Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Воронежский государственный архитектурно-строительный университет»

Д.В. Сысоев, О.В. Курипта, Д.К. Проскурин

ВВЕДЕНИЕ В ТЕОРИЮ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебное пособие

Воронеж 2013

УДК
ББК
В95

Рецензенты:
кафедра вычислительной техники и информационных систем
Воронежской государственной лесотехнической академии;
Т.Н. Князева, д-р техн. наук, проф.
Воронежской государственной медицинской академии

Сысоев, Д.В.
В95 **Введение в теорию искусственного интеллекта**: учебное пособие / Д.В. Сысоев, О.В. Курипта, Д.К. Проскурин; Воронежский ГАСУ. – Воронеж, 2013. – 170 с.

В данном учебном пособии рассматриваются современные подходы к моделированию нейронных сетей, применению нечеткой логики для решения задач прогнозирования, классификации, аппроксимации.

Учебное пособие предназначено для студентов по специальности 230700 - «Прикладная информатика». А также для научных работников, аспирантов, работающих в области моделирования искусственного интеллекта.

Табл.: 16. Рис.: 109. Библиогр.: 85 назв.

УДК
ББК

Печатается по решению редакционно-издательского совета
Воронежского ГАСУ

ISBN

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Данное учебное пособие предназначено для студентов специальности 230700 – «Прикладная информатика» в рамках учебного плана. Пособие представляет собой теоретические основы построения, обучения и применения нейронных технологий, использование нечеткой логики в построении систем искусственного интеллекта, а также практические работы, в виде лабораторных занятий.

Все лабораторные работы представляют из себя теоретические материалы и пошаговое описание практической части лабораторного задания. Практическая часть лабораторного занятия выполняется в редакторе MatLAB2009.

Выполненная работа сохраняется на винчестере в виде файла для последующей защиты. Защита работы это предоставление оформленного отчета по выполненной работе и электронной версии по выполненной работе.

В каждой лабораторной работе сформулирована ее цель, представлена методика выполнения, даны сведения об оформлении отчета для защиты лабораторной работы, а также в приложении даны варианты выполнения работы для самостоятельного обучения. Используемый в учебном пособии теоретический материал заимствован из литературы, библиографический список которого представлен в конце пособия.

Цель данного учебного пособия: овладеть основными навыками работы в редакторе MatLAB2009, применительно к моделированию нейронных сетей, к моделированию элементов нечеткой логики при построении систем искусственного интеллекта. Выполнить приведенные практические работы и защитить результат их выполнения.

ВВЕДЕНИЕ

В течение тысяч лет человек пытается понять, как он думает. В области искусственного интеллекта специалисты пытаются не только понять природу интеллекта, но и создать интеллектуальные сущности.

Искусственный интеллект (ИИ)- это одна из новейших областей науки. Первые работы в этой области начались вскоре после второй мировой войны, а само ее название было предложено в 1956 году. В настоящее время тематика искусственного интеллекта охватывает огромный перечень научных направлений, начиная с таких задач общего характера, как обучение и восприятие, и заканчивая такими специальными задачами, как игра в шахматы, доказательство математических теорем, сочинение поэтических произведений и диагностика заболеваний. В искусственном интеллекте систематизируются и автоматизируются интеллектуальные задачи и поэтому эта область касается любой сферы интеллектуальной деятельности человека. В этом смысле искусственный интеллект является поистине универсальной научной областью.

Интеллектуальные информационные системы проникают во все сферы нашей жизни, поэтому трудно провести строгую классификацию направлений, по которым ведутся активные и многочисленные исследования в области ИИ. Рассмотрим кратко некоторые из них.

Разработка интеллектуальных информационных систем или систем, основанных на знаниях. Это одно из главных направлений ИИ. Основной целью построения таких систем являются выявление, исследование и применение знаний высококвалифицированных экспертов для решения сложных задач, возникающих на практике. При построении систем, основанных на знаниях, используются знания, накопленные экспертами в виде конкретных правил решения тех или иных задач. Это направление преследует цель имитации человеческого искусства анализа неструктурированных и слабоструктурированных проблем [23]. В данной области исследований осуществляется разработка моделей представления, извлечения и структурирования знаний, а также изучаются проблемы создания баз знаний, образующих ядро систем основанных на знаниях. Частным случаем, которых являются экспертные системы (ЭС).

Разработка естественно-языковых интерфейсов и машинный перевод. Системы машинного перевода с одного естественного языка на другой обеспечивают быстроту и систематичность доступа к информации, оперативность и единообразие перевода больших потоков, как правило, научно-технических текстов [16]. Системы машинного перевода строятся как интеллектуальные системы, поскольку в их основе лежат базы знаний в определенной предметной области и сложные модели, обеспечивающие дополнительную трансляцию «исходный язык оригинала — язык смысла — язык перевода». Кроме того, в них осуществляется ассоциативный поиск аналогичных фрагментов текста и их переводов в специальных базах данных.

Данное направление охватывает также исследования методов и разработку систем, обеспечивающих реализацию процесса общения человека с компьютером на естественном языке [16].

Обучение и самообучение. Эта актуальная область ИИ включает модели, методы и алгоритмы, ориентированные на автоматическое накопление и формирование знаний с использованием процедур анализа и обобщения данных [6, 45].

Распознавание образов. Это одно из самых ранних направлений ИИ, в котором распознавание объектов осуществляется на основании применения специального математического аппарата, обеспечивающего отнесение объектов к классам [17], а классы описываются совокупностями определенных значений признаков.

Игры и машинное творчество. Машинное творчество охватывает сочинение компьютерной музыки [15], интеллектуальные системы для изобретения новых объектов [3, 46]. Создание интеллектуальных компьютерных игр является одним из самых развитых коммерческих направлений в сфере разработки программного обеспечения. Кроме того, компьютерные игры предоставляют мощный арсенал разнообразных средств, используемых для обучения.

Программное обеспечение систем ИИ. Инструментальные средства для разработки интеллектуальных систем включают специальные языки программирования, ориентированные на обработку символьной информации (*smalltalk, рефал*), языки логического программирования (*prolog*), языки представления знаний (*ops 5, krl*), интегрированные программные среды, содержащие арсенал инструментальных средств для создания систем ИИ (*arts, guru*), а также оболочки экспертных систем (*exsys professional, эксперт*), которые позволяют создавать прикладные ЭС, не прибегая к программированию [18, 40].

Новые архитектуры компьютеров. Это направление связано с созданием компьютеров не фон-неймановской архитектуры, ориентированных на обработку символьной информации. Известны удачные промышленные решения параллельных и векторных компьютеров [2, 18], однако в настоящее время они имеют весьма высокую стоимость, а также недостаточную совместимость с существующими вычислительными средствами.

Интеллектуальные роботы. Создание интеллектуальных роботов составляет конечную цель робототехники. В настоящее время в основном используются программируемые манипуляторы с жесткой схемой управления, названные роботами первого поколения. Основными сдерживающими факторами в разработке автономных роботов являются нерешенные проблемы в области интерпретации знаний, машинного зрения, адекватного хранения и обработки трехмерной визуальной информации.

Интеллектуальной информационной системой (ИИС) называют автоматизированную информационную систему, основанную на знаниях, или

комплекс программных, лингвистических и логико-математических средств для реализации основной задачи - осуществления поддержки деятельности человека и поиска информации в режиме продвинутого диалога на естественном языке.

Кроме того, информационно-вычислительными системами с интеллектуальной поддержкой для решения сложных задач называют те системы, в которых логическая обработка информации превалирует над вычислительной.

Таким образом, любая информационная система, решающая интеллектуальную задачу или использующая методы искусственного интеллекта, относится к интеллектуальным системам. Для интеллектуальных информационных систем характерны следующие признаки [42]:

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

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

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

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

Адаптивность – способность к развитию системы в соответствии с объективными изменениями модели проблемной области.

В соответствии с перечисленными признаками ИИС делятся на (рис. 1):

- системы с коммутативными способностями (*с интеллектуальным интерфейсом*);
- экспертные системы (*системы для решения сложных задач*);
- самообучающиеся системы (*системы, способные к самообучению*);
- адаптивные системы (*адаптивные информационные системы*).

Интеллектуальные базы данных отличаются от обычных баз данных возможностью выборки по запросу необходимой информации, которая может явно не храниться, а выводиться из имеющейся в базе данных.

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний. Для этого необходимо решать задачи морфологического, синтаксического и семантического анализа и синтеза

высказываний на естественном языке. Так, морфологический анализ предполагает распознавание и проверку правильности написания слов по словарям, синтаксический контроль – разложение входных сообщений на отдельные компоненты (определение структуры) с проверкой соответствия грамматическим правилам внутреннего представления знаний и выявления недостающих частей и, наконец, семантический анализ – установление смысловой правильности синтаксических конструкций. Синтез высказываний решает обратную задачу преобразования внутреннего представления информации в естественно-языковое.

Рис. 1. Классификация интеллектуальных информационных систем по типам систем

Естественно-языковой интерфейс используется для:

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

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы отличаются возможностью более сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Таким образом, механизм поиска работает прежде всего с базой знаний ключевых слов, а уже затем непосредственно с текстом.

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В отличие от обычных систем помощи, навязывающих пользователю схему поиска требуемой информации, в системах контекстной помощи пользователь описывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует и сама выполняет поиск относящихся к ситуации рекомендаций.

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Графические образы в наглядном и интегрированном виде описывают множество параметров изучаемой ситуации.

Экспертные системы предназначены для решения задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в рассматриваемой проблемной области.

Экспертные системы как самостоятельное направление в искусственном интеллекте сформировалось в конце 1970-х гг. История ЭС началась с сообщения японского комитета по разработке ЭВМ пятого поколения, в котором основное внимание уделялось развитию «интеллектуальных способностей» компьютеров с тем, чтобы они могли оперировать не только данными, но и знаниями, как это делают специалисты (эксперты) при выработке умозаключений. Одним из важных свойств ЭС является способность объяснить ход своих рассуждений понятным для пользователя образом [48].

Область исследования ЭС называют «***инженерией знаний***». Этот термин был введен Е. Фейгенбаумом и в его трактовке означает «привнесение принципов и инструментария из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов». Другими словами, ЭС применяются для решения неформализованных проблем, к которым относят задачи, обладающие одной (или несколькими) из следующих характеристик:

- задачи не могут быть представлены в числовой форме;
- исходные данные и знания о предметной области обладают неоднозначностью, неточностью, противоречивостью;
- цели нельзя выразить с помощью четко определенной целевой функции;
- не существует однозначного алгоритмического решения задачи;
- алгоритмическое решение существует, но его нельзя использовать по причине большой размерности пространства решений и ограничений на ресурсы (времени, памяти).

Главное отличие ЭС и систем искусственного интеллекта от систем обработки данных состоит в том, что в них используется символьный, а не числовой способ представления данных, а в качестве методов обработки

информации применяются процедуры логического вывода и эвристического поиска решений.

ЭС охватывают самые разные предметные области, среди которых лидируют бизнес, производство, медицина, проектирование и системы управления [6, 16, 40, 42, 48, 53].

Многоагентные системы. Для таких динамических систем характерна интеграция в базе знаний нескольких разнородных источников знаний, обменивающихся между собой получаемыми результатами на динамической основе. Для таких систем характерны следующие особенности:

- проведение альтернативных рассуждений на основе использования различных источников знаний с механизмом устранения противоречий;
- распределенное решение проблем, которые разбиваются на параллельно решаемые задачи, соответствующие самостоятельным источникам знаний;
- применение множества стратегий работы механизма вывода заключений в зависимости от типа решаемой проблемы;
- обработка больших массивов данных, содержащихся в базе данных;
- использование различных математических моделей и внешних процедур, хранимых в базе моделей;
- способность прерывания решения задач в связи с необходимостью получения дополнительных данных и знаний от пользователей, моделей, параллельно решаемых задач.

В основе ***самообучающихся систем*** лежат методы автоматической классификации примеров ситуаций реальной практики. Характерными признаками самообучающихся систем являются:

- самообучающиеся системы «с учителем», когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций (классообразующего признака);
- самообучающиеся системы «без учителя», когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

Индуктивные системы используют обобщение примеров по принципу от частного к общему. Процесс классификации примеров осуществляется следующим образом:

- Выбирается признак классификации из множества заданных (либо последовательно, либо по какому-либо правилу, например в соответствии с максимальным числом получаемых подмножеств примеров).
- По значению признака множество примеров разбивается на подмножества.
- Выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному подклассу.
- Если какое-то подмножество примеров принадлежит одному подклассу, то есть у всех примеров подмножества совпадает значение классообразующего признака, то процесс классификации заканчивается

(при этом остальные признаки классификации не рассматриваются).

- Для подмножеств примеров с несовпадающим значением классообразующего признака процесс классификации продолжается начиная с первого пункта (каждое подмножество примеров становится классифицируемым множеством).

Нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессоров. Каждый процессор такой сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам [25].

В экспертных системах, ***основанных на прецедентах*** (аналогиях), база знаний содержит описания не обобщенных ситуаций, а собственно сами ситуации или прецеденты. Поиск решения проблемы в экспертных системах, основанных на прецедентах, сводится к поиску по аналогии (то есть абдуктивный вывод от частного к частному).

В отличие от интеллектуальной базы данных, ***информационное хранилище*** представляет собой хранилище извлеченной значимой информации из оперативной базы данных, которое предназначено для оперативного ситуационного анализа данных [5]. Типичными задачами оперативного ситуационного анализа являются:

- определение профиля потребителей конкретных объектов хранения;
- предсказание изменений объектов хранения во времени;
- анализ зависимостей признаков ситуаций (корреляционный анализ).

Адаптивная информационная система – это информационная система, которая изменяет свою структуру в соответствии с изменением модели проблемной области. При этом:

- адаптивная информационная система должна в каждый момент времени адекватно поддерживать организацию бизнес-процессов;
- адаптивная информационная система должна проводить адаптацию всякий раз, как возникает потребность в реорганизации бизнес-процессов;
- реконструкция информационной системы должна проводиться быстро и с минимальными затратами.

Ядром адаптивной информационной системы является постоянно развиваемая модель проблемной области, поддерживаемая в специальной базе знаний – репозитории. На основе ядра осуществляется генерация или конфигурация программного обеспечения. Таким образом, проектирование и адаптация ИС сводится, к построению модели проблемной области и ее своевременной корректировке.

Если рассматривать интеллектуальные информационные системы с точки зрения решаемой задачи, то можно выделить системы управления и справочные системы, системы компьютерной лингвистики, системы распознавания,

игровые системы и системы создания ИИС (рис. 2).

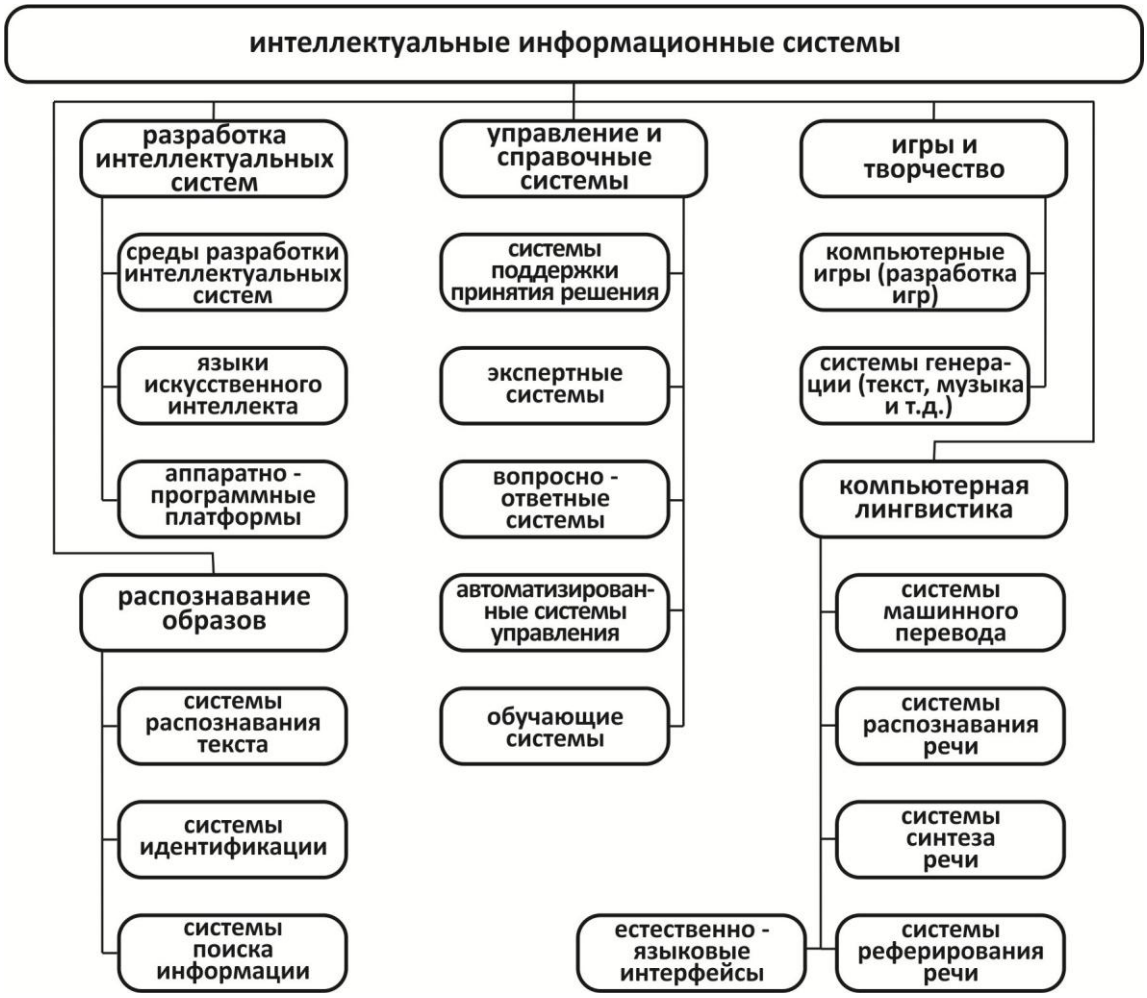


Рис. 2. Классификация интеллектуальных информационных систем по решаемым задачам

Если классифицировать интеллектуальные информационные системы по критерию «используемые методы», то они делятся на жесткие, мягкие и гибридные (рис. 3).

Рис. 3. Классификация интеллектуальных информационных систем по методам

Мягкие вычисления – это сложная компьютерная методология, основанная на нечеткой логике, генетических вычислениях, нейрокомпьютинге и вероятностных вычислениях. ***Жесткие вычисления*** – традиционные компьютерные вычисления (не мягкие). ***Гибридные системы*** – системы, использующие более чем одну компьютерную технологию (в случае интеллектуальных систем – технологии искусственного интеллекта).

1. ВВЕДЕНИЕ В ТЕОРИЮ НЕЙРОННЫХ СЕТЕЙ

Искусственные нейронные сети (ИНС) строятся по принципам организации и функционирования их биологических аналогов. Они способны решать широкий круг задач распознавания образов, идентификации, прогнозирования, оптимизации, управления сложными объектами. Дальнейшее повышение производительности компьютеров все в большой мере связывают с ИНС, в частности, с нейрокомпьютерами (НК), основу которых составляет искусственная нейронная сеть.

Термин ***«нейронные сети»*** сформировался к середине 50-х годов XX века. Основные результаты в этой области связаны с именами У. Маккалоха, Д. Хебба, Ф. Розенблатта, М. Минского, Дж. Хопфилда.

Основные области применения нейронных сетей:

- ***Классификация образов.*** Задача состоит в указании принадлежности входного образа, представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови.
- ***Кластеризация/категоризация.*** При решении задачи кластеризации, которая известна также как классификация образов без учителя, отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие образы в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.
- ***Аппроксимация функций.*** Предположим, что имеется обучающая выборка $((X_1, Y_2), (X_2, Y_2)$,..., $(X_N, Y_N))$, которая генерируется неизвестной функцией, искаженной шумом. Задача аппроксимации будет состоять в нахождении оценки этой функции.
- ***Предсказание/прогноз***. Пусть заданы N дискретных отсчетов $\{y(t_1), y(t_2)$,..., $y(t_n)\}$ в последовательные моменты времени $t_1, t_2,..., t_n$. Задача состоит в предсказании значения $y(t_n+1)$ в момент t_n+1. Прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике.
- ***Оптимизация***. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей оптимизации является нахождение решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию.
- ***Память, адресуемая по содержанию.*** В модели вычислений фон Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Память, адресуемая по содержанию, или ассоциативная память, доступна

по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании перспективных информационно-вычислительных систем.

- ***Управление.*** Рассмотрим динамическую систему, заданную совокупностью {u(t), y(t)}, где u(t) - является входным управляющим воздействием, а y(t) - выходом системы в момент времени f. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия u(f), при котором система следует по желаемой траектории, диктуемой эталонной моделью.

Как известно, для решения таких задач традиционно применяются два основных подхода. Первый, основанный на правилах (rule-based), характерен для экспертных систем. Он базируется на описании предметной области в виде набора правил (аксиом) ***«если..., то...»*** и правил вывода. Искомое знание представляется в этом случае теоремой, истинность которой доказывается посредством построения цепочки вывода. При этом подходе, однако, необходимо заранее знать весь набор закономерностей, описывающих предметную область. При использовании другого подхода, основанного на примерах (case-based), надо лишь иметь достаточное количество примеров для настройки адаптивной системы с заданной степенью достоверности. Нейронные сети представляют собой классический пример такого подхода.

1.1 Основные положения теории искусственных нейронных сетей

Под нейронными сетями подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Они представляют собой распределенные и параллельные системы, способные к адаптивному обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является искусственный нейрон или просто нейрон, названный так по аналогии с биологическим прототипом.

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

Нейрон является особой биологической клеткой, которая обрабатывает информацию (рис. 4). Он состоит из *тела* и отростков нервных волокон двух типов - *дендритов*, по которым принимаются импульсы, и единственного *аксона*, по которому нейрон может передавать импульс. Тело нейрона включает *ядро*, которое содержит информацию о наследственных свойствах, и *плазму*, обладающую молекулярными средствами для производства необходимых

нейрону материалов. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчика), который в конце разветвляется на волокна. На окончаниях этих волокон находятся специальные образования - синапсы, которые влияют на величину импульсов.

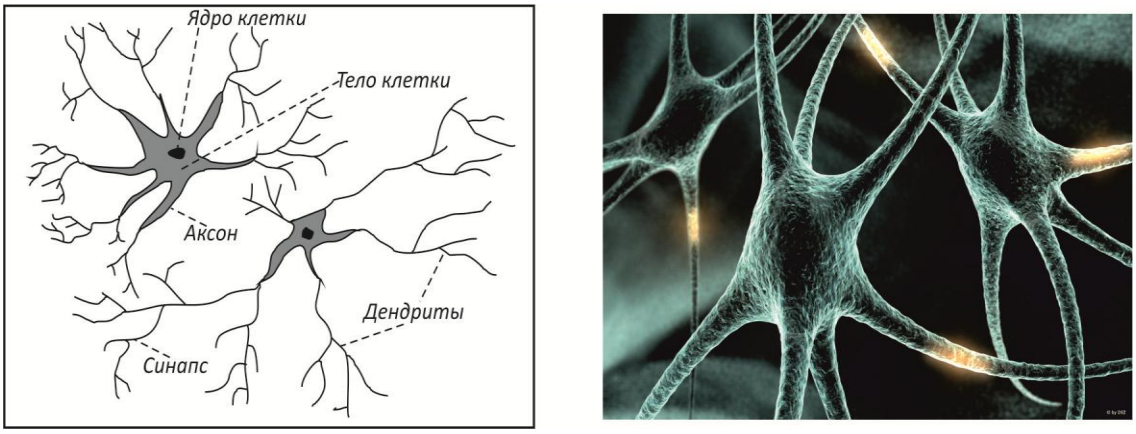


Рис. 4. Взаимосвязь биологических нейронов

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Результативность передачи импульса синапсом может настраиваться проходящими через него сигналами так, что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память. Важно отметить, что веса синапсов могут изменяться со временем, а значит, меняется и поведение соответствующих нейронов.

Структура и свойства искусственного нейрона. Нейрон является составной частью нейронной сети. На рисунке 5 показана его структура. Он состоит из элементов трех типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапсы осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи, (вес синапса). Сумматор выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента - выхода сумматора. Эта функция называется функцией активации или передаточной функцией нейрона.

Нейрон в целом реализует скалярную функцию векторного аргумента. Математическая модель нейрона:

$$S = \sum_{i=1}^{n} w_i \, x_i + b, \tag{1.1}$$

$$y = f(S), \tag{1.2}$$

где w_i - вес синапса, $i = 1...n$; b - значение смещения; S - результат суммирования; x_i - компонент входного вектора (входной сигнал); y - выходной сигнал нейрона; n - число входов нейрона; f - нелинейное преобразование (функция активации).

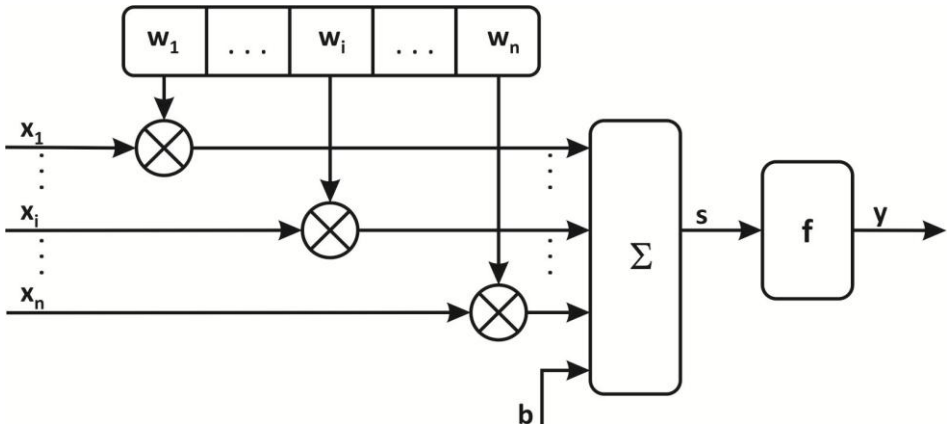


Рис. 5. Структура искусственного нейрона

В общем случае входной сигнал, весовые коэффициенты и смещение могут принимать действительные значения, а во многих практических задачах - лишь некоторые фиксированные значения. Выход (y) определяется видом функции активации и может быть как действительным, так и целым.

Синаптические связи с положительными весами называют ***возбуждающими***, с отрицательными весами - ***тормозящими***.

Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов. Чтобы подчеркнуть различие нейронов биологических и искусственных, вторые иногда называют нейроноподобными элементами или формальными нейронами.

На входной сигнал (S) нелинейный преобразователь отвечает выходным сигналом $f(S)$, который представляет собой выход у нейрона. Примеры основных активационных функций представлены в таблице 1 и на рисунке 6.

Одной из наиболее распространенных функций активации является нелинейная функция активации с насыщением, так называемая логистическая функция или сигмоид (функция S - образного вида):

$$f(S) = \frac{1}{1 + e^{-S}} \tag{1.3}$$

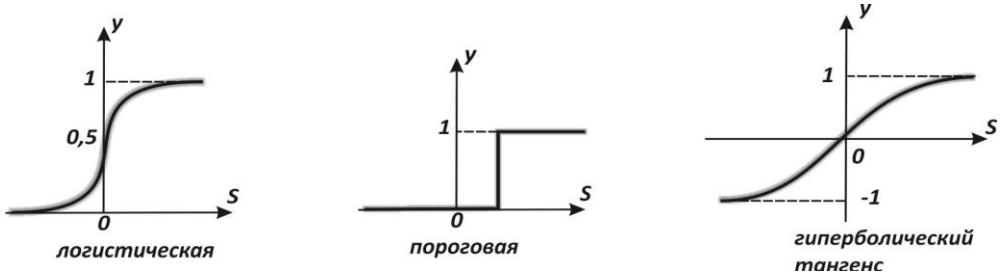


Рис. 6. Примеры функций активации нейронных сетей

Таблица 1.1.

Основные функции активации нейронных сетей

Название	**Формула**	**Область значений**
Линейная	$f(S) = kS$	$(-\infty; \infty)$
Радиальная базисная (гауссова)	$f(S) = e^{-S^2}$	$(0; 1)$
Полулинейная	$f(S) = \begin{cases} kS, & S > 0 \\ 0, & S \le 0 \end{cases}$	$(0; \infty)$
Полулинейная с насыщением	$f(S) = \begin{cases} 0, & S \le 0 \\ S, & 0 < S < 1 \\ 1, & S \ge 1 \end{cases}$	$(0; 1)$
Треугольная	$f(S) = \begin{cases} 1 - \lvert S \rvert, & \lvert S \rvert \le 1 \\ 0, & \lvert S \rvert > 1 \end{cases}$	$(0; 1)$
Логистическая (сигмоидальная)	$f(S) = \frac{1}{1 + e^{-S}}$	$(0; 1)$
Гиперболический Тангенс	$f(S) = \frac{e^{S} - e^{-S}}{e^{S} + e^{-S}}$	$(-1; 1)$
Экспоненциальная	$f(S) = e^{-kS}$	$(0; \infty)$
Синусоидальная	$f(S) = \sin(S)$	$(-1; 1)$
Линейная с насыщением	$f(S) = \begin{cases} -1, & S \le -1 \\ S, & -1 < S < 1 \\ 1, & S \ge 1 \end{cases}$	$(-1; 1)$
Пороговая	$f(S) = \begin{cases} 1, & S \ge 0 \\ 0, & S < 0 \end{cases}$	$(0; 1)$
Модульная	$f(S) = \lvert S \rvert$	$(0; \infty)$
Знаковая (сигнатурная)	$f(S) = \begin{cases} 1, & S > 0 \\ -1, & S \le 0 \end{cases}$	$(-1; 1)$
Квадратичная	$f(S) = S^2$	$(0; \infty)$

При уменьшении $\boldsymbol{k}$ сигмоид становится более пологим, в пределе при $\boldsymbol{k} = 0$ вырождаясь в горизонтальную линию на уровне 0,5, при увеличении $\boldsymbol{k}$ сигмоид приближается к виду функции единичного скачка с порогом 0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне (0, 1). Одно из ценных свойств сигмоидальной функции – простое выражение для ее производной:

$$f'(S) = kf(S)[1 - f(S)] \qquad (1.4)$$

Следует отметить, что сигмоидальная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют

областям аргументов, где сигмоид имеет пологий наклон.

