Министерство образования и науки РФ
Нижнекамский химико-технологический институт (филиал)
Федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Казанский национальный исследовательский технологический университет»

Л.А. Амаева

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ТЕКСТЫ ЛЕКЦИЙ

Нижнекамск
2012

УДК 004.8
А 61

Печатаются по решению редакционно-издательского совета Нижнекамского химико-технологического института (филиала) ФГБОУ ВПО «КНИТУ».

Рецензенты:
Ибатуллин Р.Р., кандидат физико-математических наук, доцент кафедры ИДМ филиала КФУ в г. Елабуге;
Шафигуллин Л.Н., кандидат технических наук, доцент кафедры автоматизации и информационных технологий ИНЭКА.

Амаева, Л.А.

А 61 Системы искусственного интеллекта : тексты лекций / Л.А. Амаева. – Нижнекамск : Нижнекамский химико-технологический институт (филиал) ФГБОУ ВПО «КНИТУ», 2012. – 132 с.

Предназначены для обеспечения курса лекций по дисциплине «Системы искусственного интеллекта» для студентов, обучающихся по специальности 230102 «Автоматизированные системы обработки информации и управления» и по направлению подготовки бакалавров 230100 «Информатика и вычислительная техника»

Подготовлены на кафедре информационных систем и технологий Нижнекамского химико-технологического института.

УДК 004.8

СОДЕРЖАНИЕ

ТЕМА 1. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ: ОСНОВНЫЕ ПОНЯТИЯ И ИСТОРИЯ ВОЗНИКНОВЕНИЯ

1. Основные понятия систем искусственного интеллекта
2. Общая характеристика задач решаемых методами искусственного интеллекта

1. Основные понятия систем искусственного интеллекта

В современном мире прогресс производительности программиста практически достигается только в тех случаях, когда часть интеллектуальной нагрузки берут на себя компьютеры. Одним из способов достигнуть максимального прогресса в этой области, является «искусственный интеллект», когда компьютер берет на себя не только однотипные, многократно повторяющиеся операции, но и сам сможет обучаться. Кроме того, создание полноценного «искусственного интеллекта» открывает перед человечеством новые горизонты развития.

Термин «интеллект» (intelligence) происходит от латинского intellectus — что означает ум, рассудок, разум; мыслительные способности человека. Искусственный интеллект обычно толкуется как свойство автоматических систем брать на себя отдельные функции интеллекта человека, например, выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних воздействий.

Появление ЭВМ, работа которых происходит под управлением созданных человеком программ (т.е. с максимально полным привлечением интеллектуальных способностей человека), позволило автоматизировать самые разнообразные процессы обработки данных или, по-другому, самые разнообразные вычислительные процессы.

Здесь под автоматизацией вычислительных процессов или вычислений понимается выполнение их вычислительным устройством (компьютером) без непосредственного участия человека. При этом важным является то, что

а) вычислительный процесс должен представляться в виде последовательности (сколь угодно большой, но конечной длины) элементарных или «рутинных» операций;

б) формирование последовательности элементарных операций или, по-другому, составление алгоритма решения задачи, осуществляется непосредственно человеком (пользователем ЭВМ);

в) вычислительное устройство не может само (без участия человека) ни создавать, ни менять алгоритм, если это изменение не предусмотрено самим алгоритмом.

Поэтому принято говорить, что вычислительные устройства (в дальнейшем ЭВМ или компьютеры), построенные по классической фон-неймановской схеме реализуют так называемые «жесткие» вычисления. Термин «жесткие» вычисления обозначает организацию вычислений по заранее разработанному человеком (пользователем ЭВМ) вполне определенному алгоритму.

Говорят, обработка информации на ЭВМ, понимаемая в общепринятом смысле, представляет собой обработку данных. В то же время, характерным признаком интеллектуальных систем является обработка знаний. При этом **данными** называют информацию фактического характера, описывающую объекты, процессы и явления конкретной предметной области. Как правило, эта информация не требует при своем дальнейшем использовании более глубокого осмысления и анализа. К примеру, в качестве данных могут быть координаты материальной точки (x_i, y_i) измеренные в процессе ее вращения вокруг начала координат и соответствующей данным координатам моменты времени t_i (рисунок 1).

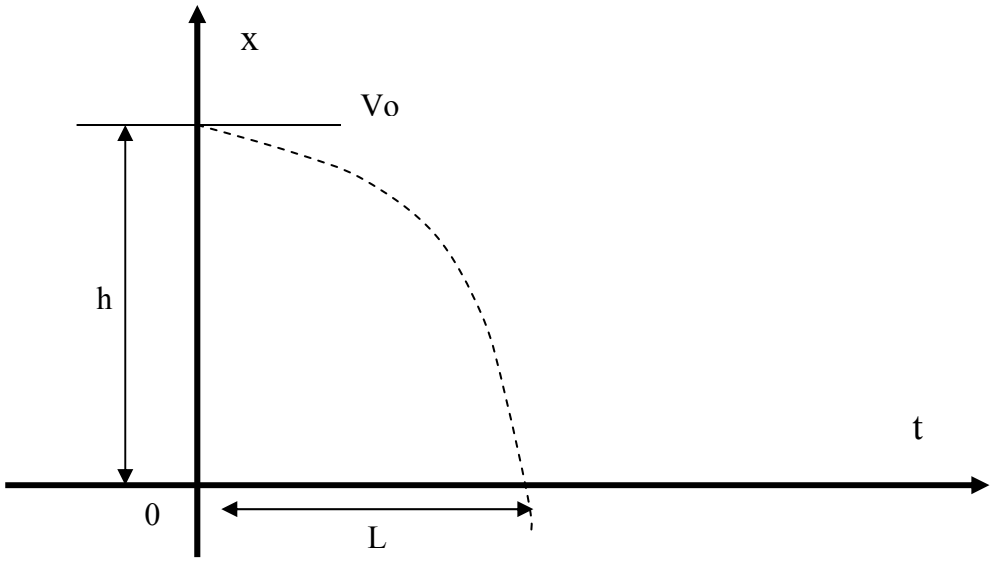


Рисунок 1 - Движение точки

Обычно данные представляются в виде таблиц, диаграмм, графиков. Так, данные о движении точки можно представить в виде таблицы.

Таблица 1 - Данные о движении точки

x	**t**
x_1	t_1
x_2	t_3
….	…
x_N	t_N

Знания являются более сложной категорией информации по сравнению с данными. Знания описывают не только отдельные факты, но и взаимосвязи между ними. Поэтому знания называют структурированными данными. Знания могут быть получены:

1) на основе обработки экспериментальных данных;

2) в результате мысленной деятельности человека.

Интеллектуальные системы позволяют производить автоматическую (без участия человека) обработку данных в условиях существенной априорной неполноты знаний о том, как нужно вести эту обработку для получения требуемого результата. Очевидно, что для этого интеллектуальные системы должны быть способны сами генерировать (получать) недостающие знания путем:

1) логического (дедуктивного) вывода;

2) обучения;

3) поиска;

4) обработки экспериментальных данных.

С помощью существующих на настоящий момент времени методов искусственного интеллекта (нечеткой логики, нейронных сетей и генетических алгоритмов) обработку информации можно производить с помощью нейросетевых методов аппроксимации и интерпретации данных (рисунок 2).

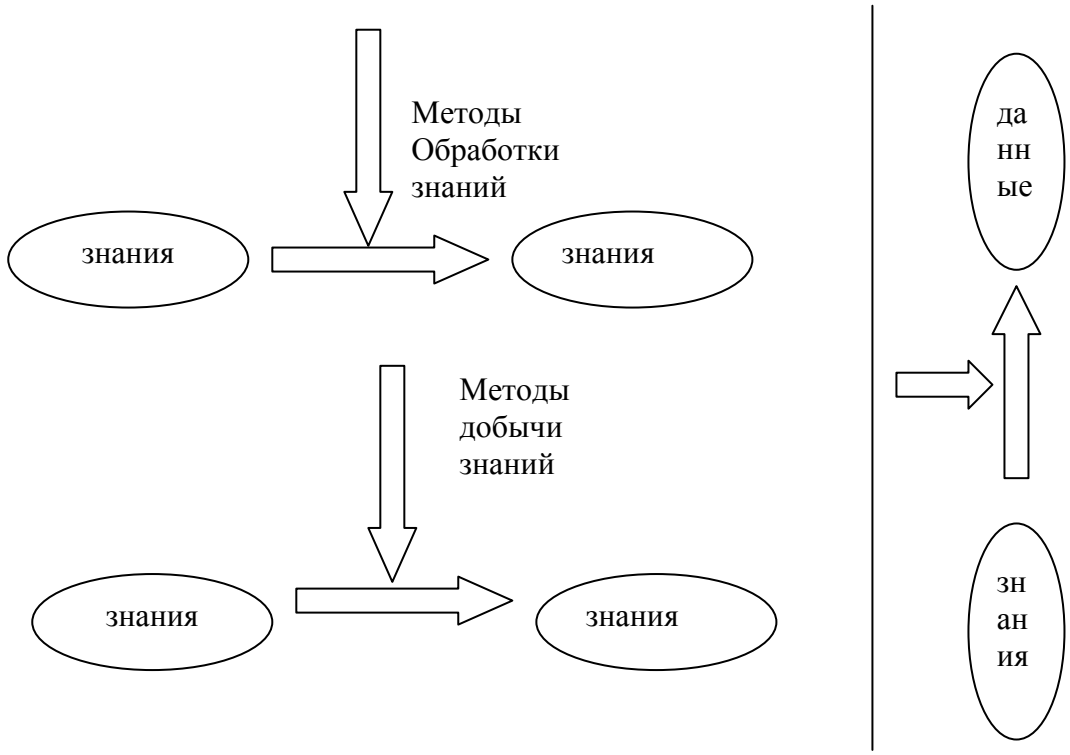


Рисунок 2 - Добыча и обработка знаний

Так, если аппроксимировать приведенные в таблице 1 данные о движении материальной точки, брошенной с высоты h под углом $\alpha = 0$ к горизонту, то получим следующую функциональную зависимость высоты x от времени t движения $x = h - \frac{gt^2}{2}$, где g=9.8 – гравитационная постоянная. Данную формулу можно трактовать как запись (на языке математических формул) знаний о законе движения материальной точки под действием силы земного притяжения.

Сопоставляя таблицу 1 и формулу, можно отметить, что запись информации о движении материальной точки с помощью знаний дает (по

сравнению с совокупностью данных о ее движении) более целостную и системную картину движения.

2. Общая характеристика задач решаемых методами искусственного интеллекта

Рассмотрим главные, или, что более правильно, сущностные отличия задач, решаемых на ЭВМ с помощью методов искусственного интеллекта, от обычных задач, решаемых традиционными методами и способами.

Степень использования человеческого интеллекта.

Как известно, традиционно решаемые на ЭВМ задачи требуют максимально полного использования интеллекта (способностей, знаний) при выборе метода и составлении алгоритма решения задачи.

При этом на ЭВМ возлагается лишь задача правильного выполнения или реализации разработанного человеком алгоритма.

Напротив, решение задач с привлечением методов искусственного интеллекта основывается не только на использовании знаний человека, но и дополнительных знаниях, полученных самой ЭВМ.

Методы и структурные решения, лежащие в основе получения (вывода) знаний, являются предметом рассмотрения сравнительно молодой науки, называемой искусственным интеллектом.

Полнота априорной информации.

Традиционно решаемые на ЭВМ задачи требуют для своего успешного решения большого объема априорной информации о закономерностях поведения исследуемого объекта или процесса. Например, если рассматривается движение летательного аппарата в атмосфере, то должны быть точно известны:

а) физические законы, определяющие силы, действующие на летательный аппарат;

б) полученные на их основе математические состояния (математическая модель объекта), определяющие реакцию летательного аппарата (изменение его высоты, скорости полета и т.п.) на эти силы и на управляющие воздействия со стороны системы управления летательным аппаратом.

На практике это весьма сложно обеспечить, учитывая существенную нестационарность условий полета (внешних и внутренних). Действительно, для современных летательных аппаратов характерен большой диапазон изменения характеристик атмосферы, возможность возникновения нештатных (или критических) ситуаций как в атмосфере (грозы, смерчи, турбулентные потоки и т.п.), так и на борту летательного аппарата (отказы оборудования, неправильные действия летчиков). Заранее все это при разработке алгоритма управления сложно предусмотреть. Поэтому, «жесткие» алгоритмы управления современными летательными аппаратами не обеспечивают требуемой эффективности их применения.

Интеллектуальные или «мягкие» алгоритмы управления, основанные на применении методов искусственного интеллекта («мягких» вычислений), существенно снижают требования к объему необходимой априорной информации за счет ее доопределения интеллектуальной системой непосредственно в процессе функционирования (в режиме on-line).

«Продвинутость» задач.

С помощью алгоритмов и алгоритмических процедур в классическом понимании можно автоматизировать решение только так называемых «рутинных» задач, не связанных с получением качественно новой информации (новых знаний), а связанных с организацией вычислительной процедуры их решения при условии, что априорно имеется вся необходимая для этого информация.

Решение же более содержательных по смыслу задач только с помощью алгоритмических процедур невозможно. Например, невозможно с помощью алгоритмов описать процессы, реализующие:

а) решение задач, начиная от словесной постановки и кончая получением результата решения;

б) перевода текстов с одного языка на другой;

в) игру в шахматы, карты и т.п.;

г) диагностики болезней;

д) доказательства математических теорем и др.

Для этого нужно располагать качественно новым математическим аппаратом и вычислительными машинами, позволяющими моделировать процесс мышления человека.

Рассмотрим характерные особенности данного процесса.

1. Деятельность человека всегда целесообразна, т.е. связана с достижениями некоторой цели. Это означает, что мыслительные процессы человека направлены на достижение цели (цель заставляет человека думать).

2. Человеческий мозг хранит огромное количество фактов и правил их использования. Для достижения определенной цели надо только обратиться к нужным фактам и правилам.

3. Принятие решений всегда осуществляется на основе специального механизма упрощения, позволяющего отбрасывать ненужные (малосущественные) факты и правила. Не имеющие отношения к решаемой в данный момент задаче и, наоборот, выделять главные, наиболее значимые факты и правила, нужные для достижения цели.

4. Достигая цели, человек не только приходит к решению поставленной перед ним задачи, но и одновременно приобретает новые знания. Та часть интеллекта, которая позволяет ему делать соответствующие заключения (выводы) на основании правил, сформулированных человеком, а также генерировать новые факты из уже существующих, называется механизмом ***логического вывода.***

Так, типовая схема решения математической задачи часто выглядит следующим образом. Выбираются неизвестные величины, подлежащие

определению. На основании анализа условий (ограничений), содержащихся в исходной формулировке задачи, составляется система уравнений, связывающих указанные неизвестные. Далее, применяя какой либо из стандартных методов решения полученных уравнений, находим искомое решение задачи. Заметим, то, что решив один раз конкретную задачу по описанной схеме, мы решим (и гораздо быстрее) другую подобную (и даже более сложную) задачу, отличающуюся значениями исходных данных, числом неизвестных, формой представления условий и т.д.

Поскольку система искусственного интеллекта (ИИ) принимает решения аналогично тому, как это делает человек, то она должна включать в себя следующие ключевые элементы – цель, факты и данные. Правила, механизмы вывода и упрощения. Все эти компоненты системы ИИ показаны на рисунке 3, где так же выделена база знаний, которая содержит всю располагаемую информацию о внешнем мире (моделях решаемых задач). Условно она может быть разделена на три части (или области), называемые базой целей, базой правил и базой данных.

Первая область содержит информацию о целях, для достижения которых предназначена система ИИ. Вторая область включает в себя сведения, которые отражают закономерности, характерные для решаемого класса задач. Это правила, механизмы упрощения и вывода, которые позволяют не только выводить новые факты, не зафиксированные ранее в базе данных, но и приобретать новые знания в ходе функционирования системы или на этапе ее обучения. В третьей области содержатся в некотором упорядоченном виде качественные данные, необходимые для решения данной задачи. В силу той особой роли, которую играет база знаний в процессе формирования решений, системы ИИ называют системами основанными на знаниях.

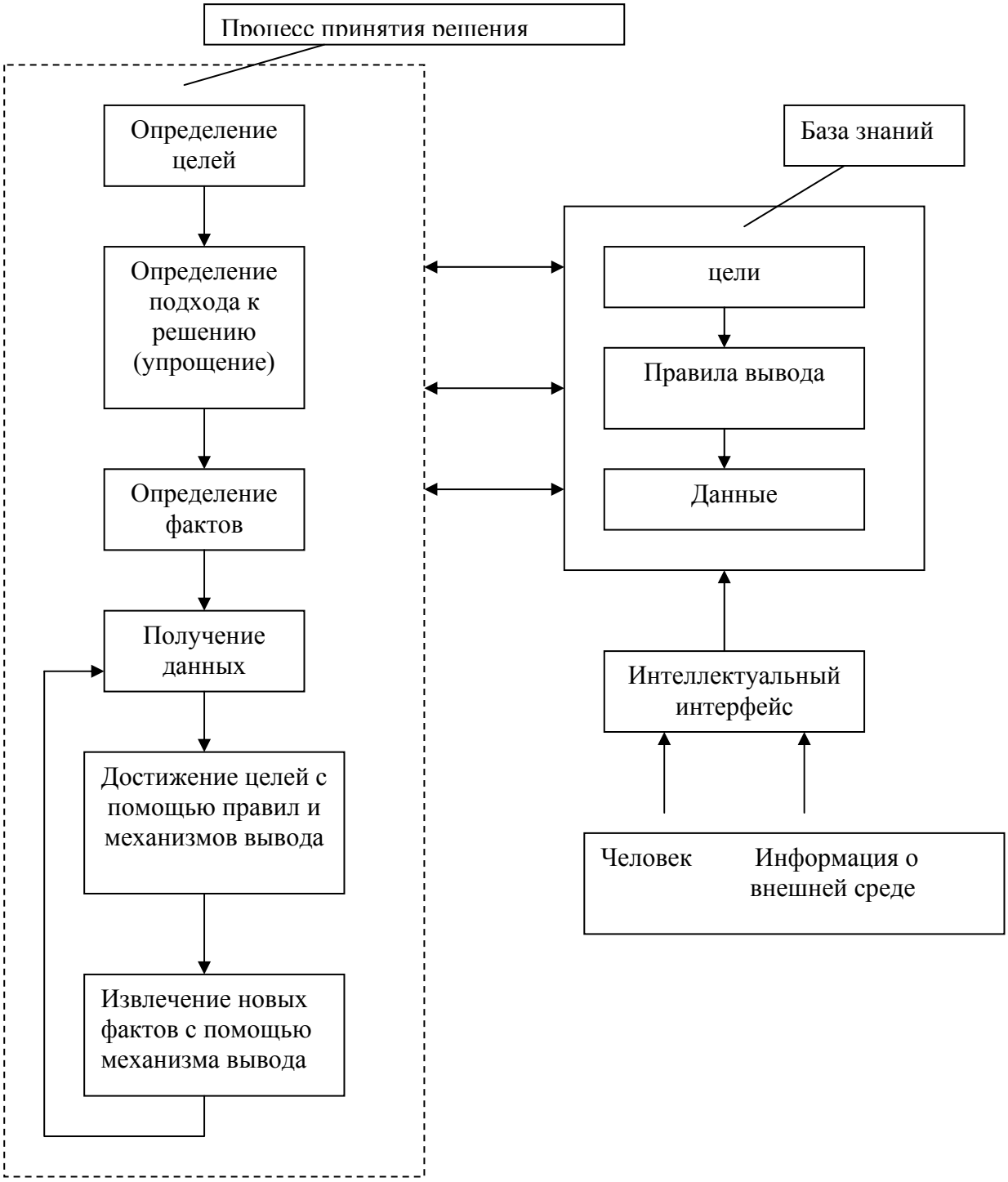


Рисунок 3 - Компоненты системы искусственного интеллекта

ТЕМА 2. ЭКСПЕРТНЫЕ СИСТЕМЫ

1. Понятие экспертной системы
2. История развития экспертных систем
3. Структура экспертной системы
4. Этапы разработки экспертных систем

1. Понятие экспертной системы

Экспертные системы (ЭС) - это яркое и быстро прогрессирующее направление в области искусственного интеллекта (ИИ). Причиной повышенного интереса, который ЭС вызывают к себе на протяжении всего своего существования, является возможность их применения к решению задач из самых различных областей человеческой деятельности. Пожалуй, не найдется такой проблемной области, в которой не было бы создано ни одной ЭС или, по крайней мере, такие попытки не предпринимались бы.

ЭС - это набор программ или программное обеспечение, которое выполняет функции эксперта при решении какой-либо задачи в области его компетенции. ЭС, как и эксперт-человек, в процессе своей работы оперирует со знаниями. Знания о предметной области, необходимые для работы ЭС, определенным образом формализованы и представлены в памяти ЭВМ в виде базы знаний, которая может изменяться и дополняться в процессе развития системы.

ЭС выдают советы, проводят анализ, выполняют классификацию, дают консультации и ставят диагноз. Они ориентированы на решение задач, обычно требующих проведения экспертизы человеком-специалистом. В отличие от машинных программ, использующий процедурный анализ, ЭС решают задачи в узкой предметной области на основе дедуктивных рассуждений. Такие системы часто оказываются способными найти решение задач, которые неструктурированны и плохо определены. Они справляются с отсутствием

структурированности путем привлечения эвристик, т. е. правил, взятых «с потолка», что может быть полезным в тех системах, когда недостаток необходимых знаний или времени исключает возможность проведения полного анализа.

Главное достоинство ЭС - возможность накапливать знания, сохранять их длительное время, обновлять и тем самым обеспечивать относительную независимость конкретной организации от наличия в ней квалифицированных специалистов. Накопление знаний позволяет повышать квалификацию специалистов, работающих на предприятии, используя наилучшие, проверенные решения.

Исследователи в области ЭС для названия своей дисциплины часто используют также термин «инженерия знаний», введенный Е.Фейгенбаумом как «привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов».

Программные средства, базирующиеся на технологии экспертных систем, или инженерии знаний, получили значительное распространение в мире. Важность экспертных систем состоит в следующем:

− технология ЭС существенно расширяет круг практически значимых задач, решаемых на компьютерах, решение которых приносит значительный экономический эффект;

− технология ЭС является важнейшим средством в решении глобальных проблем традиционного программирования: длительность и, следовательно, высокая стоимость разработки сложных приложений;

− высокая стоимость сопровождения сложных систем, которая часто в несколько раз превосходит стоимость их разработки; низкий уровень повторной используемости программ и т.п.;

− объединение технологии ЭС с технологией традиционного программирования добавляет новые качества к программным продуктам за

счет: обеспечения динамичной модификации приложений пользователем, а не программистом; большей «прозрачности» приложения (например, знания хранятся на ограниченном естественном языке, что не требует комментариев к знаниям, упрощает обучение и сопровождение); лучшей графики; интерфейса и взаимодействия.

ЭС предназначены для так называемых неформализованных задач, т.е. ЭС не отвергают и не заменяют традиционного подхода к разработке программ, ориентированного на решение формализованных задач.

Неформализованные задачи обычно обладают следующими особенностями:

− ошибочностью, неоднозначностью, неполнотой и противоречивостью исходных данных;

− ошибочностью, неоднозначностью, неполнотой и противоречивостью знаний о проблемной области и решаемой задаче;

− большой размерностью пространства решения, т.е. перебор при поиске решения весьма велик;

− динамически изменяющимися данными и знаниями.

Следует подчеркнуть, что неформализованные задачи представляют большой и очень важный класс задач. Многие специалисты считают, что эти задачи являются наиболее массовым классом задач, решаемых ЭВМ.

Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных тем, что в них в основном используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма).

Экспертные системы применяются для решения только трудных практических задач. По качеству и эффективности решения экспертные системы не уступают решениям эксперта-человека. Решения экспертных систем обладают *«прозрачностью»,* т.е. могут быть объяснены пользователю

на качественном уровне. Это качество экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях.

Необходимо отметить, что в настоящее время технология экспертных систем используется для решения различных типов задач (интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление) в самых разнообразных проблемных областях, таких, как финансы, нефтяная и газовая промышленность, энергетика, транспорт, фармацевтическое производство, космос, металлургия, горное дело, химия, образование, целлюлозно-бумажная промышленность, телекоммуникации и связь и др.

2. Структура экспертных систем

Типичная ЭС состоит из следующих основных компонентов (рисунок 4):

− решателя (интерпретатора);

− рабочей памяти (РП), называемой также базой данных (БД);

− базы знаний (БЗ);

− компонентов приобретения знаний;

− объяснительного компонента;

− диалогового компонента.

База данных (рабочая память) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

База знаний (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

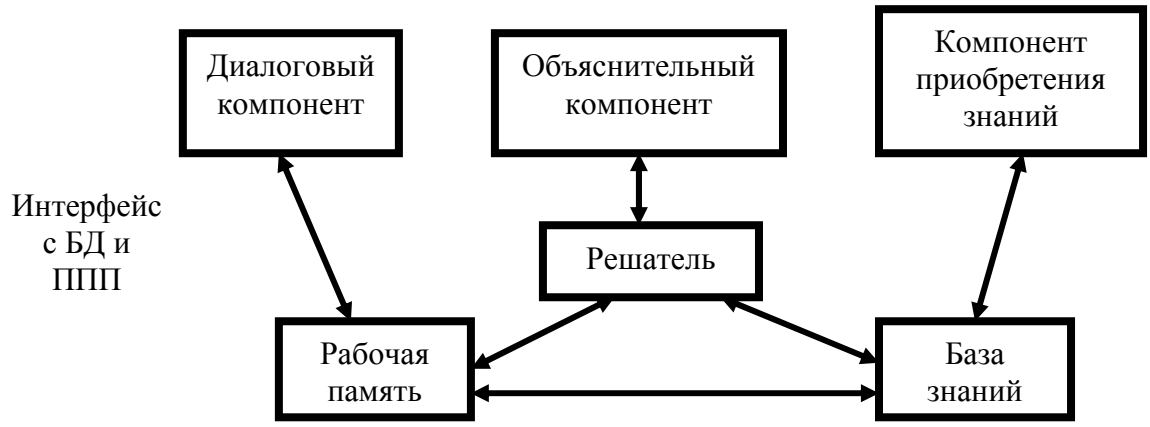


Рисунок 4 - Структура ЭС

Решатель, используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

Компонент приобретения знаний автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

Объяснительный компонент объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

Диалоговый компонент ориентирован на организацию дружественного общения с пользователем как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

В разработке ЭС участвуют представители следующих специальностей:

− эксперт в проблемной области, задачи которой будет решать ЭС;

− инженер по знаниям - специалист по разработке ЭС (используемые им технологию, методы называют технологией (методами) инженерии знаний);

− программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т. е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того инструментального средства, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Программист разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Структуру, приведенную на рисунке 4, называют *структурой статической ЭС*. ЭС данного типа используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Первые ЭС, получившие практическое использование, были статическими.

В архитектуру динамической ЭС по сравнению со статической ЭС вводятся два компонента: подсистема моделирования внешнего мира и подсистема связи с внешним окружением. Последняя осуществляет связи с внешним миром через систему датчиков и контроллеров.

Кроме того, традиционные компоненты статической ЭС (база знаний и машина вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

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

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

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

В режиме консультации общение с ЭС осуществляет конечный пользователь, которого интересует результат и способ его получения. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу). В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее.

3. Этапы разработки экспертных систем

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату.

Использовать ЭС следует только тогда, когда разработка ЭС *возможна, оправдана и* методы инженерии знаний *соответствуют* решаемой задаче.

Чтобы разработка ЭС была *возможной* для данного приложения, необходимо одновременное выполнение, по крайней мере, следующих требований:

− существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

− эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;

− эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут «извлечены» и вложены в ЭС;

− решение задачи требует только рассуждений, а не действий;

− задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);

− задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно «понятной» и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решения задачи;

− решение задачи не должно в значительной степени использовать «здравый смысл» (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный

человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование ЭС в данном приложении может быть возможно, но не оправдано. Применение ЭС может быть *оправдано* одним из следующих факторов:

− решение задачи принесет значительный эффект, например экономический;

− использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;

− использование ЭС целесообразно в тех случаях, когда при передаче информации эксперту происходит недопустимая потеря времени или информации;

− использование ЭС целесообразно при необходимости решать задачу в окружении, враждебном для человека.

Приложение *соответствует* методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

− задача может быть естественным образом решена посредством манипуляции с символами (т.е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;

− задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения ЭС;

− задача должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС. Однако она не должна быть чрезмерно сложной, чтобы ЭС могла ее решать;

− задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция «быстрого прототипа». Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип ЭС. Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны, они должны решать типичные задачи конкретного приложения, а с другой - время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально запараллелить процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора (разработки) программных средств (осуществляемым инженером по знаниям и программистом). Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области.

