в СППР, использующих концепцию ХД, критерии поиска и состав выдаваемой в виде отчета информации не фиксируются при ее разработке, пользователи оперируют в основном заранее не регламентированными запросами (ad-hoc query).

Использование концепции хранилища данных в системе поддержки принятия решений преследует следующие цели: своевременное обеспечение аналитиков всей информацией, необходимой для выработки решений; создание единой модели данных организации; создание интегрированного источника данных, предоставляющего удобный доступ к разнородной информации и гарантирующего получение одинаковых ответов на одинаковые запросы из различных аналитических подсистем (единый «источник истины»).

Сейчас хранилища данных рассматриваются как «панацея», которая может обеспечить новое качество информационной системы. Рост интереса к

ним объясняется также и умелой рекламной политикой поставщиков аппаратно-программных решений основе этой концепции.

Вернемся к определению, данному Инмоном, чтобы подробнее рассмотреть свойства, присущие хранилищам данных.

4.4 Свойства хранилищ данных

Ориентация на предметную область. Хранилище должно разрабатываться с учетом специфики предметной области, а не приложений, оперирующих данными. Структура хранилища должна отражать представления аналитика об информации, с которой ему приходится работать. Например, если система операционной обработки поставщика товаров работает с понятиями «сделка» и «заявка», то хранилище должно использовать понятия «клиенты», «товары» и «производители».

Интегрированность. Информация загружается в хранилище из приложений, созданных разными разработчиками. Необходимо объединить данные этих приложений, приведя их к единому синтаксическому и семантическому виду. Например, в таблицах БД, полученных из разных источников, могут встречаться атрибуты, которые определены на разных доменах, но обозначают те же понятия. Например, месяц года может быть задан полным наименованием (январь, февраль и т.д.), сокращенным наименованием (янв, фев и т.д.) и номером (1,2 и т.д.). В процессе загрузки хранилища требуется преобразовать эти атрибуты к единому представлению. Важно также провести проверку поступающих данных на целостность и непротиворечивость. Характерный для информационных хранилищ прием – хранение агрегированных данных. Аналитика редко интересует информация о конкретных днях и часах, ему более важны данные о месяцах, кварталах и даже годах. Чтобы при выполнении аналитических запросов избежать выполнения операций группирования, данные должны обобщаться (агрегироваться) при загрузке хранилища. Объем накопленных данных должен быть достаточным для решения аналитических задач с требуемым качеством. Используемые в настоящее время ХД содержат информацию, накопленную за годы и даже десятилетия.

Неизменяемость данных. Важное отличие аналитических систем от систем операционной обработки данных состоит в том, что данные после загрузки в них остаются неизменными, внесения каких-либо изменений, кроме добавления записей, не предполагается. Именно поэтому для СППР не столь актуальны средства для обеспечения отката транзакций, борьбы с взаимными блокировками процессов – разработчики подобных систем сосредоточивают основные усилия на достижении высокой скорости доступа к данным. Важное условие неизменности информации в хранилище – использование для его реализации надежного оборудования, которое обеспечивает защиту от сбоев.

Поддержка хронологии. Для выполнения большинства аналитических запросов необходим анализ тенденций развития явлений или характера изменения значений переменных во времени. Учет хронологии достигается введением ключевых атрибутов типа «ДАТА» и/или «ВРЕМЯ» в структуры хранилища данных. Время выполнения аналитических запросов можно уменьшить, если физически упорядочить записи по времени, то есть расположить записи по возрастанию значений атрибута «ДАТА/ВРЕМЯ».

В последнее время сформировался новый класс систем поддержки принятия решений – ***системы оперативной аналитической обработки (OnLine Analysis Processing-OLAP)***. Под *OLAP-системой* принято понимать СППР, основанную на концепции хранилища данных и обеспечивающую малое время выполнения аналитических запросов.

К числу основных задач, которые требуется решать при создании ХД, относятся: выбор оптимальной структуры хранения данных с точки зрения обеспечения приемлемого времени отклика на аналитические запросы и требуемого объема памяти; первоначальное заполнение и последующее пополнение хранилища данными; обеспечение удобства доступа пользователей к данным. Рассмотрим пути решения этих задач более детально.

Тема 5 Методы аналитической обработки данных в хранилище

5.1 Средства интеллектуального анализа данных

Одна из важнейших составных частей современных аналитических систем – средства интеллектуального анализа данных. Методология и инструментальные средства интеллектуального анализа данных основаны, прежде всего, на методах машинного обучения (machine learning) систем интеллектуального АД (ИАД), способных: выявлять скрытые взаимные влияния различных факторов и вести причинный анализ (то есть давать ответы на вопросы ***«Почему?»***); порождать возможные зависимости в накопленных данных (причем не только заранее заданного вида, например, линейные функции); анализировать наблюдаемые в накопленных данных аномалии; прогнозировать (на основе порожденных зависимостей) характер поведения объекта исследования.

Выполнение большинства аналитических запросов пользователей требует сложной статистической обработки данных, применения методов искус-

ственного интеллекта. Современные СУБД, предназначенные для реализации аналитических систем, включают довольно обширный набор средств для статистической обработки информации. Однако задачи пользователей могут потребовать выполнения специфических операций над данными, поэтому средства анализа могут встраиваться также и в клиентские приложения.

В аналитических системах для обработки данных используется очень широкая номенклатура методов. Это и традиционные статистические методы регрессионного, факторного, дисперсионного анализа, анализа временных рядов, а также новые, получившие распространение в последнее время методы, основанные на искусственном интеллекте. К последним, как правило, относят: нейронные сети, нечеткую логику, генетические алгоритмы, методы извлечения знаний. В совокупности они именуются методами ***интеллектуального анализа данных (ИАД)***. Часто используется англоязычный термин «data mining» (дословно – добыча знаний). Эти методы развивают традиционные статистические подходы, находя применение там, где обычные приемы невозможно использовать в силу отсутствия точных зависимостей, описывающих анализируемые процессы. Технологии ИАД способны существенно расширить круг практически значимых задач, решаемых с использованием вычислительной техники.

В большинстве случаев средства анализа данных в СППР на основе ХД используются для решения следующих задач:

1) выделение в данных групп сходных по некоторым признакам записей (кластерный анализ);

2) нахождение и аппроксимация зависимостей, связывающих анализируемые параметры или события, а также поиск параметров, наиболее значимых в терминах конкретной задачи;

3) поиск данных, существенно отклоняющихся от выявленных закономерностей (анализ аномалий);

4) прогнозирование развития объектов различной природы на основе хранящейся ретроспективной информации об их состоянии в прошлом.

5.2 Методы систематизации данных

Кластерный анализ (также употребляются термины «кластеризация», «самообучение», «обучение без учителя») – это метод выделения из множества элементов групп (кластеров) схожих между собой элементов. Предполагается, что элементы одного и того же кластера похожи, а элементы различных кластеров отличаются друг от друга. Как правило, число кластеров заранее не определяется. Кластерный анализ записей баз данных осуществляется на основе значений их количественных и качественных атрибутов. При этом делается попытка автоматически разнести имеющиеся записи по различным группам. Кластерный анализ применяют при решении большого

числа задач. В социологии его используют для обработки результатов опросов общественного мнения, в медицине – для выявления типичных клинических случаев, в маркетинге – для поиска родственных групп клиентов. Часто выделение кластеров – отправная точка для других алгоритмов интеллектуального анализа данных. Применение этой процедуры позволяет во многих случаях перейти от обработки всего массива записей к анализу относительно небольшого числа кластеров.

5.3 Методы обнаружения сложных нелинейных зависимостей

Системы интеллектуального анализа данных эффективно используются для автоматического нахождения взаимосвязей и нелинейных зависимостей в данных. Учет подобных зависимостей позволяет лучше осмыслить предметную область, повысить качество решений, принимаемых на основе анализа ее состояния. В отличие от традиционных корреляционных методов, способных выявлять линейную взаимосвязь между переменными, методы ИАД обнаруживают и сложные нелинейные зависимости. Пакеты программ на их основе позволяют при обнаружении зависимостей определять их статистические характеристики, производить визуализацию области действия зависимости и выпадающих точек. Некоторые продукты интеллектуального анализа, например, система IDIS (The Information Discovery System) фирмы Intelligence Ware, способны выражать выявленные зависимости в виде правил на естественном языке. Современные средства ИАД позволяют также определять переменные, оказывающие наибольшее влияние на значение заданных атрибутов. Например, анализируя медицинские данные о больных, получивших травму, можно выполнить автоматический выбор трех атрибутов, наиболее значимых для определения времени восстановления больного после травмы. В качестве таких атрибутов, например, могут быть выделены: «время до момента оказания квалифицированной помощи», «возраст» и «физическое состояние больного». Впоследствии именно эти признаки могут быть использованы для многомерного визуального анализа, например, при определении координатных осей.

5.4 Методы обнаружения аномалий в данных

Другая важная задача, решаемая сейчас с помощью систем ИАД, – выявление разного рода аномалий в данных, отклонений от общей закономерности. Эта задача связана с предыдущей – отклонения обнаруживаются на основе выявленных ранее зависимостей. Происходит это следующим образом. Система, выявляющая аномалии, обучается на множестве допустимых записей, формируя их «собирательный образ». Если запись, предъявляемая впоследствии обученной системе, не удовлетворяет этому образу, система обращает на это внимание пользователя. Наиболее известный практический пример применения подобного подхода – система обнаружения случаев

мошенничества с кредитными картами Federal Express, разработанная фирмой HNC. Она формирует портрет операций, которые совершает владелец карты, и, если структура его расходов резко меняется, подает сигнал тревоги и блокирует выплаты. Такое изменение чаще всего происходит в случае, если карта похищена и преступник хочет воспользоваться ею до того, как о пропаже заявит владелец.

Под прогнозированием, как правило, понимают процесс формирования вероятностного суждения о состоянии какого-либо объекта, процесса или явления в определенный момент времени в будущем. Методы прогнозирования основаны на принципе инерционности развития, то есть предполагается, что развитие объекта подчинено определенным закономерностям, которые сохранятся на некоторый период в будущем. При прогнозировании используется способность методов ИАД выявлять закономерности в исторических данных, описывающих развитие объекта, и использовать в дальнейшем эти тенденции для выработки гипотез о его состоянии в будущем. Особенно широко для предсказаний методы ИАД применяют в финансовой сфере при прогнозировании доходности акций, курсов валют, экономических индикаторов.

Компьютерные аналитические технологии данных переживают этап бурного развития – появляются новые программные продукты и задачи, которые успешно решаются с их помощью. Однако даже самые лучшие программные средства не заменят специалиста, способного провести интегральный анализ наблюдаемых явлений. Тем не менее, современные интеллектуальные компьютерные технологии могут быть хорошим помощником аналитика, в значительной мере упрощая ему работу.

Тема 6 Модели данных, используемые для построения хранилищ

6.1 Модели данных, используемые для построения хранилищ

Задачи, решаемые OLTP и аналитическими системами, существенно различаются, поэтому их БД тоже построены на разных принципах. Критерием эффективности для систем операционной обработки данных служит число транзакций, которое они способны выполнить в единицу времени. Для аналитических систем важнее скорость выполнения сложных запросов и про-

зрачность структуры хранения информации для пользователей. Важная особенность СППР на основе ХД состоит в том, что загрузка данных выполняется сравнительно редко, но большими порциями (до нескольких миллионов записей за один раз), поэтому в таких системах обычно не предусматриваются развитые средства обеспечения целостности, восстановления, устранения взаимных блокировок. Это не только существенно облегчает и упрощает сами средства реализации, но и значительно снижает внутренние накладные расходы при доступе к информации и, следовательно, повышает производительность анализа.

В настоящее время существуют два в чем-то конкурирующих, а в чем-то взаимодополняющих друг друга подхода к построению хранилищ данных: подход, основанный на использовании многомерной модели БД *(Multidimedsional OLAP-MOLAP)*, и подход, использующий реляционную модель БД (*Reiational OLAP-ROLAP).*

Прежде чем рассказать о каждом из них, попытаемся разобраться, какие данные могут находиться в хранилище, и как они могут быть представлены. Чаще всего там содержатся сведения о значении некоторых параметров, характеризующих предметную область в определенные моменты или за определенные промежутки времени. Пусть, например, требуется создать хранилище, накапливающее информацию об изменении социально-экономической обстановки в России. Эта обстановка характеризуется многими параметрами, в числе которых: объем промышленного производства, индекс потребительских цен и др. Госкомстат России собирает их значения для различных субъектов Российской Федерации помесячно, поквартально или за год.

В хранилище должны попадать факты вида:

***Название параметра* в *субъекте Российской Федерации* в *момент времени* был равен {значение}.**

Например, индекс потребительских цен в городе Москве в декабре 1996 года был равен 101 %. В рассматриваемом примере каждое значение связано с точкой в трехмерном пространстве *(N, S, T) с* измерениями: *N*-название параметра; *S*- субъект федерации; *T*- момент времени. Число возможных параметров, субъектов РФ, а также рассматриваемых моментов времени конечно, поэтому все возможные значения можно представить в виде гиперкуба (см. рисунок 6.1).

В этом гиперкубе каждое значение находится в строго определенной ячейке, что значительно упрощает обращение к ней.

Представленный пример, конечно, упрощен, но он позволяет понять, что такое многомерный взгляд на данные. В реальной задаче число измерений может быть больше трех. Представление данных в виде гиперкуба более наглядно, чем совокупность нормализованных таблиц, оно понятно не только администратору БД, но и рядовым сотрудникам. Это дает им дополнительные возможности построения аналитических запросов к системе, ис-

пользующей хранилище данных. Кроме того, использование многомерной модели данных позволяет резко уменьшить время поиска в ХД, обеспечивая выполнение аналитических запросов в реальном времени.

Гиперкуб может быть реализован в рамках реляционной модели или существовать как отдельная БД специальной многомерной структуры. В зависимости от этого и принято различать реляционный (ROLAP) и многомерный (MOLAP) подходы к построению ХД.

6.2 Многомерная модель хранилища

Многомерная модель БД появилась довольно давно, однако в силу присущих ей ограничений применение получила лишь в последнее время. При использовании этой модели данные хранятся не в виде плоских таблиц, как в реляционных БД, а в виде гиперкубов – упорядоченных многомерных массивов. То есть многомерное представление данных здесь реализуется физически. Конечно, такой подход требует большего объема памяти для хранения данных, при его использовании сложно модифицировать структуру данных. Например, добавление еще одного измерения приводит к необходимости полной перестройки гиперкуба. Однако многомерные СУБД обеспечивают более быстрый по сравнению с реляционными системами поиск и чтение данных, избавляют от необходимости многократно соединять таблицы. Среднее время ответа на сложный аналитический запрос при использовании многомерных СУБД обычно в 10-100 раз меньше, чем в случае реляционной СУБД с нормализованной структурой.

Основные понятия многомерной модели – ***измерение и значение*** (ячейка). *Измерение* – это множество, образующее одну из граней гиперкуба (аналог домена в реляционной модели). Измерения играют роль индексов, используемых для идентификации конкретных значений в ячейках гиперкуба. *Значения* – это подвергаемые анализу количественные или качественные данные, которые находятся в ячейках гиперкуба (см. рис. 6.1).

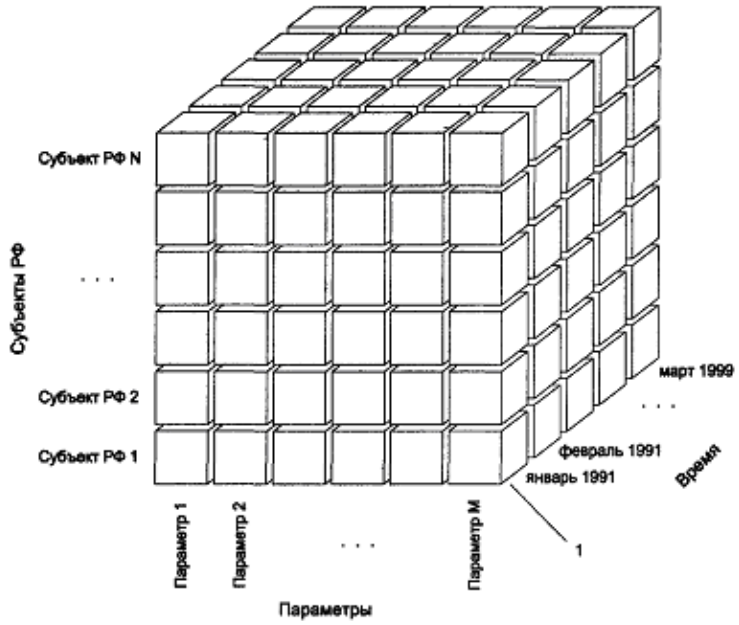


Рисунок 6.1 – Представление данных в виде гиперкуба;
1 – значение «Параметра М» для «Субъекта РФ 1 в январе 1991 года

В многомерной модели вводятся следующие основные операции манипулирования измерениями: 1) сечение; 2) вращение; 3) детализация; 4) свертка.

При выполнении операции сечения формируется подмножество гиперкуба, в котором значение одного или более измерений фиксировано. Например, если на рис. 6.1 зафиксировать значение измерения «Время» равным «январь 1991 года», то мы получим двухмерную таблицу с информацией о значениях всех параметров для всех субъектов РФ в январе 1991 года.

Операция вращения изменяет порядок представления измерений. Она обычно применяется к двухмерным таблицам, обеспечивая представление их в более удобной для восприятия форме. Если в исходной таблице по горизонтали были расположены субъекты РФ, а по вертикали параметры социально-экономической сферы, то после операции вращения параметры будут размещены по горизонтали, а названия субъектов РФ – по вертикали.

Для выполнения операций свертки и детализации должна существовать иерархия значений измерения, то есть некоторая подчиненность одних значений другим. Например, 12 месяцев образуют год, субъекты РФ образуют регионы. При выполнении операции свертки одно из значений измерения

заменяется значением более высокого уровня иерархии. Например, аналитик, узнав значения параметров для января 1991 года, желает получить их значения за весь 1991 год. Чтобы это сделать, необходимо выполнить операцию свертки. Операция детализации – это операция, обратная свертке. Она обеспечивает переход от обобщенных к детализированным данным.

Основное назначение СУБД, поддерживающих многомерную модель, – реализация систем, ориентированных на аналитическую обработку. Многомерные СУБД лучше других справляются с задачами выполнения сложных нерегламентированных запросов.

Однако у многомерных БД имеются серьезные недостатки, сдерживающие их применение. Многомерные СУБД неэффективно по сравнению с реляционными используют память. В многомерной СУБД заранее резервируется место для всех значений, даже если часть из них заведомо будет отсутствовать. Другой недостаток состоит в том, что выбор высокого уровня детализации при реализации гиперкуба может очень сильно увеличить размер многомерной БД. В силу этих, а также некоторых других причин доступные на рынке многомерные СУБД не в состоянии оперировать данными большого объема. Объем, доступный им для хранения, ограничен 10-20 гигабайт.

Целесообразно использовать многомерную модель, если объем БД невелик и гиперкуб использует стабильный во времени набор измерений.

6.3 Реляционная модель хранилища данных

Основой при построении хранилища данных может служить и традиционная реляционная модель данных. В этом случае гиперкуб эмулируется СУБД на логическом уровне. В отличие от многомерных реляционные СУБД способны хранить огромные объемы данных, однако они проигрывают по скорости выполнения аналитических запросов.

При использовании РСУБД для организации хранилища данные организуются специальным образом. Чаще всего используется так называемая радиальная схема. Другое ее название – «звезда» (star). В этой схеме используются два типа таблиц: таблица фактов (фактологическая таблица) и несколько справочных таблиц (таблицы измерений).

В таблице фактов обычно содержатся данные, наиболее интенсивно используемые для анализа. Если проводить аналогию с многомерной моделью, то запись фактологической таблицы соответствует ячейке гиперкуба. В справочной таблице перечислены возможные значения одного из измерений гиперкуба. Каждое измерение описывается своей собственной справочной таблицей. Фактологическая таблица индексируется по сложному ключу, скомпонованному из индивидуальных ключей справочных таблиц. Это обеспечивает связь справочных таблиц с фактологической по ключевым атрибутам. В качестве примера на рис. 6.2 приведена упрощенная схема струк-

туры хранилища данных, используемого для накопления информации из рассмотренного ранее примера (рис. 6.1).

В реальных системах количество строк в фактологической таблице может составлять десятки и сотни миллионов. Число справочных таблиц обычно не превышает двух десятков. Для увеличения производительности анализа в фактологической таблице могут храниться не только детализированные, но и предварительно вычисленные агрегированные данные.

Если БД включает большое число измерений, можно использовать схему «снежинка» (snowflake). В этой схеме атрибуты справочных таблиц могут быть детализированы в дополнительных справочных таблицах.

Для сокращения времени, требуемого для получения отклика от аналитической системы, можно использовать некоторые специальные средства. В состав мощных реляционных СУБД обычно входят оптимизаторы запросов. При создании хранилищ данных на основе РСУБД их наличие приобретает особую важность. Оптимизаторы анализируют запрос и определяют лучшую, с позиции некоторого критерия, последовательность операций обращения к БД для его выполнения. Например, может минимизироваться число физических обращений к дискам при выполнении запроса. Оптимизаторы запросов используют сложные алгоритмы статистической обработки, которые оперируют числом записей в таблицах, диапазонами ключей и т.д.

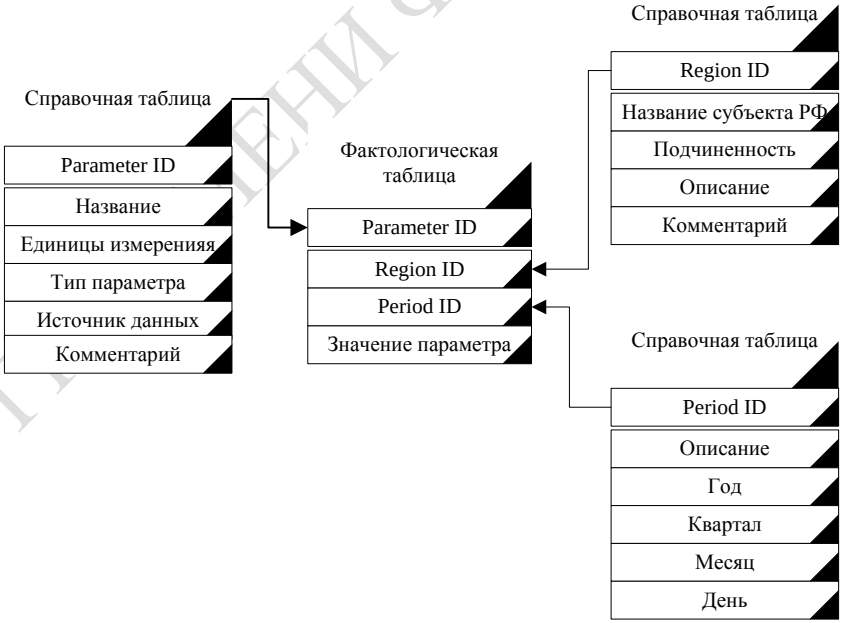


Рисунок 6.2 – Пример БД с радиально связанными таблицами (схема звезда)

1.4 Комбинация многомерного и реляционного подхода: киоски данных

Каждая из описанных моделей имеет как преимущества, так и недостатки. Многомерная модель позволяет производить быстрый анализ данных, но не позволяет хранить большие объемы информации. Реляционная модель, напротив, практически не имеет ограничений по объему накапливаемых данных, однако СУБД на ее основе не обеспечивают такой скорости выполнения аналитических запросов, как МСУБД. Нельзя ли совместить два этих подхода так, чтобы скрыть их недостатки и сделать более заметными их достоинства? Удачные проекты реализации хранилищ данных, появившиеся в последнее время, показывают, что это возможно.

Ситуация, когда для анализа необходима вся информация, находящаяся в хранилище, возникает довольно редко. Обычно каждый аналитик или аналитический отдел обслуживает одно из направлений деятельности организации, поэтому в первую очередь ему необходимы данные, характеризующие именно это направление. Реальный объем этих данных не превосходит ограничений, присущих многомерным СУБД. Возникает идея выделить данные, которые реально нужны конкретным аналитическим приложениям, в отдельный набор. Такой набор мог бы быть реализован в многомерной БД. Источником данных для него должно быть центральное хранилище организации.

Если проводить аналогии с производством и реализацией продукции, то многомерные БД выполняют роль мелких складов. В концепции ХД их принято именовать ***киосками данных (Data Marts)***. *Киоск данных* – это специализированное тематическое хранилище, обслуживающее одно их направлений деятельности организации. Логическая схема СППР, использующей центральное ХД организации и киоски данных аналитических отделов, позволяет эффективно использовать возможности реляционных СУБД по хранению огромных объемов информации и способность многомерных СУЕД обеспечивать высокую скорость выполнения аналитических запросов.

Доставка данных в хранилище

Данные должны поступать в хранилище в нужном формате и с требуемой регулярностью. Как правило, составляется расписание пополнения хранилища, в соответствии с которым специальные программы организуют передачу данных на склад и их первичную обработку. Передача данных на склад может также осуществляться при возникновении заранее определенных внешних событий.

Процесс загрузки данных обычно подразумевает решение следующие задач: приведения данных к единому формату (унификация типов данных и их представления, исключение управляющих кодов); предобработки данных

(исключение дубликатов, устранение ошибочных значений, восстановление пропущенных значений); агрегирования данных (вычисление обобщенных статистических показателей).

Метаданные

Пользователь не сможет извлечь нужные ему данные из хранилища, если не будет знать, что там находится. Прежде чем сформулировать свой запрос к системе, аналитик должен понять, какая информация в ней имеется, насколько она актуальна, точна, а также сколько времени может занять ожидание ответа.

Для описания структур данных в БД хранилища используются метаданные. Метаданные – это высокоуровневые средства отражения информационной модели СППР. Для обеспечения удобства доступа пользователей к информации ХД метаданные должны содержать: описание структур данных хранилища, структур данных, импортируемых из разных источников, сведения о периодичности импортирования, методах загрузки и обобщения данных, средствах доступа и правилах представления информации, оценки приблизительных затрат времени на получение ответа на запрос.

Метаданные помещаются в так называемый «репозитарий метаданных» системы. Этот компонент поддерживается большинством современных СУБД, средств разработки приложений и администрирования. Существует стандарт обмена метаданными MDIS, обеспечивающий возможность интеграции средств разных производителей друг с другом.

Тема 7 Этапы трансформации данных и знаний при обработке на ЭВМ

7.1 Определение данных и знаний
7.2 Этапы трансформации данных и знаний при обработке на ЭВМ
7.3 Шкалирование и пространство с семантической метрикой
7.4 Активность знаний

7.1 Определение данных и знаний

При изучении интеллектуальных систем традиционно возникает вопрос – что же такое знания и чем они отличаются от обычных данных, десятилетиями обрабатываемых. Можно предложить несколько рабочих определений, в рамках которых это становится очевидным

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

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

данные как результат измерений и наблюдений;

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

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

данные в компьютере на языке описания данных;

базы данных на машинных носителях.

Знания связаны с данными, основываются на них, но представляют результат мыслительной деятельности человека, обобщают его опыт, полученный в ходе выполнения практической деятельности. Они получаются эмпирическим путем.

Знания – это выявленные закономерности предметной области (принципы, связи, законы), позволяющие решать задачи в этой области.

При обработке на ЭВМ знания трансформируются аналогично данным:

знания в памяти человека как результат мышления;

атериальные носители знаний (учебники, методические пособия);

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

знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы – см далее);

базы знаний.

Часто используются такие определения знаний: *знания –* это хорошо структурированные данные, или данные о данных, или метаданные.

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенсионала. *Интенсионал* понятия – это определение через понятие более высокого уровня абстракции с указанием специфических свойств. Этот способ определяет знания. Другой способ определяет понятие через перечисление понятий более низкого уровня иерархии или фактов, относящихся к определяемому. Этот – определяет данные, или *экстенсионал* понятия.

Например, интенсионал понятия «персональный компьютер»: «Персональный компьютер – это дружественная ЭВМ которую можно поставить на стол и купить менее чем да $2000 – 3000». Экстенсионал этого понятия: «Персональный компьютер – это Mac, IBM PC, Sinker…»

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). *База знаний –* основа любой интеллектуальной системы.

Знания могут быть классифицированы по следующим категориям:

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

глубинные – абстракции, аналогии, схемы, отображающие структуру и процессы в предметной области

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет адекватных моделей, позволяющих работать с глубинными знаниями.

Кроме того, знания можно разделить на *процедурные* и *декларативные.* Исторически первичными были процедурные знания, т.е. 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 сосредоточивалась в структурах данных (таблицы, списки, абстрактные типы данных), т.е. увеличивалась роль декларативных знаний.

Сегодня знания приобрели чисто декларативную форму: знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному и понятных неспециалистам. Существуют десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам: продукционные, семантические сети, фреймы, формальные логические модели.

7.2 Этапы трансформации данных и знаний при обработке на ЭВМ

Термин «инженерия знаний» (русский эквивалент английского термина knowledge engineering) используется для обозначения области теории искусственного интеллекта, которая занимается языками для представления знаний, методами пополнения знаний, процедурами проверки их корректности и непротиворечивости и, наконец, использованием знаний при решении различных задач и созданием практических систем для хранения и обработки знаний. Термин «знание» в области искусственного интеллекта употребляется весьма часто и в несколько различающихся между собой значениях. Если мы обратимся к двум отделенным друг от друга немалым числом лет энциклопедическим справочникам, то прочитаем следующие определения этого термина.

«Знание в объективном смысле то же, что истинное знание (познание), в субъективном – убеждение в его истине по реальным причинам. В первом отношении оно противополагается заблуждению, как неистинному знанию, во втором – вере, мнению. По источнику познания различаются эмпирическое (опытное) и рациональное (разумное) знания; по объему – реальное (действительное) и формальное (применимое к одним лишь отношениям). Смотря по тому, далее нуждается ли или не нуждается мышление в объективных основаниях, для уверенности в его истине, различают посредственное (дискурсивное) и непосредственное (интуитивное) знания, из которых первое, в свою очередь, разлагается на фактическое и аподиктическое, смотря по тому, состоит ли обоснование его в простом удостоверении (подтвер-

ждении фактами) или в логическом доказательстве (выводе из других истин). Аподиктическое разлагается на очевидное и аксиоматическое, смотря по тому, излишне ли его обоснование (потому что истина его очевидна), или оно невозможно (потому что истина его не имеет оснований)». (Большая энциклопедия. Под ред. С. Н. Южакова, СПБ. т. 9, 1902, с. 688).

«Знание – проверенный практикой результат познания действительности, верное ее отражение в сознании человека. Знания бывают житейскими, донаучными, художественными, научными (теоретическими и эмпирическими)». (Большая Советская Энциклопедия, 3-е издания, М.:Т.9, 1972, с.555).

Для формирования рабочего определения того, что понимается под знанием в интеллектуальных системах, подобные энциклопедические определения не очень содержательны. Поэтому выберем иной путь.

В теории и практике программирования используется термин «данные», в какой-то мере родственный термину «знания». В процессе эволюции программирования то, что понималось под термином «данные», видоизменялось и усложнялось. На начальном этапе под данными подразумевались двоичные слова фиксированной длины. Их длина определялась длиной машинного слова, принятого в той или иной ЭВМ. Затем, например, путем «склеивания» между собой нескольких машинных слов или выделения из машинного слова части входящих в него разрядов. Затем стало возможным считать данными упорядоченные совокупности машинных слов. Например, в виде прямоугольных матриц, списков различной структуры, документов заданной формы. В настоящее время в программировании начинают использоваться так называемые абстрактные типы данных, структуру которых можно задавать произвольно, описывая ее средствами программирования.

Рисунок 7.1 – Схема процесса превращения данных в знания.

Развитие работ в области искусственного интеллекта породило новые типы языков программирования (Лисп, Пролог, Плэнер и другие), работающих со сложно структурированными данными. Стали появляться специальные языки для описания данных со сложной структурой (ФРЛ, КРЛ и другие), в которых данные (их стали уже называть знаниями) организованы в специальные структуры, носящие особые названия: фреймы, семантические сети, сценарии, вычислительные модели и т.п. другими словами, происходит постепенное усложнение данных и превращение их в знания. Схематически этот процесс можно представить так, как показано на рисунке 7.1. Не существует никакой объективной границы для смены термина «данные» на термин «знания», но первые пять шагов, показанные на рис.7.1, для этого обязательны. О шестом шаге речь пойдет ниже. Охарактеризуем суть каждого шага.

1 Внутренняя интерпретация. В начальном периоде развития программирования программист сам распределял *физическую* память, заложенную в ЭВМ. Каждой информационной единице – данному – он отводил определенную ячейку памяти, которой соответствовал ее адрес в памяти, т.е. раз и навсегда приписанное ей имя. Имя переходило на ту информационную единицу, которую программист заносил в ячейку. Такое насильственное приписывание имени информационной единице приводило к тому, что только сам программист знал содержимое, скрывающееся под тем или иным именем. При необходимости он использовал это содержимое в программе, вызывая его по имени, причем ЭВМ не имела никакой информации о том, что именно хранится в ее памяти.

Переход от физических адресов к относительным, а позже к символьным, создал большие удобства для программиста. Он освобождался от забот по распределению реальной памяти в ЭВМ. Перед ним возникло поле виртуальной памяти, а задача соотнесения «ячеек» виртуальной памяти с реальными ячейками физической памяти перекладывалась «на плечи» ЭВМ. Автоматически распределяя физическую память, ЭВМ формировала таблицу соответствий, в которой относительным или символьным адресам соответствовали физические адреса, назначаемые машиной. Но и теперь ЭВМ не получала информации о том, что именно скрывается за введенными программистом именами, соответствующими *относительным* или символьным адресам.

ЭВМ по-прежнему «не знала», что находится в ячейках ее памяти. Имена информационных единиц не подвергались ее анализу и она не могла ответить ни на один запрос, который касался бы содержимого ее памяти, если программист не составил для ответа специальной программы.

Для иллюстрации перехода от такого состояния к новому качеству – появлению внутренней интерпретации – рассмотрим таблицу 7.1. В ней две

части. Одна располагается справа под двойной линией. Она представляет собой массив из информационных единиц, соответствующих классическим данным в программировании. В части, расположенной выше и слева от двойной линии, содержится информация, относящаяся к именам (их часто в подобных случаях называют атрибутами), приписанным строкам и столбцам таблицы. При такой трактовке «Лев» есть имя первой строки. Информационная единица, которой оно приписано, выглядит так (Где живет?, Саванна) (Чем питается?, Животными) (Является ли хищником?, Да). А «Чем питается?» есть имя второго столбца. Информационная единица, которой оно приписано это имя, выглядит так ((Лев, Животными) (Слон, Растительной пищей) (Щука, Рыбами) (Мышь домашняя, Всеядна)). Круглые скобки здесь служат для выделения содержания информационной единицы, а угловые скобки выделяют в ней самостоятельные части, называемые обычно слотами. Полная запись информационной единицы, таким образом имеет вид (Имя информационной единицы (Имя первого слота, Значение первого слота)…(Имя *t*-го слота, Значение *t* – го слота)).

В память ЭВМ подобные информационные единицы записываются целиком. Итак, в памяти хранится не только то, что составляло в нашем примере правую нижнюю часть таблицы, а вся таблица, в которой происходит размножение и повторения некоторой информации (например, каждая информация, содержащаяся в правой нижней части таблицы, повторяется в двух информационных единицах, а атрибуты многократно).

Таблица 7.1 – Пример внутренней интерпретации

Животное	Характеристики		
	Где живет?	Чем питается?	Является ли хищником?
Лев	Саванна	Животными	Да
Слон	Тропические леса	Растительной пищей	Нет
Щука	Реки и озера	Рыбами	Да
Мышь домашняя	В домах и около них	Всеядна	Нет

Дополнительных расход памяти оправдывается тем, что при такой записи ЭВМ получает возможность отвечать на запросы, касающиеся содержимого ее памяти. Ее можно, например, спросить: «Что ты знаешь о животном «Слон»?». В качестве ответа ЭВМ может выдать пользователю всю информацию, которая составляет информационную единицу с именем «Слон». А на запрос: «Чем питаются животные?» ЭВМ может дать ответ, используя информационную единицу, чье имя совпадает с тем, что содержится в запросе. Ответы на различные запросы требуют разработки специальных встроенных в ЭВМ средств для анализа запросов и поиска нужных инфор-

мационных единиц, а наличие их вместе с моделью организации множества информационных единиц в памяти ЭВМ приводит к тому, что возникает база данных – объект весьма распространенный в современных программных системах. Среди них выделяются многочисленные реляционные базы данных, для которых системы организации информационных единиц весьма похоже на ту, которую мы только что описали.

Таким образом, наличие внутренней интерпретации *позволяет ЭВМ отвечать на вопросы, качающиеся содержимого ее памяти*. И в настоящее время оно полностью реализовано в широко распространенных реляционных базах данных.

2 Наличие внутренней структуры связей. Первоначально данные обладали очень простой и застывшей внутренней структурой. Машинное слово состояло из определенного числа двоичных разрядов и такая двухуровневая структура «машинное слово – совокупность разрядов» долгое время определяла возможное разбиение информационных единиц на части. Затем, когда отдельные машинные слова стали объединяться в более сложные структуры (например в списки), появилась возможность работать с информационными единицами более богатой внутренней структуры. Внутренняя структура данных еще более обогатилась, когда информационная единица стала такой, какой мы только что описали ее. Будем считать, что в качестве значений слотов в этой структуре смогут выступать новые информационные единицы. В этом случае слоты будут как бы вкладываться друг в друга, напоминая игрушку «матрешка». Теперь информационная единица может задаваться в виде структуры, показанной на рисунке 7.2, где s – имя всей информационной единицы, где s_i^j означает имя слота j-го уровня вложенности с порядковым номером *i*, а l_i^j- значение слота с порядковым номером i в j-м уровне. Как видим, наша информационная единица задается так, что ее первый слот в качестве своего значения содержит k слотов второго уровня, а все остальные слоты первого уровня не разбиваются на более мелкие единицы. Иногда в представлении знаний вместо термина «слот j-го уровня» используют специальные названия для частей информационной единицы. Например, части слота первого уровня называют ячейками, а их части (слоты третьего уровня) фасетами.

s	s_1^1	s_1^2	l_1^2	.....	s_k^2	l_k^2	s_2^1	l_2^1	.....	s_m^1	l_m^1

Рисунок 7.2 – Структура фрейма

Между слотами разных уровней (точнее, между их именами) могут устанавливаться отношения различного типа. На рисунке 7.3 показан пример многоуровневой информационной единицы. Она имеет глубину вложения

слотов, равную четырем. Именем слота первого уровня служит слот «животные». На втором уровне два слота: «рыбы» и «млекопитающие». На 3-м уровне имена слотов – это «лев», «слон» и т. д. Остальные слоты вместе с их значениями являются слотами четвертого уровня.

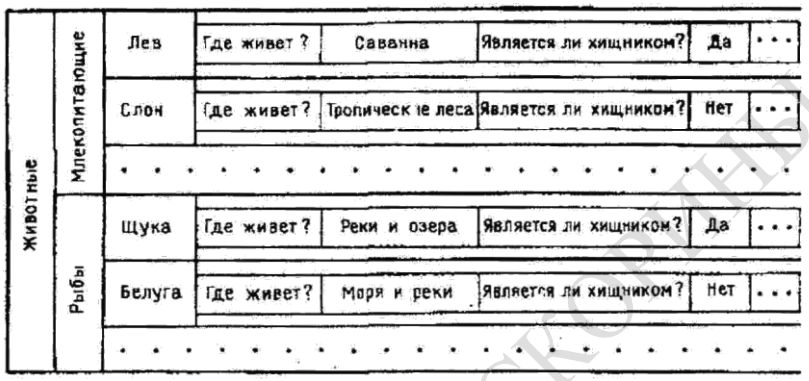


Рисунок 7.3 – Пример многоуровневой информационной единицы – фрейма

Информационные единицы, структурированные таким образом, обычно называются фреймами.

3 Наличие внешней структуры связей. Отдельные фреймы не могут представлять многие виды знаний, например, знания, описывающие динамические ситуации, когда отдельные факты и явления, содержащиеся в структуре одного фрейма, вступают в ситуативную связь с фактами или явлениями, описанными в структуре другого фрейма. Для создания таких связей используются специальные слоты. В них указываются имена фреймов, с которыми есть связь, и имена отношений, осуществляющих ее. Так возникает сеть с именами фреймов в вершинах. С помощью дуг, над которыми написаны имена соответствующих отношений, вершины сети соединяются между собой, образуя так называемую семантическую сеть.

На рисунке 7.4 приведен пример семантической сети. Отношения здесь интерпретируются следующим образом: r_1 – видеть, r_2 – существовать одновременно, r_3 – находиться в. Данной ситуации может соответствовать, например, фраза «В Московском зоопарке Петя видел льва и слона». Вершины сети могут соответствовать именам фреймов, имеющимся в системе, или слотам фреймов. Так вполне можно допустить существование в памяти фрейма с именем «Зоопарк», в котором описаны основные сведения, касающиеся зоопарков вообще.

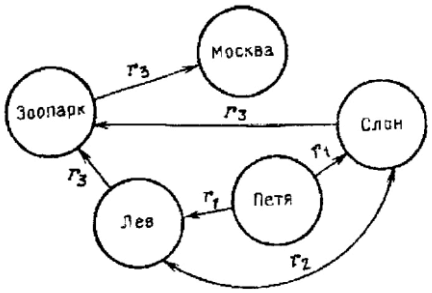


Рисунок 7.4 – Пример семантической сети

Другая возможность – хранение в памяти системы фрейма с именем «Московский зоопарк», в котором данные о зоопарках конкретизированы для конкретного зоопарка. Аналогичная ситуация может сложиться и с фреймами «Город» и «Москва». На рисунке 7.5 показана структура фреймов, которая может отражать всю совокупность сведений, необходимых для выделения всей информации, неявно связанной с информацией, содержавшейся в семантической сети, изображенной на рис. 7.4.

На рисунке 7.5 мы видим новые отношения, показанные штриховыми стрелками. Эти отношения можно трактовать двояко. Можно счисть, что они отражают тот факт, что «Москва» есть имя слота, входящего во фрейм «Город», а «Лев» – имя слота второго уровня во фрейме «Животное». Тогда отношения на рис. 7.4 между вершинами семантической сети суть отношения между соответствующими слотами, входящими в различные фреймы. Однако можно считать, что все вершины на рис. 7.5 характеризуют самостоятельные фреймы с такими именами. Тогда отношения выделенные штриховыми стрелками, приобретают смысл отношении быть элементом класса (для стрелок, направленных вверх) и содержать в себе элементы (для стрелок, направленных вниз). Такие классифицирующие отношения стандартно называют АКО-связь и ISA-связь (от английских a king of. . .; is a. . .):

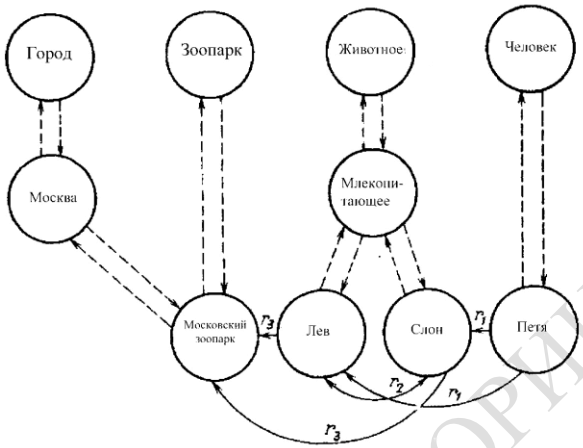


Рисунок 7.5 – Пример семантической сети с АКО- и ISA-связями

Классифицирующие связи позволяют всякую информацию, относящуюся к классу однородных понятий, записывать в памяти один раз, не дублируя ее во фреймах всех понятий, входящих в данный класс. Например, информация о том, что в состав городского хозяйства входят транспорт, связь, здравоохранение и т. п., можно иметь лишь во фрейме «Город», а во фрейме «Москва» записывать лишь то, что отличает Москву от других городов в отношении состава ее городского хозяйства.