1.2 Классификация нейронных сетей и их свойства

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

- ***входные нейроны***, на которые подается вектор, кодирующий входное воздействие или образ внешней среды; в них обычно не осуществляется вычислительных процедур, а информация передается с входа на выход путем изменения их активации;
- ***выходные нейроны***, выходные значения которых представляют выходы нейронной сети; преобразования в них осуществляются по выражениям (1.1) и (1.2);
- ***промежуточные нейроны***, составляющие основу нейронных сетей, преобразования в которых выполняются также по выражениям (1.1) и (1.2).

В большинстве нейронных моделей тип нейрона связан с его расположением в сети. Если нейрон имеет только выходные связи, то это входной нейрон, если наоборот - выходной нейрон. Однако возможен вариант, когда выход топологически внутреннего нейрона рассматривается как часть выхода сети. В процессе функционирования сети осуществляется преобразование входного вектора в выходной, т.е. происходит некоторая переработка информации. Конкретный вид выполняемого сетью преобразования данных обусловливается не только характеристиками нейроподобных элементов, но и особенностями ее архитектуры, а именно топологией межнейронных связей, выбором определенных подмножеств нейроподобных элементов для ввода и вывода информации, способами обучения сети, наличием или отсутствием конкуренции между нейронами, направлением и способами управления и синхронизации передачи информации между нейронами.

С точки зрения топологии можно выделить три основных типа нейронных сетей, внешний вид которых представлен на рисунке 7.

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

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

других слоях. В общем случае сеть состоит из ***Q*** слоев, пронумерованных слева направо. Внешние входные сигналы подаются на входы нейронов входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько скрытых слоев. Связи от выходов нейронов некоторого слоя ***q*** к входам нейронов следующего слоя (***q+1***) называются последовательными.

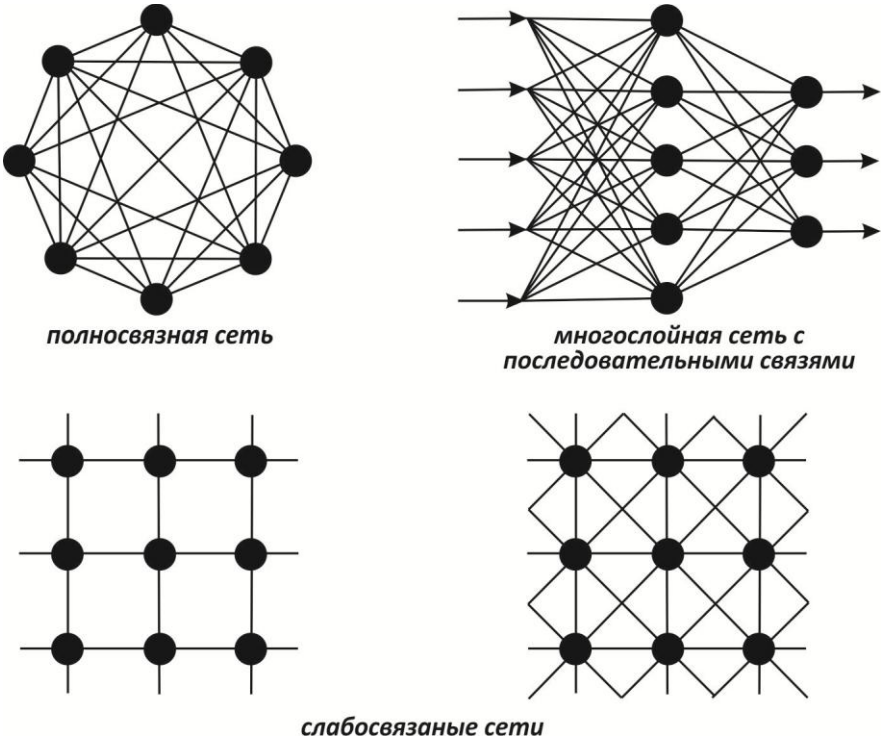


Рис. 7. Архитектуры нейронных сетей

В свою очередь, среди многослойных нейронных сетей выделяют следующие типы:

- ***Монотонные.*** Это частный случай слоистых сетей с дополнительными условиями на связи и нейроны. Каждый слой кроме последнего (выходного) разбит на два блока: *возбуждающий* и *тормозящий* (рис. 8). Связи между блоками тоже разделяются на тормозящие и возбуждающие.

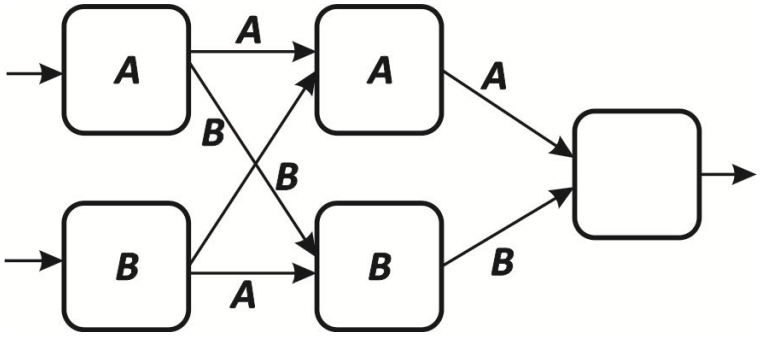


Рис. 8. Монотонная сеть

Если от нейронов блока ***A*** к нейронам блока ***B*** ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока является монотонной неубывающей функцией любого выходного сигнала блока А. Если же эти связи только тормозящие, то любой выходной сигнал блока В является невозрастающей функцией любого выходного сигнала блока А. Для нейронов монотонных сетей необходима монотонная зависимость выходного сигнала нейрона от параметров входных сигналов.

- ***Сети без обратных связей.*** В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал ***q***-го слоя подастся на вход всех нейронов (***q+1***) - го слоя; однако возможен вариант соединения ***q***-го слоя с произвольным (***q+p***)-м слоем.

  Среди многослойных сетей без обратных связей различают полносвязанные (выход каждого нейрона ***q***-го слоя связан с входом каждого нейрона (***q+1***)-го слоя) и частично полносвязанные. Классическим вариантом слоистых сетей являются полносвязанные сети прямого распространения (рис. 9).

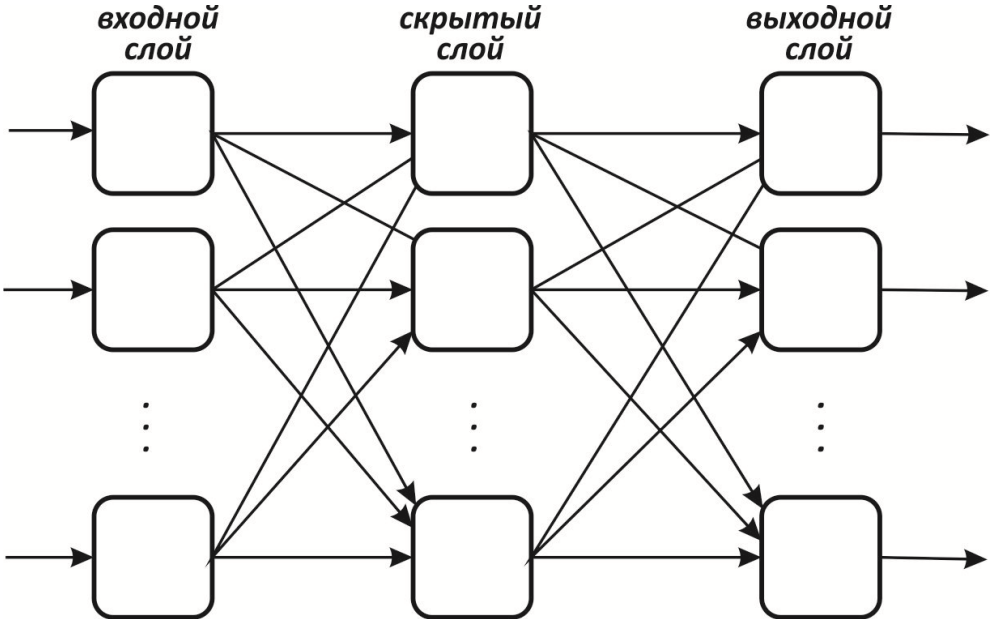


Рис. 9. Многослойная (двухслойная) сеть прямого распространения

- ***Сети с обратными связями.*** В сетях с обратными связями информация с последующих слоев передается на предыдущие слои. Среди этих сетей, в свою очередь, выделяют следующие:
  - слоисто-циклические, отличающиеся тем, что слои замкнуты в кольцо: последний слой передает свои выходные сигналы первому; все слои равноправны и могут как получать входные сигналы, так и выдавать выходные;

- слоисто-полносвязанные состоят из слоев, каждый из которых представляет собой полносвязную сеть, а сигналы передаются как от слоя к слою, так и внутри слоя; в каждом слое цикл работы распадается на три части: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача к последующему слою;
- полносвязанно-слоистые, по своей структуре аналогичные слоисто-полносвязанным, но функционирующим по-другому: в них не разделяются фазы обмена внутри слоя и передачи следующему, на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих.

В качестве примера сетей с обратными связями на рисунке 10 представлены частично-рекуррентные сети Элмана и Жордана.

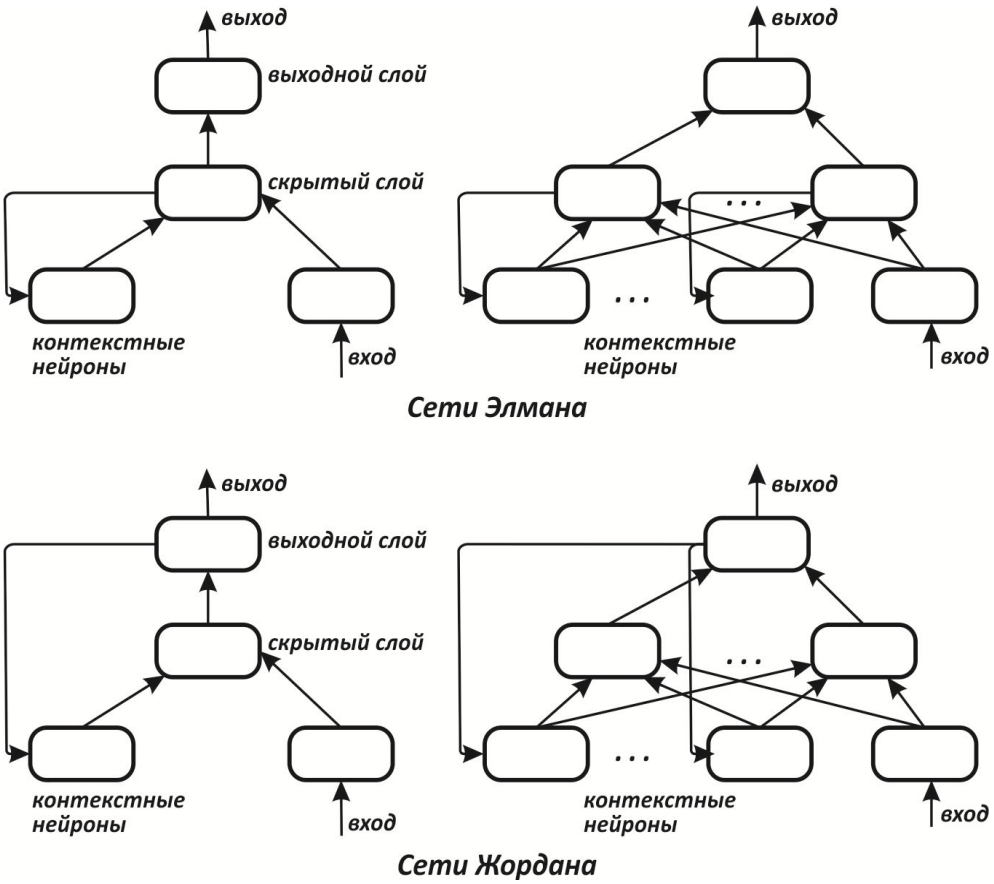


Рис. 10. Частично-рекуррентные сети Элмана и Жордана

В ***слабосвязных нейронных сетях*** нейроны располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с четырьмя (окрестность фон Неймана), шестью (окрестность Голея) или восемью (окрестность Мура) своими ближайшими соседями.

Известные нейронные сети можно разделить по типам структур нейронов на гомогенные (однородные) и гетерогенные. Гомогенные сети состоят из нейронов одного типа с единой функцией активации, а в гетерогенную сеть

входят нейроны с различными функциями активации.

Существуют бинарные и аналоговые сети. Первые из них оперируют только двоичными сигналами, и выход каждого нейрона может принимать значение либо логического ноля (заторможенное состояние) либо логической единицы (возбужденное состояние).

Сети можно классифицировать также по числу слоев. Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированных микросхем, на которых обычно реализуется нейронная сеть. Чем сложнее сеть, тем более сложные задачи она может решать.

Выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Для решения отдельных типов задач уже существуют оптимальные конфигурации. Если же задача не может быть сведена ни к одному из известных типов, приходится решать сложную проблему синтеза новой конфигурации. При этом необходимо руководствоваться следующими основными правилами:

- возможности сети возрастают с увеличением числа нейронов сети, плотности связей между ними и числом слоев;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;
- сложность алгоритмов функционирования сети, введение нескольких типов синапсов способствует усилению мощности нейронной сети.

Многие задачи распознавания образов (зрительных, речевых), выполнения функциональных преобразований при обработке сигналов, управления, прогнозирования, идентификации сложных систем, сводятся к следующей математической постановке. Необходимо построить такое отображение $X \to Y$, чтобы на каждый возможный входной сигнал X формировался правильный выходной сигнал Y. Отображение задается конечным набором пар *(<вход>, <известный выход>)*. Число этих пар (обучающих примеров) существенно меньше общего числа возможных сочетаний значений входных и выходных сигналов. Совокупность всех обучающих примеров носит название обучающей выборки.

В ***задачах распознавания образов*** X - некоторое представление образа (изображение, вектор), Y - номер класса, к которому принадлежит входной образ.

В ***задачах управления*** X - набор контролируемых параметров управляемого объекта, Y - код, определяющий управляющее воздействие, соответствующее текущим значениям контролируемых параметров.

В ***задачах прогнозирования*** в качестве входных сигналов используются временные ряды, представляющие значения контролируемых переменных на некотором интервале времени. Выходной сигнал - множество переменных, которое является подмножеством переменных входного сигнала.

При идентификации X и Y представляют входные и выходные сигналы системы соответственно. Большая часть прикладных задач может быть сведена к реализации некоторого сложного функционального многомерного преобразования. В результате отображения $X \to Y$ необходимо обеспечить формирование правильных выходных сигналов в соответствии:

- со всеми примерами обучающей выборки;
- со всеми возможными входными сигналами, которые не вошли в обучающую выборку.

Второе требование в значительной степени усложняет задачу формирования обучающей выборки. В общем виде эта задача в настоящее время еще не решена, однако во всех известных случаях может быть найдено частное решение.

1.3 Теорема Колмогорова-Арнольда - Хехт-Нильсена

Построить многомерное отображение $X \to Y$ - это значит представить его с помощью математических операций над не более, чем двумя переменными.

Проблема представления функций многих переменных в виде суперпозиции функций меньшего числа переменных восходит 13-й проблеме Гильберта (*существует ли непрерывная функция трех переменных, которая не может быть представлена через композицию непрерывных функций двух переменных*) [35]. В результате многолетней научной полемики между А. Н. Колмогоровым и В. И. Арнольдом был получен ряд важных теоретических результатов, опровергающих тезис непредставимости функции многих переменных функциями меньшего числа переменных:

- теорема о возможности представления непрерывных функций нескольких, переменных суперпозициями непрерывных функций меньшего числа переменных (1956 г.);
- теорема о представлении любой непрерывной функции трех переменных в виде суммы функций не более двух переменных (1957 г.);
- теорема о представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного сложения (1957 г.).

Теорема Хехт-Нильсена. Теорема о представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения в 1987 году была переложена Хехт-Нильсеном для нейронных сетей.

Теорема Хехт-Нильсена доказывает представимость функции многих переменных достаточно общего вида с помощью двухслойной нейронной сети с прямыми полными связями с n нейронами входного слоя, ($2n+1$) нейронами скрытого слоя с заранее известными ограниченными функциями активации (например, сигмоидальными) и m нейронами выходного слоя с неизвестными

функциями активации. Теорема, таким образом, в неконструктивной форме доказывает решаемость задачи представления функции произвольного вида на нейронной сети и указывает для каждой задачи минимальные числа нейронов сети, необходимых для ее решения.

Следствие 1.1. Из теоремы Хехт-Нильсена следует представляемость любой многомерной функции нескольких переменных с помощью нейронной сети фиксированной размерности. Неизвестными остаются следующие характеристики функций активации нейронов:

- ограничения области значений (координаты асимптот) сигмоидальных функций активации нейронов скрытого слоя;
- наклон сигмоидальных функций активации;
- вид функций активации нейронов выходного слоя.

Про функции активации нейронов второго слоя из теоремы Хехт-Нильсена известно только то, что они - нелинейные функции общего вида монотонно возрастающие. Это в некоторой степени сужает класс функций, которые могут потребоваться при реализации отображения с помощью двухслойной нейронной сети.

На практике требования теоремы Хехт-Нильсена к функциям активации удовлетворяют следующим образом. В нейронных сетях как для первого, так и для второго слоя используют сигмоидальные передаточные функции с настраиваемыми параметрами [35]. То есть, в процессе обучения индивидуально для каждого нейрона настраиваются следующие параметры функций активации:

- максимальное и минимальное значение функции,
- наклон сигмоидальной функции.

Следствие 1.2. Для любого множества пар (X_k, Y_k), (где Y_k - скаляр) существует двухслойная однородная (с одинаковыми функциями активации) нейронная сеть первого порядка с последовательными связями и с конечным числом нейронов, которая выполняет отображение $X \to Y$, выдавая на каждый входной сигнал X_k правильный выходной сигнал Y_k. Нейроны в такой двухслойной нейронной сети должны иметь сигмоидальные передаточные функции.

К сожалению, эта теорема не конструктивна. В ней не заложена методика определения числа нейронов в сети для некоторой конкретной обучающей выборки.

Для многих задач единичной размерности выходного сигнала недостаточно. Необходимо иметь возможность строить с помощью нейронных сетей функции $X \to Y$, где Y имеет произвольную размерность. Следующее утверждение является теоретической основой для построения таких функций на базе однородных нейронных сетей.

Утверждение 1.1. Для любого множества пар входных-выходных

векторов произвольной размерности $\{(X_k,\ Y_k),\ k = 1 \ldots N\}$ существует однородная двухслойная нейронная сеть с последовательными связями, с сигмоидальными передаточными функциями и с конечным числом нейронов, которая для каждого входного вектора X_k формирует соответствующий ему выходной вектор Y_k.

Таким образом, для представления многомерных функций многих переменных может быть использована однородная двухслойная нейронная сеть с сигмоидальными передаточными функциями.

Для оценки числа нейронов в скрытых слоях однородных нейронных сетей можно воспользоваться формулой для оценки необходимого числа синаптических весов L_w в многослойной сети с сигмоидальными передаточными функциями:

$$\frac{mN}{1 + \log_2 N} \leq L_w \leq m\left(\frac{N}{m} + 1\right)(n + m + 1) + m \qquad (1.5)$$

где $\boldsymbol{n}$ - размерность входного сигнала, $\boldsymbol{m}$ - размерность выходили сигнала, $\boldsymbol{N}$ - число элементов обучающей выборки.

Оценив необходимое число весов, можно рассчитать число нейронов в скрытых слоях. Например, для двухслойной сети это число составит:

$$L = \frac{L_w}{n + m}.$$

Известны и другие формулы для оценки, например:

$$2(n + L + m) < N < 10(n + L + m),$$

$$\frac{N}{10} - n - m < L < \frac{N}{2} - n - m.$$

Точно так же можно рассчитать число нейронов в сетях с большим числом слоев.

Иногда целесообразно использовать сети с большим числом слоев. Такие многослойные нейронные сети могут иметь меньшие размерности матриц синаптических весов нейронов одного слоя, чем двухслойные сети, реализующие то же самое отображение. Однако строгой методики построения таких сетей пока нет [25, 35].

Аналогичная ситуация складывается и с многослойными нейронными сетями, в которых помимо последовательных связей используются и прямые (связи от слоя с номером $\boldsymbol{q}$ к слою с номером ($\boldsymbol{q+p}$), где $\boldsymbol{p > 1}$). Нет строгой теории, которая показывала бы возможность и целесообразность построения таких сетей. Наибольшие проблемы возникают при использовании сетей циклического функционирования. К этой группе относятся многослойные сети с обратными связями (от слоя с номером $\boldsymbol{q}$ к слою с номером ($\boldsymbol{q+p}$), где $\boldsymbol{p < 0}$), а также полносвязные сети. Для успешного функционирования таких сетей необходимо соблюдение условий динамической устойчивости, иначе сеть может не сойтись к правильному решению, либо, достигнув на некоторой

итерации правильного значения выходного сигнала, после нескольких итераций уйти от этого значения. Проблема динамической устойчивости подробно исследована, пожалуй, лишь для одной модели из рассматриваемой группы - ***нейронной сети Хопфилда***. Отсутствие строгой теории для перечисленных моделей нейронных сетей не препятствует исследованию возможностей их применения.

1.4 Обучение нейронных сетей

Самым важным свойством нейронных сетей является их способность обучаться на основе данных окружающей среды и в результате обучения повышать свою производительность. Обучение нейронной сети происходит посредством интерактивного процесса корректировки синаптических весов и порогов (рис. 11). В идеальном случае нейронная сеть получает знания об окружающей среде на каждой итерации процесса обучения.

С понятием обучения ассоциируется довольно много видов деятельности, поэтому сложно дать этому процессу однозначное определение. Более того, процесс обучения зависит от точки зрения на него. Именно это делает практически невозможным появление какого-либо точного определения этого понятия. Например, процесс обучения с точки зрения психолога в корне отличается от обучения с точки зрения школьного учителя. С позиций нейронной сети, вероятно, можно использовать следующее определение:

- ***обучение*** – это процесс, в котором свободные параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть встроена. Тип обучения определяется способом подстройки этих параметров.

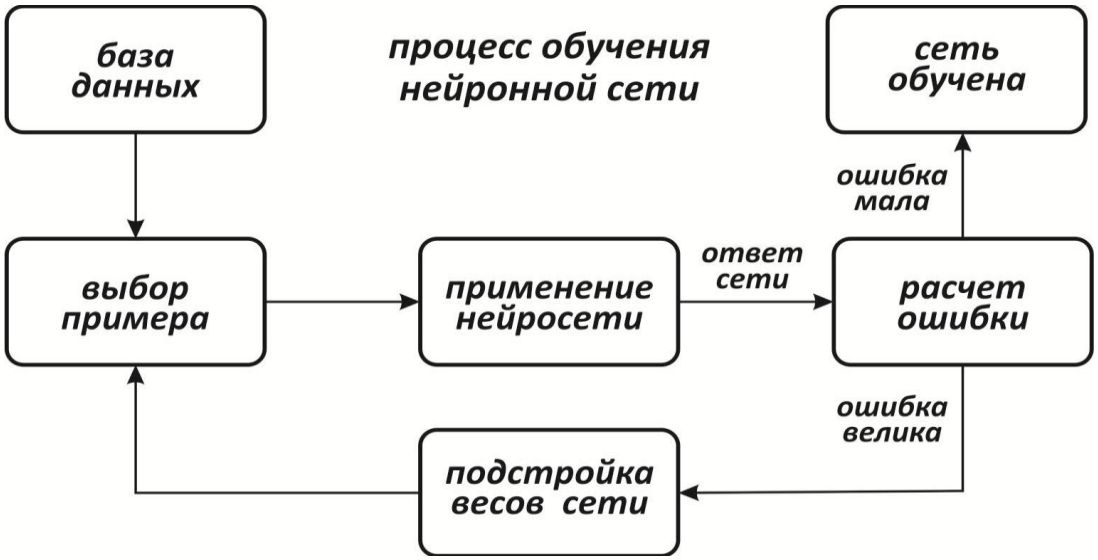


Рис. 11. Процесс обучения нейронной сети

При обучении нейронной сети мы действуем следующим образом - у нас

имеется некоторая база данных, содержащая примеры (*набор рукописных изображений букв*). Предъявляя изображение буквы «***А***» на вход нейронной сети, мы получаем от нее некоторый ответ, не обязательно верный. Нам известен и верный (желаемый) ответ - в данном случае нам хотелось бы, чтобы на выходе нейронной сети с меткой «***А***» уровень сигнала был максимален. Обычно в качестве желаемого выхода в задаче классификации берут набор (***1, 0, 0, ...***), где ***1*** стоит на выходе с меткой «***А***», а ***0*** - на всех остальных выходах. Вычисляя разность между желаемым ответом и реальным ответом сети, мы получаем ***33*** числа - *вектор ошибки*.

Алгоритм обратного распространения ошибки - это набор формул, который позволяет по вектору ошибки вычислить требуемые поправки для весов нейронной сети. Одну и ту же букву (а также различные изображения одной и той же буквы) мы можем предъявлять нейронной сети много раз. В этом смысле обучение скорее напоминает тренировку.

Оказывается, что после многократного предъявления примеров веса нейронной сети стабилизируются, причем нейронная сеть дает правильные ответы на все (или почти все) примеры из базы данных. В таком случае говорят, что ***«нейронная сеть обучена»***, или ***«нейронная сеть натренирована»***.

В программных реализациях можно видеть, что в процессе обучения величина ошибки (сумма квадратов ошибок по всем выходам) постепенно уменьшается. Когда величина ошибки достигает нуля или приемлемого малого уровня, тренировку останавливают, а полученную нейронную сеть считают ***натренированной*** и готовой к применению на новых данных.

Важно отметить, что вся информация, которую нейронная сеть имеет о задаче, содержится в наборе примеров. Поэтому качество обучения нейронной сети напрямую зависит от количества примеров в обучающей выборке, а также от того, насколько полно эти примеры описывают данную задачу. Считается, что для полноценной тренировки нейронной сети требуется хотя бы несколько десятков (а лучше сотен) примеров.

Математически процесс обучения можно описать следующим образом.

В процессе функционирования нейронная сеть формирует выходной сигнал Y в соответствии с входным сигналом X, реализуя некоторую функцию $Y = G(X)$. Если архитектура сети задана, то вид функции G определяется значениями синаптических весов и смещений сети.

Пусть решением некоторой задачи является функция $Y = F(X)$, заданная парами входных - выходных данных $(X_1, Y_1), (X_2, Y_2), \dots, (X_N, Y_N)$, для которых $Y_k = F(X_k)$ $(k = 1, 2, \dots\dots N)$.

Обучение состоит в поиске (синтезе) функции G, близкой к F в смысле некоторой функции ошибки E (см. рис. 11).

Если выбраны множество обучающих примеров — пар (X_k, Y_k) где $(k = 1, 2, \dots\dots N)$ и способ вычисления функции ошибки E, то обучение нейронной сети превращается в задачу многомерной оптимизации, имеющую очень большую размерность, при этом, поскольку функция E может иметь

произвольный вид, обучение в общем случае — многоэкстремальная невыпуклая задача оптимизации.

Для решения этой задачи могут быть использованы следующие (итерационные) алгоритмы:

- алгоритмы локальной оптимизации с вычислением частных производных первого порядка;
- алгоритмы локальной оптимизации с вычислением частных производных первого и второго порядка;
- стохастические алгоритмы оптимизации;
- алгоритмы глобальной оптимизации.

К первой группе относятся: градиентный алгоритм (метод скорейшего спуска); методы с одномерной и двумерной оптимизацией целевой функции в направлении антиградиента; метод сопряженных градиентов; методы, учитывающие направление антиградиента на нескольких шагах алгоритма.

Ко второй группе относятся: метод Ньютона, методы оптимизации с разреженными матрицами Гессе, квазиньютоновские методы, метод Гаусса-Ньютона, метод Левенберга-Марквардта и др.

Стохастическими методами являются: поиск в случайном направлении, имитация отжига, метод Монте-Карло (численный метод статистических испытаний).

Задачи глобальной оптимизации решаются с помощью перебора значений переменных, от которых зависит целевая функция (функция ошибки E).

Применение нейронной сети. После того как сеть обучена, ее можно применять для решения различных задач (рис.12).

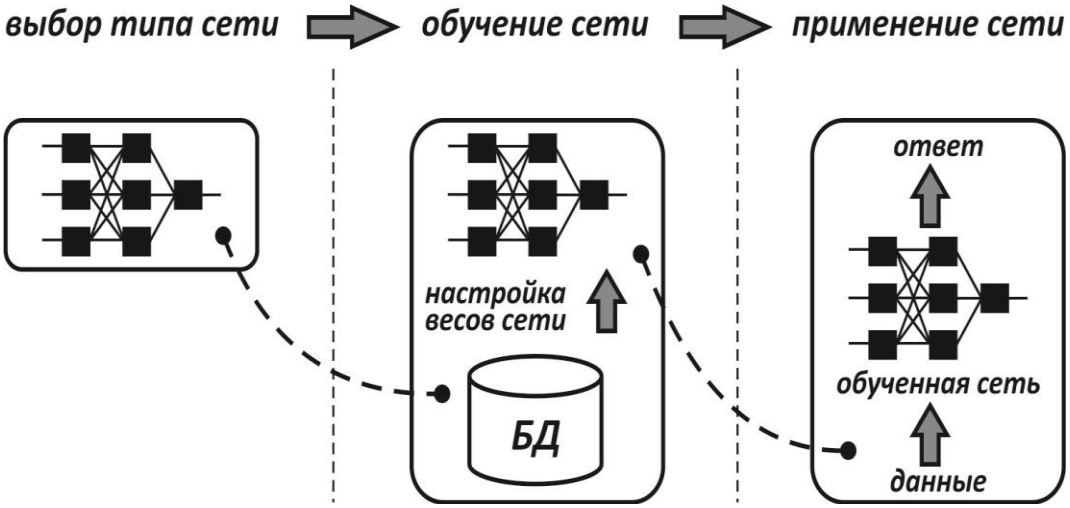


Рис. 12. Этапы нейросетевого проекта

Важнейшая особенность человеческого мозга состоит в том, что, однажды обучившись определенному процессу, он может верно, действовать и в тех ситуациях, в которых он не бывал в процессе обучения. Например, есть возможность, читать почти любой почерк, даже если видим его первый раз в

жизни. Так же и нейронная сеть, грамотным образом обученная, может с большой вероятностью правильно реагировать на новые, не предъявленные ей ранее данные. Например, мы можем нарисовать букву ***«А»*** другим почерком, а затем предложить нашей сети классифицировать новое изображение.