При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

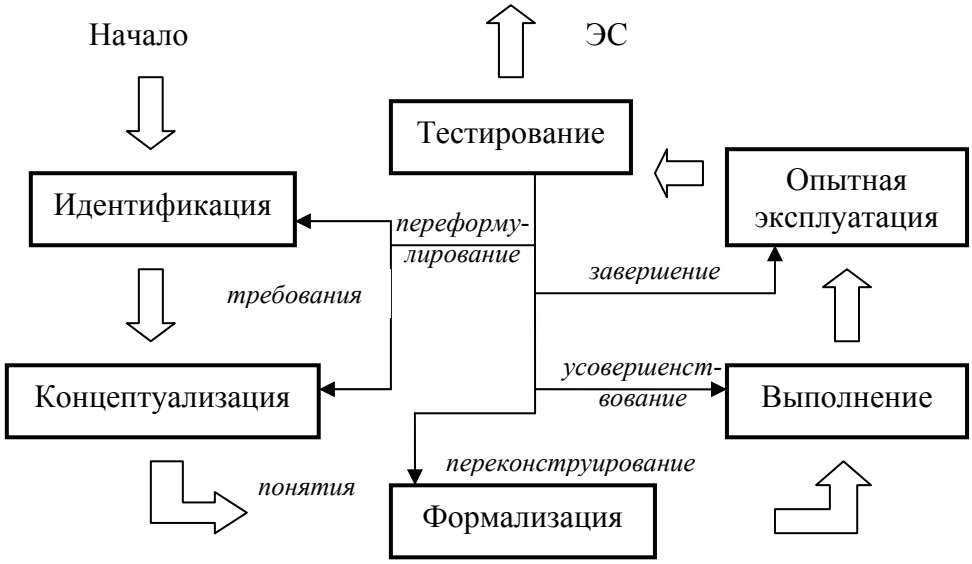


Рисунок 5 - Технология разработки ЭС

В ходе работ по созданию экспертных систем сложилась определенная технология их разработки, включающая шесть следующих этапов: идентификация, концептуализация, формализация, реализация, тестирование, опытная эксплуатация и внедрение.

На этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей. Определение участников и их ролей сводится к определению количества экспертов и инженеров по знаниям, а также формы их взаимоотношений. Обычно в основном цикле разработки экспертной системы участвуют не менее трех-четырех человек (один эксперт, один или два инженера по знаниям и один программист, привлекаемый для модификации и согласования инструментальных средств).

К процессу разработки экспертной системы могут привлекаться и другие участники. Например, инженер по знаниям может привлекать других экспертов для того, чтобы убедиться в правильности своего понимания основного эксперта; представительности тестов, демонстрирующих особенности рассматриваемой задачи; совпадении взглядов различных экспертов на качество предлагаемых решений. Формы взаимоотношений экспертов и инженеров следующие: эксперт исполняет роль информирующего или эксперт

выполняет роль учителя, а инженер - ученика. Вне зависимости от выбранной формы взаимоотношений инженер по знаниям должен быть готов и способен изучать специфические особенности той проблемной области, в рамках которой предстоит работать создаваемой экспертной системе. Несмотря на то, что основу знаний экспертной системы будут составлять знания эксперта, для достижения успеха инженер по знаниям должен использовать дополнительные источники знаний в виде книг, инструкций, которые ему рекомендовал эксперт.

На этапе *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этом этапе определяются следующие особенности задачи:

- типы доступных данных;
- исходные и выводимые данные;
- подзадачи общей задачи;
- используемые стратегии и гипотезы;
- виды взаимосвязей между объектами проблемной области;
- типы используемых отношений (иерархия, причина/следствие, часть/целое и т.п.);
- процессы, используемые в ходе решения задачи;
- типы ограничений, накладываемых на процессы, используемые в ходе решения;
- состав знаний, используемых для решения задачи и для объяснения решения.

Для определения перечисленных характеристик задачи целесообразно составить детальный протокол действий и рассуждений эксперта в процессе решения хотя бы одной конкретной задачи. Такой протокол обеспечивает инженера по знаниям словарем терминов (объектов) и некоторым приблизительным представлением о тех стратегиях, которые использует

эксперт. Кроме того, протокол помогает ответить на многие другие вопросы, возникающие в ходе разработки. На этом этапе инженер по знаниям рассматривает вопросы, относящиеся к представлению знаний и методам решения, но говорить о выборе конкретных способов и методов здесь еще рано.

На этапе *формализации* выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями. Основными задачами в процессе формализации являются проблемы структуризации исходной задачи и знаний в выбранном формализме, а именно:

1. структуризация общей задачи на связанные подзадачи;
2. структуризация предметной области на основе иерархии классов;
3. структуризация знаний на декларативные и процедурные;
4. структуризация приложения на основе иерархии «часть/целое».

Структуризация общей задачи на связанные подзадачи

Модульная организация базы знаний составляет важную часть разработки прикладной системы, хотя трудно предложить единственно правильный способ разбиения системы на модули. Процесс эволюции прикладной системы может потребовать пересмотра и ее модульной структуры. В большинстве современных средств разработки сложных экспертных систем и в особенности динамических предусматривается поддержка разбиения базы знаний на модули.

Важность модульной организации экспертной системы определяется тем, что разбиение приложения на модули существенно ускоряет разработку, снижает затраты на сопровождение и поддержку, упрощает повторное использование модулей базы знаний в последующих разработках. С другой стороны, разбиение прикладной экспертной системы на модули несколько повышает накладные расходы на загрузку и сборку прикладной системы, например, восстановление после сбоев и перезапуск системы.

Структуризация предметной области на основе иерархии классов

Необходимость ускорения темпов разработки и модификации экспертной системы всегда являлась актуальной задачей прикладной инженерии знаний. Применение *объектно-ориентированного подхода* в современных экспертных системах естественным образом реализует возможность декомпозиции задачи на совокупность подзадач. Знания при этом подходе организованы в классы. Каждый класс определяется специфическим набором атрибутов. Классы организуются в иерархию классов. Каждый класс в иерархии наследует атрибуты и ограничения своего родительского класса. Обычно производный класс определяет дополнительные специфические атрибуты и (или) ограничения.

Структуризация знаний на декларативные и процедурные

По форме описания знания подразделяются на декларативные и процедурные.

Декларативные знания – это знания, которые записаны в памяти интеллектуальной системы так, что они непосредственно доступны для использования после обращения к соответствующему полю памяти. Обычно декларативные знания используются для представления информации о свойствах и фактах предметной области. По форме представления декларативные знания противопоставляются процедурным знаниям.

Процедурные знания – это знания, хранящиеся в памяти интеллектуальной системы в виде описания процедур, с помощью которых их можно получить. Обычно процедурные знания используются для представления информации о способах решения задач в проблемной области, а также различные инструкции, методики и т.п.

Структуризация приложения на основе иерархии «часть/целое»

Модульный принцип создания приложения предоставляет разработчику различные возможности разбиения приложения на подсистемы, легче поддающиеся сопровождению и модификации. Разбиение приложения на

модули упрощает процесс тестирования за счет использования групповой работы над тестируемой системой. Модульность также обеспечивает базовые возможности для повторного использования фрагментов системы.

На этапе *выполнения* осуществляется наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач.

Этап *тестирования* экспертной системы рассматривают в качестве заключительной фазы процесса разработки, операционное прототипирование, характеризующееся возможностью изменения целей проектирования в процессе разработки, предъявляет особые требования к доказательству корректности и соответствия разрабатываемой системы предъявляемым требованиям. По аналогии с технологией тестирования традиционных программных систем можно интерпретировать процесс верификации (логического тестирования) как альфа-тестирование программной системы, а концептуальное тестирование – как этап бета-тестирования, хотя тестирование экспертных систем принципиально отличается от тестирования традиционных систем. В то время как достаточно строгие предварительные спецификации традиционной системы позволяют программисту осуществлять эти работы (в особенности верификацию системы) самостоятельно, для тестирования экспертной системы необходимо привлекать эксперта в данной предметной области.

Выделяют три аспекта тестирования экспертных систем:

- тестирование исходных данных;
- логическое тестирование базы знаний;

- концептуальное тестирование прикладной системы.

Тестирование исходных данных включает проверку фактографической информации, служащей основой для проведения экспертизы. Очевидно, что наборы данных, используемых при тестировании, должны покрывать область возможных ситуаций, распознаваемых экспертной системой.

Логическое тестирование базы знаний заключается в обнаружении логических ошибок в системе продукций, не зависящих от предметной области, таких, как избыточные, циклические и конфликтные правила; пропущенные и пересекающиеся правила; несогласуемые и терминальные клаузы (несогласуемые условия). Формальный характер этих ошибок позволяет автоматизировать процесс логического тестирования. Существует большое количество инструментальных средств для верификации наборов правил и базы знаний в целом. Однако в ряде случаев, когда цепочки правил, используемых в процессе вывода, небольшие (от 3 до 10 правил), целесообразно проводить процесс верификации вручную.

Концептуальное тестирование проводится для проверки общей структуры системы и учета в ней всех аспектов решаемой задачи. На этом этапе проведение тестирования невозможно без привлечения конечных пользователей прикладной системы.

На этапе *опытной эксплуатации и внедрения* проверяется пригодность экспертной системы для конечного пользователя. Здесь система занимается решением всех возможных задач при работе с различными пользователями. К этому этапу следует переходить лишь после того, как система, по мнению эксперта, будет успешно решать все требуемые задачи, чтобы ошибки в решениях не создавали у пользователя отрицательное представление о системе. Пригодность системы для пользователя определяется в основном удобством работы с ней и ее полезностью.

ТЕМА 3. МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ

1. Представление знаний в экспертных системах
2. Семантические сети
3. Фреймовая модель
4. Продукционная модель
5. Логическая модель

1. Представление знаний в экспертных системах

Первый и основной вопрос, который надо решить при представлении знаний, - это вопрос определения состава знаний, т.е. определение того, «что представлять» в экспертной системе. Второй вопрос касается того, «как представлять» знания. Необходимо отметить, что эти две проблемы не являются независимыми. Действительно, выбранный способ представления может оказаться непригодным в принципе либо неэффективным для выражения некоторых знаний.

Вопрос «как представлять» можно разделить на две в значительной степени независимые задачи: как организовать (структурировать) знания и как представить знания в выбранном формализме.

Стремление выделить организацию знаний в самостоятельную задачу вызвано, в частности, тем, что эта задача возникает для любого языка представления и способы решения этой задачи являются одинаковыми (либо сходными) вне зависимости от используемого формализма.

Итак, в круг вопросов, решаемых при представлении знаний, будем включать следующие:

− определение состава представляемых знаний;

− организацию знаний;

− представление знаний, т.е. определение модели представления.

Состав знаний ЭС определяется следующими факторами:

− проблемной средой;

− архитектурой экспертной системы;

− потребностями и целями пользователей;

− языком общения.

В соответствии с общей схемой экспертной системы для ее функционирования требуются следующие знания:

− знания о процессе решения задачи (т.е. управляющие знания), используемые интерпретатором (решателем);

− знания о языке общения и способах организации диалога, используемые лингвистическим процессором (диалоговым компонентом);

− знания о способах представления и модификации знаний, используемые компонентом приобретения знаний;

− поддерживающие структурные и управляющие знания, используемые объяснительным компонентом.

Для динамической ЭС, кроме того, необходимы следующие знания:

1) знания о методах взаимодействия с внешним окружением;

2) знания о модели внешнего мира.

Зависимость состава знаний от требований пользователя проявляется в следующем:

− какие задачи (из общего набора задач) и с какими данными хочет решать пользователь;

− каковы предпочтительные способы и методы решения;

− при каких ограничениях на количество результатов и способы их получения должна быть решена задача;

− каковы требования к языку общения и организации диалога;

− какова степень общности (конкретности) знаний о проблемной области, доступная пользователю;

− каковы цели пользователей.

Вопросы организации знаний необходимо рассматривать в любом представлении, и их решение в значительной степени не зависит от выбранного способа (модели) представления. Выделим следующие аспекты проблемы организации знаний:

− организация знаний по уровням представления и по уровням детальности;

− организация знаний в рабочей памяти;

− организация знаний в базе знаний.

Для того чтобы экспертная система могла управлять процессом поиска решения, была способна приобретать новые знания и объяснять свои действия, она должна уметь не только использовать свои знания, но и обладать способностью понимать и исследовать их, т.е. экспертная система должна иметь знания о том, как представлены ее знания о проблемной среде. Если знания о проблемной среде назвать знаниями нулевого уровня представления, то первый уровень представления содержит метазнания, т.е. знания о том, как представлены во внутреннем мире системы знания нулевого уровня.

Первый уровень содержит знания о том, какие средства используются для представления знаний нулевого уровня. Знания первого уровня играют существенную роль при управлении процессом решения, при приобретении и объяснении действий системы. В связи с тем, что знания первого уровня не содержат ссылок на знания нулевого уровня, знания первого уровня независимы от проблемной среды.

Число уровней представления может быть больше двух. Второй уровень представления содержит сведения о знаниях первого уровня, т.е. знания о представлении базовых понятий первого уровня. Разделение знаний по уровням представления обеспечивает расширение области применимости системы.

Выделение уровней детальности позволяет рассматривать знания с различной степенью подробности. Количество уровней детальности во многом определяется спецификой решаемых задач, объемом знаний и способом их

представления. Как правило, выделяется не менее трех уровней детальности, отражающих соответственно общую, логическую и физическую организацию знаний. Введение нескольких уровней детальности обеспечивает дополнительную степень гибкости системы, так как позволяет производить изменения на одном уровне, не затрагивая другие. Однако наличие различных уровней препятствует распространению изменений с одного уровня на другие.

2. Семантические сети

Большая часть семантических моделей создана на базе семантических сетей. Этот термин обозначает целый класс подходов, для которых общим является использование графических схем с узлами, соединенными дугами. Узлы (вершины сети) представляют некоторые понятия (объекты, события, явления), а дуги – отношения между ними. Семантические модели являются объектно-ориентированными и обеспечивают в достаточной мере такой признак, как связность, реализуя четыре типа связей между объектами: классификацию, агрегирование, обобщение, ассоциацию.

Основная идея моделирования при помощи семантических моделей заключается в том, что модель представляет данные о реальных объектах и связях между ними прямым способом, что существенно облегчает доступ к знаниям: начиная движение от некоторого понятия, по дугам отношений можно достичь других понятий.

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

- класс—элемент класса;
- свойство—значение;
- пример элемента класса.

Наиболее часто в семантических сетях используются следующие отношения:

- связи типа «часть—целое»;
- функциональные связи (определяемые обычно глаголами «производит», «влияет» и др.);
- количественные (больше, меньше, равно и т. д.);
- пространственные (далеко от, близко от и др.);
- временные (раньше, позже и др.);
- атрибутивные связи (иметь свойство, иметь значение);
- логические связи (И, ИЛИ, НЕ);
- лингвистические связи и др.

В качестве примера на рисунке 6 показана весьма простая семантическая сеть для представления объекта «чайник».

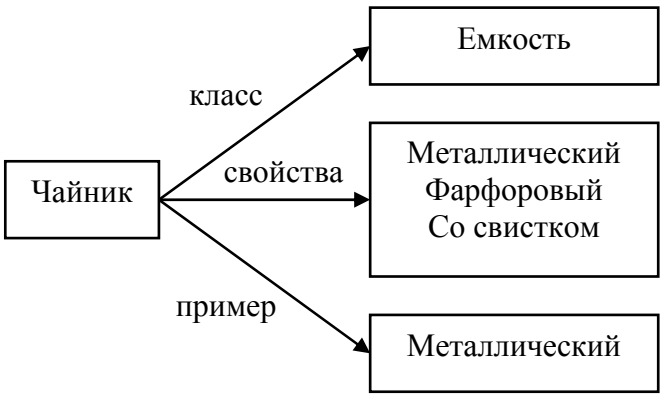


Рисунок 6 - Пример семантической сети

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, соответствующей поставленному вопросу. Подобного рода задачи решаются с помощью аппарата теории графов. Следует особо отметить роль фундаментальных признаков связей (рефлексивность, симметричность и транзитивность) в процессе вывода на сети. Так, например, связи вида «это есть» или «иметь частью» транзитивны, что позволяет говорить об установлении с помощью этой связи свойств иерархии наследования в сети.

Это означает, что элементы более низкого уровня в сети могут наследовать свойства элементов более высокого уровня в сети.

Использование семантических моделей позволяет представить в базе знаний экспертной системы знания о любой предметной области и осуществить автоматическое построение семантических сетей непосредственно из текста.

К основным достоинствам семантических моделей можно отнести: представление средств для выражения ограничений; описание связей между объектами; определение операций над объектами.

Накладывая ограничения на описание вершины дуг, можно получить сети различного вида. Если вершины не имеют собственной внутренней структуры, то такие сети называют простыми. В противном случае они являются иерархическими сетями. Одно из основных отличий иерархических семантических сетей от простых состоит в возможности разделить сеть на подсети и установить отношения не только между вершинами, но и между пространствами.

Характерной особенностью некоторых семантических моделей является интегрированное описание процедурной семантики и статической семантики – допустимые операции над объектами определяются совместно с определением структур данных.

Наряду с достоинствами семантические модели обладают некоторыми недостатками. В семантических сетях нет специальных средств, позволяющих определить временные зависимости, поэтому временные значения и события трактуются как обычные понятия. Произвольная структура и различные типы вершин и связей усложняют процедуру обработки информации. Стремление устранить эти недостатки послужило причиной появления особых типов семантических сетей: синтагматические цепи, сценарии, фреймы и т.п.

3. Фреймовая модель

Впервые термин «фрейм» был предложен в 70-е годы Марвином Минским, который определил его следующим образом:

«Фрейм – это структура данных, представляющая стереотипную ситуацию, вроде нахождения внутри некоторого рода жилой комнаты, или сбора на вечеринку по поводу дня рождения ребенка. К каждому фрейму присоединяется несколько видов информации. Часть этой информации – о том, как использовать фрейм. Часть о том, чего можно ожидать далее. Часть о том, что следует делать, если эти ожидания не подтвердятся».

Фрейм - это минимальное возможное описание сущности какого-либо явления, события, ситуации, процесса или объекта. Минимальность означает, что при дальнейшем упрощении описания теряется его полнота, она перестает определять ту единицу знаний, для которой предназначено. Например, слово «комната» вызывает у слушающих образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6-20 $м^2$». Из этого описания ничего нельзя убрать (например, убрав окна, получим уже не комнату), но в нем есть «дырки», - это незаполненные значения некоторых атрибутов - количество окон, цвет стен, высота потолка, покрытие пола и др.

Фрейм имеет определенную структуру, состоящую из множества элементов – слотов. Каждый слот в свою очередь, представляется определенной структурой данных, процедурой, или может быть связан с другим фреймом.

Структуру фрейма можно представить так:

ИМЯ ФРЕЙМА:

(имя 1-го слота: значение 1-го слота),

(имя 2-го слота: значение 2-го слота),

……..,

(имя N-го слота: значение N-го слота).

Ту же запись представим в виде таблицы, дополнив двумя столбцами.

Таблица 2. Описание фреймовой модели

Имя фрейма			
Имя слота	Тип слота	Значение слота	Присоединенная процедура

В качестве значения слота может выступать имя другого фрейма; так образуют сети фреймов.

Различают *фреймы-образцы*, или *прототипы*, хранящиеся в базе знаний, и *фреймы-экземпляры*, которые создаются для отображения реальных ситуаций на основе поступающих данных. *Прототип* – это уже не абстрактный образ, а наиболее типичный представитель своего класса, с обобщенными, но вполне конкретными, значениями своих свойств. Например, прототип понятия четырехугольник можно определить как фигуру, имеющую четыре угла.

Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире. Например:

- фреймы-структуры, для обозначения объектов и понятий (заем, залог, вексель);
- фреймы-роли (менеджер, кассир, клиент);
- фреймы-сценарии (банкротство, собрание акционеров, празднование именин);
- фреймы-ситуации (тревога, авария, рабочий режим устройства) и др.

Важнейшим свойством теории фреймов является заимствованное из теории семантических сетей наследование свойств. И во фреймах, и в семантических сетях наследование происходит по АКО-связям. Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т.е. переносятся, список и значения слотов. Возможно наследование свойств от нескольких прототипов. Такой вид наследования получи название «множественное наследование».

В качестве примера можно рассмотреть формирование понятия заказ товара (рисунок 7).

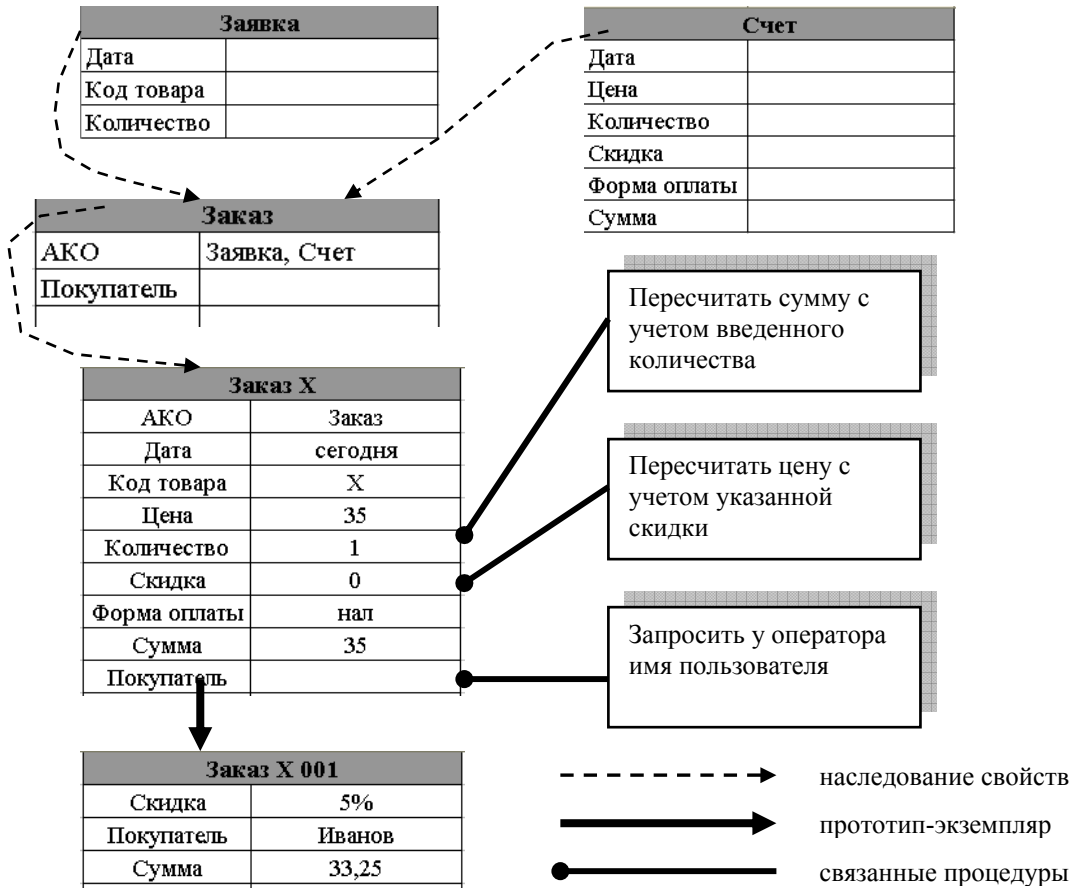


Рисунок 7 - Пример описания знаний с помощью фреймов

Основным преимуществом фреймов как модели представления знаний является способность отражать концептуальную основу организации памяти человека, а также гибкость и наглядность.

Рассмотрим примеры.

Фрейм-описание: [<овощи>, <помидоры, Болгария 30 т>, <перец, Венгрия 10 т>, <баклажаны, Молдова 20 т>]

Ролевой фрейм: [<доставить>, <что, прокат 10 т>, <откуда, Гомель>, <куда, Минск>, <чем, авто>, <когда, май>]

Во фрейме-описании в качестве имен слотов задан вид продукции, а значение слота характеризует массу и производителя конкретного вида продукции. В ролевом фрейме в качестве имен слотов выступают вопросительные слова, ответы на которые являются значениями слотов, для данного примера представлены уже описания конкретных фреймов, которые могут называться либо фреймами-примерами, либо фреймами-экземплярами.

Если в приведенном примере убрать значения слотов, оставив только имена, то получим так называемый фрейм-прототип.

Достоинства фрейма-представления во многом основываются на включении в него предположений и ожиданий. Это достигается за счет присвоения по умолчанию слотам фрейма стандартных ситуаций. В процессе поиска решений эти значения могут быть заменены более достоверными. Некоторые переменные выделены таким образом, что об их значениях система должна спросить пользователя. Часть переменных определяется посредством встроенных процедур, называемых внутренними. По мере присвоения переменным определенных значений осуществляется вызов других процедур. Этот тип представления комбинирует декларативные и процедурные знания.

Фреймовые модели обеспечивают требования структурированности и связности. Это достигается за счет свойств наследования и вложенности, которыми обладают фреймы, т.е. в качестве слотов может выступать система имен слотов более низкого уровня, а также слоты могут быть использованы как вызовы каких-либо процедур для выполнения.

Для многих предметных областей фреймовые модели являются основным способом формализации знаний.

4. Продукционная модель

Продукционная модель - модель, основанная на правилах, позволяющая представить знания в виде предложений типа: Если (условие), то (действие). Идея этого метода принадлежит Э. Посту (1943).

В качестве условия и действия в правилах может быть, например, предположение о наличии того или иного свойства, принимающее значение истина или ложь. При этом термин действие следует трактовать широко: это может быть как директива к выполнению какой-либо операции, рекомендация, или модификация базы знаний – предположение о наличии какого-либо производного свойства.

При использовании продукционной модели база знаний состоит из набора правил. Программа, управляющая перебором правил, называется машиной вывода. Чаще всего вывод бывает прямой (от данных к поиску цели) или обратный (от цели для ее подтверждения - к данным). Данные - это исходные факты, на основании которых запускается машина вывода - программа, перебирающая правила из базы.

Примером продукции может служить следующее выражение:

ЕСЛИ клиент работает на одном месте более двух лет,

ТО клиент имеет постоянную работу.

Как условие, так и действие правила могут учитывать несколько выражений, объединенных логическими связками И, ИЛИ, НЕ:

ЕСЛИ клиент имеет постоянную работу

И клиенту более 18 лет

И клиент НЕ имеет финансовых обязательств,

ТО клиент может претендовать на получение кредита.

Помимо продукционных правил база знаний должна включать и простые факты, поступающие в систему через интерфейс пользователя или выводимые в процессе поиска решения задачи. Факты являются простыми утверждениями типа «клиент работает на одном месте более двух лет». И когда в процессе интерпретации правил машиной вывода какой-либо факт согласуется с частью правила ЕСЛИ, то выполняется действие, определяемое частью ТО этого правила. Новые факты, добавляемые в базу знаний в результате действий, описанных в правилах, также могут быть использованы для сопоставления с частями ЕСЛИ других правил. Последовательное сопоставление частей правил ЕСЛИ с фактами порождает цепочку вывода. Цепочка вывода, полученная в результате последовательного выполнения правил П1 и П2 показана ниже (см. рисунок 8.). Эта цепочка показывает, как на основании правил и исходных фактов выводится заключение о возможности получения кредита. Цепочки

вывода экспертной системы могут быть предъявлены пользователю и помогают понять, как было получено решение.

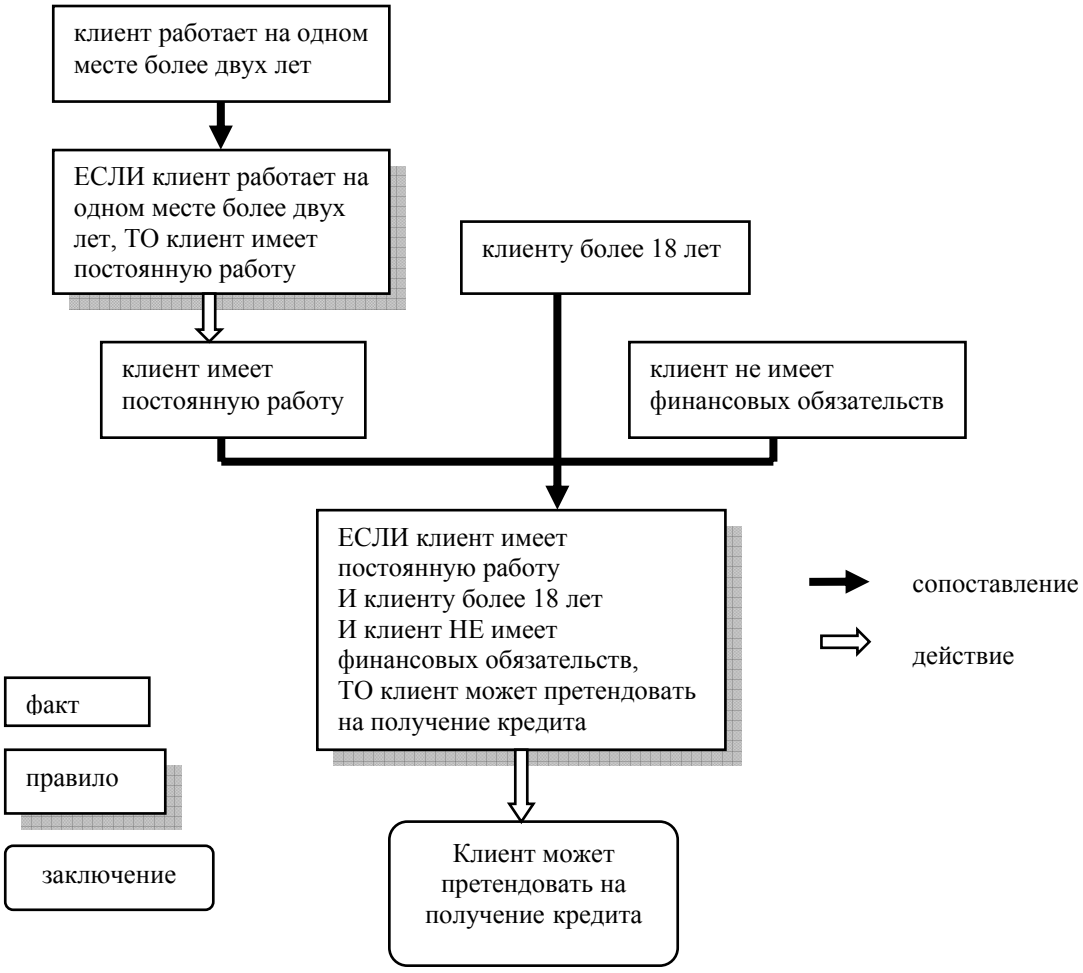


Рисунок 8 - Пример цепочки вывода

Представления знаний в виде продукций наиболее распространено в экспертных системах, так как запись знаний фактически ведется на подмножестве естественного языка.