То, что лев есть живородящее и теплокровное животное, вряд ли стоит записывать во фрейм с именем «Лев», ибо такая же информация будет нужна и для фрейма «Слон» и многих других фреймов, описывающих сведения о животных, относимых к млекопитающим.

Данную информацию достаточно записать один раз, использовав для этого слоты фрейма «Млекопитающие». При такой организации информации, используя сведения, например, из фрейма «Слон», можно пополнить ее перейдя по АКО-связям к фрейму «Млекопитающие», а затем с помощью тех же АКО-связей – к фрейму «Животное». Движение вверх по АКО-связям можно продолжать до тех пор, пока они существуют, или до получения достаточного количества информации.

Движение по АКО-связям может быть неоднозначным. На рис. 7.5, к примеру, от фрейма «Московский зоопарк» по АКО-связям можно двигаться как к фрейму «Москва», так и к фрейму «Зоопарк». Эти АКО-связи имеют различную семантику. При переходе к фрейму «Зоопарк» можно получить всю информацию, касающуюся зоопарков вообще, которая является «родовой» для Московского зоопарка. Другими словами, вся эта информация

непосредственно касается и Московского зоопарка. При переходе же к фрейму «Москва» мы получаем не «родовую» информацию, а информацию, связанную с Московским зоопарком только тем отношением, которое можно назвать ассоциативным. Итак, в разговоре о Московском зоопарке собеседники могут незаметно перейти от обсуждения нужд зоопарка к трудностям, связанным с их ликвидацией в Москве, а затем к проблемам, стоящим перед московским городским хозяйством. Другим примером подобной ассоциации может послужить ответ на вопрос «Какая среднегодовая температура в Московском зоопарке?». Вполне вероятно, что во фрейме «Московский зоопарк» нет слота с именем «Погода». Но такой слот может найтись во фрейме «Москва». Если он есть, то система будет в состоянии дать ответ на поступивший в нее запрос.

Использование ISA-связей возникает в двух случаях. Во-первых, они нужны для того, чтобы, уходя вверх по АКО-связям, найти путь назад к начальному фрейму. Во-вторых, ISA-связи используются при ответах системы на запросы типа «Какие животные содержатся в Московском зоопарке?» или «О каких городах система имеет информацию?»

В существующих сейчас реляционных, сетевых и иерархических базах данных с помощью тех или иных средств всегда реализованы механизмы, позволяющие перемещаться по классифицирующим связям типа АКО- и ISA-связей. Это означает, что у нас всегда есть возможность работать с иерархическими семантическими сетями.

7.3 Шкалирование и пространство с семантической метрикой

4 Шкалирование. В памяти человека сведения об окружающем его мире и возможных действиях в нем упорядочены не только классифицирующими и ситуативными отношениями. Для фиксации соотношений отдельных информационных единиц он использует различные шкалы. Простейшие из них – метрические шкалы. С их помощью можно устанавливать количественные соотношения и порядок тех или иных совокупностей информационных единиц.

На рисунке 7.6а) показана метрическая шкала, с помощью которой можно упорядочить по возрасту всех людей, известных системе.

Если, например, некий Петя в слоте «Возраст» характеризуется значением 16 лет, а некий Вася в своем слоте «Возраст» – значением 12 лет, то, имея шкалу, показанную на рис. 7.6а), можно ответить на вопросы типа «Кто старше – Петя или Вася?» или «На сколько лет Петя старше Васи?» Метрические шкалы такого типа могут относиться к самым различным характеристикам, если для них существует хотя бы одна единица измерения.

При наличии системы единиц с определенными количественными соотношениями получается множество метрических шкал, связанных между собой правилами переходов от одной шкалы к другой. Например, если бы Петя

и Вася были детьми грудного возраста, то шкала, в которой единицей измерения служит год (рисунке 7.6а), была бы для них малоинформативной. Здесь более подошла бы шкала с единицей, равной месяцу или неделе.

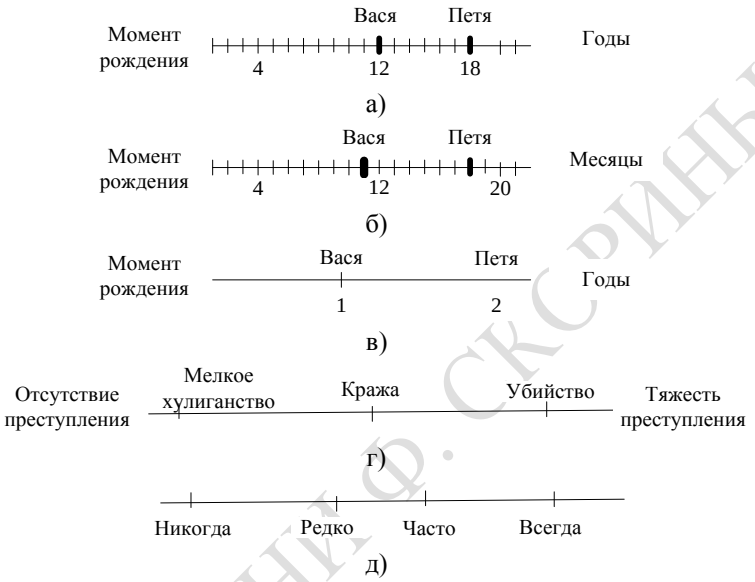


Рисунок 7.6 – Примеры шкал

На рисунке 7.6б) показана такая шкала, на которой Петя и Вася занимают некоторые положения, характеризующие их возраст. Заметим, что при переходе от этой шкалы к шкале с единицей, равной году, Петя и Вася переместились бы в позиции, отмеченные на рисунке 7.6в), причем некоторая информация о разнице их возрастов была бы утрачена. Если в первом случае она составляла семь месяцев, то теперь стала равной одному году. Обратный переход от шкалы с годовыми делениями к шкале с месячными делениями неоднозначен, если у нас нет какой-либо дополнительной информации о наших персонажах (например, что Петя родился в ноябре, а Вася в июне).

В качестве начала отсчета на шкалах, показанных на рисунке 7.6, выступает некоторое событие «Момент рождения». Оно абсолютно, так как не зависит от персонажей, характеристики которых мы проецируем на шкалу возраста. Столь же абсолютно начальное событие, от которого мы отсчитываем историческое время. Оно носит название «Рождество Христово». От него в обе стороны по шкале времени мы ведем отсчет солнечным годам. В

мусульманских странах таким событием является хиджра – переселение пророка Мухаммеда в 622 г. из Мекки в Медину. У других народов летоисчисление может основываться на иных великих в их жизни и истории событиях, идти от другого договоренного или абсолютного события. Метрические шкалы такого типа называются абсолютными метрическими шкалами.

Другой тип метрических шкал – относительные метрические шкалы. Начало отсчета на них меняется и каждый раз служит предметом специального договора. Очень часто это начало определяется текущим моментом высказывания или местом нахождения. Такие высказывания, как «Через два часа мы доедем до Москвы» или «Отсюда два километра до места моей работы», проецируются именно на относительные шкалы. Для относительных шкал используют такие же единицы измерений, как и для абсолютных.

Наконец, еще один вид шкал – порядковые шкалы. На них фиксируется лишь порядок информационных единиц. Примером такой шкалы может служить шкала тяжести преступлений, описываемая в юриспруденции. Все мы представляем себе, что кража – преступление менее значительное, чем убийство, а мелкое хулиганство не может находиться на шкале тяжести преступлений справа от кражи (см. рис.7.6г). Другим примером чисто порядковой шкалы может служить используемая при оценке успехов обучающихся в школе или институте шкала оценок: неудовлетворительно, удовлетворительно, хорошо, отлично. Во всех этих случаях на шкале нет никакой метрики, и мы не можем количественно оценить, насколько преступление кража тяжелее преступления мелкое хулиганство и насколько оценка «хорошо» превосходит оценку «удовлетворительно».

Среди порядковых шкал особый класс образуют размытые порядковые шкалы. Иногда их называют лингвистическими шкалами. Пример такой шкалы приведен на рисунке 7.6д). Метки на ней характеризуют частоту появления событий или процессов. В отличие от остальных шкал, показанных на рисунке 7.6, она ограничена с двух сторон маркерами Никогда и Всегда. В первом случае частота появления равна нулю, а во втором – единице. Этим маркерам соответствуют точечные деления на шкале. Остальным маркерам соответствуют не точечные деления, а некоторые интервалы, величина и расположение которых зависят от интерпретации словесных оценок, относящихся к шкале. Например, можно считать, что маркеру Часто соответствует случай, когда частота появления лежит в интервале от 0,7 до 0,8.

Происхождение размытых шкал тесно связано с наличием в естественных языках размытых квантификаторов, т. е. слов типа Много, Мало, Редко, Давно, Большой, Далеко, Близко и т. п. Если отобрать размытые квантификаторы, относящиеся к одному какому-либо параметру или признаку, то их можно упорядочить с помощью эксперимента с носителями языка. Для размытых квантификаторов. Относящихся к частоте появления событий или процессов, можно в результате такого эксперимента получить, например,

следующий упорядоченный список размытых квантификаторов: Никогда. Чрезвычайно редко. Очень редко. Редко, Редко, но не слишком. Не часто – не редко. Часто, но не слишком. Часто, Очень часто. Почти всегда. Всегда.

На рисунке 7.6д) показана шкала, в которой использована лишь часть этого списка. Но и приведенный нами в качестве примера список может еще более расширяться. Необходимость проведения экспериментов с людьми по упорядочиванию размытых квантификаторов объясняется тем, что мы не всегда ясно ощущаем их порядок. Например, размытые квантификаторы Нечасто и Нередко разными людьми будут помещены в разные места приведенной выше порядковой шкалы для характеристики «Частота появления событий и явлений».

После получения порядковой шкалы для размытых квантификаторов можно приступить ко второму этапу – построению размытой шкалы. Делать это можно различными способами. Один из наиболее простых – равномерно разделить отрезок между концевыми маркерами на столько отрезков, сколько промежуточных маркеров в порядковой шкале. В более сложных случаях можно опросить экспертов с целью определения границ отрезков, соответствующих каждому из промежуточных маркеров.

В связи с появлением теории нечетких множеств для построения отрезков, соответствующих маркерам размытой шкалы, стали использовать функции принадлежности, исследуемые в рамках этой теории. Такие функции интерпретируются как характеристические функции для размытых множеств. Размытое множество А обладает той особенностью по сравнению с классическими множествами, используемыми в математике, что о любом элементе а функция принадлежности дает не двузначный ответ, как характеристическая функция обычного множества (а принадлежит А или а не принадлежит А), а многозначный (а принадлежит А, а принадлежит А со степенью $0 \le \mu \le 1$, а не принадлежит А).

Функция принадлежности для множества А обычно обозначается символом $\mu_A(x)$. Ее значение, равное 0, соответствует утверждению, что данный элемент не принадлежит множеству А, а ее значение, равное 1, свидетельствует о безусловной его принадлежности множеству А.

Промежуточные значения $\mu_A(x)$ не следует трактовать в вероятностном смысле, так как степень принадлежности элемента к размытому множеству не обязана иметь статистическую природу.

Функции принадлежности можно строить на основании экспериментов с людьми. Например, на рисунке 7.7 показана функция принадлежности для понятия «Старый человек». Конечно, в графике функции не все участки бесспорны. Его начало вряд ли может вызвать какие-либо сомнения. Никто не отнесет к старым людям ребенка, которому 5 или 10 лет, а также подростка. Дальнейшее изменение значения функции принадлежности во многом опре-

деляется теми реальными экспертами, которые высказывают свое мнение.

Например, для подростка человек, которому около 40 лет, как правило, воспринимается как «Старый человек». С другой стороны, в тех человеческих сообществах, где люди редко доживают до 50 лет. сорокалетний член общества также должен считаться уже «Старым человеком». Но для сообществ, в которых средняя продолжительность жизни приближается к 55-60 годам, порог старости сдвигается в сторону увеличения значений x. Так возникает переходное значение от 0 к 1 в функции принадлежности на рис. 7.7. Однако «хвост» функции вновь представляется бесспорным. Практически все эксперты будут считать человека, достигшего возраста 65- 70 лет, относящимся к множеству «Старые люди».

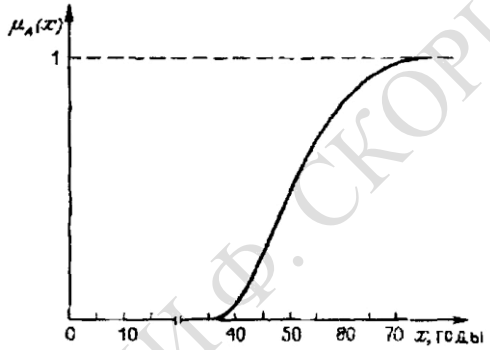


Рисунок 7.7 – Пример функции принадлежности

Из сказанного следует, что характер функции принадлежности во многом определяется некоторыми средними статистическими данными, некой нормой какого-либо явления, известной в данном социуме.

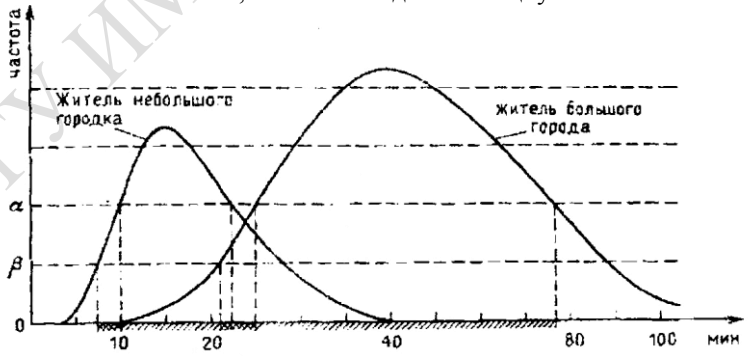


Рисунок 7.8 – Примеры функции принадлежности для двух городов

На рисунке 7.8 показаны два распределения, характерные для большого

города типа Москвы и небольшого городка типа Дубны. По оси абсцисс отложено время, затрачиваемое жителями этих городов при перемещении от дома до работы и обратно. Средства, используемые для передвижения, роли не играют, хотя влияют на количественные оценки,а качественная картина остается такой, как на рис. 7.8. Она показывает, что для жителя большого города характерна норма затрат времени на дорогу от дома на работу и обратно, близкая к 40 мин, а для жителя небольшого научного городка – близкая к 15 мин. Пересечение гистограмм горизонтальными линиями соответствует некоторым статистическим оценкам частоты появления событий, связанных с затратами времени на перемещение от дома до работы и обратно.

Выберем некоторую ординату, отвечающую частоте 0<a<1, и спроецируем на ось абсцисс соответствующие отрезки. На рис. 7.8 их проекции снабжены штриховкой. Время перемещения, характеризуемое ими, можно считать а-нормой затрат времени на перемещение между домом и работой для жителя небольшого городка и жителя большого города. В нашем случае обе а-нормы равны соответственно отрезкам времени [10 и 25 мин] и [27 и 79 мин]. Появление а-норм делает возможным говорить об отклонениях от них. Если а выбрано так, что для большинства жителей а-норма воспринимается просто как обычная норма, то становится возможным оценивать то или иное время, затрачиваемое на дорогу между домом и работой, как нормальное, большое или маленькое.

Если для оценки времени перемещения из дома на работу и обратно используется шкала из размытых квантификаторов типа Быстро, Нормально, Долго, Очень долго, то им будут соответствовать для жителя маленького городка и для жителя большого города различные отрезки на оси абсцисс. Для их нахождения надо проводить соответствующие уровни отсечки, подобные а-уровню, с помощью которого мы получили отрезки оси абсцисс, относящиеся к понятию нормы. На рис. 7.8 для примера показан уровень отсечки Р и соответствующие отрезки на оси абсцисс, которым в выбранной нами шкале размытых квантификаторов соответствует квантификатор Быстро.

Уровни отсечки должны подбираться таким образом, чтобы выделяемые ими отрезки на оси абсцисс не перекрывались и не оставляли пустых мест, которым не соответствует никакой квантификатор из выбранной размытой шкалы. Техника работы с такими объектами хорошо известна из теории нечетких множеств.

Отметим еще важное обстоятельство, а именно, что сами по себе нечеткие квантификаторы ничего конкретного не характеризуют. Они лишь фиксируют порядок, образующийся на размытой шкале. В самом деле, что значит Много? Во фразах «На концерт пришло много народу» и «В футбольном матче было забито много голов» вряд ли говорится об одинаковом количестве. Жить близко от работы, как следует из рис. 8.-совсем разные понятия для жителя небольшого городка и жителя большого города.

Такая неоднозначность затрудняет сравнение между собой информационных единиц, расположенных на различных размытых шкалах. Однако эта трудность преодолима. Она решается путем введения универсальных размытых шкал, на которые с помощью специальных процедур проецируются различные размытые шкалы, соответствующие одним и тем же спискам размытых квантификаторов.

Термин «шкала» обычно ассоциируется с одномерным образованием. Но шкалы могут иметь и больше измерений. Трехмерные системы координат иллюстрируют пример шкал такого типа.

Когда мы делили шкалы на абсолютные, относительные и размытые, то нас интересовали два параметра: наличие начала отсчета и та единица, которая либо измеряет шкалу, либо не измеряет ее (в таком случае шкала задает лишь порядок, как на размытых шкалах). Между выделенными нами типами шкал могут существовать и переходные, смешанные типы. Например, порядковая шкала с размытыми метками может быть соотнесена с четко заданным началом отсчета. Примеры такого типа демонстрируют высказывания: «Недалеко от Москвы лежит город Дмитров» или «В пять часов пятнадцать минут он сказал: «Ждать осталось недолго». Возможны и другие переходные типы шкал. Например. «Когда-то Иванов решил, что через два года он женится» или «За три километра до того, как это случилось, Петр уже почувствовал, что мотор барахлит».

Множества абсолютных и относительных метрических шкал и порядковых шкал, включая размытые шкалы, не исчерпывает всех шкал, используемых в когнитивных структурах, характерных для человека. Следующую группу применяемых нм для упорядочения информационных единиц шкал представляют оппозиционные шкалы. В истории развития человеческого мышления они сыграли выдающуюся роль. По-видимому, именно оппозиционные шкалы заложили основу восприятия мира человеком не как хаотического набора ситуаций и фактов, а как определенным образом упорядоченное их единство. Бинарная оппозиция, противопоставление двух фактов, явлений или свойств друг другу лежат в основе мировосприятия у всех народов. Инь и Ян у китайцев, левое и правое, мужское и женское, этот мир и тот (загробный) мир – все эти и многие другие примеры суть отражения бинарных оппозиций в сознании человека. И оппозиционные шкалы – одно из проявлений универсального способа постижения окружающего мира.

Оппозиционные шкалы образуются с помощью пар слов антонимов. Такие пары есть в каждом естественном языке. Их число для различных языков колеблется около 400, Примерами их могут служить Сильный-Слабый. Красивый-Безобразный. Хороший- Плохой, Острый-Тупой, Тяжелый-Легкий и т. п. В оппозиционных шкалах маркированы лишь концы шкалы. Середине ее, играющей роль начальной позиции, соответствует нейтральное значение, уравновешивающее значения маркеров на концах шкалы. Так. в середине

шкалы Хороший-Плохой находится позиция Не плохой-не хороший, а в середине шкалы Сильный-Слабый – позиция Не сильный-не слабый. Как правило, остальные позиции на оппозиционных шкалах никак не маркированы, либо эта маркированность выражается словосочетаниями типа Не очень плохой. Не слишком слабый и т. п.

Дальнейшее расширение оппозиционных шкал за границы слов антонимов может осуществляться за счет словосочетаний типа Очень сильный, Чрезвычайно острый, Беспредельно плохой. Несмотря на возможность расширения, оппозиционные шкалы ограничиваются замыкающими маркерами, как бы абсолютными значениями соответствующих качеств и свойств. Например, шкалы Добрый-Злой замыкают маркеры абсолютной доброты и зла, а шкалу Красивый-Безобразный – маркеры абсолютной красоты и безобразия.

На оппозиционные шкалы можно отображать самые разнообразные сведения об окружающем мире и нашем положении в нем. Только благодаря таким шкалам можно говорить, что «Петя красивее Васи», «Сильные люди, как правило, добры», «Острые предметы опасны». Два последних высказывания демонстрируют установление отношений между оппозиционными шкалами. Одна шкала оказывается зависимой от другой.

Постигая закономерности физического мира, отражая информацию о нем на перцептивных (связанных с восприятием) и когнитивных (связанных с понятиями) шкалах, человек затем использует те же шкалы и для отражения на них своего мира и тех понятий, которыми он оперирует. Можно выдвинуть гипотезу, согласно которой любое понятие человеческого языка как бы приобретает второй план, проецируясь на оппозиционные шкалы физического мира. Именно поэтому можно употреблять такие эпитеты, относящиеся к духовному облику человека, как низкий, тяжелый или широкий, характеризовать ум человека как острый или недалекий, приписывать состоянию горя эпитет тяжелое, а состоянию радости – беспредельная, легкая или светлая.

Существуют также оппозиционные шкалы, появление и использование которых однозначным образом обусловлено особенностями человеческой психики, формированием коллективного сознания и самосознания отдельной личности. Примером такой шкалы служит шкала Мы-Они. В процессе эволюции человека она претерпела несколько изменений. Предполагается, что подобная шкала возникла в глубокой древности, когда сосуществовали друг с другом как люди в современном понимании этого слова (т. е. представители вида Homo sapiens), гак и палеоантропы, стоящие на ступенях эволюции ниже современного человека. Эта шкала отражала глубокое различие между мы и они, т.е. между людьми и теми, кто хотя и похож на них, все-таки не человек в подлинном смысле слова. Шкала Мы – Они в тот период была эквивалентна шкале Люди- Не люди. Такое значение шкалы Мы-Они сохрани-

лось и до нашего времени. Когда мы говорим о каком-либо представителе рода человеческого, что он «сущий зверь», или характеризуем некий поступок словами «люди так не поступают», то используем оппозиционную шкалу Мы-Они в том первоначальном значении, которое дошло до нас из глубин человеческой истории.

Позже шкала Мы-Они стала играть несколько иную роль. Палеоантропы вымерли, землю заселили представители вида Homo sapiens, но они оказались разделенными на изолированные друг от друга сообщества. Уклад их жизни, обычаи, способ общения друг с другом и с природой стали различными. Сохранение единства, некоторой общности Мы требовало неукоснительного следования идее общественной имитации сложившихся форм жизни. Так возникали сложнейшие системы табу и ритуалов. Выполнение их обеспечивало принадлежность к Мы. И оно же отделяло Мы от других человеческих сообществ, которые заняли место они на шкале Мы- Они. Если они и люди, то живущие неправильно и плохие. Здесь кроются корни последующих человеческих столкновений на почве религии и различных укладов жизни.

5 Погружение в пространство с «семантической метрикой». Упорядочение сведений в когнитивных структурах человека происходит не только благодаря применению шкал, которых мы только что говорили. Уже давно выдвигалась гипотеза, что когнитивные структуры человека погружены в некое пространство, метрика которого характеризирует семантическую близость тех или иных понятий, фактов и явлений. Первую попытку построить такое пространство предпринял в конце 20-х годов американский психолог Ч. Осгуд. Для построения пространства, куда можно было бы погрузить человеческие знания, он воспользовался набором оппозиционных шкал. Осгуд, его ученики и многочисленные последователи собрали огромный статистический материал о способах размещения людьми слов из заданного им списка по набору оппозиционных шкал (число шкал колебалось около 400 при переходе от одного естественного языка к другому).

Оказалось, что эти размещения не случайны, а отражают некоторые характерные для той или иной социокультурной общности закономерности. Например, для всех народов европейской культуры слово «отец» на шкале Острый – Тупой смещается в сторону конца шкалы, маркированного словом «тупой», а слово «молоток» на шкале Сильный – Слабый явно тяготеет к левому концу шкалы. Полученный материал подвергался статистической обработке по методу главных факторов. По результатам обработки было построено трехмерное пространство Осгуда, в котором факторы-оси можно интерпретировать как шкалы **оценок, силы** и **активности.**

На ось оценок проецируются все оппозиционные шкалы с явно выраженным оценочным характером типа Хороший – Плохой, Добрый – Злой. Красивый – Безобразный и т. п. На ось силы проецируются такие оппозици-

онные шкалы, как Тяжелый-Легкий. Сильный-Слабый, Большой-Маленький и т. п. На ось активности проецируются шкалы типа Острый – Тупой, Быстрый – Медленный. Короткий – Длинный и т. п. Каждому слову-понятию в пространстве Осгуда соответствует некоторая точка.

Как же расположены слова-понятия в когнитивном пространстве Осгуда? Заполняют ли они его равномерно? Исследования показывают, что нет. Слова-понятия располагаются в этом пространства неравномерно. Точки, соответствующие им, образуют сгущения, называемые в теории распознавания образов и классификации кластерами. Расстояния между точками каждого кластера меньше расстояния до точек, входящих в другой кластер.

Можно выдвинуть гипотезу о том, что точки каждого кластера образуют совокупности понятий, семантически близких между собой. Такая гипотеза подтверждается экспериментом. В один кластер попадают, например, понятия – «отец», «мать», «дети», «сын» и т. п., т. е- понятия, объединенные семантически в ситуацию «семья». В другой кластер попадают понятия «дама», «роза», «поцелуй», «перчатка», «поединок», «рыцарь» .и т. п.. т. е. понятия, объединенные семантически в ситуацию «рыцарский турнир».

Это приводит к весьма важному для практики построения когнитивных структур в памяти интеллектуальных систем выводу, что близко группируется информация, относящаяся к некоторой типовой ситуации. С*итуационный принцип* формирует кластеры в пространстве Осгуда.

Для получения ситуативных кластеров не обязательно проводить ту процедуру, которую предложил Осгуд и которая до сих пор используется многими исследователями. Для корректного применения ее надо быть уверенным в выполнении ряда требований статистического характера, вытекающих из дисперсионного анализа. Поэтому сейчас многие специалисты предпочитают формировать кластеры на основании другой процедуры, называемой методом матриц сходства-различия.

В эксперименте испытуемые заполняют клетки матрицы-шахматки оценками попарного сходства и различия (в их качестве выступают обычно числа от +1 до – 1, взятые с некоторым шагом, например 0,1 при условии, что +1 характеризует полное сходство, а – 1 – полное различие) для заданного им списка слов-понятий. Число строк и столбцов матрицы равно числу слов-понятий в заданном списке. На основании усреднения таких матриц сходства-различия по всему контингенту испытуемых строятся средние оценки сходства-различия исходных слов-понятий, а затем по оценкам одного из методов кластеризации строятся кластеры и определяется степень принадлежности к ним исходных слов-понятий. Результаты кластеризации таким методом дают кластеры, похожие на построенные по методике Осгуда.

В пространстве Осгуда использовалась обычная евклидова метрика векторных пространств, а при построении кластеров различными способами по матрицам сходства-различия эта метрика может варьироваться, что приво-

дит к различным расстояниям между одинаковыми кластерами в пространстве Осгуда и в том «пространстве», которое получается при прямой обработке матриц сходства-различия.

Есть еще одна особенность пространства, в котором хранятся информационные единицы у человека. Каждая типовая ситуация, каждое слово-понятие имеют множество своих конкретных представителей. Выбор того или иного конкретного представителя из множества возможных подчиняется у человека закону *частоты появления*. Смысл его в том, что в качестве первого конкретного представителя типовой ситуации или слова-понятия всегда возникает тот, который в практике данного человека (его социума) встречался наиболее часто. Если, например, провести с представителями социума, в котором мы живем, эксперимент, в ходе которого испытуемый должен не задумываясь давать ответы на вопросы экспериментатора о названиях конкретных представителей типовых ситуаций или слов-понятий, то результат в подавляющем числе случаев будет демонстрировать действие закона частоты появления.

На требование назвать поэта, как правило, будет следовать ответ «Пушкин». Реже услышим ответы типа «Некрасов». «Лермонтов», «Блок». На требование назвать фрукт, как правило, будет получен ответ «яблоко». Реже прозвучат ответы типа «груша», «апельсин». А на требование назвать инструмент, как правило, получим ответ «молоток». Остальные инструменты будут называться реже. Такие эксперименты показывают, что у нас «на языке» готовые ответы на возможные запросы к нашей памяти всегда те, которые соответствуют наиболее часто встречавшемуся верному ответу. Подобный механизм позволяет при дефиците времени на обдумывание и поиск нужной информации в памяти всегда иметь наготове наиболее часто выручавший нас в прошлом отклик.

Итак, в семантическом пространстве человека существуют, по крайней мере, две системы оценки близости информационных единиц. Одна опирается на их ситуативную близость, а вторая – на частоту появления тех или иных ситуаций или конкретных представителей в типовых ситуациях.

7.4 Активность знаний

6 Наличие активности. Шестым свойством знаний является наличие активной. С самого начала своего развития программирование для ЭВМ пошло по пути разделения программы и данных, с которыми эта программа работает. В программе было сосредоточено процедурное знание. Оно хранило в себе информацию о том, как надо действовать, чтобы получить нужный результат. Знания в нужный момент использовались программой. Таким образом, программа играла роль активатора данных. Знания другого типа, которые обычно называют декларативными, хранили в себе информацию о том, над чем надо выполнять эти действия. Процедурное знание формирова-

ло обращение к декларативному знанию, воплощенному на первом этапе развития программирования в пассивно, лежащие в памяти ЭВМ данные. Как показывают исследования когнитивных структур человека, для него складывается как бы противоположная ситуация. Не процедурные знания активизируют декларативные, а наоборот – та или иная структура декларативных знаний оказывается активатором для процедурных знаний. Поясним эту важную для нас мысль примером. Часть семантической сети, в которой одна из вершин, маркированная именем Я, соответствует представлению субъекта о самом себе; остальные вершины семантической сети связаны с вершиной Я отношениями типа нравится (эти отношения показаны сплошными дугами) и не нравится (им соответствуют штриховые дуги). Например, в наблюдаемый нами момент Я нравится некая Она и некая конкретная Собака. Но Она к данной конкретной собаке или к собакам вообще относится отрицательно. Тогда простейший когнитивный треугольник таит в себе внутреннюю напряженность, ибо Я чувствует себя в сложившейся ситуации не очень уютно. Он не хочет, чтобы Она ушла из его жизни, но и не может с легким сердцем отказаться от своего преданного четвероногого друга. В теории когнитивных структур подобная ситуация относится к классу диссонирующих, требующих от субъекта принятия каких-то мер по ее ликвидации. Что же можно сделать? Покажем один из возможных путей снятия диссонанса. Любовь, которую вызвала у субъекта Она, оказалась столь сильной, что он ни в чем не может перечить любимому существу. И, найдя для себя множество оправданий, делающих вполне возможным шаг отказа от собаки. Я переходит на позицию человека, не держащего собак в доме, а возможно, и на позицию человека, не любящего собак. Когнитивный треугольник уже отличен от диссонирующего. Любовь торжествует, и оба не терпят собак в доме. Но может случиться так, что любовь к собаке в сердце Я окажется столь сильной, что нелюбовь, которую испытывает Она к животным, не останется бесследной. Субъект может припомнить и другие несовпадения вкусов, выявившихся между ними. Вспомнит, что Она эгоистична, спорит по пустякам, хочет всем командовать. Такой процесс может завершиться разрывом, и возникнет когнитивный треугольник. Он также свободен от диссонанса. Идиллия жизни с любимой собакой не нарушается никем.

Несколько более труден путь убеждения, с помощью которого субъект стремится сделать так, чтобы Она полюбила собак, поняла как они бескорыстны и добры. Для осуществления своей цели субъект может «подсовывать» будущей спутнице жизни книги о собаках, посещать с ней собачьи выставки, играть на ее самолюбии («А наша собака, смотри-ка, лучше всех!»). Возможна, наконец, ситуация, в которой субъект ничего не сможет сделать. У него нет сил отказаться от кого-либо из тех, кто ему нравится, и он не в силах убедить нравящуюся ему женщину в том, что собаки достойны любви не меньше, чем люди. В такой ситуации остается только одно – терпеть и

уповать на будущее. Это соответствует тому, что диссонирующий треугольник сохранится. Просто внимание будет сосредоточено не на нем. он будет сдвинут в сторону, в пределе-»убран в подсознание».

Несмотря на шутливость нашего примера, в нем достаточно точно отражены основные особенности того, что можно назвать активностью информационных единиц. Та или иная структура, сложившаяся между единицами определенного вида, является источником вызова (или синтеза из готовых стандартных модулей) процедур, цель которых в желании изменить когнитивные структуры через воздействие на среду, окружающую систему. Подход такого типа к современным системам пока не реализован. Вместо этого используются смешанные представления, в которых декларативные и процедурные знания понимаются единообразно и могут активизировать друг друга. Для примера смешанного представления рассмотрим фрейм, у которого в качестве значения какого-то его слота выступает имя процедуры, подлежащей выполнению. Обращение к данному слоту, как следует из уже сказанного, может обусловливаться либо другими слотами данного фрейма, либо другими фреймами. Это позволяет формировать условия, необходимые для выполнения той или иной процедуры, а следовательно, сделать активаторами процедур декларативные знания, хранящиеся в определенных слотах фреймов, образующих семантическую сеть.

Тема 8 Универсальная схема целевого функционирования

8.1 Основные понятия
8.2 Универсальная схема целевого функционирования
8.3 Описание узла связывания компонентов функционирования
8.4 Концептуальная схема целевого функционирования Естественные фазы развития носителя предмета и «плода»

8.1 Основные понятия

Понятие информации. Информацию определяют как «… одно из универсальных свойств предметов, явлений, процессов, объективной действительности, заключающееся в способности воспринимать внутреннее состояние и воздействия окружающей среды, сохранять определённое время, результаты воздействия, преобразовывать полученные сведения и передавать результаты обработки другим предметам, явлениям, процессам и т.д.». В нормативно-правовой сфере под информацией обычно понимают «…сведения о лицах, предметах, фактах, событиях, явлениях и процессах, независимо от формы их представления…». Это определение очень важно, потому что оно разделяет такие понятия, как «сведения» и «информация».

Возможность и эффективность использования информации обуславливаются основными её потребительскими показателями качества, как: репрезентативность, содержательность, достаточность, доступность, актуальность, своевременность, точность, достоверность и устойчивость.

Остановимся кратко на содержании этих качеств.

Под репрезентативностью информации понимают, прежде всего, правильность её отбора и формирования в целях адекватного отражения свойств объекта, т.е. правильности концепции, на базе которой формируется информационный блок, и обоснованность отбора существенных элементов и связей отображаемого объекта. Чем сложнее объект, тем насыщеннее отображающие его аспектные модели: сущностная, функциональная, информационная, математическая. Для построения этих моделей и проводятся системные исследования. О нерепрезентативной информации в народе говорят, что «… в огороде бузина, а в Киеве – дядька».

Следующим показателем качества информации является её содержательность. Последняя отражает семантическую ёмкость информационного блока, равную отношению количества семантической информации в сообщении к объёму содержательных данных. То есть с увеличением содержательности информации растёт семантическая пропускная способность информационной системы, так как для получения одних и тех же сведений требуется преобразовать меньший объём данных. Другими словами, семантические качества информации – это «калорийность» информации. И вот здесь самое время поговорить об информационных символах. В ходе наших предыдущих рассуждений мы уже установили, что количество сведений не всегда увеличивает количество реальных знаний. Поэтому естественным путём семантического уплотнения информации, т.е. повышением её «калорийности», является регламентирование иерархии обобщения понятийного аппарата. Для наших инженеров-прагматиков приведём небольшое сравнение: ведь никого уже не удивляет, что импульс сложной формы на экране осциллографа можно однозначно представить совокупностью синусоид, отличающихся друг от друга по частоте и амплитуде. Это ряды Фурье, их изучают в каждом уважающем себя высшем учебном заведении, а сейчас, говорят, даже и в лицеях. Пределом уплотнения семантического показателя информации является «Символ» или «Слово». В своей монографии «Философские заметки об информации, информационных технологиях, информационном обществе и Гармонии Мира» В.Д. Шкилёв приводит веские доказательства существования объёмного символа информации, объединяющего философские символы западного и восточного учений. Таким символом является объёмная монада, построенная на анализе иерархии внутренних противоречий. При таком подходе информация изучается как дуальное противопоставление философских категорий «правда- ложь», «форма-содержание», «непрерывный-дискретный» и другие.

Достаточность (полнота) информации означает, что информационное сообщение содержит минимальный, но достаточный для принятия правильного решения набор показателей. Понятие полноты информации зависит от формы её подачи. Просто факт: если разбить голографическую пластину на куски и осветить лазером любой из них, то появится полный образ, который присутствовал на пластине в целом. Таково свойство волновой информации. Доступность информации обеспечивается выполнением соответствующих процедур её получения и преобразования. Противоречивость понятия доступности проявляется через анализ философских категорий «закрытая информация – открытая информация». Такое деление информации – объективная необходимость развития общества на любых его этапах: от диктаторских форм правления до высочайших форм демократии. Различие лишь в количестве закрытой информации. Актуальность информации определяется степенью сохранения её ценности для управления в момент использования и зависит от динамики изменений характеристик модели управления, энергоинформационного блока, а также от интервала времени, прошедшего с момента возникновения данной информации. Своевременность информации означает её поступление не позднее заранее назначенного момента времени, согласованного с временем решения поставленной задачи. Недаром говорят, что «…хороша ложка к обеду…». Точность (достоверность) информации определяется степенью близости получаемой информации к реальному состоянию объекта, процесса, явления и т.п. Устойчивость информации отражает её способность реагировать на изменения исходных данных без нарушения необходимой точности. Устойчивость информации, как и репрезентативность, обусловлена выбранной методикой её отбора и формирования.

Интерпретация информации с позиций биосемиотики. В биосемиотике1 под информацией понимают микросостояние системы, способное влиять на выбор траектории динамики этой системы в неустойчивых точках бифуркации (разветвления). Смысл информации имеет два аспекта: значение и ценность. Значение – это набор запретов и ограничений, накладываемых информацией на пути развития системы, а ценность – это роль информации в повышении надежности самосохранения и самовоспроизведения системы. Эволюция смысла идет по пути экспликации (выхода наружу) во времени и пространстве, усложнения его структуры и развития межсистемных коммуникаций. В прагматическом аспекте различают потенциальную и актуальную информацию. Потенциальная информация не воспринимается приемником и потому не функционирует в проявленном виде, а актуальная – это изначально существующая в приемнике или воспринятая им информация,

[1] Семиотика- общая теория исследования знаковых систем, каждому знаку приписывается значение. Существуют три уровня исследования знаковых систем: синтаксис, семантика и прагматика

которая управляет его деятельностью (в терминах предложенной выше формализации функционирования потенциальная информация есть не что иное, как носитель идеи функционирования или суперфункционирования), а актуальная информация – проявление этого носителя идеи в акте реализации функционирования (т.е. действий). Потенциальная информация существует вне времени и пространства. Условия функционирования актуальной информации формулируются в пяти пунктах:

– система, несущая актуальную информацию, должна обладать спонтанной активностью или энергией, расширяющей ее возможности (в нашей терминологии активность связана с энергетическим планом сознания функционирования),

– информационные признаки должны быть несоизмеримо малы по сравнению со структурными признаками (информация «действует» незаметно, не мешает),

– устойчивые траектории динамики системы должны иметь неустойчивые точки бифуркации (разветвления),

– информационные признаки должны влиять на выбор траектории динамики системы в точках бифуркации,

– точки бифуркации должны быть защищены от внешних помех.

Идеальный мир (т.е. сознание функционирования) исходно не является автономным, поскольку микросостояние системы может испытывать влияние макросостояния в любой момент времени, а не только в особых точках бифуркации. Однако, в ходе эволюции в идеальном мире начинают формироваться автономные области, т.е. отдельные подмножества микропризнаков обособляются от других как макро- так и микро-признаков. Динамику системы можно образно представить, как разветвляющиеся железнодорожные пути, причем информация переключает стрелки. Защищенность неустойчивых точек бифуркации от внешнего шума необходима для увеличения надежности функционирования информации. Информация должна действовать специфически и потому спектр воспринятых воздействий должен быть узким. Например, живые организмы настроены на восприятие вполне определенной генетической информации. Так, состояние цитоплазмы клеток определяет специфичность восприятия генетической информации. Роль генов в индивидуальном развитии организма поясняется с помощью представлений об эпигенетическом поле, под которым понимается векторное поле в фазовом пространстве состояний организма, определяющее его развитие. Устойчивые траектории в этом поле получили название креодов. Гены определяют выбор траектории движения в точках разветвления креодов. Они переключают пути развития организма, но сами пути существуют помимо них. Таким образом, наследование не сводится только к передаче генов от родителей к потомкам. Необходимо также передать потомству и систему, способную эту информацию воплотить в морфологических структурах. Такой

системой служит яйцеклетка. Поэтому наследование происходит на двух уровнях: генетическом и эпигенетическом, причем, они неотделимы друг от друга.

Понятие системы. Под системой в общем плане (следуя Ю.А.Шрейдеру и А.А.Шарову [12]), будем понимать целостное образование, природа которого связана с некоторым источником организации или организующей общностью, например, у живого существа, геологического объекта или товара на рынке. Интуитивно люди по поведению целостного образования пытаются уловить: стоит ли за ним источник организации и какова идея организации. Особенность активной системы состоит в том, что у нее источник организации является автономной структурной составляющей (называемой обычно сознанием или разумом), обеспечиваю щей управление и координацию всех жизнеобразующих процессов. Путаница в понимании понятия системы создается еще и тем, что системность всегда про является двояким образом: на внутреннем и внешнем уровнях (для смысловой ассоциации эти уровни можно назвать организменным и надорганизменно-классификационным).

На **внутреннем** (организменном) уровне целостное образование воспри нимается как нерасчлененное и в нем путем исследования можно выделить ряд естественных компонент, связанных с различными внутренними представлениями системы (в качестве ассоциации под ними подразумеваются функции жизнедеятельности организма). Все эти компоненты как «функциональные» подсистемы, взаимообусловливают друг друга и образуют пространственно-временное единство.

На **внешнем** (надорганизменно-классификационном) уровне целостность системы мыслится как возможность естественного объединения в классы заранее имеющихся автономных объектов (организмов) единой природы. В ассоциативном плане под классом объектов-организмов можно понимать сообщество или клеточную ткань, являющуюся составной частью подсистемы функционирования некоторой метасистемы. Причем, функционирование «внешней» подсистемы определенным образом связано с функционированием соответствующих «внутренних» подсистем объект-организма. В общем случае объекты внешнего проявления системы могут не обладать пространственной, времен ной или генетической близостью и образовывать классы с достаточно размытыми границами.

Таким образом, мы подошли к следующему более полному определению.

Под **системой в широком смысле** будем понимать целостное образование с двухуровневым проявлением, источник организации которого связан с *жизнедеятельностью* в виде согласованных *функционирований* компонент объекта образования на внутреннем уровне и классов объектов на внешнем.

Термины «жизнедеятельность» и «функционирование» в данном определении использованы уже не в качестве смысловой ассоциации, а в качестве обобщенных фундаментальных понятий, формальная сущность которых будет рассмотрена ниже.

Под **системой в узком смысле** понимается либо *внутренняя система*, либо *внешняя система*. **Внутренняя система** – это составляющая системы, связанная с ее внутренним проявлением. Носителем внутренней системы является целостный объект (организм). **Внешняя система** – это составляющая системы, связанная с ее

внешним проявлением. Ее носителем является естественный класс или естественная классификация объектов.

8.2 Универсальная схема целевого функционирования.

На практике любое исследование посвящено некоторому целевому функционированию, которое требуется усовершенствовать. На рисунках 8.1, 8.2, 8.3 приведена структурная схема целевого функционирования социальной системы (социума).