Веса обученной сети хранят достаточно много информации о сходстве и различиях букв, поэтому можно рассчитывать на правильный ответ и для нового варианта изображения.

Области применения нейронных сетей: классификация, прогнозирование. Отметим, что задачи классификации (типа распознавания букв) очень плохо алгоритмизуются. Если в случае распознавания букв верный ответ очевиден для нас заранее, то в более сложных практических задачах обученная нейросеть выступает как эксперт, обладающий большим опытом и способный дать ответ на трудный вопрос.

Примером такой задачи служит медицинская диагностика, где сеть может учитывать большое количество числовых параметров (энцефалограмма, давление, вес и т.д.). Конечно, «мнение» сети в этом случае нельзя считать окончательным. Классификация предприятий по степени их перспективности — это уже привычный способ использования нейронных сетей в практике западных компаний (деление компаний на перспективные и убыточные). При этом сеть также использует множество экономических показателей, сложным образом связанных между собой.

Нейросетевой подход особенно эффективен в задачах экспертной оценки медицинской диагностики по той причине, что он сочетает в себе способность компьютера к обработке чисел и способность мозга к обобщению и распознаванию. Говорят, что у хорошего врача способность к распознаванию в своей области столь велика, что он может провести приблизительную диагностику уже по внешнему виду пациента.

Можно согласиться также, что опытный трейдер чувствует направление движения рынка по виду графика. Однако в первом случае все факторы наглядны, т.е. характеристики пациента мгновенно воспринимаются мозгом как «бледное лицо», «блеск в глазах» и т.д. Во втором же случае учитывается только один фактор, показанный на графике, — курс за определенный период времени. Нейронная сеть позволяет обрабатывать огромное количество факторов (до нескольких тысяч), независимо от их наглядности, — это универсальный «хороший врач», который может поставить свой диагноз в любой области.

Задачи прогнозирования. Прогнозирование - это ключевой момент при принятии решений в управлении. Конечная эффективность любого решения зависит от последовательности событий, возникающих уже после принятия решения. Возможность предсказать неуправляемые аспекты этих событий перед принятием решения позволяет сделать наилучший выбор, который, в противном случае, мог бы быть не таким удачным. Поэтому системы планирования и управления, обычно, реализуют функцию прогноза.

1.5 Типы нейронных сетей

1.5.1 Персептрон Розенблата

В середине 1958 года Фрэнк Розенблат предложил модель электронного устройства, названного им ***персептроном***, которое должно было бы имитировать процессы человеческого мышления. Персептрон должен был передавать сигналы от «глаза», составленного из фотоэлементов, в блоки электромеханических ячеек памяти, которые оценивали относительную величину электрических сигналов. Эти ячейки соединялись между собой случайным образом в соответствии с принципами коннективизма. Два года спустя была продемонстрирована первая действующая машина **«Марк-1»**, которая могла научиться распознавать некоторые из букв, написанных на карточках, которые подносили к его «глазам», напоминающие кинокамеры.

У. Маккалокк в работе [75] исследовал сетевые парадигмы для распознавания изображений, подвергаемых сдвигам и поворотам. Простая нейронная модель, показанная на рисунке 13, использовалась в большей части его работы. Элемент Σ умножает каждый вход x_n на вес w_n и суммирует взвешенные входы. Если эта сумма больше заданного порогового значения, выход равен ***единице***, в противном случае - ***нулю***. Эти системы - *персептроны*. Они состоят из одного слоя искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов (рис. 14), хотя в принципе описываются и более сложные системы.

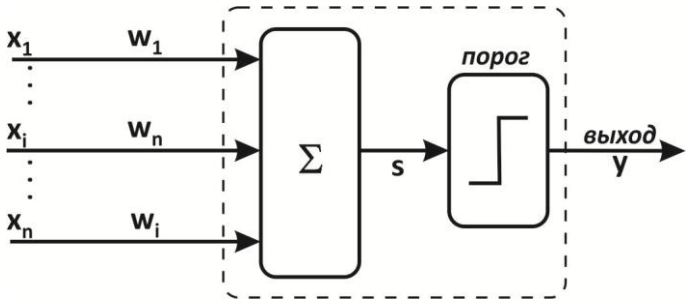


Рис. 13. Персептронный нейрон

В 60-е годы Розенблатт [77] доказал замечательную теорему об обучении персептронов. Уидроу [81, 82, 84] дал ряд убедительных демонстраций систем персептронного типа, и исследователи во всем мире стремились изучить возможности этих систем. Однако существует ряд задач, которые персептроны не способны выполнить. Минский [69] определил, что имеются жесткие ограничения на то, что могут выполнять однослойные персептроны, и, следовательно, на то, чему они могут обучаться.

Теория персептронов является основой для многих других типов искусственных нейронных сетей, и персептроны иллюстрируют важные принципы. В силу этих причин они являются логической исходной точкой для изучения искусственных нейронных сетей.

Рассмотрим в качестве примера трехслойный персептрон (при условии, что $n = 3$) (рис. 14), нейроны которого имеют функцию активации заданную в виде единичного скачка (порога).

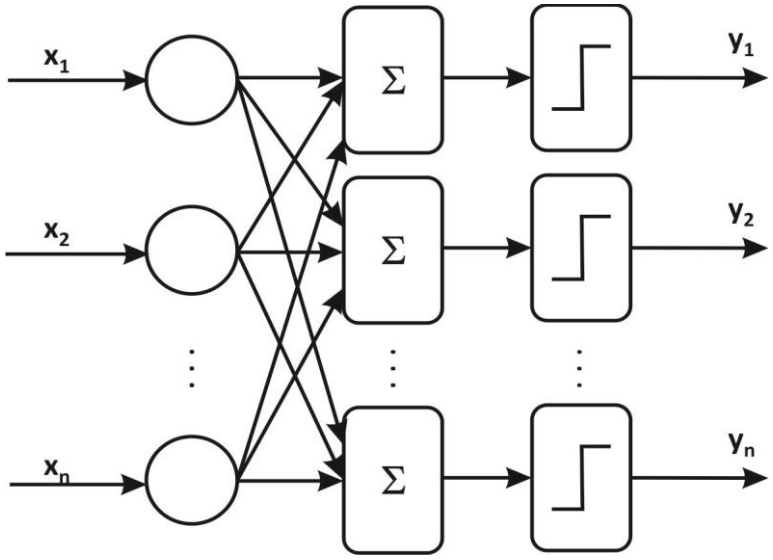


Рис. 14. Персептрон со многими выходами

Пусть на n входов поступают входные сигналы, проходящие по синапсам на три нейрона, образующие единственный слой этой сети и выдающие три входных сигнала:

$$y_j = f\left(\sum_{i=1}^{n} x_i w_{ij}\right), \ j = 1 \ldots 3.$$

Очевидно, что все весовые коэффициенты синапсов одного слоя нейронов можно свести в матрицу W, в которой каждый элемент w_{ij} задает величину i-й синаптической связи j-го нейрона. Таким образом, процесс, происходящий в нейронной сети, может быть записан в матричной форме:

$$Y = f(X\,W),$$

где X и Y - соответственно входной и выходной сигнальные векторы, $f(S)$ – функция активации, применяемая поэлементно к компонентам вектора S.

На рисунке 15 представлен двухслойный персептрон, полученный из персептрона, представленного на рисунке 14, путем добавления второго слоя, состоящего из двух нейронов.

Здесь уместно отметить важную роль нелинейности активационной функции. Если бы она не обладала данным свойством или не входила в алгоритм работы каждого нейрона, результат функционирования любой Q – слойной нейронной сети с весовыми матрицами $W^{(q)}$ для каждого слоя $q = 1 \ldots Q$ сводился бы к перемножению входного вектора сигналов X на матрицу:

$$W_\Sigma = W^{(1)} \ldots W^{(G)} \ldots W^{(Q)}.$$

Фактически такая Q – слойная нейронная сеть эквивалентна сети с одним скрытым слоем и с весовой матрицей единственного слоя W_{Σ}:

$$Y = X\, W_{(\Sigma)}$$

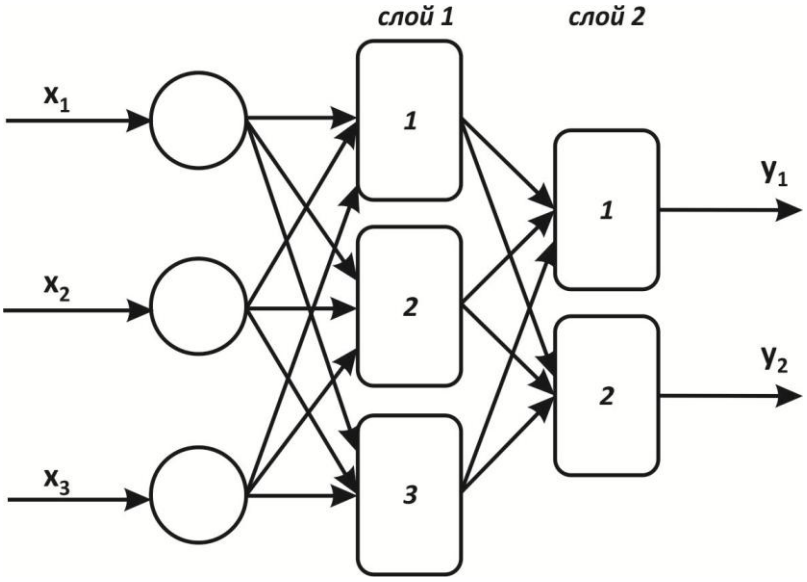


Рис. 15. Двухслойный персептрон

Работа персептрона сводится к классификации (обобщению) входных сигналов, принадлежащих n – мерному гиперпространству, по некоторому числу классов. С математической точки зрения это происходит путем разбиения гиперпространства гиперплоскостями. Для случая однослойного персептрона:

$$\sum_{i=1}^{n} x_i w_{ij} = \Theta_j, \quad j = 1,2,\dots,n.$$

Каждая полученная область является областью определения отдельного класса. Число таких классов для персептрона не превышает $2n$, где n – число его входов. Однако не все из классов могут быть выделены данной нейронной сетью.

Например, однослойный персептрон, состоящий из одного нейрона с двумя входами (рис. 16), не может реализовать логическую функцию ***«исключающее ИЛИ»***. Это функция от двух аргументов, каждый из которых может быть нулем или единицей. Она принимает значение единицы, когда один из аргументов равен единице (но не оба). Обозначим один вход через x_1, а другой через x_2, тогда все их возможные комбинации будут состоять из четырех точек на плоскости x_1- x_2, как показано на рисунке 17. Например, точка $x_1 = 0$ и $x_2 = 0$ обозначена на рисунке как точка ***A***. Таблица 1.2 показывает требуемую связь между входами и выходом, где входные комбинации, которые должны давать нулевой выход, помечены $\boldsymbol{A_0}$ и $\boldsymbol{A_1}$, единичный выход - $\boldsymbol{B_0}$ и $\boldsymbol{B_1}$.

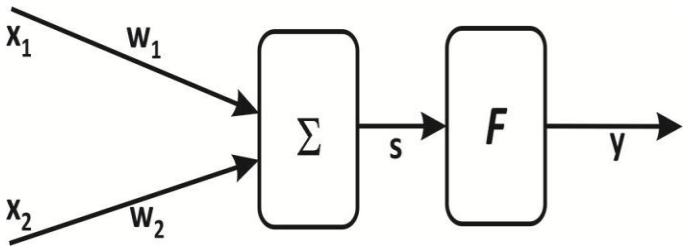


Рис. 16. Однонейронная система

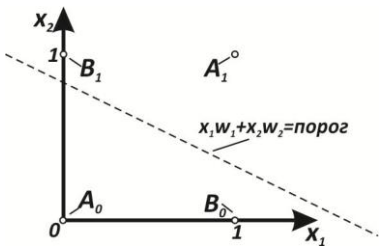


Рис. 17. *«Исключающее ИЛИ»*

В сети на рис. 16 функция F является обычным порогом, так что S принимает значение ноль, когда S меньше 0,5, и единица в случае, когда S больше или равно 0,5. Нейрон выполняет следующее вычисление:

$$S = x_1 w_1 + x_2 w_2 \qquad (1.6)$$

Никакая комбинация значений двух весов не может дать соотношения между входом и выходом, задаваемого таблице 1.2. Чтобы понять это ограничение, зафиксируем S на величине порога 0,5. Сеть в этом случае описывается уравнением (1.7). Это уравнение линейно по x_1 и x_2, т. е. все значения по x_1 и x_2, удовлетворяющие этому уравнению, будут лежать на некоторой прямой в плоскости x_1 - x_2.

$$x_1 w_1 + x_2 w_2 = 0{,}5 \qquad (1.7)$$

Таблица 1.2.

Таблица истинности для функции *«исключающее ИЛИ»*

Точки	Значения x_1	Значения x_2	Требуемый выход
A_0	0	0	0
B_0	1	0	1
A_1	0	1	1
B_1	1	1	0

Любые входные значения для x_1 и x_2 на этой линии будут давать пороговое значение 0,5 для S. Входные значения с одной стороны прямой обеспечат значения S больше порога, следовательно, Y=1. Входные значения по другую сторону прямой обеспечат значения S меньше порогового значения, делая Y равным 0. Изменения значений w_1, w_2 и порога будут менять наклон и положение прямой. Для того чтобы сеть реализовала функцию *«исключающее ИЛИ»*, заданную табл. 1.2, нужно расположить прямую так, чтобы точки А были с одной стороны прямой, а точки В - с другой. Попытавшись нарисовать такую прямую на рисунке 17, убеждаемся, что это невозможно. Это означает, что какие бы значения ни приписывались весам и порогу, сеть неспособна воспроизвести соотношение между входом и выходом, требуемое для представления функции *«исключающее ИЛИ»*.

Взглянув на задачу с другой точки зрения, рассмотрим S как поверхность над плоскостью x_1 - x_2. Каждая точка этой поверхности находится над

соответствующей точкой плоскости x_1 - x_2 на расстоянии, равном значению S в этой точке. Можно показать, что наклон этой S - поверхности одинаков для всей поверхности x_1 - x_2. Все точки, в которых значение S равно величине порога, проектируются на линию уровня плоскости S (рис. 18).

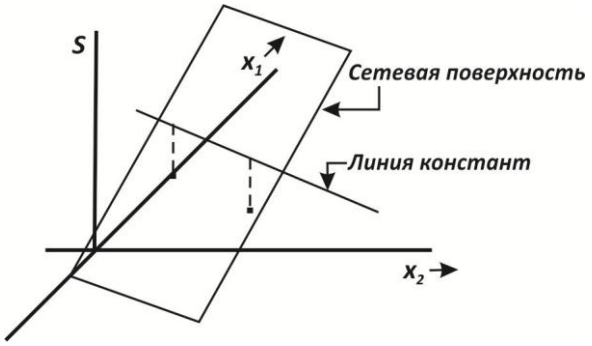


Рис. 18. Персептронная S – плоскость

Ясно, что все точки по одну сторону пороговой прямой проецируются в значения S, большие порога, а точки по другую сторону дадут меньшие значения S. Таким образом, пороговая прямая разбивает плоскость x_1 - x_2 на две области. Во всех точках по одну сторону пороговой прямой значение Y равно единице, по другую сторону - нулю.

Проблема *«исключающее ИЛИ»*, к сожалению, не единственная. Имеется обширный класс функций, не реализуемых однослойной сетью. Об этих функциях говорят, что они являются ***линейно неразделимыми***, и они накладывают определенные ограничения на возможности однослойных сетей.

Линейная разделимость ограничивает однослойные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. Для нашего случая с двумя входами разделитель является прямой линией. В случае трех входов разделение осуществляется плоскостью, рассекающей трехмерное пространство. Для четырех или более входов визуализация невозможна и необходимо мысленно представить n - мерное пространство, рассекаемое «гиперплоскостью» - геометрическим объектом, который рассекает пространство четырех или большего числа измерений.

Так как линейная разделимость ограничивает возможности персептронного представления, то важно знать, является ли данная функция разделимой. Нейрон с n двоичными входами может иметь 2^n различных входных образов, состоящих из нулей и единиц. Так как каждый входной образ может соответствовать двум различным бинарным выходам (единица и ноль), то всего имеется 2^{2n} функций от n переменных.

В таблице 1.3 показана вероятность того, что случайно выбранная функция окажется линейно разделимой, весьма мала даже для умеренного числа переменных. По этой причине однослойные персептроны на практике ограничены простыми задачами.

Таблица 1.3.

Линейно разделимые функции [76]

n переменных	2^{2n} функций	Число линейно разделимых функций
1	4	4
2	6	14
3	256	104
4	65536	1882
5	$4{,}3 \times 10^{9}$	94572
6	$1{,}8 \times 10^{19}$	15028134

К концу 60-х годов стало известно, что это серьезное ограничение представляемости однослойными сетями можно преодолеть, добавив дополнительные слои. Например, двухслойные сети можно получить каскадным соединением двух однослойных сетей. Они способны выполнять более общие классификации, отделяя те точки, которые содержатся в выпуклых ограниченных или неограниченных областях. Область называется выпуклой, если для любых двух ее точек соединяющий их отрезок целиком лежит в области. Область называется ограниченной, если ее можно заключить в некоторую фигуру. Неограниченную область невозможно заключить внутрь фигуры (например, область между двумя параллельными линиями). Примеры выпуклых ограниченных и неограниченных областей показаны на рисунке 19.

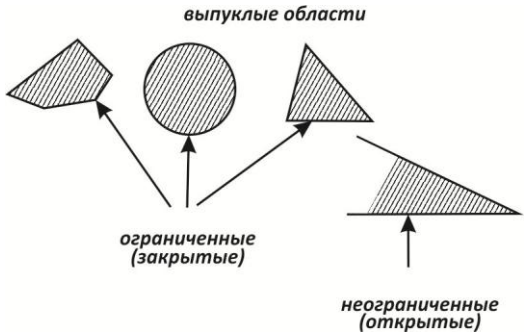


Рис. 19. Выпуклые ограниченные и неограниченные области

Чтобы уточнить требование выпуклости, рассмотрим простую двухслойную сеть с двумя входами, подведенными к двум нейронам первого слоя, соединенными с единственным нейроном в слое 2 (рис. 20). Пусть порог выходного нейрона равен 0,75, а оба его веса равны 0,5. В этом случае для того, чтобы порог был превышен и на выходе появилась единица, требуется, чтобы оба нейрона первого уровня на выходе имели единицу. Таким образом, выходной нейрон реализует логическую функцию *«И»*. На рисунке 20 каждый нейрон *слоя 1* разбивает плоскость x_1 - x_2 на две полуплоскости, один обеспечивает единичный выход для входов ниже верхней линии, другой - для входов выше нижней линии. На рисунке 20 показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V - образной области. Аналогично во втором слое может быть использовано три нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник любой желаемой формы. Так как они образованы с помощью

функции *«И»* над областями, задаваемыми линиями, то все такие многогранники выпуклы, следовательно, только выпуклые области и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

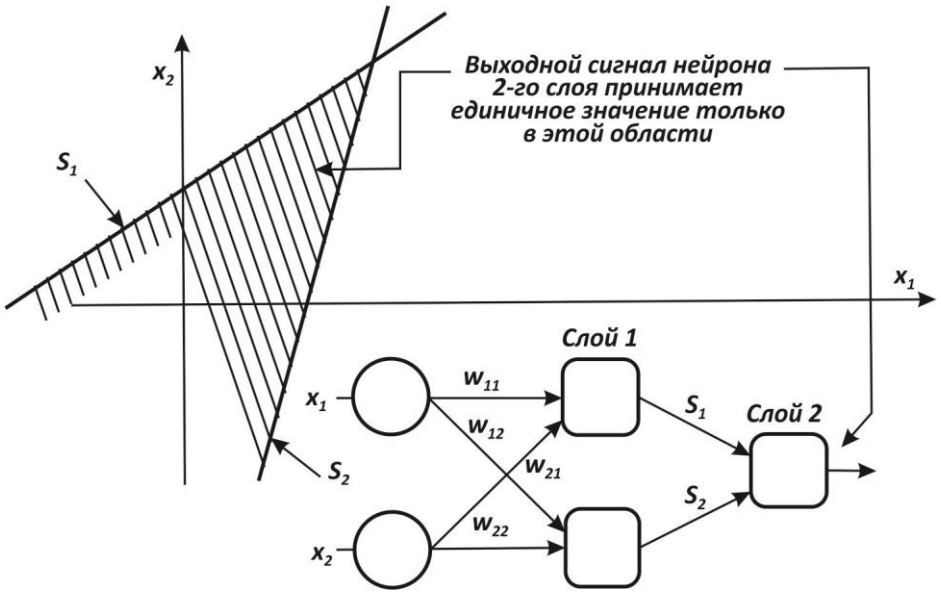


Рис. 20. Выпуклая область решений, задаваемая двухслойной сетью

Нейрон второго слоя не ограничен функцией *«И»*. Он может реализовывать многие другие функции при подходящем выборе весов и порога. Например, можно сделать так, чтобы единичный выход любого из нейронов первого слоя приводил к появлению единицы на выходе нейрона второго слоя, реализовав тем самым логическое *«ИЛИ»*. Имеется 16 двоичных функций от двух переменных. Если выбирать подходящим образом веса и порог, то можно воспроизвести 14 из них (все функции, кроме *«исключающее ИЛИ»* и *«исключающее НЕТ»*).

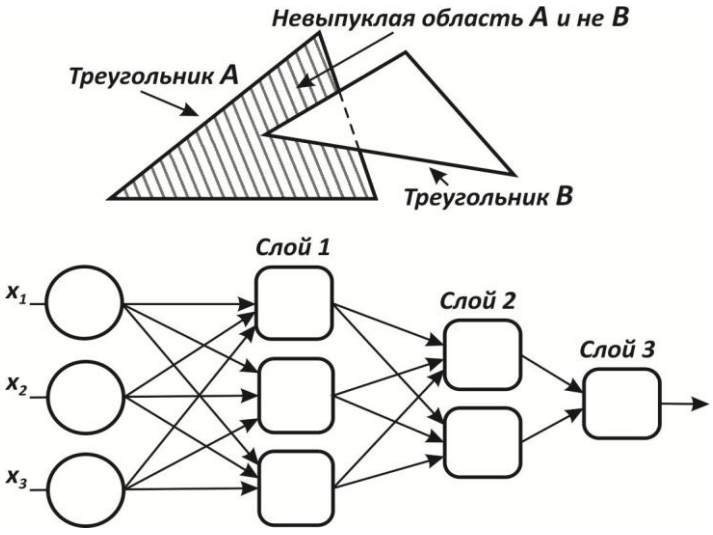


Рис. 21. «Вогнутая» область решений, задаваемая трехслойной сетью

Входы не обязательно должны быть двоичными. Вектор непрерывных входов может представлять собой произвольную точку на плоскости x_1 - x_2. В этом случае мы имеем дело со способностью сети разбивать плоскость на непрерывные области, а не с разделением дискретных множеств точек. Для всех этих функций, однако, линейная разделимость показывает, что выход нейрона второго слоя равен единице только в части плоскости x_1 - x_2, ограниченной многоугольной областью.

Трехслойная сеть, однако, является более общей. Ее классифицирующие возможности ограничены лишь числом искусственных нейронов и весов. Ограничения на выпуклость отсутствуют. Теперь нейрон третьего слоя принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть невыпуклой. На рисунке 21 иллюстрируется случай, когда два треугольника *A* и *B*, скомбинированные с помощью функций *«A и не B»,* задают невыпуклую область. При добавлении нейронов и весов число сторон многоугольников может неограниченно возрастать. Это позволяет аппроксимировать область любой формы с любой точностью. Вдобавок не все выходные области второго слоя должны пересекаться. Возможно, следовательно, объединять различные области, выпуклые и невыпуклые, выдавая на выходе единицу, всякий раз, когда входной вектор принадлежит одной из них.

Алгоритм обучения персептрона. Способность искусственных нейронных сетей обучаться является их наиболее интригующим свойством. Подобно биологическим системам, которые они моделируют, эти нейронные сети сами моделируют себя в результате попыток достичь лучшей модели поведения.

Используя критерий линейной разделимости, можно решить, способна ли однослойная нейронная сеть реализовывать требуемую функцию. Даже в том случае, когда ответ положительный, это принесет мало пользы, если у нас нет способа найти нужные значения для весов и порогов. Чтобы сеть представляла практическую ценность, нужен систематический метод (алгоритм) для вычисления этих значений. Розенблатт [77] сделал это в своем алгоритме обучения персептрона вместе с доказательством того, что персептрон может быть обучен всему, что он может реализовывать.

Обучение может быть с учителем или без него. Для обучения с учителем нужен *«внешний»* учитель, который оценивал бы поведение системы и управлял ее последующими модификациями. При обучении без учителя, рассматриваемого в последующих главах, сеть путем самоорганизации делает требуемые изменения. Обучение персептрона является обучением с учителем.

Алгоритм обучения персептрона может быть реализован на цифровом компьютере или другом электронном устройстве, и сеть становится в определенном смысле самоподстраивающейся. По этой причине процедуру подстройки весов обычно называют *«обучением»* и говорят, что сеть *«обучается»*.

Обучение персептрона сводится к формированию весов связей между первым и вторым (рис. 15) слоями в соответствии со следующим алгоритмом:

Шаг 1. Проинициализировать элементы весовой матрицы (обычно небольшими случайными значениями).

Шаг 2. Подать на входы один из входных векторов, которые сеть должна научиться различать, и вычислить ее выход.

Важное обобщение алгоритма обучения персептрона, называемое ***дельта-правилом***, переносит этот метод на непрерывные входы и выходы. Чтобы понять, как оно было получено, шаг 3 алгоритма обучения персептрона может быть сформулирован в обобщенной форме с помощью введения величины δ, которая равна разности между требуемым или целевым выходом d и реальным выходом Y.

Шаг 3. Если выход правильный, перейти на *шаг 4*. Иначе – вычислить разницу между целевым выходом d и полученным Y значениями выхода:

$$\delta = d - Y. \tag{1.8}$$

Модифицировать весовые коэффициенты в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta x_i \tag{1.9}$$

где t и $(t+1)$ - номера соответственно текущей и следующей итерации; η – коэффициент скорости обучения, лежащий в интервале $0 < \eta < 1$; i – номер входа; j – номер нейрона в слое.

Очевидно, что если $d > Y$, то весовые коэффициенты будут увеличены и, тем самым, уменьшат ошибку (разница между целевым выходом d и полученным Y). В противном случае они будут уменьшены, и Y тоже уменьшится, приближаясь к d.

Шаг 4. Организуется цикл (повтор операций) с *шага 2*, до тех пор, пока сеть не перестанет ошибаться.

На втором шаге на разных итерациях поочередно в случайном порядке предъявляются все возможные входные векторы. Т.к., нельзя заранее определить число итераций, которые потребуется выполнить, а в некоторых случаях невозможно гарантировать полный успех.

Сходимость рассмотренной процедуры устанавливается теоремами, утверждающими, что:

- для любой классификации обучающей последовательности можно подобрать такой набор (из бесконечного набора) элементарных нейронов, в котором будет осуществлено разделение обучающей выборки при помощи линейного обучающего правила;
- если относительно задуманной классификации можно найти набор элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто за конечный промежуток времени.

Свойства однослойного персептрона:

- сеть обучается с учителем, т. е. для каждого входного обучающего вектора на выходе персептрона указывается требуемое значение;
- оптимистическая оценка количества L нейронов персептрона: $L > log_2(K)$, где K — количество классов векторов в обучающей выборке;
- сеть обладает свойством обобщения — любой входной вектор, принадлежит он обучающей выборке или нет, попадает в некоторую решающую область и дает ответ, соответствующий данной области. При этом входные векторы, находящиеся внутри той или иной решающей области, на самом деле могут быть достаточно удаленными друг от друга;
- существуют функции, которые однослойный персептрон не может аппроксимировать, так называемая проблема *«исключающее ИЛИ»*.

1.5.2 Рекуррентные нейронные сети. Сети Джордана и Элмана

Рекуррентные сети представляют собой развитие однонаправленных сетей персептронного типа за счет добавления в них соответствующих обратных связей. Обратная связь может исходить либо из выходного, либо из скрытого слоя нейронов. В каждом контуре такой связи присутствует элемент единичной задержки, передающий значение на так называемый контекстный нейрон, благодаря которому поток сигналов может считаться однонаправленным (выходной сигнал предыдущего временного цикла рассматривается как априори заданный, который просто увеличивает размерность входного вектора x сети, добавляя контекстные нейроны). Представленная подобным образом рекуррентная сеть с учетом способа формирования выходного сигнала функционирует как однонаправленная персептронная сеть [29,32,38]. Тем не менее, алгоритм обучения такой сети, адаптирующий значения синаптических весов, является более сложным вследствие зависимости сигналов в момент времени t от их значений в предыдущие моменты и соответственно ввиду более громоздкой формулы для расчета вектора градиента.

Отсутствие обратной связи гарантирует безусловную устойчивость сетей. Они не могут войти в режим, когда выход беспрерывно блуждает от состояния к состоянию и не пригоден к использованию. Но это весьма желательное свойство достигается не бесплатно, сети без обратных связей обладают более ограниченными возможностями по сравнению с сетями с обратными связями.