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

Необходимо различать продукционные системы, управляемые данными (предусловиями правил) и продукционные системы, управляемые целями (действиями правил).

База знаний продукционной экспертной системы состоит из множества правил продукций (базы правил) $П=\{P_1, P_2, ..., P_m\}$ и конечного набора фактов (базы фактов) $A=(a_1, a_2, ..., a_n)$.

Если правило имеет вид $P_i=a_{i1}\ a_{i2}\ ...\ a_{is} \rightarrow a_m$, то это значит, что новый факт a_m имеет место (т.е. правило P_i применимо) при условии истинности всех фактов $a_{i1}...a_{is}$, определяющих правило P_i.

В случае, когда a_m – новый факт, имеет место модификация соответствующей базы фактов, а продукция P_i представляет собой декларативное (фактуальное) знание.

Возможен случай, когда правило продукции связано с выполнением какой-либо процедуры, а a_m – сообщение об окончании этого действия. В этом случае предусловия и действия являются утверждениями о данных, а вывод осуществляется в обратном направлении, т.е. от утверждений, которые должны быть доказаны.

Представление знаний в виде продукционных правил имеет недостатки и достоинства. Основным недостатком системы продукций является отсутствие внутренней структуры и зависимости шагов дедуктивного вывода от стратегии вывода, что делает ее трудно интерпретируемой.

Достоинствами продукционных систем является модульность организации знаний, независимость правил продукций, легкая модификация знаний на основе возможного удаления и добавления правил, возможность использования различных управляющих стратегий за счет отделения предметных знаний от управляющих.

5. Логическая модель

В основе логических моделей представления знаний лежит понятие формальной системы в виде четверки: M = <T,P,A,F>, где T – множество базовых символов теории M (например, буквы алфавита); P – множество синтаксических правил, посредством которых из базовых символов строятся

формулы; A – множество построенных формул, состоящих из аксиом; F – правила вывода, определяющие множество отношений между правильно построенными формулами.

В логическом подходе знания представляются посредством формул, которые строятся из предикатов, логических связок, кванторов и т.п. одни логические подходы ограничиваются классической логикой первого порядка, тогда как в других используется модальная логика, нечеткая логика, логика высших порядков и т.п.

Среди многих достоинств логического подхода необходимо отметить: стирание противопоставления между выводом и вычислением, что позволяет эффективно использовать метазнания; наличие четкой семантики и правил ввода.

Серьезной проблемой в логическом подходе является отсутствие структуры, так как данные представляются в виде совокупности линейных формул. К недостаткам логических моделей можно отнести следующее. На основе аппарата исчисления предикатов можно доказать существование объекта, обладающего определенными свойствами, т.е. логика первого порядка обеспечивает удобные средства описания в любой ситуации, которая определяется объектами и высказываниями, истинными относительно них. Но с другой стороны, в исчислении предикатов нет понятия процесса, что приводит к невозможности присвоения имени объекту в ходе логических преобразований и дальнейшим ссылкам на него, а также отсутствует возможность описания взаимосвязей двух ситуаций.

Логический и семантический аппарат теории исчисления предикатов не позволяет непосредственно решать такие проблемы, как совместное использование информации в альтернативных гипотезах и в различные моменты времени, создание новых структур в результате получения новых данных, планирование и т.д.

Таким образом, существует определенный круг задач, которые нельзя решать, используя только методологию исчисления предикатов. Возникает необходимость представления знаний на комбинированной основе, т.е. если некоторая часть системы представления знаний или вся эта система реализуются с помощью исчисления предикатов, то все равно остается ряд проблем, связанных с выбором оптимальной аксиоматической структуры и организации, обеспечивающей эффективность интеллектуальных операций.

Подробно процедурную интерпретацию логики высказываний на языке исчисления предикатов первого порядка можно рассмотреть на примере языка Пролог.

ТЕМА 4. МОДЕЛИ ПОИСКА РЕШЕНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ

1. Методы поиска решений
2. Поиск решений в одном пространстве
3. Поиск решений в иерархии пространств
4. Поиск в альтернативных пространствах
5. Поиск с использованием нескольких моделей
6. Выбор метода поиска решений

1. Методы поиска решений

Методы решения задач, основанные на сведении их к поиску, зависят от особенностей предметной области, в которой решается задача, и от требований, предъявляемых пользователем к решению. Особенности предметной области:

- объем пространства, в котором предстоит искать решение;
- степень изменяемости области во времени и пространстве (статические и динамические области);
- полнота модели, описывающей область, если модель не полна, то для описания области используют несколько моделей, дополняющих друг друга;
- определенность данных о решаемой задаче, степень точности (ошибочности) и полноты (неполноты) данных.

Требования пользователя к результату задачи, решаемой с помощью поиска, можно характеризовать:

1) количеством решений: одно решение, несколько решений, все решения.

2) свойствами результата: ограничения, которым должен удовлетворять полученный результат

3) способом его получения.

Существующие методы решения задач, используемые в экспертных системах, можно классифицировать следующим образом:

- методы поиска в одном пространстве - методы, предназначенные для использования в следующих условиях: области небольшой размерности, полнота модели, точные и полные данные;
- методы поиска в иерархических пространствах - методы, предназначенные для работы в областях большой размерности;
- методы поиска при неточных и неполных данных;
- методы поиска, использующие несколько моделей, предназначенные для работы с областями, для адекватного описания которых одной модели недостаточно.

Предполагается, что перечисленные методы при необходимости должны объединяться для того, чтобы позволить решать задачи, сложность которых возрастает одновременно по нескольким параметрам.

2. Поиск решений в одном пространстве

Методы поиска решений в одном пространстве обычно делятся на:

1) поиск в пространстве состояний;
2) поиск методом редукции;
3) эвристический поиск;
4) поиск методом «генерация-проверка».

Поиск в пространстве состояний

Задача поиска в пространстве состояний обычно формулируется в теоретико-графовой интерпретации.

Пусть задана тройка (S_0, F, S_T), где S_0 - множество начальных состояний (условия задачи), F - множество операторов задачи, отображающих одни состояния в другие, S_T - множество конечных (целевых) состояний (решений задачи).

Цель: определять такую последовательность операторов, которая преобразует начальные состояния в конечные.

Процесс решения можно изобразить в виде графа G=(X, Y), где X={x_0, $x_{1,...}$} - множество (в общем случае бесконечное) вершин графа, состояний, а Y - множество, содержащее пары вершин (x_i, x_j), где (x_i, x_j)$\in$X. Если каждая пара (x_i, x_j) неупорядочена, то ее называют ребром, а граф - неориентированным. Если для каждой пары (x_i, x_j) задан порядок (направление), то пару (x_i, x_j) называют дугой (ориентированным ребром), а граф называют *ориентированным (направленным)*. Вершины пары (x_i, x_j) называют *концевыми точками ребра* (дуги).

Поиск в пространстве состояний естественно представить в виде ориентированного графа. Наличие пары (x_i, x_j) свидетельствует о существовании некоторого оператора f (f$\in$F), преобразующего состояние, соответствующее вершине x_i, в состояние x_j. Для некоторой вершины x_i выделяем множество всех направленных пар (x_i, x_j)$\in$Y, т.е. множество дуг, исходящих из вершины x_i, (родительской вершины), и множество вершин (называемых дочерними вершинами), в которые эти дуги приводят. Множество дуг, исходящих из вершины x_i, соответствует множеству операторов, которые могут быть применены к состоянию, соответствующему вершине x_i.

В множестве вершин X выделяют подмножество вершин $X_0\subseteq X$, соответствующее множеству начальных состояний (S_o), и подмножество вершин $X_т\subseteq X$, соответствующее множеству конечных (целевых) состояний (S_T). Множество $X_т$ может быть задано как явно, так и неявно, т.е. через свойства, которыми должны обладать целевые состояния.

Отметим, что граф может быть задан явно и неявно. Неявное задание графа G стоит в определении множества $X_0\subseteq X$ (соответствующего множеству начальных состояний) и множества операторов, которые, будучи применимы к некоторой вершине графа, дают все ее дочерние вершины.

Итак, граф G задает пространство состояний, т.е. пространство, в котором осуществляется поиск решения. Построение пространства осуществляется с помощью следующего процесса. Берется некая вершина $x_0 \subseteq X$, к ней применяются все возможные операторы, порождающие все дочерние вершины. Этот процесс называют процессом раскрытия вершин. Если получена целевая вершина, то она не раскрывается. Процесс построения пространства состояний заканчивается, когда все нераскрытые вершины являются целевыми или терминальными (т.е. вершинами, к которым нельзя применить никаких операторов). В связи с тем, что пространство состояний может содержать бесконечное количество вершин, на практике процесс порождения пространства ограничивают либо временем, либо объемом памяти.

На практике требуется обеспечить полноту поиска, т.е. организовать поиск так, чтобы все целевые вершины были найдены, если они существуют. Надежным способом обеспечения полноты является полный перебор всех вершин. Для задания процесса перебора необходимо определить порядок, в котором будут перебираться вершины графа. Обычно выделяют два основных способа поиска:

- поиск в глубину (сначала раскрывается та вершина, которая была построена самой последней);
- поиск в ширину (вершины раскрываются в том же порядке, в котором они порождаются).

Целевые вершины помечены черными квадратами, а терминальные - белыми квадратами. При использовании каждого из способов могут быть найдены все решения. При переборе всего пространства оба метода будут анализировать одинаковое количество вершин, однако метод поиска в ширину будет требовать существенно больше памяти, так как он запоминает все пути поиска (а не один, как при поиске в глубину).

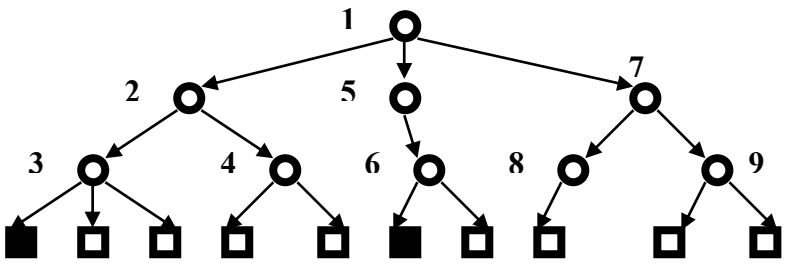


Рисунок 9 - Пространство состояний, построенное поиском в глубину

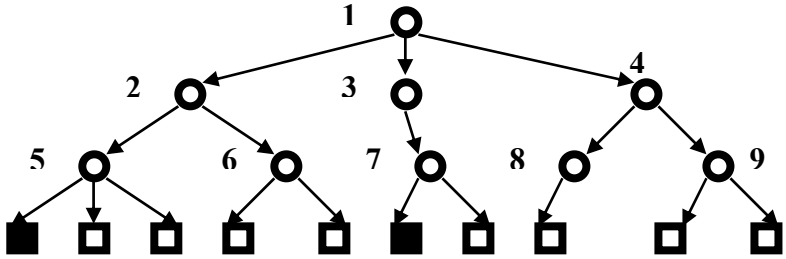


Рисунок 10 - Пространство состояний, построенное поиском в ширину

Поиск методом редукции

При поиске методом редукции решение задачи сводится к решению совокупности образующих ее подзадач. Этот процесс повторяется для каждой подзадачи до тех пор, пока каждая из полученного набора подзадач, образующих решение исходной задачи, не будет иметь очевидное решение. Процесс решения задачи разбиением ее на подзадачи можно представить в виде специального направленного графа G, называемого И/ИЛИ-графом; Каждой вершине этого графа ставится в соответствие описание некоторой задачи (подзадачи). В графе выделяют два типа вершин: конъюнктивные вершины и дизъюнктивные вершины.

Решение задачи при поиске методом редукции сводится к нахождению в И/ИЛИ-графе решающего графа.

Цель процесса поиска в И/ИЛИ-графе - показать, что начальная вершина разрешима, т.е. для этой вершины существует решающий граф. Определение *разрешимой вершины* в И/ИЛИ-графе можно сформулировать рекурсивно следующим образом:

1. Конечные (целевые) вершины разрешимы, так как их решение известно по исходному предположению.

2. Вершина ИЛИ разрешима тогда и только тогда, когда разрешима, по крайней мере, одна из ее дочерних вершин.

3. Вершина И разрешима только тогда, когда разрешима каждая из ее дочерних вершин.

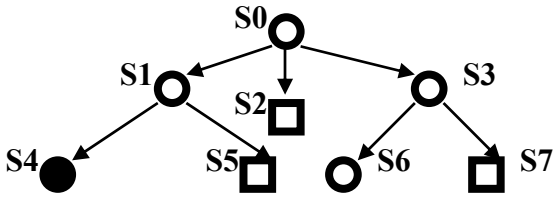


Рисунок 11 - Графическое представление процесса разбиения задачи на подзадачи

Решающий граф определяется как подграф из разрешимых вершин, который показывает, что начальная вершина разрешима (в соответствии с приведенным выше определением). На рисунке 11 разрешимые вершины зачернены, а неразрешимые оставлены белыми.

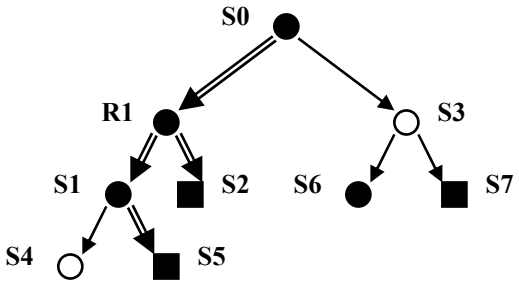
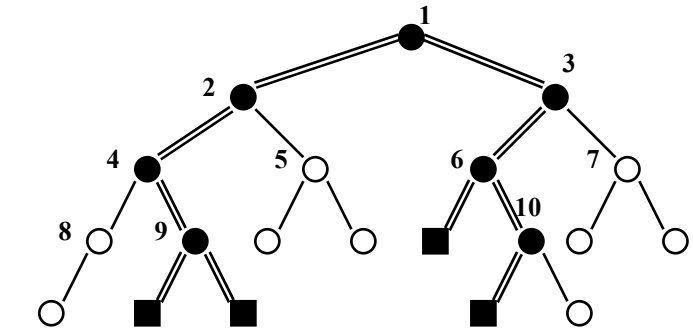
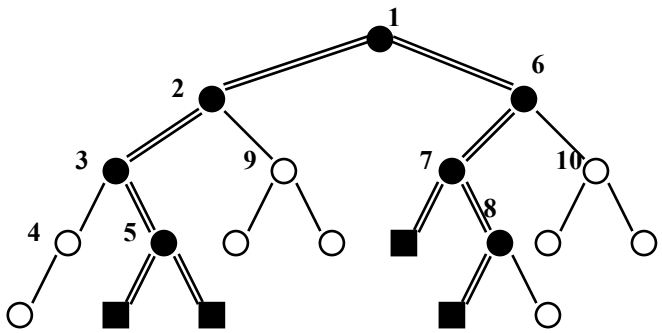


Рисунок 12 - Пример И/ИЛИ графа

Для графа И/ИЛИ, так же как для поиска в пространстве состояний, можно определить поиск в глубину и поиск в ширину как в прямом, так и в обратном направлении. На рисунке 13 приведен пример поиска в ширину (а) и поиска в глубину (б). На рисунке вершины пронумерованы в том порядке, в котором они раскрывались, конечные вершины обозначены квадратами, разрешимые вершины зачернены, дуги решающего графа выделены двойными линиями.



а



б

Рисунок 13 - Пример разбиения задач на подзадачи при поиске в ширину (а) или при поиске в глубину (б)

Эвристический поиск

При увеличении пространства поиска методы слепого поиска требуют чрезмерных затрат времени и (или) памяти. Это привело к созданию эвристических методов поиска, т.е. методов, использующих некоторую информацию о предметной области для рассмотрения не всего пространства поиска, а таких путей в нем, которые с наибольшей вероятностью приводят к цели.

Поиск методом «генерация-проверка»

Процесс поиска может быть сформулирован в терминах «генерация-проверка». Для осуществления процесса поиска необходимо генерировать очередное возможное решение (состояние или подзадачу) и проверить, не является ли оно результирующим.

3. Поиск в иерархии пространств

Методы поиска в одном пространстве не позволяют решать сложные задачи, так как с увеличением размера пространства время поиска экспоненциально растет. При большом размере пространства поиска можно попробовать разбить общее пространство на подпространства и осуществлять поиск сначала в них. Пространство поиска представлено иерархией пространств.

Методы поиска решения в иерархических пространствах обычно делятся:

1) поиск в факторизованном пространстве;
2) поиск в фиксированном множестве пространств;
3) поиск в изменяющемся множестве пространств.

Поиск в факторизованном пространстве

Во многих приложениях требуется найти все решения. Например - постановка диагноза. Пространство называется факторизованным, если оно разбивается на непересекающиеся подпространства (классы) частичными (неполными) решениями. Причем по виду частичного решения можно определить, что оно не приведет к успеху, т.е. что все полные решения, образованные из него, не приведут к целевым решениям. Поиск в факторизованном пространстве осуществляется на основе метола «иерархическая генерация-проверка». Если пространство поиска удается факторизовать, то поиск даже в очень большом пространстве можно организовать эффективно.

Поиск в фиксированном множестве пространств

Применение метода факторизации пространства ограничено тем, что для ряда областей не удается по частичному решению сделать заключение о его непригодности. Например, задачи планирования и конструирования. В этих случаях могут быть применены методы поиска, использующие идею абстрактного пространства. Абстракция должна подчеркнуть важные особенности рассматриваемой задачи, позволить разбить задачу на более

простые подзадачи и определить последовательность подзадач (план решения), приводящую к решению основной задачи.

Поиск в изменяющемся множестве иерархических пространств

В ряде приложений не удается все решаемые задачи свести к фиксированному набору подзадач. План решения задачи в данном случае должен иметь переменную структуру и не может быть сведен к фиксированному набору подзадач. Для решения подобных задач может быть использован метод нисходящего уточнения. Этот метод базируется на следующих предположениях:

- возможно осуществить частичное упорядочение понятий области, приемлемое для всех решаемых задач;
- решения, принимаемые на верхних уровнях, нет необходимости отменять на более нижних.

4. Поиск в альтернативных пространствах

Рассмотренные выше методы поиска исходят из молчаливой предпосылки, что знания о предметной области и данные о решаемой задаче являются точными и полными и для них справедливо следующее:

- все утверждения, описывающие состояние, являются истинными;
- применение оператора к некоторому состоянию формирует некоторое новое состояние, описание которого состоит только из истинных фактов.

Однако при решении любых практических задач и особенно при решении неформализованных задач распространена обратная ситуация. Эксперту приходится работать в условиях неполноты и неточности знаний (данных) и, как правило, в условиях дефицита времени. Когда эксперт решает задачу, он использует методы, отличающиеся от формальных математических рассуждений. В этом случае эксперт делает правдоподобные предположения, которые он не может доказать; тем самым вопрос об их истинности остается

открытым. Все утверждения, полученные на основе этих правдоподобных предположений, также не могут быть доказаны.

Итак, для того чтобы система могла делать умозаключения, основанные на здравом смысле, при работе с неполными (неточными) данными и знаниями, она должна быть способна делать предположения, а при получении новой информации, показывающей ошибочность предположений, отказываться как от сделанных предположений, так и от умозаключений, полученных на основе этих предположений. Мнение системы о том, какие факты имеют место, изменяется в ходе рассуждения, т.е. можно говорить о ревизии мнений. Таким образом, даже если рассматривать проблемную область как статическую, неполнота (и неточность) знаний и данных влечет за собой рассмотрение этой области при различных (и даже противоположных) предположениях, что, в свою очередь, приводит к представлению области в виде альтернативных пространств, соответствующих различным, возможно, противоречивым и (или) взаимодополняющим предположениям и мнениям.

Все неудачи, возникшие при поиске в одном направлении, не запоминаются при переходе к поиску в другом направлении. Та же самая причина неудачи может заново обнаруживаться и на новом направлении.

Осуществлять возврат целесообразно не к состоянию, непосредственно предшествующему данному, а к тому состоянию, которое является причиной возникновения неудачи. В используемых нами терминах причиной неудач являются предположения, т.е. недоказуемые утверждения. Поэтому при обнаружении неудачи необходимо возвращаться в состояние, где это предположение было сделано, и испытывать другое предположение.

Этот метод поиска называют поиском, направляемым зависимостью.

5. Поиск с использованием нескольких моделей

Все методы поиска, рассмотренные до сих пор, использовали при представлении проблемной области какую-то одну модель, т.е. рассматривали

область с какой-то одной точки зрения. При решении сложных задач в условиях ограниченных ресурсов использование нескольких моделей может значительно повысить мощность системы. Объединение в одной системе нескольких моделей дает возможность преодолеть следующие трудности:

- переход с одной модели на другую позволяет обходить тупики, возникающие при поиске в процессе распространения ограничений;
- использование нескольких моделей позволяет в ряде случаев уменьшить вероятность потери хорошего решения (следствие неполного поиска, вызванного ограниченностью ресурсов) за счет конструирования полного решения из ограниченного числа частичных кандидатов путем их расширения и комбинации;
- наличие нескольких моделей позволяет системе справляться с неточностью (ошибочностью) данных.

Следует отметить, что использование нескольких моделей требует дополнительных знаний о том, как создавать и объединять различные точки зрения.

6. Выбор метода поиска решений

Выбор метода решения задачи зависит, прежде всего, от сложности задачи, которая определяется особенностями проблемной области и требованиями, предъявляемыми пользователем к решению задачи. Для преодоления трудностей, вызванных большим пространством поиска, используются методы, основанные на введении иерархии пространств (конкретных, абстрактных и метапространств). Простейший из этих методов основывается на факторизуемости пространства решений, что позволяет производить раннее отсечение. Метод обеспечивает получение всех решений. Если пространство поиска не удается факторизовать, но при этом не требуется получать все решения или выбирать лучшее, то могут быть применены методы, использующие иерархию однородных абстрактных пространств. Если

пространство поиска таково, что любая задача может быть сведена к известной заранее последовательности подзадач, то используется фиксированное абстрактное пространство.

Эффективность этого метода определяется возможностью использовать безвозвратную стратегию. В случае если подзадачи взаимозависимы, т.е. для решения некоторой подзадачи может требоваться информация, получаемая другой подзадачей, и подзадачи не могут быть упорядочены, целесообразно применять принцип наименьших свершений. Этот подход позволяет приостанавливать решение подзадачи, для которой недостает информации, переходить к решению другой подзадачи и возвращаться к исходной задаче, когда отсутствующая информация станет доступной.

Для преодоления трудностей, вызванных неполнотой и (или) неточностью данных (знаний), используют вероятностные, размытые и точные методы. Все эти методы основываются на идее увеличения надежности путем комбинирования фактов и использования метазнаний о возможностях комбинирования фактов.

Для преодоления неадекватности модели проблемной области используются методы, ориентированные на использование нескольких моделей. Эти методы позволяют объединить возможности различных моделей, описывающих проблемную область с различных точек зрения. Кроме того, использование нескольких моделей позволяет уменьшить вероятность потери хорошего решения, несмотря на неполноту поиска, вызванную ограниченностью вычислительных ресурсов.

ТЕМА 5. НЕЧЕТКАЯ ЛОГИКА

1. Понятие нечеткой логики и нечетких систем
2. Нечеткие множества и лингвистические переменные
3. Операции с нечеткими множествами
4. Нечеткие алгоритмы

1. Понятие нечеткой логики и нечетких систем

Системы искусственного интеллекта, основанные на использовании математического аппарата нечеткой логики, являются наиболее простыми представителями систем искусственного интеллекта. Данные системы принято называть также термином «нечеткие» системы. Обычно они используются для выработки (автоматической) командных или управляющих воздействий на управляемые объекты, не относящиеся к категории объектов ответственного назначения. Характерной (отличительной) особенностью нечетких систем (по сравнению с другими классами интеллектуальных систем) является использование правил логического вывода, «заложенных» в них человеком. Данные правила относятся к простейшим продукционным правилам, используемых в экспертных системах (еще более продвинутый класс интеллектуальных систем) и имеют вид: ЕСЛИ (словесная запись условия), ТО (словесная запись действия).

Так нечеткие интеллектуальные системы можно использовать:

а) в объектах бытовой техники: стиральных машинах, микроволновых печах, холодильниках и т.д.;

б) в промышленных и хозяйственных объектах: системах управления паровыми и водяными котлами, строительными кранами, устройствами транспортировки грузов и т.д.;

в) в компьютерных играх для формирования интеллекта компьютера и т.д.

Нечеткие системы используются тогда, когда:

а) неопределенность наших знаний о рассматриваемом объекте носит сравнительно простой (по аналогии с вероятностной неопределенностью – «хороший») и преимущественно количественный характер и может быть описана с помощью функции принадлежности (аналогов функции распределения в теории вероятности);

б) возможно применение для формирования управляющих воздействий (команд) математически строго определенных правил нечеткого вывода (аналогично известным правилам логического вывода в булевой алгебре);

в) используется словесная или лингвистическая форма (достаточно простая) представления управляющих воздействий (команд) ограниченного потока слов и предложений словесного языка, типа ЕСЛИ (имя условия), ТО (имя действия).

Нечеткие системы предназначены для управления объектами в условиях особого вида неточности (не путать с неполнотой) наших знаний об объекте. Данную неточность иногда называют неточностью лингвистических знаний (или лингвистической неопределенностью). Лингвистическая неопределенность (неточность лингвистических знаний) складывается из

а) неточности отнесения измеренного значения количественной переменной объекта (скорость, путь, время пути) к одному из понятий (высокое, большой, малое), представляемых в виде слов (термов). Отметим, что данные слова (термы) называются лингвистическими переменными (ошибка фаззификации);

б) ошибка дефаззификации, появляющаяся вследствие неточности обратной процедуры перевода значения лингвистической переменной в числовое значение непрерывной количественной переменной;

в) неточность лингвистических процедур нечеткого вывода (логического вывода с помощью математического аппарата нечеткой логики), не

позволяющих охватить все смысловое возможных формулировок на естественном языке управляющих воздействий (команд).

Рассмотрим в качестве объекта управления водяной котел (бойлер), предназначенный для нагрева воды с целью отопления помещений. Топливо (газ, мазут) поступает в камеру сгорания (печь), которая нагревает воду в котле. В качестве входной величины (управляющего воздействия) данного объекта выступает расход топлива *x* в камеру сгорания, в качестве выходной величины (управляемой переменной) выступает температура воды *y* [^{o}C] (рисунок 14).

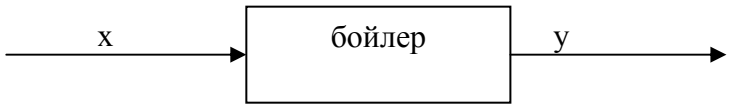


Рисунок 14 - Процесс описания нагрева воды

При обычном подходе к описанию процесса нагрева воды в котле уравнений (линейных, нелинейных, дифференциальных), величины *x* и *y* принимают бесконечное множество значений из интервалов $x_{max} \geq x \geq x_{min}$, $y_{max} \geq y \geq y_{min}$. Например, $10 \frac{кг}{сек} \geq x \geq 1 \frac{кг}{сек}$, $C \geq y \geq 10^{0} C$.

При описании процесса нагрева воды с помощью аппарата булевой алгебры, температура воды описывается логической переменной *a*, принимающей два значения, отображаемые кодами или словами: 0 (низкая температура воды) и 1 (высокая температура воды):

$$a = \begin{cases} 0, & если \quad y < y_{ном} = 80^{0} \\ 1, & если \quad y \geq y_{ном} = 80^{0} \end{cases}$$

Аналогично можно ввести логическую переменную *b* , описывающую расход топлива в камеру сгорания. Переменная *b* также принимает два значения 0 (низкий расход топлива) и 1 (высокий расход топлива).

$$b = \begin{cases} 0, & если \quad x < x_{ном} = 5 \frac{кг}{сек} \\ 1, & если \quad x \geq x_{ном} = 5 \frac{кг}{сек} \end{cases}.$$

где $x_{ном}$ - номинальный (штатный) расход топлива.

На рисунке 15 приведены графики, иллюстрирующий преобразование переменных *x* и *y* в логические переменные *a* и *b*.