Целевое функционирование есть выделяемая исследователем логическая единица взаимодействия:

– активной составляющей исследуемых процессов – жизнедеятельности субъект-системы, являющейся трехуровневым образованием из жизнедеятельностей индивидов социума (работников трудового коллектива молочного производства), жизнедеятельности самого социума, а также жизнедеятельности метасоциума – вышестоящих управленческих, образовательных, информационных и других формирований районного и регионального масштаба;

– пассивной составляющей исследуемых процессов – жизнедеятельности объект-системы, представляющей собой фрагмент природной и социо-культурной среды (окружающей среды), который вовлекается активной субъект-системой в сферу своей жизнедеятельности;

– окружающей среды.

Отметим, что выделение активной и пассивной составляющих в этом представлении не носит абсолютный характер, а обусловлено заданием системы координат исследователя, являющегося здесь представителем субъект-системы. В действительности же, то есть в абсолютной (онтологической) системе координат, объект-система может быть как пассивной, так и активной. В последнем случае исследователю необходимо дополнительно моделировать систему координат объект-системы для понимания ее поведения. Например, когда доярка кормит или доит корову, она может обращаться с ней либо как с машиной, либо как с одушевленным существом, пытаясь «войти в ее положение», приласкать или подбодрить.

Структура жизнедеятельности субъект-системы представлена единой связкой из пяти деятельностей:

1) *познания* – интеллектуального освоения мира на уровне идей;

2) труда – моторно-преобразовательного овладения материалом действительности;

3) *эстетического освоения мира на уровне цельных образов, фрагментов своей действительности;*

4) *коммуникации (обмена)* – взаимодействия функционирований подсубъектов субъект-системы по поводу накопления и передачи друг другу информации и других жизненных ресурсов как в рамках одного уровня (например, между индивидами или между социумами), так и между уровнями (в частности, между коллективом и индивидом посредством руководителя коллектива);

5) *балансировки и перестройки ценностей* функционирований подсубъектов субъект-системы в рамках ее жизнедеятельности.

Единство связки данных пяти деятельностей состоит в том, что ни одна из них не может существовать (и даже мыслиться) изолированно от других (как видим, подобная конструкция принципиально отличается от любой теоретико-множественной

конструкции). Природа единства жизнедеятельности определяется тем, что каждая деятельность субъект-системы формируется сетью функционирований разных направлений (познавательного, трудового и т.д.), среди которых лидирует подсеть функционирований с направлением этой деятельности. То есть, функционирования с другими направлениями в сети данной деятельности являются вспомогательными.

Само целевое функционирование субъект-системы может быть любого направления. На практике чаще всего приходится иметь дело с трудовым направлением, как в случае с примером молочного производства. Прежде чем остановиться на описании структуры целевого функционирования, отметим ее независимость от уровня детальности интерпретации. Как бы мы ни дробили исследуемую сеть функционирований, разлагая некоторые из них на цепочки из более мелких функционирований, формальная структура для их описания будет неизменной. Искусство интерпретатора состоит в том, чтобы он, остановившись на некотором уровне обобщения целевого функционирования, интерпретировал его компоненты также с помощью понятий одного уровня. Например, при описании функционирования трудового коллектива хозяйства -мы не должны его характеристики совмещать с характеристиками трудового коллектива отдельной фермы.

Аналогично от этого же зависит реализация процедур целевого функционирования и контроль за выполнением работы и результатами труда.

Будем различать два вида фона:

– социально-культурный фон субъекта функционирования – это то, в чем формируется жизнедеятельность субъект-системы на всех трех ее уровнях (индивида, социума и метасоциума). В частности, это освоение знаний, трудовых навыков, эстетических норм, способов общения и ценностных ориентаций в семье, школе, населенном пункте и в целом в государстве;

– природный и социальный фон объекта функционирования – то, в чем сформировалась жизнедеятельность объект-системы. В нашем случае на жизнедеятельности хозяйства сказываются существующие природно-экологические условия в районе данной фермы, социально-демографические условия в населенном пункте, а также в целом экономическая ситуация в хозяйстве и регионе.

Заметим, что в практике экономических исследований социальный фон и вообще субъективная составляющая анализируемой проблемы часто игнорируются или учитываются в крайне ограниченном объеме. Между тем, как показывают результаты социологического опроса, и эстетическая, и ценностно-балансировочная, и коммуникативная деятельности существенно сказываются на здоровье и мотивированности труда. Несмотря на то, что эти деятельности (на рисунке 8.2) удалены от узла связывания компонентов трудового функционирования, их роль может быть решающей. Так, например, мотивами работы у одного члена коллектива фермы (или команды животноводческих специалистов) могут быть преимущественно интересы или ценности метасоциума, выраженные, в частности, в качественном и эффективном выращивании скота, у другого будут доминировать ценности коллективного содружества и неформального лидера бригады, склонного к вредным привычкам, наконец, у третьего – сугубо индивидуальные или семейные интересы.

8.3 Описание узла связывания компонентов функционирования

Формируется цель функционирования и определяется предмет функционирования в виде тройки:

ПредмЦелФ = (НоситПредмЦелФ, ВхСостПредмЦелФ, ВыхСостПредмЦелФ), где

ПредмЦелФ – предмет целевого функционирования (ЦелФ) – в широком смысле понимается как пространственно-временное явление в виде развивающегося «плода» функционирования, в узком смысле – только результат – созревший «плод»;

ВхСостПредмЦелФ – входное состояние функционирования – состояние необслуженного данным функционированием предмета ЦелФ (состояние оплодотворения или точка «старта»);

ВыхСостПредмЦелФ – результат или продукт функционирования, определяющий содержание цели (состояние отчуждения плода или «точка снятия созревшего плода);

НоситПредмЦелФ – носитель предмета – организм или цельная природная система, на базе которого происходит развитие «плода». Следует различать две составляющие носителя предмета:

Ян-носитель (ЯнНоситПредм) – мужское начало, обеспечивающее, прежде всего, информационно-ориентационную сторону развития (в частности, посредством генетического механизма);

Инь-носитель (ИньНоситПредм) – женское начало, являющееся лоном развития «плода», отвечающее в основном за поставку структурного материала для роста.

Выделяются средства вложения в предмет целевого функционирования (СредсВложПредмЦелФ) – сырье в виде периодически поступающих порций материала (вещественного, энергетического и иногда информационного характера), которое реализуется в узле связывания носителем предмета для развития «плода».

Выделяется носитель узла связывания компонентов функционирования (НоситУзлаСвязЦелФ) – рабочее место для осуществления преобразований ЦелФ.

Описывается инструмент целевого функционирования (ИнстрЦелФ) – орудие, механизм, устройство либо орган тела человека, с помощью которого субъект реализует управляющее воздействие в узле связывания компонентов функционирования.

Выделяются управляющие воздействия (УпрВоздЦелФ) – комплекс команд (управлений), вырабатываемых управляющей подсистемой субъект-системы (УпрПодССубъектС), на основании которых реализуются действия.

8.4 Концептуальная схема целевого функционирования

Описание управляющей подсистемы субъект-системы. Структура управляющей подсистемы субъект-системы (УпрПодССубъектС) формируется тремя планами рассмотрения (информационным, энергетическим и оценочно-отражательным), каждый из которых дополнительно представляется в потенциальной (психика и сознание) и актуальной (реализация управления) плоскостях. К информационному плану относятся знания, т.е. наличие у работника потенциальных идей, моделей, алгоритмов и норм, объясняющих ему процессы целевого функционирования и помогающих ориентироваться непосредственно при организации работы (т.е. на актуальном уровне). Обычно знания формируются в результате специального обучения, а также во время работы путем общения с более квалифицированным напарником или специалистом. Реальная организация работы зависит не только от знаний работника, но,

согласно схеме, также от: желаний (определяемых в свою очередь мотивами, эмоциями и волевым настроем); эстетических чувств соответствия стандартам имеющихся условий для работы (отражающихся дополнительно в эмоциях).

Аналогично от этого же зависит реализация процедур целевого функционирования и контроль за выполнением работы и результатами.

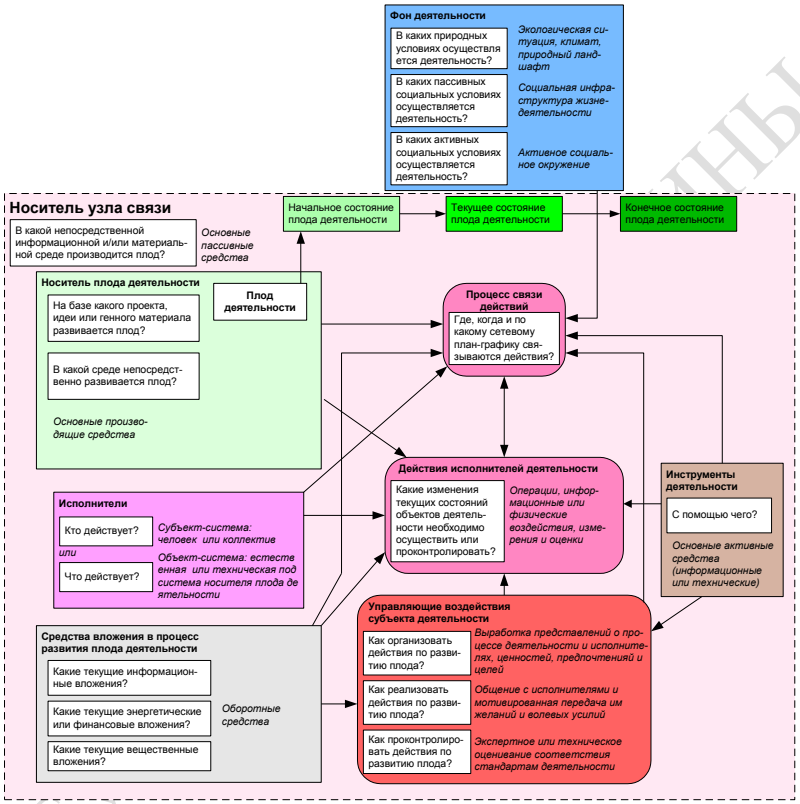


Рисунок 8.1 – Концептуальная схема целевого функционирования

Естественные фазы развития носителя предмета и «плода»

1 *Фаза проявления:*
   - *подготовка к оплодотворению ян и инь-компонент носителя предмета;*
   - *оплодотворение;*
   - *выход из зародышевой стадии.*

2 *Фаза роста.*

3 *Фаза созревания «плода»:*
   - *проявление нового плода;*
   - *рост нового «плода»;*

– *созревание нового плода.*

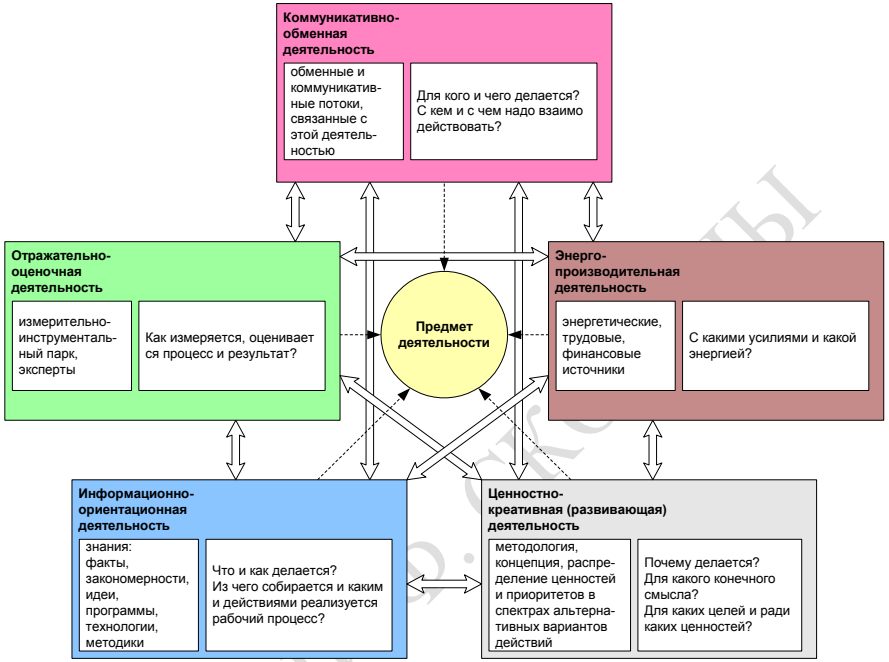


Рисунок 8.2 – Структура деятельности

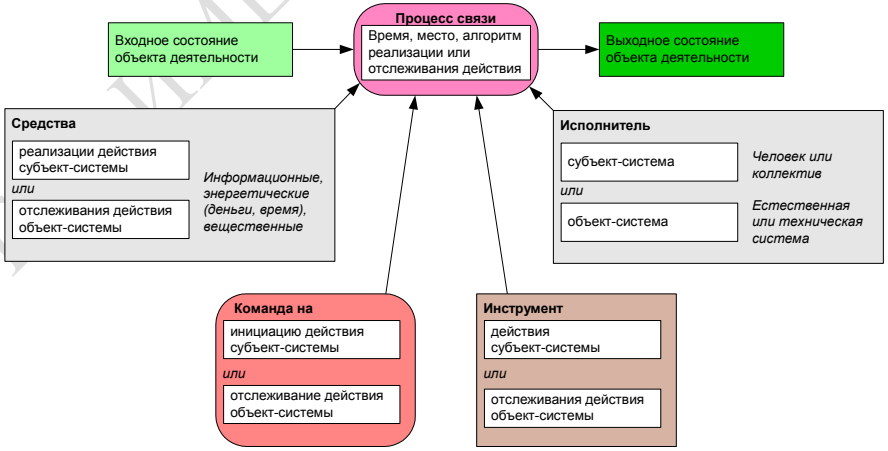


Рисунок 8.3 – Схема узла связывания

Раздел 2 Методы аналитической обработки данных

Тема 9 Моделирование человеческих рассуждений на основе индукции Джона Стюарта Милля

9.1 Индуктивное и дедуктивное рассуждения
9.2 Индукция Джона Стюарта Милля (ДСМ)
9.3 Рассуждения по аналогии
9.4 ДСМ-метод

9.1 Индуктивное и дедуктивное рассуждения

Аристотель упоминал о двух основных процессах рассуждений: нисходящем или дедуктивном и восходящем или индуктивном. Иногда говорят, что дедукция – это рассуждение «от общего к частному», а индукция- «от частного к общему». При таком понимании этих двух процессов возникает иллюзия, что они как будто обратны друг другу и одну схему рассуждений можно получить из другой прямым обращением. Этой иллюзии поддался и Аристотель. Увлеченный красотой и стройностью воздвигнутого им здания силлогистики, он попытался втиснуть в его объемы и индуктивное рассуждение, ввести схему индуктивного силлогизма. Но здесь его подстерегала неудача. Индуктивные рассуждения никак не хотели отливаться в ту стройную форму, которая так подошла дедуктивным рассуждениям.

Создавая силлогистику, Аристотель преследовал цель: сделать ее непобедимым оружием в споре. Если оппонент признавал общее положение, относящееся к классу однородных объектов или явлений (а как он мог не признать, например, столь очевидную ИСТИНУ, что «Все люди смертны»), и принадлежность какого-либо объекта или явления к этому классу (например, что «Сократ есть человек»), то ему ничего не оставалось сделать, как признать, что общий для всего класса признак переносится и на отдельный элемент этого класса. Возражать против такого хода рассуждений мог бы только человек, спорить с которым не имеет никакого смысла, ибо он отвергает очевидное.

Если бы аналогичная цель стояла перед спорящим, который пользуется методом индукции, то схема его рассуждений должна была быть следующей. Сначала он мог бы сообщить оппоненту несколько утверждений об отдельных представителях класса, в существование которого должны верить оба спорящих. Каждое такое утверждение должно касаться одного и того же признака, связанного с элементами этого класса (например, оппоненту надо было сообщить, что «Гомер смертен», «Фидий смертен», «Эзоп смертен», и добиться от оппонента признания правильности этих

утверждений). После этого надо было прийти с противником к согласию, что все эти элементы принадлежат одному классу (в нашем примере, что Гомер, Фидий и Эзоп являются людьми). Далее нужно было совершить главный индуктивный шаг, перейти к утверждению о классе (т е. ввести утверждение «Все люди смертны») и заставить противника принять это утверждение.

Трудность таится именно на последнем шаге спора. Примет или не примет этот шаг оппонент, зависит от степени его уверенности в правильности данного шага. Этот шаг требует не умения логически обосновывать свои действия и рассуждения, а веры в свою справедливость. Можно ли от трех конкретных утверждений о Гомере, Фидии и Эзопе перейти к общему утверждению обо всех людях? Ответ на этот вопрос не снимается, если мы к названным трем великим представителям греческой культуры добавим еще кого-нибудь. Где граница, после которой индуктивный шаг станет оправданным? Ответа на этот вопрос нет и быть не может. Именно поэтому *индуктивное умозаключение* всегда является *правдоподобным рассуждением.* Его надо принимать на веру. И обсуждать можно только то, как оценить обоснованность этой веры, т. е. как оценить степень правдоподобности выведенного утверждения.

Мы получили весьма важный вывод о том, что каждое правдоподобное утверждение *А* должно сопровождаться некоторой *оценкой правдоподобности (достоверности) Q(A).* Интерпретация *Q(A)* может быть различной. Некоторые из них, сейчас наиболее распространенные, будут обсуждены.

Подчеркнем еще раз принципиальное различие между дедуктивной и индуктивной схемами рассуждений. Если посылки в дедуктивной схеме выбраны правильно, являются истинными, то получаемые с их помощью заключения не могут быть ложными. Если они нас чем-то настораживают, вызывают недоумение, то надо еще раз проверить истинность посылок. Убедившись в их правоте, ничего не остается делать, как полностью принять следующие из них выводы. Если посылки в индуктивной схеме выбраны правильно, являются истинными, то получаемые с их помощью заключения могут быть как истинными, так и ложными. Та или иная точка зрения на заключения зависит от степени субъективной уверенности в достаточности посылок для получения заключения. Именно поэтому вместо оценки истинности или ложности заключения в правдоподобных рассуждениях используется оценка правдоподобности (или истинности) *Q(A).*

Известный специалист по психологии восприятия Р. Грегори писал: «В самой природе дедуктивных утверждений содержится нечто в высшей степени странное. Дедукция оперирует формальным символическим алфавитом. Вправе сказать, что дедукция небиологична, поскольку ее не

могло быть до появления формального языка. В связи с этим чрезвычайно заманчива мысль об индуктивной природе процесса решения проблем, который сопровождает работу воспринимающего мозга, и о переходе к *дедукции* в работе мозга, занятого абстрактным мышлением, передачей сообщений, выполнением расчетов. Если верно, то дедукция окажется свойственной только мозгу человека, поскольку лишь человек обладает формальной речью. Но это можно отнести также к электронным вычислительным машинам, работа которых подчинена правилам некоторого формального языка. По-видимому, можно утверждать, что – поскольку в отличие от владения формальной речью восприятие не является исключительной привилегией человека – *перцептивные процессы в своей сущности не дедуктивны. Остается принять, что они индуктивны.»*

Таким образом, индукция тесно связана с восприятием, опытом. Когда в развитии научного мировоззрения возник этап понимания, что опытные данные, эксперимент, реальная деятельность по достижении определенных целей служат единственным мерилом обоснования научных построений, тогда наступила пора индукции.

Понимание роли индуктивных рассуждений в научном познании связано с именем двух людей, носивших одинаковую фамилию. Один из них Роджер Бэкон был францисканским монахом и выдающимся мыслителем. С целью прославления церкви и воплощения своей мечты о том, что католическая церковь должна царить над всем миром, этот монах в 1265 году посвятил папе Клименту IV свою работу, где сделал набросок новой экспериментальной науки, которая должна была дать в руки человечества инструмент к познанию природы и роли высшего разума в ее существовании. Только через опыт возможно постижение истины – к такому выводу пришел Роджер Бэкон. И, критикуя метод Аристотеля, он писал: «Было бы лучше сжечь сочинения Аристотеля и начать все сызнова, нежели принимать его заключения без проверки».

Но францисканец поспешил. В XIII веке схоластическая наука еще не собиралась сдавать свои позиции. Аристотель считался вершиной научной мысли. И надо было дожидаться XVII века, когда человек, обладавший большой политической властью и непревзойденным красноречием, лорд Веруламский Фрэнсис Бэкон опубликует, свой труд под красноречивым и недвусмысленным названием Novum Organum. В этой работе философ обратил внимание ученых на важность экспериментального метода в науке. Мысль о том, что всякое научное положение, полученное в теории, должно подтверждаться практикой, сформулирована им четко и исчерпывающе. Фрэнсису Бэкону повезло куда больше, чем его однофамильцу. Он высказал свои мысли в нужное время, когда экспериментальная наука начала победное шествие по миру. И за это он стал признанным отцом нового направления в научном познании.

Но если внимательно разобраться в сочинениях Фрэнсиса Бэкона, то в них вряд ли удастся обнаружить пропагандируемый им метод индуктивного развития науки. Ничего подобного силлогистике Аристотеля у него нет. А поэтому вплоть до середины XIX века в области индуктивных рассуждений ничего не менялось. Их теории просто не существовало.

Появлению метода ДСМ предшествовали иерархические алгоритмы классификации, например схема Бонгарда

Переборная схема Бонгарда характеризуется тем, что решающая функция ищется в классе логических функций в виде дизъюнкции конъюнкций, определенных на описаниях объектов. Этот метод не требует априорной информации о характере функции распределения изображений в пространстве признаков; и допускает неполное описание (пропуски).

Алгоритм «Кора» основан на поиске сочетаний признаков (дизъюнкций конъюнкций), которыми обладают объекты одного класса и не обладают объекты другого. Признаки класса образуются в виде конъюнкций, составленных из признаков исходного описания. Качество каждой конъюнкции как признака класса определяется числом объектов, на которых эта конъюнкция принимает 1-ное значение. При распознавании определяется число конъюнкций – признаков каждого класса, принимающих на конкретном объекте единичное значение. Объект относится к тому классу, для которого это число оказывается наибольшим.

Обучение алгоритма «Кора» состоит в следующем. Рассматривается произвольная конъюнкция. Подсчитывается число объектов каждого класса, на которых она принимает 1-ное значение (пусть это l_j). Если для некоторого класса A_{j*}, $l_{j*} > l_0$, l_0 – некоторый порог, а для $i \neq j*, l_i = 0$, то эта конъюнкция – достаточный признак класса A_{j*}. Алгоритм «Кора», как и другие логические методы распознавания образов, является достаточно трудоемким, поскольку при отборе конъюнкций необходим полный или частично направленный перебор. Суть алгоритма: отыскивание конъюнкций, голосующих за класс. Решение принимается по большинству голосов.

9.2 Индукция Джона Стюарта Милля (ДСМ)

В процессе наблюдения за окружающим миром мы решаем две главные задачи, связанные с созданием модели, его описывающей. Прежде всего, мы выделяем в наблюдаемом некоторые сущности. В логике им соответствуют некоторые понятия, а, кроме того, мы устанавливаем между этими понятиями определенные отношения. Эти отношения могут быть как наблюдаемыми непосредственно с помощью наших органов чувств (например, отношения типа «субъект-действие» или «быть раньше»), так и достраиваемыми на основании некоторой «логики знаний» (например, отношения типа «причина – следствие» или «цель – средство»).

Среди всех этих отношений едва ли не главнейшую роль для познания окружающего мира играют *каузальные отношения,* отражающие в наиболее общей форме связи причин и следствий, внимание к которым привлекли исследования английского логика середины XIX века Джона Стюарта Милля. Он поставил перед собой задачу нахождения связей между фактами и явлениями на основе анализа их совместного появления или непоявления в последовательности экспериментов. При этом он принял меры к тому, чтобы

не повторять знаменитой ошибки при установлении *причинно-следственных* связей, которая вошла в историю науки под названием Post hoc ergo propter hoc, т. е. «После этого, значит вследствие этого». А ошибки такого типа не только встречались и встречаются в бытовых человеческих рассуждениях до сих пор, но иногда подобные выводы делаются сознательно, например, для создания неожиданных поэтических образов.

Принципы установления причинно-следственных отношений, которые предложил Милль, основываются на идеях выделения сходства и различия в наблюдаемых ситуациях внешнего мира.

Способность улавливать сходство и выделять различия – фундаментальная способность, по-видимому, всех живых существ. Опираясь на эту способность, Милль сформулировал свои принципы индукции.

– *Принцип единственного различия*

«Если после введения какого-либо фактора появляется, или после удаления его исчезает, известное явление, причем мы не вводим и не удаляем никакого другого обстоятельства, которое могло бы иметь в данном случае влияние, и не производим никакого изменения среди первоначальных условий явления, то указанный фактор и составляет причину явления».

Схематически этот принцип можно описать в виде следующей схемы:

$$\mathrm{n}\begin{cases} a,b,c \Rightarrow d, \\ a,b,c \Rightarrow d, \\ \ldots\ldots\ldots\ldots \\ a,b,c \Rightarrow d, \end{cases} \quad \mathrm{n}\begin{cases} b,c \not\Rightarrow d, \\ b,c \not\Rightarrow d, \\ \ldots\ldots\ldots \\ b,c \not\Rightarrow d, \end{cases} \quad a,b,c \Rightarrow d.$$

Здесь знак $\Rightarrow$ трактуется лишь как появление d при наличии a, b и c, а $\not\Rightarrow$ означает, что d не появляется. Повторение ситуаций n раз необходимо для того, чтобы убедиться в устойчивости всей ситуации в целом, для исключения случая, когда d появляется случайным образом, не будучи никак связанным с a. Если n, с точки зрения экспериментатора, достаточно для уверенного вывода, то, используя Принцип единственного различия, можно утверждать, что a является причиной, а d следствием, т. е. что между a и d имеет место причинно-следственное отношение. В дальнейшем будем называть реализации $a,b,c \Rightarrow d$ *положительными примерами* для d, а реализации $b,c \not\Rightarrow d$ – *отрицательными примерами* для d или *контрпримерами.*

– *Принцип единственного сходства*

«Если все обстоятельства явления, кроме одного, могут отсутствовать, не уничтожая этим явления, то это одно обстоятельство находится в отношении причинной связи с явлением условии, что приняты были все меры к тому,

чтобы никаких других обстоятельств, кроме принятых во внимание, налицо не оказалось».

Схематическое представление этого принципа Милля выглядит следующим образом:

$$\mathrm{n}\begin{cases} a,b,c \Rightarrow d, \\ a,b,c \Rightarrow d, \\ \ldots\ldots\ldots\ldots \\ a,b,c \Rightarrow d, \end{cases} \quad \mathrm{n}\begin{cases} a,b \Rightarrow d, \\ a,b \Rightarrow d, \\ \ldots\ldots\ldots\ldots \\ a,b \Rightarrow d, \end{cases} \quad \mathrm{n}\begin{cases} a \Rightarrow d, \\ a \Rightarrow d, \\ \ldots\ldots\ldots \\ a \Rightarrow d, \end{cases}$$

В этой схеме все примеры являются положительными. Из нее по Принципу единственного сходства вытекает, что *a* и *d* связаны причинно-следственным отношением.

– *Принцип единственного остатка*

«Если вычесть из какого-либо явления ту часть его, которая согласно прежним исследованиям оказывается следствием известных причин, присутствующих в явлении причин, то остаток явления есть следствие остальных причин».

Принцип единственного остатка можно проиллюстрировать следующей схемой:

$$\mathrm{n}\begin{cases} a,b,c \Rightarrow d,e, \\ a,b,c \Rightarrow d,e, \\ \ldots\ldots\ldots\ldots\ldots \\ a,b,c \Rightarrow d,e, \end{cases} \quad \mathrm{n}\begin{cases} b,c \Rightarrow e, \\ b,c \Rightarrow e, \\ \ldots\ldots\ldots\ldots \\ b,c \Rightarrow e, \end{cases}$$

Следовательно, *a* и *d* связаны причинно-следственным отношением, а *b* и *c* являются возможными причинами *e.* Для дальнейшего уточнения зависимости надо посмотреть, приводит ли исключение *b* к появлению *e.* Если приводит, то отношением «причина – следствие» связаны между собой *c* и *e. B* противном случае это отношение имеется между *b* и *e.*

Отметим ряд особенностей схем Милля. Прежде всего, они справедливы лишь при условии, что в описании ситуации имеется полное множество наблюдаемых фактов или явлений. Например, в последнем случае может оказаться, что и исключение *b,* и исключение *c* не влияют на появление *e* Тогда можно предположить, что для появления *e* необходимо либо одновременное наличие *b* и с, либо *e* вызывается чем-то, не вошедшим описание ситуации.

Другими словами, появление некоторого элемента ситуации может определяться не отдельными факторами или элементами, И их совокупностью, задаваемой с помощью сложного логического выражения. В ле-

вой части причинно-следственного отношения может стоять сложное выражение, в котором отдельные элементы могут быть связаны между собой конъюнктивными и (или) дизъюнктивными связками.

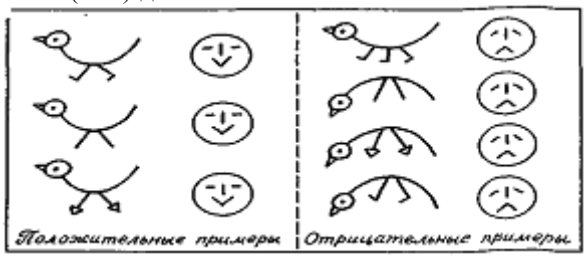


Рисунок 9.1 – Примеры и контрпримеры реакции человека

Проиллюстрируем это на примере. На рис.9.1 показаны ситуации, когда некий человек встречает на улице необычных зверюшек, то, глядя на них, он или радуется, или печалится. Нас интересует, какие качества зверюшек приводят человека в хорошее расположение духа. Другими словами, что является причиной его улыбки. Для удобства ответа на этот вопрос на рис.9.1 положительные примеры контрпримеры разделены штриховой чертой. Как видно из рисунка, зверюшки обладают тремя признаками: формой спины, числом ног и формой ног. Что же вызывает улыбку? Используем метод Милля. Возьмем в качестве первой возможной причины форму спины у зверюшки. Положительные примеры таковы, что во всех наблюдаемых случаях эта форма выгнута вниз. Обозначим этот признак через *a,* а реакцию человека, когда он радуется, через *d.* Можно ли утверждать, что *a* есть причина *d*? Согласно Принципу единственного сходства наличие спины такой формы должно всегда вызывать улыбку. Но первый же контрпример опровергает это Число ног (обозначим этот признак как *b)* также не может быть причиной улыбки. В положительных примерах *b* везде равно двум, и можно подумать, что именно две ноги зверюшки веселят человека. Но в трех контрпримерах ног тоже две. С формой ног (этот признак обозначим как *c*) ситуация в положительных примерах такова, что сразу ясно, что *c* не может быть причиной *d.*

Таким образом, ни один из признаков зверюшки по отдельности не может быть причиной улыбки человека. Попробуем выделить *общее ядро сходства* у всех зверюшек в положительных примерах. Такое ядро есть. Все зверюшки в этих примерах имеют выгнутую вниз спину и две ноги. Другими словами, для них всегда истинно утверждение $P_1(a) \& P_2(b)$, в котором $P_1(a)$ – предикат, интерпретируемый как «форма спины, выгнутая вниз», а $P_2(b)$ – предикат, интерпретируемый как «число ног равно

двум». Проверим, будет ли истинным выделенное ядро в отрицательных примерах. Простой проверкой убеждаемся, что оно везде ложно. Таким образом, причина улыбки человека найдена. Она возникает тогда и только тогда, когда встреченная им зверюшка имеет выгнутую вниз спину и две ноги.

Приведенный пример показывает, что при использовании методов индуктивных рассуждений, которые предложил Милль, весьма важную роль играет способ выделения признаков или фактов, с помощью которых описываются ситуации.

9.3 Рассуждения по аналогии

Первая попытка формализовать понятие рассуждения по аналогии была предпринята Лейбницем. В своем сочинении «Фрагменты логики» он ввел понятие пропорции для отношения аналогии. *Пропорция Лейбница* формулируется следующим образом: «Вещь *A* так относится к вещи *B,* как вещь *A′ к* вещи *B′«*. Обычно пропорцию Лейбница представляют в виде диаграммы рис 9.2.

Рисунок 9.2 – Иллюстрация рассуждений по аналогии

Для иллюстрации того, как может быть использована диаграмма Лейбница, рассмотрим *семантическое пространство Осгуда.* Это пространство, которое американский психолог Чарльз Осгуд строил экспериментально, проводя опыты с людьми, должно было, по его мнению, характеризовать организацию размещения информации в памяти человека. Упрощенное пространство Осгуда является обычным трехмерным евклидовым пространством. Близость по метрике этого пространства характеризует семантическую близость понятий, фактов и утверждений, а рассуждения, проведенные в пространстве относительно группы элементов, могут проецироваться по аналогии на группы, состоящие из семантически близких элементов. Проиллюстрируем эту мысль, взяв «кусок» пространства Осгуда, относящийся к понятиям, используемым для указания родства. То,

что они в семантическом пространстве расположены компактно, было доказано экспериментально. Этот «кусок» пространства Осгуда показан на рис.9.3. Для удобства введена система, координат и сделано такое преобразование, чтобы все точки, соответствующие интересующим нас понятиям, оказались лежащими в вершинах единичного куба (правомочность такого преобразования в пространстве Осгуда мы тут не обсуждаем).

Пусть даны три элемента пропорции Лейбница *A, A′* и *B*. И необходимо узнать элемент *B′*. Для рассматриваемого примера примем следующий способ нахождения координат понятия *B′*: $B'_i = B'_i + a'_i - a'_i$, где i = 1,2,3. Пусть, например, нас интересует пропорция Сын: Дочь = Дядя : ? Для определения неизвестного члена пропорции произведем необходимые вычисления, используя координаты понятий, отмеченные на рис.9.3.

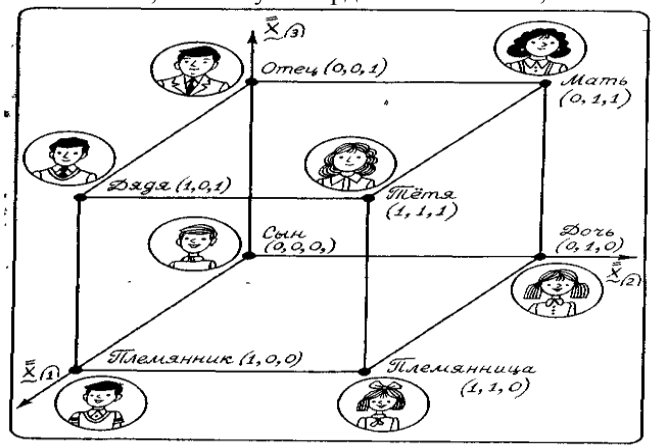


Рисунок 9.3 – Иллюстрация семантического пространства Осгуда

Получим $Б'_1 = 0+1-0=1$; $B'_2 = 1+0-0=1$; $B'_3 = 0+1-0=1$. Таким образом, понятие *B′* имеет координаты (1,1,1). Этим координатам соответствует понятие «Тетя».

Для дальнейшего необходимо уточнить понятия «похожесть» «аналогия», использованные в диаграмме для пропорции Лейбница, и придать им по возможности строгий смысл. Сделать это можно следующим образом. Выберем некоторый алгебраический язык для описания *A* и *B,* который обозначим I_1 и некоторый (вообще говоря, другой) алгебраический язык для описания *A′* и *B′,* который обозначим I_2 .

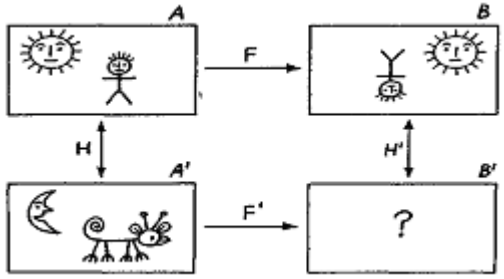


Рисунок 9.4 – Иллюстрация пропорции Лейбница

Переход от, A к B и от A' к B' будем интерпретировать как преобразование соответствующих описаний в языках I_1 и I_2. Поскольку выбранные языки являются алгебраическими, то в них выделены элементы и операции, определённые над этими элементами. Учитывая дальнейший пример, будем считать, что в качестве элементов языков I_1 и I_2 выступают некоторые изображения или их совокупности, связанные отношениями из заданного набора двуместных отношений. А операции состоят в том, что над элементами можно совершать различные геометрические преобразования, определяемые их движениями. Это приводит к изменению отношений между элементами, входящими в анализируемые совокупности.

Чтобы все сказанное стало понятнее, рассмотрим конкретный пример. На рис.9.4 показана серия изображений, соответствующая пропорции Лейбница, в которой, как всегда, надо восстановить недостающее звено, т. е. осуществить (если это возможно) вывод по аналогии. Для описания изображений введем языки I_1 и I_2. В языке I_1 в качестве элементов возьмем изображение солнца s, и человечка m. В качестве отношений будем рассматривать отношения R_1 – «быть слева вверху» и R_2 – «быть справа вверху». Тогда ситуация A может быть описана как sR_1m. В качестве операций в I_1 будем использовать перестановку объектов относительно друг друга O_1 и вращение на 180° по часовой стрелке O_2. Тогда преобразование F южно описать как O_1 (s, m); O_2 (m). В результате этого возникает ситуация B, описание которой в языке I_1 выглядит как $sR_2(0_2(m))$.

Введем теперь элементы языка I_2. Это луна l и фантастическое животное q. В качестве отношений, используемых в I_2 , возьмем снова отношения R_1и R_2, а в качестве операций I_2 сохраним операции O_1 и 0_2 языка I_1. Описание A' выглядит следующим образом: $l\ R_1q$. Для получения описания B' установим между A и A' отношение взаимно однозначного соответствия H, например, так, что имеют место взаимно однозначные соответствия s ↔l и m↔q. Тогда $sR_1m \leftrightarrow lR_1q$ и $A \leftrightarrow A'$. Преобразование F' в наших

предположениях совпадает с *F*. Значит, B и *B′* должны находиться также во взаимно однозначном соответствии. Но *B* есть $sR_2(0_2(m))$. Учитывая соответствие между элементами I_1 и I_2, выводим описание для *B′*: $lR_2(0_2(q))$.

Рассмотренная процедура носит общий характер. Можно строго доказать, что если в пропорции Лейбница *A, A′ к B* описаны с помощью алгебраического языка, использующего лишь двуместные отношения, задан характер преобразований *F* и установлено взаимно однозначное соответствие между I_1 и I_2, то описание *B′* также возможно на языке I_2 и существуют взаимно однозначные соответствия *F*↔*F′* и *B*↔*B′,* так что, применяя к *A* преобразование F и к *A′* преобразование *F′*, поучаем *B* и *B′,* такие, что *B*↔*B′*.

Заметим, что из этого утверждения вытекает, что необходимым условием для возможности рассуждений по аналогии с использованием пропорции Лейбница служит требование коммутативности ее диаграммы. Требование *коммутативности диаграммы* означает, что описание *B′,* полученное из *A* с помощью взаимно однозначного соответствия *H*, ничем не отличается от описания *B′,* полученного из *A* с помощью взаимно однозначного соответствия *H* и последующего применения к этому результату преобразования *F′*.

Несмотря на все сказанное, полное описание модели рассуждений по аналогии всё еще не получено, так как пропорция Лейбница явно не исчерпывает всех случаев рассуждений подобного типа. Да и в случае, когда мы имеем дело действительно с пропорцией Лейбница, остаются нерешенными по крайней мере два вопроса: как построить языки I_1 и I_2 и как установить взаимно однозначное соответствие между ними. Возможные в этом случае трудности иллюстрирует рис.9.5 с изображением ситуации *A* и *A′*..

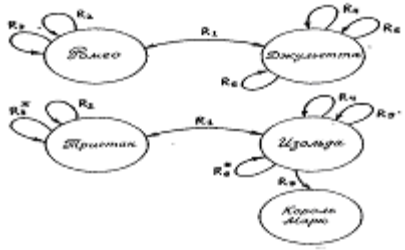


Рисунок 9.5 – Проблемы при рассуждениях по аналогии

Ситуация *A* может быть писана следующим текстом: < Ромео любит Джульетту. Джульетта любит Ромео (на рис.9.5 это отношение R_1). Ромео мужчина (R_2). Он итальянец (*R*з). Джульетта женщина (R_4). Она красива (R_5). Она не замужем (R_6)>. Ситуация *A′* может быть описана

следующим текстом: <Тристан любит Изольду. Изольда любит Тристана *(R_1).* Тристан мужчина *(R_2).* Он бретонец (R_2 *). Изольда женщина *(R_4).* Она красива *(R_2).* Она замужем (R_6*). Ее муж – король Марк *(R_7)*>.

Готовы ли мы признать описанные две ситуации аналогичными? И должен ли Тристан действовать так же, как Ромео? Из соответствующих литературных произведений мы знаем, что развитие ситуации *A* было таково, что оно привело к совместной смерти Ромео и Джульетты. А Тристан и Изольда имели другую судьбу. Почему это произошло? И можно было бы это формально установить в процессе сравнения ситуаций *A* и *A*'? Ведь во второй ситуации имелся король Марк, а различное число отношений заведомо не позволяло установить взаимно однозначное отношение между их описаниями. Но может быть вместо изоморфизма (т. е. взаимно однозначного отношения) для I_1 и I_2 достаточно какого-нибудь гомоморфизма? Этот вопрос пока остается без ответа. Поэтому ограничимся лишь тем, что для рассуждений по аналогии можно считать твердо установленным.

9.4 ДСМ-метод

Сокращение ДСМ, вынесенное в название метода, означает инициалы автора Джон Стюарт Милль. Оно показывает, что метод поиска закономерностей по множествам положительных и отрицательных примеров, к описанию которого мы переходим, опирается на методы индукции, предложенные этим ученым. Их реализация в виде комплекса действующих программ на ЭВМ выполнена современными исследователями.

Введем три множества: *причин* $A=\{ a_1, a_2 ,..., a_p,\}$ *следствий* $З = \{ b_1, b_2 ,..., b_p,\}$ и *оценок* $Q =\{ q_1, q_2 ,..., q_p,\}$. Выражение вида $a,=>b, q_i,$ будем называть *положительной гипотезой.* Оно связано с утверждением типа «a_i, является причиной b_i, *с* оценкой достоверности q_k». Выражение вида $a_i \neq> b_i$; q_k будем называть *отрицательной гипотезой.* Оно связано с утверждением типа «a_i, не является причиной b_i, *с* оценкой достоверности q_k». Для сокращения записи положительные гипотезы будем обозначать *hij+* а отрицательные – *hij-*. Среди значений q_i выделим два специальных, которые можно обозначить 0 и 1. Значение 0 приписанное положительной или отрицательной гипотезе, означает, что соответствующее утверждение является ложным. Приписывание гипотезам значения 1 означает, что данная гипотеза является тождественно истинной. Таким образом, гипотезы с оценками 0 и 1 можно рассматривать как высказывания, ложность и истинность которых твердо установлены. Все остальные оценки, отличные от 0 и 1, будем обозначать рациональными числами вида *s/n,* где s пробегает значения от *1* до *n–1*. Величина *n* характеризует дробность используемых оценок достоверности. Чем больше n, тем с большей точностью оценивается степень достоверности гипотез.

Пусть мы вдруг оказались в стране, где до этого нам не приходилось бывать. Выйдя из гостиницы, мы увидели, что у подъезда стоит такси, выкрашенное в ярко-желтый цвет. Через некоторое время рядом останавливается еще одно такси такого же цвета. В нашей голове возникает положительная гипотеза вида <В этой стране, если автомобиль выполняет роль такси, то цвет его будет желтым>. Оценка достоверности этой гипотезы при двух наблюдениях будет невелика. Но если во время прогулки по улицам города мы увидим, что такси окрашены в тот же желтый цвет, то оценка выдвинутой при выходе из гостиницы гипотезы будет все время возрастать. Станет ли она когда-нибудь равной единице? Если после недельного пребывания в стране наша гипотеза будет подтверждаться лишь положительным примерами, то на родине, рассказывая знакомым и друзьям о своих впечатлениях, связанных с поездкой, мы вполне можем заявить: < А такси у них покрашены в ярко-желтый цвет, что очень удобно – сразу можно найти его, когда нужно>. Значит ли это, что гипотеза о цвете такси приобрела оценку достоверности, равную 1?