Так как сети с обратными связями имеют пути, передающие сигналы от выходов к входам, то отклик таких сетей является динамическим, т. е. после приложения нового входа вычисляется выход и, передаваясь по сети обратной связи, модифицирует вход. Затем выход повторно вычисляется, и процесс повторяется снова и снова. Для устойчивой сети последовательные итерации приводят к все меньшим изменениям выхода, пока, в конце концов, выход не становится постоянным. Для многих сетей процесс никогда не заканчивается, такие сети называют неустойчивыми. Неустойчивые сети обладают интересными свойствами и изучались в качестве примера хаотических систем.

Сконцентрируем внимание на устойчивых сетях, т. е. на тех, которые, в конце концов, дают постоянный выход.

Классы сетей с обратной связью:

- сеть Джордана;
- сеть Элмана.

Сеть Джордана. Один из простейших способов построения рекуррентной сети на базе однонаправленной нейронной сети состоит во введении в персептронную сеть обратной связи [7,8]. Таковой является сеть Джордана структура, которой представлена на рисунке 22.

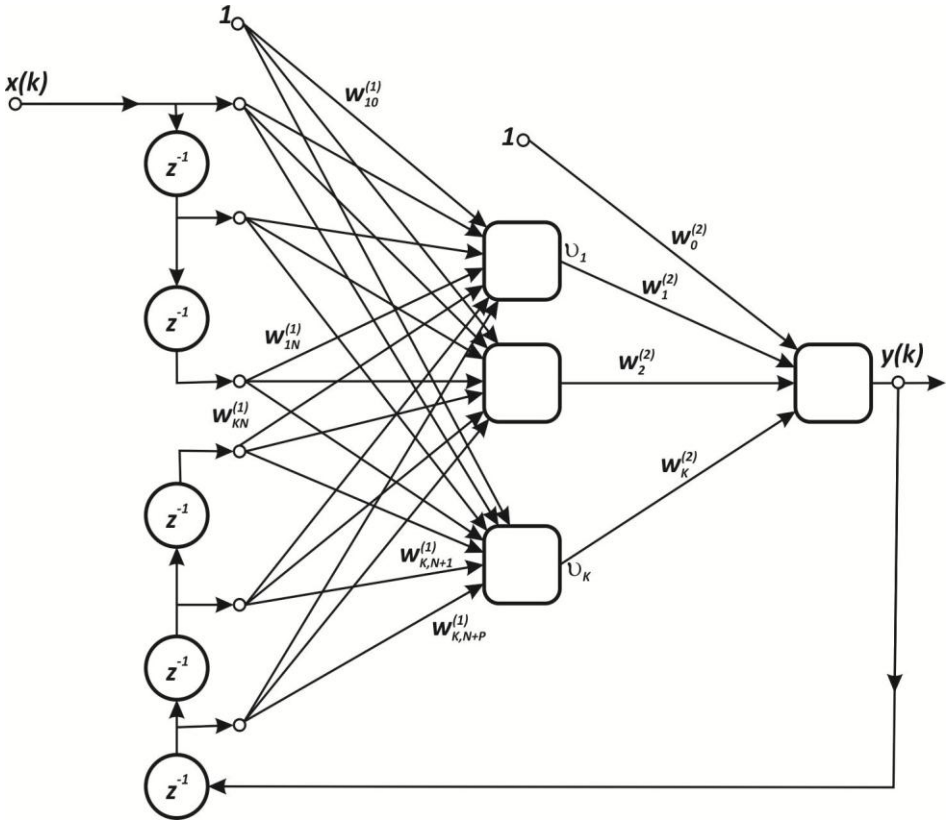


Рис. 22. Структура рекуррентной сети Джордана.

Это динамическая сеть, характеризующаяся запаздыванием входных и выходных сигналов реализуемой с помощью единичных элементов запаздывания z^{-1}, объединяемых во входной вектор сети. Рассмотрим только один входной узел $x(k)$ и один выходной нейрон, а также один скрытый слой. Такая система реализует отображение:

$$y(k+1) = f\big(x(k), x(k-1), \dots, x\big(k-(N-1)\big), y(k-1), y(k-2), \dots, y(k-P)\big) \quad (1.10)$$

где $N-1$ - количество задержек входного сигнала, а P – количество задержек выходного сигнала. Обозначим K количество нейронов в скрытом слое. В этом случае сеть Джордана можно характеризовать тройкой чисел (N, P, K).

Подаваемый на вход сети вектор x имеет вид: $x(k) = \left[1,\ x(k), x(k-1), \ldots, x\big(k-(N-1)\big), y(k-P), y\big(k-(P+1)\big), \ldots, y(k-1)\right]^T$. Допустим, что все нейроны имеют сигмоидальную функцию активации. Обозначим v_i взвешенную сумму сигналов i-го нейрона скрытого слоя, а взвешенную сумму сигналов выходного нейрона, как g. При введенных обозначениях выходные сигналы конкретных нейронов описываются зависимостями:

$$v_i = \sum_{j=0}^{N+P} w_{ij}^{(1)} x_j \qquad (1.11)$$

$$v_i = f(v_i) \qquad (1.12)$$

$$g = \sum_{i=0}^{K} w_i^{(2)} f(v_i) \qquad (1.13)$$

$$y = f(g) \qquad (1.14)$$

Сеть Элмана. Рекуррентная сеть Элмана характеризуется частичной рекуррентностью в форме обратной связи между скрытым и входным слоем, реализуемой с помощью единичных элементов запаздывания z^{-1} [7,8]. Обобщенная структура этой сети представлена на рисунке 23. Каждый скрытый нейрон имеет свой аналог в контекстном слое, образующем совместно с внешними входами сети входной слой. Выходной слой состоит из нейронов однонаправленно связанных только с нейронами скрытого слоя, подобно сети Джордана. Обозначим внутренний вектор возбуждения сети x, состояния скрытых нейронов – $v \in R^K$, а выходные сигналы сети – $y \in R^M$.

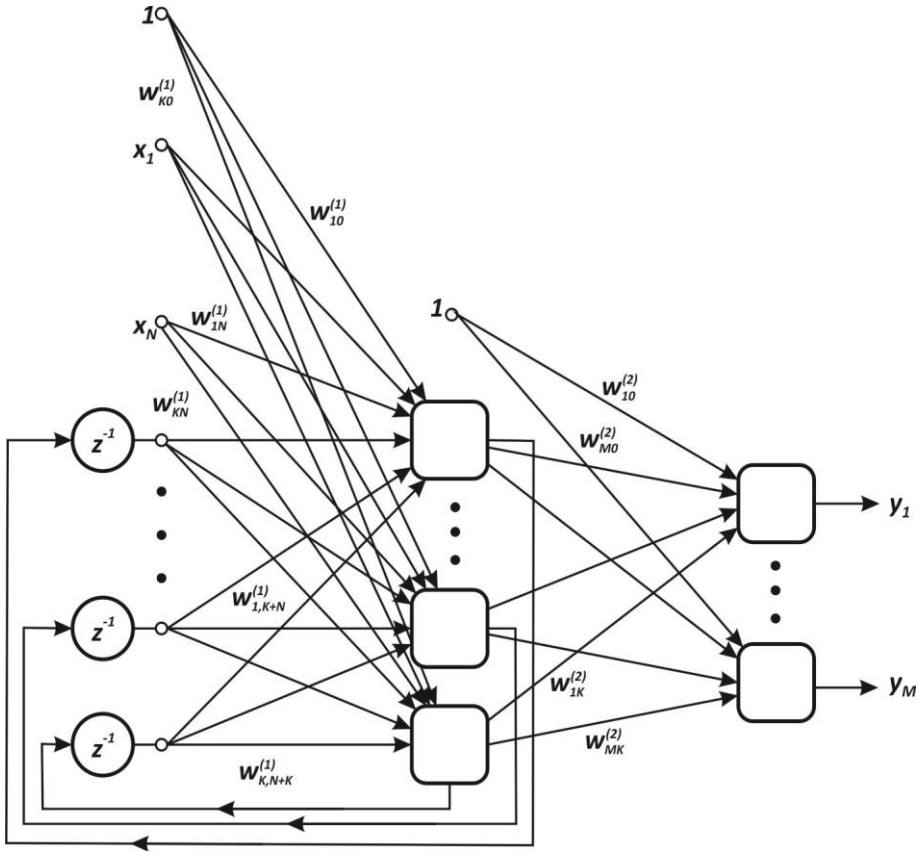


Рис. 23. Структура рекуррентной сети Элмана

При таких обозначениях входной вектор сети в момент t имеет форму

$$x(k) = [x_0(k), x_1(k), \dots, x_N(k), v_1(k-1), v_2(k-1), \dots, v_K(k-1)] \qquad (1.15)$$

Веса синаптических связей первого (скрытого) слоя сети обозначим $w_{ij}^{(1)}$, а второго (выходного) слоя $-w_{ij}^{(2)}$. Если взвешенную сумму i-го нейрона скрытого слоя обозначить v_i, а его выходной сигнал - v_i, то

$$v_i(k) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(k) \qquad (1.16) \qquad v_i = f_1(v_i(k)) \qquad (1.17)$$

Веса $w_{ij}^{(1)}$ образуют матрицу $W^{(1)}$ синаптических связей скрытого слоя, а $f_1(v_i)$ – функция активации i-го нейрона этого слоя. Аналогично можно обозначить взвешенную сумму i-го нейрона выходного слоя g_i, а соответствующий ему выходной сигнал сети – y_i. Эти сигналы описываются формулами

$$g_i(k) = \sum_{j=0}^{K} w_{ij}^{(2)} v_j(k) \qquad (1.18) \qquad y_i = f_2(g_i(k)) \qquad (1.19)$$

В свою очередь, веса $w_{ij}^{(2)}$ образуют матрицу $W^{(2)}$, описывающую синаптические связи нейронов выходного слоя, а $f_2(g_i)$ - функция активации i-го нейрона выходного слоя.

В общем случае можно использовать объединённую сеть Джордана-Элмана.

1.5.3 Рекуррентные нейронные сети. Сети Хопфилда и Хэмминга

Среди различных конфигураций искусственных нейронных сетей встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем, ни обучение без учителя. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации можно расценивать как помощь учителя, но с другой — сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с внешним миром (учителем) не приходится. Из сетей с подобной логикой работы наиболее известны ***сеть Хопфилда*** и ***сеть Хэмминга*** (представляющие собой разновидности сетей с обратными связями), которые обычно используются для организации ассоциативной памяти.

Синхронная сеть Хопфилда. В работе Хопфилда функция F была просто пороговой функцией. Выход такого нейрона равен единице, если

взвешенная сумма выходов с других нейронов больше порога T_j, в противном случае она равна нулю. Он вычисляется следующим образом:

$$Y_j = \sum_{i \neq j} w_{ij} S_i + x_j, \quad \begin{cases} Y = 1, & \text{если } S_j > T_j \\ Y = 0, & \text{если } S_j < T_j \\ Y \text{ не изменяется}, & \text{если } S_j = T_j \end{cases} \qquad (1.20)$$

Состояние сети – это множество текущих значений сигналов Y от всех нейронов. В первоначальной сети Хопфилда состояние каждого нейрона менялось в дискретные случайные моменты времени, в последующей работе состояния нейронов могли меняться одновременно. Так как выходом бинарного нейрона может быть только ноль или единица (промежуточных уровней нет), то текущее состояние сети является двоичным числом, каждый бит которого является сигналом Y некоторого нейрона.

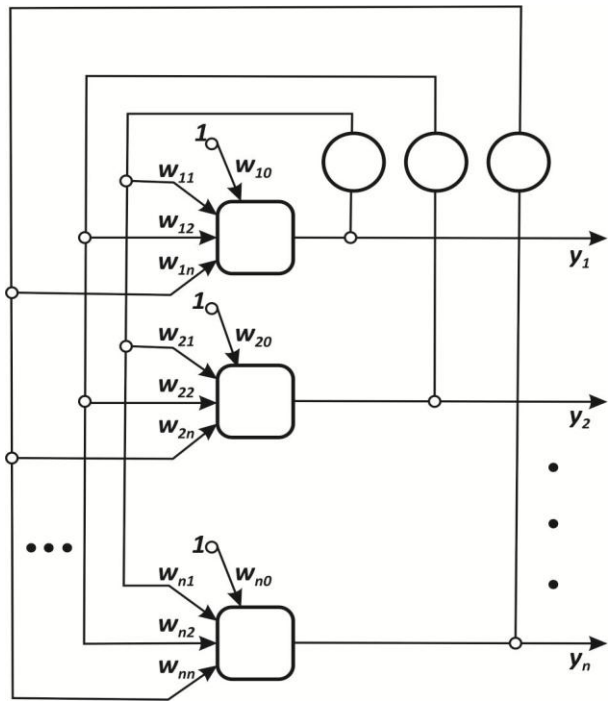


Рис. 24. Структура нейронной сети Хопфилда

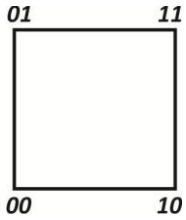


Рис. 25. Два нейрона порождают систему с четырьмя состояниями

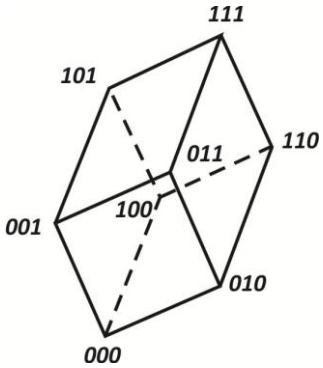


Рис. 26. Три нейрона порождают систему с восемью состояниями

Функционирование сети легко визуализируется геометрически. На рисунке 25 показан случай двух нейронов в выходном слое, причем каждой вершине квадрата соответствует одно из четырех состояний системы $(00, 01, 10, 11)$. На рисунке 26 показана трехнейронная система, представленная кубом (в трехмерном пространстве), имеющим восемь вершин, каждая из которых помечена трехбитовым бинарным числом. В общем случае система с n нейронами имеет 2^n различных состояний и представляется n-мерным гиперкубом.

Когда подается новый входной вектор, сеть переходит из вершины в вершину, пока не стабилизируется. Устойчивая вершина определяется

сетевыми весами, текущими входами и величиной порога. Если входной вектор частично неправилен или неполон, то сеть стабилизируется в вершине, ближайшей к желаемой.

Как и в других сетях, веса между слоями в этой сети могут рассматриваться в виде матрицы W. Сеть с обратными связями является устойчивой, если ее матрица симметрична и имеет нули на главной диагонали, т. е. если $w_{ij} = w_{ji}$ и $w_{ij} = 0$ для всех i.

Устойчивость такой сети может быть доказана с помощью элегантного математического метода. Допустим, что найдена функция, которая всегда убывает при изменении состояния сети. В конце концов, эта функция должна достичь минимума и прекратить изменение, гарантируя тем самым устойчивость сети. Такая функция, называемая функцией Ляпунова, для рассматриваемых сетей с обратными связями может быть введена следующим образом:

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} Y_i Y_j - \sum_j \Omega_j Y_j + \sum_j T_j Y_j \qquad (1.21)$$

где E – искусственная энергия сети; w_{ij} – вес от выхода нейрона i к входу нейрона j; Y_j – выход нейрона j; Ω_j – внешний вход нейрона j; T_j – порог нейрона j.

Изменение энергии E, вызванное изменением состояния j - нейрона, есть

$$\delta E = \left[\sum_{i \neq j} \left(w_{ij} Y_i\right) + \Omega_j - T_j\right] \delta Y_i = -\left[Y_i - T_j\right] \delta Y_i \qquad (1.22)$$

где δY_i – изменение выхода j-го нейрона.

Допустим, что величина S нейрона j больше порога. Тогда выражение в скобках будет положительным, а из уравнения (1.20) следует, что выход нейрона j должен измениться в положительную сторону (или остаться без изменения). Это значит, что δY_i может быть только положительным или нулем и δE должно быть отрицательным. Следовательно, энергия сети должна либо уменьшиться, либо остаться без изменения.

Допустим, что величина S меньше порога. Тогда величина δY_i может быть только отрицательной или нулем. Следовательно, опять энергия должна уменьшиться или остаться без изменения. И окончательно, если величина S равна порогу, δ_j равна нулю и энергия остается без изменения.

Это показывает, что любое изменение состояния нейрона либо уменьшит энергию, либо оставит ее без изменения. Благодаря такому непрерывному стремлению к уменьшению энергия, в конце концов, должна достигнуть минимума и прекратить изменение. По определению такая сеть является устойчивой.

Симметрия сети является достаточным, но не необходимым условием для

устойчивости системы. Можно продемонстрировать примеры, в которых незначительное отклонение от симметрии может приводить к непрерывным осцилляциям. Однако приближенной симметрии обычно достаточно для устойчивости системы.

Сеть Хопфилда называется синхронной тогда и только тогда, когда все её нейроны изменяют состояние согласно функции активации одновременно. Для дискретного состояния значение функции активации вычисляется согласно выражению (1.20), как значение Y, либо согласно выражению:

$$F = \sum_{i \neq j} w_{ij} Y_j(t-1),$$
$$Y_i(t) = \begin{cases} 1, & \text{если } F > 0 \\ -1, & \text{если } F < 0, \\ Y_i(t-1), & \text{если } F = 0 \end{cases} \quad (1.23)$$

либо для непрерывного состояния:

$$Y_i(t) = th(\sum_{i \neq j} w_{ij} Y_j(t-1)). \quad (1.24)$$

Асинхронная сеть Хопфилда. Сеть Хопфилда называется асинхронной тогда и только тогда, когда все её нейроны изменяют состояние согласно функции активации последовательно – каждый через n тактов, где n – число нейронов. Асинхронная сеть Хопфилда также может быть сетью с дискретным состоянием (1.23), либо – сетью с непрерывным состоянием (1.24).

Пусть есть сеть Хопфилда с непрерывным состоянием.

$$y_i(t+1) = th\big(S_i(t)\big), \quad (1.25)$$

$$S_i(t) = \sum W_{ij} Y_j - T_i. \quad (1.26)$$

Чтобы изменение состояния и входной активности нейрона приводило к уменьшению энергии такой сети, выходное значение i-го нейрона должно быть пропорционально градиенту энергии:

$$y_i(t) = {-dE(y_i(t))}\Big/{dS_i(t)}. \quad (1.27)$$

Доказательство:

$$E\big(y_i(t)\big) = -\int y_i(t) d\, S_i = -y_i(t) * S_i(t) + \int S_i(t)\, dy_i \quad (1.28)$$

$$S_i(t) = F^{-1}(y_i(t)) \quad (1.29)$$

$$E\big(y_i(t)\big) = -\int y_i(t) d\, S_i = -y_i(t) * S_i(t) + \int F^{-1}(y_i(t))\, dy_i \quad (1.30)$$

$$\Delta E\left(y_j(t+1)\right) = E\left(y_j(t+1)\right) - E\left(y_j(t)\right); \quad (1.31)$$

Для асинхронного режима:

$$\Delta E\left(y_j(t+1)\right) = -y_j(t+1) * S_i(t+1) + \int F^{-1}\left(y_i(t+1)\right) dy_i + y_j(t) * S_i(t) - \int F^{-1}(y_i(t))\, dy_i \quad (1.32)$$

$$\Delta E\left(y_j(t+1)\right) = -y_j(t+1) * S_i(t) + \int F^{-1}\left(y_i(t+1)\right) dy_i + y_j(t) * S_i(t) - \int F^{-1}(y_i(t))\, dy_i \quad (1.33)$$

$$\Delta E\left(y_j(t+1)\right) = \left(y_j(t) - y_i(t+1)\right) * S_i(t) + \int F^{-1}\left(y_i(t+1)\right) dy_i - \int F^{-1}(y_i(t))\, dy_i \quad (1.34)$$

По теореме о среднем:

$$\Delta E\left(y_j(t+1)\right) = \left(y_j(t) - y_i(t+1)\right) * \left(S_i(t) - F^{-1}(\varepsilon)\right). \quad (1.35)$$

Если $y_j(t) > y_j(t+1)$, то $S_i(t) = F^{-1}(y_i(t))$, то и в силу монотонности функции активации: $y_j(t+1) \le \varepsilon \le y_j(t)$: $F^{-1}(\varepsilon) \le F^{-1}\left(y_i(t)\right)$.

Если $y_j(t) < y_j(t+1)$, то $S_i(t) = F^{-1}(y_i(t))$, то и в силу монотонности функции активации: $y_j(t+1) \ge \varepsilon \ge y_j(t)$: $F^{-1}(\varepsilon) \ge F^{-1}\left(y_i(t)\right)$.

Для дискретного:

$$E\left(y_j(t)\right) = -y_j(t) * S_i(t) \quad (1.36)$$

Для синхронного:

$$E\left(y_j(t)\right) = -y_j(t-1) * S_i(t) \quad (1.37)$$

Сеть Хэмминга. Сеть Хемминга – это трехслойная рекуррентная структура, которую можно считать развитием сети Хопфилда. Она позиционируется как специализированное гетероассоциативное запоминающее устройство. Основная идея функционирования этой сети состоит в минимизации расстояния Хемминга между тестовым вектором, подаваемым на вход сети, и векторами обучающих выборок, закодированными в структуре сети [7,8].

На рисунке 27 представлена обобщенная схема сети Хемминга. Первый ее слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов. Второй слой, «$MAXNET$», состоит из нейронов связанных обратными связями по принципу «каждый с каждым», при этом в

отличие от структуры сети Хопфилда - существует ненулевая связь входа нейрона со своим собственным выходом. Веса нейронов в слое «$MAXNET$» также постоянны. Разные нейроны связаны отрицательной (подавляющей) обратной связью с весом $-\varepsilon$, при этом обычно величина ε обратно пропорциональна количеству образов. С собственным выходом нейрон связан положительной (возбуждающей) обратной связью с весом, равным $+1$. Веса поляризации нейронов принимают значения, соответствующие нулю. Нейроны этого слоя функционируют в режиме WTA ($Winner\ Takes\ All$), при котором в каждой фиксированной ситуации активизируется только один нейрон, а остальные пребывают в состоянии покоя. Выходной однонаправленный слой формирует выходной вектор, соответствующий входному вектору. Веса нейронов этого слоя подбираются в зависимости от входных обучающих выборок.

В процессе функционирования сети можно выделить три фазы. В первой из них на ее вход подается N - элементный вектор x. После предъявления этого вектора на выходах нейронов первого слоя генерируются сигналы, задающие начальные состояния нейронов второго слоя, т.е. «$MAXNET$».

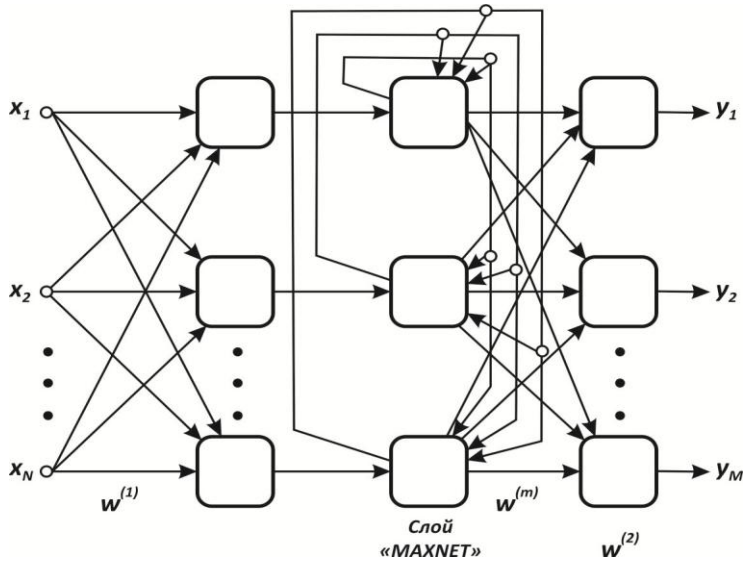


Рис. 27. Структура нейронной сети Хемминга

Во второй фазе инициировавшие «$MAXNET$» сигналы удаляются, и из сформированного ими начального состояния запускается итерационный процесс внутри этого слоя. Итерационный процесс завершается в момент, когда все нейроны, кроме одного (победителя с выходным сигналом, равным 1), перейдут в нулевое состояние. *Нейрон-победитель* с ненулевым выходным сигналом становится представителем класса данных, к которому принадлежит входной вектор.

В третьей фазе этот же нейрон посредством весов, связывающих его с нейронами выходного слоя, формирует на выходе сети отклик в виде вектора y, соответствующий возбуждающему вектору x.

Сеть Хемминга считается гетероассоциативным запоминающим устройством с парой связанных между собой векторов (y, x), где x и y - это соответственно входной и выходной биполярные векторы сети со значениями элементов ± 1. Входные узлы сети $1, 2, \ldots, N$ принимают значения, задаваемые аналогичными компонентами вектора x. Нейроны первого слоя рассчитывают расстояние Хемминга между фактически предъявленным входным вектором x и каждым из p закодированных векторов-образцов $x^{(i)}$, образующих веса нейронов первого слоя. Нейроны в слое «$MAXNET$» выбирают вектор с наименьшим расстоянием Хемминга, определяя, таким образом, класс, к которому принадлежит предъявленный входной вектор x. Веса нейронов выходного слоя формируют вектор, соответствующий предъявленному входному вектору. При p нейронах первого слоя емкость запоминающего устройства Хемминга также равна p, поскольку каждый нейрон представляет единственный класс.

Подбор весов сети Хемминга оказывается чрезвычайно простым. Весы первого слоя соответствуют очередным векторам образов $x^{(i)}$, поэтому

$$w_{ij}^{(1)} = x_j^i \tag{1.38}$$

для $i = 1, 2, \ldots,$ p. Аналогично веса выходного слоя соответствуют очередным векторам образов $y^{(i)}$ связанным с $x^{(i)}$:

$$w_{ji}^{(2)} = y_j^i \tag{1.39}$$

В случае нейронов слоя «$MAXNET$», функционирующих в режиме WTA, веса сети должны усиливать собственный сигнал нейрона и ослаблять остальные. Для достижения этого эффекта принимается

$$w_{ii}^{(m)} = 1, \tag{1.40}$$

а также

$$-\frac{1}{(p-1)} < w_{ii}^{(m)} < 0 \tag{1.41}$$

для $i \neq j$. Для обеспечения абсолютной сходимости алгоритма веса должны отличаться друг от друга. Будем считать, что

$$w_{ii}^{(m)} = -\frac{1}{(p-1)} + \xi, \tag{1.42}$$

где ξ - случайная величина с достаточно малой амплитудой [54].

Нейроны различных слоев сети Хемминга функционируют по-разному. Нейроны первого слоя рассчитывают расстояния Хемминга между поданными на вход сети вектором x и векторами весов $w^{(i)} = x^{(i)}$ отдельных нейронов этого слоя $(i = 1, 2, \ldots, p)$. Значения выходных сигналов этих нейронов определяются по формуле

$$\frac{dE}{dw_{ij}^{[q]}} = \left(\frac{dE}{dy_j^{[q]}}\right) * \left(\frac{dy_j^{[q]}}{dS_j^{[q]}}\right) * \left(\frac{dS_j^{[q]}}{dw_{ij}^{[q]}}\right), \quad q = 1, \ldots, p.$$

$$\widehat{y_i} = 1 - \frac{d_H(x^{(i)}, x)}{N}, \tag{1.43}$$

где $d_H(x^{(i)}, x)$ обозначает расстояние Хемминга между входными векторами x и $x^{(i)}$, т.е. количество битов, на которое различаются эти два вектора. Значение равно 1, если $x = x^{(i)}$ и равно 0, если $x = -x^{(i)}$. В остальных случаях значения $\widehat{y_i}$ располагаются в интервале [0, 1].

Сигналы нейронов первого слоя становятся начальными состояниями $\widehat{y_i}$ нейронов слоя «$MAXNET$» на второй фазе функционирования сети. Задача нейронов этого слоя состоит в определении победителя, т.е. нейрона, уровень возбуждения которого наиболее близок к 1. Такой нейрон указывает на вектор образа с минимальным расстоянием Хемминга до входного вектора x. Процесс определения победителя – это рекуррентный процесс, выполняемый согласно формуле

$$y_i(k) = f\left(\sum_j w_{ii}^{(m)} y_j(k-1)\right) = f\left(y_j(k-1) + \sum_{i \neq j} w_{ii}^{(m)} y_j(k-1)\right), \tag{1.44}$$

при начальном значении $y_i(0) = \widehat{y_i}$. Функция активации $f(y)$ нейронов слоя «$MAXNET$» задается выражением

$$f(y) = \begin{cases} y, & y \geq 0 \\ 0, & y < 0. \end{cases} \tag{1.45}$$

Итерационный процесс (1.44) завершается в момент, когда состояние нейронов стабилизируется и активность продолжает проявлять только один нейрон, тогда как остальные пребывают в нулевом состоянии. Активный нейрон становится победителем и через веса линейных нейронов выходного слоя $w_{ii}^{(2)}$ представляет вектор $y^{(i)}$, который соответствует вектору $x^{(i)}$, признанному слоем «$MAXNET$» в качестве ближайшего к входному вектору x.

Важным достоинством сети Хемминга считается небольшое количество взвешенных связей между нейронами. Например, 100 - входовая сеть Хопфилда, кодирующая 10 различных векторных классов, должна содержать 10000 взвешенных связей с подбираемыми значениями весов. При построении аналогичной сети Хемминга количество взвешенных связей уменьшается до 1100, из которых 1000 весов находятся в первом слое и 100 – в слое «$MAXNET$». Выходной слой в этом случае не учитывается, поскольку сеть Хемминга, аналогичная сети Хопфилда, является ассоциативной.

В результате многочисленных экспериментов доказано, что рекуррентная сеть Хемминга дает лучшие результаты, чем сеть Хопфилда, особенно в ситуациях, когда взаимосвязанные векторы x и y являются случайными. В частности, реализованная в программе *Mathlab* сеть Хемминга, протестированная на 10 цифрах позволила почти безошибочно распознать все

представленные зашумленные образы. Достигнутая эффективность распознавания зашумленных образов составила 100%.