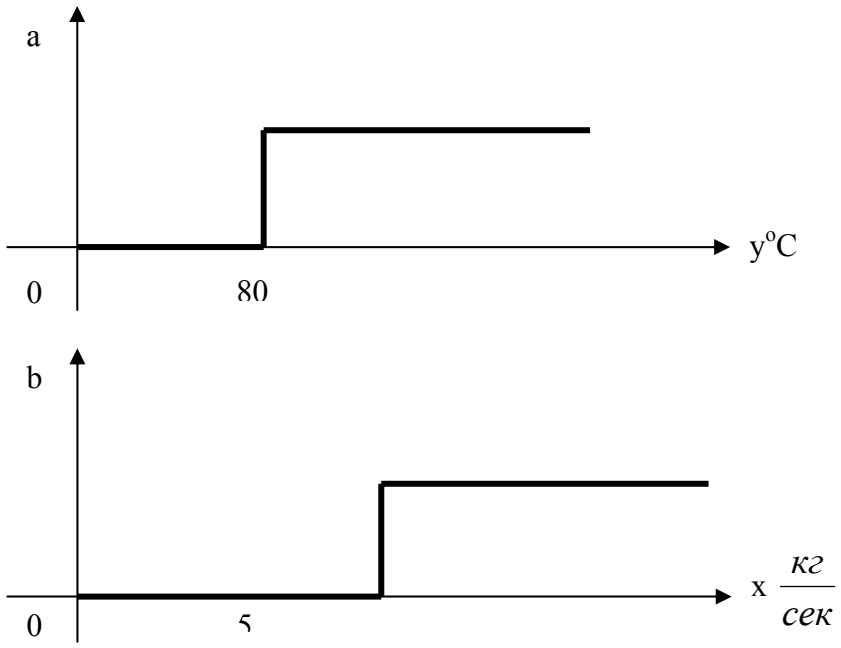


Рисунок 15 - Преобразование переменных x и y в логические переменные a и b

При использовании аппарата алгебры – логики (четкой логики), граница между множествами значений переменных, соответствующим различным смысловым понятиям (высокая и низкая температура, высокий и низкий расход топлива), является вполне четкой и определенной.

Однако в реальности в большинстве случаев четкую границу между данными понятиями провести трудно. Например, часть людей могут считать высокой температуру воды (в кранах, батареях) начиная со значения 70°С и выше, другая часть - 75°С и т.д. То же касается и расхода топлива в камере сгорания: высокий расход топлива различными операторами (работниками бойлера) может считаться начиная со значения 4кг/сек и выше, 4.5 кг/сек и выше, 5 кг/сек и выше.

В этом случае графики функций принадлежности числовых переменных *x* и *y* к одному из двух классов (значений логических переменных *a* и *b*), будут иметь следующий вид (рисунке 16).

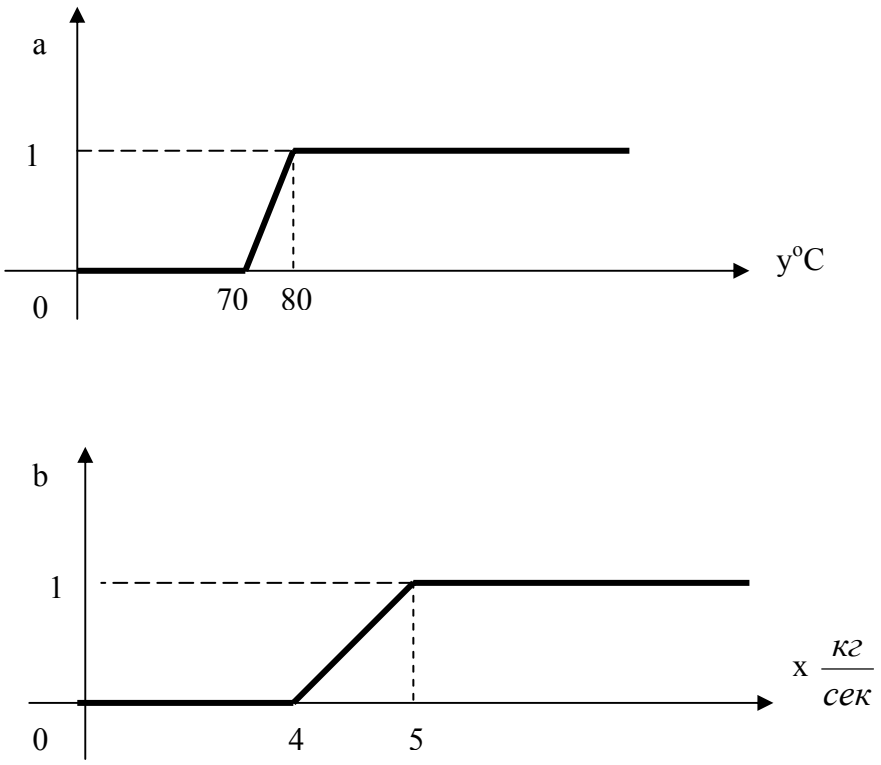


Рисунок 16 - Графики функций принадлежности числовых переменных *x* и *y* к одному из двух классов

Как видно в данном случае на графиках имеются т.н. зоны неопределенности: это интервалы $80^o \geq y \geq 70^o$ и $5 \geq x \geq 4$. Если переменные *x* и *y* находятся в данных интервалах, то они могут одновременно принадлежать как первому классу значений логической переменной (*a*=0, *b*=0), так и второму классу значений (*a*=1, *b*=1). Иными словами, в этом случае граница между множествами значений переменных *x* и *y*, соответствующим различным смысловым классам размывается и является нечеткой. Отметим, что подобная нечеткость при описании реальных объектов и систем возникает вследствие недостаточности располагаемых знаний (информации) о его свойствах.

Для описания подобных систем Л. Заде в 1965 г. разработал теорию размытых (нечетких) множеств (теорию нечеткой логики), которая в настоящее время лежит в основе построения особого класса интеллектуальных систем, получивших название нечетких систем. Нечеткие системы реализуют т.н. нечеткие (лингвистические) алгоритмы управления реальными (физическими) объектами. Обобщенная структурная схема нечеткой системы представлена на

рисунке 17, где x, y - непрерывные переменные, представляющие собой управляющее воздействие объекта (расход топлива в камеру сгорания) и управляющую переменную (температура воды); a и b – нечеткие логические переменные, принимающие значения a=(низкая температура, высокая температура), b=(низкий расход топлива, высокий расход топлива); Ф – устройство преобразования переменной x (температура воды) в логическую переменную a, данная процедура называется фазификацией, ДФ – устройство преобразования логической переменной b в непрерывную переменную y (расход топлива), данная процедура называется дефаззификацией.

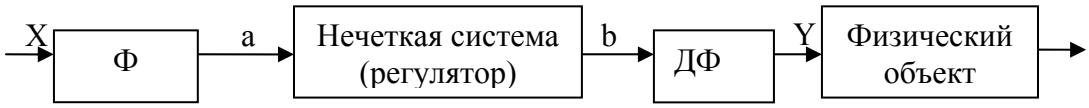


Рисунок 17 - Обобщенная структурная схема нечеткой системы

2.Нечеткие множества и лингвистические переменные

Прежде чем дать строгое толкование понятию «нечеткое множество», обратимся к следующему примеру. Допустим, что объектом нашего исследования является множество «взрослых людей», к которому формально можно отнести всех людей, достигших совершеннолетия (18 лет). Если обозначить через переменную x - «возраст человека», а функцию $\mu(x)$ задать следующим образом

$$\mu(x)=\begin{cases}1, если & x \geq 18\\ 0, если & x \leq 18\end{cases}$$

Тогда множество «взрослых людей» A может быть задано с помощью выражения $A=\{x(\mu(x))=1\}$, $x \in X$, где X – множество всех возможных значений x. Другими словами, множество A образуют такие «объекты» (элементы), для которых указанная функция $\mu(x)$, называемая функцией принадлежности, принимает значение 1. Напротив, те значения $x \in X$, для которых $\mu(x)=0$, не принадлежат множеству A.

Очевидно, что двухзначная логика типа «да-нет», определяемая функцией принадлежности $\mu(x): X \to \{0;1\}$ не учитывает возможного разброса мнений различных субъектов относительно границ исследуемого множества *A* (влияний чисто биологических факторов, национальных особенностей и др.). Поэтому, более естественным является задание функции принадлежности в виде некоторой непрерывной или гладкой (плавной) зависимости, определяющая плавный переход из одного крайнего состояния в другое (т.е. от принадлежности элемента множеству *A* до принадлежности ему). В данном случае функция принадлежности $\mu(x)$ ставит в соответствие каждому элементу $x \in X$ число $\mu(x)$ из интервала [0,1], описывающее степень принадлежности *x* множеству *A*.

Заданное таким образом множество пар $A = \{(x, \mu(x)) \mid x \in X\}$ называется нечетким (или размытым) множеством.

Перечислим *основные свойства нечетких множеств*.

Будем называть *носителем A* множество тех его элементов *x*, для которых $\mu(x)$ положительна, т.е. *supp A*=$\{x \in X \mid \mu(x) > 0\}$

Если носитель нечеткого множества *A* состоит из единственной точки *x*, то такое множество называется одноточечным. Данное одноточечное множество обычно записывают в виде $A = \mu / x$, где μ - степень принадлежности *x* множеству *A*.

Если носитель *A* состоит из конечного числа элементов, то для записи такого дискретного множества используется выражение:

$$A = \mu_1 / x_1 + \mu_2 / x_2 + ... + \mu_n / x_n$$

или

$$A = \sum_{i=1}^{n} \mu_i / x_i$$

где числа μ_i - степени принадлежности элементов x_i к множеству А.

Заметим, что знак «+» в формуле обозначает объединение, а не арифметическое суммирование.

Если носитель нечеткого множества А состоит из бесконечного числа точек, например, представляет собой некоторый интервал (*а, в*) на числовой оси *x*, то функция принадлежности $\mu(x)$ обычно задается графически или в виде аналитической зависимости.

Возможен и табличный способ задания нечеткого множества *A*.

x_i	14	16	18	20	22
μ_i	0.1	0.3	0.5	0.8	1.0

Например, таблица обозначает, что носитель А состоит из 5 элементов x_1=14, x_2=16, x_3=18, x_4=20, x_5=22, степени принадлежности которых множеству А равны соответственно: 0.1, 0.3, 0.5, 0.8, 1.0.

Точка перехода A – это элемент *x* множества *A*, для которого $\mu(x)$=0.5.

α - *разрезом нечеткого множества* (A_α) называется множество элементов *x*, для которых функция принадлежности $\mu(x)$ принимает значения не меньше заданного числа α $(0<\alpha<1)$: $A_\alpha = \{x \in X \mid \mu(x) \geq \alpha\}$

Высота нечеткого множества A находится как точная верхняя грань (максимум) его функции принадлежности: $h(A) = \sup_{x \in X} \mu(x)$

Если высота нечеткого множества равна 1, то такое множество называется нормализованным. В том случае, когда высота нечеткого множества А меньше 1, такое множество называется субнормальным. От субнормального множества к нормализованному можно перейти путем деления его функции принадлежности $\mu(x)$ на высоту $\sup_{x \in X} \mu(x)$.

Пример.

Допустим, что для косвенного измерения скорости вращения вала нагруженного электропривода используется выходное напряжения генератора постоянного тока. Известно значение этого напряжения x=5В. Кроме того известно, что ошибка такого измерения составляет $\pm$1В. Тогда переход от

четкого значения x=5 к нечеткому множеству «x=приблизительно 5» осуществляется следующим образом (рисунок 18).

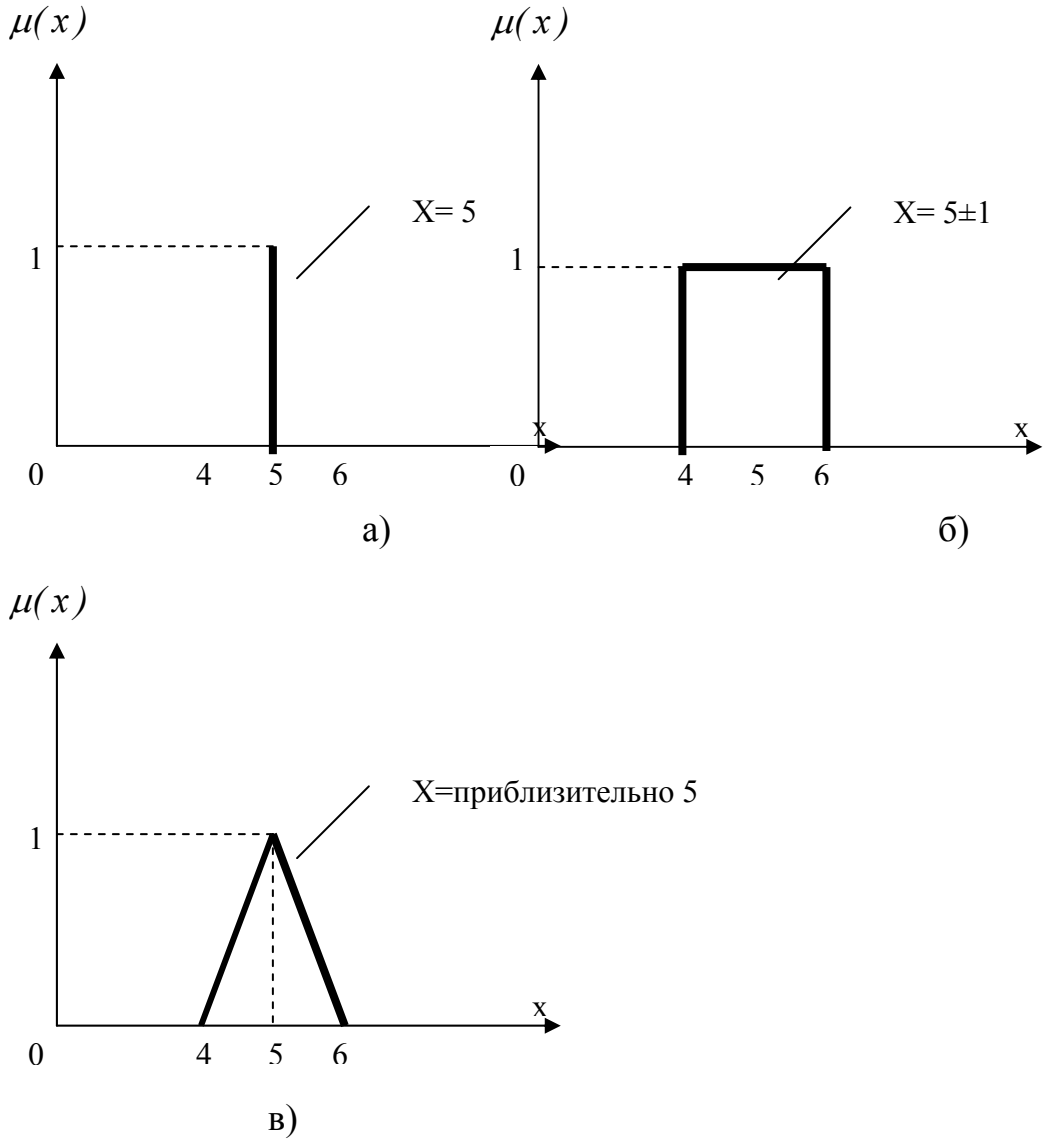


Рисунок 18 - Функции принадлежности

Представленный на рисунке 18 (а – в) процесс перехода от «четкого» (т.е. измеренного) значения x=5 к его «нечеткой» интерпретации x=«приблизительно 5» называется фаззификацией.

Вопрос о том, как выбирается или задается в каждом конкретном случае функция принадлежности $\mu(x)$ и какой она имеет смысл, остается в значительной мере спорным и мало изученным. Наиболее распространенным является мнение, что $\mu(x)$ может рассматриваться как коэффициент уверенности эксперта в том, что элемент x принадлежит множеству A.

Одним из ключевых понятий нечеткой логики является понятие лингвистической переменной. Суть данного понятия состоит в том, что конкретные значения числовой переменной *x* обычно подвергающиеся субъективной оценке человеком, причем результат такой оценки выражается на естественном языке. Так, переменная «рост (высота) человека» может характеризоваться одним из следующих описаний: «маленький», «невысокий», «среднего роста», «высокий». Другая переменная «скорость движения автомобиля» может быть «малой», «средней», «большой» и т.д. Каждый из приведенных здесь термов может рассматриваться как символ некоторого нечеткого подмножества в составе полного множества значений *x*. Переменные, значениями которых являются термы (слова, фразы, предложения), выраженные на естественном языке называют лингвистическими переменными.

Задать нечеткое подмножество A_i, соответствующее определенному (i-тому) терму (значению) лингвистической переменной – это значит задать область определения числовой переменной *x* и функцию принадлежности элемента x подмножеству A_i.

Пример.

Рассмотрим лингвистическую переменную «яркость». Будем полагать, что различные значения физической переменной *x* (яркость, в кд/м^2) могут быть охарактеризованы набором из 5 нечетких подмножеств (значений лингвистической переменной): «очень темно», «темно», «средне», «светло», «очень светло».

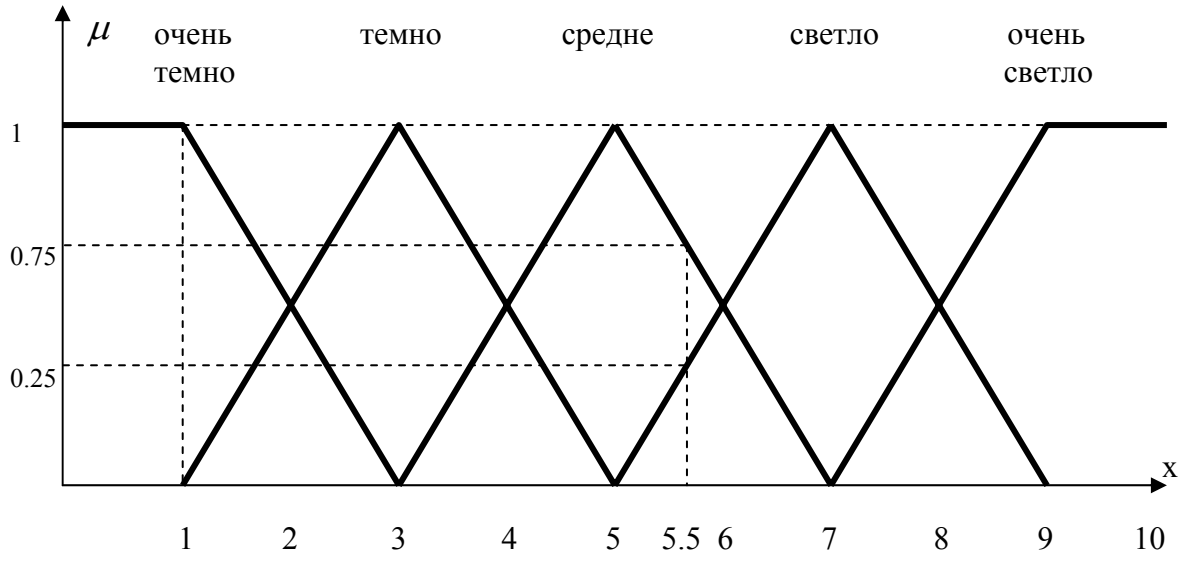


Рисунок 19 - Функции принадлежности множества

Допустим, что фактическое значение яркости равно x=5.5. тогда в соответствии с рисунком 19., это значение относится одновременно к двум термам (подмножествам) – «средне» и «светло» со степенями принадлежности $\mu_{средне}(5.5) = 0.75$ и $\mu_{светло}(5.5) = 0.25$ соответственно. $\mu_{средне} > \mu_{светло} \Rightarrow$ «светло».

3. Операции с нечеткими множествами

Определение операций, выполняемых с нечеткими множествами, во многом аналогично операциям с обычными (четкими) множествами.

Эквивалентность. Два нечетких множества A и B эквивалентны (это обозначается как A=B) тогда и только тогда когда для всех $x \in X$ имеет место $\mu_A(x) = \mu_B(x)$

Включение. Нечеткое множество A содержится в нечетком множестве B $(A \subseteq B)$ тогда и только тогда когда $\mu_A(x) \le \mu_B(x)$, $\forall x \in X$ (1)

Объединение или дизъюнкция $A \cup B$ двух нечетких множеств A и B соответствует логической операции «ИЛИ» и определяется как наименьшее нечеткое множество, содержащее оба множества A и B. Функция принадлежности для этого находиться с помощью операции взятия максимума (рисунок 20б): $\mu_{AUB}(x) = \max\{\mu_A(x), \quad \mu_B(x)\}, \forall x \in X$ (2).

Пересечение или конъюнкция $A \cap B$ соответствует логической операции «И» и определяется как наибольшее нечеткое множество являющееся одновременно подмножеством обоих множеств. Функция принадлежности множества $A \cap B$ выражается с помощью операции нахождения минимума (рисунок 20в): $\mu_{A\cap B}(x) = \min\{\mu_A(x), \quad \mu_B(x)\}, \forall x \in X$ (3).

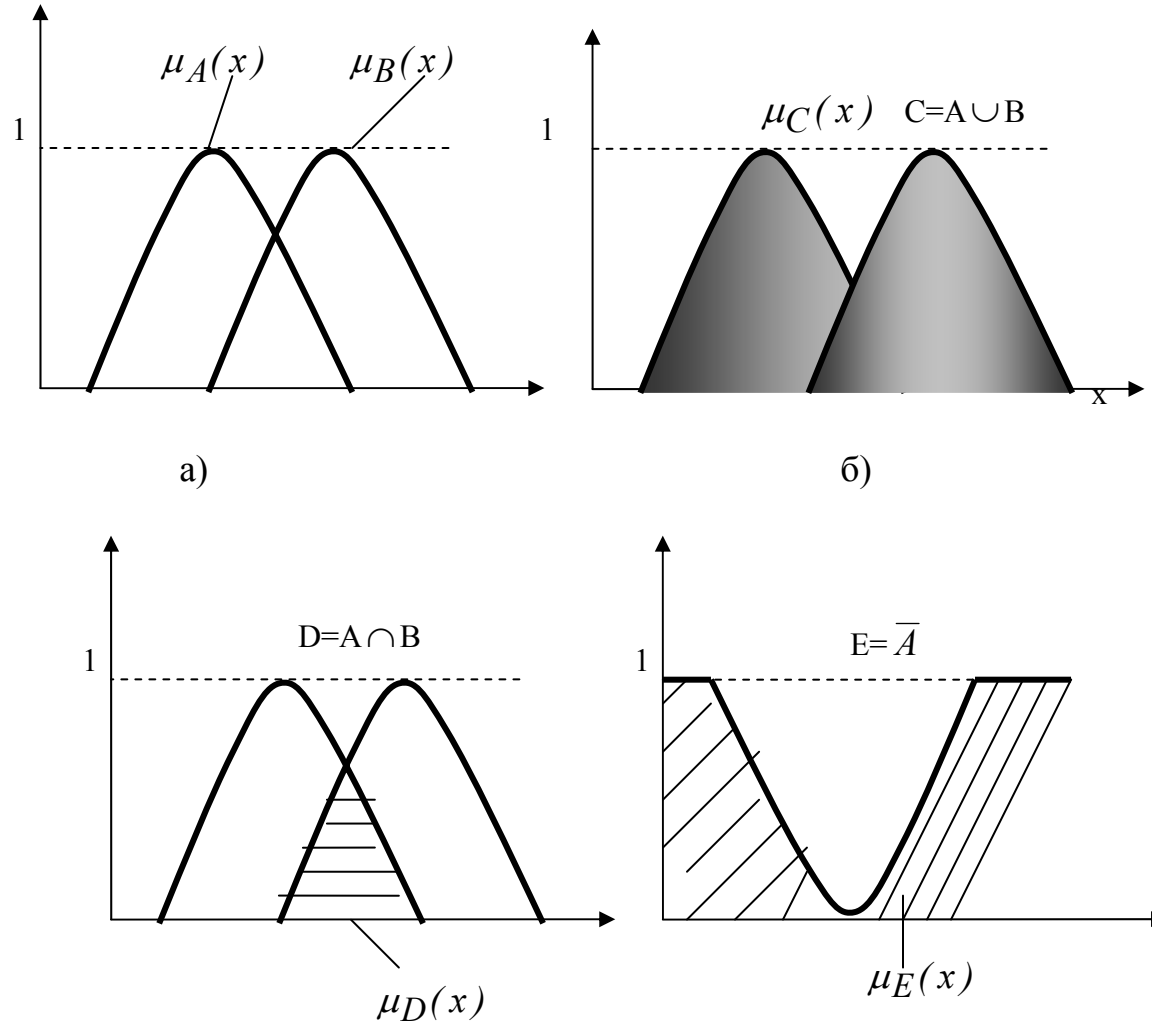


Рисунок 20 - Операции с нечеткими множествами

Дополнение нечеткого множества А, обозначаемое через $\overline{A}$, соответствует логическому отрицанию «НЕ» и определяется формулой (рисунок 20г): $\mu_{\overline{A}}(x) = 1 - \mu_A(x), \quad \forall x \in X$ (4)

Легко видеть, что применительно к классическим четким множествам, для которых функции принадлежности принимают только 2 значения 0 и 1, формулы 2-4 определяют известные операции логического «ИЛИ», «И», «НЕ».

Приведем определения еще двух достаточно распространенных операций над нечеткими множествами – алгебраического произведения и алгебраической суммы нечетких множеств.

Алгебраическое произведение $A \cdot B$ нечетких множеств A и B определяется следующим образом: $\mu_{AB}(x) = \mu_A(x) \cdot \mu_B(x), \quad \forall x \in X$ (5)

Алгебраическая сумма $A \oplus B$:

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x), \quad \forall x \in X \qquad (6)$$

Кроме этих операций существуют еще несколько более специфических операций для лингвистических переменных.

Операция концентрации $CON(A)$ определяется как алгебраическое произведение нечеткого множества A на самого себя:

$$CON(A) = A^2,$$
$$\mu_{CON_A}(x) = \mu_A^2(x) \qquad (7) – (8)$$

В результате применения этой операции к множеству A уменьшаются $(\mu < 1)$ степени принадлежности элементов к этому множеству. Причем если $\mu(x) \approx 1$, то это уменьшение мало. А для элементов с малой степенью принадлежности – относительно велико. В естественном языке применение этой операции к тому или иному значению лингвистической переменной A соответствует использование усиливающего терма «очень» (например, «очень высокий», «очень старый» и т.д.)

Операция растяжения $DIL(A)$ определяется как

$$DIL(A) = A^{0.5} \qquad (9)$$

$$DIL(A) = \sqrt{\mu_A(x)} \qquad (10)$$

Действие этой операции противоположно действию операции концентрации и соответствует неопределенному терму «довольно», выполняющему функцию ослабления следующего за ним (основного) терма А: «довольно высокий», «довольно старый» и т.п.

Можно ввести и другие аналогичные по смыслу операции, позволяющие модифицировать значения лингвистической переменной увеличивая, таким образом, их количество.

Так, терм «более чем» можно определить следующим образом: $\{(x, \mu_A(x)^{1.25}\}, \quad x \in X$

Составной терм «очень очень»: $CON(CON(A)) = \{(x, \mu_A^4(x)\}, \quad x \in X$

Рассмотрим применение указанных операций на следующем наглядном примере.

Пусть переменная *x* характеризует возраст человека, *x* – интервал [0, 100]. Тогда нечеткие подмножества, описываемые термами «молодой» и «старый» можно представить с помощью функций принадлежности (рисунок 21).

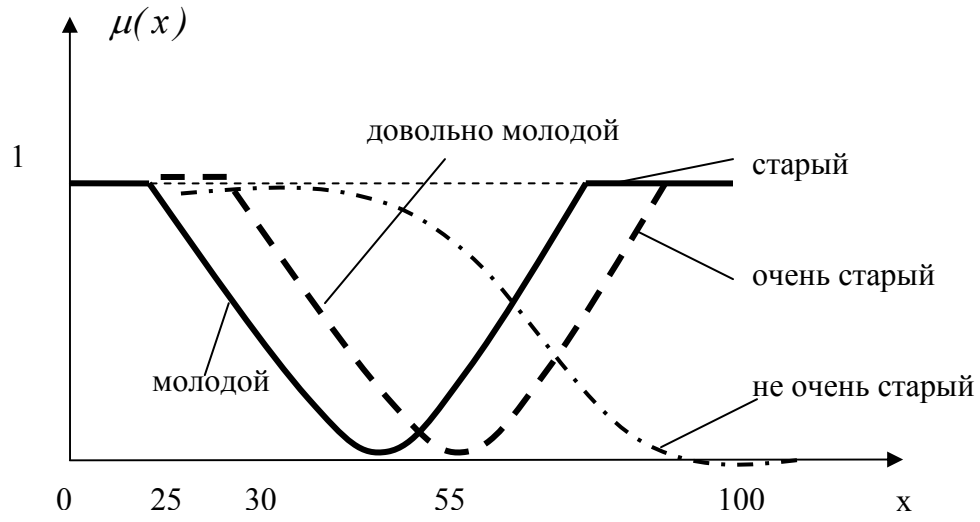


Рисунок 21 - Представление нечеткого множества с помощью функции принадлежности

$$\mu_{молодой}(x)=\begin{cases}1, & для \quad 0\le x\le 25\\ \left[1+\left(\frac{x-25}{5}\right)^2\right]^2, & для \quad 25\le x\le 10\end{cases}$$

$$\mu_{старый}(x)=\begin{cases}0, & для \quad 0\le x\le 50\\ \left[1+\left(\frac{x-50}{5}\right)^2\right]^{-1}, & для \quad 50\le x\le 100\end{cases}$$

тогда в соответствии с (7) находим

$$\mu_{очень\_старый}(x)=\begin{cases}0, & для \quad 0\le x\le 50\\ \left[1+\left(\frac{x-50}{5}\right)^{-2}\right]^{-2}, & для \quad 50\le x\le 100\end{cases}$$

$$\mu_{довольно\_молодой}(x)=\begin{cases}1, & для \quad 0\le x\le 25\\ \left[1+\left(\frac{x-25}{5}\right)^2\right]^{0.5}, & для \quad 25\le x\le 100\end{cases}$$

Например, если конкретному человеку 55 лет (т.е. x=55), то в соответствии с данными функциями принадлежности имеем:

$\mu_{не\ очень\ старый}(55)=0.75$

$\mu_{\ старый}(55)=0.5$

$\mu_{очень\ старый}(55)=0.25$

$\mu_{довольно\ молодой}(55)=0.164$

$\mu_{молодой}(55)=0.027$

4. Нечеткие алгоритмы

Понятие нечеткого алгоритма является важным инструментом для приближенного анализа сложных систем и процессов принятия решения. Под нечетким алгоритмом понимается упорядоченное множество нечетких инструкций (правил), в формулировке которых содержатся нечеткие указания (термы).