Можно ввести два типа истинности: *эмпирическую истину* и *теоретическую истину.* В нашем примере высказыванию о цвете такси мы, конечно, приписываем эмпирическую истину. Просто все наши наблюдения были в пользу данной гипотезы. Но мы вполне можем допустить, что есть небольшое количество такси иного цвета. Они ни разу не попадались нам на глаза. Совсем другое положение будет в том случае, когда в путеводителе, обнаруженном в гостинице, будет сказано, что закон данной страны запрещает окрашивать такси в какие-либо другие цвета, кроме желтого. При такой информации высказывание о желтом цвете такси будет оценено как теоретическая истина.

На этом простом примере видна разница между дедуктивным и индуктивным умозаключением. При использовании информации из путеводителя о цвете такси вы уже не нуждаетесь в эксперименте. Полученное знание носит общий характер. В каждом конкретном случае (например, при поиске такси) его можно механически применять, фиксируя цвета проходящих машин. Никакого нового знания при решении конкретных задач, связанного с цветом такси, получить нельзя. При получении же информации из наблюдений формируется новое знание, которого раньше не было. Гипотеза о цвете такси в данной стране – это новая информация. Таким образом, индуктивное рассуждение способно порождать новые знания. В этом смысле оно куда более интеллектуально, чем дедуктивное рассуждение.

Достижение эмпирической истины (а только такая истина и возможна при индуктивных рассуждениях) вполне возможно. Для этого достаточно некоторого множества положительных примеров при полном отсутствии отрицательных примеров, опровергающих выдвинутую гипотезу. А число необходимых положительных примеров, необходимых для того, чтобы считать гипотезу эмпирически истинной, может быть разным в различных обстоятельствах и у разных

людей. Недаром же все представители рода человеческого делятся на тех, кто готов верить в нечто всего по одному примеру, и тех, кто подобно евангельскому Фоме никогда не может уверовать до конца даже в самые очевидные для остальных истины.

Рассмотренный пример иллюстрирует процесс оценивания степени достоверности гипотезы, когда предполагаемая причина (в нашем случае – принадлежность автомашины к множеству такси) уже выделена из множества возможных причин. В ДСМ-методе формализован не только этот этап, но и предшествующий ему этап нахождения кандидата в причины, которая могла бы вызвать интересующее нас следствие. В примере это соответствовало бы следующему. Наблюдая на улицах города потоки автотранспорта и выделяя среди автомашин ярко-желтые, надо < сообразить>, что желтыми являются только такси.

Причины могут быть различными по типу. Наиболее редкими (являются *необходимые и достаточные причины.* Если a_i – причина такого типа, то b_j происходит всегда, и если b_j произошло, то наверняка было a_i. Примерами такой < жесткой> связи двух явлений может служить падение тела, если для него отсутствует опора. Чаще встречаются *достаточные причины,* всегда вызывающие появление b_j. Но появление b_j не служит стопроцентным обоснованием того, что до этого было a_i. Следствие b_j могло быть вызвано и какими-то другими достаточными причинами. Если, например, ваш друг не пришел в условленное место и в условленное время на свидание, то, возможно, он заболел, ибо болезнь – достаточная причина для отказа от свидания, но весьма вероятно, что были какие-то другие причины нарушения им своего обещания.

Дополнительные причины обладают тем свойством, что их наличие не вызывает следствия b_j. Для того чтобы b_j, появилось, нужен вполне определенный набор дополнительных причин, который выступает в роли обобщенной достаточной причины появления b_j. Легко себе представить такой набор причин, который приводит к попаданию мяча в сетку ворот при игре в футбол. Перечисление и обсуждение дополнительных причин, приведших к голу,- знакомое занятие для каждого истинного любителя футбола. Среди дополнительных причин могут быть *необходимые дополнительные причины.* Их вхождение в набор, образующий обобщенную достаточную причину, обязательно для того, чтобы b_j, реализовалось. Остальные дополнительные причины можно назвать *факультативными.* В окончательный набор могут входить те или иные комбинации факультативных причин. Так, в ситуации забивания гола две дополнительные причины являются заведомо необходимыми: удар, посылающий мяч в ворота, и ошибка вратаря. Остальные дополнительные причины являются факультативными. Наконец, *возможные причины* a_i обладают тем свойством, что появление *a,* необязательно вызывает b_j – но увеличивает возможность появления b_j,.

Кроме причин a_i важную роль в процессах реализации причинно-следственных зависимостей играют так называемые *тормоза*. Наличие тормоза наряду с причиной, вызывающей b_j, в обычных условиях, приводит к тому, что b_j не появляется. Так, принятие смертельной дозы яда не приводит к ожидаемому исходу, если до этого было принято противоядие.

В ДСМ-методе нахождение причин – кандидатов для формируемых гипотез – дело далеко не простое. В положительных и отрицательных примерах эти причины скрыты в описаниях реальных объектов, обладающих или не обладающих интересующими нас свойствами. Из этих описаний надо выделить кандидатов в причины, а затем убедиться, что выбор оказался не случайным.

При первом реальном использовании ДСМ-метода одной из конкретных задач была задача нахождения причин того, что некоторое органическое химическое соединение будет обладать свойством биологической активности. Постулировалась, что информация о причинах биологической активности скрыта в структурной формуле того или иного соединения. Какие-то особенности этих формул оказывали влияние на интересующее исследователей свойство. Экспериментально для многих соединений было установлено наличие или отсутствие в них биологической активности. Эти экспериментальные факты составляли множество положительных и отрицательных примеров. На основании их программы, реализующие ДСМ-метод, должны были найти новые, не известные химикам и фармакологам закономерности, позволяющие без экспериментальной проверки (весьма дорогой и длительной) оценивать возможность того, что вновь синтезированное вещество будет обладать биологической активностью. Суть того, как это делалось с помощью ДСМ-метода, состоит в следующем. Рассмотрим группу положительных примеров. Находим некоторую часть описания объектов, общую для определенной совокупности примеров из этой группы. Например, обнаруживаем в значительной части структурных формул соединений, обладающих свойством биологической активности, кольцевую структуру с фиксированным заполнением позиций в этой структуре. Тогда есть основания считать ее кандидатом в причины. Таких кандидатов может оказаться несколько. Образуем матрицу M^+, в которой строки соответствуют выделенным кандидатам a_i, а столбцы – интересующим нас следствиям b_j (при одном интересующем нас следствии в M^+ будет один столбец). На пересечении строк и столбцов будем записывать оценки достоверности q_k гипотез $h_i{}^+{}_{jk}$. Об их нахождении будет сказано ниже. Для множества отрицательных примеров аналогичным образом строится другая матрица M^-, в которой содержатся оценки достоверности отрицательных гипотез $h_i{}^-{}_{ik}$. Кандидаты в причины в матрицах M^+ и M^- могут частично совпадать, так как положительные и отрицательные примеры не образуют полной выборки из всего множества возможных примеров.

На каждом шаге работы ДСМ-метода используются новые наблюдения, пополняющие множества положительных и отрицательных примеров. Эти новые наблюдения могут либо подтверждать сформированные гипотезы, либо противоречить им. В первом случае надо увеличивать оценки достоверности соответствующих гипотез, а во втором – уменьшать их. Механизм изменения оценок q_k, может быть различным. В ДСМ-методе он устроен следующим образом. Значение n совпадает с числом имеющихся в данный момент положительных или отрицательных примеров. Таким образом, для M^+ и M^- значение n может оказаться различным. С ростом n растет «дробность» оценок достоверности. Оценка $1/n$ играет особую роль. Она соответствует полному незнанию о достоверности гипотезы. Поэтому в начальный момент M^+ и M^- заполнены лишь нулями, единицами и оценками $1/n$. Значения истинности и лжи могут иметь гипотезы, у которых в качестве причин даны полные описания объектов, образующих множества примеров.

Если некоторая положительная или отрицательная гипотеза h_{ijk} имела оценку k/n, то при появлении нового примера (n заменяется на $n+1$) проверяется, подтверждает или не подтверждает новый пример эту гипотезу. При подтверждении оценка k/n заменяется на $(k+1)/(n+1)$, а при неподтверждении новым примером ранее выдвинутой гипотезы ее оценка меняется с k/n на $(k-1)/(n+1)$. Таким образом, в процессе накопления новой информации оценки гипотез либо приближаются к О или 1, либо ведут себя каким-либо «колеблющимся» образом. В первом случае гипотеза может на некотором шаге (когда будет пройден некоторый априорно заданный нижний порог достоверности) исчезнуть из M^+ или M^-. Во втором случае при достижении некоторого верхнего порога достоверности гипотеза может получить оценку, отражающую эмпирическую истину, и запомниться как некий установленный факт в системе или эта гипотеза сообщается человеку, работающему с ДСМ-программами. В третьем случае, если колебания оценок достаточно сильны, может также произойти исключение сформированной ранее гипотезы из тех, которые описаны в M^+ или M^-. Новые гипотезы формируются не только на основании выделения в примерах определенного сходства (общей части в описании). Они могут использовать и метод различия, также сформулированный Миллем. Различие выявляется для примеров, относящихся к группам положительных и отрицательных примеров. Найденное различие служит кандидатом для гипотез, включаемых в M^+ или M^-.

Кроме выявления кандидатов в причины a_i для положительных и отрицательных гипотез в описываемом методе ищутся также тормоза, наличие которых снимает влияние a_i на появление b_i. В новых версиях метода в качестве a_i выступают весьма сложные утверждения, в которых отдельные части описаний объектов могут быть связаны между собой

произвольными логическими выражениями, например, следующего типа: <Если в объекте есть a' и a» и нет a'« или в объекте есть a»«, то свойство b имеет место >.

В ДСМ-методе кроме прямой реализации идей Милля используются еще некоторые выводы по аналогии. Для этого на множестве описаний объектов вводится тем или иным способом понятие сходства. Если, например, речь идет о структурных формулах химических соединений, то мерой сходства для них могут быть совпадение самих структур при различных заполнителях позиций или, наоборот, наличие в некоторых фиксированных позициях структур одинаковых элементов. Если установлено отношение сходства, то в ДСМ-методе происходит вывод по аналогии. Он осуществляется следующим способом. Если гипотеза h_{ijk} имеет оценку k/n и такова, что причина, используемая в ней, сходна с причиной в гипотезе $h'_{i\ j\ k}$ имеющейся в той же матрице M и оцениваемой с точки зрения достоверности значением k/n, то на гипотезу $h'_{i\ jk}$ переносится оценка гипотезы h_{ijk} и она получает оценку достоверности k/n. Подобная процедура в ДСМ-методе называется правилом положительной аналогии. Существует в этом методе и правило отрицательной аналогии, а также градация тех и других правил по силе учитывающегося в них сходства. Таким образом, ДСМ-метод демонстрирует возможность проведения правдоподобных рассуждений весьма широкого спектра.

Тема 10 Метод группового учета аргументов

10.1 Общая схема метода группового учета аргументов

10.2 Основные принципы группового учета аргументов

10.3 Отбор лучшего решения в методе группового учета аргументов

10.4 Алгоритм с ковариациями и с квадратичными описаниями

10.1 Общая схема метода группового учета аргументов

Метод группового учета аргументов (МГУА) предложен в конце 60-х – 70-х академиком А.Г.Ивахненко (ИК АН УССР). Этот метод использует идеи самоорганизации и механизмы живой природы – скрещивание (гибридизацию) и селекцию (отбор). Заимствование алгоритмов переработки информации у природы является одной из основных идей кибернетики. «Гипотеза селекции» утверждает, что алгоритм массовой селекции растений или животных является оптимальным алгоритмом переработки информации в сложных задачах. При массовой селекции высевается некоторое количество семян. В результате опыления образуются сложные наследственные комбинации. Селекционеры выбирают некоторую часть растений, у которых инте-

ресующее их свойство выражено больше всего (эвристический критерий). Семена этих растений собирают и снова высевают для образования новых, еще более сложных комбинаций. Через несколько поколений селекция останавливается и ее результат является оптимальным. Если чрезмерно продолжать селекцию, то наступит «инцухт» – вырождение растений. Существует оптимальное число поколений и оптимальное количество семян, отбираемых в каждом из них.

Алгоритмы МГУА воспроизводят схему массовой селекции, в них есть генераторы усложняющихся из ряда в ряд комбинаций и пороговые самоотборы лучших из них. Так называемое «полное» описание объекта

$$\varphi = f(x_1,x_2,x_3,...,x_m),$$

где f – некоторая элементарная функция, например степенной полином, заменяется несколькими рядами «частных» описаний:

1-ряд селекции: $y_1 = f(x_1,x_2)$, $y_2 = f(x_1,x_3)$,..., $y_s = f(x_{m-1}x_m)$,

2-ряд селекции: $z_1 = f(y_1y_2)$, $z_2 = f(y_1y_2)$,..., $z_p = f(y_{s-1}y_s)$, где s=c2, p=cs2 и т.д.

Входные аргументы и промежуточные переменные сопрягаются попарно, и сложность комбинаций на каждом ряду обработки информации возрастает (как при массовой селекции), пока не будет получена единственная модель оптимальной сложности.

Каждое частное описание является функцией только двух аргументов. Поэтому его коэффициенты легко определить по данным обучающей последовательности при малом числе узлов интерполяции. Исключая промежуточные переменные (если это удается), можно получить «аналог» полного описания. Например, по десяти узлам интерполяции можно получить в результате оценки коэффициентов полинома сотой степени и т. д.

Из ряда в ряд селекции пропускается только некоторое количество самых регулярных переменных. Степень регулярности оценивается по величине среднеквадратичной ошибки (средней для всех выбираемых в каждом поколении переменных или для одной самой точной переменой) на отдельной проверочной последовательности данных. Иногда в качестве показателя регулярности используется коэффициент корреляции.

Ряды селекции наращиваются до тех пор, пока регулярность повышается. Как только достигнут минимум ошибки, селекцию, во избежание «инцухта», следует остановить. Практически рекомендуется остановить селекцию даже несколько раньше достижения полного минимума, как только ошибка начинает падать слишком медленно. Это приводит к более простым и более достоверным уравнениям.

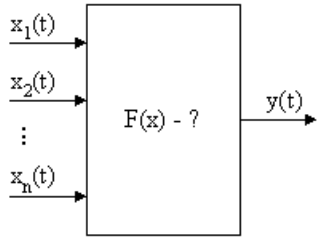


Рисунок 10.1 – Схема метода группового учета аргументов

По результатам наблюдений надо определить F(x). Причем даже структура модели F(x) неизвестна.

10.2 Основные принципы метода группового учета аргументов

Пусть имеется выборка из N наблюдений: $\{x_1, y_1\}$, $\{x_2, y_2\}$, … $\{x_N, y_N,\}$.

Наиболее полная зависимость между входами x_i и выходами y_i может быть представлена с помощью обобщенного полинома Колмогорова-Габора. Пусть есть $X=(x_1, x_2, ..., x_N)$, тогда такой полином имеет вид:

$$Y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{j=1}^{N} \sum_{i \le j} a_{ij} x_i x_j + \sum_{i=1}^{N} \sum_{j \le i} \sum_{k<=j} a_{ijk} x_i x_j x_k + \dots$$

где все коэффициенты $\boldsymbol{a}$ не известны.

При построении модели (при определении значений коэффициентов) в качестве критерия используется критерий регулярности (точности):

$$\varepsilon^2 = \left(\sum_{i-1}^{N} (y_i - f(x_i))^2 \right) \Big/ N \to min$$

Принцип множественности моделей: существует множество моделей на данной выборке, обеспечивающих нулевую ошибку (достаточно повышать степень полинома модели). Т.е. если имеется N узлов интерполяции, то можно построить целое семейство моделей, каждая из которых при прохождении через экспериментальные точки будет давать нулевую ошибку $\bar{\varepsilon}^2 = 0$

Обычно степень нелинейности берут не выше $n-1$, если n – количество точек выборки.

Обозначим S – сложность модели (определяется числом членов полинома Колмогорова-Габора).

Значение ошибки $\bar{\varepsilon}^2$ зависит от сложности модели. Причем по мере роста сложности сначала она будет падать, а затем расти. Нам же нужно выбрать такую оптимальную сложность, при которой ошибка будет минималь-

на. Кроме того, если учитывать действие помех, то можно выделить следующие моменты:

1 При различном уровне помех зависимость $\bar{\varepsilon}^2$ от сложности S будет изменяться, сохраняя при этом общую направленность (имеется в виду, что с ростом сложности она сначала будет уменьшаться, а затем – возрастать).

2 При увеличении уровня помех величина $min\limits_{s}\bar{\varepsilon}^2$ будет расти.

3 С ростом уровня помех, $S_0 = arg\,min(\bar{\varepsilon}^2)$ будет уменьшаться (оптимальное значение сложности будет смещаться влево) см.рисунок 10.2. Причем $\bar{\varepsilon}^2(S_0) > 0$, если уровень помех не нулевой.

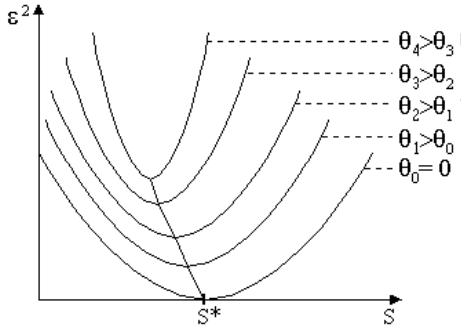


Рисунок 10.2 – Поведение оптимального значения с ростом сложности модели

Теорема неполноты Гёделя: В любой формальной логической системе имеется ряд утверждений и теорем, которые нельзя ни опровергнуть, ни доказать, оставаясь в рамках этой системы аксиом. В данном случае эта теорема означает, что выборка **всегда неполна**. Один из способов преодоления этой неполноты – принцип внешнего дополнения. В качестве внешнего дополнения используется дополнительная выборка (проверочная), точки которой не использовались при обучении системы (т.е. при поиске оценочных значений коэффициентов полинома Колмогорова-Габора).

Поиск наилучшей модели осуществляется таким образом:

– Вся выборка делится на обучающую и экзаменационную (проверочную): $N_{выб}=N_{обуч}+N_{пров}$

– На обучающей выборке $N_{обуч}$ определяются значения a_0, a_i, a_{ij}.

– На проверочной выборке $N_{пров}$ отбираются лучшие модели.

Входной вектор имеет размерность N: $X=(x_1, x_2, \ldots, x_N)$.

Принцип свободы выбора (неокончательности промежуточного решения):

1 Для каждой пары x_i и x_j строятся частичные описания (всего C^2_N) вида:

-линейные: $\hat{Y}^{(s)} = \varphi(x_i, x_j) = a_0 + a_i x_i + a_j x_j, s = 1 \ldots C_N^2$

-квадратичные: $\hat{Y}^{(s)} = \varphi(x_i, x_j) = a_0 + x_i + a_j x_j + a_{ii} x_i^2 + a_{ij} x_i x_j + a_{jj} x_j^2, s = 1 \ldots C_N^2$

2 Определяем коэффициенты этих моделей по МНК, используя обучающую выборку. Т.е. находим $a_0, a_1, \ldots a_j, \ldots a_N, a_{11}, \ldots a_{1j}, \ldots a_{NN}$.

3 Далее на проверочной выборке для каждой из этих моделей ищем оценку

$$\bar{\varepsilon}_s^2 = \left(\sum_{л-1}^{N_{пров}} (Y(k) - Y_k^{(s)})^2 \right) \Big/ N_{пров}$$

где $Y(K)$ – действительное значение выходное значение в k-той точке проверочной выборки; $Y_k^{(S)}$- выходное значение в k-той точке проверочной выборки в соответствии с s-той моделью и определяем F лучших моделей.

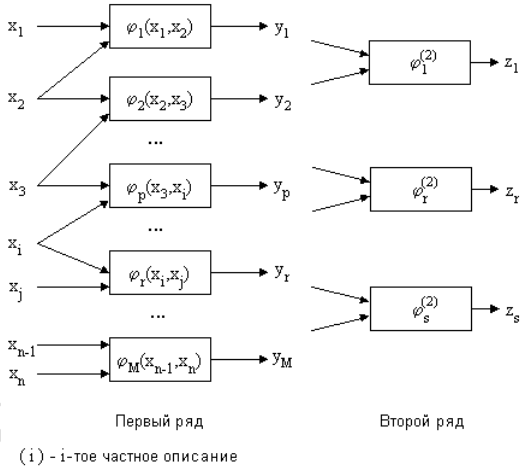


Рисунок 10.3 – Схема селекции решения

Выбранные y_i подаются на второй ряд. Ищем

$$z_i = \varphi^{(2)}(x_i, x_j) = a_0^{(2)} + a_1^{(2)} x_i + a_2^{(2)} x_j + a_3^{(2)} x_i^2 + a_4^{(2)} x_i x_j + a_5^{(2)} x_j^2$$

Оценка здесь такая же, как на первом ряде. Отбор лучших осуществляется опять так же, но $F_2 < F_1$. Процесс конструирования рядов повторяется до тех, пока средний квадрат ошибки будет падать. Когда на слое m получим увеличение ошибки $\bar{\varepsilon}^2$,то прекращаем.

Если частичные описания квадратичные и число рядов полинома S, то получаем, что степень полинома $k=2^S$.

В отличие от обычных методов статистического анализа, при таком подходе можно получить достаточно сложную зависимость, даже имея короткую выборку.

Есть проблема: на первом ряде могут отсеяться некоторые переменные x_i и x_j, которые оказывают влияние на выходные данные.

В связи с этим предложена такая модификация: на втором слое подавать y_i и x_j, т.е.:

$$z_i = a_0^{(2)} + a_1^{(2)} y_i + a_2^{(2)} x_j + a_3^{(2)} y_i^2 + a_4^{(2)} y_i x_j + a_5^{(2)} x_j^2$$

.

Это важно при большем уровне помех, чтобы обеспечить несмещенность.

10.3 Отбор лучшего решения в методе группового учета аргументов

Возникает два способа отбора лучших кандидатов частичных описаний передаваемых на определенном слое.

1. *Критерий регулярности* (точности) $\bar{\varepsilon}_{пр}^2$

$$\overline{\varepsilon^2} = \frac{1}{N_{пр}} \sum_{i-1}^{N_{пр}} (y_i - y_i^*)^2 \quad , \quad \overline{\Delta_{пр}^2} = \sum_{i-1}^{N_{пр}} (y_i - y_i^*)^2 \Big/ \sum_{i-1}^{N_{пр}} (y_i - \bar{y})^2 \quad .$$

,

2. *Критерий несмещенности*.

Берем всю выборку, делим на две части $R = R_1 + R_2$.

Первый эксперимент: R_1- обучающая выборка, R_2- проверочная; определяем выходы модели y^*_i ,i=1..R.

Второй эксперимент: R_2 – обучающая выборка, R_1 – проверочная; определяем выходы модели y^{**}_i, $i=1..R$.

Сравниваем результаты. Определяем критерий несмещенности:

$$n_{cm} = \Big(\sum_{i-1}^{R} (y_i^* - y_i^{**})^2 \Big) \Big/ N$$

Чем меньше $n_{cm,}$ тем более несмещенной является модель. Такой критерий определяется для каждого частичного описания первого уровня и затем находится n_{cm} для уровня в целом

$$n_{cm} = \Big(\sum_{i-1}^{F} n_{cm\,i}^{(1)} \Big) \Big/ F$$

для F лучших моделей. В ряде вариантов $F=1$. Такое же самое на втором слое $n^{(2)}_{cm}$.

Процесс селекции осуществляется до тех пор, пока этот критерий не перестанет уменьшаться, т.е. до достижения условия

$$n_{cm\,i}^{(2)} \to min$$

10.4 Алгоритм с ковариациями и с квадратичными описаниями

В этом алгоритме используются частные описания, представленные в следующих формулах:

$$y_i = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j;\ \ y_k = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2.$$

Сложность модели в методе группового учета аргументов увеличивается от ряда к ряду селекции как по числу учитываемых аргументов, так и по степени. Степень полного описания быстро растет. На первом ряду – квадратичные описания, на втором – четвертой степени, на третьем – восьмой и т.д. В связи с этим минимум критерия селекции находится быстро, но не совсем точно. Кроме того, имеется опасность потери существенного аргумента, особенно на первых рядах селекции (в случае отсутствия протекции). Специальные теоремы теории МГУА определяют условия, при которых результат селекции не отличается от результата полного перебора моделей.

Для того чтобы степень полного уравнения повышалась с каждым рядом селекции на единицу, достаточно рассматривать все аргументы и их ковариации как обобщенные аргументы и пользоваться составленными для них линейными описаниями.

Тема 11 Нечёткая логика и её применение

11.1 Основы нечеткой логики

11.2 Операции над нечёткими множествами

11.3 Пример использования нечёткие правила вывода в экспертной системе

11.4 Программное обеспечение на основе нечеткой логики

11.1 Основы нечеткой логики

Довольно часто оптимальное решение практической задачи трудно найти, используя классические методы математики. Причины этого состоят в следующем. Во-первых, не всегда возможно сделать приемлемое с точки зрения точности и компактности аналитическое описание решаемой задачи. Во многих случаях затраты на его разработку превысили бы эффект от решения, а кроме того, время, необходимое для получения аналитического описания, как правило неприемлемо велико. Но всегда ли такое описание

требуется, ведь человек способен находить оптимальные решения, пользуясь лишь абстрактными сведениями и субъективными представлениями о задаче? Во-вторых, в жизни нам постоянно приходится оперировать неточными значениями и не вполне ясными понятиями, однако традиционные методы математики не допускают таких «вольностей». Осознание этих проблем привело к появлению новой математической дисциплины – нечёткой логики, претендующей на устранение противоречий между математикой и реальным миром.

Нечёткая логика (fuzzy logic) – это надмножество классической булевой логики. Она расширяет возможности классической логики, позволяя применять концепцию неопределённости в логических выводах. Употребление термина «нечёткий» применительно к математической теории может ввести в заблуждение. Более точно её суть характеризовало бы название «непрерывная логика». Аппарат нечёткой логики столь же строг и точен, как и классический, но вместе со значениями «ложь» и «истина» он позволяет оперировать значениями в промежутке между ними. Говоря образно, нечёткая логика позволяет ощущать все оттенки окружающего мира, а не только чистые цвета.

Нечёткая логика как новая область математики была представлена в 60-х годах профессором калифорнийского университета Лотфи Заде (Lotfi Zadeh). Первоначально она разрабатывалась как средство моделирования неопределённости естественного языка, однако впоследствии круг задач, в которых нечёткая логика нашла применение, значительно расширился. В настоящее время она используется для управления линейными и нелинейными системами реального времени, при решении задач анализа данных, распознавания, исследования операций.

Нечёткое множество F может быть представлено как отображение элементов множества S на интервал I = [0, 1]. Это отображение определяется множеством упорядоченных пар: $\{s_i, m_F(s_i)\}$, $i \in [1, n]$, где s_i – i-й элемент множества S; n – мощность множества S; $m_F(s_i) \in [0, 1]$ – степень вхождения элемента s_i в множество F. Значение $m_F(s_i)$, равное 1, означает полное вхождение, $m_F(s_i) = 0$ указывает на то, что элемент s_i не принадлежит множеству F. Часто отображение задаётся функцией $m_F(x)$ принадлежности x нечёткому множеству F. В силу этого термины «нечёткое подмножество» и «функция принадлежности» употребляются как синонимы. Степень истинности предиката «$s_k \in F$» определяется путём нахождения парного элементу s_k значения $m_F(s_k)$, определяющего степень вхождения s_k в F.

Обобщая геометрическую интерпретацию традиционного подмножества на нечёткий случай, получаем представление F точкой в гиперкубе I^n, I = [0, 1]. В отличие от традиционных подмножеств точки, изображающие нечёткие подмножества, могут находиться не только на вершинах гиперкуба, но и внутри него (см. рисунок 11.1).

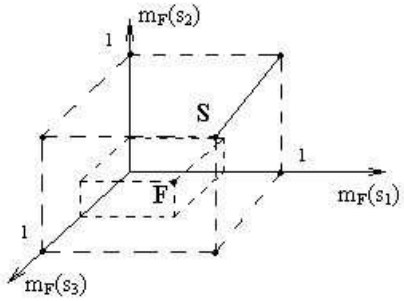


Рисунок 11.1 – Графическое представление функции нечёткого множества.

Рассмотрим пример определения нечёткого подмножества. Имеется множество всех людей S. Определим нечёткое подмножество T всех высоких людей этого множества. Введём для каждого человека степень его принадлежности к множеству подмножеству T. Для этого зададим функцию принадлежности $m_T(h)$, определяющую, в какой степени можно считать высоким человека ростом h сантиметров.

$$m_T(h) = \begin{cases} 0, h < 150 \\ \dfrac{h-150}{60}, 150 \le h \le 210 \\ 1, h > 210 \end{cases}$$

где h – рост конкретного человека в сантиметрах. График этой функции представлен на рисунке 11.2.

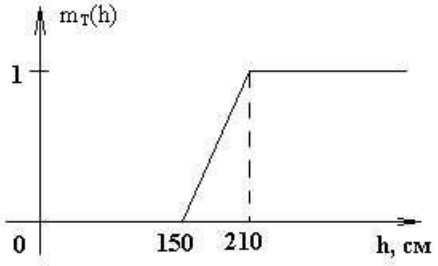


Рисунок 11.2 – График функции принадлежности $m_T(h)$.

Пусть рост Михаила – 163 см, тогда истинность высказывания «Михаил высок» будет равна 0,21. Использованная в данном случае функция принадлежности тривиальна. При решении большинства реальных задач подобные

функции имеют более сложный вид, кроме того, число их аргументов может быть большим.

Методы построения функций принадлежности для нечётких подмножеств довольно разнообразны. В большинстве случаев они отражают субъективные представления экспертов о предметной области. Так, например, кому-то человек ростом 180 см может показаться высоким, а кому-то – нет. Однако часто такая субъективность помогает снизить степень неопределённости при решении слабо формализованных задач. Как правило, для задания функций принадлежности используются типовые зависимости, параметры которых определяются путём обработки мнений экспертов. Представление произвольных функций при реализации автоматизированных систем часто затруднено, поэтому в реальных разработках такие зависимости аппроксимируются кусочно-линейными функциями.

Необходимо осознавать разницу между нечёткой логикой и теорией вероятностей. Заключается она в различии понятий вероятности и степени принадлежности. Вероятность определяет, насколько возможен один из нескольких взаимоисключающих исходов или одно из множества значений. Например, может определиться вероятность того, что утверждение истинно. Утверждение может быть либо истинным, либо ложным. Степень принадлежности показывает, насколько то или иное значение принадлежит определённому классу (подмножеству). Например, при определении истинности утверждения её возможные значения не ограничены «ложью» или «истиной», а могут попадать в промежуток между ними. Ещё одно различие выражено в математических свойствах этих понятий. В отличие от вероятности для степени принадлежности не требуется выполнение аксиомы аддитивности.

Описанные выше положения можно применять для логического вывода утверждений. Например, если известно, что $x \in A$ и $A \subseteq B$, то из аксиомы вывода имеем: $x \in B$. Пусть A – множество всех народных депутатов, а B – множество тех, кто пользуется правом бесплатного проезда в общественном транспорте. Тогда утверждение о вхождении A в B трансформируется в правило вывода: «Если лицо является народным депутатом, то оно пользуется правом бесплатного проезда в общественном транспорте». Если множества не сравнимы непосредственно, может потребоваться дополнительное функциональное преобразование, которое позволит рассматривать одно множество как подмножество другого. Например, результатом преобразования посылки «температура в комнате 30° С» для кондиционера может служить указание «включить вентилятор». Все значения температур, при которых необходимо его включение, образуют подмножество во множестве условий, приводящих к включению вентилятора. Преобразование производится функцией управления, роль которой в данном случае может выполнять термостат.

Нечёткое правило логического вывода представляет собой упорядоченную пару (A, B), где A – нечёткое подмножество пространства входных значений X, B – нечёткое подмножество пространства входных значений Y. Система нечёткого вывода – это отображение $I^{разм(X)}$ в $I^{разм(Y)}$, где разм(Z) – оператор определения размерности пространства Z. Число элементов $I^{разм(X)}$ и $I^{разм(Y)}$ бесконечно велико, поэтому невозможно задать правила нечёткого вывода соответствующими парами точек. Однако они могут быть описаны в терминах теории нечётких множеств. Например, вполне работоспособная система кондиционирования может быть описана правилами: «если температура в комнате высокая, то скорость вращения вентилятора высокая» и «если температура в комнате низкая, то скорость вращения вентилятора низкая».

11.2 Операции над нечёткими множествами

Правила в теории нечётких множеств как правило имеют вид:

Если утверждение1 истинно, то утверждение2 истинно.

Например: «*если цена мала, то спрос велик*».

Так как значение истинности утверждения «цена мала» не обязательно целое число, то необходимо провести оценку истинности утверждения «спрос велик», то есть произвести суперпозицию левой и правой части правила. Для этого можно воспользоваться одним методов: методом «минимума» или методом «произведения».

В первом методе значение истинности левой части используется как предел, выше которого не могут быть значения функции принадлежности (см. Рисунок 11.3).

Рисунок 11.3 – Метод «минимума»

Во втором методе значение истинности левой части используется как коэффициент, на который умножаются значения функции принадлежности (см. рисунок 11.4).

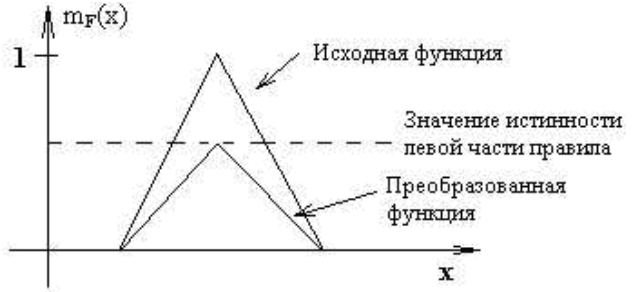


Рисунок 11.4 – Метод «произведения»

Результат выполнения правила – нечёткое множество. Говоря более строго, происходит ассоциирование переменной и функции принадлежности, указанных в правой части.

Выводы всех правил вычисляются нечёткой экспертной системой отдельно, однако в правой части нескольких из них может быть указана одна и та же нечёткая переменная. Как было сказано выше, при определении обобщённого результата необходимо учитывать все правила. Для этого система производит суперпозицию нечётких множеств, связанных с каждой из таких переменных. Эта операция называется нечётким объединением правил вывода. Например, правая часть правил

$$\begin{cases} \textit{если цена мала, то спрос велик} \\ \textit{если цена велика, то спрос мал} \end{cases}$$

содержит одну и ту же переменную – спрос. Два нечётких подмножества, получаемые при выполнении этих правил, должны быть объединены экспертной системой.

Традиционно суперпозиция функций принадлежности нечётких множеств $m_{1F}(x)$, $m_{2F}(x)$, … , $m_{nF}(x)$ определяется как:

$M_{sum\,F}(x) = \max \{ m_{iF}(x) \}\ \forall x,\ i \in [1, n]$.

Графическое представление подобной суперпозиции приведено на рисунке 11.5.

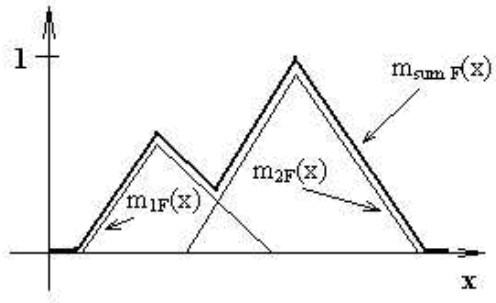


Рисунок 11.5 – Метод «Max Combination»

Другой метод суперпозиции состоит в суммировании значений всех функций принадлежности (графическая интерпретация приведена на рисунке 11.6)

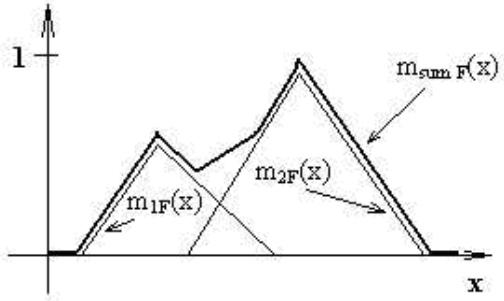


Рисунок 11.6 – Метод «Sum Combination»

Самым простым (но и наименее часто используемым) является подход, когда суперпозиция не производится. Выбирается одно из правил вывода, результат которого используется в качестве интегрального результата.

Конечный этап обработки базы правил вывода – переход от нечётких значений к конкретным скалярным. Процесс преобразования нечёткого множества в единственное значение называется «скаляризацией» или «дефазификацией» (defuzzification). Чаще всего в качестве такого значения используется «центр тяжести» функции принадлежности нечёткого множества (centroid defuzzification method) (см. Рисунок 11.7).

Рисунок 11.7 – Скаляризация методом «центра тяжести»

Другой пространственный подход – использование максимального значения функции принадлежности (modal defuzzification method) (см. Рисунок 11.8). Конкретный выбор методов суперпозиции и скаляризации осуществляется в зависимости от желаемого поведения нечёткой экспертной системы.

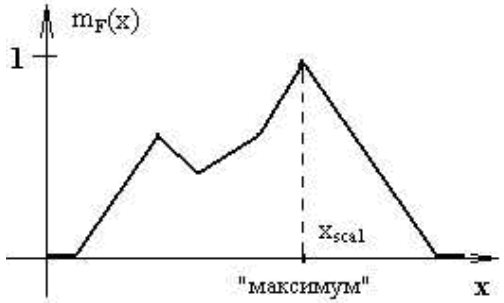


Рисунок 11.8 – Скаляризация методом «максимума»

11.3 Пример использования нечёткие правила вывода в экспертной системе

Рассмотрим пример того, как обрабатываются нечёткие правила вывода в экспертной системе, управляющей вентилятором комнатного кондиционера. Задача кондиционера – поддерживать оптимальную температуру воздуха в комнате, охлаждая его, когда жарко, и нагревая, когда холодно. Пусть, изменяя скорость вращения вентилятора, прогоняющего воздух через охлаждающий элемент, мы можем менять температуру воздуха, тогда алгоритм работы кондиционера может быть задан следующими правилами:

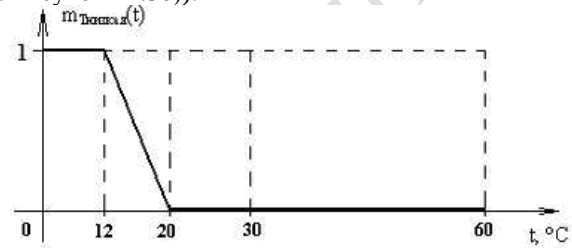
- если температура воздуха в комнате высокая, то скорость вращения вентилятора высокая;

- если температура воздуха в комнате средняя, то скорость вращения вентилятора средняя;
- если температура воздуха в комнате низкая, то скорость вращения вентилятора низкая.

Для того, чтобы система могла обрабатывать эти правила, надо задать функции принадлежности для нечётких подмножеств, определённых на значениях температуры (t) и скорости вращения (v). Пусть температура воздуха в комнате находится в пределах от 0°C до 60°C – в противном случае кондиционер вряд ли поможет. Функцию принадлежности для нечёткого подмножества низкая, определённую на интервале изменения температуры, можно задать, например, так (см. Рисунок 11.9б)):

а)

$$m_{Тнизка}(t) = \begin{cases} 1, t \le 12 \\ \dfrac{20-t}{8}, 12 < t < 20 \\ 0, t \ge 20 \end{cases}$$



б)

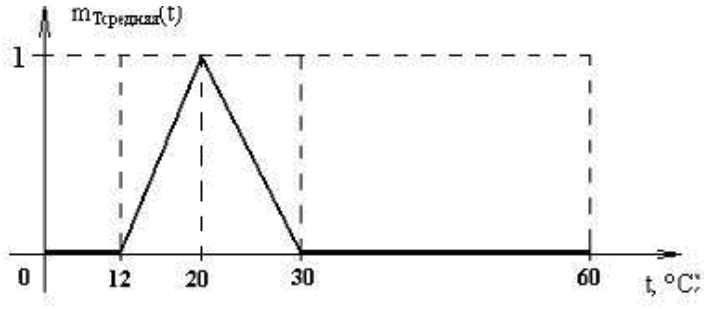
Рисунок 11.9 – Нечёткое подмножество «низкая», определённое на множестве значений температуры

Если температура меньше 12°C, то это – определённо низкая температура для комнаты ($m_{Тнизкая}(t) = 1, t \le 12$). Температуру выше 20°C никак нельзя назвать низкой ($m_{Тнизкая}(t) = 0, t \ge 20$). В промежутке между этими значениями функция принадлежности линейно убывает – с увеличением температуры уменьшается истинность утверждения «температура воздуха в комнате низкая». Аналитическое определение $m_{Тнизкая}(t)$ приведено на рисунке 11.9а)

Сходные рассуждения позволяют задать функции принадлежности для оставшихся подмножеств: средняя и высокая (см. рисунки 11.10-11).

$$m_{Тсредня}(t)=\begin{cases}0, t\le 12\wedge t\ge 30\\ \dfrac{-12+t}{8}, 12<t\le 20\\ \dfrac{30-t}{10}, 20<t<30\end{cases}$$

а)

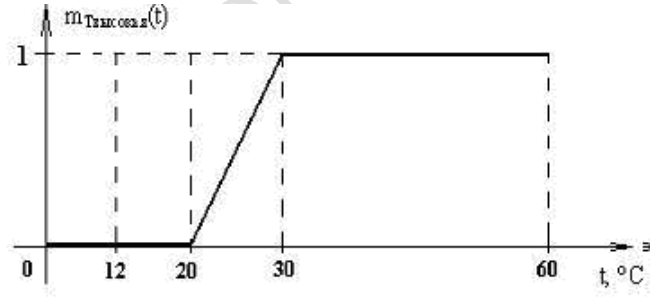


б)

Рисунок 11.10 – Нечёткое подмножество «средняя», определённое на множестве значений температуры

$$m_{Тввысока}(t)=\begin{cases}0, t\le 20\\ \dfrac{-20+t}{10}, 20<t<\\ 1, t\ge 30\end{cases}$$

а)



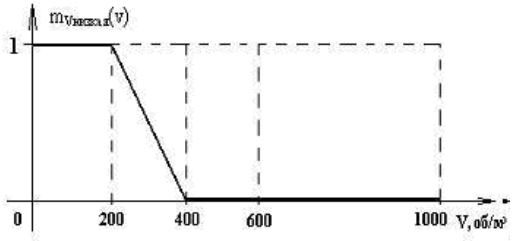
б)

Рисунок 11.11 – Нечёткое подмножество «высокая», определённое на множестве значений температуры

Аналогично определяем подмножества «низкая», «средняя» и «высокая» для множества значений скорости вращения вентилятора.

$$m_{Vннизка}(t)=\begin{cases}1, v\le 200\\ \dfrac{400-v}{200}, 200<v<400\\ 0, v\ge 400\end{cases}$$

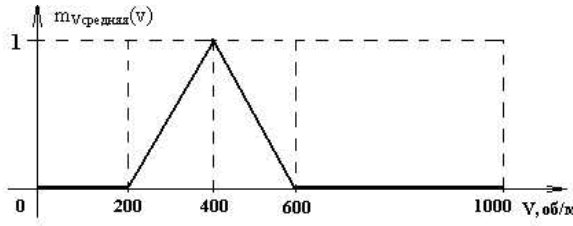
а)



б)

Рисунок 11.12 – Нечёткое подмножество «низкая», определённое на множестве значений скорости вращения вентилятора

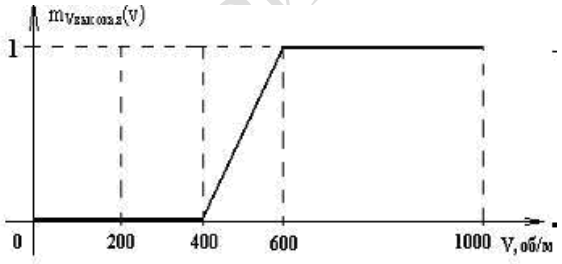
$$m_{Vсредня}(t) = \begin{cases} 0, v \le 200 \text{ или } v \ge 600 \\ \dfrac{-200+v}{200}, 200 < v \le 400 \\ \dfrac{600-v}{200}, 400 < v < 600 \end{cases}$$



а) б)

Рисунок 11.13 – Нечёткое подмножество «средняя», определённое на множестве значений скорости вращения вентилятора

$$m_{Vвысока}(t) = \begin{cases} 0, v \le 400 \\ \dfrac{-400+v}{200}, 400 < v < 600 \\ 1, v \ge 600 \end{cases}$$



а) б)

Рисунок 11.14 – Нечёткое подмножество «высокая», определённое на множестве значений скорости вращения вентилятора.