1.5.4 Двунаправленная ассоциативная память

Сеть Хопфилда реализует так называемую автоассоциативную память. Это означает, что образ может быть завершен или исправлен, но не может быть ассоциирован с другим образом. Двунаправленная ассоциативная память (BAM - Bidirectorial Associative Memory), разработанная в 1988 г. Бертом Коско [66], является гетероассоциативной: она сохраняет пары образов и выдает второй образец пары, когда ассоциированный с ним первый образец подается на вход сети. Как и сеть Хопфилда, BAM способна к обобщению, вырабатывая правильные реакции, несмотря на искаженные входы. Сеть BAM (рис. 28) содержит два слоя нейронов.

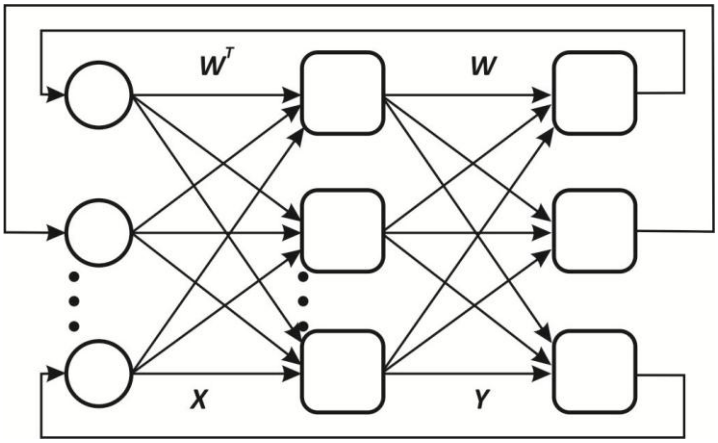


Рис. 28. Двунаправленная ассоциативная память

Элементы весовой матрицы w_{ij} отражают связь между i-м нейроном первого слоя и j-м нейроном второго слоя $i = 1,n;\ \ j = 1,m$. В процессе функционирования сети входной вектор X умножается на транспонированную матрицу весов сети W^T и подается на вход первого слоя, в результате чего вырабатывается вектор выходных сигналов нейронов первого слоя Y. Вектор Y затем умножается на матрицу W и подается на вход второго слоя, который вырабатывает выходные сигналы, представляющие собой новый входной вектор X. Этот процесс повторяется до тех пор, пока сеть не достигнет стабильного состояния, в котором ни вектор X, ни вектор Y не изменяются.

Нейроны в обоих слоях сети BAM функционируют аналогично нейронам сети Хопфилда. Этот процесс может быть выражен следующим образом:

$$y_j^{N+1} = f\left(\sum_{i=1}^{n} x_i^N w_{ji}\right),\ \ j = \overline{1,m}$$

$$x_i^{N+1} = f\left(\sum_{j=1}^{n} y_j^N w_{ij}\right),\ \ i = \overline{1,n}$$

$$f(S) = \begin{cases} -1, & S < \Theta_j \\ 1, & S < \Theta_j \\ f^{pred}(S), & S = \Theta_j, \end{cases}$$

где $f^{pred}(S)$ - значение функции активации данного нейрона на предыдущем

шаге.

Пусть задана обучающая выборка ассоциированных образов (X^k, Y^k), $k = \overline{1,K}$. Весовая матрица сети BAM вычисляется как сумма произведений всех векторных пар обучающего набора:

$$w_{ij} = \sum_{k=1}^{K} x_i^k y_j^k, \quad i = \overline{1,n}, \qquad j = \overline{1,m}. \tag{1.46}$$

В отличие от сети Хопфилда, весовая матрица в сети BAM не квадратная, что во многих случаях позволяет оптимизировать вычислительные затраты, необходимые для функционирования сети.

Основным недостатком сети BAM (как и сети Хопфилда) является небольшая емкость памяти. Так, например, число запоминаемых ассоциаций не может превышать числа нейронов в меньшем слое. Если все пороговые значения Θ_j будут нулевыми, то оценка еще ухудшается: размер запоминаемой выборки не должен превосходить $L/2\log_2 L$, где L – число нейронов в меньшем слое. Если этот лимит превышен, сеть начинает вырабатывать неверные выходные сигналы, воспроизводя ассоциации, которым не обучена.

1.5.5 Нейронная сеть Кохонена. Нейроны Гроссберга

Сеть Кохонена — самоорганизующаяся карта признаков — была предложена Кохоненом в 1984 году. В настоящее время существует множество модификаций исходной модели с развитой математической теорией построения и функционирования [43,10].

Задача классификации заключается в разбиении объектов на классы, причем основой разбиения служит вектор параметров объекта. Сами классы часто бывают, неизвестны заранее, а формируются динамически. Назовем *прототипом* класса объект, наиболее типичный для своего класса. Один из самых простых подходов к классификации состоит в том, чтобы предположить существование определенного числа классов и произвольным образом выбрать координаты прототипов. Затем каждый вектор из набора данных связывается с ближайшим к нему прототипом, и новыми прототипами становятся центроиды всех векторов, связанных с исходным прототипом. В качестве меры близости двух векторов обычно выбирается евклидово расстояние:

$$d(x,y) = \sum_i (x_i - y_i)^2.$$

На этих принципах основано функционирование сети Кохонена, обычно используемой для решения задач классификации. Данная сеть обучается *без учителя* на основе самоорганизации. По мере обучения вектора весов нейронов становятся прототипами классов – групп векторов обучающей выборки. На этапе решения информационных задач сеть относит новый предъявленный образ к одному из сформированных классов.

Сеть Кохонена состоит из одного слоя нейронов. Число входов каждого нейрона n равно размерности вектора параметров объекта. Количество нейронов m совпадает с требуемым числом классов, на которые нужно разбить объекты (меняя число нейронов, можно динамически менять число классов).

Обучение начинается с задания небольших случайных значений элементам весовой матрицы W. В дальнейшем происходит процесс самоорганизации, состоящий в модификации весов при предъявлении на вход векторов обучающей выборки. Каждый столбец весовой матрицы представляет собой параметры соответствующего нейрона-классификатора. Для каждого j-го нейрона ($j = \overline{1,m}$) определяется расстояние от него до входного вектора X:

$$d_j = \sum_{i=1}^{n} (x_i - w_{ij})^2 \qquad (1.47)$$

Далее выбирается нейрон с номером k, $1 \le k \le m$, для которого это расстояние минимально (то есть сеть отнесла входной вектор к классу с номером k). На текущем шаге обучения N будут модифицироваться только веса нейронов из окрестности нейрона k:

$$w_{ij}^{N+1} = w_{ij}^{N} + \alpha_N \left(x_i - w_{ij}^{N}\right). \qquad (1.48)$$

Первоначально в окрестности любого из нейронов находятся все нейроны сети, но с каждым шагом эта окрестность сужается. В конце этапа обучения подстраиваются только веса только нейрона с номером k. Темп обучения α_N с течением времени также уменьшается (часто полагают $\alpha_0 = 0.9$, $\alpha_{N+1} = \alpha_N - 0.001$). Образы обучающей выборки предъявляются последовательно, и каждый раз происходит подстройка весов.

Обучение сети Кохонена происходит следующим образом:

Шаг 1. Инициализация сети. Весовым коэффициентам сети w_{ij}, $i = \overline{1,n}$, $j = \overline{1,m}$, присваиваются случайные значения. Задаются значения α_0 – начальный темп обучения и D_0 – максимальное расстояние между весовыми векторами (столбцами матрицы W).

Шаг 2. Предъявление сети нового входного сигнала X.

Шаг 3. Вычисление расстояния от входа X до всех нейронов сети:

$$d_j = \sum_{i=1}^{n} (x_i - w_{ij}^{N})^2, \quad j = \overline{1,m}.$$

Шаг 4. Выбор нейрона k, $1 \le k \le m$ с наименьшим расстоянием d_k.

Шаг 5. Настройка весов нейрона k и всех нейронов, находящихся от него на расстоянии, не превосходящем D_N.

$$w_{ij}^{N+1} = w_{ij}^{N} + \alpha_N \left(x_i - w_{ij}^{N}\right).$$

Шаг 6. Уменьшение значений α_N, D_N.

Шаг 7. Шаги 2–6 повторяются до тех пор, пока веса не перестанут меняться (или пока суммарное изменение всех весов станет очень мало). После обучения классификация выполняется посредством подачи на вход сети испытуемого вектора, вычисления расстояния от него до каждого нейрона с последующим выбором нейрона с наименьшим расстоянием как индикатора правильной классификации.

Замечание 1. Если предварительно провести единичную нормировку всех входных векторов, то есть подавать на вход сети образы X', компоненты которого связаны с компонентами векторами X по формулам:

$$x_i^{'} = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}},$$

а также если после каждой итерации процесса обучения осуществлять нормировку весов каждого нейрона (столбцов матрицы W), то в качестве меры близости входных векторов и весовых векторов нейронов сети можно рассматривать скалярное произведение между ними.

Действительно в этом случае

$$d_j = \sum_{i=1}^{n}(x_i - w_{ij}^N)^2 = \sum_{i=1}^{n} x_i^2 + 2\sum_{i=1}^{n} x_i\, w_{ij}^N + \sum_{i=1}^{n} w_{ij}^2 = 2 - 2\sum_{i=1}^{n} x_i w_{ij}^N.$$

Таким образом, наименьшим будет расстояние до того нейрона, скалярное произведение с весами которого у входного вектора максимально. В этом случае можно считать, что каждый нейрон Кохонена реализует тождественную активационную функцию $f(S) = S$, где $S = \sum_{i=1}^{n} w_{ij}\, x_i$. Нейрон с максимальным значением активационной функции объявляется «победителем» и его веса (а также веса нейронов из его окружения) пересчитываются.

Замечание 2. Диапазон изменений начальных значений весовых коэффициентов должен совпадать с диапазоном изменения соответствующих им входов. Сеть Кохонена нашла самое широкое применение в задачах финансового анализа. С ее помощью успешно решаются задачи предсказания рисков, рейтингования и многие другие.

Нейроны Гроссберга. Входная звезда Гроссберга, как показано на рисунке 29, состоит из нейрона, на который подается группа входов, умноженных на синаптические веса.

Выходная звезда, показанная на рисунке 30, является нейроном, управляющим группой весов. Входные и выходные звезды могут быть взаимно соединены в сети любой сложности.

Входная звезда выполняет распознавание образов, т. е. она обучается реагировать на определенный входной вектор X и ни на какой другой. Это обучение реализуется путем настройки весов таким образом, чтобы они соответствовали входному вектору.

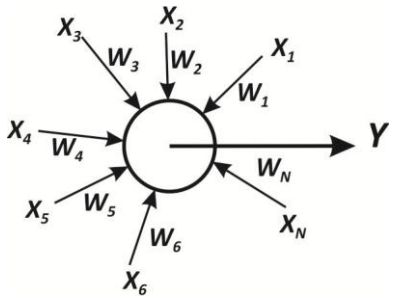


Рис. 29. Входная звезда Гроссберга

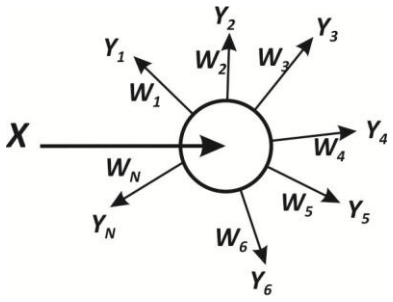


Рис. 30. Выходная звезда Гроссберга

Обучение входной звезды. Входная звезда имеет тождественную активационную функцию: $f(S) = S$, то есть выход входной звезды определяется как взвешенная сумма ее входов: $Y = \sum_{i=1}^{n} w_i x_i$. С другой точки зрения, выход можно рассматривать как скалярное произведение входного вектора с весовым вектором. Если эти векторы имеют единичную норму, то скалярное произведение будет максимальным для того входного образа, которому нейрон был обучен.

В процессе обучения веса корректируются следующим образом:

$$w_i^{N+1} = w_i^N + \alpha_N\left(x_i - w_i^N\right), \tag{1.49}$$

где w_i – весовой коэффициент входа x_i; α – нормирующий коэффициент обучения, который имеет начальное значение $0{,}1$ и постепенно уменьшается в процессе обучения.

После завершения обучения предъявление входного вектора X будет активизировать обученный входной нейрон. Хорошо обученная входная звезда будет реагировать не только на определенный запомненный вектор, но также и на незначительные изменения этого вектора. Это достигается постепенной настройкой нейронных весов при предъявлении в процессе обучения векторов, представляющих нормированные вариации входного вектора. Веса настраиваются таким образом, чтобы усреднить величины обучающих векторов, и нейроны получают способность реагировать на вектор этого класса.

Обучение выходной звезды. В то время как входная звезда учится реагировать на определенный вход, выходная звезда обучается выдавать требуемый целевой выход. Для того чтобы обучить нейрон выходной звезды, его веса настраиваются в соответствии с требуемым целевым выходным вектором Y. Формула коррекции весов имеет вид:

$$w_i^{N+1} = w_i^N + \beta_N\left(y_i - w_i^N\right), \tag{1.50}$$

где β представляет собой нормирующий коэффициент обучения, который в начале, приблизительно равен единице и постепенно уменьшается до нуля в процессе обучения.

Как и в случае входной звезды, веса выходной звезды постепенно настраиваются на множестве векторов, представляющих собой возможные вариации запоминаемого выходного вектора.

1.5.6 Двухслойная сеть встречного распространения

Сеть встречного распространения состоит из двух слоев: слоя нейронов Кохонена и слоя нейронов Гроссберга. Автор сети Р. Хехт-Нильсен удачно объединил эти две архитектуры, в результате сеть приобрела свойства, которых не было у каждой их них в отдельности [19,12,10].

Слой Кохонена классифицирует входные векторы в группы схожих векторов. Это достигается с помощью такой подстройки весов слоя Кохонена, что близкие входные векторы активируют один и тот же нейрон данного слоя. Затем задачей слоя Гроссберга является получение требуемых выходов. На рисунке 31 показана сеть встречного распространения полностью. В режиме нормального функционирования предъявляются входные векторы X и Y, и обученная сеть дает на выходе векторы X' и Y', являющиеся аппроксимациями соответственно для X и Y. Векторы (X, Y) предполагаются здесь нормированными векторами единичной длины, следовательно, порождаемые на выходе векторы также должны быть нормированными.

В процессе обучения векторы X и Y подаются одновременно и как входные векторы сети, и как желаемые выходные сигналы. В результате получается отображение, при котором предъявление пары входных векторов порождает их копии на выходе. Благодаря обобщению, предъявление только вектора X (с вектором Y, равным нулю) порождает как выходы X', так и выходы Y'. Если F – функция, отображающая X в Y', то сеть аппроксимирует ее. Также, если F обратима, то предъявление только вектора Y (приравнивая X нулю) порождает X'. Уникальная способность порождать функцию и обратную к ней делает сеть встречного распространения полезной в ряде приложений.

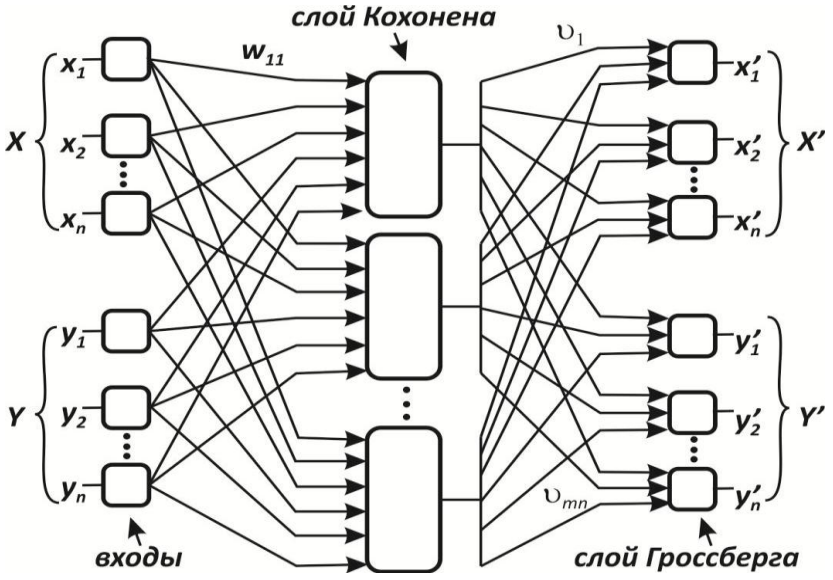


Рис. 31. Сеть встречного распространения

Алгоритм обучения сети.

Шаг 1. Произвести единичную нормировку всех векторов (X, Y) обучающего множества.

Шаг 2. Весовым коэффициентам сети $w_{ij}, v_{ji},\ i = 1,n;\ j = 1,m$ присвоить малые случайные значения и произвести единичную нормировку матриц W, V по столбцам. Положить $\alpha_0 = 0{,}7,\ \beta_0 = 0{,}1$.

Шаг 3. Подать на вход сети обучающий набор $(X,\ Y)$ и определить единственный ***«нейрон-победитель»*** в слое Кохонена (весовой вектор которого дает максимальное скалярное произведение с входным вектором). Выход этого нейрона установить равным 1, выходы всех остальных нейронов слоя Кохонена положить равными 0. Скорректировать веса выигравшего нейрона: $w_{ij}^{N+1} = w_{ij}^{N} + \alpha_N\left(z_i - w_{ij}^{N}\right)$, где $z = (X,\ Y)$.

Шаг 4. Подать выходной вектор слоя Кохонена на вход слоя Гроссберга. Скорректировать веса слоя Гроссберга, связанные с выигравшим нейроном слоя Кохонена: $v_{ki}^{N+1} = v_{ki}^{N} + \beta_N\left(z_i - v_{ki}^{N}\right)$ (здесь k – номер выигравшего нейрона).

Шаг 5. Уменьшить значения $\alpha_N,\ \beta_N$.

Шаг 6. Повторять шаги 3–5 до тех пор, пока каждая входная пара из обучающего множества на выходе будет порождать аналогичную выходную пару.

Замечание. Для улучшения обобщающих свойств сети встречного распространения темп уменьшения значений α и β должен быть очень маленьким, а общее количество итераций достаточно большим. (Все образы обучающей выборки желательно предъявить сети несколько десятков или даже несколько сотен раз).

1.5.7 Сети с радиальными базисными функциями (RBFN)

В нервных системах биологических организмов существуют нейроны, чей выходной сигнал «локален», или «настроен» на некоторую узкую ограниченную область входного пространства [78]. Сеть, построенная на искусственных нейронах, обладающих выраженными локальными характеристиками, была предложена в 1988 г. [70] в качестве альтернативы многослойным персептронам и получила название радиально-базисной нейронной сети (Radial Basis Function Neural Network - RBFN). Основные идеи радиально-базисных нейронных сетей восходят к методу потенциальных функций [1], оценкам Парзена [74, 30], ядерной [61] и непараметрической [13, 36] регрессиям. Подобно многослойным структурам с прямой передачей информации эти сети являются универсальными аппроксиматорами [60, 73].

На рисунке 32 приведена стандартная схема радиально-базисной сети с n - входами и m-выходами, осуществляющая нелинейное преобразование вида

$$y_j = F_j(x) = w_{j0} + \sum_{i=1}^{h} w_{ji}\,\varphi_i(x), \qquad j = 1,2,\dots,m, \tag{1.51}$$

где $\varphi_i(x)$ – радиально-базисные функции, определяющие характер

отображения из n - мерного пространства входов в m - мерное пространство выходов $R_n \rightarrow R_m$.

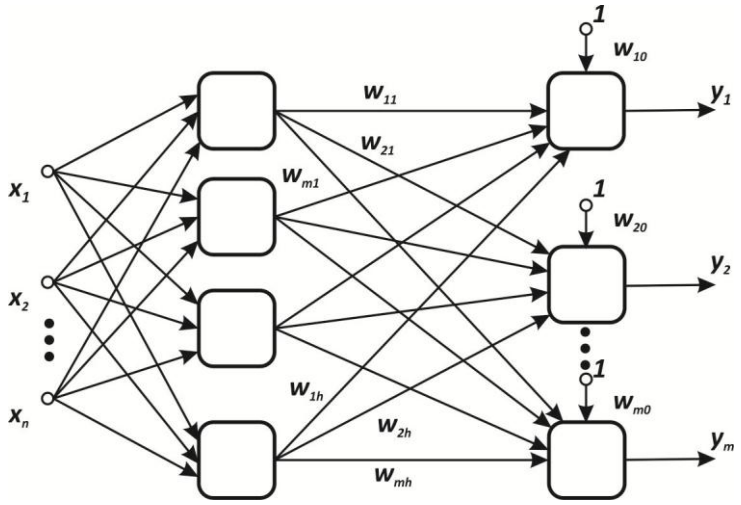


Рис. 32. Радиально-базисная нейронная сеть

Входной слой такой сети – это сенсоры, которые связывают нейронную сеть с окружающей средой. Единственный скрытый слой, образованный нейронами Φ, осуществляет нелинейное преобразование входного пространства R_n в скрытое пространство R_h, как правило, высокой размерности $(h \gg n)$. И, наконец, выходной слой, образованный адаптивными линейными ассоциаторами, формирует отклик сети $y = (y_1, y_2, \dots, y_m)^T$ на входной сигнал сети $x = (x_1, x_2, \dots, x_n)^T$.

Радиально-базисные сети реализуют идею, сформулированную Т.Кавером [51] и состоящую в том, что линейно неразделимая задача распознавания образов в пространстве R_n может стать линейно разделимой в пространстве более высокой размерности R_h.

Свойства такой сети полностью определяются радиально-базисными функциями Φ, используемыми в нейронах скрытого слоя и формирующими некоторый базис для входных векторов x. Радиально-базисная функция

$$\varphi(x) = \Phi(\|x - c\|, \sigma) = \Phi(r, \sigma) \qquad (1.52)$$

– это многомерная функция, зависящая от расстояния $r = x - c$ между входным вектором x и собственным центром c и параметра ширины (масштаба) σ, определяющей локальную область входного пространства, на которую «*реагирует*» данная функция. Таким образом, каждый нейрон скрытого слоя вычисляет расстояние между входным вектором и своим центром и осуществляет над ним некоторое нелинейное преобразование $\Phi(r, \sigma)$.

Важно заметить, что в отличие от монотонных активационных функций многослойных сетей, радиально-базисные функции, как правило, симметричны и «*накрывают*» узкую область входного пространства. Достаточно часто радиально-базисные функции имеют колоколообразную форму и могут быть представлены в виде производных функций активации стандартных нейронов.

Наибольшее распространение получили гауссовские функции, имеющие пик в центре c и монотонно убывающие по мере удаления от центра. В связи с этим в теории и практике нейронные сети кроме схемы-рисунка 32 достаточно широко распространено представление, приведенное на рисунке 33, при этом без потери общности рассматривается структура с одним выходом, осуществляющая отображение $R^n \to R^1$.

В большинстве случаев, связанных с практическими приложениями, центры узлов c_i и параметры ширины σ_i фиксированы, а настраиваются только синаптические веса w_i. При решении более сложных задач (распознавание образов, классификация и кластеризация и т.п.) во внимание принимаются все три множества параметров $c_i \in R^n,\ \sigma_i, w_i \in R^1,\ i = 1,2,3,\dots,h$. Однако при этом следует учитывать, что число базисных функций экспоненциально растет с размерностью входного пространства n.

Таким образом, в радиально-базисных сетях на первый план выступает так называемое «*проклятие размерности*», ограничивающее эффективность этих нейронных сетей в задачах с большим числом входных признаков.

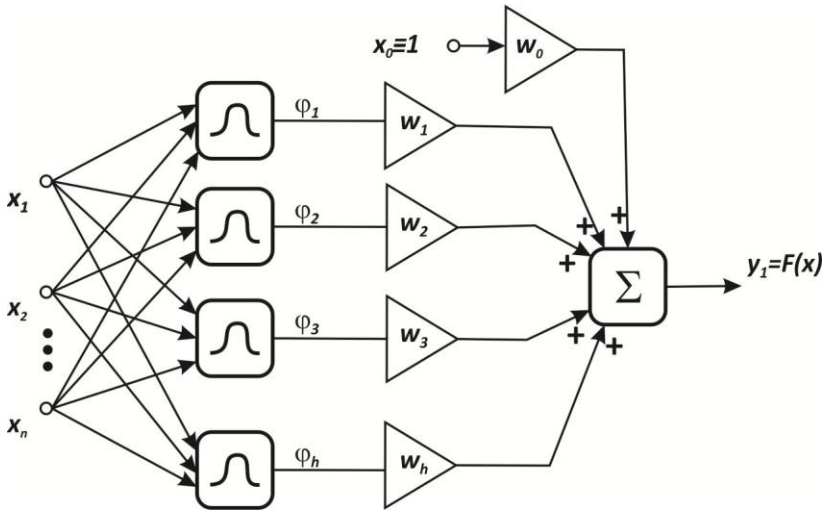


Рис. 33. Радиально-базисная сеть, реализующая отображение $y = F(x) = w_0 + \sum_{i=1}^{h} w_i \varphi_i(x)$

Кроме наиболее популярного гауссиана

$$\varphi(x) = \Phi_1(\|x - c\|, \sigma) = \Phi_1(r, \sigma) = \exp\left(-\frac{r^2}{\sigma^2}\right), \tag{1.53}$$

в радиально-базисных сетях используются функции [61, 50, 72], такие как

- мультиквадратичная: $\Phi_2(r, \sigma) = (r^2 + \sigma^2)^{1/2}, \tag{1.54}$
- обратная мультиквадратичная: $\Phi_3(r, \sigma) = (r^2 + \sigma^2)^{-\frac{1}{2}}, \tag{1.55}$
- сплайновая: $\Phi_4(r, \sigma) = \left(\frac{r}{\sigma}\right)^2 \log\left(\frac{r}{\sigma}\right) \tag{1.56}$

и хотя не все они имеют колоколообразную форму, например (1.54), (1.55), (1.56), их применение в задачах идентификации, моделирования, прогнозирования, распознавания образов, кластеризации и нейроуправления [50,51,61] оказалось вполне оправданным.

Как отмечалось выше, наиболее широкое распространение получила функция активации (1.53), расширить возможности которой можно, используя многомерный гауссиан

$$\varphi(x) = \Phi(\|x - c\|, \Sigma) = exp\big(-(x - c)^T \Sigma^{-1}(\mathrm{x} - \mathrm{c})\big) = \exp\big(-\|x - c\|^2_{\Sigma^{-1}}\big) \quad (1.57)$$

где ковариационная матрица Σ определяет форму, размер и ориентацию так называемого рецепторного поля радиально-базисной функции. При $\Sigma = \sigma^2 L$ (здесь L – $(n \times n)$ единичная матрица) рецепторное поле представляет гиперсферу с центром c и радиусом σ; при $\Sigma = diag(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2)$ - это гиперэллипсоид, чьи оси совпадают с осями входного пространства и имеют длину $2\sigma_i$ по i -й оси, и, наконец, при Σ – недиагональной положительно определенной матрице

$$\Sigma = \vartheta^T \Lambda \vartheta, \quad (1.58)$$

матрица собственных значений Λ определяет форму и размер рецепторного поля, а ортогональная матрица вращения ϑ – его ориентацию.

Радиально-базисные сети подобно многослойным искусственным нейронным сетям являются универсальными аппроксиматорами, однако в силу того, что в них присутствует только один нелинейный скрытый слой, а настраиваются параметры линейного выходного слоя, для их обучения могут быть использованы стандартные процедуры [24, 47], обладающие высоким быстродействием и фильтрующими свойствами, что крайне важно в задачах обработки «*зашумленных*» наблюдений.

Вместе с тем, объединение достоинств многослойных и радиально-базисных сетей может привести к весьма интересным результатам. Так в [52, 85] предложены архитектуры многослойных радиально-базисных сетей, подобные многослойным персептронам (см. рис. 14-15), где в качестве отдельных нейронов используются радиально-базисные сети с n входами и одним выходом (см. рис. 33). По сравнению с многослойными персептронами они обладают более высокой скоростью обучения, не страдая при этом от «*проклятия размерности*», ограничивающего возможности обычных радиально-базисных нейронных сетей.

Модификацией RBFN-сети является радиальная базисная *сеть с нулевой ошибкой.* Ее веса определяются таким образом, что обеспечивается нулевая ошибка по примерам обучающей выборки.

Другими модификациями являются, так называемые, *вероятностная нейронная сеть* и *обобщенно-регрессионная нейронная сеть.*

Вероятностная нейронная сеть (PNN). Нейронные сети PNN (Probabilistic Neural Networks) предназначены для решения вероятностных

задач и, в частности, задач классификации.

Архитектура сети PNN базируется на архитектуре радиальной базисной сети, но в качестве второго слоя используют так называемый конкурирующий слой, который подсчитывает вероятность принадлежности входного вектора к тому или иному классу, и, в конечном счете, сопоставляет вектор с тем классом, вероятность принадлежности к которому выше.

Предполагается, что задано обучающее множество, состоящее из N пар векторов вход/цель. Каждый вектор цели имеет m элементов, указывающих класс принадлежности, и, таким образом, входы ставятся в соответствие одному из m классов.

Наиболее важные преимущества PNN-сетей состоят в том, что выходное значение имеет вероятностный смысл (и поэтому его легче интерпретировать), и в том, что сеть быстро обучается. При обучении такой сети время тратится практически только на то, чтобы подавать ей на вход обучающие наблюдения, и сеть работает настолько быстро, насколько это вообще возможно.

Существенным недостатком таких сетей является их объем. PNN-сеть фактически вмещает в себя все обучающие данные, поэтому она требует много памяти и как следствие работает медленно.

PNN-сети особенно полезны при пробных экспериментах (например, когда нужно решить, какие из входных переменных использовать), так как благодаря короткому времени обучения можно быстро проделать большое количество пробных тестов.