Например, нечеткие алгоритмы могут включать в себя инструкции типа:

А) x=очень малое

Б) x приблизительно равно 5

В) слегка увеличить

Г) если x – в интервале [4.9, 5.4]

Д) если x –малая, ТО y – большое, ИНАЧЕ – y – небольшое.

Использованные здесь термы «очень малое», «приблизительно равно», «слегка увеличить», «выбрать в интервале» и т.п. отражают неточность представления исходных данных и неопределенность, присущую самому процессу принятия решения.

Две последние инструкции (г и д) представляют собой (или нечеткие высказывания), построенные по схеме логической импликации ЕСЛИ – ТО , где условие ЕСЛИ соответствует принятию лингвистической переменной x некоторого значения A , а вывод (действие) ТО означает необходимость выбора значения B для лингвистической переменной y

$$(x = A) \rightarrow (y = B)$$

Указанные правила получили широкое распространение в технике. Механизм построения правил принятия решений в конкретной задаче выгладит при этом следующим образом. На основе заданной цели с помощью механизма упрощения, позволяющего выделить наиболее существенные и отсечь второстепенные факторы, определяются начальное состояние системы, желаемое конечное состояние и правила действия, приводящих систему в желаемое конечное состояние.

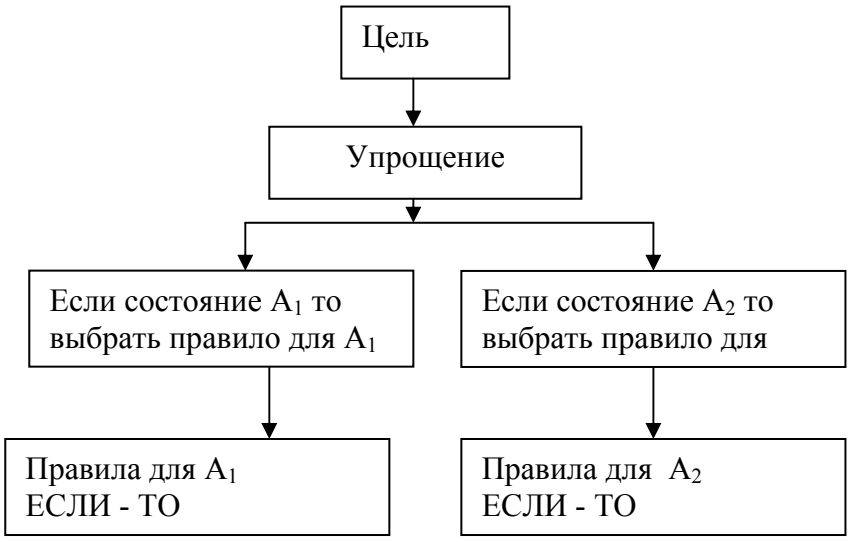


Рисунок 22 - Нечеткий алгоритм

Набор таких правил, обеспечивающих получение «хорошего», приближенного решения поставленной задачи, реализуется с помощью механизма вывода.

Рассмотрим особенности выполнения нечетких правил на следующем примере.

Допустим, что необходимо регулировать открытие охлаждающего вентиля $\varphi_{вых}$ в зависимости от измеренного значения температуры $T_{вх}$.

Воспользуемся для этих целей двумя правилами, записанными в лингвистической форме первое из которых имеет следующий вид:

Правило 1. ЕСЛИ температура = низкая, ТО охлаждающий вентиль полуоткрыт.

Будем полагать, что нечеткие подмножества A_1 (температура = низкая) и B_1 (вентиль=полуоткрыт) определяются функциями принадлежности, приведенными на рисунке 23.

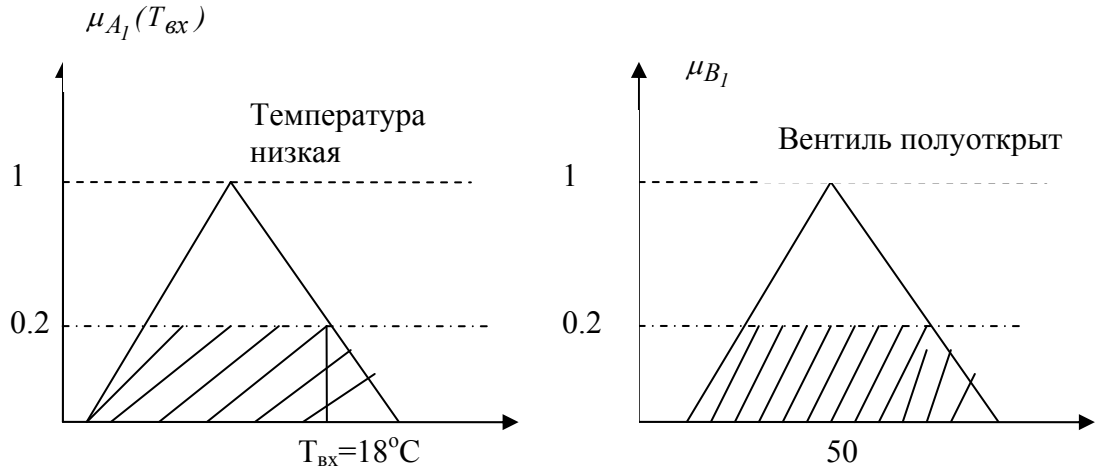


Рисунок 23 - Функции принадлежности

Если измеренное значение температуры $T_{вх}$ равна, например, 18°C, то степень принадлежности этого значения подмножеству A_1. Полагая, что меньшее значение степени выполнения условия ЕСЛИ должно сопровождаться уменьшением значения функции принадлежности вывода ТО ограничим возможные значения функции $\mu_{B_1}(\varphi_{вых})$ на уровне 0.2, т.е. получим

$$\mu_{B_1}(\varphi_{вых})|_{T_{вх}=18^oC} = \min\{0.2; \mu_{B_1}(\varphi_{вых})\}$$

Сформулируем второе лингвистическое правило следующим образом:

Правило 2. ЕСЛИ температура = средняя, ТО охлаждающий вентиль = почти открыт.

Функции принадлежности $\mu_{A_2}(T_{вх})$ и $\mu_{B_2}(\varphi_{вых})$, где A_2 и B_2 обозначают соответственно нечеткие подмножества, содержащиеся в условии и выводе правила 2, показаны на рисунке 24.

Степень принадлежности измеренного значения $T_{вх}$=18°С подмножеству A_2 здесь уже равна 0.5.

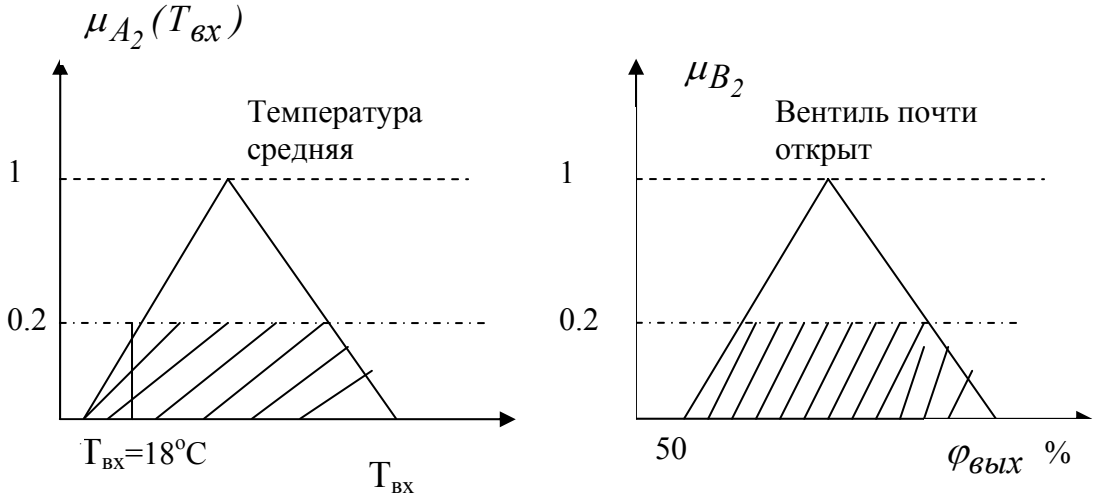


Рисунок 24 - Функции принадлежности

Следуя тому же приему, для функции принадлежности $\mu_{\hat{A}_2}(\varphi_{âûõ})$ получаем:

$$\mu_{B_2}(\varphi_{вых})|_{T_{вх}=18^oC} = \min\{0.5; \mu_{B_2}(\varphi_{вых})\}.$$

Заметим, что приведенные выше правила 1 и 2 действуют совместно и связаны друг с другом с помощью союза ИЛИ, т.е. можно записать.

Правило 1. ЕСЛИ температура = низкая, ТО охлаждающий вентиль =полуоткрыт

ИЛИ

Правило 2. ЕСЛИ температура = средняя, ТО охлаждающий вентиль = полуоткрыт.

Результирующая функция принадлежности $\mu_{B_2}(\varphi_{вых}) = \mu_{B_1 \cup B_2}(\varphi_{вых})$ находится по формуле

$$\mu(\varphi_{вых}) = max\left\{\mu_{B_1}(\varphi_{вых}) \mid \ T_{вх} = 18^o C \qquad \mu_{B_2}(\varphi_{вых}) \mid \ T_{вх} = 18^o C\right\}$$

График полученной функции принадлежности представлен на рисунке 25.

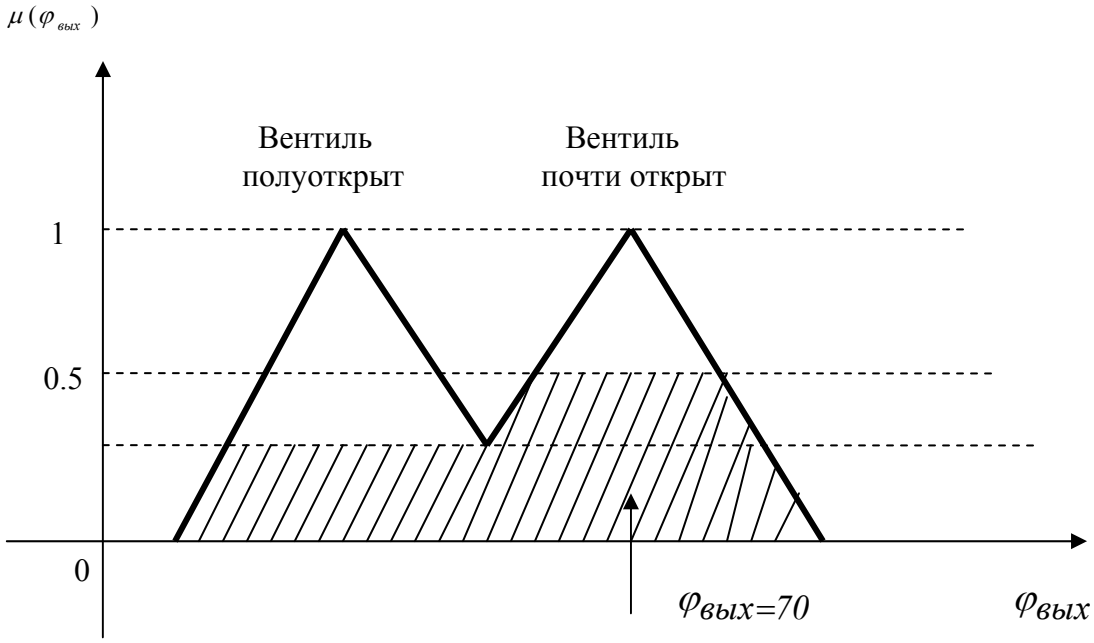


Рисунок 25 - График функции принадлежности

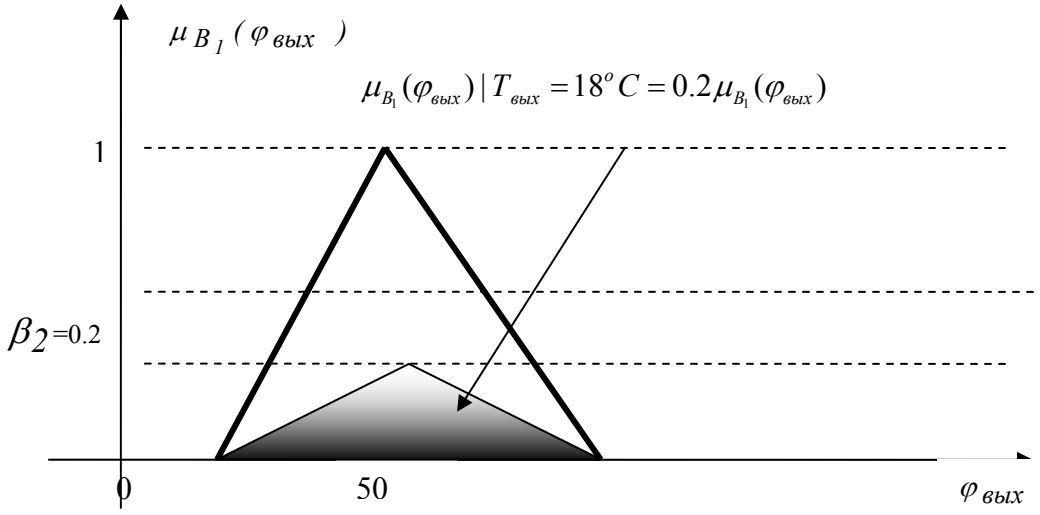
Описанный выше метод построения функции принадлежности носит название метода Максимума – Минимума.

На практике часто используется еще один метод построения функции принадлежности выходного нечеткого множества, получивший название метода Максимума – Произведения.

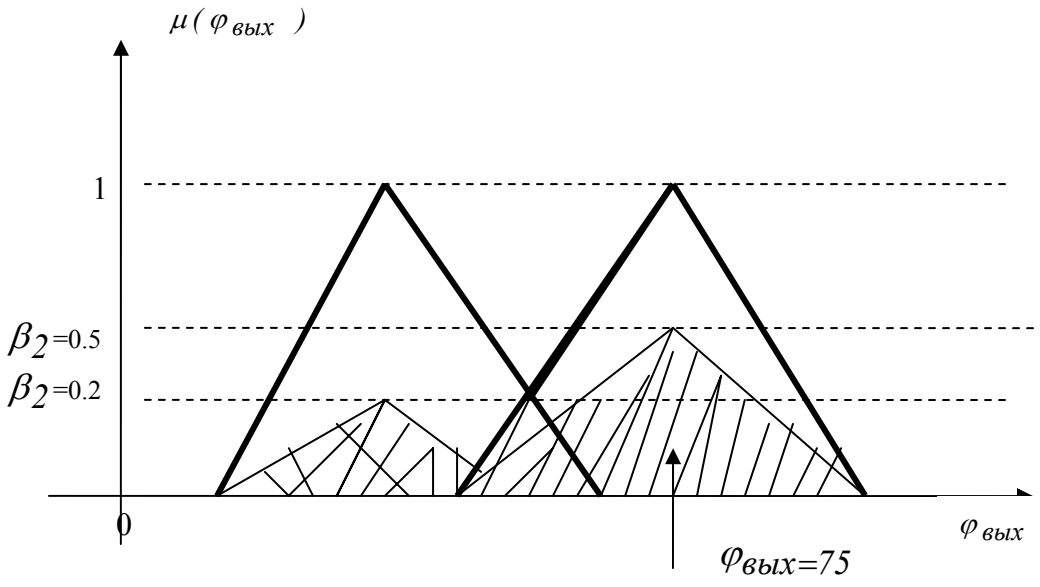
Суть данного метода заключается в следующем. При вычислении функции принадлежности вывода (заключения) «ТО» для каждого из правил осуществляется не ограничение их на уровне выполнения соответствующего условия «ЕСЛИ», а пропорциональное уменьшение их значений в соответствии с уровнем выполнения указанного условия (рисунок 26 а) с последующим использованием операций «ИЛИ» (рисунок 26 б).

Важно отметить, что при использовании любого из указанных выше методов вывода, результатом выполнения правил 1-2 является не конкретное

число $\varphi_{вых}$, а некоторое нечеткое множество, описываемое функцией принадлежности $\mu(\varphi_{вых})$. В то же время данное решение не может считаться окончательным, поскольку сохраняется неопределенность выбора искомой переменной $\varphi_{вых}$ внутри рассматриваемого интервала – носителя нечеткого множества B1 и B2.



а)



б)

Рисунок 26 - Функции принадлежности

Рассмотрим наиболее известные методы дефаззификации.

1. **Метод максимума** – выбирается тот элемент нечеткого множества, который имеет наивысшую степень принадлежности к этому множеству. Если такой элемент не является единственным, т.е. функция принадлежности имеет несколько локальных максимумов $y_1, y_2, ..., y_m$ со значениями $\mu_B(y_1) = \mu_B(y_2) = ... = \mu_B(y_m)$ или, если имеется максимальное «плато» между y_1 и y_m, то выбор среди элементов, имеющих наивысшую степень принадлежности множеству, осуществляется на основе определенного критерия.

2. **Метод левого (правого) максимума** – выбирается наименьшее (наибольшее) из чисел $y_1, y_2, ..., y_m$, имеющих наивысшую степень принадлежности нечеткому множеству.

3. **Метод среднего из максимумов** – в качестве искомого «четкого» значения y_o принимается среднее арифметическое значение координат локальных максимумов

$$y_o = \frac{1}{m} \sum_{j=1}^{m} y_j \qquad (13)$$

4. **Метод центра тяжести** – в качестве выходного значения y_o выбирается абсцисса центра тяжести площади, расположенной под функцией принадлежности $\mu_B(y)$, $y \in Y$:

$$y_o = \frac{\int_Y y \mu_B(y) dy}{\int_Y \mu_B(y) dy} \qquad (14)$$

Обычно при реализации этого метода на ЭВМ, используют численные методы интегрирования.

Существует простая возможность использования для этих целей взвешенного среднего значения

$$y_o = \frac{\sum_{i=1}^{n} \beta_i y_i^*}{\sum_{i=1}^{n} \beta_i} \qquad (15)$$

где y_i^* - центральные значения нечетких множеств $\beta_i(y)$ выходной переменной y;

β_i - веса, учитывающие уровень выполнения условия ЕСЛИ i-го правила, называемые также уровнями активности соответствующих правил; n – число правил выхода.

5. **Модифицированный метод центра тяжести**. Здесь интегрирование производится только в тех областях, где $\mu_B(y) > \alpha$, $\alpha \in (0,1)$, $y \in Y$. Параметр α используется здесь для подавления шумов, отсеивания влияния малосущественных для процедуры вывода факторов (на практике обычно $\alpha = 0.05 - 0.1$).

ТЕМА 6. МОДЕЛИ НЕЧЕТКОГО ВЫВОДА

1. Нечеткий логический вывод
2. Модель нечеткого вывода Мамдани
3. Модель нечеткого вывода Цукамото
4. Модель нечеткого вывода Сугено

1. Нечеткий логический вывод

Основой для проведения операции нечеткого логического вывода является база правил, содержащая нечеткие высказывания в форме «Если-то» и функции принадлежности для соответствующих лингвистических термов. При этом должны соблюдаться следующие условия:

1. Существует хотя бы одно правило для каждого лингвистического терма выходной переменной.

2. Для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

Пусть в базе правил имеется m правил вида:

R_1: ЕСЛИ x_1 это A_{11} … И … x_n это A_{1n}, ТО y это B_1

…

R_i: ЕСЛИ x_1 это A_{i1} … И … x_n это A_{in}, ТО y это B_i

…

R_m: ЕСЛИ x_1 это A_{m1} … И … x_n это A_{mn}, ТО y это B_m,

где x_k , k=1..n – входные переменные; y – выходная переменная; A_{ik} – заданные нечеткие множества с функциями принадлежности.

Результатом нечеткого вывода является четкое значение переменной y^* на основе заданных четких значений x_k , k=1..n.

В общем случае механизм логического вывода включает четыре этапа: введение нечеткости (фазификация), нечеткий вывод, композиция и приведение к четкости, или дефазификация.

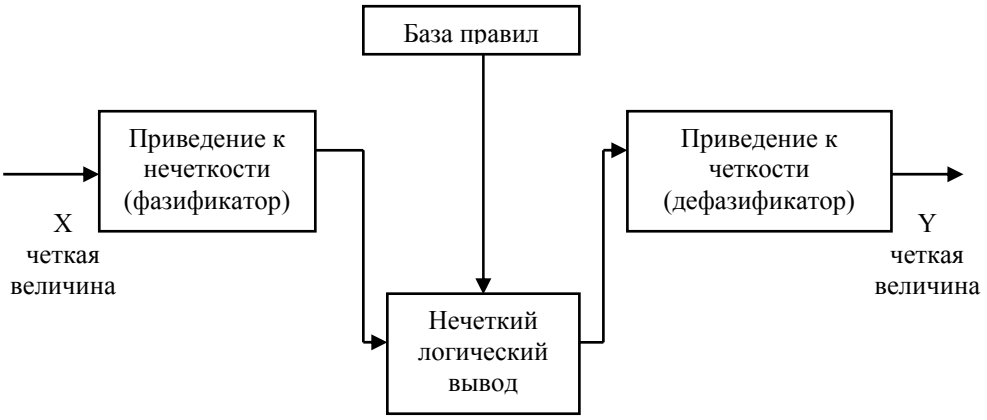


Рисунок 27 - Механизм логического вывода

Алгоритмы нечеткого вывода различаются главным образом видом используемых правил, логических операций и разновидностью метода дефазификации.

2. Модель нечеткого вывода Мамдани

Рассмотрим подробнее нечеткий вывод на примере механизма Мамдани (Mamdani). Это наиболее распространенный способ логического вывода в нечетких системах. В нем используется минимаксная композиция нечетких множеств. Данный механизм включает в себя следующую последовательность действий.

1. *Процедура фазификации.* Определяются степени истинности, т.е. значения функций принадлежности для левых частей каждого правила (предпосылок). Для базы правил с m правилами обозначим степени истинности как $A_{ik}(x_k)$, $i=1..m$, $k=1..n$.

2. *Нечеткий вывод*. Сначала определяются уровни «отсечения» для левой части каждого из правил $\alpha_i = \min(A_{ik}(x_k))$

Далее находятся «усеченные» функции принадлежности $B_i^*(y) = \min_i(\alpha_i, B_i(y))$

3. *Композиция*, или объединение полученных усеченных функций, для чего используется максимальная композиция нечетких множеств $MF(y) = \max_i(B_i^*(y))$, где MF(y) – функция принадлежности итогового нечеткого множества.

4. *Дефазификация*, или приведение к четкости.

Рисунок 28 графически показывает процесс нечеткого вывода по Мамдани для двух входных переменных и двух нечетких правил R1 и R2.

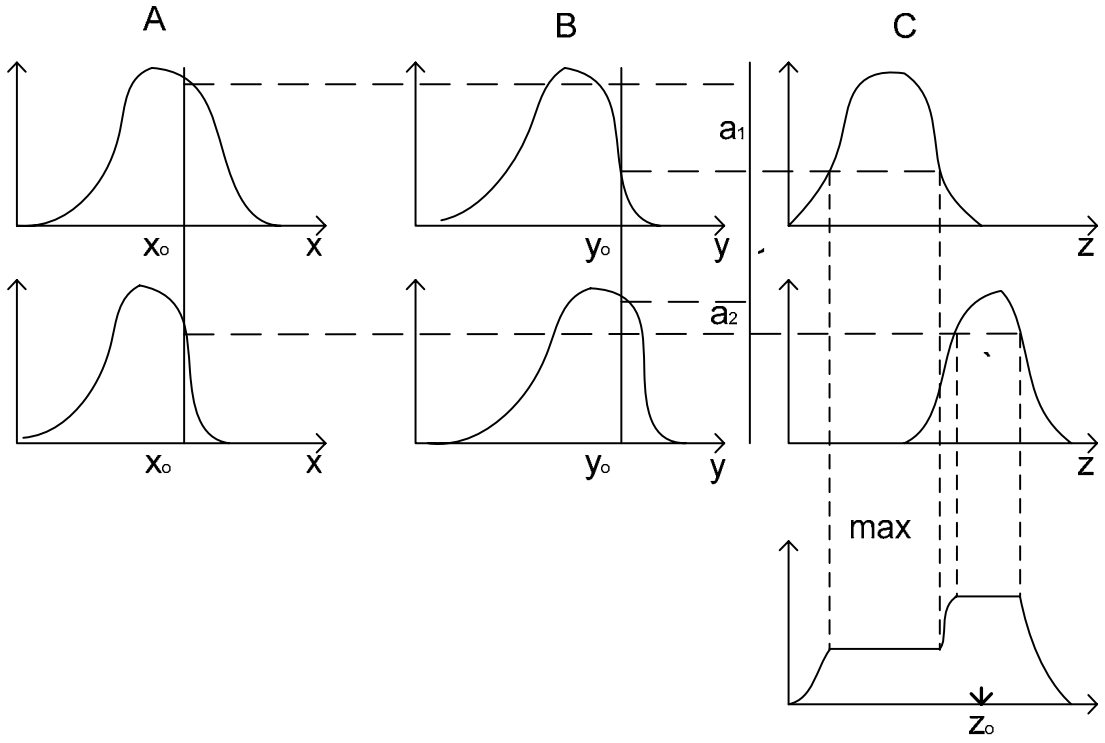


Рисунок 28 - Схема нечеткого вывода по Мамдани

Алгоритм Мамдани

Отметим вначале, что используемый в различного рода экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечетких предикатных правил вида:

$П_1$: если x есть A_1, тогда z есть B_1,

$П_2$: если x есть A_2, тогда z есть B_2,

. . . . . .

$П_n$: если x есть A_n, тогда z есть B_n,

где x - входная переменная (имя для известных значений данных), z - переменная вывода (имя для значения данных, которое будет вычислено); A_i и B_i - нечеткие множества, определенные соответственно на X и Z с помощью функций принадлежности $\mu_{A_1}(x)$ и $\mu_{B_1}(z)$.

Пример подобного правила: если x - низко, то z - высоко.

Механизм нечеткого вывода при аппроксимации функции $z(x)$ можно представить в следующем виде.

Предпосылка:

$П_1$: если x есть A_1, тогда z есть B_1,

$П_2$: если x есть A_2, тогда z есть B_2,

. . . . . .

$П_n$: если x есть A_n, тогда z есть B_n.

Факт: x есть A

Следствие: z есть B

Вывод в форме алгоритма Мамдани математически может быть описан следующим образом:

1. Введение нечеткости (fuzzification): для заданного (четкого) значения аргумента $x = x_0$ находятся степени истинности для предпосылок каждого правила $a_i = \mu_{A_i}(x_0)$

2. Нечеткий вывод по каждому правилу: находятся «усеченные» функции принадлежности для переменной вывода $\mu^*_{B_i}(z) = \min_z(a_i, \mu_{B_i}(z))$

3. Композиция: с использование операции МАКСИМУМ (max) производится объединение найденных усеченных функций, что приводит к получению итогового нечеткого подмножества для переменной вывода с функцией принадлежности $\mu_\Sigma(z) = \mu_B(z) = \max_z[\mu^*_{B_1}(z), \mu^*_{B_2}(z), ..., \mu^*_{B_3}(z)]$

4. Приведение к четкости (defuzzification) - для нахождения $z_0 = F(x_0)$ - обычно проводится центроидным методом: четкое значение выходной

переменной определяется как центр тяжести для кривой $\mu_\Sigma(z)$, т.е. $z_0 = \frac{\int_\Omega z \cdot \mu_\Sigma(z)dz}{\int_\Omega \mu_\Sigma(z)dz}$, где W - область определения $\mu_\Sigma(z)$.

3. Модель нечеткого вывода Цукамото

Алгоритм Цукамото использует те же исходные посылки, что и алгоритм Мамдани, однако предполагается, что функции являются монотонными.

Рассмотрим подробно данный алгоритм.

1) Введение нечеткости (как в алгоритме Мамдани).

2) Нечеткий вывод. Сначала находятся уровни «отсечения» (как в алгоритме Мамдани), а затем решения уравнений $\alpha_1 = C_1(z_1)$ и $\alpha_2 = C_2(z_2)$.

Определяются четкие значения (z_1 и z_2) для каждого исходного правила.