Рассмотрим теперь, как нечёткая экспертная система определяет скорость вращения вентилятора в зависимости от температуры воздуха в комнате. Пусть эта температура равна 22°С.

Сначала экспертной системе надо определить истинность левых частей правил вывода при подстановке в них текущего значения температуры. Для этого она должна найти степень вхождения t = 22°С в каждое из указанных слева нечётких подмножеств. В левых частях правил указаны три подмножества, заданных на интервале значений температуры: высокая, низкая и средняя. Степень вхождения находим, вычисляя значение функции принадлежности каждого из подмножеств от t = 22°С:

$m_{Тнизкая}(22) = 0$

$m_{Тсредняя}(22) = 0,8$

$m_{Твысокая}(22) = 0,2$

Значения истинности левой части каждого правила используются для модификации нечёткого множества, указанного в его правой части. Модификацию будем производить указанным выше методом «произведения». На ри-

сунке 11.15 изображено, как трансформируются находящиеся в правых частях правил нечёткие подмножества высокая, средняя и низкая.

Далее нечёткой экспертной системе необходимо обобщить результаты действия всех правил вывода, то есть произвести суперпозицию полученных нечётких множеств. Воспользуемся для этого методом «Max Combination» (см. Рисунок 11.5).

Теперь необходимо осуществить переход от суперпозиции множеств к скалярному значению. Скаляризацию произведём методом «центра тяжести». Рисунок 11.16 иллюстрирует результат.

Практика подтверждает, что такая несложная система кондиционирования обеспечивает наименьшие колебания температуры по сравнению с аналогами.

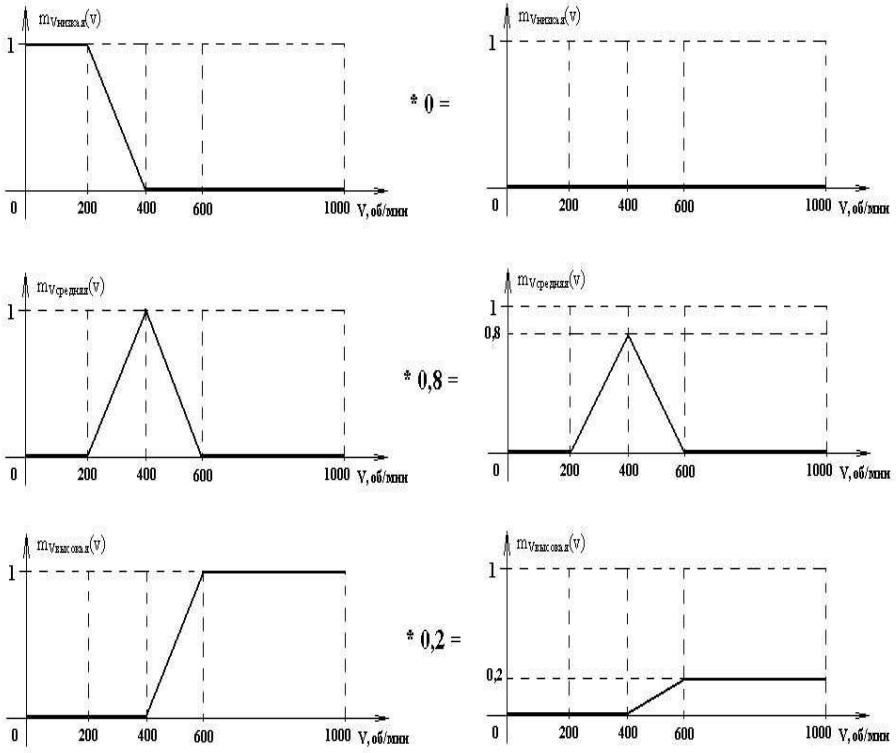


Рисунок 11.15 – Модификация нечётких подмножеств, определённых на интервале изменения скорости вращения вентилятора.

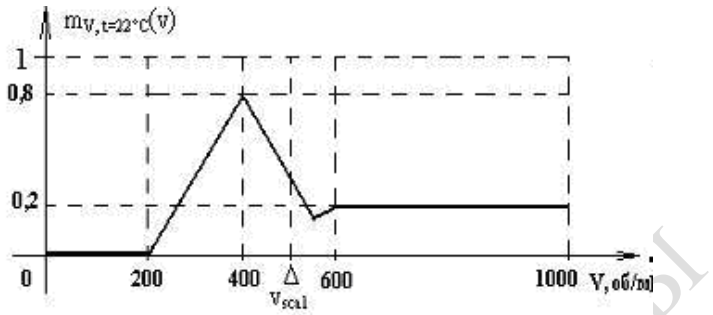


Рисунок 11.16 – Результат выполнения суперпозиции и скаляризации.

11.4 Программное обеспечение на основе нечеткой логики

Язык нечётких множеств и алгоритмов в настоящее время наиболее адекватный математический аппарат, который позволяет максимально сократить переход от вербального словесного качественного описания объекта, которое характеризует человеческое мышление, к численным количественным оценкам его состояния и сформулировать на этой основе простые и эффективные алгоритмы, то есть позволяет моделировать человеческие размышления и человеческую способность решения задач.

Нечеткая логика нашла свое применение в экспертных системах, системах управления, промышленной диагностике, медицине и др

Помимо своего основного преимущества – лучшей адаптированности к условиям реального мира, обладают еще двумя достоинствами по сравнению с традиционными. Во-первых, они свободны от так называемых <циклических блокировок> при построении заключений. Во-вторых, различные базы нечетких правил можно с легкостью объединять, что редко удается в обычных экспертных системах.

Для решения сложнейших задач прогнозирования различных финансовых индикаторов банкиры и финансисты используют дорогостоящие комплексные системы, в состав которых входит и нечеткая логика. Начало этому процессу положила японская финансовая корпорация Yamaichi Securuties. Задавшись целью автоматизировать игру на рынке ценных бумаг, эта компания привлекла к работе около 30 специалистов по искусственному интеллекту. В первую версию системы, завершенную к началу 1990 года, вошли 600 нечетких правил – воплощение опыта десяти ведущих брокеров корпорации. Прежде чем решиться на использование новой системы в реальных условиях, ее протестировали на двухлетней выборке финансовых данных (1987-1989 г). Система с блеском выдержала испытание. Особое изумление экзаменаторов вызвало то, что за неделю до наступления биржевого краха (знаменитого <Черного Понедельника> на токийской бирже в 1988 году) система распродала весь пакет акций, что свело ущерб практически к нулю. Надо ли говорить, что после этого вопрос о целесообразности применения нечеткой логики в финансовой сфере уже не поднимался. Хотя скептики могут привести и другие примеры – например, ни одна из банковских систем не смогла предсказать падение биржевого индекса Nikkei весной 1992 года.

Можно привести и другие примеры применения нечеткой логики в бизнесе. Удачный опыт Ганса Зиммермана (Hans Zimmermann) по использованию экспертной

системы с нечеткими правилами для анализа инвестиционной активности в городе Аахене (ФРГ) привел к созданию коммерческого пакета ASK для оценки кредитных и инвестиционных рисков. А система управления складскими запасами, описанная в качестве примера в пакете CubiCalc, настолько проста, что может быть с легкостью использована самым неподготовленным оптовым торговцем.

Российский рынок коммерческих систем на основе нечеткой логики начал формироваться в середине 1995 года. Наиболее популярны у российских заказчиков следующие пакеты :

CubiCalc 2.0 RTC – одна из наиболее мощных коммерческих экспертных систем на основе нечеткой логики, позволяющая создавать собственные прикладные экспертные системы ;

CubiQuick – дешевая <университетская> версия пакета CubiCalc ;

RuleMaker – программа автоматического извлечения нечетких правил из входных данных ;

FuziCalc – электронная таблица с нечеткими полями, позволяющая делать быстрые оценки при неточно известных данных без накопления ошибки ;

OWL – пакет, содержащий исходные тексты всех известных видов нейронных сетей, нечеткой ассоциативной памяти и т.д.

Основными <потребителями> нечеткой логики на рынке России являются банкиры и финансисты, а также специалисты в области политического и экономического анализа. Они используют CubiCalc для создания моделей различных экономических, политических, биржевых ситуаций. Что же касается изумительно легкого в освоении пакета FuziCalc, то он занял свое место на компьютерах крупных банкиров и специалистов по чрезвычайным ситуациям – т.е. тех, для кого более всего важна быстрота проведения расчетов в условиях неполноты и неточности входной информации. Однако можно с уверенностью сказать, что эпоха расцвета прикладного использования нечеткой логики на отечественном рынке еще впереди.

Тема 12 Генетические алгоритмы

12.1 Эволюционные вычисления и традиционные методы оптимизации

12.2 Основы теории генетических алгоритмов

12.3 Примеры решения оптимизационных задач с использованием генетических алгоритмов

12.4 Примеры программного обеспечения

12.1 Эволюционные вычисления и традиционные методы оптимизации

Принятие решения в большинстве случаев заключается в генерации всех возможных альтернатив решений, их оценке и выборе лучшей среди них. Для генерации и выбора альтернатив решений стали реализовываться новые задачи формализации и алгоритмизации процесса принятия решений, кото-

рые сейчас составляют отдельную большую область исследований. Формализация задачи предполагает описание всех факторов, влияющих на достижение цели, их взаимодействия, ограничительных условий и критерия оценки качества принимаемого решения, на основе которого можно осуществлять выбор между альтернативами. Обычно в качестве критерия оценки выступает некая целевая функция, аргументами которой являются количественные характеристики, описывающие состояние факторов, влияющих на достижение цели в решаемой задаче. При этом решению, приводящему к наилучшему результату, как правило, соответствует экстремальное значение целевой функции, то есть точка ее максимума или минимума.

Таким образом, процесс генерации вариантов решений и выбора наилучшей из полученных альтернатив сводится, в общем случае, к созданию всех возможных комбинаций значений характеристик, влияющих на целевую функцию, и нахождение такой комбинации, которая приводит к ее экстремальному значению. Все возможные комбинации аргументов при этом образуют пространство поиска задачи, размерность которого определяется числом аргументов целевой функции. А каждая из указанных комбинаций образует точку в данном пространстве.

При принятии решений автоматизирован наиболее трудоемкий процесс: генерации альтернатив решений и выбора наилучшего из них.

Среди формальных математических методов поиска оптимальных решений (экстремума целевой функции), не перебирающих все возможные комбинации ее аргументов можно выделить три основных типа:

- методы, основанные на математических вычислениях,
- перечислительные методы,
- методы, использующие элемент случайности.

Методы, основанные на математических вычислениях, изучены наиболее полно. Они подразделяются на:

- направленные методы, реализующие перемещение в допустимой области, на которое указывает градиент (например, метод касательных);
- ненаправленные методы поиска локального экстремума путем решения системы уравнений, построенной путем приравнивания градиента целевой функции к нулю (например, метод градиентного спуска или покоординатного спуска).

Эти методы находят лишь локальные экстремумы целевой функции и применимы лишь в случаях гладких, всюду дифференцируемых унимодальных (имеющих один экстремум на пространстве поиска) целевых функций.

Перечислительные методы также изучены достаточно подробно и имеют множество видов и форм. Их основная идея состоит в том, что пространство поиска любой задачи можно представить в виде совокупности дискретных точек. Даже если пространство поиска непрерывно, то конечная точность представления чисел в вычислительных машинах позволяет сделать такое

допущение. В этом случае поиск решения будет сводиться к перебору всех точек пространства поиска и вычислению в них целевой функции, в одной из которых она, несомненно, примет экстремальное значение. Для реализации непосредственно процесса поиска сейчас разработано большое количество соответствующих алгоритмов.

При увеличении размерности пространства поиска (числа аргументов целевой функции) количество точек пространства значительно увеличивается. В то же время размерность решаемых сейчас задач постоянно растет, а время, доступное для принятия решений сокращается. Таким образом, перечислительные методы также применимы для решения все более сужающегося класса задач.

Методы, использующие элементы случайности, стали появляться по мере того как становились очевидными недостатки методов первых двух видов. В основе первого из таких методов лежит случайный поиск в пространстве задачи с сохранением наилучшего полученного результата. Очевидно, что применение такого метода не гарантирует получения оптимального решения.

При решении очень сложных задач основной целью является поиск уже не оптимального, а более «хорошего» решения по сравнению с полученным ранее или заданным в качестве начального. Здесь методы, использующие элемент случайности, получают определенное преимущество перед остальными. Однако даже с такими допущениями непосредственный случайный поиск является малоэффективным. Исследования показали, что внесение в такие методы элементов детерминированности дает значительное улучшение показателей.

Одним из типов таких «частично» случайных методов являются эволюционные вычисления. ***Эволюционные вычисления*** – термин, обычно используемый для общего описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формализованных принципах естественного эволюционного процесса. Основное преимущество эволюционных вычислений в этой области заключается в возможности решения многомодальных (имеющих несколько локальных экстремумов) задач с большой размерностью за счет сочетания элементов случайности и детерминированности точно так, как это происходит в природной среде. Детерминированность этих методов заключается в моделировании природных процессов отбора, размножения и наследования, происходящих по строго определенным правилам. Основным правилом при этом является закон эволюции: «выживает сильнейший», который обеспечивает улучшение искомого решения. Другим важным фактором эффективности эволюционных вычислений является моделирование размножения и наследования. Рассматриваемые варианты решений могут по определенному правилу порождать новые решения, которые будут наследовать лучшие черты своих «предков».

В качестве случайного элемента в методах эволюционных вычислений может использоваться, например, моделирование процесса мутации. В этом случае характеристики того или иного решения могут быть случайно изменены, что приведет к новому направлению в процессе эволюции решений и может ускорить процесс выработки лучшего решения.

История эволюционных вычислений началась с разработки ряда различных независимых моделей эволюционного процесса. Среди этих моделей можно выделить три основные парадигмы:

- генетические алгоритмы;
- эволюционные стратегии;
- эволюционное программирование.

Основное отличие генетических алгоритмов заключается в представлении любой альтернативы решения в виде битовой строки фиксированной длины, манипуляции с которой производятся без связи со смысловой интерпретацией. То есть в данном случае применяется единое универсальное представление любой задачи. Парадигму генетических алгоритмов предложил Джон Холланд, общепризнанной после выхода в свет в 1975 году его классического труда «Адаптация в естественных и искусственных системах».

Эволюционные стратегии, напротив, оперируют объектами, тесно связанными с решаемой задачей. Каждая из альтернатив решения представляется единым массивом численных параметров, за каждым из которых скрывается, по сути, аргумент целевой функции. Воздействие на данные массивы осуществляется, в отличие от генетических алгоритмов, с учетом их смыслового содержания и направлено на улучшение значений входящих в них параметров. Парадигму эволюционных стратегий предложили в 1973 году Реченберг (I. Rechenberg) в своей работе «Эволюционные стратегии: оптимизация технических систем на основе принципов биологической эволюции» и в 1977 году Щефель (H.-P. Schwefel) в работе «Численная оптимизация компьютерных моделей посредством эволюционной стратегии».

В основе направления эволюционного программирования лежит идея представления альтернатив в виде универсальных конечных автоматов, способных реагировать на стимулы, поступающие из окружающей среды. Соответствующим образом разрабатывались и операторы воздействия на них. Идеи эволюционного программирования были предложены в 1966 году Фогелем, Оуэнсом и Уолшем (L.J. Fogel, A.J. Owens, M.J. Walsh) в работе «Построение систем искусственного интеллекта путем моделирования эволюции».

Как и всякий метод, использующий элемент случайности, эволюционные вычисления не гарантируют обнаружения глобального экстремума целевой функции (или оптимального решения) за определенное время. Основное их преимущество в том, что они позволяют найти более «хорошие» ре-

шения очень трудных задач за меньшее время, чем другие методы. Естественно, эволюционные вычисления не являются оптимальным средством для решения любых задач, поскольку было доказано, что не существует метода поиска, который был бы наилучшим во всех случаях. Тем не менее, методы эволюционных вычислений оказались достаточно эффективными для решения ряда реальных задач инженерного проектирования, планирования, маршрутизации и размещения, управления портфелями ценных бумаг, прогнозирования, а также во многих других областях.

Отрицательной чертой эволюционных вычислений является то, что они представляют собой, скорее, подход к решению задач оптимизации, чем алгоритм, поэтому требуют адаптации к каждому конкретному классу задач путем выбора определенных характеристик и параметров.

В настоящее время наблюдается взаимное проникновение указанных парадигм (генетические алгоритмы; эволюционные стратегии; эволюционное программирование) и их сращивание в единую концепцию эволюционных вычислений.

12.2 Основы теории генетических алгоритмов

Генетические алгоритмы представляют собой алгоритмы поиска, построенные на принципах, сходных с принципами естественного отбора и генетики: выживания наиболее перспективных особей – решений и структуризированный обмен информацией, в котором присутствует элемент случайности, который моделирует природные процессы наследования и мутации. Опосредованное вмешательство человека в развивающийся процесс поиска осуществляется через задание исходных параметров.

Будучи разновидностью методов поиска с элементами случайности, генетические алгоритмы позволяют найти лучшее, а не оптимального решения задачи. Рассмотрим отличия генетических алгоритмов от традиционных методов оптимизации.

Первое. Генетические алгоритмы работают с кодами, в которых представлен набор параметров, напрямую зависящих от аргументов целевой функции. Причем интерпретация этих кодов происходит только перед началом работы алгоритма и после завершения его работы для получения результата. В процессе работы манипуляции с кодами происходят совершенно независимо от их интерпретации, код рассматривается просто как битовая строка.

Второе. Для поиска генетический алгоритм использует несколько точек поискового пространства одновременно, а не переходит от точки к точке, как это делается в традиционных методах. Это позволяет преодолеть один из их недостатков – опасность попадания в локальный экстремум целевой функции, если она не является унимодальной, то есть имеет несколько таких

экстремумов. Использование нескольких точек одновременно значительно снижает такую возможность.

Третье. Генетические алгоритмы в процессе работы не используют никакой дополнительной информации, что повышает скорость работы. Единственной используемой информацией может быть область допустимых значений параметров и целевой функции в произвольной точке.

Четвертое. Генетический алгоритм использует как вероятностные правила для порождения новых точек анализа, так и детерминированные правила для перехода от одних точек к другим. Выше уже говорилось, что одновременное использование элементов случайности и детерминированности дает значительно больший эффект, чем раздельное.

Прежде чем рассматривать непосредственно работу генетического алгоритма, введем ряд терминов, которые широко используются в данной области.

Выше было показано, что генетический алгоритм работает с кодами безотносительно их смысловой интерпретации. Поэтому сам код и его структура описываются понятием генотип, а его интерпретация, с точки зрения решаемой задачи, понятием фенотип. Каждый код представляет, по сути, точку пространства поиска. С целью максимально приблизиться к биологическим терминам, экземпляр кода называют хромосомой, особью или индивидуумом. Далее для обозначения строки кода мы будем в основном использовать термин «особь».

На каждом шаге работы генетический алгоритм использует несколько точек поиска одновременно. Совокупность этих точек является набором особей, который называется популяцией. Количество особей в популяции называют размером популяции. Для классических генетических алгоритмов размер популяции является фиксированным и представляет одну из характеристик генетического алгоритма. На каждом шаге работы генетический алгоритм обновляет популяцию путем создания новых особей и уничтожения старых. Чтобы отличать популяции на каждом из шагов и сами эти шаги, их называют поколениями и обычно идентифицируют по номеру. Например, популяция, полученная из исходной популяции после первого шага работы алгоритма, будет первым поколением, после следующего шага – вторым, и т.д.

В процессе работы алгоритма генерация новых особей происходит на основе моделирования процесса размножения. При этом, естественно, порождающие особи называются родителями, а порожденные – потомками. Родительская пара, как правило, порождает пару потомков. Непосредственная генерация новых кодовых строк из двух выбранных происходит за счет работы оператора скрещивания, который также называют кроссинговером (от англ. crossover). При порождении новой популяции оператор скрещивания может применяться не ко всем парам родителей. Часть этих пар может

переходить в популяцию следующего поколения непосредственно. Насколько часто будет возникать такая ситуация, зависит от значения вероятности применения оператора скрещивания, которая является одним из параметров генетического алгоритма.

Моделирование процесса мутации новых особей осуществляется за счет работы оператора мутации. Основным параметром оператора мутации также является вероятность мутации.

Поскольку размер популяции фиксирован, то порождение потомков должно сопровождаться уничтожением других особей. Выбор пар родителей из популяции для порождения потомков производит оператор отбора, а выбор особей для уничтожения – оператор редукции. Основным параметром их работы является, как правило, качество особи, которое определяется значением целевой функции в точке пространства поиска, описываемой этой особью.

Таким образом, можно перечислить ***основные понятия и термины***, используемые в области генетических алгоритмов: генотип и фенотип, особь и качество особи, популяция и размер популяции, поколение, родители и потомки. К ***характеристикам*** генетического алгоритма относятся: размер популяции, оператор скрещивания и вероятность его использования, оператор мутации и вероятность мутации, оператор отбора, оператор редукции, критерий останова. Операторы отбора, скрещивания, мутации и редукции называют еще генетическими операторами.

Критерием останова работы генетического алгоритма может быть одно из трех событий: сформировано заданное пользователем число поколений; популяция достигла заданного пользователем качества (например, значение качества всех особей превысило заданный порог); достигнут некоторый уровень сходимости (высокая похожесть особей привела к чрезвычайно медленному улучшению).

Характеристики генетического алгоритма выбираются таким образом, чтобы обеспечить малое время работы, с одной стороны, и поиск как можно лучшего решения, с другой.

Схематично процесс работы генетического алгоритма можно описать следующей последовательностью действий:

1) Создание исходной популяции.
2) Выбор родителей для процесса размножения (работает оператор отбора).
3) Создание потомков выбранных пар родителей (работает оператор скрещивания).
4) Мутация новых особей (работает оператор мутации).
5) Расширение популяции за счет добавления новых, только что порожденных, особей.

6) Сокращение расширенной популяции до исходного размера (работает оператор редукции).

7) Критерий останова работы алгоритма выполнен? Если нет, то на 2.

8) Поиск лучшей достигнутой особи в конечной популяции – результата работы алгоритма.

Формирование исходной популяции происходит, как правило, с использованием какого-либо случайного закона, на основе которого выбирается нужное количество точек поискового пространства. Исходная популяция может также быть результатом работы какого-либо другого алгоритма оптимизации. Все здесь зависит от разработчика конкретного генетического алгоритма.

В основе ***оператора отбора***, который служит для выбора родительских пар и уничтожения особей, лежит принцип «выживает сильнейший». В качестве примера можно привести следующий оператор. Выбор особи для размножения производится случайно. Вероятность участия особи в процессе размножения вычисляется по формуле:

$$P_i = f_i / \sum_{j=1}^{n} f_j ,$$

где n - размер популяции, i – номер особи, P_i – вероятность участия особи в процессе размножения, f_i – значение целевой функции для i-й особи. Очевидно, что одна особь может быть задействована в нескольких родительских парах.

Аналогично может быть решен вопрос уничтожения особей. Только вероятность уничтожения, соответственно, должна быть обратно пропорциональна количеству особей. Однако обычно происходит просто уничтожение особей с наихудшим качеством. Таким образом, выбирая для размножения наиболее качественные особи и уничтожая наиболее слабые, генетический алгоритм постоянно улучшает популяцию, ведя к нахождению все лучших решений.

Оператор скрещивания призван моделировать природный процесс наследования, то есть обеспечивать передачу свойств родителей потомкам. Опишем простейший ***оператор скрещивания***. Он выполняется в два этапа.

Пусть особь представляет собой строку из n элементов. На первом этапе равновероятно выбирается натуральное число k от 1 до $n-1$. Это число называется точкой разбиения. В соответствии с ним обе исходные строки разбиваются на две подстроки. На втором этапе строки обмениваются своими подстроками, лежащими после точки разбиения, то есть элементами с $k+1$-го по n-й. Так получаются две новые строки, которые наследовали частично свойства обоих родителей. Этот процесс проиллюстрирован ниже.

```
Строка1 X1X2...XkXk+1...Xn X1X2...XkYk+1...Yn

Строка2 Y1Y2...YkYk+1...Yn Y1Y2...YkXk+1...Xn
```

Вероятность применения оператора скрещивания обычно выбирается достаточно большой, в пределах от 0,9 до 1, чтобы обеспечить постоянное появление новых особей, расширяющих пространство поиска. При значении вероятности меньше 1 часто используют элитизм. Это особая стратегия, которая предполагает переход в популяцию следующего поколения элиты, то есть лучших особей текущей популяции, без всяких изменений. Применение элитизма способствует сохранению общего качества популяции на высоком уровне. При этом элитные особи участвуют еще и в процессе отбора родителей для последующего скрещивания. Количество элитных особей определяется обычно по формуле: $K=(1-P)*N$, где K – количество элитных особей, P – вероятность применения оператора скрещивания, N – размер популяции. В случае использования элитизма все выбранные родительские пары подвергаются скрещиванию, несмотря на то, что вероятность применения оператора скрещивания меньше 1. Это позволяет сохранять размер популяции постоянным.

Оператор мутации служит для моделирования природного процесса мутации. Его применение в генетических алгоритмах обусловлено следующими соображениями. Исходная популяция, какой бы большой она ни была, охватывает ограниченную область пространства поиска. Оператор скрещивания, безусловно, расширяет эту область, но все же до определенной степени, поскольку использует ограниченный набор значений, заданный исходной популяцией. Внесение случайных изменений в особи позволяет преодолеть это ограничение и иногда значительно сократить время поиска или улучшить качество результата. Как правило, вероятность мутации, в отличие от вероятности скрещивания, выбирается достаточно малой. Сам процесс мутации заключается в замене одного из элементов строки на другое значение. Это может быть перестановка двух элементов в строке, замена элемента строки значением элемента из другой строки, в случае битовой строки может применяться инверсия одного из битов и т.д.

В процессе работы алгоритма все указанные выше операторы применяются многократно и ведут к постепенному изменению исходной популяции. Поскольку операторы отбора, скрещивания, мутации и редукции по своей сути направлены на улучшение каждой отдельной особи, то результатом их работы является постепенное улучшение популяции. В этом и заключается основной смысл работы генетического алгоритма – улучшить популяцию решений по сравнению с исходной.

После завершения работы генетического алгоритма из конечной популяции выбирается та особь, которая дает максимальное (или минимальное)

значение целевой функции и является, таким образом, результатом работы генетического алгоритма. За счет того, что конечная популяция лучше исходной, полученный результат представляет собой улучшенное решение.

12.3 Примеры решения оптимизационных задач с использованием генетических алгоритмов

Пример 1. Поиск максимума одномерной функции

Пусть имеется набор натуральных чисел от 0 до 31 и функция, определенная на этом наборе чисел, $f(x) = x$. Требуется найти максимальное значение функции. Задача, конечно, тривиальна и не требует применения столь изощренных методов поиска, но мы решаем ее лишь для иллюстрации функционирования генетического алгоритма.

В качестве кода будем использовать 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.

Определим некоторые характеристики генетического алгоритма. Пусть размер популяции будет 4, вероятность мутации 0,001, сам процесс мутации заключается в инверсии одного из битов строки, выбираемого случайно по равномерному закону. Оператор скрещивания и отбора аналогичны описанным выше. Поскольку задача является простейшей, будем считать, что алгоритм не использует элитизм.

Пусть на основе равномерного распределения создана исходная популяция из четырех особей, представленная в таблице 12.1.

Таблица 12.1

№ строки	Код	Значение целевой функции	Вероятность участия в процессе размножения
1	01011	11	11/43
2	10010	18	18/43
3	00010	2	2/43
4	01100	12	12/43

Предположим, что оператор отбора выбрал для производства потомков две пары строк (1,2) и (2,4). Работа оператора скрещивания проиллюстрирована в таблице 12.2. При этом в каждой паре разбиение на подстроки происходит независимо.

Таблица 12.2

№ строки	Родители	Потомки	Значение целевой функции для потом-

			ков
1	0 | 1011	00010	2
2	1 | 0010	11010	27
2	100 | 10	10000	16
4	011 | 00	01110	14

Пусть оператор мутации, несмотря на низкую вероятность, сработал для младшего бита потомка в строке 3, и данный код изменил свое значение с 10000 на 10001.

Таким образом, популяция за счет порожденных потомков расширилась до восьми особей, представленных в таблице 12.3.

Таблица 12.3.

№ строки	Код	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 сократит популяцию до исходного числа особей, исключив из нее те, чье значение целевой функции минимально. То есть будут исключены строки 1, 3, 4 и 5, и популяция первого поколения примет вид, представленный в таблице 12.4.

Таблица 12.4.

№ строки	Код	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

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 возросло до 19, а минимальное значение составило 14. Лучшее же решение увеличилось с 18 до 27 при оптимальном решении 31.

Таким образом, данный пример наглядно иллюстрирует процесс улучшения как популяции в целом, так и наилучшего решения в частности в результате работы генетического алгоритма.

Пример 2. Решение задачи коммивояжера.

Задача коммивояжера является классической оптимизационной задачей. Суть ее заключается в следующем. Дано множество из n городов и матрица расстояний между ними или стоимостей переезда (в зависимости от интерпретации). Цель коммивояжера – объехать все эти города по кратчайшему пути или с наименьшими затратами на поездку. Причем в каждом городе он должен побывать один раз и свой путь закончить в том же городе, откуда начал.

Для решения предлагается следующая задача: имеется пять городов, стоимость переезда между которыми представлена следующей матрицей [4]:

0 4 6 2 9
4 0 3 2 9
6 3 0 5 9
2 2 5 0 8
9 9 9 8 0

Для решения задачи применим следующий генетический алгоритм. Решение представим в виде перестановки чисел от 1 до 5, отображающей последовательность посещения городов. А значение целевой функции будет равно стоимости всей поездки, вычисленной в соответствии с вышеприведенной матрицей. Сразу заметим, что одним из оптимальных решений задачи является последовательность 514235 стоимостью 29.

В качестве оператора отбора будем использовать традиционный оператор, применявшийся в предыдущем примере. При этом заметим, что чем меньше значение целевой функции, тем лучше.

В качестве оператора скрещивания выберем более изощренную процедуру, похожую на двухточечный оператор скрещивания. Поясним его работу на примере. Пусть есть две родительские перестановки (12345) и (34521). Случайно и равновероятно выбираются две точки разрыва. Для примера возьмем ситуацию, когда первая точка разрыва находится между первым и вторым элементами перестановки, а вторая точка – между четвертым и пятым: (1|234|5), (3|452|1). На первом этапе перестановки обмениваются фрагментами, заключенными между точками разрыва: (*|452|*), (*|234|*). На втором этапе вместо звездочек вставляются соответствующие числа из исходной родительской перестановки, начиная со второго числа выделенного фрагмента и пропуская уже имеющиеся в новой перестановке числа. В данном случае в первой перестановке (1|234|5) таким начальным числом является 3, за ним идет 4, которое есть в новой перестановке, и мы его пропускаем, также пропускаем число 5, переходим на начало перестановки и выбираем число 1. В итоге вместо (*|452|*) получаем (34521), аналогично из (3|452|1) и (*|234|*) получаем (52341).

Оператор мутации будет представлять собой случайную перестановку двух чисел в хромосоме, также выбранных случайно по равномерному закону. Вероятность мутации 0,01.

Размер популяции выберем равным 4. Исходная популяция представлена в таблице 12.5.

Таблица 12.5

№ строки	Код	Значение целевой функции	Вероятность участия в процессе размножения
1	12345	29	32/122
2	21435	29	32/122
3	54312	32	29/122
4	43125	32	29/122

Пусть для скрещивания были выбраны следующие пары: (1, 3) и (2, 4). В результате были получены потомки, представленные в таблице 12.6.

Пусть для потомка (12354) сработал оператор мутации и обменялись местами числа 2 и 3. В данном случае строка (12354) изменилась и приняла значение (13254).

Таблица 12.6

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	1 |23|45	5|43|12	32
3	5 |43|12	1 |23|54	28
Мутация 13254			
2	2 |143|5	4 |312|5	32
4	4 |312|5	2 |143|5	29

Популяция первого поколения после отсечения худших особей в результате работы оператора редукции приняла вид, представленный в таблице 12.7.

Таблица 12.7.

№ строки	Код	Значение целевой функции	Вероятность участия в процессе размножения
1	12345	29	28/111
2	21435	29	28/111
3	13254	28	29/111
4	2435	29	28/111

Пусть для получения второго поколения были выбраны следующие пары строк: (1, 4) и (2, 3). И в результате были получены потомки, показанные в таблице 12.8.

Таблица 12.8

№ строки	Родители	Потомки	Значение целевой функции для потомков
1	|123|45	|214|35	29
4	|214|35	|123|45	29
2	|21|435	|13|452	32
3	|13|254	|21|354	29

Популяция второго поколения после отсечения худших особей приняла вид показанный в таблице 12.9.

Таким образом, после двух итераций значение целевой функции для лучшего решения изменилось с 29 на 28, среднее значение изменилось с 30,5 до 28,75, а общее качество с 122 до 111. То есть также налицо незначительное, но улучшение популяции.

Таблица 12.9

№ строки	Код	Значение целевой функции	Вероятность участия в процессе размножения
1(1)	12345	29	28/111
2(2)	21435	29	28/111
3(3)	13254	28	29/111
4(н)	21354	29	28/111

Пример 3. Обучение нейронной сети

Обучение нейронных сетей является одной из основных областей применения генетических алгоритмов [4]. Для дальнейшего описания рассмотрим в общем виде, что представляет собой задача обучения нейронной сети.

Для построения и обучения нейронной сети обычно создается набор примеров, который представляет собой совокупность векторов вида (X,Y), где $X = (x_1, x_2, ..., x_n)$ - значения всех входов нейронной сети, а $Y = (y_1, y_2, ..., y_m)$ – значения всех выходов нейронной сети, которые должны получаться в процессе ее работы. Структура эталонных векторов задает одновременно количество входных и выходных нейронов (в данном случае n и m соответственно).

Целью обучения нейронной сети является достижение такой ситуации, когда при подаче на вход сети любого вектора X из набора примеров на ее выходе получается выходной вектор Y', отличающийся от эталонного вектора Y не более чем на заданную заранее и вычисляемую определенным образом величину δ.

Процесс обучения нейронной сети заключается в подборе значений всех ее характеристик таким образом, чтобы отличия выходных векторов от эталонных не превышали заранее установленной величины.

Основными характеристиками нейронной сети являются:

- количество скрытых слоев,
- количество нейронов в каждом из скрытых слоев,
- веса входов каждого из скрытых и выходных нейронов w_{ij},
- функция активации $F_j()$ для каждого скрытого и выходного нейрона.

Таким образом, мы можем обобщенно сформулировать оптимизационную задачу обучения нейронной сети. Исходными данными для нее будут: количество входных и выходных нейронов, набор обучающих примеров.

В результате решения этой задачи мы должны получить следующие значения:

- количество скрытых слоев,
- количество нейронов в каждом скрытом слое,
- значения весов всех входов для каждого скрытого и выходного нейрона,
- функции активации для каждого скрытого и выходного нейрона.

Очевидно, что целевой функцией в данном случае будет максимальное отличие выходного вектора Y' от эталонного $Y-\delta_{\max}$, полученное среди всех векторов набора примеров. Подбирая значения характеристик нейросети и вычисляя каждый раз $\delta_{\max}$, мы можем найти такое значение $\delta_{\max}$, которое требуется. И чем меньше будет это значение, тем качественнее построенная нейронная сеть.

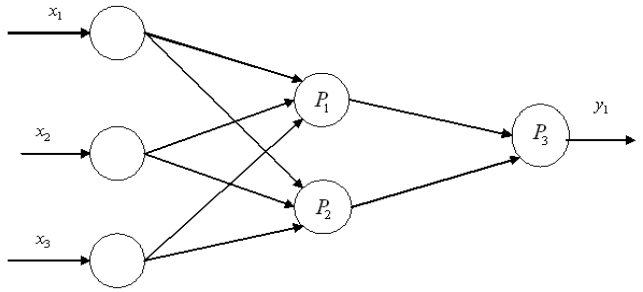
Таким образом, задача обучения нейронной сети сводится к задаче поиска оптимального решения. Заметим, что даже для простейших нейронных сетей эта задача является многомерной и крайне сложной. С другой стороны, цель найти сеть, удовлетворяющую условию $\delta_{\max}=0$, для реальных задач недостижима и обычно не ставится. Поэтому поиск оптимального решения превращается в поиск лучшего решения (как можно меньшее значение $\delta_{\max}$), где с успехом можно применять генетические алгоритмы.

Заметим, что для обучения нейросетей генетические алгоритмы в чистом виде, как правило, не применяют. Они обычно используются на различных этапах построения и обучения в качестве основного или вспомогательного средства. В частности, генетический алгоритм может использоваться на первом этапе работы для поиска общих параметров нейросети: количества скрытых слоев и нейронов. Также генетический алгоритм может использо-

ваться на заключительном этапе работы для поиска всех значений весов (матрицы весов) нейросети и функций активации. Причем функции активации, как правило, выбираются из ограниченного набора, а еще чаще подбирается не сама формула функции активации, а один или несколько ее параметров (коэффициентов).

В основном генетические алгоритмы используются для улучшения сетей, уже созданных и обученных с помощью какого-либо другого алгоритма. При этом оценка качества нейросети производится, как правило, средствами отдельного специализированного приложения.

Для иллюстрации применения генетических алгоритмов в области создания и обучения нейросетей рассмотрим пример. Для упрощения возьмем небольшую сеть прямого распространения (рисунок 12.3) и построим для нее обучающий генетический алгоритм.



входной слой скрытый слой выходной слой

Рисунок 12.3 – Нейронная сеть

Сеть состоит из 6 нейронов: трех входных, двух скрытых и одного выходного. Таким образом, сеть имеет всего один скрытый слой нейронов. В качестве функции активации всех скрытых и выходного нейронов используется сигмоидальная функция:

$$F(Sum) = \frac{1}{1 + e^{-P*Sum}}, \qquad Sum = \sum_{i=1}^{n} x_i * w_i \qquad (12.3)$$

где n - количество входов нейрона, x_i – значение на i-м входе, w_i – вес i-го входа, а P – параметр функции активации, влияющий на ее крутизну: чем больше P, тем ближе сигмоидная функция приближается к обычной пороговой (рисунок 12.3).

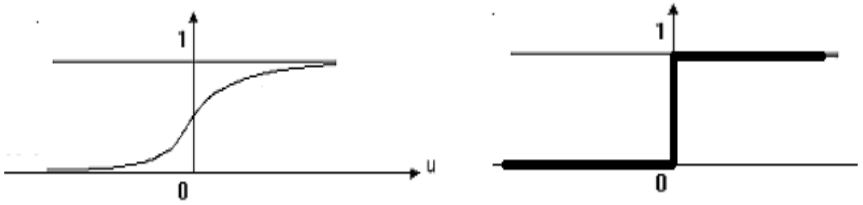


Рисунок 12.3 – Сигмоидальная и пороговая функции

Таким образом, векторы примеров будут иметь вид (x_1, x_2, x_3, y_1), где x_i – входные значения, а y_1 – эталонное выходное значение.

Для вычисления значения целевой функции будем использовать следующую формулу:

$$C = \max_{j=1}^{k} \frac{| y'_{1j} - y_{1j} |}{y_{1j}} * 100\% \qquad (12.4)$$

где k – количество примеров, y'_{1j} – значение выхода нейронной сети для j-го примера, y_{1j} – эталонное значение выхода нейронной сети для j-го примера.

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.

При обучении нейросетей размер популяции выбирается достаточно большим, как минимум 100 особей.

Хромосома в нашем случае будет представлять массив из 11 действительных чисел (таблица 12.10), по четыре на каждый скрытый нейрон и три для выходного нейрона, которые представляют собой веса соответствующих входов нейронов и параметры их функций активации (рисунок 12.2):

Таблица 12.10

Связь										
X_1 и P_1	X_2 и P_1	X_3 и P_1		X_1 и P_2	X_2 и P_2	X_3 и P_2		P_1 и P_3	P_2 и P_3	
w_{11}	w_{12}	w_{13}	P_1	w_{21}	w_{22}	w_{23}	P_2	w_{31}	w_{32}	P_3

Для элементов хромосомы – генов – введем ограничения, обусловленные природой нейросетей: $-1 \le w_{ij} \le 1$ и $P_i > 0$.

В качестве операторов скрещивания, отбора и редукции выберем традиционные операторы, рассмотренные выше при решении задачи нахождения максимума одномерной функции. Дополнительно для оператора скрещивания можно установить вероятность применения 0,95 и использовать элитизм,

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 нейрона 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_{13} уменьшается на 0,05; w_{21} – уменьшается на 0,01; w_{31} – увеличивается на 0,09.

Исходная популяция будет формироваться на основе равномерного распределения для каждого элемента хромосомы.

Целью работы генетического алгоритма будет поиск значений весов для всех скрытых и выходных нейронов, а также параметра P 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.

Задача полного построения нейросети обычно решается в два этапа, на каждом из которых используется свой генетический алгоритм.

На первом этапе генетический алгоритм используется для вычисления базовых характеристик: числа скрытых слоев и числа нейронов в каждом скрытом слое. При этом для оценки качества каждого решения используется следующий подход. Нейронная сеть, закодированная в проверяемой особи, обучается в течение фиксированного ограниченного промежутка времени. Обучение может происходить как средствами специального генетического алгоритма, так и другими. Если используется генетический алгоритм, то время обучения может оцениваться количеством поколений. Например, каждая сеть обучается в течение 100 поколений. Затем выбирается лучшая особь, качество которой и определяет качество решения по базовым характеристикам сети.

На втором этапе генетический алгоритм используется для вычисления матрицы весов и определения функций активации так, как это было рассмотрено выше.

На обоих этапах используется один и тот же набор обучающих примеров, являющийся единственной информацией, использующейся в качестве исходных данных для построения и обучения нейронной сети при решении конкретной задачи.

Рассмотренный процесс показывает, насколько сложной является задача построения и обучения нейросети как с точки зрения ее размерности, так и с точки зрения ее вычислительной сложности. Тем не менее, генетические алгоритмы представляют один из эффективных и изящных путей ее решения.

В заключение еще раз повторим, что эволюционные вычисления, в том числе и генетические алгоритмы, представляют собой подход к решению задачи поиска лучшего решения, а не четко определенный алгоритм. Для решения конкретной задачи, помимо ее формализации, формулировки генотипа и фенотипа, требуется создавать и конкретный генетический алгоритм. Для этого задают значения размера популяции, вероятности мутации, описывают процесс работы операторов отбора, скрещивания, мутации и редукции, что и было показано в рассмотренных примерах. И может оказаться, что алгоритм, успешно решающий одну задачу, совершенно не подходит для решения другой.

В настоящее время существует отдельная область исследований, целью которых является создание генетических алгоритмов, эффективно решающих как можно большее число задач. Результатом этих исследований являются значения характеристик генетических алгоритмов. Кроме создания новых алгоритмов, идет работа по усовершенствованию уже созданных. Также ведутся исследования по сокращению времени работы генетических алгоритмов за счет распараллеливания вычислений и применения ряда новых подходов

12.4 Примеры программного обеспечения

Решение конкретной оптимизационной задачи зачастую требует построения генетического алгоритма с уникальными значениями параметров. Однако ряд базовых свойств этих алгоритмов остается постоянным при решении совершенно разных задач. Поэтому в большинстве случаев для реализации конкретного генетического алгоритма не требуется создавать отдельный программный продукт. Приведем здесь несколько примеров программного обеспечения, позволяющего реализовывать широкий набор генетических алгоритмов, которые можно применять для решения самых различных задач.