Обобщенно-регрессионная нейронная сеть (GRNN). Данная сеть устроена аналогично вероятностной нейронной сети (PNN), но она предназначена для решения задач регрессии, а не классификации. Как и в случае PNN-сети, в точку расположения каждого обучающего наблюдения помещается гауссова ядерная функция. Мы считаем, что каждое наблюдение свидетельствует о некоторой степени уверенности в том, что поверхность отклика в данной точке имеет определенную высоту, и эта уверенность убывает при отходе в сторону от точки. GRNN-сеть копирует внутрь себя все обучающие наблюдения и использует их для оценки отклика в произвольной точке. Окончательная выходная оценка сети получается как взвешенное среднее выходов по всем обучающим наблюдениям:

$$y = \frac{\sum_{k=1}^{N} y^k \varphi\left(\frac{\left\|X - X^k\right\|}{\sigma}\right)}{\sum_{k=1}^{N} \varphi\left(\frac{\|X - X^k\|}{\sigma}\right)},$$

где X^k, y^k — точки обучающей выборки.

Первый промежуточный слой сети GRNN состоит из радиальных элементов. Второй промежуточный слой содержит элементы, которые помогают оценить взвешенное среднее. Каждый выход имеет в этом слое свой элемент, формирующий для него взвешенную сумму. Чтобы получить из

взвешенной суммы взвешенное среднее, эту сумму нужно поделить на сумму весовых коэффициентов. Последнюю сумму вычисляет специальный элемент второго слоя. После этого в выходном слое производится собственно деление (с помощью специальных элементов «деления»). Таким образом, число элементов во втором промежуточном слое на единицу больше, чем в выходном слое. Как правило, в задачах регрессии требуется оценить одно выходное значение, и, соответственно, второй промежуточный слой содержит два элемента.

Можно модифицировать GRNN-сеть таким образом, чтобы радиальные элементы соответствовали не отдельным обучающим случаям, а их кластерам. Это уменьшает размеры сети и увеличивает скорость обучения. Центры для таких элементов можно выбирать с помощью любого предназначенного для этой цели алгоритма (выборки из выборки, *K*-средних или Кохонена).

Достоинства и недостатки у сетей GRNN в основном такие же, как и у сетей PNN, — единственное различие состоит в том, что GRNN используются в задачах регрессии, а PNN — в задачах классификации. GRNN-сеть обучается почти мгновенно, но может получиться большой и медленной (хотя здесь, в отличие от PNN, не обязательно иметь цо одному радиальному элементу на каждый обучающий пример, их число все равно будет большим). Как и сеть RBF, сеть GRNN не обладает способностью экстраполировать данные.

1.5.8 Структура и назначение когнитрона, неокогнитрона и свёрточных нейронных сетей

Когнитрон. Когнитрон конструируется в виде слоев нейронов, соединенных синапсами. Как показано на рисунке 34, предсинаптический нейрон в одном слое связан с постсинаптическим нейроном в следующем слое [56]. Имеются два типа нейронов: возбуждающие узлы, которые стремятся вызвать возбуждение постсинаптического узла, и тормозящие узлы, которые тормозят это возбуждение.

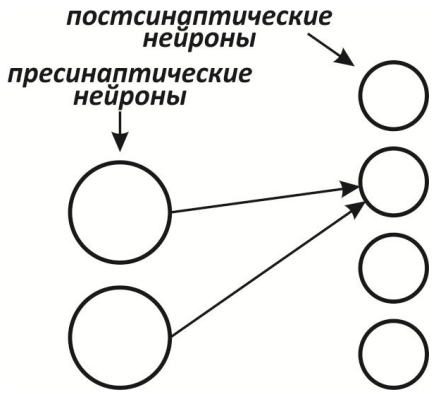


Рис. 34. Предсинаптические и постсинаптические нейроны

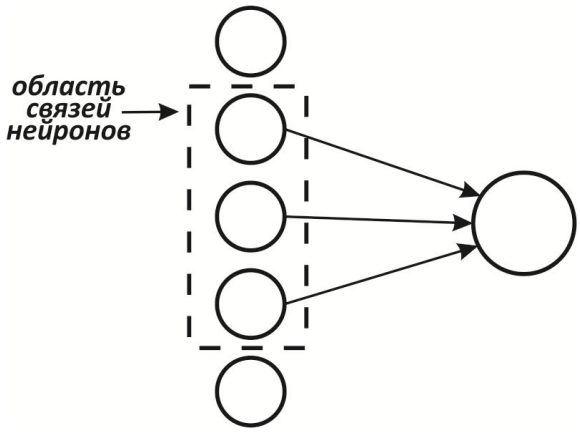


Рис. 35. Область связей нейрона

Возбуждение нейрона определяется взвешенной суммой его возбуждающих и тормозящих входов, однако в действительности механизм

является более сложным, чем простое суммирование.

На рисунке 35 показано, что каждый нейрон связан только с нейронами в соседней области, называемой «*областью связи*». Это ограничение области связи согласуется с анатомией зрительной коры, в которой редко соединяются между собой нейроны, располагающиеся друг от друга на расстоянии более одного миллиметра. В рассматриваемой модели нейроны упорядочены в виде слоев со связями от одного слоя к следующему. Это также аналогично послойной структуре зрительной коры и других частей головного мозга.

На рисунке 36 показано, что области связи соседних узлов значительно перекрываются. Это расточительное дублирование функций оправдывается взаимной конкуренцией между ближайшими узлами. Даже если узлы в начальный момент имеют абсолютно идентичный выход, небольшие отклонения всегда имеют место; один из узлов всегда будет иметь более сильную реакцию на входной образ, чем соседние. Его сильное возбуждение будет оказывать сдерживающее воздействие на возбуждение соседних узлов, и только его синапсы будут усиливаться; синапсы соседних узлов останутся неизменными.

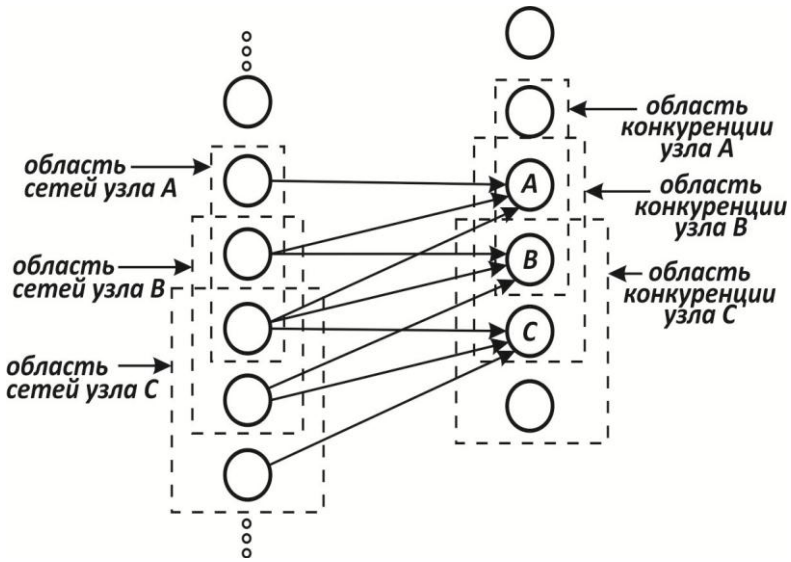


Рис. 36. Область связи с областью конкуренции

Суммарный возбуждающий вход в нейрон взвешенной суммой входов от возбуждающих предшествующем слое. Аналогично суммарный вход является взвешенной суммой входов от всех тормозящих нейронов. В символьном виде

$$E = \sum_i w_{\text{в}i} u_i, \qquad L = \sum_j w_{\text{т}j} v_j \ , \tag{1.59}$$

где $w_{\text{в}i}$ - вес i –го возбуждающего синапса, u_i – выход i-го возбуждающего нейрона, $w_{\text{т}j}$ – вес j-го тормозящего синапса, v_j – выход j-го тормозящего нейрона.

Заметим, что веса имеют только положительные значения. Выход

нейрона затем вычисляется следующим образом:

$$S = \frac{1+E}{1+L} - 1, \qquad \begin{cases} y = S, & \text{при } S \geq 0 \\ y = 0, & \text{при } S < 0 \end{cases} \tag{1.60}$$

Предполагая, что NET имеет положительное значение, это можно записать следующим образом:

$$y = \frac{E-L}{1+L} \tag{1.61}$$

Когда тормозящий вход мал ($L \ll 1$), y может быть аппроксимирован как

$$y = E - L \tag{1.62}$$

что соответствует выражению для обычного линейного порогового элемента (с нулевым порогом).

Алгоритм обучения когнитрона позволяет весам синапсов возрастать без ограничений. Благодаря отсутствию механизма уменьшения весов они просто возрастают в процессе обучения. В обычных линейных пороговых элементах это привело бы к произвольно большому выходу элемента. В когнитроне большие возбуждающие и тормозящие входы суммируются в ограничивающей формуле вида:

$$y = \frac{E}{L} - 1, \quad \text{если} \quad E \gg 1 \text{ и } L \gg 1. \tag{1.63}$$

В данном случае y определяется отношением возбуждающих входов к тормозящим входам, а не их разностью. Таким образом, величина y ограничивается, если оба входа возрастают в одном и том же диапазоне X. Предположив, что это так, E и L можно выразить следующим образом:

$$E = pX, \qquad L = qX, \qquad p, q - \text{константы}, \tag{1.64}$$

и после некоторых преобразований

$$y = \frac{p-q}{2q} \cdot \left[1 + th\left(\frac{\log(pq)}{2}\right)\right]. \tag{1.65}$$

Эта функция возрастает по закону Вебера-Фехнера, который часто используется для аппроксимации нелинейных соотношений входа/выхода нейронов. При использовании этого соотношения нейрон когнитрона в точности эмулирует реакцию биологических нейронов. Это делает его как мощным вычислительным элементом, так и точной моделью для моделирования.

Тормозящие нейроны. В когнитроне слой состоит из возбуждающих и тормозящих узлов. Как показано на рисунке 37, нейрон слоя 2 имеет область связи, для которой он имеет синаптические соединения с набором выходов нейронов в слое 1. Аналогично в слое 1 существует тормозящий нейрон, имеющий ту же область связи. Синаптические веса тормозящих узлов не

изменяются в процессе обучения; их веса заранее установлены таким образом, что сумма весов в любом из тормозящих нейронов равна единице. В соответствии с этими ограничениями, выход тормозящего узла Q является взвешенной суммой его входов, которые в данном случае представляют собой среднее арифметическое выходов возбуждающих нейронов, к которым он подсоединен.

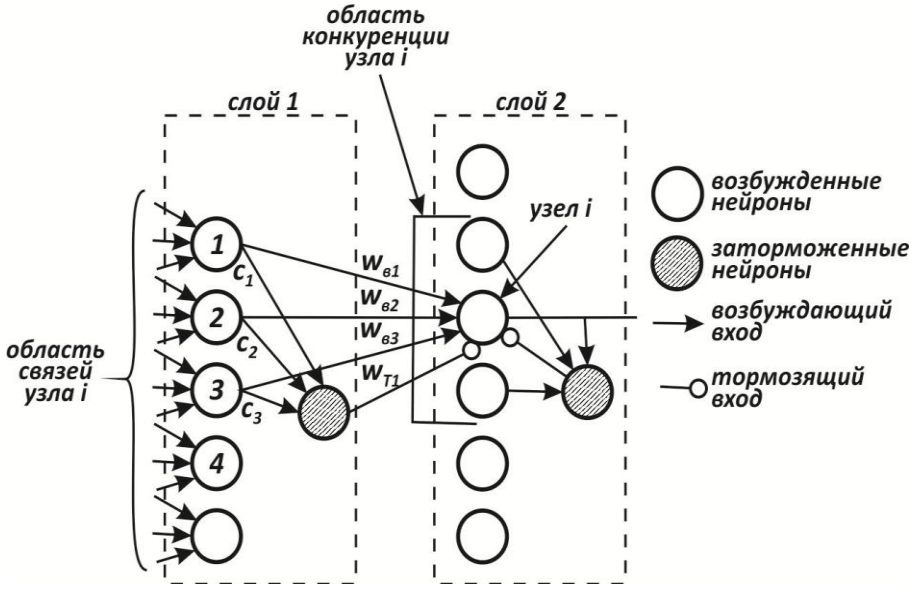


Рис. 37. Слои когнитрона

Таким образом,

$$Q = \sum_i c_i y_i, \tag{1.66}$$

где $\sum_i c_i$, c_i- возбуждающий вес i.

Процедура обучения. Веса возбуждающих нейронов изменяются только тогда, когда нейрон возбужден сильнее, чем любой из узлов в области конкуренции. Если это так, изменение в процессе обучения любого из его весов может быть определено следующим образом:

$$\delta w_{\text{в}i} = q c_j u_j, \tag{1.67}$$

где c_j – тормозящий вес связи нейрона j в слое 1 с тормозящим нейроном i, u_j – выход нейрона j в слое 1, $w_{\text{в}i}$ – возбуждающий вес i, q – нормирующий коэффициент обучения.

Изменение тормозящих весов нейрона i в слое 2 пропорционально отношению взвешенной суммы возбуждающих входов к удвоенному тормозящему входу. Вычисления проводятся по формуле

$$\delta w_{Ti} = \frac{q \sum_j w_{\text{в}j} u_j}{2 \cdot Q_i}. \tag{1.68}$$

Когда возбужденных нейронов в области конкуренции нет, для

изменения весов используются другие выражения. Это необходимо, поскольку процесс обучения начинается с нулевыми значениями весов; поэтому первоначально нет возбужденных нейронов ни в одной области конкуренции, и обучение производиться не может. Во всех случаях, когда победителя в области конкуренции нейронов нет, изменение весов нейронов вычисляется следующим образом:

$$\Delta w_{\text{в}i} = q^{\xi} c_j u_j, \quad \delta w_{Ti} = q^{\xi} Q, \tag{1.69}$$

где q^{ξ} – положительный обучающий коэффициент меньший, чем q.

Приведенная стратегия настройки гарантирует, что узлы с большой реакцией заставляют возбуждающие синапсы, которыми они управляют, увеличиваться сильнее, чем тормозящие синапсы. И наоборот, узлы, имеющие малую реакцию, вызывают малое возрастание возбуждающих синапсов, но большее возрастание тормозящих синапсов. Таким образом, если узел 1 в слое 1 имеет больший выход, синапс $w_{\text{в}1}$ возрастет больше, чем синапс w_{T1}. И наоборот, узлы, имеющие малый выход, обеспечат малую величину для приращения $w_{\text{в}i}$. Однако другие узлы в области связи будут возбуждаться, тем самым увеличивая сигнал Q и значения w_{Ti}.

В процессе обучения веса каждого узла в слое 2 настраиваются таким образом, что вместе они составляют шаблон, соответствующий образам, которые часто предъявляются в процессе обучения. При предъявлении сходного образа шаблон соответствует ему, и узел вырабатывает большой выходной сигнал. Сильно отличающийся образ вырабатывает малый выход и обычно подавляется конкуренцией.

На рисунке 37 показано, что каждый нейрон слоя 2 получает латеральное торможение от нейронов [56], расположенных в его области конкуренции. Тормозящий нейрон суммирует входы от всех нейронов в области конкуренции и вырабатывает сигнал, стремящийся к торможению целевого нейрона. Этот метод является эффектным, но с вычислительной точки зрения медленным. Он охватывает большую систему с обратной связью, включающую каждый нейрон в слое; для его стабилизации может потребоваться большое количество вычислительных итераций.

Для ускорения вычислений в работе [56] используется метод ускоренного латерального торможения (рис. 38). Здесь дополнительный узел латерального торможения обрабатывает выход каждого возбуждающего узла для моделирования требуемого латерального торможения. Сначала он определяет сигнал, равный суммарному тормозящему влиянию в области конкуренции:

$$Q_{lat} = \sum_i q_i y_i \,, \tag{1.70}$$

где y_i - выход i-го нейрона в области конкуренции, q_i – вес связи от этого нейрона к латерально-тормозящему нейрону; q_i выбраны таким образом, что $\sum_i q_i = 1$.

Выход тормозящего нейрона $y^{'}$ затем вычисляется следующим образом:

$$y^{i} = \frac{1 + y_i}{1 + Q_{lat}} - 1. \qquad (1.71)$$

Благодаря тому что все вычисления, связанные с таким типом латерального торможения, являются нерекурсивными, они могут быть проведены за один проход для слоя, тем самым определяя эффект в виде большой экономии в вычислениях.

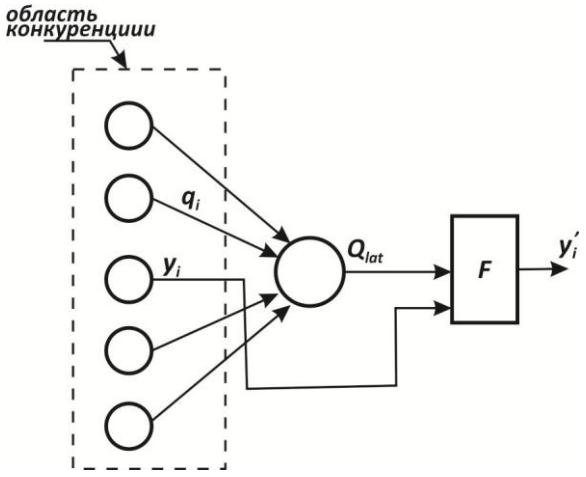


Рис. 38. Ускоренное торможение

Этот метод латерального торможения решает и другую сложную проблему. Предположим, что узел в слое 2 возбуждается сильно, но возбуждение соседних узлов уменьшается постепенно с увеличением расстояния. При использовании обычного латерального торможения будет обучаться только центральный узел. Другие узлы определяют, что центральный узел в их области конкуренции имеет более высокий выход. С предлагаемой системой латерального торможения такой ситуации случиться не может. Множество узлов может обучаться одновременно и процесс обучения является более достоверным.

Неокогнитрон. В попытках улучшить когнитрон была разработана мощная парадигма, названная *неокогнитрон* [58,55]. Неокогнитрон ориентирован на моделирование зрительной системы человека. Он получает на входе двумерные образы, аналогичные изображениям на сетчатой оболочке глаза, и обрабатывает их в последующих слоях аналогично тому, как это было обнаружено в зрительной коре человека [63,64,65]. Конечно, в неокогнитроне нет ничего, ограничивающего его использование только для обработки визуальных данных, он достаточно универсален и может найти широкое применение как обобщенная система распознавания образов.

В зрительной коре были обнаружены узлы, реагирующие на такие элементы, как линии и углы определенной ориентации. На более высоких уровнях узлы реагируют на более сложные и абстрактные образы такие, как

окружности, треугольники и прямоугольники. На еще более высоких уровнях степень абстракции возрастает до тех пор, пока не определятся узлы, реагирующие на лица и сложные формы. В общем случае узлы на более высоких уровнях получают вход от группы низкоуровневых узлов и, следовательно, реагируют на более широкую область визуального поля. Реакции узлов более высокого уровня менее зависят от позиции и более устойчивы к искажениям [59].

Неокогнитрон имеет иерархическую структуру, ориентированную на моделирование зрительной системы человека. Он состоит из последовательности обрабатывающих слоев, организованных в иерархическую структуру (рис. 39). Входной образ подается на первый слой и передается через плоскости, соответствующие последующим слоям, до тех пор, пока не достигнет выходного слоя, в котором идентифицируется распознаваемый образ.

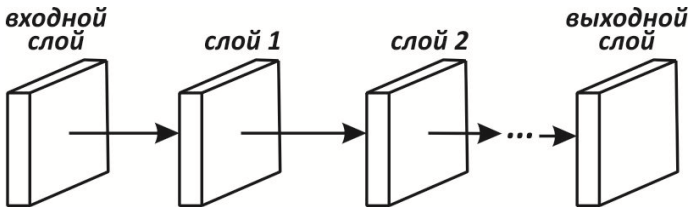


Рис. 39. Структура слоёв неокогнитрона

Структура неокогнитрона трудна для представления в виде диаграммы, но концептуально проста. Чтобы подчеркнуть его многоуровневость (с целью упрощения графического представления), используется анализ верхнего уровня. Неокогнитрон показан, состоящим из слоев, слои состоят из набора плоскостей и плоскости состоят из узлов.

Слои. Каждый слой неокогнитрона состоит из двух массивов плоскостей (рис. 40). Массив плоскостей, содержащих простые узлы, получает выходы предыдущего слоя, выделяет определенные образы и затем передает их в массив плоскостей, содержащих комплексные узлы, где они обрабатываются таким образом, чтобы сделать выделенные образы менее позиционно зависимыми.

Плоскости. Внутри слоя плоскости простых и комплексных узлов существуют парами, т. е. для плоскости простых узлов существует одна плоскость комплексных узлов, обрабатывающая ее выходы. Каждая плоскость может быть визуально представлена как двумерный массив узлов.

Простые узлы. Все узлы в данной плоскости простых узлов реагируют на один и тот же образ. Как показано на рисунке 41, плоскость простых узлов представляет массив узлов, каждый из которых «настраивается» на один специфический входной образ. Каждый простой узел чувствителен к ограниченной области входного образа, называемой его рецептивной областью. Например, все узлы в верхней плоскости простых узлов на рисунке 41

реагируют на «C». Узел реагирует, если «C» встречается во входном образе и если «C» обнаружено в его рецептивной области.

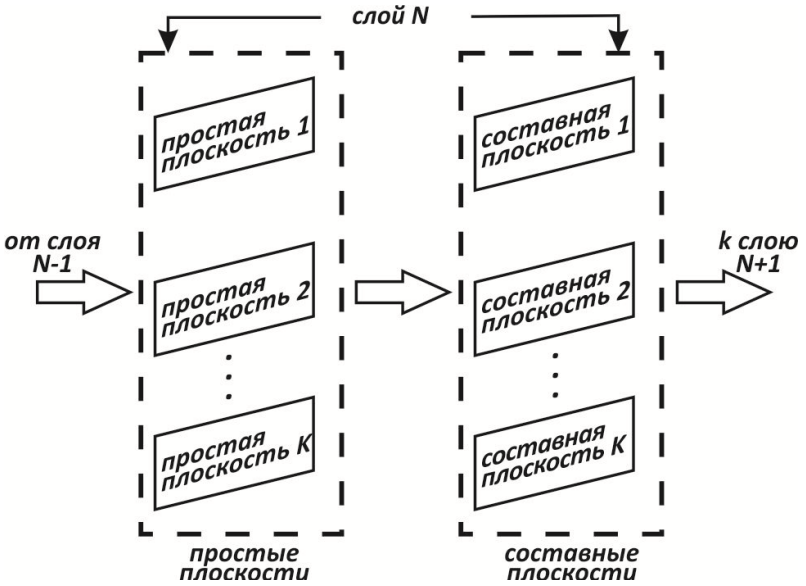


Рис. 40. Структура плоскостей неокогнитрона

На рисунке 41 показано, что другие плоскости простых узлов в этом слое могут реагировать на поворот «C» на 90°, другие на поворот на 180° и т. д. Если должны быть выделены другие буквы (и их искаженные версии), дополнительные плоскости требуются для каждой из них.

Рецептивные области узлов в каждой плоскости простых узлов перекрываются с целью покрытия всего входного образа этого слоя. Каждый узел получает входы от соответствующих областей всех плоскостей комплексных узлов в предыдущем слое. Следовательно, простой узел реагирует на появление своего образа в любой сложной плоскости предыдущего слоя, если он окажется внутри его рецептивной области.

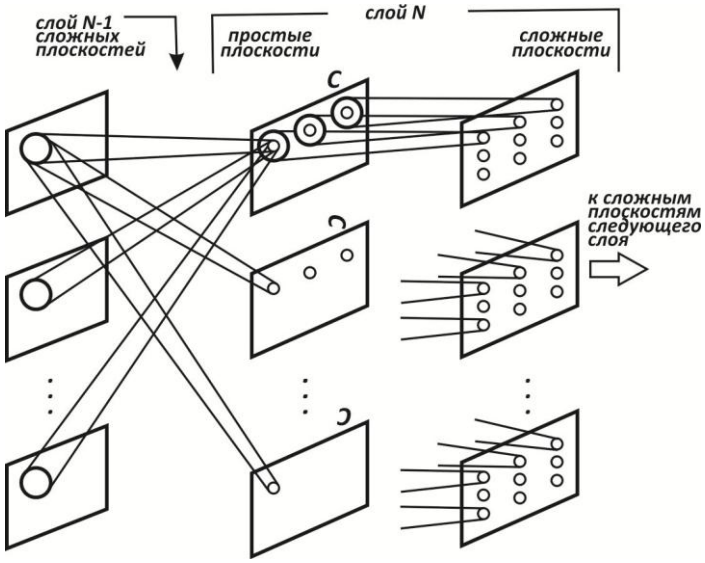


Рис. 41. Система неокогнитрона

Комплексные узлы. Задачей комплексных узлов является уменьшение зависимости реакции системы от позиции образов во входном поле. Для достижения этого каждый комплексный узел получает в качестве входного образа выходы набора простых узлов из соответствующей плоскости того же слоя. Эти простые узлы покрывают непрерывную область простой плоскости, называемую рецептивной областью комплексного узла. Возбуждение любого простого узла в этой области является достаточным для возбуждения данного комплексного узла. Таким образом, комплексный узел реагирует на тот же образ, что и простые узлы в соответствующей ему плоскости, но он менее чувствителен к позиции образа, чем любой из них.

Каждый слой комплексных узлов реагирует на более широкую область входного образа, чем это делалось в предшествующих слоях. Эта прогрессия возрастает линейно от слоя к слою, приводя к требуемому уменьшению позиционной чувствительности системы в целом.

Каждый нейрон в слое, близком к входному, реагирует на определенные образы в определенном месте, такие, как угол с определенной ориентацией в заданной позиции. Каждый слой в результате этого имеет более абстрактную, менее специфичную реакцию по сравнению с предшествующим; выходной слой реагирует на полные образы, показывая высокую степень независимости от их положения, размера и ориентации во входном поле. При использовании в качестве классификатора комплексный узел выходного слоя с наибольшей реакцией реализует выделение соответствующего образа во входном поле. В идеальном случае это выделение нечувствительно к позиции, ориентации, размерам или другим искажениям.

Тормозящий узел вырабатывает выход, пропорциональный квадратному корню из взвешенной суммы квадратов его входов. Заметим, что входы в тормозящий узел идентичны входам соответствующего простого узла и область включает область ответа во всех комплексных плоскостях. В символьном виде

$$v = \sqrt{\sum_i (w_{Ti} u_i)^2}\,, \tag{1.72}$$

где v – выход тормозящего узла; i – область над всеми комплексными узлами, с которыми связан тормозящий узел; w_{Ti} – вес i-й синаптической связи от комплексного узла к тормозящему узлу; u_i– выход i-го комплексного узла.

Веса w_{Ti} выбираются монотонно уменьшающимися с увеличением расстояния от центра области реакции, при этом сумма их значений должна быть равна единице.

Только простые узлы имеют настраиваемые веса. Это веса связей, соединяющих узел с комплексными узлами в предыдущем слое и имеющих изменяемую силу синапсов, настраиваемую таким образом, чтобы выработать максимальную реакцию на определенные стимулирующие свойства. Некоторые из этих синапсов являются возбуждающими и стремятся увеличить выход

узлов, в то время как другие являются тормозящими и уменьшают выход узла.

На рисунке 42 показана полная структура синаптических связей между простым узлом и комплексными узлами в предшествующем слое. Каждый простой узел реагирует только на набор комплексных узлов внутри своей рецептивной области. Кроме того, существует тормозящий узел, реагирующий на те, же самые комплексные узлы. Веса синапсов тормозящего узла не обучаются, – они выбираются таким образом, чтобы узел реагировал на среднюю величину выходов всех узлов, к которым он подключен. Единственный тормозящий синапс от тормозящего узла к простому узлу обучается, как и другие синапсы.

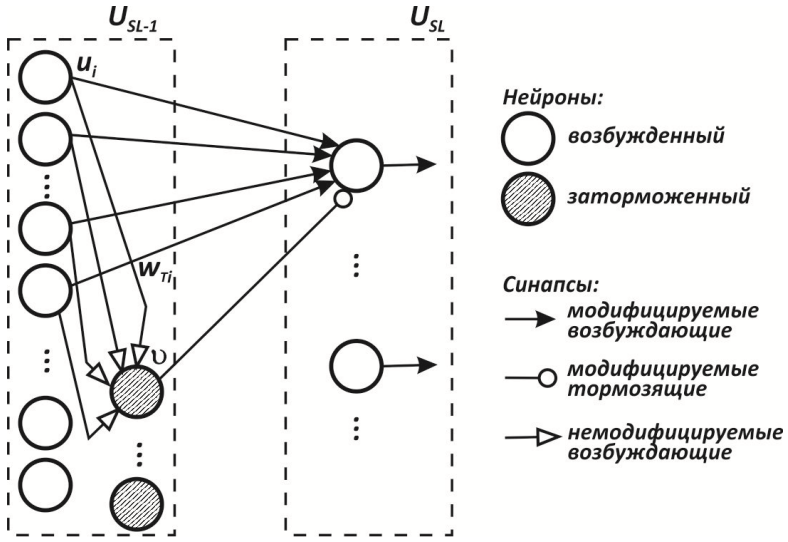


Рис. 42. Синапсы сложных нейронов одного уровня к простым нейронам следующего уровня

Обучение без учителя. Для обучения неокогнитрона на вход сети подается образ, который необходимо распознать, и веса синапсов настраиваются слой за слоем, начиная с набора простых узлов, ближайших к входу. Величина синаптической связи от каждого комплексного узла к данному простому узлу увеличивается тогда и только тогда, когда удовлетворяются следующие два условия:

- комплексный узел реагирует;
- простой узел реагирует более сильно, чем любой из его соседних узлов.

Таким образом, простой узел обучается реагировать более сильно на образы, появляющиеся наиболее часто в его рецептивной области, что соответствует результатам исследований, полученных в экспериментах с котятами. Если распознаваемый образ отсутствует на входе, тормозящий узел предохраняет от случайного возбуждения.

Математическое описание процесса обучения и метод реализации латерального торможения аналогичны описанным, для когнитрона, поэтому здесь они не повторяются. Необходимо отметить, что выходы простых и

комплексных узлов являются аналоговыми, непрерывными и линейными и что алгоритм обучения предполагает их неотрицательность.