3) Определяется четкое значение переменной вывода (как взвешенное среднее z_1 и z_2)

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}$$

4. Модель нечеткого вывода Сугено

Рассмотрим алгоритм Сугено (0-го порядка). Исходный набор правил представляется в виде:

Π_i: если x есть A_i, тогда z есть z_i, i = 1,2,…,n, где $z_i = z(x_i)$.

Алгоритм состоит всего из двух этапов.

Первый этап идентичен первому этапу алгоритма Мамдани.

На втором этапе находится (четкое) значение переменной вывода:

$$z_0 = \frac{\sum_{i=1}^{n} z_i \cdot \alpha_i}{\sum_{i=1}^{n} \alpha_i}$$

ТЕМА 7. СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, ОСНОВАННЫЕ НА НЕЙРОННЫХ СЕТЯХ

1. Понятие нейронной сети
2. Структура нейронной сети
3. Классификация нейронных сетей
4. Применение нейронных сетей

1. Понятие нейронной сети

Искусственные нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессоров. Такие процессоры обычно исключительно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам, и, тем не менее, будучи соединенными в достаточно большую сеть с управляемым взаимодействием, таки локально простые процессоры вместе способны выполнять довольно сложные задачи.

Разработка искусственных нейронных сетей (НС) началась в начале XX века, но только в 90-х годах, когда вычислительные системы стали достаточно мощными, НС получили широкое распространение. Создание НС было вызвано попытками понять принципы работы человеческого мозга. Однако в сравнении с человеческим мозгом НС сегодня представляют собой весьма упрощенную модель. Несмотря на это, НС весьма успешно используются при решении самых различных задач. Например, программная реализация НС может использоваться для составления плана кредитных выплат людей, обращающихся в банк за займом. Хотя решение на основе нейронных сетей может выглядеть и вести себя как обычное программное обеспечение, они

различны в принципе, поскольку большинство реализаций на основе нейронных сетей «обучается», а «не программируется»: сеть учиться выполнять задачу, а не программируется непосредственно. На самом деле НС используются тогда, когда невозможно написать подходящую программу или найденной решение НС оказывается более совершенным. Например, эксперт по продаже недвижимости из своего опыта может знать, какие факторы влияют на продажную цену каждого конкретного дома, но при этом часто имеются такие особенности, которые будет весьма трудно объяснить программисту. Агентство по продаже недвижимости может пожелать иметь «предсказателя цен на основе НС», обученного на множестве примеров реальных продаж тому, какие факторы влияют на цену продаваемого объекта, и тому, какую относительную важность имеет каждый из этих факторов. Но здесь более важным оказывается то, что решение на основе НС является более гибким, поскольку соответствующая система может в дальнейшем совершенствовать точность предсказаний по мере накопления ею опыта и адаптироваться к происходящим на рынке изменениям.

НС представляет собой совокупность элементов, соединенных некоторым образом так, чтобы между ними обеспечивалось взаимодействие. Эти элементы, называемые также нейронами или узлами, представляют собой простые процессоры, вычислительные возможности которых обычно ограничиваются некоторым правилом комбинирования входных сигналов и правилом активизации, позволяющим вычислить выходной сигнал по совокупности входных сигналов. Выходной сигнал элемента может посылаться другим элементам по взвешенным связям, с каждой из которых связан весовой коэффициент или вес. В зависимости от значения весового коэффициента передаваемый сигнал или усиливается, или подавляется. Элемент нейронной сети, также называемый нейроном, схематически показан на рисунке 29.

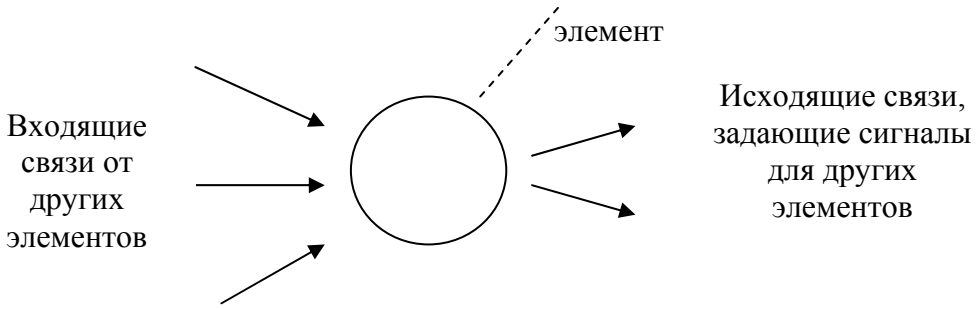


Рисунок 29 - Представление нейрона

Структура связей отражает детали конструкции сети, а именно то, какие элементы соединены и в каком направлении работают соединения, каков уровень значимости (т.е. вес) каждого соединения. Задача, которую понимает связь, описывается в терминах весовых значений связей, связывающих элементы. Структура связей обычно определяется в два этапа: сначала разработчик системы указывает, какие элементы должны быть связаны, и в каком направлении, а затем в процессе фазы обучения определяются значения соответствующих весовых коэффициентов.

Весовые коэффициенты можно определить и без проведения обучения, но как раз самое большое преимущество НС заключается в их способности обучаться выполнению задачи на основе тех данных, которые сеть будет получать в процессе реальной работы. Для многих приложений обучение является не только средством программирования сети, когда нет достаточных знаний о способе решения задачи, позволяющих выполнить программирование в традиционной форме, но часто единственной целью обучения является проверка того, что сеть действительно сможет научиться решать поставленные перед ней задачи.

2. Структура нейронной сети

Все рассматриваемые НС можно представить с помощью следующих абстракций.

- Множество простых процессоров.
- Структура связей.

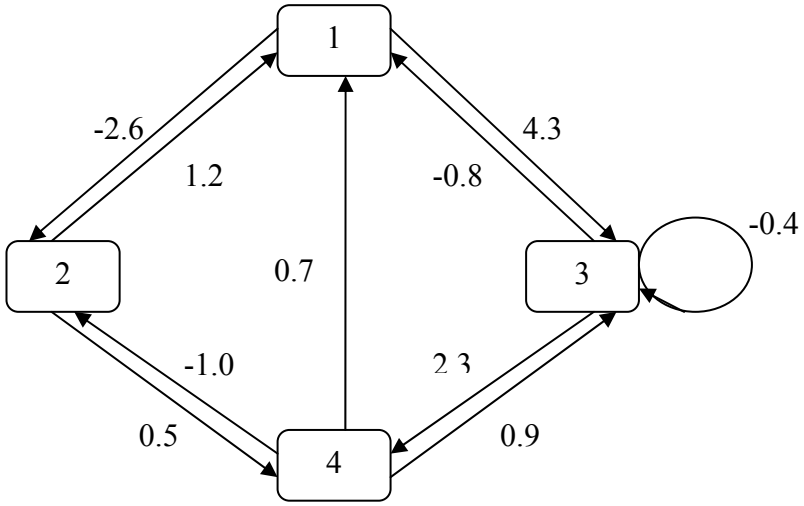
- Правило распространения сигналов в сети.
- Правило комбинирования входящих сигналов.
- Правило вычисления сигнала активности.
- Правило обучения, корректирующее связи.

Множество простых процессоров

С каждым процессором, т.е. обрабатывающим элементом сети, связывается набор входящих связей, по которым к данному элементу поступают сигналы от других элементов сети, и набор исходящих связей по которым сигналы данного элемента передаются другим элементам. Некоторые элементы предназначены для получения сигналов из внешней среды (и поэтому называются входными элементами), а некоторые – для вывода во внешнюю среду результатов вычислений (выходные элементы).

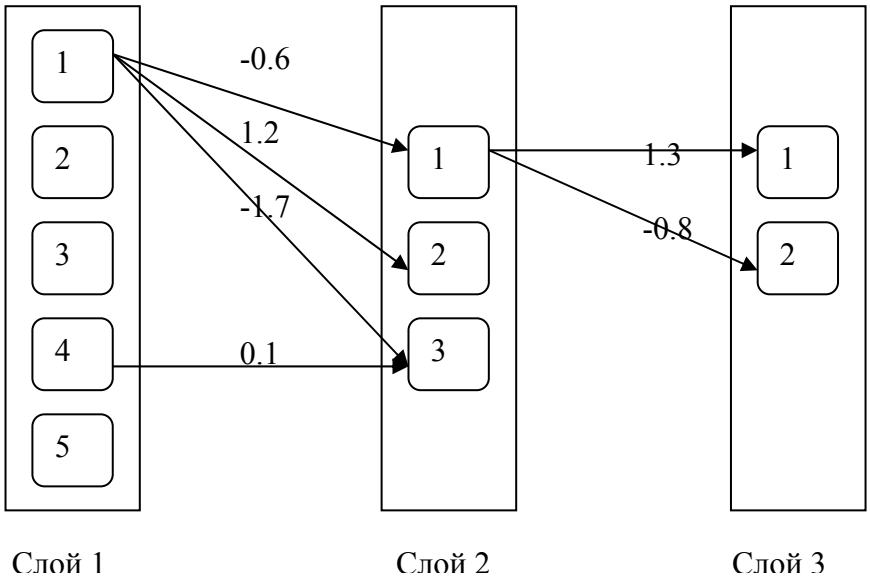
Структура связей

Структура связей отражает то, как соединены элементы сети. В одной модели (т.е. для одного типа сети) каждый элемент может быть связан со всеми другими элементами сети, в другой модели элементы могут быть организованы в некоторой упорядоченной по уровням (слоям) иерархии, где связи допускаются только между элементами в смежных слоях, а в третьей – могут допускаться обратные связи между смежными слоями или внутри одного слоя, или же допускаться посылка сигналов элементами самим себе. Возможности здесь бесконечны, но обычно для каждой конкретной модели сети указывается тип допустимых связей. Каждая связь определяется тремя параметрами: элементами, от которого исходит данная связь, элементом, к которому данная связь направлена, и числом, указывающим весовой коэффициент. Отрицательное значение веса соответствует подавлению активности соответствующего элемента, а положительной значение – усилению его активности. Абсолютное значение весового коэффициента характеризует силу связи.



$$w = \begin{bmatrix} 0.0 & -2.6 & 4.3 & 0.0 \\ 1.2 & 0.0 & 0.0 & 0.5 \\ -0.8 & 0.0 & -0.4 & 0.9 \\ 0.7 & -1.0 & 2.3 & 0.0 \end{bmatrix}$$

Рисунок 30 - Представления структуры связей в виде матриц для однослойной НС



$$W_1 = \begin{bmatrix} -0.6 & 1.2 & -1.7 \\ . & . & . \\ . & . & . \\ . & . & 0.1 \\ . & . & . \end{bmatrix} \quad W_2 = \begin{bmatrix} 1.3 & -0.8 \\ . & . \\ . & . \end{bmatrix}$$

Рисунок 31 - Представления структуры связей в виде матриц для многослойной сети

Структура связей обычно представляется в виде весовой матрицы W, в которой каждый элемент w_{ij} представляет величину весового коэффициента для связи, идущей от элемента i к элементу j. Для описания структуры связей может использоваться не одна, а несколько весовых матриц, если элементы сети оказываются не одна, а несколько весовых матриц, если элементы сети оказываются сгруппированными в слои. На рисунках 30 и 31 предлагаются примеры представления структуры связей в виде соответствующих матриц.

Правило распространения сигналов в сети

Довольно часто входящие сигналы элемента предполагается комбинировать путем суммирования их взвешенных значений. Пример этого метода суммирования показан на рисунке 32, где net_j обозначает результат комбинирования ввода элемента j, x_i - выход элемента i, а n – число задействованных связей. Используются и другие формы комбинирования входящих сигналов, и другим часто встречающимся методом является рассмотрение квадрата разности между значением силы связи и значением передаваемого по связи сигнала с последующим суммированием таких разностей для всех входящих связей данного элемента $net_j = \sum_{i=1}^{n} x_i w_i$

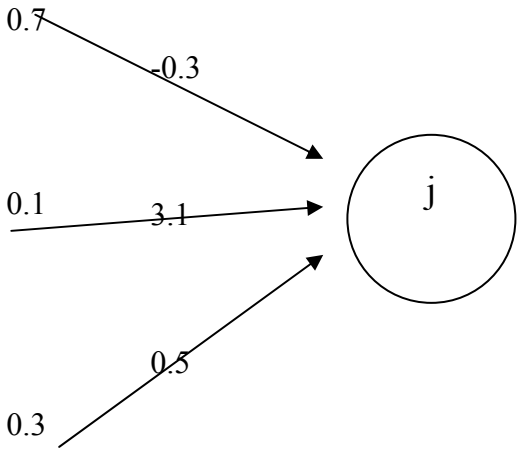


Рисунок 32 - Метод суммирования

$$net_j = (0.7 \times -0.3) + (0.1 \times 3.1) + (0.3 \times 0.5) = 0.25$$

Также можно представить и в векторном представлении:

$$\begin{bmatrix} 0.7 & 0.1 & 0.3 \end{bmatrix} \begin{bmatrix} -0,3 \\ 3.1 \\ 0.5 \end{bmatrix}$$

Правило вычисления сигнала активности

Для всех элементов имеется правило вычисления выходного значения, которое предполагается передать другим элементам или во внешнюю среду. Это правило называют правилом активности, а соответствующее выходное значение называют активностью соответствующего элемента. Активность может представляться либо некоторым действительным значением произвольного вида, либо действительным значением из некоторого ограниченного интервала значений (например, из интервала [0,1]), или же некоторым значением из определенного дискретного набора значений (например, $\{0,1\}$ или $\{+1,-1\}$). На вход функции активности поступает значение комбинированного ввода данного элемента. Примеры функций активности приводятся ниже.

Тождественная функция

Функция активности для входных элементов может быть тождественной функцией, и это просто означает, что значение (сигнал, посылаемый другим элементам) оказывается в точности равным комбинированному вводу (рисунок 33). Входные элементы обычно предназначены для распределения вводимых сигналов между другими элементами сети, поэтому для входных элементов обычно требуется, чтобы исходящий от элемента сигнал был таким же, как и входной. В отличие от других элементов сети, входные элементы сети только по одному входному значению. Например, каждый входной элемент может получать сигнал от одного соответствующего ему датчика, размещенного на фюзеляже самолета. Один этот элемент связывается со многими другими элементами сети, так что данные, полученные от одного датчика, оказываются распределенными между многими элементами сети.

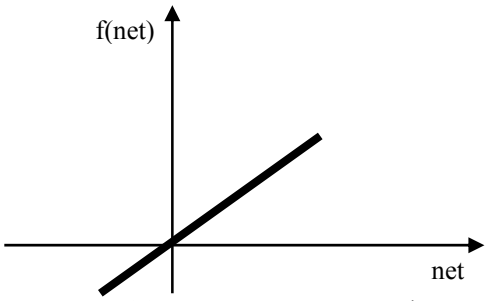


Рисунок 33 - Тождественная функция

Пороговая функция

В большинстве моделей НС используются нелинейные функции активности. Пороговая функция ограничивает активность значениями 1 или 0 в зависимости от значения комбинированного ввода в сравнении с некоторой пороговой величиной θ (рисунок 34).

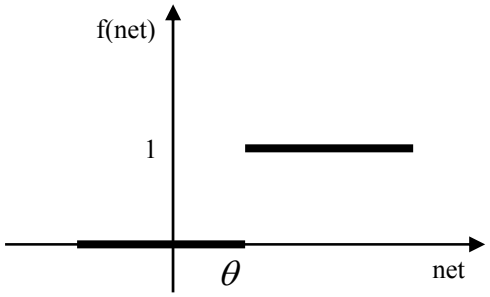


Рисунок 34 - Пороговая функция

$$f(net) = \begin{cases} 1, & если \quad net_i \geq \theta \\ 0, & если \quad net_i \leq \theta \end{cases}$$

Чаще всего удобнее вычесть пороговое значение (называемое смещением или сдвигом) из значения комбинированного ввода и рассмотреть пороговую функцию в ее математически эквивалентной форме, показанной на рисунке 35.

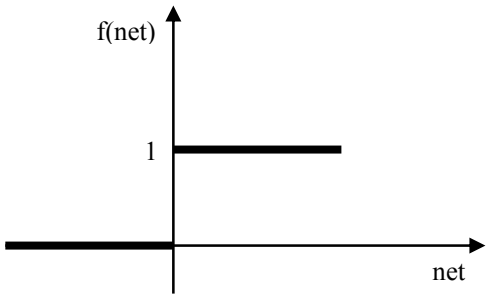


Рисунок 35 - Пороговая функция

Сдвиг w_o в данном случае оказывается отрицательным, а значение комбинированного ввода вычисляется по формуле $net_j = w_o + \sum_{i=1}^{n} x_i w_i$

$$f(net) = \begin{cases} 1, & если \quad net_i \geq 0 \\ 0, & если \quad net_i \leq 0 \end{cases}$$

Сдвиг обычно интерпретируется как связь. Исходящая от элемента активность которого всегда равна 1. комбинированный ввод в этом случае можно представить в виде $net_j = \sum_{i=0}^{n} x_i w_i$, где x_0 всегда считается равным 1.

Сигмоидальная функция

Наиболее часто используемой функцией активности является сигмоидальная функция $f(net) = \frac{1}{1 + exp(-net)}$.

Выходные значения такой функции непрерывно заполняют диапазон от 0 до 1. примером может служить логистическая функция, показанная на рисунке 36.

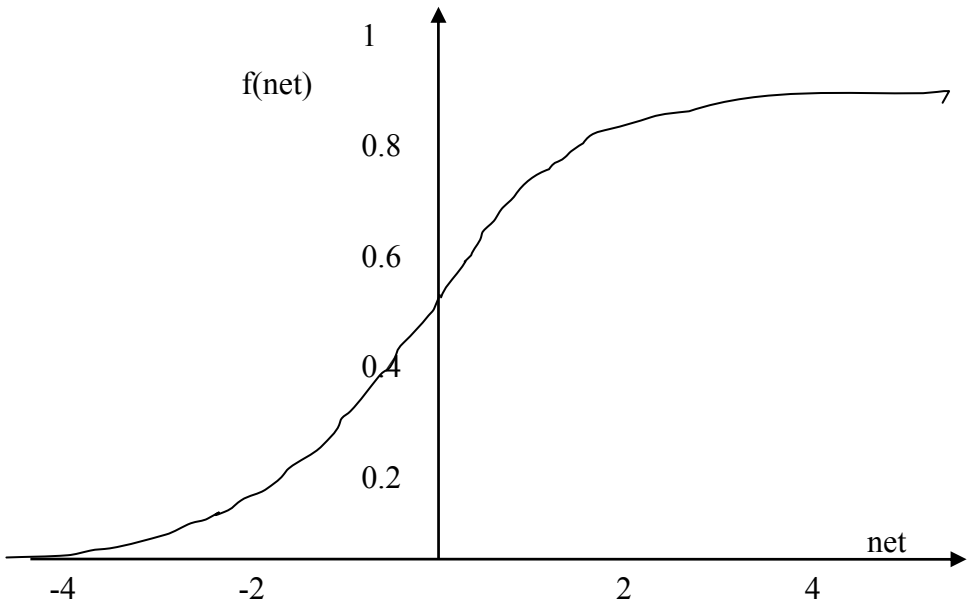


Рисунок 36 - Сигмоидальная функция

Наклон и область выходных значений логистической функции могут быть разными. Например, для биполярного сигмоида областью выходных значений является диапазон -1 и 1.

3. Классификация нейронных сетей

Нейронные сети обычно различают по трем признакам:

- структуре сети (типу связей между нейронами);
- особенностям моделей нейронов;
- особенностям обучения сети.

По структуре нейронные сети можно разделить на

- неполносвязные;
- полносвязные;
- со случайными связями;
- с регулярными связями;
- с симметричными связями;
- с несимметричными связями.

Неполносвязные нейронные сети обычно описываются неполносвязным ориентированным графом. Их подразделяют на однослойные (простейшие перцептроны) и многослойные.

Многослойные нейронные сети разделяют на сети с прямыми, перекрестными и обратными связями. В нейронных сетях с прямыми связями нейроны какого-либо слоя по входам могут соединяться только с нейронами предыдущих слоев. В нейронных сетях с перекресными связями допускаются связи внутри одного слоя. В нейронных сетях с обратными связями ограничений на связи нет.По используемым на входах и выходах сигналам нейронные сети можно разделить на:

- аналоговые;
- бинарные.

По моделированию времени нейронные сети подразделяются на

- сети с непрерывным временем;
- сети с дискретным временем.

Для программной реализации применяется как правило дискретное время.

По способу подачи информации на входы нейронной сети различают:

- подачу сигналов на синапсы входных нейронов;
- подачу сигналов на выходы входных нейронов;
- подачу сигналов в виде весов синапсов входных нейронов;
- аддитивную подачу на синапсы входных нейронов.

По способу съема информации с выходов нейронной сети различают:

- съем с выходов выходных нейронов;
- съем с синапсов выходных нейронов;
- съем в виде значений весов синапсов выходных нейронов;
- аддитивный съем с синапсов выходных нейронов.

По организации обучения разделяют обучение нейронных сетей с учителем и без учителя.

При обучении с учителем предполагается, что есть внешняя среда, которая предоставляет обучающие примеры (значения входов и соответствующие им значения выходов) на этапе обучения или оценивает правильность функционирования нейронной сети и в соответствии со своими критериями меняет состояние нейронной сети или поощряет (наказывает) нейронную сеть, запуская тем самым механизм изменения ее состояния.

Под состоянием нейронной сети, которое может изменяться обычно понимается:

- веса синапсов нейронов;
- веса синапсов и пороги нейронов (обычно в этом случае порог является более легко изменяемым параметром, чем веса синапсов);

- установление новых связей между нейронами (свойство биологических нейронов устанавливать новые связи и ликвидировать старые называется пластичностью).

По способу обучения разделяют обучение по входам и по выходам.

При обучении по входам обучающий пример представляет собой только вектор входных сигналов, а при обучении по выходам в него входит и вектор выходных сигналов, соответствующий входному вектору.По способу предъявления примеров различают предъявление одиночных примеров и множеств примеров.

В первом случае изменение состояния нейронной сети (обучение) происходит после предъявления каждого примера. Во втором - после предъявления множества примеров на основе анализа сразу их всех.

4. Применение нейронных сетей

Распознавание образов и классификация

В качестве образов могут выступать различные по своей природе объекты: символы текста, изображения, образцы звуков и т. д. При обучении сети предлагаются различные образцы образов с указанием того, к какому классу они относятся. Образец, как правило, представляется как вектор значений признаков. При этом совокупность всех признаков должна однозначно определять класс, к которому относится образец. В случае, если признаков недостаточно, сеть может соотнести один и тот же образец с несколькими классами, что неверно. По окончании обучения сети ей можно предъявлять неизвестные ранее образы и получать ответ о принадлежности к определённому классу.

Топология такой сети характеризуется тем, что количество нейронов в выходном слое, как правило, равно количеству определяемых классов. При этом устанавливается соответствие между выходом нейронной сети и классом, который он представляет. Когда сети предъявляется некий образ, на одном из

её выходов должен появиться признак того, что образ принадлежит этому классу. В то же время на других выходах должен быть признак того, что образ данному классу не принадлежит. Если на двух или более выходах есть признак принадлежности к классу, считается, что сеть «не уверена» в своём ответе.

Принятие решений и управление

Эта задача близка к задаче классификации. Классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появиться признак решения, которое она приняла. При этом в качестве входных сигналов используются различные критерии описания состояния управляемой системы.

Кластеризация

Под кластеризацией понимается разбиение множества входных сигналов на классы, при том, что ни количество, ни признаки классов заранее не известны. После обучения такая сеть способна определять, к какому классу относится входной сигнал. Сеть также может сигнализировать о том, что входной сигнал не относится ни к одному из выделенных классов — это является признаком новых, отсутствующих в обучающей выборке, данных. Таким образом, подобная сеть может выявлять новые, неизвестные ранее классы сигналов. Соответствие между классами, выделенными сетью, и классами, существующими в предметной области, устанавливается человеком. Кластеризацию осуществляют, например, нейронные сети Кохонена.

Прогнозирование

Способности нейронной сети к прогнозированию напрямую следуют из ее способности к обобщению и выделению скрытых зависимостей между входными и выходными данными. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений или каких-то существующих в настоящий момент факторов. Следует отметить, что прогнозирование возможно только тогда, когда предыдущие изменения действительно в какой-то степени

предопределяют будущие. Например, прогнозирование котировок акций на основе котировок за прошлую неделю может оказаться успешным, тогда как прогнозирование результатов завтрашней лотереи на основе данных за последние 50 лет почти наверняка не даст никаких результатов.

Аппроксимация

Нейронные сети — могут аппроксимировать непрерывные функции. Доказана обобщённая аппроксимационная теорема: с помощью линейных операций и каскадного соединения можно из произвольного нелинейного элемента получить устройство, вычисляющее любую непрерывную функцию с некоторой наперёд заданной точностью. Это означает, что нелинейная характеристика нейрона может быть произвольной: от сигмоидальной до произвольного волнового пакета или вейвлета, синуса или многочлена. От выбора нелинейной функции может зависеть сложность конкретной сети, но с любой нелинейностью сеть остаётся универсальным аппроксиматором и при правильном выборе структуры может достаточно точно аппроксимировать функционирование любого непрерывного автомата.

Сжатие данных и ассоциативная память

Способность нейросетей к выявлению взаимосвязей между различными параметрами дает возможность выразить данные большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Обратный процесс — восстановление исходного набора данных из части информации — называетсяассоциативной памятью. Ассоциативная память позволяет также восстанавливать исходный сигнал или образ из зашумленных или поврежденных входных данных. Решение задачи гетероассоциативной памяти позволяет реализовать память, адресуемую по содержимому.

ТЕМА 8. ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

1.Постановка задачи обучения нейронной сети

2. Правило обучения Розенблатта

3. Правило обучения Видроу-Хоффа

4. Многослойные нейронные сети

5. Алгоритм обратного распространения ошибки

1. Постановка задачи обучения нейронной сети

Под обучением понимается целенаправленное изменение (подстройка) весовых коэффициентов синаптических связей НС из условий достижения требуемых характеристик сети, т.е. желаемая реакция на входные воздействия.

Базовый принцип обучения – это минимизация эмпирической ошибки между желаемым выходом сети и фактической реакцией сети.

Процедура обучения

1. Создание обучающих выборок (задачник)

$\Omega = \{<x_1, D_1>, < x_2, D_2>, \ldots, < x_R, D_R>\}$, где

x – вектор входов;

D – вектор эталонов (желаемых выходов).

2. Осуществление выбора очередной пары из задачника (x_i, D_i).

3. Подача на вход нейросети x_i.

4. Вычисление разницы между желаемым и фактическим выходом сети

$$\varepsilon = D_i - Y_i^{факт}$$

5. Корректировка $\Delta\omega_{ij}^{k} = f(\varepsilon)$ для того, чтобы минимизировать ε.

6. Шаги 2 – 5 повторяются многократно до тех пор, пока ошибка ε не станет равна некоторой допустимой величине ошибки. В этом случае сеть называется натренированной, и теперь обученная сеть имеет возможность дать правильный ответ при подаче на вход нового вектора x, которого не было ранее в обучающей выборке. Эта способность соответствует свойству обобщения НС.

Качество работы НС сильно зависти от предъявляемого ей в процессе набора учебных данных. Учебные данные должны быть типичными для задачи, решению которой обучается сеть. Обучение часто оказывается уникальным процессом, когда приемлемые решения многих проблем могут быть получены только в процессе многочисленных экспериментов.

2. Персептрон Розенблатта

Одной из первых искусственных сетей, способных к восприятию и формированию реакции на воспринятый стимул, явился PERCEPTRON Розенблатта (F.Rosenblatt, 1957). Персептрон рассматривался не как конкретное техническое вычислительное устройство, а как модель работы мозга.

Простейший классический персептрон содержит нейроподобные элементы трех типов (рисунок 37). S-элементы формируют сетчатку сенсорных клеток, принимающих двоичные сигналы от внешнего мира. Далее сигналы поступают в слой ассоциативных или А-элементов. Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей. R-элементы с фиксированными весами формируют сигнал реакции персептрона на входной стимул. Розенблатт называл такую нейронную сеть трехслойной, однако по современной терминологии представленную сеть обычно называют *однослойной*, так как имеет только один слой нейропроцессорных элементов.

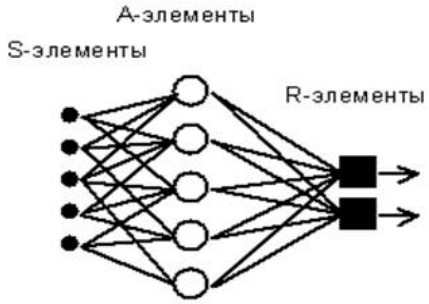


Рисунок 37 - Элементарный персептрон Розенблатта

Однослойный персептрон характеризуется *матрицей* синаптических связей W от S- к A-элементам. Элемент матрицы W_{ij} отвечает связи, ведущей от i-го S-элемента к j-му A-элементу.

Однослойный персептрон способен распознавать простейшие образы. Отдельный нейрон вычисляет взвешенную сумму элементов входного сигнала, вычитает значение смещения и пропускает результат через жесткую пороговую функцию, выход которой равен +1 или -1. В зависимости от значения выходного сигнала принимается решение: +1 - входной сигнал принадлежит классу A, -1 - входной сигнал принадлежит классу B.