Заметим, что изменяемыми параметрами генетических алгоритмов в таких приложениях обычно являются значения различных вероятностей, размер популяции и ряд специфических свойств алгоритма. А реализация генетических операторов, как правило, едина для всех алгоритмов и скрыта от пользователя.

Пакет Evo1ver 4.0 компании Palisade Corp. Пакет Evolver представляет собой дополнение к программе MS Excel версий 5.0 и 7.0. При этом Excel используется как средство описания исходных данных алго ритма и расчетов в процессе его работы. В процессе установки Evo1ver добавляет в Excel дополнительную панель инструментов, которая обеспечивает доступ к пакету. Если Evo1ver не запущен на выполнение, то панель управления не отображается. При запуске Evo1ver приложение Excel запускается автоматически.

Исходные данные для работы формируются обычным образом. При этом можно применять весь набор средств, предоставляемых Excel. Особенность заключается лишь в том, что обязательно должна присутствовать ячейка, в ко торой задана формула для вычисления целевой функции.

Задав параметры генетического алгоритма и описав исходные данные, нужно закрыть окно установки и нажать кнопку запуска оптимизации на панели инструментов. В процессе оптимизации на экране отображается график изменения целевой функции, а в строке состояния Excel – номер текущего поколения и лучшее для текущей популяции значение целевой функции. Остановить алгоритм можно, нажав соответствующую кнопку на панели инструментов или клавишу Esc. По завершении работы алгоритма появляется окно, в котором пользователь может указать дальнейшие действия. Обычно это формирование файла отчета на отдельном листе Excel и изменение информации на листе с исходными данными, соответствующее лучшему найденному решению. Заметим, что все установки генетического алгоритма автоматически сохраняются вместе с файлом Excel. Evo1ver поставляется с подробной справочной системой и обучающим видеороликом, которые значительно облегчают знакомство с пакетом. Дополнительно к пакету может поставляться средство расширенной разработки генетических алгоритмов, позволяющее конструировать собственные операторы скрещивания, отбора и мутации.

Пакет ***GeneHunter*** во многом схож с пакетом Evo1ver. Он также является надстройкой над MS Excel версий 5.0 и 7.0 и запускается из меню Сервис. Пакет является русифицированным и имеет ряд дополнительных настроек для генетических алгоритмов: включение стратегий элитизма и разнообразия. Поля окна ***GeneHunter*** практически повторяют поля Evo1ver. Однако окно имеет ряд отличий, а также существенное отличие в параметрах оператора скрещивания. В Evo1ver вероятность скрещивания всегда равна 1, зато пользователь может задавать положение точки разбиения хромосомы. В ***GeneHunter***, наоборот, пользователь определяет вероятность скрещивания, а точка разбиения выбирается случайно по равномерному закону.

GeneHunter имеет некоторые неудобства интерфейса по сравнению с Evo1ver, зато работает значительно быстрее него. А наличие русифицированной вер сии, включая систему подсказки, делает его еще более предпочтительным для использования. Кроме того, в комплекте поставляются несколько самостоятельных приложений, реализующих генетические алгоритмы решения задач оптимизации одномерной и двухмерной функций и задачи коммивояжера. В комплекте также содержится файл примеров в формате Excel, иллюстрирующий возможности решения с помощью ***GeneHunter*** задач оптимизации функций, коммивояжера, управления пакетом ценных бумаг, задачи о рюкзаке, обучения нейронной сети и ряд других. Рассмотрение этих примеров значительно упрощает знакомство с пакетом и дает первоначальные навыки формализации исходных задач для их последующего решения с помощью ***GeneHunter***.

Пакет Genetic Training Option компании California ScintificSoftware (GTO) является дополнительной утилитой, поставляемой для нейросетевого пакета BrainMaker производства компании ***California ScintificSoftware***. Он применяется как для построения нейронных сетей, так и для улучшения созданной с помощью BrainMaker сети. Но в обоих случаях отдельно от BrainMaker использоваться не может.

Мы кратко рассмотрели несколько программных продуктов, реализующих генетические алгоритмы. Однако даже приведенные примеры позволяют оценить, насколько они сложны в реализации и просты в применении – требуют от пользователя только формализации задачи и формирования исходных данных. Такая ситуация во многом способствует расширению области применения генетических алгоритмов.

Тема 13 Нейронные сети

13.1 Проблемы построения нейронных сетей

Нейронная сеть – это набор соединенных между собой нейронов. Функции всех нейронов сети постоянны, а веса и параметры (импульсы) могут изменяться. Нейронная сеть имеет внешние входы и внешние выходы. Построение нейронной сети заключается в выборе архитектуры сети и подборе весов сети. Подбор весов – это обучение сети.

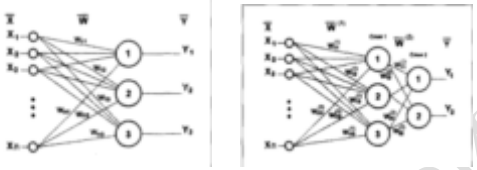
При разработке архитектуры сети нужно учитывать:

– число входов и передаточные функции;
– способ соединения нейронов между собой внутри сети;
– количество выходов и то, что будет на каждом выходе.

Разработка архитектуры нейронная сети – сложная задача, но ее можно разрабатывать не «с нуля», а выбрать ту из уже существующих архитектур нейронных сетей, которая лучше всего подходит для решения вашей задачи. При этом нельзя забывать, что эффективность многих архитектур для решения тех или иных задач доказана математически. К всеобщим услугам: сети Кохонена, сети с общей регрессией или многослойный персептрон. Кстати, математическая модель нейрона – это тоже архитектура нейронной сети, которая называется однонейронным персептроном. Выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные на сегодняшний день конфигурации. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации.

Теоретически число слоев и число нейронов в каждом слое нейронной сети может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется нейронная сеть.

При этом, если в качестве активационной функции для всех нейронов сети используется функция единичного скачка, нейронная сеть называется однослойным (многослойным) персептроном (рис. 13.1).



а) б)

Рисунок 13.1 – Персептроны: а) однослойный; б) многослойный

13.2 Обучение нейронных сетей

Очевидно, что функционирование нейронной сети, т. е. действия, которые она способна выполнять, зависит от величин синоптических связей. Поэтому, задавшись структурой нейронной сети, отвечающей определенной задаче, разработчик должен найти оптимальные значения для всех весовых коэффициентов w.

Этот этап называется обучением нейронной сети, и от того, насколько качественно он будет выполнен, зависит способность сети решать во время эксплуатации поставленные перед ней проблемы.

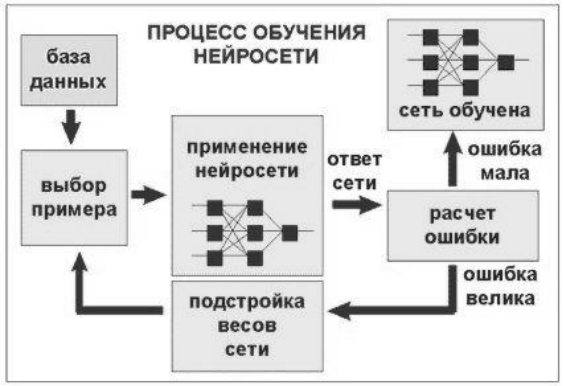


Рисунок 13.2 – Схема процесса обучения нейронной сети

Важнейшими параметрами обучения являются: качество подбора весовых коэффициентов и время, которое необходимо затратить на обучение.

Как правило, два этих параметра связаны между собой обратной зависимостью и их приходится выбирать на основе компромисса.

В настоящее время все алгоритмы обучения нейронных сетей можно разделить на два больших класса: с учителем и без учителя. Общая схема процесса обучения нейронной сети показана на рисунке 13.2

Обучение с учителем. Нейронной сети предъявляются значения как входных, так и выходных параметров, и она по некоторому внутреннему алгоритму подстраивает веса своих синоптических связей.

Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются ***представительской*** или ***обучающей*** выборкой. Обычно нейронная сеть обучается на некотором числе таких выборок. Предъявляется выходной вектор, вычисляется выход нейронной сети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в нейронную сеть, и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

Обучение без учителя. Нейронной сети предъявляются только входные сигналы, а выходы сети формируются самостоятельно с учетом только входных и производных от них сигналов.

Несмотря на многочисленные прикладные достижения, обучение с учителем критиковалось за свою биологическую неправдоподобность. Трудно вообразить обучающий механизм в естественном человеческом интеллекте, который бы сравнивал желаемые и действительные значения выходов, выполняя коррекцию с помощью обратной связи. Если допустить подобный механизм в человеческом мозге, то откуда тогда возникают желаемые выходы? Обучение без учителя является более правдоподобной моделью обучения в биологической системе. Развитая Кохоненом и многими другими, она не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами.

Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса нейронной сети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет статистические свойства обучающего множества и группирует сходные векторы в классы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, но до обучения невозможно предсказать, какой выход будет производиться данным классом входных векторов.

Следовательно, выходы подобной сети должны трансформироваться в некоторую понятную форму, обусловленную процессом обучения. Это не является серьезной проблемой. Обычно не сложно идентифицировать связь между входом и выходом, установленную сетью.

13.3 Пример алгоритма обучения: обратного распространения ошибки

В результате обучения на примерах строятся математические решающие функции нейронной сети (передаточные функции или функции активации), которые определяют зависимости между входными (*Xi*) и выходными (*Yj*) признаками (сигналами) (рисунок 13.3).

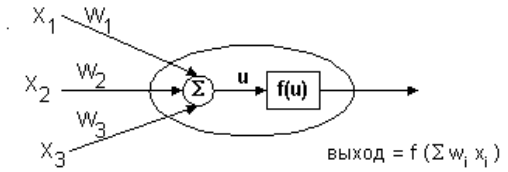


Рисунок 13.3 – Решающая функция – «нейрон»

Каждая такая функция, называемая по аналогии с элементарной единицей человеческого мозга – нейроном, отображает зависимость значения выходного признака (*Y*) от взвешенной суммы (*U*) значений входных признаков (*X*i), в которой вес входного признака (*Wi*) показывает степень влияния входного признака на выходной:

$$Y = f(\sum_i W_i * X_i)$$

Решающие функции используются в задачах классификации на основе сопоставления их значений при различных комбинациях значений входных признаков с некоторым пороговым значением. В случае превышения заданного порога считается, что нейрон сработал и таким образом распознал некоторый класс ситуаций. Нейроны используются и в задачах прогнозирования, когда по значениям входных признаков после их подстановки в выражение решающей функции получается прогнозное значение выходного признака.

Функциональная зависимость может быть линейной, но, как правило, используется сигмоидальная форма, которая позволяет вычленять более сложные пространства значений выходных признаков. Такая функция называется логистической (рисунок.13.4).

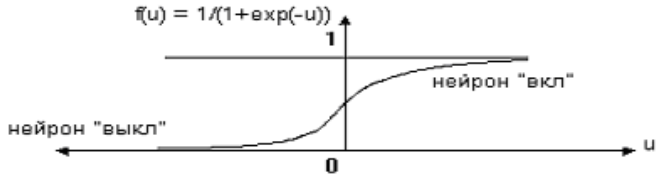


Рисунок 13.4 – Логистическая (сигмоидальная) функция

Нейроны могут быть связаны между собой, когда выход одного нейрона является входом другого. Таким образом, строится нейронная сеть (рисунок.13.5), в которой нейроны, находящиеся на одном уровне, образуют слои.

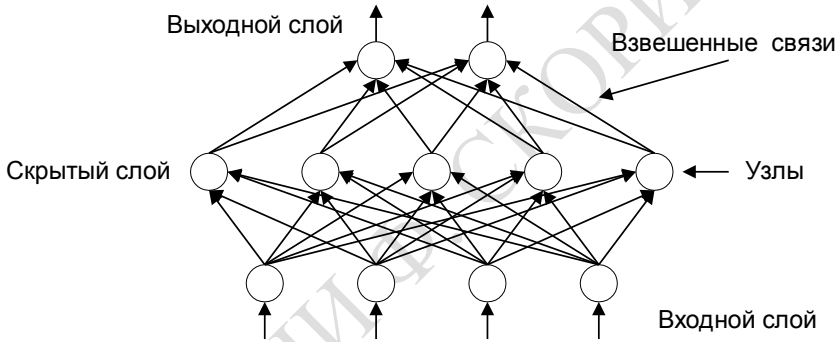


Рисунок.13.5 – Нейронная сеть

Обучение нейронной сети сводится к определению связей (синапсов) между нейронами и установлению силы этих связей (весовых коэффициентов). Алгоритмы обучения нейронной сети упрощенно сводятся к определению зависимости весового коэффициента связи двух нейронов от числа примеров, подтверждающих эту зависимость.

Наиболее распространенным алгоритмом обучения нейронной сети является алгоритм обратного распространения ошибки. Целевая функция по этому алгоритму должна обеспечить минимизацию квадрата ошибки в обучении по всем примерам:

$$\min \sum_i (T_i - Y_i)^2 ,$$

где Ti – заданное значение выходного признака по i – му примеру;

Yi – вычисленное значение выходного признака по i – му примеру.

Сущность алгоритма обратного распространения ошибки сводится к следующему:

– Задать произвольно небольшие начальные значения весов связей нейронов.

– Для всех обучающих пар «значения входных признаков – значение выходного признака» (примеров из обучающей выборки) вычислить выход сети (Y).

– Выполнить рекурсивный алгоритм, начиная с выходных узлов по направлению к первому скрытому слою, пока не будет достигнут минимальный уровень ошибки.

Вычислить веса на (t+1) шаге по формуле:

$$W_{ij}(t+1) = W_{ij}(t) + \eta\delta_i X_i,$$

где $W_{ij}(t)$ – вес связи от скрытого i – го нейрона или от входа к j-му нейрону на шаге t;

X_i – выходное значение i – го нейрона;

η – коэффициент скорости обучения;

δ_i – ошибка для j-го нейрона.

Если j-й нейрон – выходной, то

$$\delta_i = Y_i(1-Y_i)(T_i - Y_i)$$

Если j-й нейрон находится в скрытом внутреннем слое, то

$$\delta_j = X_j(1-X_j)\sum_k \delta_k W_{jk},$$

где k – индекс всех нейронов в слое, расположенном вслед за слоем с j-м нейроном.

Выполнить шаг 2.

Достоинство нейронных сетей перед индуктивным выводом заключается в решении не только классифицирующих, но и прогнозных задач. Возможность нелинейного характера функциональной зависимости выходных и входных признаков позволяет строить более точные классификации.

Сам процесс решения задач в силу проведения матричных преобразований проводится очень быстро. Фактически имитируется параллельный процесс прохода по нейронной сети в отличие от последовательного в индуктивных системах. Нейронные сети могут быть реализованы и аппаратно в виде нейрокомпьютеров с ассоциативной памятью.

Последнее время нейронные сети получили стремительное развитие и очень активно используются в финансовой области. В качестве примеров внедрения нейронных сетей можно назвать: «Система прогнозирования динамики биржевых курсов для Chemical Bank» (фирма Logica); «Система прогнозирования для Лондонской фондовой биржи» (фирма SearchSpace); «Управление инвестициями для Mellon Bank» (фирма NeuralWare) и др.

В качестве инструментальных средств разработки нейронных сетей следует выделить инструментальные средства NeurOn-line (фирма GENSYM), NeuralWorks Professional II/Plus (фирма NeuralWare), отечественную разработку FOREX – 94 (Уралвнешторгбанк) и др.

13.4 Персептрон

Персептрон – иерархическая система распознавания образов (рис.13.6), моделирующая зрительное восприятие живых организмов. Автор – Розенблатт Ф. предложил его как модель мозга. Он состоит из множеств: S-элементов, реагирующих на сигналы внешней среды (называемого полем рецепторов, сетчаткой); ассоциативных А-элементов, связанных в единую сеть; R-элементов, выполняют эффективную роль. Связь между А и R-элементами регулируется изменяемой матрицей взаимодействия. Каждый элемент поля рецепторов может находиться в одном из 2-х состояний (1,0): возбужден или нет.

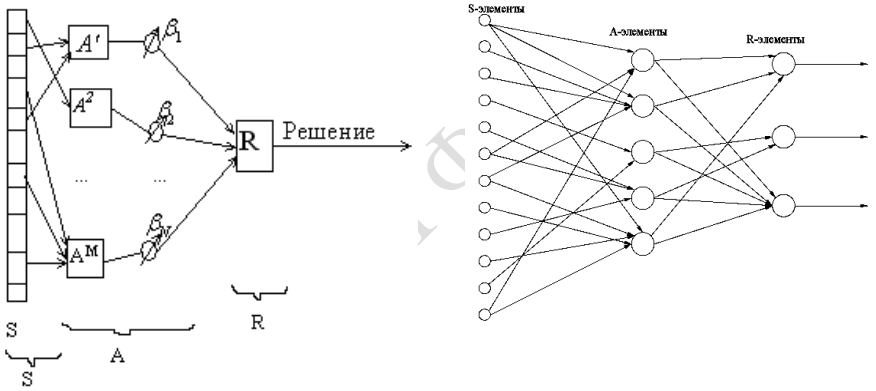


Рисунок.13.6 – Структурная схема простейшего (трехслойного) персептрона

Матрица связей между *S* и А-элементами определяет, какие рецепторы связаны с определенным А-элементом и знак связи. Удачность выбора матрицы влияет на качество обучения и распознания

А – элементы суммируют сигналы и сравнивают результат с порогом θ_1. Сигнал a_j на выходе из A_j зависит от превышения θ_i

$$a_j = \begin{cases} 1, \sum_{i=1}^{K} b_{ij} x_i \geq \theta \\ 0, \sum_{i=1}^{K} b_{ij} x_i < \theta \end{cases}, \; b_{ij} = \pm 1,0;$$

определяет наличие и знак связи *i*-го рецептора с A_j-элементом.

Выходные сигналы A – элементов взвешиваются с помощью коэффициента β_i: $r = \sum_{i=1}^{M} \beta_j a_j$

Этот сигнал поступает на вход R и реализует пороговое правило принятия решения: $X \in \begin{cases} \pi_1, r \geq 0 \\ \pi_2, r < 0 \end{cases}$

Существует большое число алгоритмов обучения персептрона с помощью итеративной процедуры коррекции весов $\beta_i (j = \overline{1}, \overline{M})$. На каждом шаге обучения на вход персептрона подается изображение объекта одного из образов. В зависимости от принятого персептроном решения производится коррекция коэффициента β. За конечное число шагов обучения можно научить персептрон надежно распознавать предъявляемые изображения. Например, обучение может состоять в том, что коэффициент β_i изменяется при ошибках персептрона. Если для $X \in \pi_1$, персептрон выдал $X \in \pi_2$, то веса возбужденных *A-элемент*ов, для которых $a_j = 1$ увеличиваются, чтобы увеличивать *r*.

Обучение происходит под контролем учителя, который сообщает персептрону правильный ответ d(t) для любого входного вектора из обучающей выборки. Персептрон многократно проходит цикл обучения, состоящий:

- из предъявления вектора входных данных;
- вычисления ответа персептрона y(t);
- сообщения ему правильного ответа и корректировки весов связей.

Алгоритм обучения продолжает работу до тех пор, пока все входные векторы не будут правильно классифицированы. Скорость процесса обучения зависит от параметра обучения μ, $\mu \in (0,1)$. Он связан с минимизацией общей ошибки функционирования персептрона.

Алгоритм обучения элементарного персептрона может быть записан следующим образом:

1) инициализировать веса и пороговые значения, как случайные числа из диапазона [-0.1;0.1];
2) выбрать очередной входной вектор x(t) из обучающей выборки;
3) рассчитать выходной сигнал персептрона

$$y(t) = f\left(\sum_{i=1}^{M} \beta_j a_j(t) - \vartheta\right)$$

4) выполнить коррекцию весов связей

$$\beta_j(t+1) = \beta_j(t) + \mu[d(t) - y(t)]a_i(t),\ \mu \in (0,1)$$

$$d(t) = \begin{cases} 1, если\_выходной\_вектор\_из\_класса\_\pi_1 \\ -1, если\_выходной\_вектор\_из\_класса\_\pi_2 \end{cases}$$

5) если не все векторы из обучающей выборки классифицированы правильно, то перейти к шагу 2

6) конец.

Если перцептрон действует по описанной схеме и в нем допускаются лишь связи, идущие от бинарных S-элементов к А-элементам и от А-элементов к единственному R-элементу, то такой перцептрон принято называть элементарным а-перцептроном. Обычно классификация d(t) задается учителем. Перцептрон должен выработать в процессе обучения классификацию, задуманную учителем.

О перцептронах было сформулировано и доказано несколько основополагающих теорем, две из которых, определяющие основные свойства перцептрона, приведены ниже.

Теорема 1. Класс элементарных а-перцептронов, для которых существует решение для любой задуманной классификации, не является пустым. {Эта теорема утверждает, что для любой классификации обучающей последовательности можно подобрать такой набор (из бесконечного набора) А-элементов, в котором будет осуществлено задуманное разделение обучающей последовательности при помощи линейного решающего правила.}

Теорема 2. Если для некоторой классификации d(t) решение существует, то в процессе обучения а – перцептрона с коррекцией ошибок, начинающегося с произвольного исходного состояния, это решение будет достигнуто в течение конечного промежутка времени. {Смысл этой теоремы состоит в том, что если относительно задуманной классификации можно найти набор А-элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто в конечный промежуток времени.}

Обычно обсуждают свойства бесконечного перцептрона, т. е. перцептрона с бесконечным числом А-элементов со всевозможными связями с S-элементами (полный набор А-элементов). В таких перцептронах решение всегда существует, а раз оно существует, то оно достижимо и в а – перцептронах с коррекцией ошибок. Очень интересную область исследований представляют собой многослойные перцептроны и перцептроны с перекрестными связями, но теория этих систем практически еще не разработана.

Раздел 3 Ситуационное управление

Тема 14 Типологизация систем управления

14.1 Структура взаимодействия среды, объекта и системы управления

14.2 Типы систем управления. Простые системы управления

14.3 Типы систем управления: системы с адаптацией

14.4 Типы систем управления: модельные системы управления

14.1 Структура взаимодействия среды, объекта и системы управления

Управление сложными объектами принципиально невозможно без привлечения информации, которая не может быть выражена количественно. Это семантическая, т. е. смысловая, качественная информация.

Обстоятельства, побудившие специалистов в области управления начать работы по изучению моделей, называемые логико-лингвистическими моделями управления, возникли, когда в сферу автоматизации оказались вовлеченными объекты столь сложной природы, что традиционные методы теории управления оказались для них либо малоэффективными, либо просто непригодными. Укажем некоторые из причин.

1. Не все цели управления объектом могут быть выражены в виде количественных соотношений.

2. Между рядом параметров, оказывающих влияние на процесс управления, не удается установить точных количественных зависимостей.

3. Процесс управления является многошаговым, и содержание каждого шага не может быть заранее однозначно определено.

4. Существующие способы описания объектов и протекающих в них процессов приводят к столь громоздким конструкциям, что их практическое использование невозможно.

5. Если несколько расширить понимание термина «объект управления», включив в него, например экономические или социальные объекты, то к перечисленным причинам неэффективности классических методов управления можно добавить, по крайней мере, еще три.

6. Цель существования самого объекта не может быть строго сформулирована и, тем более, количествен но выражена.

7. Объект эволюционирует во времени, меняется его структура и функции, что приводит к эволюции самого процесса управления.

8. Элементы, входящие в структуру управляемого объекта, имеют активную природу. Их поведение может противоречить целям управления; они могут оказывать обратное воздействие на саму систему управления.

В проблеме управления сложными техническими системами основное значение имеют первые четыре из названных причин. Именно в них кроется необходимость перехода от классических моделей теории управления к логико-лингвистическим моделям.

Опыт специалиста в управлении объектами, для которых цели управления выражаются не столько количественными соотношениями, сколько качественными формулировками, может быть использован в автоматизированной системе управления лишь в том виде, в котором он реально зафиксирован. Следовательно, приходится использовать информацию, содержащуюся в текстах на обычном, естественном языке. А это в свою очередь делает необходимым использование для представления такой информации в ЭВМ языков с развитой семантикой, приближающихся по своим выразительным возможностям к естественному языку. Их использование позволит, кроме того, выразить и те зависимости между параметрами, которые носят качественный характер.

То обстоятельство, что невозможно заранее определить содержание каждого шага управления, вызывает, как правило, настолько большое число ситуаций, характеризующих состояние объекта, что практически невозможно проанализировать влияние каждой из них на принимаемые решения. Таким образом, вместо алгоритма управления, предписывающего на каждом шаге его реализации некоторое однозначное решение, остается использовать совокупность указаний, похожую на то, что в математике принято называть исчислением. В отличие от алгоритма в исчислении продолжение процесса на каждом шаге не является фиксированным, и на каждом из этих шагов имеется возможность произвольного продолжения процесса поиска решения. Исчисления и подобные им системы изучаются в математической логике.

Наконец, специалист в области управления сложными объектами испытывает весьма большие трудности от несоответствия степени подробности описания необходимой точности принятия решений. Выбор этой степени подробности зависит от решения некоторой конкретной задачи управления и сам по себе представляет проблему, которую необходимо решать. Для сложных объектов различные задачи управления могут требовать различной степени подробности описания, что приводит к необходимости введения иерархической системы описаний. В конкретных ситуациях необходимы конкретные, детальные описания, в соответствии с которыми можно реализовать конкретные действия по управлению. Но закономерности управления в силу большого числа конкретных ситуаций могут быть сформулированы лишь на достаточно общем уровне. Поскольку все задачи управления относятся к одному и тому же объекту, все решения должны быть согласованы между собой, равно как и различные уровни описаний. Эта увязка осуществляется с помощью введения иерархической системы понятий, связанных операциями

обобщения и конкретизации. Итак, в системах управления рассматриваемого типа при необходимости возникают классифицирующие системы.

Уточним некоторые понятия, которые будут использоваться в дальнейшем. На рис. 14.1 показана структура взаимодействия среды, объекта и системы управления.

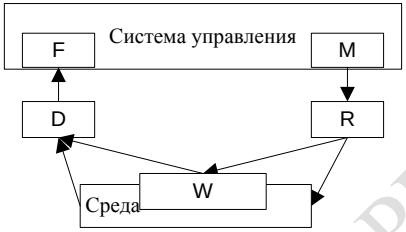


Рисунок 14.1 – Структура взаимодействия среды, объекта и системы управления. Обозначения на рисунке: D- входной преобразователь; R-выходной преобразователь; M- модель знаний; F- механизм порождения решений

Отличие среды от объекта состоит в том, что на среду система управления непосредственно не воздействует, хотя опосредствованное ее воздействие на среду, осуществляемое через объект, вполне возможно. Система управления получает сведения от среды и объекта через входной преобразователь D. Решения, формируемые системой управления, поступают на объект через выходной преобразователь R. Обозначим через S^N наблюдаемую ситуацию, т. е. совокупность факторов, устанавливаемых в результате непосредственной обработки данных, приходящих от объекта и среды в систему управления. В дальнейшем, когда будем подробно рассматривать структуру системы управления, будут введены другие типы ситуаций (проблемная, обобщенная, пополненная).

Практически любая система управления должна строиться на основе определенных знаний об объекте управления. Они отражают фундаментальные свойства объекта, не зависящие от текущей ситуации. Например, это может быть совокупность передаточных функций отдельных элементов объекта управления. Множество этих знаний в дальнейшем будет называться моделью знаний об объекте. Аналогичными сведениями система управления может располагать и о среде, что дает возможность говорить о модели знаний о среде. Эта модель знаний совместно с моделью знаний об объекте будет называться просто моделью знаний и обозначаться символом M. Кроме M в системе управления всегда присутствует некоторый механизм порождения решений в виде совокупности определенных процедур. Этот механизм в дальнейшем будем обозначать символом F.

14.2 Типы систем управления: простые системы управления

Проследим эволюцию структур систем управления, целью которой является расширение возможностей автоматизированного управления. Рассмотрим ступени эволюции: введение обратной связи, адаптации и системы знаний об объекте.

На рис. 14.2 показаны наиболее простые структуры систем управления. На этом рисунке использованы те же обозначения, что и на рис. 14.1. Символом W обозначен объект управления (вместе со средой, если это необходимо).

а) Разомкнутая СУ без обратной связи б) СУ с обратной связью

Процедура управления жесткая или вероятностная

Нет смысла говорить о наблюдаемой ситуации S^N

Рисунок 14.2 – Простые структуры систем управления

На рис. 14.2а показана структурная схема разомкнутой системы или, иначе, системы без обратной связи. Такие системы реализуют заложенную в них процедуру управления, «не интересуясь», тем, что из этого на самом деле получается. Эта процедура может быть либо жесткой, детерминированной, как например при использовании профилированных кулачков, либо вероятностной, как в игральных автоматах. Стандартным примером для иллюстрации работы разомкнутых систем всегда было программное управление станками, несмотря на то, что в действительности трудно найти современную систему управления станком, в которой совершенно отсутствовали бы элементы обратной связи, хотя бы в виде подстройки для учета износа инструмента. Формальной моделью разомкнутых систем дискретного типа является автономный конечный или вероятностный автомат. Автономный конечный автомат обладает фиксированным конечным множеством внутренних состояний $Y = \{y_1, y_2, ..., y_n\}$. Смена состояний автомата происходит в дискретные такты времени t. Если через y^t обозначить состояние автомата,

в котором он находится в момент времени t, то работа автономного автомата будет полностью определяться функцией перехода $y^{t+1} = f(y^t)$. С каждым состоянием автомата однозначно связан выходной сигнал, поступающий на преобразователь R. Основным свойством автономного автомата является то, что на его выходе всегда реализуется циклически повторяемая последовательность сигналов (допускается наличие конечного апериодического ряда сигналов в начальные такты работы автомата). Вероятностный автономный автомат имеет вместо функции переходов фиксированное распределение вероятностей смены состояний. Такой автомат представляет собой марковский источник сигналов (или композицию таких источников). Выходная последовательность сигналов при фиксированном начальном состоянии автомата всегда характеризуется заданными финальными вероятностями появления выходных сигналов. Для систем непрерывного типа аналогом автономного автомата является устройство, реализующее на выходе периодическую функцию времени (в частности, упомянутый ранее профилированный кулачок). Заметим, что в детерминированной системе блок F реализует жесткий алгоритм, а в вероятностной – процедуру, для которой возможно вероятностное предсказание ее работы на каждом шаге.

Вообще же отсутствие обратных связей не типично для управляющих систем. Заметим, что для подобных систем не имеет смысла говорить о наблюдаемых ситуациях S^N, поскольку их описания внутри системы нет.

На рис. 14.26 показана структурная схема системы с обратной связью, хорошо знакомая всем специалистам в области автоматизированного управления. Существует множество примеров подобных систем (автономные регуляторы, релейные устройства защиты и т. п.), и нет необходимости в их специальном рассмотрении. В такой системе работа блока F определяется не только заложенными в него процедурами, но и наблюдаемыми ситуациями, появление которых внутри системы управления определяется обратными связями, идущими от объекта и среды. В системах дискретного типа формальной моделью блока F является алгоритм в общепринятом смысле этого слова. В частности, работа блока F в простом случае может определяться некоторым детерминированным или вероятностным автоматом, имеющим активные входы. Такой автомат при отсутствии вероятностей задается тремя конечными множествами: входов $X = \{x_1, x_2, .., x_k\}$, состояний $Y = \{y_1, y_2, ..., y_n\}$ и выходов $Z = \{z_1, z_2, ..., z_m\}$, а также двумя функциями: функцией переходов $y_i^{t+1} = f_i(x_1^t, ..., x_k^t, y_1^t, ..., y_n^t), \quad i = 1,2,...,n;$ и функцией выходов $z_j^{t+1} = \varphi_j(x_1^t, ..., x_k^t, y_1^t, ..., y_n^t), j = 1,2,...,m$. Здесь, как и ранее, индексы *t* и *t+1* обозначают такты работы автомата. При реализации вероятностного автомата вместо функций перехода и выхода используются

условные вероятностные распределения смены состояний автомата от значений входного сигнала, а также вероятностные распределения значений выходных сигналов. Описание наблюдаемых ситуаций совпадает с заданием значений сигналов на входах автомата. В более сложном случае F реализуется в виде алгоритма с условными переходами, определяемыми наличием тех или иных исходных данных. Эти исходные данные отождествляются с элементами наблюдаемой ситуации. В системах непрерывного типа работа блока F определяется заданием некоторой функции $z = \Phi(u_1, u_2, ..., u_r, t)$. Аргументы этой функции определяются на основании значений элементов наблюдаемой ситуации.

Системы, рассмотренные на рис 14.2, будем называть системами типа F, или F-системами. Характерной особенностью их является то, что при появлении на входе системы управления любой наблюдаемой ситуации из множества потенциально возможных ситуаций заранее определено, как будут выполняться процедуры, заложенные в F. Число наблюдаемых ситуаций может быть небольшим или большим и даже бесконечным. Когда перечисление всех возможных ситуаций затруднительно или невозможно, они объединяются в классы, для которых характерна однотипная реализация процедур в F. Отнесение наблюдаемых ситуаций к тому или иному классу заранее регламентировано в самом F.

14.3 Типы систем управления: системы с адаптацией

На рис. 14.3 показана структурная схема системы управления, отличающаяся от изображенной на рис. 14.26 наличием блока А. Этот блок будем называть адаптатором. Его функция – выбор определенной совокупности процедур F на основании *Sn*- наблюдаемой ситуации и соответствующее изменение структуры F. Блок А целесообразно считать отдельным, самостоятельным, а не частью F в связи с тем, что интенсивность работы адаптатора существенно меньше, чем других блоков, а также тем, что построить ССУ проще в виде композиции простых и переключателя.

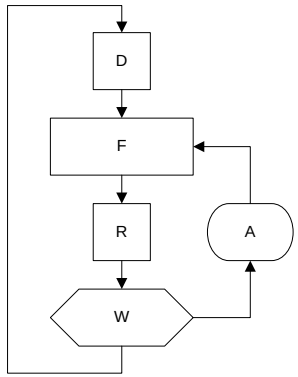


Рисунок 14.3 – Схема системы с адаптацией

Основным назначением адаптатора является выбор определений совокупности процедур, реализуемых в F из множества потенциально допустимых процедур на основании анализа наблюдаемых ситуаций.

Примером адаптации непрерывной системы является автоматическое изменение уставок регулятора или изменение закона регулирования при использовании линейного регулятора для управления нелинейным объектом, когда нелинейная характеристика аппроксимируется отрезками прямых линий разного наклона.

Может возникнуть вопрос, почему в приведенных примерах процедуры адаптатора отделены от процедур, реализуемых блоком F? Почему на рис. 14.3 эти блоки нарисованы отдельно? Действительно, нет большого труда представить себе, что все процедуры адаптации включены в состав процедур блока F. В этом смысле всякая система управления замкнутого типа является системой с адаптацией. Однако есть, по крайней мере, два соображения, по которым целесообразно выделить процедуры адаптации в отдельный блок. Во-первых, интенсивность работы адаптатора существенно меньше интенсивности работы других блоков системы управления. Во-вторых, построить сложную систему управления в виде композиции значительно более простых систем и переключателя, с помощью которого можно на каждом шаге работы выбирать ту или иную подсистему, значительно проще, чем построить единую систему.

Правда, то, что процедуры адаптации ничем не отличаются от процедур, реализуемых блоком F, верно лишь тогда, когда сам адаптатор работает по жесткому закону, а цели адаптации зафиксированы. Если же адаптатор имеет нежесткие законы работы, то это означает наличие адаптатора второго уровня, для которого исходный адаптатор выступает в качестве блока F. Ес-

ли такой адаптатор второго уровня сам работает по жестким законам, а цели его работы зафиксированы, то это ничем принципиально не отличается от структуры, рассмотренной выше. Наращивание подобных ступеней адаптации не вносит качественных изменений в структуру, хотя и меняет число уровней декомпозиции системы управления.

Единственным, случаем, когда в систему вносится качественное изменение, является появление на очередном уровне адаптации человека, у которого могут быть не зафиксированы цели адаптации и способы воздействия на нижние уровни системы.

Высказанные соображения позволяют выделить системы с адаптацией в отдельный тип систем управления. Эти системы назовем системами типа F-A, или F-A-системами.

14.4 Типы систем управления: модельные системы управления

Структурная схема системы управления, которую будем называть модельной, показана на рис. 14.4. Новым в ней является блок M, называемый моделью. M- модель знаний об объекте, M выделяется из F в отдельный блок. Назовем их F-M-системами

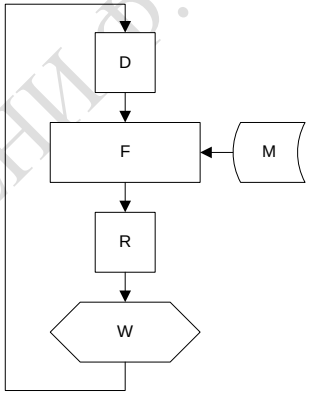


Рисунок 14.4 – Схема модельной системы управления

Модель содержит в себе совокупность определенных знаний об объекте, используемых при управлении. Необходимость выделения знаний такого типа из F в специальный блок M можно проиллюстрировать на двух примерах. Пусть интересующий нас процесс описывается уравнением $H(h_1,...,h_s,g)=0$. В этом уравнении h_i, аргументы, получаемые из наблюдаемой ситуации, а g значение, которое необходимо вычислить для

системы управления. Если представить исходное уравнение в виде, разрешенном относительно g, т. е, как $g = Q(h_1,...,h_s)$, и для решения этого уравнения написать алгоритм, то этот алгоритм может быть включен в состав процедур, реализуемых блоком F. Если же исходное уравнение не удается разрешить относительно g, то поиск значений g может осуществляться, например способом слепого перебора. В этом случае в F содержится процедура для этого перебора, а соотношение H=0 хранится в модели.

Другим примером появления блока M может служить решение задачи о выборе параметров системы автоматического регулирования с использованием аналоговой модели. В этом случае априорно известен тип дифференциального уравнения, описывающего динамику системы. Известны также характеристики объекта управления, содержащиеся в наблюдаемой ситуации, но не известны характеристики регулятора, отражаемые в коэффициентах уравнения. Если бы была известна процедура прямого нахождения этих коэффициентов, то в F были бы записаны соответствующие алгоритмы реализации этой процедуры. Однако положение, как правило, таково, что алгоритмы не известны. Тогда поиск коэффициентов дифференциального уравнения происходит с помощью подбора таких сочетаний их значений, при которых качественное поведение решения уравнения (динамика моделируемой системы регулирования) является удовлетворительным. В этом примере роль модели играет само дифференциальное уравнение, а в F хранятся процедуры подбора коэффициентов.

Что показывают приведенные примеры? Во-первых, что в ряде случаев, представление знаний об объекте в модели оказывается более компактным, чем их отражение в алгоритме, следствием чего является существенное упрощение описания системы управления и ее функционирования. Это положение носит довольно общий характер, что нашло свое отражение в появлении специальных названий для этих двух способов хранения знаний. Хранение знаний в моделях называют декларативным. представлением знаний, а хранение их в описании алгоритмов – процедурным представлением знаний. Эти термины особенно широко используются в области интеллектуальных систем и роботов.

Во-вторых, что появление модели M в структуре системы управления приводит к необходимости организовать поиск знаний в этой модели. Этот поиск осуществляется блоком F, и процедуры поиска становятся характерными для работы данного блока.

Удобство такого разделения информации между блоками M и F связано также и с тем, что сменить информацию в M гораздо легче, чем написать новые процедуры для блока F. Если набор процедур блока F достаточно богат, то можно ожидать, что такая модель управления за счет лишь небольших усилий, связанных с введением новой информации в M, может адаптиро-

ваться к новому объекту управления.

14.5 Типы систем управления: семиотические системы

Для F-M-систем предполагается, что содержимое блока M в процессе управления остается неизменным. Другими словами, предполагается, что априорная система знаний об объекте и протекающих в нем процессах обладает исчерпывающей полнотой и достоверностью. Для сложных объектов, эволюционирующих во времени, для которых критерии их функционирования и управления ими четко не сформулированы, это предположение неразумно. На самом деле содержимое блока M, безусловно, меняется в процессе работы системы управления. Это содержимое непрерывно обновляется, уточняется и пополняется.

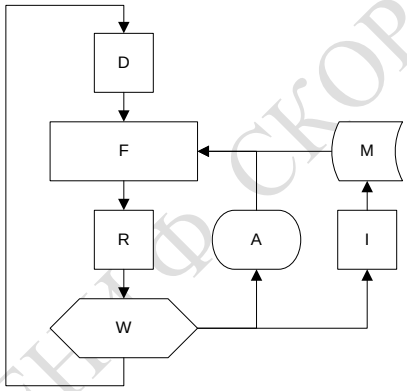


Рисунок 14.5 – Схема семиотической системы управления

В F-M системах предполагается, что M не изменяется, т.е достоверность информации достаточна для управления. Если W – объект управления эволюционирует во времени, критерии функционирования и управления четко не сформулированы, то содержимое M непрерывно наполняется и обновляется. Перестройка M происходит с помощью I – интерпретатора. I интерпретирует реакции W в терминах M. Эту задачу реализуют процедуры I: выделение причинно-следственных цепочек, обнаружение закономерностей, фиксация фактов заданного типа, специальные процедуры статистической обработки.

Тема 15 Ситуационное управление

15.1 Основная схема
15.2 Обучение системы ситуационного управления

15.3 Управление дислокационными операциями в орском порту
15.4 Управление отраслью заготовок

15.1. Основная схема ситуационного управления

Ситуационное управление было первым случаем практического использования логико-лингвистических моделей в управлении[2]. В настоящее время накоплен большой опыт построения систем, основанных на схеме, лежащей в основе ситуационного управления. Проиллюстрируем, как логико-лингвистические модели могут быть использованы при решении реальных задач.

Основная схема системы ситуационного управления показана на рис. 15.1. Поясним ее функционирование и особенности задач, решаемых отдельными подсистемами.

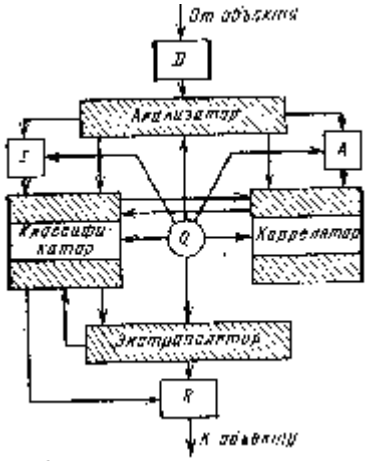


Рисунок 15.1 – Основная схема системы ситуационного управления

Блоки системы, названные на рис. 15.1 *«Анализатор»*, *«Классификатор»*, *«Коррелятор»* и *«Экстраполятор»*, в совокупности выполняют функции блоков ***F*** и ***M*** семиотической системы управления. На рис. 15.1 штриховыми линиями условно показана доля участия блока ***F***. После преобразования информации, поступающей от объекта управления, в описание наблюдаемой ситуации на реляционном или предикатном языке (в зависимости от принятого проектировщиком решения), «Анализатор» на основе

[2] Первая задача, на которой сформировался подход, названный впоследствии ситуационным управлением, была решена в 1967 г. Это была задача автоматизации диспетчерской службы на шлюзованных участках водных путей.