Когда выбирается простой узел, веса синапсов которого должны быть увеличены, он рассматривается как представитель всех узлов в плоскости, вызывая увеличение их синаптических связей на том же самом образе. Таким образом, все узлы в плоскости обучаются распознавать одни и те же свойства, и после обучения будут делать это независимо от позиции образа в поле комплексных узлов в предшествующем слое.

Эта система имеет ценную способность к самовосстановлению. Если данный узел выйдет из строя, будет найден другой узел, реагирующий более сильно, и этот узел будет обучен распознаванию входного образа, тем самым перекрывая действия своего отказавшего товарища.

Обучение с учителем. В работах [55,57] описано самоорганизующееся неуправляемое обучение. Наряду с этими впечатляющими результатами, были опубликованы отчеты о других экспериментах, использующих обучение с учителем [59]. Здесь требуемая реакция каждого слоя заранее определяется экспериментатором. Затем веса настраиваются с использованием обычных методов для выработки требуемой реакции. Например, входной слой настраивался для распознавания отрезков линий в различных ориентациях во многом аналогично первому слою обработки. Последующие слои обучались реагировать на более сложные и абстрактные свойства до тех пор, пока в выходном слое требуемый образ не будет выделен.

Как когнитрон, так и неокогнитрон производят большое впечатление с точки зрения точности, с которой они моделируют биологическую нервную систему. Неокогнитрон является сложной системой и требует существенных вычислительных ресурсов. Несмотря на то, что многие подходы, казавшиеся нереализуемыми несколько лет назад, являются общепринятыми сегодня и могут оказаться тривиальными через несколько лет, реализация моделей неокогнитрона на универсальных компьютерах является бесперспективной. Необходимо достигнуть тысячекратных улучшений стоимости и производительности компьютеров за счет специализации архитектуры и внедрения технологии больших информационных систем, чтобы сделать неокогнитрон практической системой для решения сложных проблем распознавания образов, однако ни эта, ни какая-либо другая модель искусственных нейронных сетей не должны отвергаться только на основании их высоких вычислительных требований.

2. ВВЕДЕНИЕ В ТЕОРИЮ НЕЧЕТКОЙ ЛОГИКИ

2.1. Нечёткие знания и нечёткая информация

Пожалуй, наиболее поразительным свойством человеческого интеллекта является способность принимать правильные решения в обстановке неполной и нечёткой информации. Построение моделей приближённых рассуждений человека и использование их в интеллектуальных компьютерных системах представляет одно из самых перспективных направлений развития современной вычислительной техники.

При попытке формализовать человеческие знания исследователи вскоре столкнулись с проблемой, затруднявшей использование традиционного математического аппарата для их описания. Существует целый класс описаний, оперирующих качественными характеристиками объектов (*много, мало, сильный, очень сильный* и т.п.). Эти характеристики обычно размыты и не могут быть однозначно интерпретированы, однако содержат важную информацию (например, «Одним из возможных признаков гриппа является *высокая* температура»).

Кроме того, в задачах, решаемых интеллектуальными системами, часто приходится пользоваться неточными знаниями, которые не могут быть интерпретированы как полностью истинные или ложные, (логические *true/false* или $0/1$). Существуют знания, достоверность которых выражается некоторой промежуточной цифрой, например $0{,}7$.

Как, не разрушая свойства размытости и неточности, представлять подобные знания формально? Для разрешения таких проблем, в начале, 70-х годов американский математик Лотфи Заде предложил формальный аппарат нечёткой алгебры и нечёткой логики [4]. Его работа, опубликованная в 1965г., явилась толчком к развитию новой математической теории. Л.Заде расширил классическое понятие множества, допустив, что характеристическая функция (функция принадлежности элемента множеству) может принимать любые значения в интервале $(0{,}1)$, а не только значения 0 либо 1. Такие множества были названы им нечёткими**.** Л.Заде определил также ряд операций над нечёткими множествами и предложил обобщение известных методов логического вывода ***modus ponens*** и ***modus tollens*.**

Позднее это направление получило широкое распространение и положило начало одной из ветвей искусственного интеллекта под названием «мягкие вычисления». Л.Заде ввёл одно из главных понятий в нечёткой логике — понятие лингвистической переменной.

Лингвистическая переменная (ЛП) — это переменная, значение которой определяется набором вербальных (то есть словесных) характеристик некоторого свойства.

Например, ЛП *«рост»* определяется через набор *{карликовый, низкий, средний, высокий, очень высокий}*.

Введя понятие *лингвистической переменной* и допустив, что в качестве её значений (термов) выступают нечёткие множества, Л.Заде предложил аппарат для описания процессов интеллектуальной деятельности, включая нечёткость и неопределённость выражений. Это позволило создать фундамент теории нечётких множеств и нечёткой логики, а также предпосылки для внедрения методов нечёткого управления в инженерную практику.

Смещение центра исследований нечётких систем в сторону практических приложений привело к постановке целого ряда проблем, таких как новые архитектуры компьютеров для нечётких вычислений, элементная база нечётких компьютеров и контроллеров, инструментальные средства разработки, инженерные методы расчёта и разработки нечётких систем управления и многое другое.

Математическая теория нечётких множеств позволяет описывать нечёткие понятия и знания, оперировать этими знаниями и делать нечёткие выводы. Нечёткое управление оказывается особенно полезным, когда исследуемые процессы являются слишком сложными для анализа с помощью общепринятых методов, или когда доступные источники информации интерпретируются некачественно, неточно или неопределённо. Нечёткая логика, предоставляющая эффективные средства отображения неопределённостей и неточностей реального мира, и на которой основано нечёткое управление, ближе к человеческому мышлению и естественным языкам, чем традиционные логические системы.

2.2. Основы теории нечётких множеств

Значения лингвистической переменной (ЛП) определяются через так называемые *нечёткие множества* (НМ), которые в свою очередь определены на некотором базовом наборе значений или базовой числовой шкале, имеющей размерность. Каждое значение ЛП определяется как нечёткое множество (например, НМ *«низкий рост»*).

Нечёткое множество определяется через некоторую базовую шкалу B и функцию принадлежности НМ — $\mu(x), x \in B$, принимающую значения на интервале $[0\ldots1]$. Таким образом, нечёткое множество B — это совокупность пар вида $\big(x, \mu(x)\big)$, где $x \in B$. Часто встречается и такая запись:

$$B = \sum_{i=1}^{n} \frac{x_i}{\mu(x_i)}, \tag{2.1}$$

где $x_i - i -$ е значение базовой шкалы.

Функция принадлежности определяет субъективную степень уверенности эксперта в том, что данное конкретное значение базовой шкалы соответствует определяемому НМ. Эту функцию не стоит путать с вероятностью, носящей объективный характер и подчиняющейся другим математическим

зависимостям. Например, для двух экспертов определение НМ *«высокая»* для ЛП *«цена автомобиля»* в условных единицах может существенно отличаться в зависимости от их социального и финансового положения.

Пример 2.1. Пусть перед нами стоит задача интерпретации значений ЛП *«возраст»*, таких как *«молодой»* возраст, *«преклонный»* возраст или *«переходный»* возраст. Определим *«возраст»* как ЛП (рис. 43). Тогда *«молодой»*, *«преклонный»*, *«переходный»* будут значениями этой лингвистической переменной. Более полно базовый набор значений ЛП *«возраст»* следующий: B — {*младенческий, детский, юный, молодой, зрелый, преклонный, старческий*}.

Рис. 43. Лингвистическая переменная «возраст» и нечеткие множества, определяющие ее значения

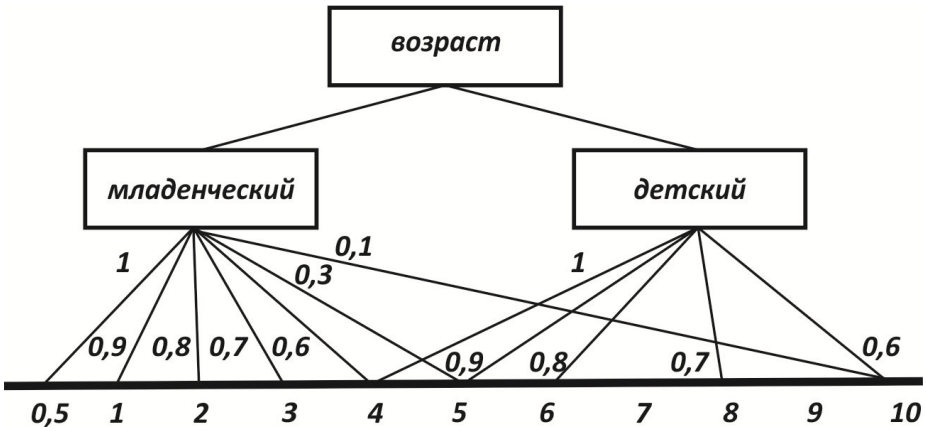


Рис.44. Формирование нечетких множеств

Для ЛП *«возраст»* базовая шкала — это числовая шкала от 0 до 120, обозначающая количество прожитых лет, а функция принадлежности определяет, насколько есть уверенность в том, что данное количество лет можно отнести к данной категории возраста. На рисунке 44 отражено, как одни и те же значения базовой шкалы могут участвовать в определении различных НМ.

Например, определить значение НМ *«младенческий возраст»* можно так:

$$\text{«младенческий»} = \left\{\frac{0{,}5}{1} + \frac{1}{0{,}9} + \frac{2}{0{,}8} + \frac{3}{0{,}7} + \frac{4}{0{,}6} + \frac{5}{0{,}3} + \frac{10}{0{,}1}\right\}.$$

Рисунок 44 иллюстрирует оценку НМ неким усредненным экспертом, который ребёнка до полугода с высокой степенью уверенности относит к младенцам ($m = 1$). Дети до четырёх лет причисляются к младенцам тоже, но с меньшей степенью уверенности ($0.5 < m < 0.9$), а в десять лет ребёнка называют так только в очень редких случаях — к примеру, для девяностолетней бабушки и 15 лет может считаться младенчеством. Таким образом, нечёткие множества позволяют при определении понятия учитывать субъективные мнения отдельных индивидуумов.

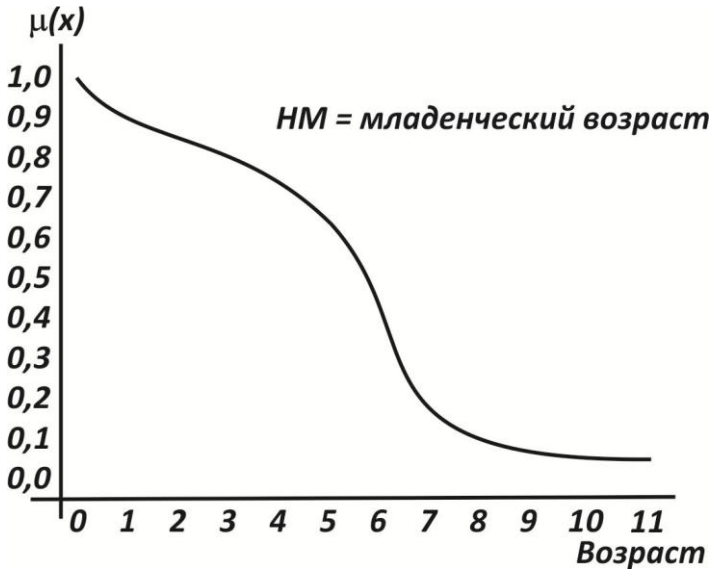


Рис. 45. График функции принадлежности нечеткому множеству «младенческий возраст»

Основные характеристики нечётких множеств.

Пусть $M = [0,1]$ и A — нечёткое множество с элементами из универсального множества E и множеством принадлежностей M.

- Величина $sup_{x \in U}\, \mu_A(x)$, называется *высотой* нечёткого множества A. Нечёткое множество A *нормально,* если его высота равна 1, т.е. верхняя граница его функции принадлежности равна $sup_{x \in U}\, \mu_A(x) = 1$. При $sup_{x \in U}\, \mu_A(x) < 1$ нечёткое множество называется *субнормальным*.
- Нечёткое множество *пусто*, если $\forall x \in E \mu_A(x) = 0$. Непустое субнормальное множество можно нормализовать по формуле:

$$\mu_A(x) = \frac{\mu_A(x)}{sup_{x \in U}\, \mu_A(x)}.$$

- Нечёткое множество *унимодально*, если $\mu_A(x) = 1$ только на одном x из E.
- Носителем нечёткого множества A является обычное подмножество со свойством $\mu_A(x) > 0$, т.е. носитель

$$A = \left\{ {}^{x}/_{\mu_A(x) > 0} \right\}, \forall x \in E.$$

- Элементы $x \in E$, для которых $\mu_A(x) = 0{,}5$, называются точками перехода множества A.

Примеры нечетких множеств.

Пример 2.2. Пусть $E = \{0,1,2,\dots,10\}, M = [0,1]$. Нечеткое множество «несколько» можно определить следующим образом: «несколько» $= 0.5/3 + 0.8/4 + 1/5 + 1/6 + 0.8/7 + 0.5/8$; его характеристики: высота $= 1$, носитель $= \{3,4,5,6,7,8\}$, точки перехода – $\{3,8\}$.

Пример 2.3. Пусть $E = \{0,1,2,\dots,n,\dots\}$. Нечеткое множество «малый» можно определить:

$$\text{«малый»} = \mu_{\text{малый}}(n) = \frac{1}{1 + \left(\frac{n}{10}\right)^2} \Big/ n.$$

Пример 2.4. Пусть $E = \{0,1,2,\dots,100\}$ и соответствует понятию «возраст», тогда нечеткое множество «молодой» может быть определено следующим образом:

$$\mu_{\text{молодой}}(x) = \begin{cases} 1, x \in [1,25] \\ \dfrac{}{1 + \left(\dfrac{x-25}{5}\right)^2}, x > 25 \end{cases}.$$

Нечеткое множество «молодой» на универсальном множестве $E' = \{\text{Галкин}, \text{Уткин}, \text{Воробьев}, \dots\}$ задается с помощью функции принадлежности $\mu_{\text{молодой}}(x)$ на $E = \{0,1,2,\dots,100\}$ (возраст), называемым по отношению к E' функцией совместимости, при этом:

$$\mu_{\text{молодой}}(\text{Воробьев}) = \mu_{\text{молодой}}(x),$$

где x – возраст Воробьева.

Пример 2.5. Пусть $E = \{\text{Ока}, \text{Жигули}, \text{Мерседес}, \dots\}$ - множество марок автомобилей, а $E' = [0, \infty)$ – универсальное множество «стоимость», тогда на E' можно определить нечеткие множества типа: «для малоимущих», «для среднего класса», «престижные», с функциями принадлежности, как показано на рисунке 46.

Имея эти функции и зная стоимости автомобилей из E в данный момент времени, можно определить на E' нечёткие множества с этими же названиями.

Так, например, нечёткое множество «для малоимущих», заданное на универсальном множестве $E = \{\text{Ока}, \text{Жигули}, \text{Мерседес}, \dots\}$, выглядит так, как показано на рисунке 47.

Аналогично можно определить нечеткое множество «скоростные», «средние», «тихоходные» и т.д.

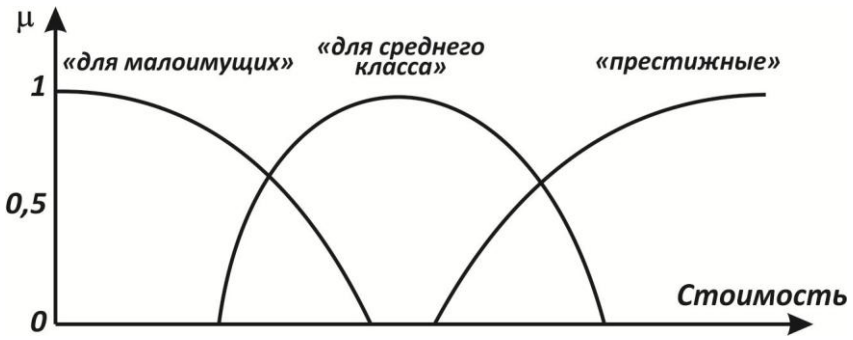


Рис. 46. Примеры функции принадлежности

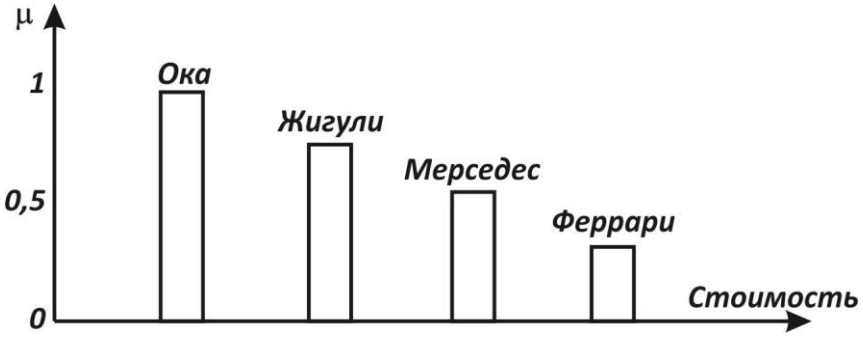


Рис. 47. Пример задания нечеткого множества

Методы построения функции принадлежности нечетких множеств.

Существуют прямые и косвенные методы построения функций принадлежности.

При использовании *прямых методов* эксперт просто задаёт для каждого $x \in E$ значение $\mu_A(x)$. Как правило, прямые методы задания функции принадлежности используются для измеримых понятий, таких как скорость, время, расстояние, давление, температура и т.д., или когда выделяются полярные значения.

Во многих задачах при характеристике объекта можно выделить набор признаков и для каждого из них определить полярные значения, соответствующие значениям функции принадлежности, 0 или 1. Для конкретного объекта эксперт, исходя из приведённой шкалы; задаёт $\mu_A(x) \in [0,1]$, формируя векторную функцию принадлежности $\{\mu_A(x_1), \mu_A(x_2), \dots, \mu_A(x_n)\}$.

Разновидностью прямых методов построения функций принадлежности являются *прямые групповые методы,* когда, например, группе экспертов предъявляют конкретный объект, и каждый должен дать один из двух ответов: принадлежит или не принадлежит этот объект к заданному множеству. Тогда число утвердительных ответов, делённое на общее число экспертов, даёт значение функции принадлежности объекта к представленному нечёткому множеству.

Косвенные методы определения значений функции принадлежности используются в случаях, когда нет измеримых элементарных свойств, через

которые определяется нечёткое множество. Как правило, это методы попарных сравнений. Если бы значения функций принадлежности были известны, например, $\mu_A(x_i) = w_i, \quad i = 1,2,\dots,n,$ то попарные сравнения можно представить матрицей отношений $A = \{a_{ij}\}$, где $a_{ij} = w_i/w_j$ (операция деления).

На практике эксперт сам формирует матрицу A, при этом предполагается, что диагональные элементы равны 1, а для элементов, симметричных относительно главной диагонали, $a_{ij} = 1/a_{ij}$, т.е. если один элемент оценивается в α раз значимее чем другой, то этот последний должен быть в $1/\alpha$ раз значимее, чем первый. В общем случае задача сводится к поиску вектора w, удовлетворяющего уравнению вида $Aw = \lambda_{max} w$, где λ_{max} - наибольшее собственное значение матрицы A Поскольку матрица A положительна по построению, решение данной задачи существует и является положительным.

Использование типовых форм кривых для задания функций принадлежности с уточнением их параметров в соответствии с данными эксперимента.

Использование относительных частот по данным эксперимента в качестве значений принадлежности.

2.3. Операции над нечеткими множествами

Включение. Пусть A и B - нечеткие множества на универсальном подмножестве E. Тогда A содержится в B, если $\forall x \in E\ \mu_A(x) \le \mu_B(x)$.

Обозначение: $A \subset B$.

Иногда используется термин «доминирование», т.е. в случае когда $A \subset B$, говорят, что B доминирует A.

Равенство. A и B равны, если $\forall x \in E\ \mu_A(x) = \mu_B(x)$.

Обозначение: $A = B$.

Дополнение. Пусть $M = [0,1]$, A и B – нечеткие множества, заданные на E, A и B дополняют друг друга, если $\forall x \in E\ \mu_A(x) = 1 - \mu_B(x)$.

Обозначение: $B = \overline{A}$ или $A = \overline{B}$.

Очевидно, что $\left(\overline{\overline{A}}\right) = A$, (дополнение определено для $M = [0,1]$, но очевидно, что его можно определить для любого упорядоченного M).

Пересечение. $A \cap B$ – наибольшее нечеткое подмножество, содержащееся одновременно в A и B:

$$\mu_{A\cap B}(x) = min\big(\mu_A(x), \mu_B(x)\big).$$

Объединение. $A \cup B$ – наименьшее нечеткое подмножество, включающее как A, так и B, с функцией принадлежности:

$$\mu_{A\cup B}(x) = max\big(\mu_A(x), \mu_B(x)\big).$$

Разность. $A - B = A \cap \overline{B}$ с функцией принадлежности:

$$\mu_{A-B}(x) = \mu_{A\cap\overline{B}}(x) = min\big(\mu_A(x), 1 - \mu_B(x)\big).$$

Дизъюнктивная сумма. $A \oplus B = (A - B) \cup (B - A) = \big(A \cap \overline{B}\big) \cup (\overline{A} \cap B)$ с функцией принадлежности:

$$\mu_{A-B}(x) = max\{[min\{\mu_A(x), 1 - \mu_B(x)\}]; [min\{1 - \mu_A(x), \mu_B(x)\}]\}.$$

Наглядное представление логических операций над нечёткими множествами.

Для нечётких множеств можно строить визуальное представление. Рассмотрим прямоугольную систему координат, на оси ординат которой откладываются значения $\mu_A(x)$, на оси абсцисс в произвольном порядке расположены элементы E. Если E по своей природе упорядочено, то этот порядок желательно сохранить в расположении элементов на оси абсцисс. Такое представление делает наглядными простые логические операции над нечёткими множествами (рис. 48).

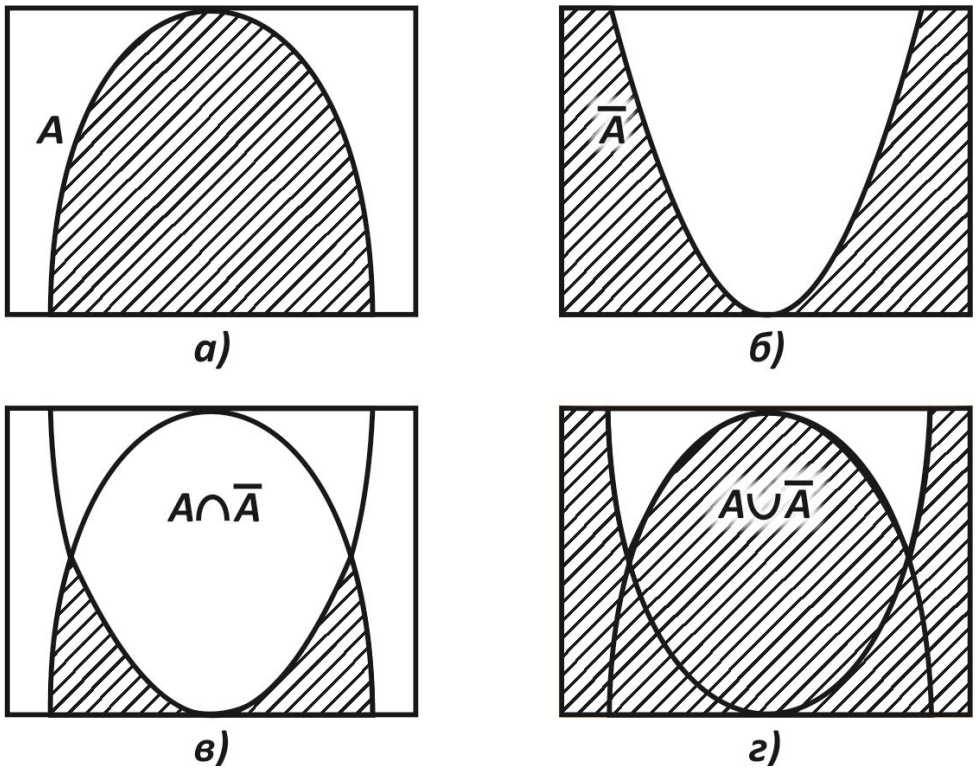


Рис. 48. Графическая интерпретация нечетких логических операций:
а – нечеткое множество A; б – $\overline{A}$; в – $A \cap \overline{A}$; г – $A \cup \overline{A}$.

Свойства операций объединения и пересечения.

Пусть A, B, C – нечеткие множества, тогда выполняются следующие свойства:

- $\left.\begin{matrix} A \cap B = B \cap A \\ A \cup B = B \cup A \end{matrix}\right\}$ – коммутативность;
- $\left.\begin{matrix} (A \cap B) \cap C = A \cap (B \cap C) \\ (A \cup B) \cup C = A \cup (B \cup C) \end{matrix}\right\}$ – ассоциативность;
- $\left.\begin{matrix} A \cap A = A \\ A \cup A = A \end{matrix}\right\}$ – идемпотентность;
- $\left.\begin{matrix} A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \end{matrix}\right\}$ – дистрибутивность;
- $\begin{matrix} A \cup \emptyset = A \\ A \cup \emptyset = \emptyset \end{matrix}$ – где $\emptyset$ – пустое множество, т.е. $\mu_i(x) = 0 \; \forall x \in E$;
- $\begin{matrix} A \cap E = A \\ A \cup E = E \end{matrix}$ – где E – универсальное множество;
- $\left.\begin{matrix} \overline{A \cap B} = \overline{A} \cup \overline{B} \\ \overline{A \cup B} = \overline{A} \cap \overline{B} \end{matrix}\right\}$ – формулы де Моргана.

В отличие от чётких множеств, для нечётких множеств в общем случае:

$$A \cap \overline{A} \neq \emptyset,$$
$$A \cup \overline{A} \neq E.$$

Введённые выше операции над нечёткими множествами основаны на использовании операций *max* и *min*. В теории нечётких множеств рассмотрены вопросы построения обобщённых, параметризованных операторов пересечения, объединения и дополнения, позволяющих учесть разнообразные смысловые оттенки соответствующих им связок «И», «ИЛИ», «НЕ».

Один из подходов к обобщению операторов пересечения и объединения заключается в их определении в классе треугольных норм и конорм.

Треугольной нормой (t-нормой) называется двуместная действительная функция T: [0, 1] x [0, 1] $\rightarrow$ [0, 1], удовлетворяющая следующим условиям:

- T: $(0, 0) = 0$; T: $(\mu_A, 1) = \mu_A$; T: $(1, \mu_A) = \mu_A$ – ограниченность;
- $T(\mu_A, \mu_B) \leq T(\mu_C, \mu_D)$, если $\mu_A, \leq \mu_C$, $\mu_B, \leq \mu_D$ – монотонность;
- $T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$ – коммутативность;
- $T\big(\mu_A, T(\mu_B, \mu_C)\big) = T(T(\mu_A, \mu_B), \mu_C)$ – ассоциативность.

Треугольной конормой (t-конормой) называется двуместная действительная функция S: [0, 1] x [0, 1] $\rightarrow$ [0, 1], удовлетворяющая следующим условиям:

- S: $(1, 1) = 1$; S: $(\mu_A, 0) = \mu_A$; S: $(0, \mu_A) = \mu_A$ – ограниченность;
- $S(\mu_A, \mu_B) \geq S(\mu_C, \mu_D)$, если $\mu_A, \geq \mu_C$, $\mu_B, \geq \mu_D$ – монотонность;
- $S(\mu_A, \mu_B) = S(\mu_B, \mu_A)$ – коммутативность;

- $S(\mu_A, S(\mu_B, \mu_C)) = S(S(\mu_A, \mu_B), \mu_C)$ – ассоциативность.

Алгебраическое произведение A и B обозначается $A \cdot B$ и формируется следующим образом:

$$\forall x \in E \ \ \mu_{A \cdot B} = \mu_A(x)\mu_B(x).$$

Алгебраическая сумма этих множеств обозначается $A \hat{+} B$ и определяется как:

$$\forall x \in E \ \ \mu_{A \hat{+} B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x).$$

Для операций $\{\cdot, \hat{+}\}$ справедливы следующие свойства:

- $\left.\begin{matrix} A \cdot B = B \cdot A \\ A \hat{+} B = B \hat{+} A \end{matrix}\right\}$ – коммутативность;
- $\left.\begin{matrix} (A \cdot B) \cdot C = A \cdot (B \cdot C) \\ (A \hat{+} B) \hat{+} C = A \hat{+} (B \hat{+} C) \end{matrix}\right\}$ – ассоциативность;
- $A \cdot \emptyset = \emptyset; \ A \hat{+} \emptyset = A; \ \ A \hat{+} E = E; \ \ A \cdot E = A;$
- $\left.\begin{matrix} \overline{A \cdot B} = \overline{A} \hat{+} \overline{B} \\ \overline{A \hat{+} B} = \overline{A} \cdot \overline{B} \end{matrix}\right\}$ – формулы де Моргана;

Не выполняется:

- $\left.\begin{matrix} A \cdot A = A \\ A \hat{+} A = A \end{matrix}\right\}$ – идемпотентность;
- $\left.\begin{matrix} A \cdot (B \hat{+} C) = (A \cdot B) \hat{+} (A \cdot C) \\ A \hat{+} (B \cdot C) = (A \hat{+} B) \cdot (A \hat{+} C) \end{matrix}\right\}$ – дистрибутивность;
- $A \cdot \overline{A} = \emptyset; \ \ A \hat{+} \overline{A} = E.$

При совместном использовании операций $\{\cup, \cap, \hat{+}, \cdot\}$ справедливы свойства:

- $A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C);$
- $A \cdot (B \cap C) = (A \cdot B) \cap (A \cdot C);$
- $A \hat{+} (B \cup C) = (A \hat{+} B) \cup (A \hat{+} C);$
- $A \hat{+} (B \cap C) = (A \hat{+} B) \cap (A \hat{+} C).$

На основе операции алгебраического произведения определена операция *возведения в степень* α нечёткого множества A, где α — положительное число. Нечёткое множество A_α, определяется функцией принадлежности $\mu_A^\alpha = \mu_A^\alpha(x)$.