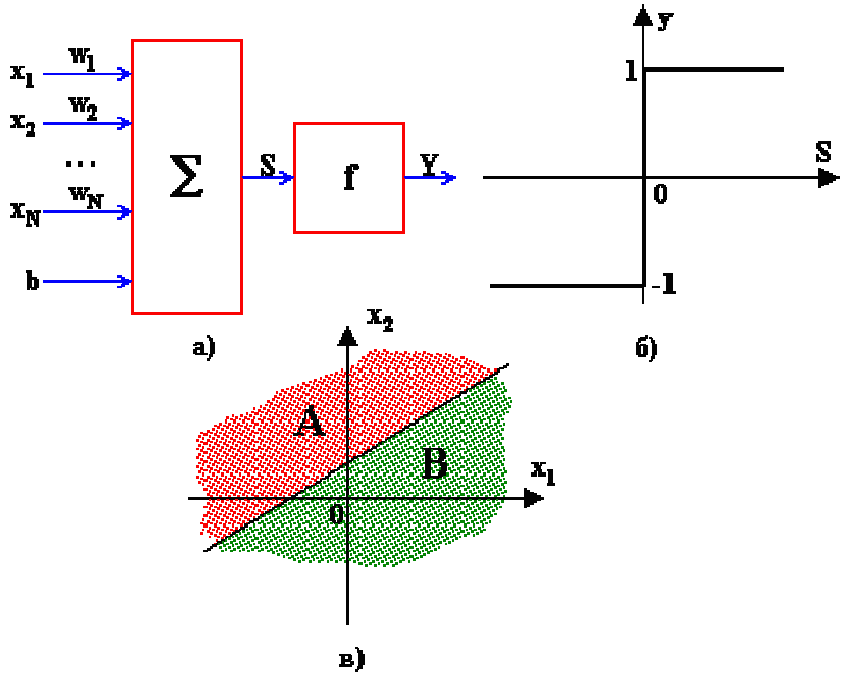


Рисунок 37 - а) Схема нейрона; б) График передаточной функции;
в) Разделяющая поверхность

На рисунке 38 показана схема нейронов, используемых в однослойных персептронах, график передаточной функции и схема решающих областей, созданных в многомерном пространстве входных сигналов. Решающие области определяют, какие входные образы будут отнесены к классу A, какие - к классу B. Персептрон, состоящий из одного нейрона, формирует две решающие области, разделенные гиперплоскостью. Здесь показан случай, когда размерность входного сигнала равна 2. При этом разделяющая поверхность

представляет собой прямую линию на плоскости. Входные сигналы над разделяющей линией относятся к классу А, под линией - к классу В. Уравнение, задающее разделяющую прямую, зависит от значений синаптических весов и смещения.

Рассмотрим процедуру настройки персептрона, предложенную Розенблаттом.

Алгоритм обучения однослойного персептрона

1. Инициализация синоптических весов и сдвига: синоптические весы принимают небольшие случайные значения.

2. Предъявление сети нового входного и желаемого выходного сигналов: входной сигнал $x=(x_1, x_2, ..., x_n)$ предъявляется нейрону вместе с желаемым выходным сигналом d.

3. Вычисление выходного сигнала нейрона $y(t) = f(\sum_{i=1}^{N} W_i(t)x_i(t) - b)$

4. Настраивание значений весов $w_i(t+1)=w_i(t)+r[d(t)-y(t)]x_i(t),\ i=1, ..., N$

$$d(t) = \begin{cases} +1, \text{если это класс } A \\ -1, \text{если это класс } B \end{cases},$$

где $w_i(t)$ - - вес связи от i-го элемента входного сигнала к нейрону в момент времени t, r - скорость обучения (меньше 1); $d(t)$ - желаемый выходной сигнал.

Если сеть принимает правильное решение, синоптические весы не модифицируются.

5. Переход к шагу 2.

Достоинства: программные или аппаратные реализации модели очень просты. Простой и быстрый алгоритм обучения.

Недостатки: примитивные разделяющие поверхности (гиперплоскости) дают возможность решать лишь самые простые задачи распознавания

3. Правило обучения Видроу – Хоффа

Персептрон Розенблатта ограничивается бинарными выходами. Видроу и Хофф изменили модель Розенблатта. Их первая модель имела один выходной нейрон и непрерывную линейную функцию активации нейронов $y = \sum_{j=1}^{n} w_j x_j$.

Метод обучения Видроу-Хоффа известен еще как *дельта-правило*. Этот метод ставит своей целью минимизацию функции ошибки E в пространстве весовых коэффициентов:

$E = \sum_{k=1}^{P} E(k) = \frac{1}{2} \sum_{k=1}^{P} (d^k - y^k)^2$, где

- P - количество обработанных и сетью примеров
- E(k) - ошибка для k-го примера
- y^k - реальный выход сети для k-го примера
- d^k - желаемый (идеальный) выход сети для k-го примера

Минимизация E осуществляется методом градиентного спуска:

$$w_j(t+1) = w_j(t) - a * \frac{\partial E(k)}{\partial w_j(t)},$$

где $\frac{\partial E(k)}{\partial w_j(t)} = \frac{\partial E(k)}{\partial y^k} \cdot \frac{\partial y^k}{\partial w_j} = (y^k - d^k) \cdot x_j^k$

Таким образом, весовые коэффициенты изменяются по правилу $w_i(t+1) = w_i(t) - a * (y^k - d^k) * x_i^k$

4. Многослойные нейронные сети

Многослойный персептрон (MLP) - нейронная сеть прямого распространения сигнала (без обратных связей), в которой входной сигнал преобразуется в выходной, проходя последовательно через несколько *слоев*.

Первый из таких слоев называют входным, последний - выходным. Эти слои содержат так называемые вырожденные нейроны и иногда в количестве слоев не учитываются. Кроме входного и выходного слоев, в многослойном

персептроне есть один или несколько промежуточных слоев, которые называют скрытыми. В этой модели *персептрона* должен быть хотя бы один скрытый *слой*. Присутствие нескольких таких *слоев* оправдано лишь в случае использования нелинейных функций активации.

Пример двухслойного *персептрона* представлен на рисунке 39.

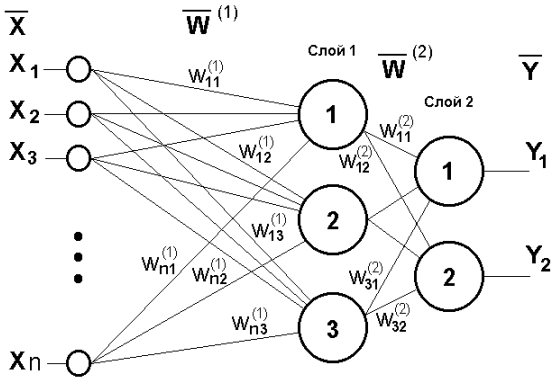
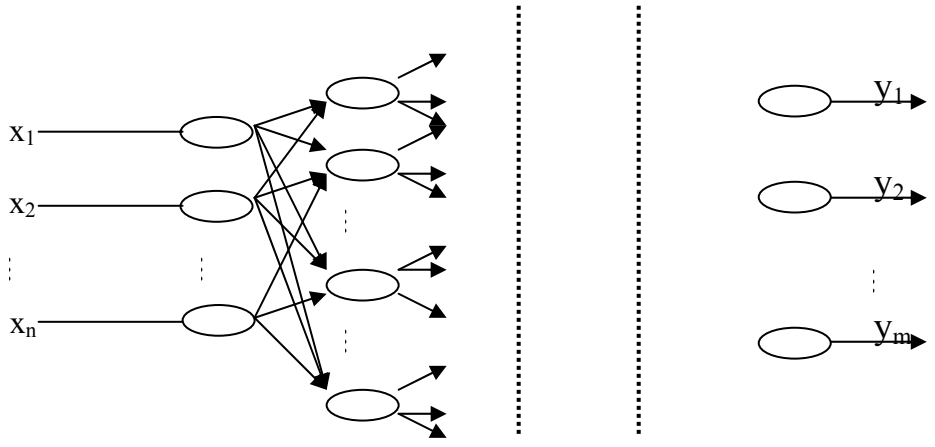


Рисунок 38 - Двухслойный персептрон

Сеть, изображенная на рисунке, имеет n входов. На них поступают сигналы, идущие далее по *синапсам* на 3 нейрона, которые образуют первый *слой*. Выходные сигналы первого *слоя* передаются двум нейронам второго *слоя*. Последние, в свою очередь, выдают два выходных сигнала.

В общем виде многослойный персептрон можно изобразить следующим образом:



входной слой n_1 | 1 скрытый слой n_2 | n_k | выходной слой n_N

Рисунок 39 - Многослойный персептрон

5. Алгоритм обратного распространения ошибки

Метод обратного распространения ошибки (Back propagation, backprop) - алгоритм обучения многослойных персептронов, основанный на вычислении градиента функции ошибок. В процессе обучения веса нейронов каждого слоя нейросети корректируются с учетом сигналов, поступивших с предыдущего слоя, и невязки каждого слоя, которая вычисляется рекурсивно в обратном направлении от последнего слоя к первому.

В общем случае задача обучения НС сводится к нахождению некой функциональной зависимости $Y=F(X)$ где X– входной, а Y– выходной векторы. В общем случае такая задача, при ограниченном наборе входных данных, имеет бесконечное множество решений. Для ограничения пространства поиска при обучении ставится задача минимизации целевой функции ошибки НС, которая находится по методу наименьших квадратов:

$$E(w) = \frac{1}{2} \sum_{j=1}^{p} (y_i - d_j)^2 \qquad (1)$$

где

y_j– значение j-го выхода НС;

d_j– целевое значение j-го выхода;

p– число нейронов в выходном слое.

Обучение НС производится методом градиентного спуска, т.е. на каждой итерации изменение веса производится по формуле:

$$\Delta w_{ij} = -h \cdot \frac{\partial E}{\partial w_{ij}} \qquad (2)$$

где h – параметр, определяющий скорость обучения.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \cdot \frac{\partial S_j}{\partial w_{ij}} \qquad (3)$$

где y_j – значение выхода j-го нейрона,

S_j – взвешенная сумма входных сигналов, определяемая по формуле $S = \sum_{i=1}^{n} x_i w_i$

При этом

$$\frac{\partial S_j}{\partial w_{ij}} = x_i \qquad (4)$$

где x_i – значение i-го входа нейрона

Далее рассмотрим определение первого множителя формулы (3)

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot w_{jk}^{(n+1)} \qquad (5)$$

где k – число нейронов в слое $n+1$.

Введем вспомогательную переменную

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \qquad (6)$$

Тогда можно определить рекурсивную формулу для определения $\delta_j^{(n)}$ n-ного слоя, если нам известно $\delta_j^{(n+1)}$ следующего $(n+1)$-го слоя.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{dS_j} \qquad (7)$$

Нахождение же $\delta_j^{(n)}$ для последнего слоя НС не представляет трудности, так как нам известен целевой вектор, т.е. вектор тех значений, которые должна выдавать НС при данном наборе входных значений:

$$\delta_j^{(N)} = (y_i^{(N)} - d_i) \cdot \frac{dy_i}{dS_i} \qquad (8)$$

Запишем формулу (6) в раскрытом виде:

$$\Delta w_{ij}^{(n)} = -h \cdot \delta_j^{(n)} \cdot x_i^n \qquad (9)$$

Рассмотрим теперь полный алгоритм обучения НС:

1. подать на вход НС один из требуемых образов и определить значения выходов нейронов НС;

2. рассчитать $\delta^{(N)}$ для выходного слоя НС по формуле (8) и рассчитать изменения весов $\Delta w_{ij}^{(N)}$ выходного слоя N по формуле (9);

3. рассчитать по формулам (7) и (9) соответственно $\delta^{(N)}$ и $\Delta w_{ij}^{(N)}$ для остальных слоев НС, $n = N-1,...,1$;

4. скорректировать все веса НС по формуле:

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \qquad (10)$$

5. если ошибка существенна, то перейти на шаг 1.

Достоинства метода обратного распространения

• Достаточно высокая эффективность.

• Через каждый нейрон проходит информация только о связных с ним нейронах. Поэтому алгоритм легко реализуется на вычислительных устройствах с параллельной архитектурой.

• Высокая степень общности. Алгоритм легко записать для произвольного числа слоёв, произвольной размерности выходов и выходов, произвольной функции потерь и произвольных функций активации, возможно, различных у разных нейронов. Кроме того, он не накладывает никаких ограничений на используемый метод оптимизации. Его можно применять вместе с методом скорейшего спуска, сопряженных градиентов, Ньютона-Рафсона и др.

Недостатки метода обратного распространения

• Метод не всегда сходится. Для улучшения сходимости приходится применять большое количество различных эвристических ухищрений.

• Процесс градиентного спуска склонен застревать в многочисленных локальных минимумах. Обратное распространение использует разновидность градиентного спуска, т. е. осуществляет спуск вниз по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму. Поверхность ошибки

сложной сети сильно изрезана и состоит из холмов, долин, складок и оврагов в пространстве высокой размерности. Сеть может попасть в локальный минимум (неглубокую долину), когда рядом имеется гораздо более глубокий минимум. В точке локального минимума все направления ведут вверх, и сеть неспособна из него выбраться. Статистические методы обучения могут помочь избежать этой ловушки, но они медленны. В предложен метод, объединяющий статистические методы машины Коши с градиентным спуском обратного распространения и приводящий к системе, которая находит глобальный минимум, сохраняя высокую скорость обратного распространения.

• Приходится заранее фиксировать число нейронов скрытого слоя. В то же время, это критичный параметр сложности сети, от которого может существенно зависеть качество обучения и скорость сходимости.

• При чрезмерном увеличении числа весов сеть склонна к переобучению.

• Если применяются функции активации с горизонтальными асимптотами, типа сигмоидной, то сеть может попадать в состояние «паралича». Если нейрон один раз попадает в такую «мёртвую зону», то у него практически не остаётся шансов из неё выбраться. Парализоваться могут отдельные связи, нейроны, или вся сеть в целом.

ТЕМА 9. РЕЛАКСАЦИОННЫЕ МОДЕЛИ НЕЙРОННЫХ СЕТЕЙ

1. Нейронная сеть Хопфилда
2. Нейронная сеть Хемминга
3. Самоорганизующиеся нейронные сети Кохонена

1. Нейронная сеть Хопфилда

Релаксационные сети - это сети, в которых распространение сигнала происходит как в прямом, так и в обратном направлении. На каждом такте работы НС один или несколько нейронных элементов обрабатывают информацию, полученную на предыдущем шаге, т.е. функционирование релаксационных НС носит итеративный характер. Такая обработка информации происходит до тех пор пока все нейронные элементы не перейдут в состояние равновесия. При этом состояния нейронных элементов перестаёт изменяться.

Нейронная сеть Хопфилда является однослойной НС, предназначенной для распознавания бинарных образов. Сеть охвачена обратной связью.

Сеть Хопфилда использует три слоя: входной, слой Хопфилда и выходной слой. Каждый слой имеет одинаковое количество нейронов. Входы слоя Хопфилда подсоединены к выходам соответствующих нейронов входного слоя через изменяющиеся веса соединений. Выходы слоя Хопфилда подсоединяются ко входам всех нейронов слоя Хопфилда, за исключением самого себя, а также к соответствующим элементам в выходном слое. В режиме функционирования, сеть направляет данные из входного слоя через фиксированные веса соединений к слою Хопфилда. Слой Хопфилда колебается, пока не будет завершено определенное количество циклов, и текущее состояние слоя передается на выходной слой. Это состояние отвечает образу, уже запрограммированному в сеть.

Обучение сети Хопфилда требует, чтобы обучающий образ был представлен на входном и выходном слоях одновременно. Рекурсивный

характер слоя Хопфилда обеспечивает средства коррекции всех весов соединений. Недвоичная реализация сети должна иметь пороговый механизм в передаточной функции. Для правильного обучение сети соответствующие пары «вход-выход» должны отличаться между собой.

Если сеть Хопфилда используется как память, адресуемая по смыслу она имеет два главных ограничения. Во-первых, число образов, которые могут быть сохранены и точно воспроизведены является строго ограниченным. Если сохраняется слишком много параметров, сеть может сходится к новому несуществующему образу, отличному от всех запрограммированных образов, или не сходится вообще.

Граница емкости памяти для сети приблизительно 15% от числа нейронов в слое Хопфилда. Вторым ограничением парадигмы есть то, что слой Хопфилда может стать нестабильным, если обучающие примеры являются слишком похожими. Образец образа считается нестабильным, если он применяется за нулевое время и сеть сходится к некоторому другому образу из обучающего множества. Эта проблема может быть решена выбором обучающих примеров более ортогональных между собой.

Структурная схема сети Хопфилда приведена на рисунке 41.

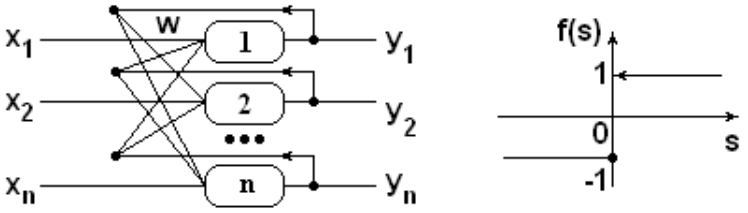


Рисунок 40 - Структурная схема сети Хопфилда

Задача, решаемая данной сетью в качестве ассоциативной памяти, как правило, формулируется так. Известен некоторый образцовый набор двоичных сигналов (изображений, звуковых оцифровок, других данных, которые описывают определенные объекты или характеристики процессов). Сеть должна уметь с зашумленого сигнала, представленного на ее вход, выделить («припомнить» по частичной информации) соответствующий образец или «дать

вывод» о том, что входные данные не отвечают ни одному из образцов. В общем случае, любой сигнал может быть описан вектором x_1, x_i, x_n..., n - число нейронов в сети и величина входных и выходных векторов. Каждый элемент x_i равняется или +1, или -1. Обозначим вектор, который описывает k-ий образец, через X_k, а его компоненты, соответственно, - x_{ik}, k=0, ..., m-1, m - число образцов. Если сеть распознает (или «вспоминает») определенный образец на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть $Y = X_k$, где Y - вектор выходных значений сети: y_1, y_i, y_n. В противном случае, выходной вектор не совпадет ни с одним образцом.

Если, например, сигналы представляют собой какое-то изображение, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, которая целиком совпадает с одной из образцовых (в случае успеха) или же «свободную импровизацию» сети (в случае неудачи).

Алгоритм функционирования сети

1. На стадии инициализации сети синаптические коэффициенты устанавливаются таким образом:

$$w_{ij} = \begin{cases} \sum_{k=1}^{m} x_i^k x_j^k, i \neq j \\ 0, \quad i = j \end{cases}$$

Здесь i и j - индексы, соответственно, предсинаптического и постсинаптического нейронов; x_i^k, x_j^k - i-ый и j-ый элементы вектора k-ого образца.

2. На входы сети подается неизвестный сигнал. Его распространение непосредственно устанавливает значения выходов $y_i(0) = x_i$, $i = 0...n-1$

Поэтому обозначения на схеме сети входных сигналов в явном виде носит чисто условный характер. Нуль в скобке y_i означает нулевую итерацию в цикле работы сети.

3. Рассчитывается новое состояние нейронов $S(t+1)=\sum_{i=1}^{n} w_{ij} y_j(t), j=0...n-1$ и новые значения выходов $y_i(t+1)=f\,|\,S_j(t+1)\,|$, где f - передаточная функция в виде пороговой, приведена на рисунке 42.

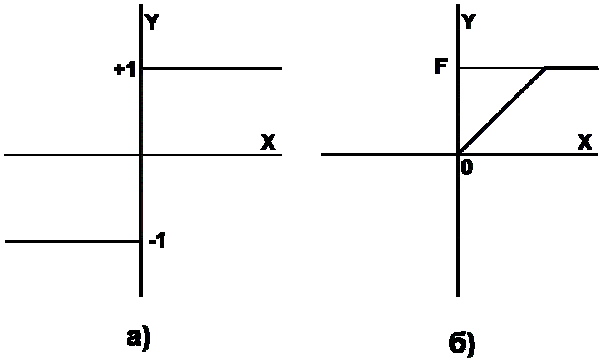


Рисунок 41 - Передаточные функции

4. Проверяем изменились ли выходные значения выходов за последнюю итерацию. Если да - переход к пункту 2, иначе (если выходы стабилизировались) - конец. При этом выходной вектор представляет собой образец, что лучше всего отвечает входным данным.

Число запоминаемых образов должно удовлетворять ограничению: $m \le n/7$. Кроме того, эталоны попарно должны иметь существенные корреляционные отличия. Появление на входе системы нового сигнала вызывает переходный процесс, который описывается векторными соотношениями: $Y(0)=X$, $s(p+1)=W\cdot Y(p)$, $Y(p+1)=f(s(p+1))$.

Возможными результатами переходного процесса могут являться:

- Одно из эталонных изображений $Y=X^{(k)}$.
- Негатив одного из эталонных изображений $Y=-X^{(k)}$.
- Зацикливания длины 2 некоторых битов выходного сигнала – отказ от распознавания.

Рассмотрим пример распознавания бинарных изображений с помощью НС Хопфилда. Даны два образца изображений:

$$Z1=\begin{pmatrix}-1 & 1 & 1 & -1\\ 1 & 1 & 1 & 1\\ 1 & -1 & -1 & 1\\ -1 & -1 & -1 & -1\end{pmatrix}\quad Z2=\begin{pmatrix}-1 & -1 & -1 & -1\\ 1 & -1 & -1 & 1\\ -1 & 1 & 1 & -1\\ 1 & 1 & 1 & 1\end{pmatrix}$$

и входное, распознаваемое изображение

$$Z3=\begin{pmatrix}1 & -1 & 1 & -1\\ 1 & 1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & -1\end{pmatrix}.$$

В этом случае m=2, n=16, т.е. выполняется неравенство $2 \le 16/7 \approx 2.29$, возможно использование НС Хопфилда.

Данные матрицы преобразуем по столбцам в вектора:

$$Z1=[-1\ \ 1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1]^T,$$

$$\mathbf{Z2}=[\mathbf{-1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1\ \ -1\ \ 1}]^{\text{Ò}},$$

$$\mathbf{Z3}=[\mathbf{1\ \ 1\ \ 1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ -1\ \ -1\ \ -1\ \ -1\ \ -1}]^{\text{Ò}}$$

Вычислим матрицу весовых коэффициентов сети:

$W2_{25}=$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	-2	0	0	0	0	0	0	0	0	0	0	2	-2	0	0
1	-2	0	0	0	0	0	0	0	0	0	0	0	-2	2	0	0
2	0	0	0	-2	2	2	-2	-2	2	2	-2	-2	0	0	2	-2
3	0	0	-2	0	-2	-2	2	2	-2	-2	2	2	0	0	-2	2
4	0	0	2	-2	0	2	-2	-2	2	2	-2	-2	0	0	2	-2
5	0	0	2	-2	2	0	-2	-2	2	2	-2	-2	0	0	2	-2
6	0	0	-2	2	-2	-2	0	2	-2	-2	2	2	0	0	-2	2
7	0	0	-2	2	-2	-2	2	0	-2	-2	2	2	0	0	-2	2
8	0	0	2	-2	2	2	-2	-2	0	2	-2	-2	0	0	2	-2
9	0	0	2	-2	2	2	-2	-2	2	0	-2	-2	0	0	2	-2
10	0	0	-2	2	-2	-2	2	2	-2	-2	0	2	0	0	-2	2
11	0	0	-2	2	-2	-2	2	2	-2	-2	2	0	0	0	-2	2
12	2	-2	0	0	0	0	0	0	0	0	0	0	0	-2	0	0
13	-2	2	0	0	0	0	0	0	0	0	0	0	-2	0	0	0
14	0	0	2	-2	2	2	-2	-2	2	2	-2	-2	0	0	0	-2
15	0	0	-2	2	-2	-2	2	2	-2	-2	2	2	0	0	-2	0

Формирование i-го элемента выходного сигнала происходит следующим образом: i-я строка матрицы скалярно умножается на вектор входного сигнала. Полученное значение подставляется в активационную функцию. Например, первый элемент выходного вектора равен:

$$Y1_1 = f(0\cdot 1+(-2)\cdot 1+0\cdot 1+0\cdot 1+0\cdot(-1)+0\cdot 1+0\cdot 1+0\cdot(-1)+0\cdot 1+0\cdot 1+0\cdot(-1)+0\cdot(-1)+2\cdot(-1)+$$
$$(-2)\cdot(-1)+0\cdot(-1)+0\cdot(-1))=\mathrm{f}(-2)=-1.$$

Выходной вектор НС имеет вид:

$$Y1 = [-1 \quad -1 \quad 1 \quad -1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad -1]^T$$

Второй и третий выходные сигналы рассчитываются аналогично, но в качестве входного сигнала используется предыдущий выходной сигнал.

$$Y2 = [1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad 1 \quad -1]^T,$$

$$\mathbf{Y3} = [\mathbf{-1} \quad \mathbf{-1} \quad \mathbf{1} \quad \mathbf{-1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{-1} \quad \mathbf{-1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{-1} \quad \mathbf{-1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{-1}]^{\mathrm{T}}.$$

Третий выходной сигнал в точности повторяет первый, это говорит о зацикливании нейронной сети.

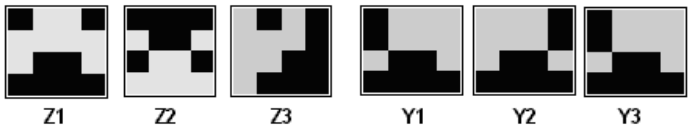


Рисунок 42 - Два образцовых изображения Z1, Z2, входной вектор Z3, выходные вектора Y1, Y2, Y3.

2. Нейронная сеть Хемминга

Сеть Хемминга (Hamming) - расширение сети Хопфилда. Сеть Хемминга реализует классификатор, базирующийся на наименьшей погрешности для векторов двоичных входов, где погрешность определяется расстоянием Хемминга. Расстояние Хемминга определяется как число бит, отличающихся между двумя соответствующими входными векторами фиксированной длины. Один входной вектор является незашумленым примером образа, другой - испорченным образом. Вектор выходов обучающего множества является вектором классов, к которым принадлежат образы. В режиме обучения входные векторы распределяются по категориям, для которых расстояние между образцовыми входными векторами и текущим входным вектором является минимальным.

Сеть Хемминга имеет три слоя: входной слой с количеством узлов, сколько имеется отдельных двоичных признаков; слой категорий (слой Хопфилда), с количеством узлов, сколько имеется категорий или классов; выходной слой, который отвечает числу узлов в слое категорий.

Сеть есть простой архитектурой прямого распространения с входным уровнем, полностью подсоединенным к слою категорий. Каждый элемент обработки в слое категорий является обратно подсоединенным к каждому нейрону в том же самом слое и прямо подсоединенным к выходному нейрону. Выход из слоя категорий к выходному слою формируется через конкуренцию.

Обучение сети Хемминга похоже на методологию Хопфилда. На входной слой поступает желаемый обучающий образ, а на выход выходного слоя поступает значение желаемого класса, к которому принадлежит вектор. Выход содержит лишь значение класса к которому принадлежит входной вектор. Рекурсивный характер слоя Хопфилда обеспечивает средства коррекции всех весов соединений.

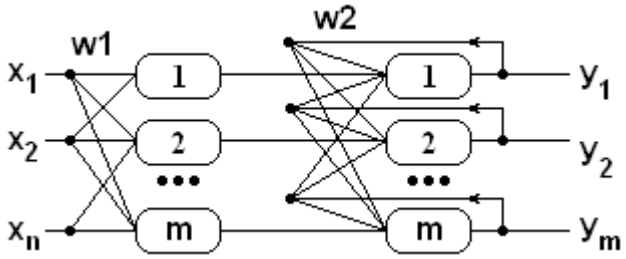


Рисунок 43 - Структурная схема сети Хемминга

Алгоритм функционирования сети Хемминга

1. На стадии инициализации весовым коэффициентам первой слоя и порогу передаточной функции присваиваются такие значения:

$$W_{ik}=x_{ik}/2,\ i=0...n-1,\ k=0...m-1$$

$$b_k = n / 2,\ k = 0...m-1$$

Здесь x_{ik} - i-ый элемент k-ого образца.

Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине $0 < v < 1/m$. Синапс нейрона, связанный с его же выходом имеет вес +1.

2. На входы сети подается неизвестный вектор $x_1,\ x_i,\ x_n$ ... Рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$y_j^{(1)} = S_j^{(1)} = \sum_{i=1}^{n} w_{ij} x_i, + b_j, \quad j = 0...m-1$$

После этого получения значения инициализируют значения выходов второго слоя:

$$y_j(2) = y_j(1), j = 0...m-1$$

3. Вычисляются новые состояния нейронов второго слоя:

$$S_j^{(2)}(t+1) = y_j(t) - \varepsilon \sum_{k=1}^{m} y_k^{(2)}(t), \ j = 1..m$$

И значения их выходов:

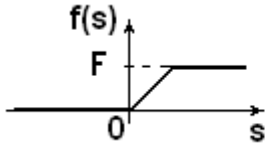
$$y_j^{(1)}(t+1) = f \mid S_j^{(2)}(t+1) \mid$$

Передаточная функция f имеет вид порога, причем величина b должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

4. Проверяется, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да - перейти к шагу 3. Иначе - конец.