анализа поступившего описания определяет его дальнейшее движение в системе. Если поступившая информация содержит данные для пополнения модели знаний, то она передается в «Интерпретатор» ***I***. Если в поступившей информации имеются оценочные данные, то адресатом ее становится «Адаптатор» ***A***. Если во входной информации содержатся данные об изменении базовых параметров объекта или среды, в которой он функционирует, то эта информация передается непосредственно в «Коррелятор». Наконец, если на вход системы поступило чистое описание наблюдаемой ситуации, анализатор оценивает принадлежность этой ситуации к классу конфликтных. Под *конфликтными ситуациями* понимаются те наблюдаемые ситуации, появление которых на входе системы управления вызывает необходимость формирования управляющего воздействия на объект. Если такой необходимости нет, то ситуация называется неконфликтной. Такая ситуация, опознанная в «Анализаторе», в дальнейшем никуда не поступает. Конфликтные ситуации поступают в «Классификатор».

«Классификатор» хранит в своей памяти сведения о прошедших ситуациях в виде того иерархического «слоеного пирога», содержит в себе процедуры обобщения и конкретизации описаний, а также процедуры, связанные с функциями информационного банка. При поступлении на вход «Классификатора» конфликтной ситуации происходит проверка принадлежности ее к обобщенным классам самого верхнего уровня. Если такой класс найден и этому классу соответствует единственное решение по управлению, то это решение выдается непосредственно на выходной преобразователь ***R***. Этот преобразователь переводит коды императивов в управляющие сигналы, пригодные для передачи на объект управления. Если же для поступившей в «Классификатор» конфликтной ситуации находится лишь такое обобщенное описание, для которого имеется альтернативный выбор решения по управлению, либо поступившая ситуация вообще не входит ни в какое из имеющихся в «Классификаторе» обобщенных описаний, то, прежде чем выдать решение, «Классификатор» обращается к другим блокам. При наличии нескольких альтернативных решений окончательное решение о выборе управляющего воздействия переходит к «Экстраполятору».

«Экстраполятор» содержит набор процедур, среди которых имеются процедуры дедуктивного и индуктивного выводов. Кроме того, в «Экстраполяторе» могут содержаться и обычные имитационные процедуры, с помощью которых происходит «проигрывание» последствий принятия тех или иных решений на несколько шагов вперед. Процедуры планирования позволяют решать две задачи: строить многошаговое управление и оценивать эффективность тех или иных альтернативных решений. После выбора управляющего воздействия или плана многошагового воздействия «Экстраполятор» выдает его на объект управления и одновременно сообщает о своем

выборе «Классификатору», который использует эту информацию для улучшения данных в «слоеном пироге».

Наиболее трудным для управления является тот случай, когда поступившая ситуация никак не классифицируется в «Классификаторе». В этом случае информация о ней поступает в «Коррелятор», в котором хранятся знания о законах функционирования объекта, ограничениях на управляющие воздействия, подаваемые на объект, и целевые структуры типа ***RX***-кодов. На основании поступившей информации «Коррелятор» выбирает допустимое воздействие на объект управления и выдает информацию об этом выборе в «Классификатор». «Классификатор» по этому решению относит поступившую наблюдаемую ситуацию в те обобщенные описания, которым соответствует решение, сформированное в «Корреляторе». Если это обобщение не порождает других альтернативных решений, то «Классификатор» выдает информацию о принятом решении в ***R***. Если же в результате обобщения окажется, что в наблюдаемой ситуации можно принять и другие решения, отличные от рекомендованного «Коррелятором», то эти решения сообщаются последнему. «Коррелятор» проверяет допустимость новых решений. Все допустимые решения затем передаются в «Экстраполятор», который и производит среди них выбор окончательного решения по управлению.

15.2 Обучение системы ситуационного управления

Был описан процесс функционирования системы ситуационного управления на этапе ее работы. Однако, как и для всех открытых систем управления, этому этапу предшествует этап начального обучения системы. Основными процедурами этого этапа является процедура построения «слоеного пирога» в «Классификаторе» и процедура построения системы правил вывода в «Корреляторе». На этом этапе система ситуационного управления работает в режиме диалога с проектировщиками системы и специалистами-технологами. Этот диалог осуществляется через специальный блок обучения, который на рис. 15.1 обозначен через **Q**. Через этот же блок происходит первоначальное заполнение всех основных блоков априорной информацией, которую можно вложить в систему управления и без обучения. Процесс обучения, по существу, продолжается все время (частично за счет тех процедур, которые связаны с деформированием «слоеного пирога» из-за деятельности блоков системы в процессе работы).

Естественно, схема, приведенная на рис. 15.1, не отражает структуру реальной системы ситуационного управления. В реальной структуре функции указанных блоков могут выполнять системы, реализующие, наряду с указанными и другие функции. Кроме того, распределение обязанностей между «Классификатором» и «Коррелятором» может быть иным, чем было описано выше. Но в любой системе управления, построенной по типу систем ситуационного управления, обязательно должен присутствовать блок, в котором

хранится модель знаний о внешнем мире. В этой модели знаний должны храниться описания ситуаций, складывающихся на объекте управления, процедуры обобщения этих описаний, схемы принятия решений по управлению и процедуры их адаптации в реальных условиях, данные об ограничениях, присущих объекту с точки зрения управления им и т.п. Другими словами, эта модель знаний должна реализовать функции как «Коррелятора», так и «Классификатора». Столь же важно наличие процедур по определению конфликтности ситуации, независимо от того, где, в каком из блоков системы эти процедуры будут реализованы. Именно совокупность указанных процедур и обусловливает специфику систем ситуационного управления.

Вся информация внутри системы ситуационного управления хранится и обрабатывается различными процедурами, будучи записанной на некотором языке ситуационного управления. Этот язык может быть как реляционным, так и предикатным. Имеются примеры реальных систем, в которых оба этих типа описаний используются внутри системы.

Опишем нескольких конкретных систем ситуационного управления.

15.3 Управление дислокационными операциями в морском порту

Эта модель ситуационного управления разработана для управления перемещением судов в рыбном морском порту. В качестве базового порта для модели был выбран рыбный порт г. Калининграда.

Рассматриваемая система управления работает в режиме человек – ЭВМ. Диспетчер порта формулирует для ЭВМ задание-запрос на необходимость выполнения в порту дислокационных операций таких, как постановка, перестановка, перетяжка, кантовка, отшвартовка определенных объектов. Кроме того, диспетчер вводит в ЭВМ сведения об изменении структуры управляемого объекта, сообщая ей, например, сведения о закрытии тех или иных причалов на ремонт или о том, что какое-то судно по техническим причинам определенное время не может перемещаться по акватории порта (например, начало покраски борта).

Задачей системы является, во-первых, проверка возможности реализации заданий диспетчера, и во-вторых, если такая возможность имеется, поиск решения с учетом всех ограничений и возможной оптимизации решения по заранее введенным в систему критериям.

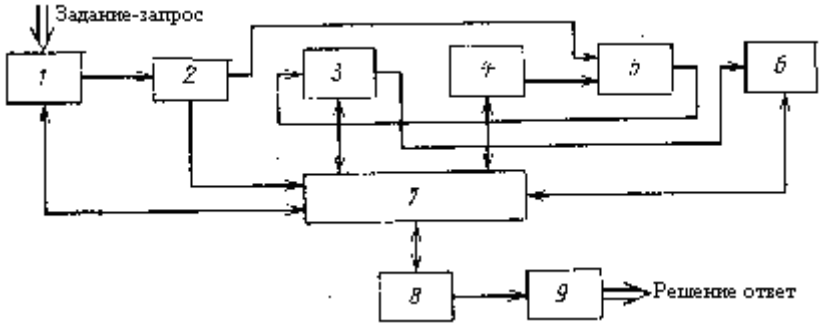


Рисунок 15.2 – Общая структура ситуационной системы управления перемещением судов

Общая структура ситуационной системы управления для – рассматриваемого объекта приведена на рис. 15.2. Опишем кратко работу блоков, указанных на этом рисунке.

В рассматриваемой системе используется следующий фрейм задания-запроса:

ЗАПРОС [<ОБЪЕКТ> <ИМЯ> <ВРЕМЯ ПРИХОДА> <ВРЕМЯ ОТХОДА> <ВРЕМЯ ГОТОВНОСТИ СУДНА К ГРУЗОВЫМ ОПЕРАЦИЯМ> <ВИД ТРЕБУЕМОЙ РАБОТЫ> <СИЛА ВЕТРА В ПОРТУ> <МЕТЕОУСЛОВИЯ> <УРОВЕНЬ ВОДЫВКАНАЛЕ> <ОСАДКА НОСОМ> <ОСАДКА КОРМОЙ> <ПОТРЕБНОСТЬ В ЛОЦМАНЕ> <ПОТРЕБНОСТЬ В БУКСИРЕ> <РАЗРЕШЕНИЕ>].

В этом фрейме некоторые роли сами представляют собой фреймы. Так роли ВРЕМЯ ПРИХОДА, ВРЕМЯ ОТХОДА и ВРЕМЯ ГОТОВНОСТИ К ГРУЗОВЫМ ОПЕРАЦИЯМ имеют фрейм одинаковой структуры, три позиции которого определяют месяц, число и часы со ответствующего времени. Роль РАЗРЕШЕНИЯ в качестве своих значений имеет: специальное разрешение портнадзора, разрешение пожарной инспекции, разрешение капитана порта. Роль ВИД ТРЕБУЕМОЙ РАБОТЫ в качестве значений принимает один из кодов 1 – 5 (соответствующих пяти типам работ с объектами в порту): постановка, перестановка, перетяжка, кантовка и отшвартовка.

Такая форма представления входной информации удобна для диспетчера и не требует от переводящей системы больших затрат на перевод поступающих текстов в реляционные описания. Этот перевод осуществляется в блоке ***1*** кроме того, этот блок реализует процедуры вызова в рабочие поля памяти 3 системы всех необходимых данных описаний объектов, текущих микроситуаций и макроситуаций состояния на причалах и в акватории порта и т. п. Кроме того, с помощью простых априорно заданных средств этот блок

пополняет описание ситуации за счет поступившей информации и информации, хранящейся в системе.

Блок ***2*** играет в значительной мере роль «Классификатора». В рассматриваемой системе априорно выделены и введены в систему 54 класса ситуаций (макроситуаций), каждому из которых соответствует некоторый шаблон решения. Блок ***2*** состоит из шести модулей. В первом из них проверяется допустимость дислокационных работ в акватории порта с точки зрения правил безопасности мореплавания. Восемь правил, хранящихся в памяти модуля, описывают эти ограничения. Пример правила этого типа: «Если судно необходимо переставить или перетянуть, или кантовать, или буксировать, или поставить первым, вторым или третьим бортом и ветер равен' 8 баллам или более, то запретить перестановку первым, вторым или третьим бортом, или отшвартовку судна». Если в текущей ситуации содержится структурный фрейм, соответствующий фрейму какого-либо из правил первого модуля, то система дает диспетчеру указание на невозможность выполнения его задания. Если же этого не происходит, то начинает работать второй модуль. Этот модуль проверяет возможность про ведения дислокационных работ по условиям безопасности стоянки судов. Во втором модуле имеются 16 таких правил-фреймов. При наличии запрета информация выдается диспетчеру, а при отсутствии запретов задание-запрос поступает, в зависимости от типа необходимой операции, в 3 – 6-й модули. Третий модуль проверяет возможность постановки судна к причалу. Число проверяемых правил 23. Четвертый модуль с помощью восьми правил проверяет возможность постановки судна в отстой. Пятый модуль работает с 15 правилами перетяжки, перестановки и кантовки судов. Наконец, модуль 6 проверяет применимость одного из трех правил проведения вспомогательных операций на судах (заправка топливом, водой и т. п.).

Шаблоны команд, вырабатываемых в модулях блока 2, посылаются в блок 5.

Блок ***4*** проверяет наличие определенных признаков и отношений между объектами, содержащимися в описании ситуации. Этот блок состоит из четырех модулей. Первый из них проверяет наличие нужных отношений и отсутствие запрещающих отношений между объектами класса «судно», например, проверяет отношение водоизмещения судов, когда их пришвартовывают друг к другу (по правилам, водоизмещение судна, стоящего первым бортом, должно быть больше водоизмещения судна, стоящего вторым бортом). Второй модуль проверяет наличие необходимых отношений между понятиями-классами «судно» и «причал», третий – между понятиями-классами «судно» и «судно портфлота», а четвертый – между понятиями-классами «причал» и «судно портфлота». Первый работает с шестью правилами, второй, третий и четвертый – каждый с двумя правилами.

При невозможности выполнения той или иной операции выдается сообщение диспетчеру. В блоке 5 шаблон команды, сформированный в блоке ***2***, заполняется конкретными данными, полученными из блока ***4***. Сформированные команды отсылаются в блок ***3***.

В блоке ***6*** происходит соединение. отдельных выработанных команд в единый план дислокационных работ. При этом учитывается допустимая последовательность выполнения отдельных команд и возможность уменьшения времени и улучшения других показателей работы диспетчерской службы порта.

Блок 7 является монитором системы. Он организует взаимодействие всех блоков в системе.

Блок ***8*** выполняет многочисленные функции. Примерами их могут служить: поиск свободной части акватории порта, поиск свободного причала, поиск свободного судна портового флота нужного типа, определение возможности одновременной обработки двух судов. Кроме того, в блоке ***8*** реализована имитационная модель, позволяющая в сомнительных случаях по заданию диспетчера имитировать процессы дислокационного типа.

Наконец, блок ***9*** преобразует решение либо на естественный язык для выдачи диспетчеру, либо на язык кодов команд для выдачи их на документоносители.

Описанная модель не содержит блоков, которые осуществляли бы обучение и пополнение модели. Эти блоки должны быть связаны с формированием условий на выбор лучшего решения из множества альтернативных решений. В том варианте, который показан на рис. 15.2, система управления может выдать диспетчеру все допустимые варианты выполнения задания, а ему останется выбрать наилучший, с его точки зрения, вариант.

15.4 Управление отраслью заготовок

Производство муки, различных хлебопродуктов и комбикормов связано с непрерывной работой различных предприятий, исходным сырьем для которых являются различные виды и сорта зерна. Заготовка и переработка зерна является основной задачей отрасли заготовки. Рассмотрим модель управления заготовкой и переработкой зерна на основе месячного республиканского плана. В качестве базового для модели рассмотрен регион обеспечивающий свои нужды в основном за счет привозного зерна. Это обстоятельство, а также ряд других факторов, приводит к тому, что поступление зерна для хранения и переработки на многочисленные предприятия отрасли вызывает частые конфликтные ситуации: то зерна слишком много, то его запасы приближаются к нижнему допустимому уровню. Так как причины такой нестабильности весьма многочисленны, то ситуационное управление может оказать пользу при оперативной корректировке месячных планов, по которым осуществляется управление отраслью.

Объект управления в рассматриваемом случае для ситуационного управления является типовым, так как его структура может быть представлена в виде *дискретной ситуационной сети* (ДСС).

Понятие ДСС играет в ситуационном управлении важную роль. Дискретная ситуационная сеть представляет собой сеть с вершинами трех типов: истоки, стоки и решатели, между которыми проводятся дуги. По ДСС перемещаются объекты, каждый из которых задается совокупностью признаков $<q_1,...,q_h,d_1,...,d_2>$. При этом признаки q_i являются статическими и не меняются при движении объекта по сети, а признаки d_i, могут меняться во времени (динамические признаки) или в решателях (ситуативные признаки). В истоках объекты порождаются по детерминированным или вероятностным законам либо возникают с неизвестными законами порождения. В стоках они исчезают из сети. Решатели первого типа представляют собой задержки на определенное время, в них меняются лишь временные параметры объектов, в частности, значения их динамических признаков. Активные решатели при нахождении в них объектов могут воздействовать на значения их ситуативных параметров, а также на выбор дальнейшего движения объекта по сети.

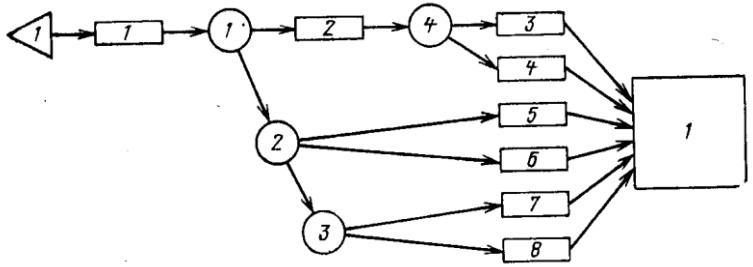


Рисунок 15.3 – Пример дискретной ситуационной сети

На рис.15.3 показана ДСС. В единственном истоке, который обозначен на этой схеме треугольником, порождаются объекты. В единственном стоке они исчезают. Сток обозначен на рисунке в виде квадрата. Прямоугольники соответствуют временным за держкам, а кружки – активным решателям. Интерпретацией этой ДСС может служить участок одноколейной железной дороги с разъездом на нем. Активные решатели в этом случае символизируют собой систему управления стрелками, так что каждый такой решатель соответствует одной стрелке. Его функцией является изменение маршрута движения объекта. Часть пассивных решателей не имеет внешнего управления и соответствует задержке движения объекта. Другая часть пассивных решателей является управляемой и соответствует решению диспетчера разъезда о задержке тех или иных объектов на разъезде.

Дискретные ситуационные сети характерны для всех транспортных систем, в частности и для задачи о дислокационных перемещениях в порту, ко-

торая была рассмотрена в предыдущем пункте. В задаче об управлении заготовкой и переработкой зерна ДСС строится следующим образом: вершины ДСС, соответствующие истокам, отождествляются с пунктами прибытия транспорта с зерном и пунктами порождения разного рода заявок на зерно и различные хлебопродукты. Зерноперерабатывающие предприятия играют роль активных решателей, а хлебоприемные предприятия – пассивных решателей. В качестве стоков можно рассматривать потребителей зерна и муки, производящих хлебопродукты и комбикормы, а также другие виды продукции.

Наблюдаемой ситуацией для системы управления является информация о состоянии ДСС, которая включает информацию о расположении объектов на сети и характеристики объектов и самих вершин сети.

Как и в предыдущей,- в этой модели реализуется режим человек – ЭВМ, где ЭВМ работает в режиме советчика диспетчерской службы отрасли. Задания и сообщения в систему поступают в виде ролевых фреймов определенной структуры. Примером сообщений, поступающих на вход системы управления, может служить следующее сообщение: [<ОПЕРАЦИЯ> (распределение); <ОБЪЕКТ> (зерно); <ИМЯ> (пшеница); <ТИП> (сильная); <КЛЕЙКОВИНА> (25) <СТЕКЛОВИДНОСТЬ> (60); <КОЛИЧЕСТВО> (15 000); <МЕСТОПОЛОЖЕНИЕ> (порт Батуми) <ВРЕМЯ ПРИБЫТИЯ> (12 мая 1979 г)]. Это сообщение во входном блоке 1 системы перерабатывается в описание в виде *RX-кода.* Дальнейшая работа системы управления иллюстрируется с помощью блок-схемы, показанной на рис. 15.4.

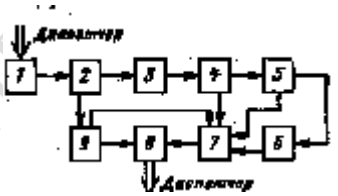


Рисунок 15.4 – Схема работы системы управления

Опишем работу блоков. В блоке *2* происходит анализ поступившего сообщения. После анализа либо в память системы заносится новая информация, либо начинает работать система принятия решений. В блоке *3* формируются ситуативные структуры. Ситуативные структуры представляют собой RX-коды, в которых некоторые вершины остаются незаполненными. Заполнение этих вершин (или позиций в соответствующем RX-коду ролевом фрейме) на основании информации, хранящейся в системе или присутствующей в наблюдаемой ситуации, и составляет суть формирования ситуативных структур. Альтернативность этого заполнения отражает многовариантность процесса принятия решений. Последующие три бло-

ка производят заполнение ситуативной структуры. В блоке *4* устанавливается приоритет классов объектов в соответствии с правилами приоритета, хранящимися в этом блоке. Эти правила получены в результате обработки экспертных мнений специалистов из управления элеваторной промышленностью Министерства заготовок рассматриваемого региона. Они устанавливают приоритет между зерноперерабатывающими предприятиями мельничного и комбикормового типов и хлебоприемными предприятиями. В блоке 5 определяется приоритетность выполнения тех или иных операций, связанных с размещением привозимого и перераспределением запасов хранимого зерна. Работа этого блока также сводится к проверке системы корреляционных правил, полученных в результате экспертизы специалистов. В блоке *6* устанавливается приоритет возможных решений (приоритет в классификации, которая осуществляется в блоке 7). Для этого также используются априорно заданные в системе правила. Примером может служить правило: «Если клейковина завозимого зерна не менее 20%, то его необходимо распределить в зерноперерабатывающее предприятие мельничного типа, работающее на сортовом помоле, если предприятие не находится в капитальном ремонте и там имеется свободная емкость для приема зерна». Роль классификатора выполняет блок 7, на вход которого поступают готовые ситуативные структуры, заполненные в блоках *4 и 5, а* также приоритетность проверки тех или иных решений, которая была найдена в блоке *6*.

Заметим, что описание наблюдаемой ситуации, задание ситуативных структур и корреляционных правил по форме согласованы между собой. Все они представлены в виде *RX-кодов* или синтагматических цепей. Признаки и отношения, участвующие в правилах, отражены в них структурно, как и в описании ситуаций. Поэтому проверка условий выполнимости правил сводится к проверке вхождения одной структуры в другую.

Для примера приведем запись в виде синтагматической цепи одного из правил, используемых в блоке *6*. Правило гласит: «Зерно сильной пшеницы можно распределять лишь в зерноперерабатывающее предприятие мельничного типа, работающее на сортовом помоле». Введем понятия-классы: B_1 – «зерно», B_2-- «зерноперерабатывающие предприятия мельничного типа». Пусть b_{22} означает понятие-класс «пшеница», b_{23} – понятие-класс «зерноперерабатывающее предприятие», модификатор n_7 – - «сильная», n_8 – «работающее на сортовом помоле», а императив p_4 – «распределитель». Тогда это правило имеет следующую запись:

$$\forall(b_{22} \in B_1)\forall(b_{23} \in B_2)[(b_{22}r_{20}n_7b_{22})(b_{23}r_{20}n_8b_{23}) \mid\Longrightarrow (b_{22}p_4b_{22})];$$

где отношение r_{20} означает отношение «объект-свойство».

Выработанные в порядке приоритетности решения из блока 7 поступают в блок *8,* который выполняет роль экстраполятора. В рассматриваемой системе экстраполяция осуществляется вперед на десятидневный интервал времени. Экстраполяция происходит на ДСС; законы опустошения запасов зерна на предприятиях априорно заданы.

Выбор оптимального варианта решения на основе экстраполяции на ДСС производится на основе таких заложенных в систему критериев, как качество помольной смеси, простой транспортных средств и учет транспортных издержек, связанных с доставкой сырьевых ресурсов. После выбора оптимального решения система выдает его диспетчеру. При этом возможен режим, когда диспетчеру выдается не только одно оптимальное решение, а все решения, близкие к нему. Окончательный выбор решения в этом случае выполняет сам диспетчер.

Если очередной текст, поступивший на вход системы, блоком *2* распознан не как задание, а как новая информация о структуре объекта управления или технологии управления, то блок *2* передает ее в блоки 7 и *8* для учета при классификации и экстраполяции.

Тема 16 Основные типы и конечные цели задач классификации

16.1 Комбинационные группировки
16.2 Основные типы и конечные цели задач классификации
16.3 Простая типологизация
16.4 Задачи связной неупорядоченной типологизации

Таблица 16.1 – Типы и конечные цели задач классификации

Тип задач классификации	Варианты прикладных целей исследования
1. Комбинационные группировки и их непрерывные обобщения.	Составление частотных таблиц и графиков, характеризующих распределения статически обследованных объектов по градациям или интервалам группирования описывающих их признаков.
2. Простая типологизация: выявление «стратификационной[3] структуры» множества статистически обследованных объектов, «нащупывание» и описание четко выраженных скоплений этих объектов в анализируемом многомерном пр-ве и построение правила отнесения нового объекта к одному из выявленных классов	2.1. До основной статистической обработки данных (построение регрессии и др.) добиваются расслоения множества на однородные порции данных 2.2. Выявление и описание расслоенной природы анализируемой совокупности статистически обследованных объектов с целью формирования плана выборочных обследований этой совокупности 2.3. Первый шаг в построении связных типологий

[3] от лат. stratum - слой

3. Связная неупорядоченная типологизация: исследование зависимостей между не поддающимися упорядочению классификациями одного множества объектов в разных признаковых пространствах одного из которых построено на результирующих признаках (характеризующих функционирование, поведение, здоровье), а другое – на описательных (отражающих условия функционирования и влияющих на результирующие признаки).	3.1. Прогноз экономико-социологических ситуаций (или отдельных показателей), включая задачу выявления типообразующих признаков (в том числе и латентных) 3.2. Диагностика в промышленности, технике, георазведке, медицине и т.д. 3.3. Автоматическое распознавание образов – зрительных, слуховых
4. Связная упорядоченная типологизация: модификация связной неупорядоченной типологизации. за счет дополнительного допущения, что классы, получаемые в пр-ве результирующих признаков, поддаются экспертному упорядочению по некоторому сводному (латентному) свойству: эффективности функционирования, качеству, степени прогрессивности поведения...	Построение и интерпретация единого (сводного) латентного признака – классификатора в виде функции от исходных описательных признаков: классификация химических элементов по заряду атомного ядра, классификация сельского населения по уровню их социально – экономического развития, построение фактора общей одаренности в педагогике и психологии.
5. Структурная типологизация: дополнение и развитие простой типологизации в направлении изучения и описания структуры взаимосвязей полученных классов, включая построение соответствующих иерархических систем (на классифицируемых элементах), анализ роли и места каждого элемента и класса в общей структурной классификационной схеме (она определяется составляющими классами и характеристиками их взаимодействия).	5.1 Классификация задач многоцелевого комплекса (научной программы, научного направления производственного комплекса) 5.2 Классификация элементов и подсистем по их функциональному назначению (производство в территориально-производственном комплексе) 5.3 Классификация лиц, принимающих решение, по роли и близости позиций в понимании ситуации и способе решения задачи 5.4 Классификация используемых признаков и анализ структуры связей между ними.
6. Классификация динамических траекторий развития систем: типологизация траекторий многомерных временных рядов $X(t) = (x^{(1)}(t),\ldots,x^{(p)}(t))$, среди $x^{(j)}(t)$ могут быть количественные и качественные.	Задачи: биологической статистики, анализа типов динамики семейной структуры, анализа типов динамики потребительского поведения семей.

16.1 Комбинационные группировки

Комбинационные группировки – это самый простой и исторически первый практический тип задач классификации. Пример табличного представления комбинационной группировки для двумерного случая может быть приведен в виде таблицы.

Заметим, что *непрерывным аналогом комбинационной группировки* является обычный переход от исходных наблюдений непрерывной случайной величины к «группированным» выборочным данным. Результат такого перехода представляется либо в виде таблицы, либо в виде графика *(гистограммы)*.

16.2 Основные типы и конечные цели задач классификации

Анализ примеров решения практических задач классификации в экономике, социологии, психологии, технике, медицине, геологии, археологии и других сферах практической и научной деятельности человека позволяет произвести определенную систематизацию этих задач в соответствии с их

основными типами и конечными прикладными целями исследования. Результат такого исследования приведен в таблице 16.1.

16.3 Простая типологизация

Наибольшее распространение получила простая типологизация. Она связана с выявлением структуры множества статистически обследованных объектов, с описанием четко выраженных скоплений этих объектов в анализируемом многомерном пространстве и построением правила отнесения нового объекта к одному из выявленных классов

16.4 Задачи связной неупорядоченной типологизации

Поясним методологическую общность задач 16.1-16.3 из таблицы 16.1: прогноза экономико-социологических ситуаций, диагностики и автоматического распознавания зрительных и слуховых образов. Для этого лежащую в основе их решения методологическую схему связной неупорядоченной типологизации представим следующим образом. Пусть в качестве исходных данных об объекте O_i (i = 1, 2, ...,*n)* имеем вектор *описательных* (объясняющих) признаков $X_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)})'$ (это, в частности, характеристики условий жизнедеятельности i-й обследованной семьи, значения параметров исследуемого технологического процесса, геофизических характеристик грунта и результаты обследований i-го пациента в задачах диагностики, геометрические или частотные характеристики распознаваемого образа) и некоторую информацию Y_i о том результирующем свойстве, по которому производиться классификация объектов (специфика социально-экономического поведения i-й семьи; наличие или отсутствие сбоев в i-м анализируемом технологическом процессе, месторождений полезных ископаемых на i-м обследованном участке, заболевания у i-го обследуемого пациента в задачах диагностики; конкретный содержательный смысл распознаваемого зрительного или слухового образа). Разница между задачами типа 16.1 и задачами 16.2 и 16.3 заключается в том, что в задачах прогноза экономико-социологических ситуаций информация Y_i, об исследуемом результирующем свойстве объекта *не является окончательной,* т. е. не задает однозначно, как это делается в задачах 16.2 и 16.3, образа (класса, типа), к которому относится этот объект. Эта информация в задачах типа 16.1 носит лишь *промежуточный характер* и представляется, как правило, в виде вектора результирующих показателей $Y=(y_i^{(1)}, y_i^{(2)}, ..., y_i^{(q)})'$. Поэтому в отличие от задач 16.2 и 16.3 (в которых уже «на входе» задачи имеем распределение анализируемых объектов-векторов X_i по классам, что и составляет так называемую «обучающую выборку») в задачах типа 16.1 нужно предварительно осуществить простую типологизацию множества объектов $\{O_i\}$ *(i* = 1, ..., *n)* в пространстве результирующих показателей и лишь затем использовать полученные в результате этой типологизации классы в качестве обучающих выборок для

построения классифицирующего правила в пространстве описательных признаков П*(X)*.

«На выходе» задач типа 16.1 – 16.3 должны быть: 1) набор наиболее информативных объясняющих переменных (так называемых *типообразующих признаков)* $z^{(1)}(X)$, $z^{(2)}(X)$, ..., $z^{(p')}(X)$, которые либо отбираются по определенному правилу из числа исходных описательных признаков $x^{(1)}$, $x^{(2)}$, ..., $x^{(p)}$, либо строятся в качестве некоторых их комбинаций; 2) правило отнесения *(дискриминантная функция, классификатор)* каждого нового объекта O*, заданного значениями своих описательных признаков X*, к одному из заданных (или выявленных в процессе предварительной простой типологизации) в пространстве П(Y) классов или образов. При этом типообразующие признаки $Z=(z^{(1)}(X), z^{(2)}(X), ..., z^{(p')}(X))'$ и искомое правило классификации должны быть подобраны таким образом, чтобы обеспечивать наивысшую (в определенном смысле) точность решения задачи отнесения объекта к одному из анализируемых классов по заданным значениям его описательных признаков X.

Из сформулированных выше конечных целей классификации видно, что тематику разбиения многомерных данных на однородные (в определенном смысле) группы подчас трудно отделить от *задач снижения размерности исследуемых данных.* Однако прикладные цели методов снижения размерности не исчерпываются сформулированной выше задачей перехода от исходного набора описательных признаков $x^{(1)}, ..., x^{(p)}$ к существенно более скромному (по численному составу) набору так называемых типообразующих признаков $z^{(1)}(X)$, $z^{(2)}(X)$, ..., $z^{(p')}(X)$, которые являются наиболее характерными, наиболее определяющими с точки зрения полноты и точности разбиения исследуемых объектов на классы.

Тема 17 Типологизация математических постановок задач классификации

17.1 Типологизация математических постановок задач классификации

17.2 Статистический подход в распознавании образов

17.3 Постановка задачи лингвистического подхода в распознавании образов

17.4 Схема лингвистического подхода в распознавании образов

17.1 Типологизация математических постановок задач классификации

Целесообразность и эффективность применения тех или иных методов классификации и снижения размерности гак же, как их предметная осмыс-

ленность, обусловлены конкретизацией базовой математической модели, т. е. математической постановкой задачи. Определяющим моментом в выборе математической постановки задачи является ответ на вопрос, *на какой исходной информации строится модель.* При этом исходная информация складывается из двух частей: 1) *из априорных сведений об исследуемых классах;* 2) *из информации статистической, выборочной,* т. е. так называемых *обучающих или частично обучающих*.

Априорные сведения об исследуемых генеральных совокупностях относятся обычно к виду или некоторым общим свойствам закона распределения исследуемого случайного, вектора X в соответствующем пространстве и получаются либо из теоретических, предметно-профессиональных соображений о природе исследуемого объекта, либо как результат предварительных исследований. Получение выборочной исходной информации в экономике и социологии, как правило, связано с организацией системы экспертных оценок или с проведением специального предварительного этапа, посвященного решению задачи простой типологизации анализируемых объектов в пространстве результирующих показателей.

Классификация задач разбиения объектов на однородные группы (в зависимости от наличия априорной и предварительной выборочной информации) и соответствующее распределение описания аппарата решения этих задач представлены в таблице 17.1.

17.2 Статистический подход в распознавании образов

В распознавании образов используются статистический и лингвистические подходы. Статистический подход в распознавании образов используется тогда, когда прямая идентификация образа или класса невозможна. Детерминированный или лингвистический подход – когда возможна прямая идентификация образа. Например, по числу углов многоугольника можно определить его тип.

Представление данных на статистическом языке иногда оказывается затруднительным. Эти трудности преодолеваются иногда за счет многошаговости процедуры путем попеременного использования статистического и эвристического подхода.

Статистические методы основываются на минимизации ошибки классификации (ε). Эта ошибка представляет собой вероятность неправильной классификации поступившего на распознавание произвольного *k*- мерного объекта *x:*

Таблица 17.1 – Типологизация математических постановок задач классификации

Априорные сведения о классах (генеральных совокупностях)	Предварительная выборочная информация		
	нет информации	есть частично обучающие выборки	есть обучающие выборки
Некоторые самые общие предположения о законе распределения исследуемого вектора: гладкость, сосредоточенность внутри ограниченной области и т.п.	Классификация без обучения: кластер-анализ, таксономия, распознавание образов «без учителя», иерархические процедуры классификации	Методы кластер-анализа, дополненные основанным на частично обучающих выборках выбором начальных приближений числа и центров классов, их ковариационных матриц	Непараметрические методы дискриминантного анализа
Различаемые генеральные совокупности заданы в виде параметрического семейства законов распределения вероятностей (параметры неизвестны)	Интерпретация исследуемой генеральной совокупности как смеси нескольких генеральных совокупностей. «Расщепление» этой смеси с помощью методов оценивания неизвестных параметров	Методы расщепления смеси, дополненные оценками, полученными из частично обучающих выборок. Модификация методов кластер-анализа	Параметрические методы дискриминантного анализа
Различаемые генеральные совокупности заданы однозначным описанием соответствующих законов распределения	Классификация при полностью описанных классах: различение статистических гипотез	Обучающие выборки не нужны	

$$\varepsilon = \sum_{i=1}^{L} q_i \, p_{rob}(Д(x) \neq l / x \in классу \quad l) \qquad (17.1)$$

L – число классов.

$Д(x)$ – функция, выполняющая классификационное решение $(1,\ldots,L)$;

q_i – априорная вероятность принадлежности произвольного объекта x классу l $\{p_{rob}(x \in классу \quad l)\}$.

Иногда минимизируются ожидаемые потери

$$\sum_{l=1}^{L} \sum_{l'=1}^{L} C(l/l') q_{l'} \, p_{rob}(Д(x) = l / x \in кл \; e') \qquad (17.2)$$

$C(l/l')$-потери, связанные с отношением объекта x к классу l в то время как $x \in l'$.

Можно доказать, что (17.1) достигает *min*, если

$$Д(x) = l, \qquad если \qquad q_l f_l(x) > q_{l'} f_{l'}(x), \forall l \neq l' \qquad (17.3)$$

где $f_{l'}(x)$ - плотность распределения класса l в k – мерном признаковом пространстве. Это бейесово правило классификации.

Пример применения правила классификации вида (9.3) для $k = 1, L = 3$ приведен на рисунке 17.1: *Д(x)=2* для *x* принадлежащего средней области рисунка.

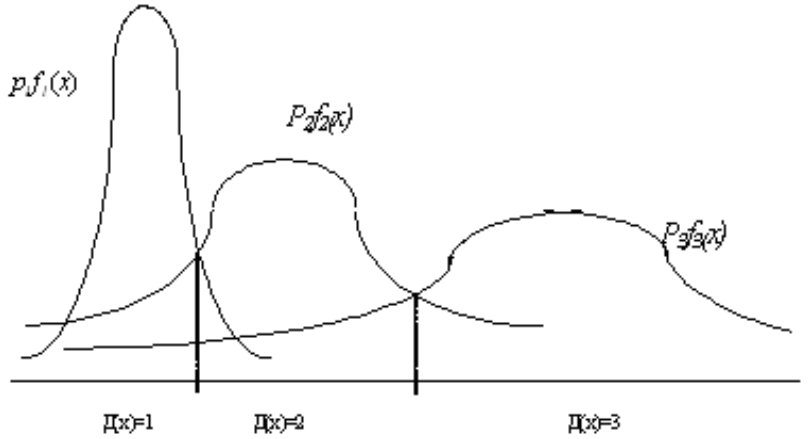


Рисунок 17.1 – Пример статистического подхода в распознавании образов

17.3 Постановка задачи лингвистического подхода в распознавании образов

В лингвистическом подходе к распознаванию признаками служат подобразы, называемые непроизводными элементами, а также отношения между ними, характеризующие структуру образа.

Для описания образов через непроизводные элементы и их отношения используется «язык» образов. Правила такого языка, позволяющие составить образы из непроизводных элементов называются грамматикой. При этом образ представляется некоторым предложением в соответствии с действующей грамматикой.

Для распознавания некоторого образа необходимо: определить его непроизводные элементы и отношения между ними; провести синтаксический анализ для установления согласования с грамматикой (этот анализ называется грамматический разбор).

Синтезировать грамматику можно, опираясь на априорные сведения об образах или на результаты анализа некоторого конечного множества репрезентативных образов (вывод грамматики).

17.4 Схема лингвистического подхода в распознавании образов

Схема лингвистического подхода в распознавании образов приведена на рис.17.2.

Грамматика образов может быть использована: для порождения предложений, представляющий некоторый образ; для грамматического разбора предложений, цель которого состоит в определении соответствия их структуры применяемой грамматике.

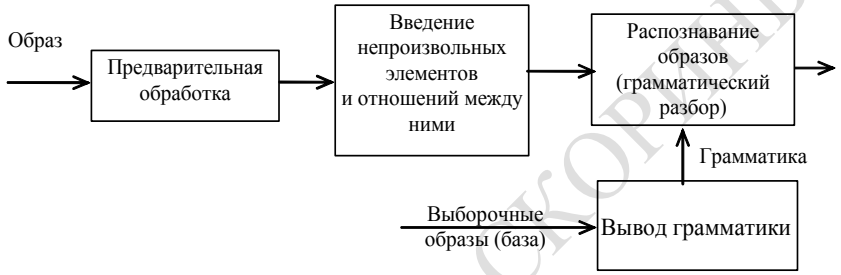


Рисунок 17.2 – Схема лингвистического подхода в распознавании образов

Тема 18 Типологизация математических постановок задач снижения размерности

18.1 Форма задания исходной информации

18.2 Тип оптимизируемого критерия информативности искомого набора признаков

18.3 Класс допустимых преобразований исходных признаков

18.4 Типологизация математических постановок задач снижения размерности

18.1 Форма задания исходной информации

Математическая модель, лежащая в основе построения того или иного метода снижения размерности, включает в себя обычно три основных компонента. Первый связан с формой задания исходной информации**.** Речь идет об ответе на следующие вопросы: а) в каком виде (т. е. в виде объект-признак или объект-объект, или еще каком-либо) задана описательная информация об объектах? б) имеется ли среди исходных статистических данных обучающая информация, т.е. какие-либо сведения об анализируемом ре-

зультирующем свойстве? в) если обучающая информация присутствует в исходных статистических данных, то в какой именно форме она представлена? Это могут быть, в частности, в привязке к объекту O^i (i = 1, 2, ..., n): значения «зависимой» количественной переменной («отклика») y_i, в моделях регрессии; номер однородного по анализируемому свойству класса, к которому относится объект O_i в задаче классификации; порядковый номер (ранг) объекта O_i, в ряду всех объектов, упорядоченных по степени проявления рассматриваемого свойства, в задачах анализа предпочтений и построения упорядоченных типологизаций; наконец, значения $Y_I = (y^{(1)},...,y^{(q)})'$ набора результирующих признаков, характеризующих анализируемое в классификационной задаче свойство.

18.2 Тип оптимизируемого критерия информативности искомого набора признаков

Пусть $I_{p'}(Z)$ – тип оптимизируемого критерия информативности искомого набора признаков $Z = (z(1), ..., z^{(p')})'$. Критерий информативности может быть ориентирован на достижение разных целей.

Следует выделить целый класс *критериев автоинформативности,* т.е. критериев, оптимизация которых приводит к набору вспомогательных переменных $Z=(z^{(1)}, ..., z^{(p)})'$, позволяющих максимально точно воспроизводить (в том или ином смысле, в зависимости от конкретного вида критерия) информацию, содержащуюся в описательном массиве данных типа (2.1) или (2.2). Если описательная информация представлена в виде матрицы (2.1), то речь идет о максимально точном восстановлении *pxn* значений исходных переменных $x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)}$ по значениям существенно меньшего числа $(p'xn)$ вспомогательных переменных $z_i^{(1)}, ..., z_i^{(p')}$. Если же описательная информация представлена в виде матрицы попарных сравнений объектов (2.2), то речь идет о максимально точном воспроизведении n^2 элементов этой матрицы a_{ij} (i, j=1, ..., *n)* по значениям существенно меньшего числа $(p'xn)$ вспомогательных переменных $z_i^{(1)}, ..., z_i^{(p')}$ $(i = 1, ..., n)$.

Будем называть *критериями внешней информативности* (имеется в виду информативность, внешняя по отношению к информации, содержащейся в описательном массиве (2.1) или (2.2)), такие критерии $I_{p'}(Z)$, которые нацелены на поиск экономных наборов вспомогательных переменных Z (X) $=(z^{(1)}(X), \ldots z^{(p')}(X))'$ обеспечивающих максимально точное воспроизведение (по значениям Z, а значит в конечном счете по значениям X) информации, относящейся к *результирующему* признаку (варианты ее задания перечислены выше, в п.18.1)).

18.3 Класс допустимых преобразований исходных признаков

Вспомогательные признаки $Z = (z^{(1)}, ...,z^{(p')})'$ в случае представления исходной описательной информации в форме матрицы (2.1) конструируются в

виде функций от X, т. е. Z=Z (X). Как обычно в таких ситуациях, чтобы обеспечить содержательность и конструктивную реализуемость решения оптимизационной задачи

$$I_{p'}(Z(X)) \rightarrow \underset{Z(X)}{\text{extr}},$$

следует предварительно договориться об ограниченном классе допустимых решений L(X), в рамках которого эта оптимизационная задача будет решаться. Очевидно, от выбора L(X) будет существенно зависеть и получаемое решение $Z(X) = (z^{(1)}(X), ..., z^{(p')}(X))$ упомянутой оптимизационной задачи.

Итак, следуя предложенной выше логике, типологизацию задач снижения размерности необходимо произвести по трем «входам» (или «срезам»): форме задания исходной информации, типу (смыслу) оптимизируемого критерия информативности и классу допустимых преобразований исходных переменных. На рисунке 18.1 типологизация задач снижения размерности эти принципы реализованы в упрощенном виде за счет следующих двух практических соображений: 1) подавляющее большинство методов снижения размерности базируется на *линейных* моделях, т. е. класс допустимых преобразований *L(X)* – это класс линейных (как правило, подходящим образом нормированных) преобразований исходных признаков $x^{(1)},...,x^{(p)}$); 2) спецификация формы задания исходной информации связана со спецификацией смысловой нацеленности критерия информативности, а поэтому их удобнее давать в общей графе.

18.4 Типологизация математических постановок задач снижения размерности

Данная на рисунке 18.1 типологизация, как и всякая иная классификация, не претендует на исчерпывающую полноту.