Роль первой слоя является условной: воспользовавшись один раз на первом шаге значениями его весовых коэффициентов, сеть больше не возвращается к нему, поэтому первый слой может быть вообще исключен из сети.

Активационная функция первого слоя – линейная функция, второго слоя – положительная линейная ограниченная функция. Активационная функция и матрица второго слоя НС Хэмминга показаны на рисунке 45.



$$\mathbf{w}^{(2)} = \begin{pmatrix} 1 & -\varepsilon & -\varepsilon \\ -\varepsilon & 1 & -\varepsilon \\ -\varepsilon & -\varepsilon & 1 \end{pmatrix}$$

Рисунок 44 - Активационная функция и матрица весовых коэффициентов второго слоя

Матрица $W^{(2)}$ имеет отрицательные и положительные элементы. Такая матрица сохраняет наибольший сигнал и ослабляет до нуля меньшие сигналы.

От каждого сигнала, кроме максимального, вычитается $1/m$ часть других, пока он не станет нулевым (отрицательные значения исключены активационной функцией).

Сеть Хемминга имеет ряд преимуществ над сетью Хопфилда. Она способна найти минимальную погрешность, если погрешности входных бит являются случайными и независимыми. Для функционирования сети Хемминга нужно меньшее количество нейронов, поскольку средний слой требует лишь один нейрон на класс, вместо нейрона на каждый входной узел. И, в конце концов, сеть Хемминга не страдает от неправильных классификаций, которые могут случиться в сети Хопфилда. В целом, сеть Хемминга быстрее и точнее, чем сеть Хопфилда.

3. Самоорганизующиеся нейронные сети Кохонена

Характерной особенностью мозга является то, что его структура отражает организацию внешних раздражителей, которые в него поступают. Например, известно соответствие относительного положения рецепторов на поверхности кожи и нейронов в мозге, которые воспринимают и обрабатывают сигналы от этих рецепторов, а участки кожи, которые плотно «населены» рецепторами (лицо, руки), ассоциированы с пропорционально большим числом нейронов. Это соответствие образует то, что называется соматотропической картой, которая отражает поверхность кожи в часть мозга — соматосенсорную кору, которая и воспринимает ощущение касания.

Системы, построенные на базе, например, многослойных сетей с обратным распределением ошибок имеют очевидный недостаток, заключающийся в том, что мы должны заранее заготовить входы и запастись соответствующими правильными ответами для обучения сети. Принципы функционирования природной соматотропической карты легли в основу создания самоорганизующихся сетей (карт, решеток) Кохонена, для которых не требуется предварительное обучение на примерах. Сеть Кохонена

воспринимает только вход и способна вырабатывать свое собственное восприятие внешних стимулов.

Самоорганизующиеся сети Кохонена — это карты или многомерные решетки, с каждым узлом которой ассоциирован входной весовой вектор, то есть набор из *к* входных весов нейрона трактуется как вектор в пространстве. Входной весовой вектор имеет ту же размерность, что и вход в сеть. Обучение происходит в результате конкуренции, возникающей между узлами сети за право отклика на полученный входной сигнал. Элемент сети, который выигрывает в этой конкуренции (победитель), и его ближайшее окружение (свита) модифицируют веса своих входных связей. Перед обучением каждая компонента входного весового вектора инициализируется случайным образом. Обычно каждый вектор нормализуется в вектор с единичной длиной в пространстве весов. Это делается делением соответствующего веса на корень из суммы квадратов компонент этого весового вектора. Входные вектора нормализуются аналогично.

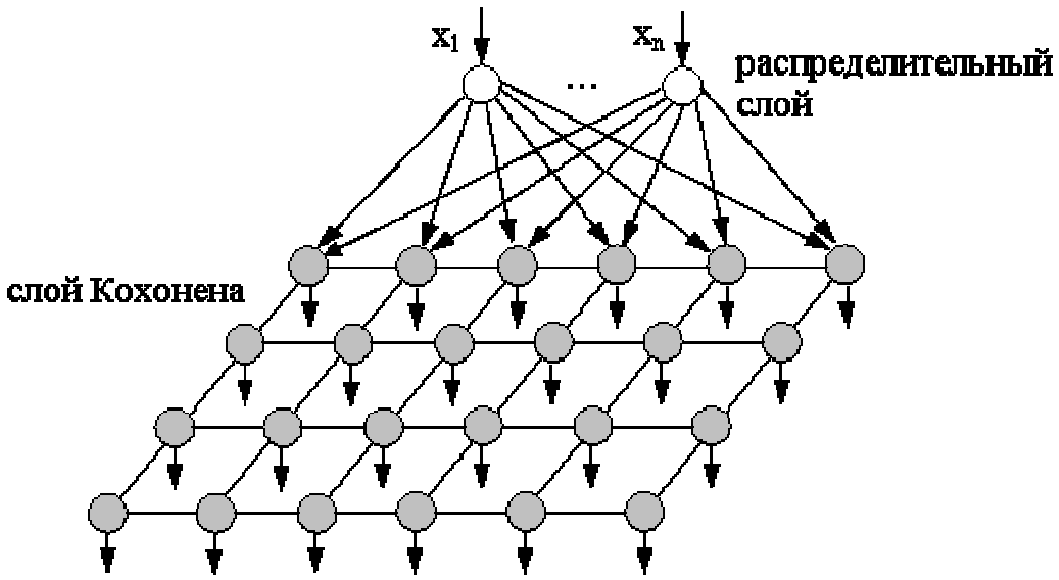


Рисунок 45 - Топология самоорганизующейся карты Кохонена

Алгоритм функционирования сети Кохонена

1. Инициализация сети. Весовым коэффициентам сети даются небольшие случайные значения. Общее число синаптических весов - M*N.

2. Предъявление сети нового входного сигнала.

3. Вычисление расстояния до всех нейронов сети.

Расстояния d_j от входного сигнала до каждого нейрона j определяются по формуле:

$$d_j = \sum_{i=1}^{N} (x_i(t) \cdot w_{ij}(t))^2 ,$$

где x_i - i-ый элемент входного сигнала в момент времени t, $w_{ij}(t)$ - вес связи от i-го элемента входного сигнала к нейрону j в момент времени t.

4. Выбор нейрона с наименьшим расстоянием. Выбирается нейрон-победитель j^*, для которого расстояние d_j самое малое.

5. Настраивание весов нейрона j^* и его соседей. Делается настраивание весов для нейрона j^* и всех нейронов из его окрестности N_E. Новые значения весов:

$$w_{ij}(t+1) = w_{ij}(t) + r(t)(x_i(t) - w_{ij}(t)),$$

где $r(t)$ - скорость обучения, которая уменьшается с течением времени (положительное число, меньше единицы).

6. Возвращение к шагу 2.

В алгоритме используется коэффициент скорости обучения, которое постепенно уменьшается. В результате центр устанавливается в определенной позиции, которая удовлетворительным образом кластеризует примеры, для которых данный нейрон является победителем.

Свойство топологической упорядоченности достигается в алгоритме с помощью использования понятия окрестности. Окрестность - это несколько нейронов, окружающих нейрон-победитель. Соответственно скорости обучения, размер окрестности постепенно уменьшается, так что сначала к нему принадлежит довольно большое число нейронов, на самых последних этапах окрестность становится нулевой и состоит лишь из нейрона-победителя. В алгоритме обучения коррекция применяется не только к нейрону-победителю, но и ко всем нейронам из его текущей окрестности. В результате такого

изменения окрестности, начальные довольно большие участки сети иммигрируют в сторону обучающих примеров.

Сеть формирует грубую структуру топологического порядка, при которой похожие примеры активируют группы нейронов, которые близко находятся на топологической карте. С каждой новой эпохой скорость обучения и размер окрестности уменьшаются, и внутри участков карты обнаруживаются более тонкие расхождения, что приводит к точному настраиванию каждого нейрона. Часто обучения умышленно разбивают на две фазы: более короткую, с большой скоростью обучения и больших окрестностей, и более продолжительную с маленькой скоростью обучения и нулевыми или почти нулевыми окрестностями.

После того, как сеть научена распознаванию структуры данных, ее можно использовать как средство визуализации при анализе данных.

ТЕМА 10. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

1. Основные понятия и принципы нейронных сетей
2. Пример работы простого генетического алгоритма
3. Достоинства и недостатки генетических алгоритмов
4. Применение генетических алгоритмов

1. Основные понятия и принципы генетических алгоритмов

Генетические Алгоритмы (ГА) – это адаптивные методы функциональной оптимизации, основанные на компьютерном имитационном моделировании биологической эволюции. Основные принципы ГА были сформулированы Джоном Голландом (Holland, 1975), и хорошо описаны во многих работах.

В настоящее время существует ряд теорий биологической эволюции (Ж.-Б.Ламарка, П.Тейяра де Шардена, К.Э.Бэра, Л.С.Берга, А.А.Любищева, С.В.Мейена и др.), однако, ни одна из них не считается общепризнанной. Наиболее известной и популярной, конечно, является теория Чарльза Дарвина, которую он представил в работе «Происхождение Видов» в 1859 году.

Эта теория, как и другие, содержит довольно много нерешенных проблем. Здесь мы можем отметить лишь некоторые наиболее известные из них. Дело в том, что возникновение нового вида «по алгоритму Дарвина» является крайне маловероятным событием, т.к. для этого требуется случайное возникновение в одной точке пространства и времени сразу не менее 100 особей нового вида, т.е. особей, которые могли бы иметь плодовитое потомство. При меньшем количестве особей вид обречен на вымирание. Поэтому процесс видообразования на основе случайных мутаций должен был бы занять несуразно много времени. Кроме того, «алгоритм Дарвина» не объясняет явной системности в многообразии возникающих форм, типа закона гомологичных рядов Н.И. Вавилова. Поэтому Л.С. Берг предложил очень интересную концепцию номогенеза – закономерной или направленной эволюции живого. В

этой концепции предполагается, что филогенез имеет определенное направление и смена форма является не случайной, а задается некоторым вектором, природа которого не ясна. Идеи номогенеза глубоко разработал и развил А.А. Любищев, высказавший гипотезу о математических закономерностях, которые определяют многообразие живых форм. Кроме того, Дарвин не смог показать механизм наследования, при котором поддерживается и закрепляется изменчивость. Это было на пятьдесят лет до того, как генетическая теория наследственности начала распространяться по миру, и за тридцать лет до того, как «эволюционный синтез» укрепил связь между теорией эволюции и молодой генетикой.

Тем ни менее и не смотря на свои недостатки, именно теория Дарвина традиционно и моделируется в ГА. Более того, возможно именно такое компьютерное моделирование и сравнение его результатов с картиной реальной эволюции жизни на Земле может быть и сыграет положительную роль в дальнейшей разработке наиболее адекватной теории биологической эволюции.

Теория Дарвина применима не к отдельным особям, а к популяциям – большому количеству особей одного вида, т.е. способных давать плодовитое потомство, находящейся в определенной статичной или динамичной внешней среде.

В основе модели эволюции Дарвина лежат случайные изменения отдельных материальных элементов живого организма при переходе от поколения к поколению. Целесообразные изменения, которые облегчают выживание и производство потомков в данной конкретной внешней среде, сохраняются и передаются потомству, т.е. наследуются. Особи, не имеющие соответствующих приспособлений, погибают, не оставив потомства или оставив его меньше, чем приспособленные (считается, что количество потомства пропорционально степени приспособленности). Поэтому в

результате естественного отбора возникает популяция из наиболее приспособленных особей, которая может стать основой нового вида.

Естественный отбор происходит в условиях конкуренции особей популяции, а иногда и различных видов, друг с другом за различные ресурсы, такие, например, как пища или вода. Кроме того, члены популяции одного вида часто конкурируют за привлечение брачного партнера. Те особи, которые наиболее приспособлены к окружающим условиям, будут иметь относительно больше шансов воспроизвести потомков. Слабо приспособленные особи либо совсем не произведут потомства, либо их потомство будет очень немногочисленным. Это означает, что гены от высоко адаптированных или приспособленных особей будут распространяться в увеличивающемся количестве потомков на каждом последующем поколении.

Таким образом, по сути дела каждый конкретный генетический алгоритм представляют имитационную модель некоторой определенной теории биологической эволюции или ее варианта. Вместе с тем необходимо отметить, что сами исследователи биологической эволюции пока еще не до конца определились с критериями и методами определения степени существенности для поддерживаемой ими теории эволюции тех или иных биологических процессов, которые собственно и моделируются в генетических алгоритмах.

2. Пример работы простого генетического алгоритма

На рисунке 47 приведен пример простого генетического алгоритма.

Работа ГА представляет собой итерационный процесс, который продолжается до тех пор, пока поколения не перестанут существенно отличаться друг от друга, или не пройдет заданное количество поколений или заданное время. Для каждого поколения реализуются отбор, кроссовер (скрещивание) и мутация. Рассмотрим этот алгоритм.

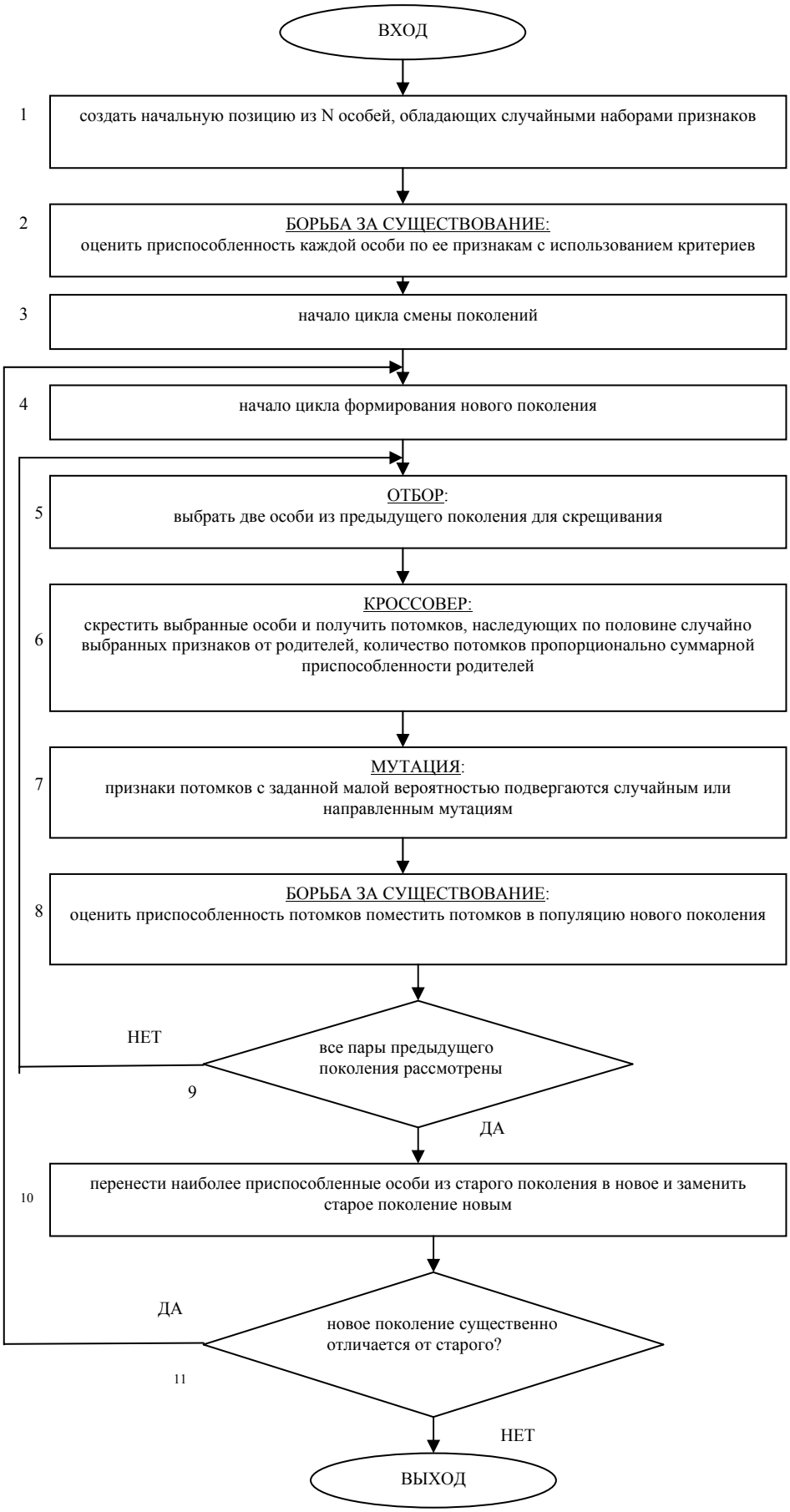


Рисунок 46 - Простой генетический алгоритм

Шаг 1: генерируется начальная популяция, состоящая из *N* особей со случайными наборами признаков.

Шаг 2 (борьба за существование): вычисляется *абсолютная приспособленность* каждой особи популяции к условиям среды *f(i)* и суммарная приспособленность особей популяции, характеризующая приспособленность всей популяции. Затем при *пропорциональном отборе* для каждой особи вычисляется ее *относительный вклад в суммарную приспособленность популяции* $P_s(i)$, т.е. отношение ее абсолютной приспособленности *f(i)* к суммарной приспособленности всех особей популяции (1):

$$P_s(i) = \frac{f(i)}{\sum_{i=1}^{n} f(i)} \qquad (1)$$

В выражении (1) сразу обращает на себя внимание возможность сравнения абсолютной приспособленности *i-й* особи *f(i)* не с суммарной приспособленностью всех особей популяции, а со средней абсолютной приспособленностью особи популяции (2):

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f(i) \qquad (2)$$

Тогда получим (3):

$$P_s(i) = \frac{f(i)}{\bar{f}} = \frac{f(i)}{\frac{1}{N}\sum_{i=1}^{N} f(i)} \qquad (3)$$

Если взять логарифм по основанию 2 от выражения (3), то получим количество информации, содержащееся в признаках особи о том, что она выживет и даст потомство (4).

$$L(i) = Log_2 \frac{f(i)}{\bar{f}} \qquad (4)$$

Необходимо отметить, что эта формула совпадает с формулой для семантического количества информации Харкевича, если целью считать индивидуальное выживание и продолжение рода. Это значит, что даже чисто

формально приспособленность особи представляет собой количество информации, содержащееся в ее фенотипе о продолжении ее генотипа в последующих поколениях.

Поскольку количество потомства особи пропорционально ее приспособленности, то естественно считать, что если это количество информации:

– *положительно,* то данная особь выживает и дает потомство, численность которого пропорциональна этому количеству информации;

– *равно нулю,* то особь доживает до половозрелого возраста, но потомства не дает (его численность равна нулю);

– *меньше нуля,* то особь погибает до достижения половозрелого возраста.

Таким образом, можно сделать фундаментальный вывод, имеющий даже мировоззренческое звучание, о том, что естественный отбор представляет собой процесс генерации и накопления информации о выживании и продолжении рода в ряде поколений популяции, как системы.

Это накопление информации происходит на различных уровнях иерархии популяции, как системы, включающей:

– элементы системы: отдельные особи;

– взаимосвязи между элементами: отношения между особями в популяции, обеспечивающие передачу последующим поколениям максимального количества информации об их выживании и продолжении рода (путем скрещивания наиболее приспособленных особей и наследования рациональных приобретений);

– цель системы: сохранение и развитие популяции, реализуется через цели особей: индивидуальное выживание и продолжение рода.

Фенотип соответствует генотипу и представляет собой его внешнее проявление в признаках особи. Особь взаимодействует с окружающей средой и другими особями в соответствии со своим фенотипом. В случае, если это

взаимодействие удачно, то особь передает генетическую информацию, определяющую фенотип, последующим поколениям.

Шаг 3: начало цикла смены поколений.

Шаг 4: начало цикла формирования нового поколения.

Шаг 5 (отбор): осуществляется пропорциональный отбор особей, которые могут участвовать в продолжении рода. Отбираются только те особи популяции, у которых количество информации в фенотипе и генотипе о выживании и продолжении рода положительно, причем вероятность выбора пропорциональна этому количеству информации.

Шаг 6 (кроссовер): отобранные для продолжения рода на предыдущем шаге особи с заданной вероятностью P_c подвергаются скрещиванию или кроссоверу (рекомбинации).

Если кроссовер происходит, то потомки получают по половине случайным образом определенных признаков от каждого из родителей. Численность потомства пропорциональна суммарной приспособленности родителей. В некоторых вариантах ГА потомки после своего появления заменяют собой родителей и переходят к мутации.

Если кроссовер не происходит, то исходные особи – несостоявшиеся родители, переходят на стадию мутации.

Шаг 7 (мутация): выполняются операторы мутации. При этом признаки потомков с вероятностью P_m случайным образом изменяются на другие. Отметим, что использование механизма случайных мутаций роднит генетические алгоритмы с таким широко известным методом имитационного моделирования, как метод Монте-Карло.

Шаг 8 (борьба за существование): оценивается приспособленность потомков (по тому же алгоритму, что и на шаге 2).

Шаг 9: проверяется, все ли отобранные особи дали потомство.

Если нет, то происходит переход на шаг 5 и продолжается формирование нового поколения, иначе – переход на следующий шаг 10.

Шаг 10: происходит смена поколений:

– потомки помещаются в новое поколение;

– наиболее приспособленные особи из старого поколения переносятся в новое, причем для каждой из них это возможно не более заданного количества раз;

– полученная новая популяция замещает собой старую.

Шаг 11: проверяется выполнение условия останова генетического алгоритма. Выход из генетического алгоритма происходит либо тогда, когда новые поколения перестают существенно отличаться от предыдущих, т.е., как говорят, «алгоритм сходится», либо когда пройдено заданное количество поколений или заданное время работы алгоритма (чтобы не было «зацикливания» и динамического зависания в случае, когда решение не может быть найдено в заданное время).

Если ГА сошелся, то это означает, что решение найдено, т.е. получено поколение, идеально приспособленное к условиям данной фиксированной среды обитания.

Иначе – переход на шаг 4 – начало формирования нового поколения.

В реальной биологической эволюции этим дело не ограничивается, т.к. любая популяция кроме освоения некоторой экологической ниши пытается также выйти за ее пределы освоить и другие ниши, как правило «смежные». Именно за счет этих процессов жизнь вышла из моря на сушу, проникла в воздушное пространство и поверхностный слой почвы, а сейчас осваивает космическое пространство.

Конечно, реальные генетические алгоритмы, на которых проводятся научные исследования, чаще всего мало похожи на приведенный пример. Исследователи экспериментируют с различными параметрами генетических алгоритмов, например: способами отбора особей для скрещивания; критериями приспособленности и жесткостью влияния факторов среды; способами выбора признаков, передающихся от родителей потомкам (рецессивные и не

рецессивные гены и т.д.); интенсивностью, видом случайного распределения и направленностью мутаций; различными подходами к воспроизводству и отбору.

Поэтому под термином «генетические алгоритмы» по сути дела надо понимать не одну модель, а довольно широкий класс алгоритмов, подчас мало похожих друг на друга.

В настоящее время рассматривается много различных операторов отбора, кроссовера и мутации:

- турнирный отбор, реализует n турниров, чтобы выбрать n особей, при этом каждый турнир построен на выборке k элементов из популяции, и выбора лучшей особи среди них (наиболее распространен турнирный отбор с k=2);
- элитный отбор гарантирует, что при отборе обязательно будут выживать лучший или лучшие члены популяции совокупности (наиболее распространена процедура обязательного сохранения только одной лучшей особи, если она не прошла как другие через процесс отбора, кроссовера и мутации);
- двухточечный кроссовер - выбираются 2 точки раздела, и родители обмениваются промежутками между ними;
- равномерный кроссовер - один из детей наследует каждый бит с вероятностью $p0$ у первого родителя и с $(1 - p0)$ у второго, второй ребенок получает не унаследованные первым биты. Обычно $p0 = 0.5$.

Несмотря на то, что модели биологической эволюции, реализуемые в ГА, обычно сильно упрощены по сравнению с природным оригиналом, тем ни менее ГА являются мощным средством, которое может с успехом применяться для решения широкого класса прикладных задач, включая те, которые трудно, а иногда и вовсе невозможно, решить другими методами.

3. Достоинства и недостатки генетических алгоритмов

Однако генетический алгоритм не гарантирует обнаружения глобального решения за приемлемое время. И не гарантируют и того, что найденное решение будет оптимальным решением. Тем ни менее они применимы для *поиска* «достаточно хорошего» решения задачи за «достаточно короткое время». ГА представляют собой разновидность алгоритмов поиска и имеют преимущества перед другими алгоритмами при очень больших размерностях задач и отсутствия упорядоченности в исходных данных, когда альтернативой им является метод полного перебора вариантов.

В случаях, когда задача может быть решена специально разработанным для нее методом, практически всегда такие методы будут эффективнее генетических алгоритмов как по быстродействию, так и по точности найденных решений.

Главным же достоинством ГА является то, что они могут применяться для решения сложных неформализованных задач, для которых не разработано специальных методов, т.е. они обеспечивают решение проблем. Но даже в тех случаях, для которых хорошо работают существующие методики, можно достигнуть интересных результатов сочетая их с генетическими алгоритмами.

4. Применение генетических алгоритмов

Генетические алгоритмы в разных формах применяются к решению многих научных и технических проблем. Они используются при создании других вычислительных структур, например, автоматов или сетей сортировки. В машинном обучении используются при проектировании нейронных сетей или управлении роботами. Они также применяются при моделировании развития в разных предметных областях, включая биологические (экология, иммунология и популярная генетика) и социальные (такие как экономика и политические системы) системы.

Тем не менее, возможно популярное применение генетических алгоритмов - оптимизация многопараметрических функций. Большинство реальных задач могут быть сформулированы как поиск оптимального значения, где значение - сложная функция, зависящая от определенных входных параметров. В некоторых случаях, нужно найти те значения параметров, при которых достигается точное значение функции. В других случаях, точный оптимум не нужен - решением может считаться любое значение, лучшее за определенную заданную величину. В этом случае, генетические алгоритмы - приемлемый метод для поиска «приемлемых» значений. Сила генетического алгоритма состоит в его способности манипулировать одновременно многими параметрами, которая используется в сотнях прикладных программ, включая проектирование самолетов, настраивании параметров алгоритмов и поиске стойких состояний систем нелинейных дифференциальных уравнений.

Генетические алгоритмы являются эффективной процедурой поиска, которая конкурирует с другими процедурами. Эффективность генетических алгоритмов сильно зависит от таких деталей, как метод кодирования решений, операторов, настраивания параметров, отдельных критериев успеха.

Можно выделить следующие задачи, для решения которых можно использовать генетические алгоритмы:

1. Оптимизация функций.
2. Оптимизация запросов в базах данных.
3. Разнообразные задачи на графах (задача коммивояжера, раскраска).
4. Настройка и обучение искусственной нейронной сети.
5. Задачи компоновки.
6. Составление расписаний.
7. Игровые стратегии
8. Теория приближений.
9. Искусственная жизнь.
10. Биоинформатика (фолдинг белков).

ЛИТЕРАТУРА

1. Башмаков, А.И. Интеллектуальные информационные технологии: учебное пособие / А.И. Башмаков, И.А. Башмаков. - М. : Изд-во МГТУ им. Н.Э. Баумана, 2005. - 304 с.

2. Достоверный и правдоподобный вывод в интеллектуальных системах : учебник / В.Н. Вагин [и др.] ; общ. ред. В.Н. Вагина. - М.: ФИЗМАТЛИТ, 2008. - 712 с.

3. Калан, Р. Основные концепции нейронных сетей. : пер. с англ. - М. : Издательский дом «Вильямс», 2001. - 287 с.

4. Костров, Б.В. Искусственный интеллект и робототехника / Б.В. Костров, В.Н. Ручкин, В.А. Фулин. – М. : Диалог-МИФИ, 2008. – 224 с.

5. Круглов, В.В. Искусственные нейронные сети. Теория и практика / В.В. Круглов, В.В. Борисов. - М. : Горячая линия-Телеком, 2001. - 382с.

6. Рассел, С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг. - М. : Издательский дом «Вильямс», 2006. - 1410 с.

7. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: пер. с польского / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М. : Горячая линия – Телеком, 2006. - 452 с.

8. Ручкин, В. Н. Универсальный искусственный интеллект и экспертные системы / В.Н. Ручкин, В.А. Фулин. - СПб. : БХВ-Петербург, 2009.– 240 с.

9. Смолин, Д.В. Введение в искусственный интеллект : конспект лекций / Д.В. Смолин. - М. : ФИЗМАТЛИТ, 2007. - 264 с.

10. Ясницкий, Л.Н. Введение в искусственный интеллект : учебное пособие / Л.Н. Ясницкий. - М. : Издательский центр «Академия», 2005. - 176 с.

11. Яхъяева, Г.Э. Нечеткие множества и нейронный сети : учебное пособие / Г.Э. Яхъяева. – М. : Интернет-Университет информационных технологий; Бином. Лаборатория знаний, 2006. – 316 с.

Учебное издание

Амаева Лиана Анатолиевна

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ТЕКСТЫ ЛЕКЦИЙ

Корректор Габдурахимова Т.М.
Худ. редактор Федорова Л.Г.

Сдано в набор 30.10.2010.
Подписано в печать 3.12.2010.
Бумага писчая. Гарнитура Таймс.
Усл. печ. л. 8,25. Тираж 100.
Заказ №53.

НХТИ (филиал) ГОУ ВПО «КГТУ», г. Нижнекамск, 423570,
ул. 30 лет Победы, д. 5а.