Тема 19 Кластерный анализ

19.1 Постановка задачи

Кластерный анализ трактуется как нестатический метод исследования и не связывают с задачами статистической классификации.

Пусть имеется смешанная выборка $X_1,\ldots,X_n \in \Pi^p$, и пусть она разбита на подмножество сходных между собой элементов. Возникает задача найти разбиение $S = \{X_1,\ldots,X_m\}$ m может быть не известна.

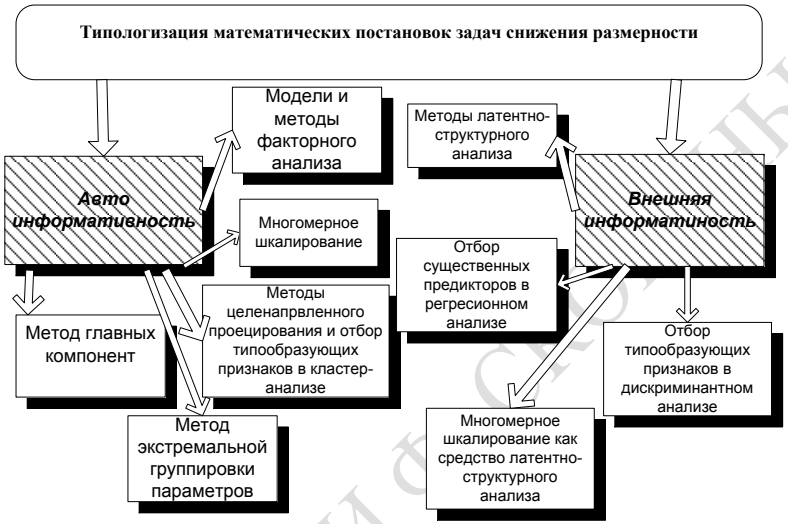


Рисунок 18.1- Типологизация математических постановок задач снижения размерности

Для расщепления смешанной выборки необходимо ввести меру близости элементов и групп элементов.

Мера близости может отражать метрические свойства пространства *X* или быть мерой множества. Она может быть задана самой функцией $\mu(x_i,x_j),\ldots,\mu(x_1,\ldots,x_n)$ от 2-х до *n* аргументов. Значения $\mu(x_1,\ldots,x_n)$ характеризуют степень близости ее аргументов. Задание конкретного вида меры близости зависит от специфики производимого исследования.

Обычно из физических соображений удается задать меру попарной близости элементов $\mu(x_i,x_j)$. Из попарной близости могут получиться различные варианты мер групповой близости.

Например диаметр множества *X* $\sup \mu(x_i,x_j) = \mu(X_k) \qquad x_i,x_j \in X_k$ или усреднение $\mu(x_i,x_j)$

$$\left[\frac{1}{nk}\sum_{x_i\in X_k}\sum_{x_j\in X_k}\mu^t(x_i,x_j)\right]^{\frac{1}{t}}=\mu(x_k),\ -\infty<t<+\infty \qquad (19.1)$$

Возможен подход, не использующий явно меру групповой близости. Тогда выделение классов происходит методом последовательного объединения элементов и групп элементов.

В приложениях встречаются задачи, когда число классов не известно. Сами классы представлены смешанной выборкой. О природе смешанной выборки есть некоторые априорные предположения. Необходимо построить алгоритм обработки смешанной выборки для расщепления ее на классы. Процесс расщепления смешанной выборки и последующее построение распределений для полученных классов называют генерацией гипотез (выделение и описание классов). 1-я схема генерации гипотез основана на использовании «внутренних» свойств смешанной выборки, например меры близости между отдельными элементами выборки или группами элементами выборки. Набор методов выделения классов, использующих внутренние свойства смешанной выборки, относится к кластер анализу. 2-я схема использует свойства смешанной выборки, характеризующие её как целое. Например: число мод выборочной плотности, свойства ковариационной матрицы, ранговые свойства выборки. Смешанная выборка как бы рассматривается извне, при этом могут обнаружиться неоднородности, которые позволят выделить классы. Среди таких методов, основанных на «внешних» свойствах смешанной выборки, факторный анализ, метод главных компонент, метод смесей.

19.2 Измерение близости объектов

Проблема измерения близости объектов неизбежно возникает при первичной статистической обработке. При этом возникают две трудности: неоднозначность выбора способа нормировки и определение расстояния между объектами.

Наиболее известными для количественных шкал являются расстояния: линейное, евклидово, обобщенное степенное, Махаланобиса.

19.3 Измерение близости между классами

На первом шаге, когда каждый объект представляет собой отдельный кластер, расстояния между этими объектами определяются выбранной мерой. Однако когда связываются вместе несколько объектов, возникает вопрос, как следует определить расстояния между кластерами? Другими словами, необходимо правило объединения или связи для двух кластеров. Здесь имеются различные возможности: например, есть возможность связать два кластера вместе, когда любые два объекта в двух кластерах ближе друг к другу, чем соответствующее расстояние связи. Другими словами, использу-

ется «правило ближайшего соседа» для определения расстояния между кластерами; этот метод называется методом *одиночной связи*. Это правило строит «волокнистые» кластеры, т.е. кластеры, «сцепленные вместе» только отдельными элементами, случайно оказавшимися ближе остальных друг к другу. Как альтернативу можно использовать соседей в кластерах, которые находятся дальше всех остальных пар объектов друг от друга. Этот метод называется метод *полной связи*. Существует также множество других методов объединения кластеров, подобных тем, что были рассмотрены.

Одиночная связь (метод ближайшего соседа). Как было описано выше, в этом методе расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Это правило должно, в известном смысле, нанизывать объекты вместе для формирования кластеров, и результирующие кластеры имеют тенденцию быть представленными длинными «цепочками».

Полная связь (метод наиболее удаленных соседей). В этом методе расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. «наиболее удаленными соседями»). Этот метод обычно работает очень хорошо, когда объекты происходят на самом деле из реально различных «рощ». Если же кластеры имеют в некотором роде удлиненную форму или их естественный тип является «цепочечным», то этот метод непригоден.

Невзвешенное попарное среднее. В этом методе расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них. Метод эффективен, когда объекты в действительности формируют различные «рощи», однако он работает одинаково хорошо и в случаях протяженных («цепочного» типа) кластеров.

Взвешенное попарное среднее. Метод идентичен методу *невзвешенного попарного среднего*, за исключением того, что при вычислениях размер соответствующих кластеров (т.е. число объектов, содержащихся в них) используется в качестве весового коэффициента. Поэтому предлагаемый метод должен быть использован (скорее даже, чем предыдущий), когда предполагаются неравные размеры кластеров.

Невзвешенный центроидный метод. В этом методе расстояние между двумя кластерами определяется как расстояние между их центрами тяжести.

Взвешенный центроидный метод (медиана). Этот метод идентичен предыдущему, за исключением того, что при вычислениях используются веса для учёта разницы между размерами кластеров (т.е. числами объектов в них). Поэтому, если имеются (или подозреваются) значительные отличия в размерах кластеров, этот метод оказывается предпочтительнее предыдущего.

Метод Варда. Этот метод отличается от всех других методов, поскольку он использует методы дисперсионного анализа для оценки расстояний между кластерами. Метод минимизирует сумму квадратов для любых двух (ги-

потетических) кластеров, которые могут быть сформированы на каждом шаге.

19.4 Подходы в кластерном анализе

Существует огромное количество алгоритмов кластер – анализа. Они отличаются вычислительными приемами и концепциями. Точной постановки задачи кластер – анализа нет. Единая теория находится в стадии становления. Для создания теории необходимо уметь учитывать разные конфигурации точек и предъявляемые к разбиениям требования.

Требования к хорошей классификации формируются двумя способами: 1) в терминах определений отдельных кластеров; 2) в виде некоторого функционала, оптимальное значение которого соответствует лучшей классификации.

На рисунке 19.1 приведены сочетания классов на плоскости, которые трудно четко разделить, так как необходимо учитывать разнообразные обстоятельства: расстояние между точками класса *C* больше , чем межклассовые расстояния некоторых точек в классах *B* и *C*, среднее значение в классах *E* и *F, K* и *H* одинаковы; классы *P* и *Q* соединены цепочкой, которую надо выделить.

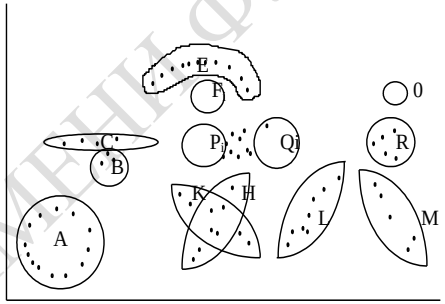


Рисунок 19.1 – Примеры сочетания классов на плоскости

Границы классов на рисунке приведены именно такими в соответствии с интуитивными представлениями о том, что кластер – скопление точек-представляет собой некоторую целостность (образ), чем-то отличающийся от другого скопления точек. Причем кластеры могут касаться друг друга (*B* и *C, L* и *M*) и пересекаться (*K* и *H*). Различать подобные кластеры формальным образом трудно. Это означало бы машинную реализацию чисто человеческого процесса распознавания образов.

Многолетние попытки решить задачу «структурной классификации» привели к тому, что традиционную для статистики проблему выделения од-

нородных групп стали называть распознаванием образов без учителя (самообучением).

Основой 1-го направления решения задачи структурной классификации является: (1) формулировка понятия кластера в смысле такого скопления точек, которое обладает некоторым свойством, например, в котором среднее межточечное расстояние меньше среднего расстояния от данных точек до остальных; (2) разбиение совокупности на части, представляющие собой кластер в смысле п.(1).

Алгоритмы, ориентированные на кластеры с заранее заданными свойствами, называют процедурами прямой классификации. Сам подход называют эвристическими, но некоторые алгоритмы находят локальный экстремум определенному функционалу.

Подавляющая часть практических исследований выполнена с помощью этих алгоритмов.

Требования к хорошей классификации предъявляют не только в терминах определения отдельных кластеров. Они могут быть сформулированы в виде некоторого функционала, экстремальное значение которого соответствует наилучшей классификации. Это второе оптимизационное направление. Здесь возникают чисто математические проблемы: определение свойства функционала, путей достижения оптимума, трудоемкости алгоритма. Редко в этих задачах нужен глобальный оптимум.

В кластер – анализе нет четкой связи между функционалом и действительной целью исследователя, т.к. она не может быть точно сформулирована в условиях исходной неопределенности в данных.

Разные функционалы качества должны давать разные результаты. Так как истинная структура данных неизвестна, желательно работать с несколькими функционалами. Если функционалы качества дают сильно отличающиеся классификации, надо полагать, что структура данных не вполне отчетлива. Но если классификации похожи, то скорее всего выявлена реальная структура.

Эти два подхода: прямой и оптимизационный являются разными способами формализации представления о хорошей классификации.

Было выявлено, что некоторые эвристические процедуры оптимизируют определенные функционалы. Резкой границы между подходами нет, наоборот есть предположение, что каждая прямая процедура на самом деле доставляет экстремум какой-то функции, которую надо отыскать. И каждому функционалу соответствует свое определение кластера.

Тема 20 BPwin – средство концептуального моделирования бизнес-процессов предприятия

20.1 Два способа улучшения бизнес-процесса: непрерывное усовершенствование и реинжиниринг

20.2 Понятие консалтинга в области информационных технологий

20.3 CASE-технологии – методологическая и инструментальная база консалтинга

20.4 BPwin – средство концептуального моделирования бизнес-процессов предприятия

20.1 Два способа улучшения бизнес-процесса: непрерывное усовершенствование и реинжиниринг

Что такое бизнес-процесс, зачем и как его усовершенствовать. Приведем примеры бизнес-процессов: заказ одежды у компании «товары – почтой», регистрация новых телефонных услуг на вашей АТС, разработка новых товаров, строительство нового жилища и т. д. Если вы когда-нибудь стояли в очереди за продуктами, то поймете, что такое необходимость совершенствования процесса. В данном случае процесс – это продвижение в очереди, а цель процесса – заплатить за покупку и унести ее с собой. Процесс начинается, когда вы занимаете очередь, а заканчивается, когда берете чек и уходите из магазина. Вы – клиент (у вас деньги и вы пришли за продуктами), а магазин – поставщик. Этапы процесса – это ваши и продавцов действия по проведению указанной операции.

Таким образом, бизнес-процесс – это комплекс различных действий, преобразующих ряд данных на входе в ряд данных на выходе (товары или услуги) для другого человека или процесса, с использованием людей и оборудования. Мы все заняты процессами и в разное время исполняем роли то клиента, то поставщика.

«Непрерывное совершенствование бизнес-процессов». Совершенствование бизнес-процессов имеет первостепенное значение для предприятий, желающих сохранить конкурентоспособность на рынке. В течение последних 10-15 лет компании вынуждены совершенствовать свои бизнес-процессы, потому что клиенты требуют все лучших товаров и услуг. А если не получают от поставщика желаемого, могут выбирать из многих других (отсюда и конкуренция среди предприятий).

Основные этапы улучшения бизнес-процессов в традиционном варианте выглядит следующим образом. Начинаем с составления описи того, что имеем на сегодняшний день; решаем, как оценивать процесс исходя из потребностей клиента; выполняем процесс, оцениваем результаты, а затем на основе полученных данных выясняем возможности для улучшений. Вносим изменения и анализируем новый процесс. Этот цикл повторяется раз за разом и называется «непрерывным совершенствованием процессов», «совершен-

ствованием бизнес-процессов», «функциональным совершенствованием процессов» и т. д.

Реинжиниринг. Приведенная выше модель совершенствования бизнес-процессов эффективна для постепенных, накопительных улучшений. Однако за последние 10 лет в силу ряда причин возникла необходимость ускорить эту работу. Наиболее очевидная причина – технологическая. Современные технологии (например, Интернет) стремительно открывают новые возможности, тем самым, поднимая планку соревнования и требуя коренного улучшения бизнес-процессов.

Другая очевидная тенденция – увеличение открытости мировых рынков и объема свободной торговли. На рынок приходит все больше компаний, ужесточается конкуренция. Серьезные перемены нужны уже лишь для того, чтобы не сдать позиции. Для многих предприятий это буквально вопрос жизни и смерти. Таким компаниям требуются не постепенные изменения, а прорыв, и немедленно. Редкое предприятие может позволить себе роскошь постепенных преобразований. Один из новых методов стремительно менять и существенно улучшать бизнес-процессы – реинжиниринг (Business Process Reengineering – BPR).

BRP по своей философии отличается от постоянного совершенствования процессов. В своей радикальной форме он вообще не берет в расчет существующие процессы: раз они не работают, значит испорчены – забудь о них и начни сначала. Такой подход «с чистого листа» позволяет отстраниться от настоящего и сосредоточиться на будущем, задать себе вопрос: как должен выглядеть процесс? Каким хотели бы его видеть мои клиенты? Каким хотели бы его видеть другие сотрудники? Как это делают первоклассные компании? Чего мы могли бы достичь с помощью новой технологии?

Этапы модели реинжиниринга: выяснить рамки проекта; научиться у других; поставить цели; спланировать переход; применить. Т.е. сначала задается объем и цели проекта по реинжинирингу, затем проходит процесс обучения. На этой базе можно выстроить перспективу и разработать новые бизнес-процессы. Определив, «как надо», можно создать план действий исходя из разницы между существующими процессами, технологиями и структурами и теми, которых нужно достичь. В дальнейшем все зависит от реализации.

Таким образом, существуют два полярных способа улучшить бизнес-процессы: непрерывное усовершенствование и реинжиниринг. Разница между ними в том, что принимается за основу (существующий процесс или «чистый лист»), а также в величине и скорости итоговых изменений.

Со временем эти две крайности, то есть постепенное улучшение и реформа «скачком», дали много производных. Все они – попытка решить проблему глобальных преобразований на предприятии. Трудно найти единый подход, точно отвечающий конкретным требованиям компании. Задача со-

стоит в том, чтобы понять, какой метод в каком случае выбрать и как применить, чтобы достичь искомого результата.

20.2 Понятие консалтинга в области информационных технологий

Термин консалтинг является каким-то аморфным и неконкретным. Практически каждая фирма, работающая на рынке информационных технологий, заявляет о предоставлении ею неких консалтинговых услуг. Что же следует понимать под консалтингом на самом деле? Какие требования предъявляются к консалтинговым структурам? Постараемся ответить на эти вопросы.

Консалтинг – это деятельность специалиста или целой фирмы, занимающихся стратегическим планированием проекта, анализом и формализацией требований к информационной системе, созданием системного проекта, иногда – проектированием приложений. Но все это до этапа собственно программирования или настройки каких-то уже имеющихся комплексных систем управления предприятием, выбор которых и осуществляется на основе системного проекта. И уж ни в коей мере сюда не входит системная интеграция. Консалтинг предваряет и регламентирует названные этапы.

Фактически консультантом выполняется два вида работ. Прежде всего, это элементарное наведение порядка в организации: бизнес-анализ и реструктуризация (реинжиниринг бизнес-процессов). Это направление получило название «бизнес-консалтинг». В конечном итоге речь, разумеется, идет об автоматизации, однако если мы будем автоматизировать существующий хаос, царящий на российских предприятиях, то в итоге получим не что иное как «автоматизированный хаос».

Любая организация – это довольно сложный организм, функционирование которого одному человеку понять просто невозможно. Руководство в общих чертах представляет себе общий ход дел, а клерк досконально изучил только свою деятельность, уяснил свою роль в . сложившейся системе деловых взаимоотношений. Но как организация функционирует в целом, не знает, как правило, никто. И именно деятельность, направленная на то, чтобы разобраться б функционировании таких организмов, построить соответствующие модели и на их основе выдвинуть некоторые предложения по поводу улучшения работы некоторых звеньев, а еще лучше – бизнес-процессов (деятельностей, имеющих ценность для клиента) считается бизнес-консалтингом.

Другой вид работ – собственно системный анализ и проектирование. Выявление и согласование требований заказчика приводит к пониманию того, что же в действительности необходимо сделать. За этим следует проектирование или выбор готовой системы так, чтобы она в итоге как можно в большей степени удовлетворяла требованиям заказчика.

Кроме того, важный элемент консалтинга – формирование и обучение рабочих групп. Здесь речь идет не только о традиционной учебе – любые проекты, модели должны в итоге кем-то сопровождаться. Поэтому сотрудники предприятия с самого начала участвуют в проекте, им частично передаются внутрифирменные технологии. И по окончании работ они способны анализировать и улучшать бизнес-процессы в рамках собственной отдельно взятой организации.

Исторически консалтинговые компании появляются на рынке последними. Это связано с появлением спроса на их услуги, который возникает с переходом от «островковой» к комплексной автоматизации, когда предприятие не способно самостоятельно справится с вставшими перед ним проблемами, а следовательно рождается понимание, что необходимо платить не только за программное и аппаратное обеспечение, но и за отчеты и рекомендации. Консалтинговые структуры характеризуются следующими пятью позициями:

– знания и информация – главный и единственный их продукт;
– опыт персонала, приобретаемый годами и десятилетиями при работе над конкретными проектами;
– наличие методологии выполнения консалтинговых проектов;
– независимость;
– объективность.

Понятно, что полной независимости и объективности не бывает да и быть не может. Тем не менее, когда консалтинговая структура входит в компанию, занимающуюся собственными разработками или продвигающую западную систему автоматизации, внедрение которой стоит сотни тысяч и миллионы долларов – это нонсенс.

С другой стороны, у каждого специалиста есть свои пристрастия, излюбленные продукты или подходы. Так что не стоит думать, что нанимая специалистов по консалтингу, предприятие получает истину в последней инстанции. Другое дело, что оно вправе рассчитывать на профессионализм и получение одного из лучших решений своей проблемы.

20.3 CASE-технологии – методологическая и инструментальная база консалтинга

За последнее десятилетие сформировалось новое направление в программотехнике – CASE (Computer-Aided Software/System Engineering). В настоящее время не существует общепринятого определения CASE. Содержание этого понятия обычно определяется перечнем задач, решаемых с помощью CASE, а также совокупностью применяемых методов и средств. Очень грубо, CASE – технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных си-

стем программного обеспечения (ПО), поддержанную комплексом взаимоувязанных средств автоматизации. CASE – это инструментарий для системных аналитиков, разработчиков и программистов, заменяющий им бумагу и карандаш на компьютер для автоматизации процесса проектирования и разработки ПО.

К настоящему моменту дисциплина CASE оформилась в самостоятельное наукоемкое направление в программотехнике, повлекшее за собой образование мощной CASE-индустрии, объединившей сотни фирм и компаний различной ориентации. Среди них выделяются компании – разработчики средств анализа и проектирования ПО с широкой сетью дистрибьютерских и дилерских фирм; фирмы – разработчики специальных средств с ориентацией на узкие предметные области или на отдельные этапы жизненного цикла ПО; обучающие фирмы, которые организуют семинары и курсы подготовки специалистов; консультационные фирмы, оказывающие практическую помощь при использовании CASE-пакетов для разработки конкретных приложений; фирмы, специализирующиеся на выпуске периодических журналов и бюллетеней по CASE. Основными покупателями CASE-пакетов за рубежом являются военные организации, центры обработки данных и коммерческие фирмы по разработке ПО.

Существует мнение, что CASE является наиболее перспективным направлением в программотехнике. С этим, естественно, можно и нужно спорить, но то, что CASE – наиболее бурно и интенсивно развиваемое направление, является в настоящее время фактом. Практически ни один серьезный зарубежный программный проект не осуществляется без использования CASE-средств. Известная методология структурного системного анализа SADT (точнее ее подмножество IDEFO) принята в качестве стандарта на разработку ПО Министерством обороны США. Более того, среди менеджеров и руководителей компьютерных фирм считается чуть ли не правилом хорошего тона знать основы SADT и при обсуждении каких либо вопросов нарисовать простейшую диаграмму, поясняющую суть дела.

CASE позволяет не только создавать «правильные» продукты, но и обеспечить «правильный» процесс их создания. Основная цель CASE состоит в том, чтобы отделить проектирование ПО от его кодирования и последующих этапов разработки, а также скрыть от разработчиков все детали среды разработки и функционирования ПО. Чем больше деятельности будет вынесено в проектирование из кодирования, тем лучше.

При использовании CASE-технологий изменяются все этапы жизненного цикла программной системы, при этом наибольшие изменение касаются этапов анализа и проектирования. В большинстве современных CASE-систем применяются методологии структурного анализа и проектирования, основанные на наглядных диаграммных техниках, при этом для описания модели проектируемой системы используются графы, диаграммы, таблицы и схемы.

Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру ее все большим числом уровней.

Несмотря на то, что структурные методологии зарождались как средства анализа и проектирования ПО, сфера их применений в настоящее время выходит далеко за рамки названной предметной области. Поэтому CASE-технологии успешно применяются для моделирования практически всех предметных областей, однако устойчивое положение они занимают в следующих областях:

– бизнес-анализ (фактически, модели деятельности предприятий «как есть» и «как должно быть» строятся с применением методов структурного системного анализа и поддерживающих их CASE-средств);

– системный анализ и проектирование (практически любая современная крупная программная система разрабатывается с применением CASE-технологий по крайней мере на этапах анализа и проектирования, что связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ).

Следует отметить, что CASE – не революция в программотехнике, а результат естественного эволюционного развития всей отрасли средств, называемых ранее инструментальными или технологическими. Однако это и не Confuse Array of Software that does Everything, существует ряд признаков и свойств, наличие которых позволяет классифицировать некоторый продукт как CASE-средство. Одним из ключевых признаков является поддержка методологий структурного системного анализа и проектирования.

С самого начала CASE-технологии развивались с целью преодоления ограничений при использовании структурных методологий проектирования 60-70-х годов (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т.д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом, CASE-технологии, вообще говоря, не могут считаться самостоятельными методологиями, они только развивают структурные методологии и делают более эффективным их применение за счет автоматизации.

Помимо автоматизации структурных методологий и, как следствие, возможности применения современных методов системной и программной инженерии, CASE обладают следующими основными достоинствами:

– улучшают качество создаваемого ПО за счет средств автоматического контроля (прежде всего, контроля проекта);

– позволяют за короткое время создавать прототип будущей системы, что позволяет на ранних этапах оценить ожидаемый результат;

– ускоряют процесс проектирования и разработки;

– освобождают разработчика от рутинной работы, позволяя ему цели ком сосредоточиться на творческой части разработки;

– поддерживают развитие и сопровождение разработки;

– поддерживают технологии повторного использования компонент разработки.

Большинство CASE-средств основано на парадигме методология/метод/нотация/средство. Методология определяет руководящие указания для оценки и выбора проекта разрабатываемого ПО, шаги работы и их последовательность, а также правила распределения и назначения методов. Метод – это систематическая процедура или техника генерации описаний компонент ПО (например, проектирование потоков и структур данных). Нотации предназначены для описания структуры системы, элементов данных, этапов обработки и включают графы, диаграммы, таблицы, блок-схемы, формальные и естественные языки. Средства – инструментарий для поддержки и усиления методов. Эти инструменты поддерживают работу пользователей при создании и редактировании графического проекта в интерактивном режиме, они способствуют организации проекта в виде иерархии уровней абстракции, выполняют проверки соответствия компонентов

20.4 BPwin – средство концептуального моделирования бизнес-процессов предприятия

Общие сведения о работе в среде пакета BPwin. Чтобы выжить, нужно быстро реагировать на изменение окружающей среды. Однако руководителям предприятий, прежде всего, необходимо ясно понять текущую ситуацию на предприятии, а затем определить характер предполагаемых изменений для получения желаемой выгоды. Существует инструмент, способный помочь решению таких сложных задач. Перед руководителями предприятий все чаще встает задача реорганизации деятельности. Это может быть пересмотр штатной структуры, внедрение новой технологии или информационной системы и т. д. – все то, что должно резко повысить эффективность работы и помочь выжить в условиях нестабильного внешнего окружения. Бессмысленно планировать изменения и улучшения без понимания того, как в настоящий момент работает предприятие. А это не такая простая задача, как может показаться на первый взгляд. Практически невозможно полностью и достаточно подробно описать бизнес-процессы организации. Рядовые сотрудники хорошо представляют, что происходит на их конкретном рабочем месте, но могут и не знать, как работают их коллеги, и вряд ли представляют себе, как работает организация в целом. Руководитель, наоборот, хорошо знает, как работает предприятие в общем, но не в состоянии удержать в голове специфику деятельности на каждом рабочем месте. Следовательно, для того чтобы получить адекватное описание функций организации, нужно ак-

кумулировать знания многих людей в единой модели, которая поможет найти слабые места в производственном процессе. Она же послужит основой при оценке стоимости продукции (или обслуживании клиентов) и ляжет в фундамент идеальной модели, т. е. такого конечного состояния бизнес-процессов, к которому следует стремиться.

Не зависимо от того пытаетесь ли вы изучить и изменить существующую систему или создавать абсолютно новую, самая большая трудность успешного преобразования заключается в невозможности провести анализ и установить связь между огромным количеством взаимодействующих друг с другом процессов, происходящих на предприятии.

Моделирование – один из наиболее эффективных методов изучения и установления связей. В модели процесса исключена излишняя детализация, что упрощает и без того сложную систему при анализе. Для устранения неоднозначности и выявления важной информации, оставшиеся данные подвергаются структурированию. Графики (а именно рамки и дуги) используются для отображения многозначности структуры, что и является причиной необходимости графического представления модели процесса. Однако, корректные определения объектов, появляющиеся в модели в качестве дополнительной текстовой информации, также являются необходимыми для выявления роли модели как инструмента связи.

При моделировании процесса, вы можете рассмотреть представляющую интерес систему на необходимую «глубину», что дает возможность подробного анализа и изучения работы вашей организации и что возможно наиболее важно, обменяться информацией с другими.

Итак, пакет BPwin – мощный инструмент моделирования с возможностью анализа, документирования и корректирования бизнес процессов. Он поможет устранить лишние или неэффективные операции, уменьшить издержки, повысить гибкость и улучшить уровень обслуживания заказчика. Модель BPwin обеспечивает интегрированное изображение того, как работает ваша организация. Это изображение, в свою очередь, состоит из подмоделей отделов.

BPwin как SADT технология. BPwin- представляет собой SADT-технологию, являющуюся одной из самых известных и широко используемых систем проектирования. SADT-аббревиатура слов Structured Analysis Design Technique (Технология структурного анализа и проектирования)-это графические обозначения и подход к описанию систем. SADT-технология представляет собой иерархическую многоуровневую модельную систему сверху-вниз до нужного уровня детализации. Каждый уровень представляет собой законченную систему(блок), поддерживаемую и контролируемую системой(блоком), находящейся над ней. Под словом «система» мы понимаем совокупность взаимодействующих компонент и взаимосвязей между ними. Под термином «моделирование» мы понимаем процесс создания точного

описания системы. SADT является полной методологией для создания описания систем, основанной на концепциях системного моделирования. С точки зрения SADT модель может быть сосредоточена либо на функциях системы, либо на ее объектах. SADT-модели, ориентированные на функции, принято называть функциональными моделями, а ориентированные на объекты системы – моделями данных. BPwin ориентирован на построение функциональных моделей с учётом особенностей SADT-технологии. Эта модель представляет с требуемой степенью детализации систему функций, которые в свою очередь отражают свои взаимоотношения через объекты системы.

Три методологии – IDEF0, DFD и IDEF3, поддерживаемые в пакете BPwin, позволяют посмотреть с разных сторон на деятельность предприятия.

IDEF0 – это функциональная модель, предназначенная для описания бизнес-процессов на предприятии. Она позволяет понять, какие объекты или информация служат сырьем для процессов, какие результаты производят работы, что является управляющими факторами и какие ресурсы для этого необходимы. Методология структурного моделирования предполагает построение модели AS-IS (как есть), анализ и выявление недостатков существующих бизнес-процессов и построение модели TO-BE (как должно быть), то есть модели, которая должна использоваться при построении автоматизированной системы управлением предприятия. DFD (Data flow diagramming) переводится на русский как «схемы потоков данных». С их помощью описываются документооборот и обработка информации. Подобно IDEF0, DFD представляет модельную систему как сеть связанных между собой работ. DFD можно использовать как дополнение к модели IDEF0, когда требуется более наглядное отображение текущих операций документооборота, описания функций обработки информации, документов, объектов, а также сотрудников или отделов, которые участвуют в обработке информационных потоков. Для описания логики взаимодействия информационных потоков более подходит IDEF3. Иногда ее называют workflow diagramming (моделирование потоков работ)- моделирование с использованием графического описания информационных потоков, взаимоотношений между процессами обработки информации и объектами, являющимися частью этих процессов. У IDEF3 имеется специфический элемент перекресток. Им описывают последовательность выполнения работ, очередность их запуска и завершения. С помощью workflow-схем можно моделировать сценарии действий сотрудников организации, например порядок обработки заказа или события, на которые необходимо реагировать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования любой функции, моделируемой на схеме IDEF0. Если в одной модели необходимо учесть специфические стороны бизнес-процессов предприятия, BPwin позволяет переключиться на любую нотацию (IDEF0, IDEF3, DFD),

находясь на любой ветви схемы, и создать смешанную модель. Пакет BPwin оснащён мощным инструментом навигации под названием Model Explorer. В нем смешанная модель может быть представлена в виде дерева схем, что существенно облегчает навигацию. В пакете BPwin версии 2.5 с помощью Model Explorer и техники перетаскивания можно переносить и копировать работы вместе со всеми соответствующими стрелками как внутри моделей, так и между ними. Все работы IDEF0 показываются в Model Explorer зеленым цветом, DFD – желтым, а IDEF3 – синим.

Основные элементы и понятия методологии IDEF0. В основе методологии IDEF0 лежат четыре основных понятия.

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде прямоугольника (см. рис. 20.1) и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «производить услуги», а не «производство услуг»).

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом:

верхняя сторона имеет значение «Управление» (Control);

левая сторона имеет значение «Вход» (Input);

правая сторона имеет значение «Выход» (Output);

нижняя сторона имеет значение «Механизм» (Mechanism).

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер.

Вторым «китом» методологии IDEF0 является понятие интерфейсной дуги (Arrow). Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое уникальное наименование (Arrow Label). По требованию стандарта, наименование должно быть оборотом существительного.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

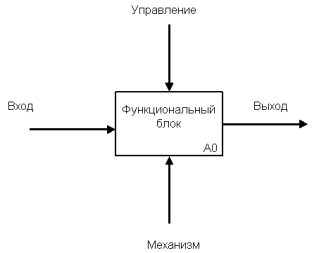


Рисунок 20.1 – Функциональный блок

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название «входящей», «исходящей» или «управляющей». В связи с этим существует схема кодирования дуг-»ICOM». Эта схема получила название по первым буквам английских эквивалентов слов вход (Input), выход (Output), управление (Control) и механизм (Mechanism). Кроме того, «источником» (началом) и «приемником» (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом «источником» может быть только выходная сторона блока, а «приемником» любая из трех оставшихся.

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь по крайней мере одну управляющую интерфейсную дугу и одну исходящую. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Третьим основным понятием стандарта IDEF0 является декомпозиция (Decomposition). Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и обозначается идентификатором «A-0».

В пояснительном тексте к контекстной диаграмме должна быть указана цель (Purpose) построения диаграммы в виде краткого описания и зафиксирована точка зрения (Viewpoint).

Определение и формализация цели разработки IDEF0 – модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь. Например, если мы моделируем деятельность предприятия с целью построения в дальнейшем на базе этой модели информационной системы, то эта модель будет существенно отличаться от той, которую бы мы разрабатывали для того же самого предприятия, но уже с целью оптимизации логистических цепочек.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Например, функциональные модели одного и того же предприятия с точек зрения главного технолога и финансового директора будут существенно различаться по направленности их детализации. Это связано с тем, что в конечном итоге, финансового директора не интересуют аспекты обработки сырья на производственных станках, а главному технологу ни к чему прорисованные схемы финансовых потоков. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы и называется дочерней (Child diagram) по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме соответственно называется дочерним блоком – Child Box). В свою очередь, функциональный блок – предок называется родительским блоком по отношению к дочерней диаграмме (Parent Box), а диаграмма, к которой он принадлежит – родительской диаграммой (Parent Diagram). Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. Важно отметить, что в каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок, или исходящие из него фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0 – модели. Наглядно принцип декомпозиции представлен на рисунке 20.2. Следует обратить внимание на взаимосвязь нумерации функциональных блоков и диаграмм. Каждый блок имеет свой уникальный порядковый номер на диаграм-

ме (цифра в правом нижнем углу прямоугольника). Блоки никогда не размещаются на диаграмме случайным образом. Они размещаются по степени важности, как ее понимает автор диаграммы. В SADT этот относительный порядок называется доминированием. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наименее доминирующий – в правом нижнем углу. В результате получается «ступенчатая» схема. При этом, самому верхнему блоку будет соответствовать цифра 1. Обозначение под правым углом указывает на номер дочерней для этого блока диаграммы. Отсутствие этого обозначения говорит о том, что декомпозиции для данного блока не существует. В правом верхнем углу всего «рабочего листа», в поле CONTEXT изображаются блоки уровня, подвергшегося декомпозиции, где декомпозируемый блок окрашен в чёрный цвет.

Часто бывают случаи, когда отдельные интерфейсные дуги не имеет смысла продолжать рассматривать в дочерних диаграммах ниже какого-то определенного уровня в иерархии, или наоборот – отдельные дуги не имеют практического смысла выше какого-то уровня, чтобы не перегружать диаграммы и не делать их сложными для восприятия. С другой стороны, случается необходимость избавиться от отдельных «концептуальных» интерфейсных дуг и не детализировать их глубже некоторого уровня. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования.

При построении диаграмм возможны случаи, когда «входит в туннель» или «выходит из туннеля».

Обозначение «туннеля» (Arrow Tunnel) в виде двух скобок вокруг начала интерфейсной дуги обозначает, что эта дуга не была унаследована от функционального родительского блока и появилась (из «туннеля») только на этой диаграмме (то есть дуга имеет скрытый источник). В свою очередь, такое же обозначение вокруг конца (стрелки) интерфейсной дуги в непосредственной близи от блока – приёмника означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет (то есть дуга имеет скрытый приёмник). Чаще всего бывает, что отдельные объекты и соответствующие им интерфейсные дуги не рассматриваются на некоторых промежуточных уровнях иерархии – в таком случае, они сначала «погружаются в туннель», а затем, при необходимости «возвращаются из туннеля».

Последним из понятий IDEF0 является глоссарий (Glossary). Для каждого из элементов IDEF0: диаграмм, функциональных блоков, интерфейсных дуг существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом.

Этот набор называется глоссарием и является описанием сущности данного элемента. Например, для управляющей интерфейсной дуги «распоряжение об оплате» глоссарий может содержать перечень полей соответствующего дуге документа, необходимый набор виз и т.д. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

IDEF0-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в соответствующем стандарте приняты соответствующие ограничения сложности:

ограничение количества функциональных блоков на диаграмме тремя-шестью. Верхний предел (шесть) заставляет разработчика использовать иерархии при описании сложных предметов, а нижний предел (три) гарантирует, что на соответствующей диаграмме достаточно деталей, чтобы оправдать ее создание;

ограничение количества подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг четырьмя;

количество уровней детализации – 9.

Разумеется, строго следовать этим ограничениям вовсе необязательно, однако, как показывает опыт, они являются весьма практичными в реальной работе.

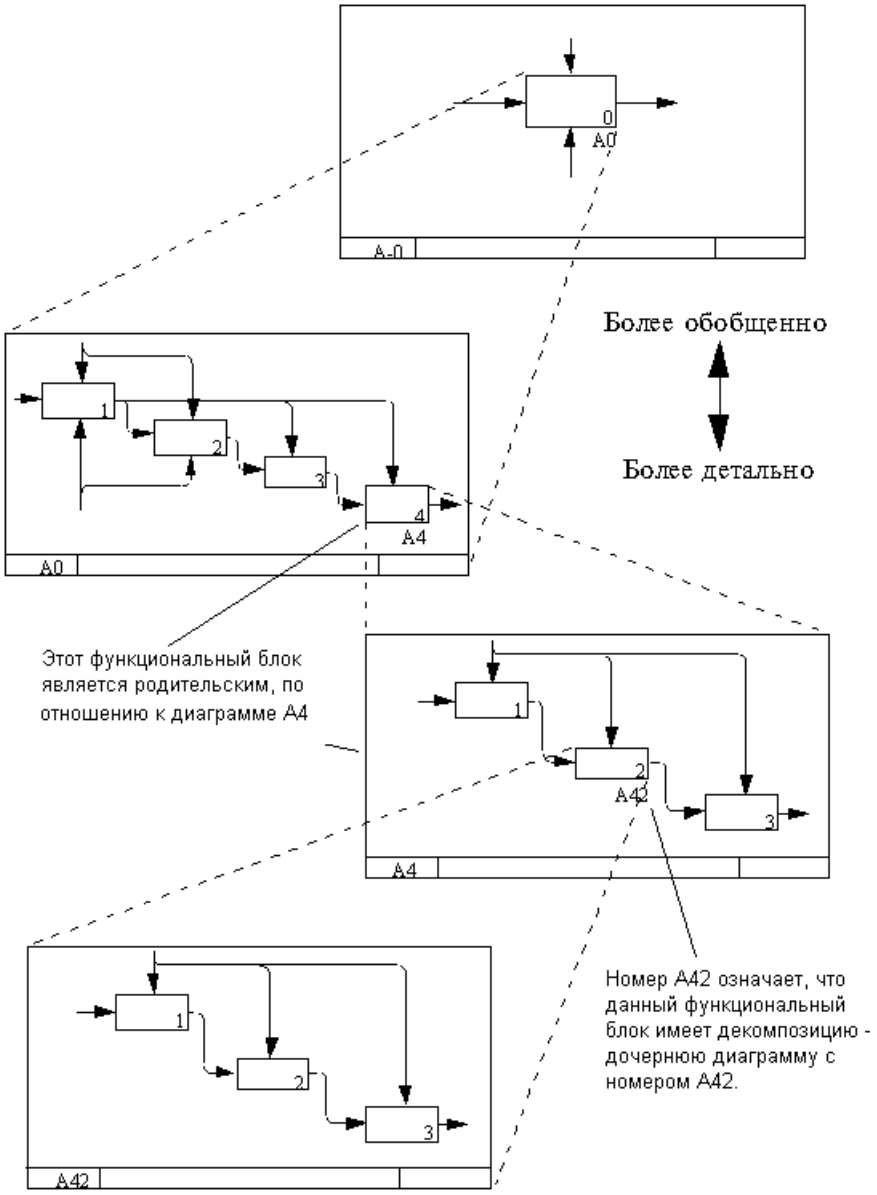


Рисунок 20.2 – Декомпозиция функциональных блоков

Литература

1 Айвазян, С.А.. Прикладная статистика: Классификация и снижение размерности / С.А. Айвазян, В.М. Бухштабер, И.С. Енюков. – М.: Финансы и статистика. 1989. – 605с.

2 Айвазян, С.А. Прикладная статистика: основы моделирования и первичная обработка данных / С.А. Айвазян, И.С. Енюков, Л.Д. Мешалкин. – М.: Финансы и статистика. 1983. – 472с.

3 Айвазян, С.А. Прикладная статистика: исследование зависимостей / С.А Айвазян, И.С. Енюков, Л.Д. Мешалкин. – М.: Финансы и статистика. 1985. – 488с.

4 Корнеев, В.В. Базы данных. Интеллектуальная обработка информации / В.В.Корнеев [b lh/],– М.: Номидж, 2000. –351с.

5 Гаврилова, Т.А. Базы знаний интеллектуальных систем / Т.А.Гаврилова, В.Ф.Хорошевский. –СПб: Питер, 2000. 384с.

6 Змитрович, А.И. Интеллектуальные информационные системы / А.И.Змитрович. –Мн.: НТООО «ТетраСистемс», 1997. – 368с.

7 Тельнов, Ю.Ф. Интеллектуальные информационные системы / Ю.Ф.Тельнов. – М.: Финансы и статистика, 2000. – 121с.

8 Мандель, И.Д. Кластерный анализ / И.Д. Мандель. – М.: Финансы и статистика, 1988. – 172с.

9 Мелник, М. Основы прикладной статистики / М. Мелник. – М.: Энергоатомиздат. 1983. – 416 с.

10 Бонгард, М. М. Проблема узнавания / М. М. Бонгард. – М.: Наука, 1967. – 320с.

11 Представление знаний в человеко-машинных и робототехнических системах. Прикладные человеко-машинные системы, ориентированные на знания. – М.: ВЦ АН СССР; ВИНИТИ, 1984. – 380с.

12 Шрейдер, Ю. А. Системы и модели / Ю. А. Шрейдер, А.А. Шаров. – М.: Радио и связь, 1982. – 152с.

13 Пособие для лабораторных занятий по спецкурсу «Обработка экспериментальных данных на ЭВМ» (для специальностей Н.01.01 и Н.01.08) / И.В. Максимей, Н.Б. Осипенко, А.Н. Осипенко. – Гомель: ГГУ им.Ф.Скорины, 1999. – 54с.

14 Справочное пособие по первичной статистической обработке и исследованию зависимостей (для специальностей Н.01.01 и Н.01.08) / Н.Б. Осипенко, А.Н. Осипенко. – Гомель: ГГУ им.Ф.Скорины, 2000. – 85с.

15 Калянов, Г.Н. Case-технологии. Консалтинг при автоматизации бизнес-процессов / Г.Н. Калянов. – М.: «Горячая линия – Телеком», 2002. – 320с.

.

Учебное издание

Н. Б. ОСИПЕНКО

Интеллектуальные информационные системы

ТЕКСТЫ ЛЕКЦИЙ
для студентов специальности 1 – 31 03 03 01
«Прикладная математика (научно-производственная деятельность)»

Подписано в печать 12.11.2009 (104). Формат 60 x 84 1/16.
Бумага писчая №1. Гарнитура «Таймс». Усл.печ.л.10,7. Уч.-изд.л. 8,3.
Тираж 25 экз.

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»
246019, г. Гомель, ул. Советская, 104.