Частным случаем возведения в степень является:

- $CON(A) = A^2$ - операция *концентрирования* (уплотнения);
- $DIL(A) = A^{0,5}$ – операция растяжения, которые используются при работе с лингвистическими неопределенностями (рис. 49).

Умножение на число. Если α — положительное число, такое, что $\alpha \max_{x \in A} \mu_A(x) \leq 1$, то нечеткое множество αA имеет функцию принадлежности:

$$\mu\alpha A(x) = \alpha\mu A(x).$$

Выпуклая комбинация нечётких множеств. Пусть $A_1, A_2, \ldots, A_n$ - нечеткие множества универсального множества E, а $\omega_1, \omega_2, \ldots, \omega_n$ – неотрицательные числа, сумма которых равна 1.

Выпуклой комбинацией $A_1, A_2, \ldots, A_n$ называется нечеткое множество A с функцией принадлежности:

$$\forall x \in E \;\; \mu_A(x_1, x_2, \ldots, x_n) = \omega_1\mu_{A1}(x) + \omega_2\mu_{A2}(x) + \cdots + \omega_i\mu_{Ai}(x).$$

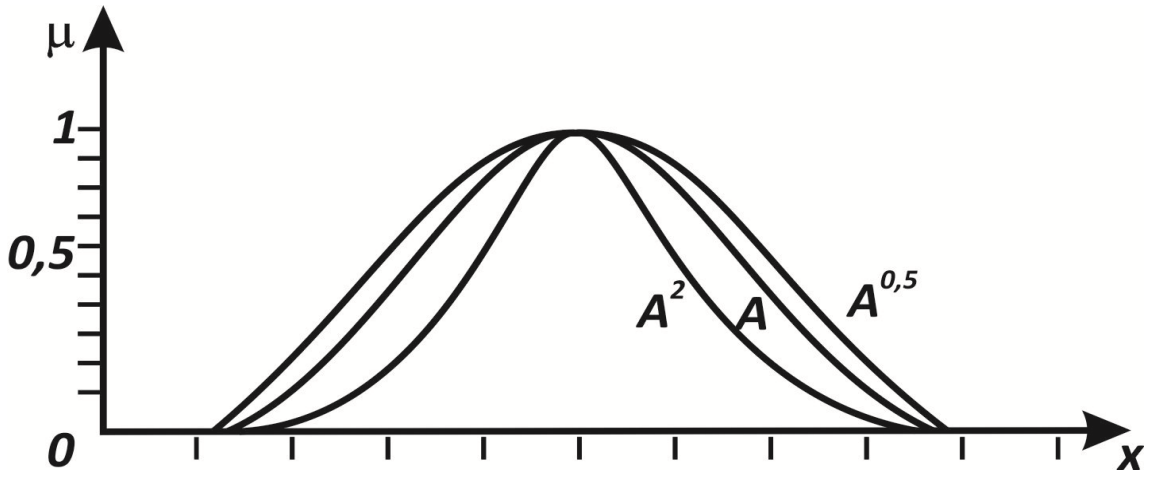


Рис. 49. Операции концентрирования (уплотнения) и растяжения.

Декартово (прямое) произведение нечетких множеств. Пусть $A_1, A_2, \ldots, A_n$ - нечеткие подмножества универсальных множеств $E_1, E_2, \ldots, E_n$ соответственно. Декартово или прямое произведение $A = A_1 \times A_2 \times \ldots \times A_n$ является нечетким подмножеством множества $E = E_1 \times E_2 \times \ldots \times E_n$ с функцией принадлежности:

$$\mu_A(x_1, x_2, \ldots, x_n) = min\{\mu_{A1}(x_1), \mu_{A2}(x_2), \ldots, \mu_{Ai}(x_n), \}.$$

Оператор увеличения нечёткости используется для преобразования чётких множеств в нечёткие и для увеличения нечёткости нечёткого множества.

Пусть A — нечёткое множество, E — универсальное множество и для всех $x \in E$ определены нечёткие множества $K(x)$. Совокупность всех $K(x)$ называется ядром оператора увеличения нечёткости H. Результатом действия оператора H на нечёткое множество A является нечёткое множество вида:

$$H(A, K) = \bigcup_{x \in E} \mu_A(x)K(x),$$

где $\mu_A(x)K(x)$ - произведение числа на нечёткое множество.

2.4. Нечёткие и лингвистические переменные

Понятия нечёткой и лингвистической переменных используются при описании объектов и явлений с помощью нечётких множеств.

Нечёткая переменная характеризуется тройкой параметров

$$\langle \alpha, X, A \rangle,$$

где α — наименование переменной,

X — универсальное множество (область определения α),

A — нечёткое множество на X, описывающее ограничения (т.е. $\mu_A(x)$) на значения нечёткой переменной α.

Лингвистическая переменная (ЛП) характеризуется набором параметров

$$\langle \beta, T, X, G, M \rangle,$$

где β — наименование лингвистической беременной;

T — множество её значений (терм-множество), представляющих наименования нечётких переменных, областью определения каждой из которых является множество X. Множество T называется базовым терм-множеством лингвистической переменной;

G — синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения). Множество $T \cup G(T)$, где $G(T)$ — множество сгенерированных термов, называется расширенным терм-множеством лингвистической переменной;

M — семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой G, в нечёткую переменную, т.е. сформировать соответствующее нечёткое множество.

Нечёткие числа - нечёткие переменные, определённые на числовой оси, т.е. нечёткое число, определяется как нечёткое множество A на множестве действительных чисел R с функцией принадлежности $\mu_A(x) \epsilon [0,1]$, где $x \in R$.

Нечёткое число A ***нормально***, если $max_x\, \mu_A(x) = 1$, и ***выпуклое,*** если для любых $x \leq y \leq z$ выполняется

$$\mu_A(x) \geq \mu_A(y) \wedge \mu_A(z).$$

Множество α -уровня нечеткого числа A определяется как

$$A\alpha = \{x / \mu_A(x) \geq \alpha\}.$$

Подмножеством $S_A \subset R$ называется носителем нечеткого числа A, если:

$$S_A = \{x / \mu_A(x) > 0\}.$$

Нечеткое число A ***унимодально***, если условие $\mu_A(x) = 1$ справедливо только для одной точки действительной оси.

Выпуклое нечёткое число A называется ***нечётким нулем***, если справедливо:

$$\mu_A(0) = sup_x(\mu_A(x).$$

Нечёткое число A ***положительно***, если $\forall x \in S_A, x > 0$ и ***отрицательно***, если $\forall x \in S_A, x < 0$.

Операции над нечёткими числами. Расширенные бинарные арифметические опёрации (сложение, умножение и др.) для нечётких чисел определяются через соответствующие операции для чётких чисел с использованием принципа обобщения следующим образом.

Пусть A и B — нечёткие числа и $\tilde{*}$ — нечёткая операция, соответствующая операции умножения над обычными числами. Тогда (используя здесь и в дальнейшем обозначения V_x вместо max_x и $\wedge_x$ вместо min_x) можно записать:

$$C = A \,\tilde{*}\, B \Leftrightarrow \mu_C(z) = V_{Z=X*Y}\big(\mu_A(x) \wedge \mu_B(y)\big).$$

Отсюда

$$C = A\tilde{+}B \Leftrightarrow \mu_C(z) = V_{Z=X+Y}\big(\mu_A(x) \wedge \mu_B(y)\big),$$

$$C = A\tilde{-}B \Leftrightarrow \mu_C(z) = V_{Z=X-Y}\big(\mu_A(x) \wedge \mu_B(y)\big),$$

$$C = A \,\tilde{\cdot}\, B \Leftrightarrow \mu_C(z) = V_{Z=X\cdot Y}\big(\mu_A(x) \wedge \mu_B(y)\big),$$

$$C = A \,\tilde{\div}\, B \Leftrightarrow \mu_C(z) = V_{Z=X\div Y}\big(\mu_A(x) \wedge \mu_B(y)\big),$$

$$C = m\tilde{a}x(A,B) \Leftrightarrow \mu_C(z) = V_{Z=max\,(X,Y)}\big(\mu_A(x) \wedge \mu_B(y)\big),$$

$$C = m\tilde{\imath}n(A,B) \Leftrightarrow \mu_C(z) = V_{Z=min\,(X,Y)}\big(\mu_A(x) \wedge \mu_B(y)\big).$$

Нечеткие числа $(\boldsymbol{L-R})$ ***– типа*** – это разновидность нечётких чисел специального вида, задаваемых по определенным правилам с целью снижения объема вычислений при операциях над ними.

Функции принадлежности нечётких чисел $(L-R)$ - типа задаются с помощью невозрастающих на множестве неотрицательных действительных чисел функций действительного переменного $L(x)$ и $R(x)$, удовлетворяющих свойствам:

1. $L(-x) = L(x), R(-x) = R(x)$;
2. $L(0) = R(0)$.

Очевидно, что к классу $(L-R)$ - функций относятся функции, графики которых имеют вид, представленный на рисунке 50.

Примерами аналитического задания $(L-R)$ - функций могут быть функции:

$$L(x) = e^{-|x|^P}, \qquad P \geq 0;$$

$$R(x) = \frac{1}{1+|x|^P}, P \geq 0.$$

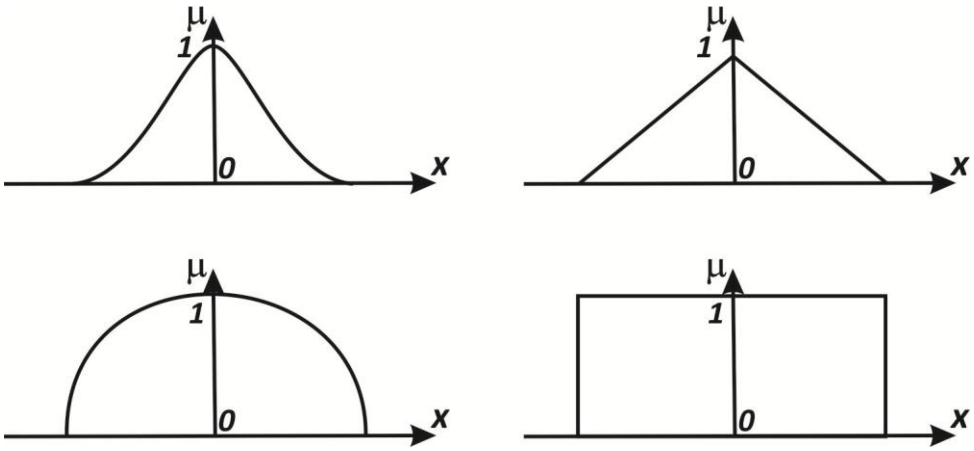


Рис. 50. Возможный вид $(L-R)$ - функции

Пусть $L(y)$ и $R(\text{y})$ — функции $(L-R)$ - типа (конкретные). Унимодальное нечёткое число A с *модой* a (т.е.$\mu_A(x)=1$) с помощью $L(\text{y})$ и $R(\text{y})$ задаётся следующим образом:

$$\mu_A(x)=\begin{cases} L\left(\dfrac{a-x}{\alpha}\right), \text{если } x \le a, \\ R\left(\dfrac{x-a}{\beta}\right), \text{если } x > a, \end{cases} \qquad (2.2)$$

где a – мода;

$\alpha > 0, \beta > 0$ - левый и правый коэффициенты нечёткости.

Таким образом, при заданных $L(y)$ и $R(\text{y})$ нечёткое число (унимодальное) задаётся тройкой $A=(a,\alpha,\beta)$.

Толерантное нечёткое число задается, соответственно, четвёркой параметров $A=(a_1,a_2,\alpha,\beta)$, где a_1 и a_2 — границы толерантности, т.е. в промежутке $[a_1,a_2]$значение функции принадлежности равно 1.

Примеры графиков функций принадлежности нечётких чисел $(L-R)$ - типа приведены на рисунке 51.

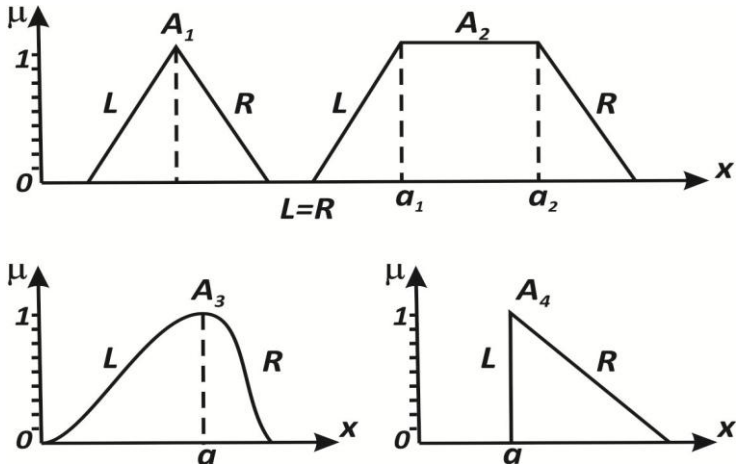


Рис. 51. Примеры графиков функций принадлежности нечетких чисел $(L-R)$ - функций

Отметим, что в конкретных ситуациях функции $L(y)$ и $R(\text{y})$, а также параметры a, в нечётких чисел (a, α, β) и $(a_1, a_2, \alpha, \beta)$ должны подбираться таким образом, чтобы результат операции (сложения, вычитания, деления и т.д.) был точно или приблизительно равен нечёткому числу с теми же $L(y)$ и $R(\text{y})$, а параметры α' и β' результата не выходили за рамки ограничений на эти параметры для исходных нечётких чисел, особенно, если результат будет участвовать в дальнейших операциях.

Примечание. Моделирование сложных систем с применением аппарата нечётких множеств требует выполнения большого объёма операций над разного рода, лингвистическими и другими нечёткими переменными. Для удобства исполнения операций, а также для ввода-вывода и хранения данных желательно выбирать функции принадлежности стандартного вида.

Нечёткие множества, которыми приходится оперировать в большинстве задач, являются, как правило, унимодальными и нормальными. Одним из возможных методов аппроксимации унимодальных нечётких множеств является аппроксимация с помощью функций $(L-R)$ - типа. Примеры $(L-R)$ - представлений лингвистических переменных приведены в таблице 2.1.

Таблица 2.1.

Возможное $(L-R)$ – представление некоторых лингвистических переменных

Терм лингвистической переменной	$(L-R)$ **– представление**	**Графическое представление**
Средний	$A = (a, \alpha, \beta)_{LR}$ $\alpha = \beta > 0$	$\alpha\ \beta$
Малый	$A = (a, \infty, \beta)_{LR}$ $\alpha = \infty$	$\alpha = \infty\ \beta$
Большой	$A = (a, \alpha, \infty)_{LR}$ $\beta = \infty$	$\alpha\ \beta = \infty$
Приблизительно в диапазоне	$A = (a_1, a_2, \alpha, \beta)_{LR}$ $\alpha = \beta > 0$	$\alpha\ \beta$ $a_1\ a_2$
Определенный	$A = (a, \alpha, \beta)_{LR}$ $\alpha = \beta = 0$	$\alpha = 0\ \beta = 0$
Разнообразной: зона полной неопределенности	$A = (a, \infty, \infty)_{LR}$ $\alpha = \beta = \infty$	$\alpha = \beta = \infty$

2.5. Нечёткие отношения

Пусть $E = E_1 \times E_2 \times \ldots \times E_n$ — прямое произведение универсальных множеств и M — некоторое множество принадлежностей (например, $M = [0,1]$). Нечёткое n-ное отношение определяется как нечёткое подмножество R на E, принимающее свои значения в M. В случае $n = 2$ и $M = [0,1]$ нечётким отношением R между множествами $X = E_1$ и $Y = E_2$ будет называться функция $R\colon (X, Y) \to [0,1]$, которая ставит в соответствие каждой паре элементов

$(x, y) \in X \times Y$ величину $\mu_R(x, y) \epsilon [0, 1]$.

Обозначение: нечёткое отношение на $X \times Y$ записывается в виде: $x \in X, y \in Y: xRy$. В случае, когда $X = Y$, т,е. X и Y совпадают, нечёткое отношение $R: X \times X \to [0, 1]$ называется нечётким отношением на множестве X.

Пример 2.6. Отношение R, для которого $\mu_R(x, y) = e^{k(x \cdot y)^2}$, при достаточно больших k можно интерпретировать так: «x и y близкие друг к другу числа».

Пример 2.7. Пусть $X = Y = (-\infty, \infty)$, т.е. множество всех действительных чисел. Отношение $x \gg y$ можно задать функцией принадлежности:

$$\mu = \begin{cases} 0, \text{если}_1\ x \leq y \\ \dfrac{1}{1 + \dfrac{1}{(x - y)^2}}, \text{если } y > x. \end{cases} \qquad (2.3)$$

Пример 2.8. Пусть $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_{3,} y_4\}$, $M = [0, 1]$. Нечеткое отношение $R = X\ R\ Y$ может быть задано, например, таблицей 2.2.

Таблица 2.2.

Задание нечеткого отношения

	y_1	y_2	y_3	y_4
x_1	0	0	0,1	0,3
x_2	0	0,8	1	0,7
x_3	1	0,5	0,6	1

Операции над нечеткими отношениями.

Объединение двух отношений $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ обозначается $\boldsymbol{R_1 \cup R_2}$ и определяется выражением:

$$\boldsymbol{\mu_{R_1 \cup R_2}(x, y) = \mu_{R_1}(x, y) \vee \mu_{R_2}(x, y)}.$$

Пересечение двух отношений $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ обозначается $\boldsymbol{R_1 \cap R_2}$ и определяется выражением:

$$\boldsymbol{\mu_{R_1 \cap R_2}(x, y) = \mu_{R_1}(x, y) \wedge \mu_{R_2}(x, y)}.$$

Алгебраическое произведение двух отношений $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ обозначается $\boldsymbol{R_1 \cdot R_2}$ и определяется выражением:

$$\boldsymbol{\mu_{R_1 \cdot R_2}(x, y) = \mu_{R_1}(x, y) \cdot \mu_{R_2}(x, y)}.$$

Алгебраическая сумма двух отношений $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ обозначается $\boldsymbol{R_1 \tilde{+} R_2}$ и определяется выражением:

$$\boldsymbol{\mu_{R_1 \tilde{+} R_2}(x, y) = \mu_{R_1}(x, y) + \mu_{R_2}(x, y) - \mu_{R_1}(x, y) \cdot \mu_{R_2}(x, y)}.$$

Для введенных операций справедливы следующие свойства

дистрибутивности:

$$R_1 \cap (R_2 \cup R_3) = (R_1 \cap R_2) \cup (R_1 \cap R_3),$$

$$R_1 \cup (R_2 \cap R_3) = (R_1 \cup R_2) \cap (R_1 \cup R_3),$$

$$R_1 \cdot (R_2 \cup R_3) = (R_1 \cdot R_2) \cup (R_1 \cdot R_3),$$

$$R_1 \cdot (R_2 \cap R_3) = (R_1 \cdot R_2) \cap (R_1 \cdot R_3),$$

$$R_1 \widetilde{+} (R_2 \cup R_3) = (R_1 \widetilde{+} R_2) \cup (R_1 \widetilde{+} R_3),$$

$$R_1 \widetilde{+} (R_2 \cap R_3) = (R_1 \widetilde{+} R_2) \cap (R_1 \widetilde{+} R_3).$$

Дополнительные отношения R обозначаются, $\overline{R}$ и определяется функцией принадлежности:

$$\mu_{\overline{R}}(x, y) = 1 - \mu_R(x, y).$$

Дизъюнктивная сумма двух отношений $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ обозначается $\boldsymbol{R_1 \oplus R_2}$ и определяется выражением:

$$\boldsymbol{R_1 \oplus R_2} = \left(\boldsymbol{R_1} \cap \overline{\boldsymbol{R_2}}\right) \cup \left(\overline{\boldsymbol{R_1}} \cap \boldsymbol{R_2}\right).$$

Обычное отношение, ближайшее к нечётному. Пусть $\boldsymbol{R}$ – нечеткое отношение с функцией принадлежности $\boldsymbol{\mu_R(x, y)}$. Обычное отношение, ближайшее к нечетному, обозначается $\underline{\boldsymbol{R}}$ и определяется выражением:

$$\mu_{\underline{R}}(x, y) = \begin{cases} 0, \text{если } \mu_R(x, y) < 0{,}5 \\ 1, \text{если } \mu_R(x, y) > 0{,}5 \\ 0 \text{ или } 1, \text{если } \mu_R(x, y) = 0{,}5. \end{cases}$$

По договоренности принимают $\mu_{\underline{R}}(x, y) = 0$ при $\mu_R(x, y) = 0{,}5$.

Композиция (свертка). Пусть $\boldsymbol{R_1}$ – нечеткое отношение $\boldsymbol{R_1}: (\boldsymbol{X} \times \boldsymbol{Y}) \to [\boldsymbol{0}, \boldsymbol{1}]$ между $\boldsymbol{X}$ и $\boldsymbol{Y}$, $\boldsymbol{R_2}$ – нечеткое отношение $\boldsymbol{R_2}: (\boldsymbol{Y} \times \boldsymbol{Z}) \to [\boldsymbol{0}, \boldsymbol{1}]$ между $\boldsymbol{Y}$и $\boldsymbol{Z}$. Нечеткое отношение между $\boldsymbol{X}$ и $\boldsymbol{Z}$, обозначаемое $\boldsymbol{R_1 \cdot R_2}$, определенное через $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ выражением:

$$\mu_{\boldsymbol{R_1 \cdot R_2}}(x, z) = V_y\left[\mu_{\boldsymbol{R_1}}(x, y) \wedge \mu_{\boldsymbol{R_2}}(y, z)\right],$$

Называется $(max - min)$ – композицией ($(max - min)$ – сверткой) отношений $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$.

Пример 2.9. Пусть

$\boldsymbol{R_1}$	y_1	y_2	y_3
x_1	0,1	0,7	0,4
x_2	1	0,5	0

$\boldsymbol{R_2}$	z_1	z_2	z_3	z_4
y_1	0,9	0	1	0,2
y_2	0,3	0,6	0	0,9
y_3	0,1	1	0	0,5

Тогда

$\boldsymbol{R_1 \cdot R_2}$	z_1	z_2	z_3	z_4
x_1	0,3	0,6	0,1	0,7
x_2	0,9	0,5	1	0,5

При этом

$$\mu_{R_1 \cdot R_2}(x_1, z_1) = \left[\mu_{R_1}(x_1, y_1) \wedge \mu_{R_2}(y_1, z_1)\right] \vee \left[\mu_{R_1}(x_1, y_2) \wedge \mu_{R_2}(y_2, z_1)\right] \vee$$
$$\vee \left[\mu_{R_1}(x_1, y_3) \wedge \mu_{R_2}(y_3, z_1)\right] = (0{,}1 \wedge 0{,}9) \vee (0{,}7 \wedge 0{,}3) \vee (0{,}4 \wedge 0{,}1) =$$
$$= 0{,}1 \vee 0{,}3 \vee 0{,}1 = 0{,}3;$$

$$\mu_{R_1 \cdot R_2}(x_1, z_2) = (0{,}1 \wedge 0) \vee (0{,}7 \wedge 0{,}6) \vee (0{,}4 \wedge 1) = 0 \vee 0{,}6 \vee 0{,}4 = 0{,}6;$$

$$\mu_{R_1 \cdot R_2}(x_1, z_3) = 0{,}1;$$

…

$$\mu_{R_1 \cdot R_2}(x_1, z_5) = 0{,}5.$$

Примечание. В этом примере сначала использован «аналитический» способ композиции отношений R_1 и R_2, т.е. i – я строка R_1 «умножается» на j – й столбец R_2 с использованием операции $\wedge$, затем полученный результат «свёртывается» с использованием операции $\vee$ в $\mu(x_i, z_j)$.

Свойства max-min композиции. Операция (*max-min*) – композиция ассоциативна, т.е.

$$R_3 \cdot (R_2 \cdot R_1) = (R_3 \cdot R_2) \cdot R_1,$$

дистрибутивна относительно объединения, но не дистрибутивна относительно пересечения:

$$R_3 \cdot (R_2 \cup R_1) = (R_3 \cdot R_2) \cup (R_3 \cdot R_1),$$
$$R_3 \cdot (R_2 \cap R_1) \neq (R_3 \cdot R_2) \cap (R_3 \cdot R_1).$$

Кроме того, для (*max-min*) – композиции выполняется следующее важное свойство: если $R_1 \subset R_2$, то $R_3 \cdot R_1 \subset R_3 \cdot R_2$.

(max-*) – композиция. В выражении $\mu_{R_1 \cdot R_2}(x, z) = V_y\left[\mu_{\boldsymbol{R_1}}(x, y) \wedge \mu_{\boldsymbol{R_2}}(y, z)\right]$ для (*max-min*) – композиции отношений R_1 и R_2 операцию $\wedge$ можно заменить любой другой, для которой выполняется те же ограничения, что и для

∧: ассоциативность и монотонность (в смысле неубывания) по каждому аргументу. Тогда

$$\mu_{R_1 \cdot R_2}(x, z) = V_y\left[\mu_{\boldsymbol{R_1}}(x, y) * \mu_{\boldsymbol{R_2}}(y, z)\right].$$

В частности, операция ∧ может быть заменена алгебраическим умножением, тогда говорят о (*max-prod*) – композиции.

2.6. Нечёткий логический вывод

Используемый в различного рода экспертных и управляющих системах механизм нечётких выводов в своей основе имеет базу знаний, формируемую специалистами предметной области в виде совокупности нечётких предикатных правил вида:

$$\Pi_1: \text{если } x \text{ есть } A_1, \text{то } y \text{ есть } B_1,$$

$$\Pi_2: \text{если } x \text{ есть } A_2, \text{то } y \text{ есть } B_2,$$

…

$$\Pi_n: \text{если } x \text{ есть } A_n, \text{то } y \text{ есть } B_n;$$

где x – входная переменная (имя для известных значений данных); y – переменная вывода (имя для значений данных, которые будут вычислены); A и B – функции принадлежности, определенные соответственно на x и y.

Приведем более детальное пояснение. Знание эксперта $A \to B$ отражает нечёткое причинное отношение предпосылки и заключения, поэтому его можно назвать нечётким отношением и обозначить через R:

$$R = A \to B,$$

где « → » называют нечеткой импликацией.

Отношение R можно рассматривать как нечёткое подмножество прямого произведения $X \times Y$ полного множества предпосылок X и заключений Y. Таким образом, процесс получения (нечёткого) результата вывода B' с использованием данного наблюдения A' и знания $A \to B$ можно представить в виде композиционного правила *нечеткий «modus ponens»*:

$$B' = A' \cdot R = A' \cdot (A \to B),$$

где « · » - введенная выше операция свертки.

Как операцию композиции, так и операцию импликации в алгебре нечётких множеств можно реализовывать по-разному (при этом будет отличаться и получаемый результат), но в любом случае общий логический вывод осуществляется, за следующие четыре этапа.

1) ***Введение нечёткости*** (фаззификация, fuzzification). Функции принадлежности, определенные на входных переменных, применяются к их фактическим значениям для определения степени истинности каждой

предпосылки каждого правила.

2) ***Логический вывод.*** Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечёткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила. В качестве правил логического вывода обычно используются только операции min (МИНИМУМ) или $prod$ (УМНОЖЕНИЕ). В логическом выводе МИНИМУМА функция принадлежности вывода «отсекается» по высоте, соответствующей вычисленной степени истинности предпосылки правила (нечёткая логика «И»). В логическом выводе УМНОЖЕНИЯ функция принадлежности вывода масштабируется при помощи вычисленной степени истинности предпосылки правила.

3) ***Композиция.*** Все нечёткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы сформировать одно нечёткое подмножество для всех переменных вывода. При подобном, объединении обычно используются операции max (МАКСИМУМ) или sum (СУММА) При композиции МАКСИМУМА комбинированный вывод нечёткого подмножества конструируется как поточечный максимум по всем нечётким подмножествам (нечёткая логика - «ИЛИ»). При композиции СУММЫ комбинированный вывод нечёткого подмножества формируется как поточечная сумма по всем нечётким подмножествам, назначенным переменной вывода правилами логического вывода.

4) ***Приведение к чёткости*** (дефаззификация, defuzzification) используется, если требуется преобразовать нечёткий набор выводов в чёткое число. Существует большее количество методов приведения к чёткости, некоторые из которых рассмотрены ниже.

Пример 2.10. Пусть некоторая система описывается следующими нечёткими правилами:

$$\Pi_1\text{: если } x \text{ есть } A, \text{то } \omega \text{ есть } D,$$

$$\Pi_2\text{: если } y \text{ есть } B, \text{то } \omega \text{ есть } E,$$

$$\Pi_3\text{: если } z \text{ есть } C, \text{то } \omega \text{ есть } F,$$

где x, y, z – имена входных переменных; ω - имя переменной выхода; A, B, C, D, E, F – заданные функции принадлежности (треугольной формы).

Процедура получения логического вывода иллюстрируется рисунком 52. Предполагается, что заданы конкретные (чёткие) значения входных переменных: x_0, y_0, z_0.

На первом этапе на основании данных значений и, исходя из функций принадлежности A, B, C находятся степени истинности $\alpha(x_0)$, $\alpha(y_0)$ и $\alpha(z_0)$ для предпосылок каждого из трёх приведённых правил.

На втором этапе происходит «отсекание» функций принадлежности заключений правил (D, E, F) на уровнях $\alpha(x_0)$, $\alpha(y_0)$ и $\alpha(z_0)$.

На третьем этапе рассматриваются функции принадлежности, усечённые

на предыдущем этапе, и производится их объединение с использованием операции max, в результате чего получается комбинированное нечёткое подмножество, описываемое функцией принадлежности $\mu_\Sigma(\omega)$ и соответствующее логическому выводу для выходной переменной ω.

Наконец, на четвёртом этапе находится, при необходимости, чёткое значение выходной переменной, например, с применением центроидного метода: чёткое значение выходной переменной определяется как центр тяжести для кривой $\mu_\Sigma(\omega)$:

$$\omega_0 = \frac{\int_\Omega \omega \cdot \mu_\Sigma(\omega) d\omega}{\int_\Omega \mu_\Sigma(\omega) d\omega} \tag{2.4}$$

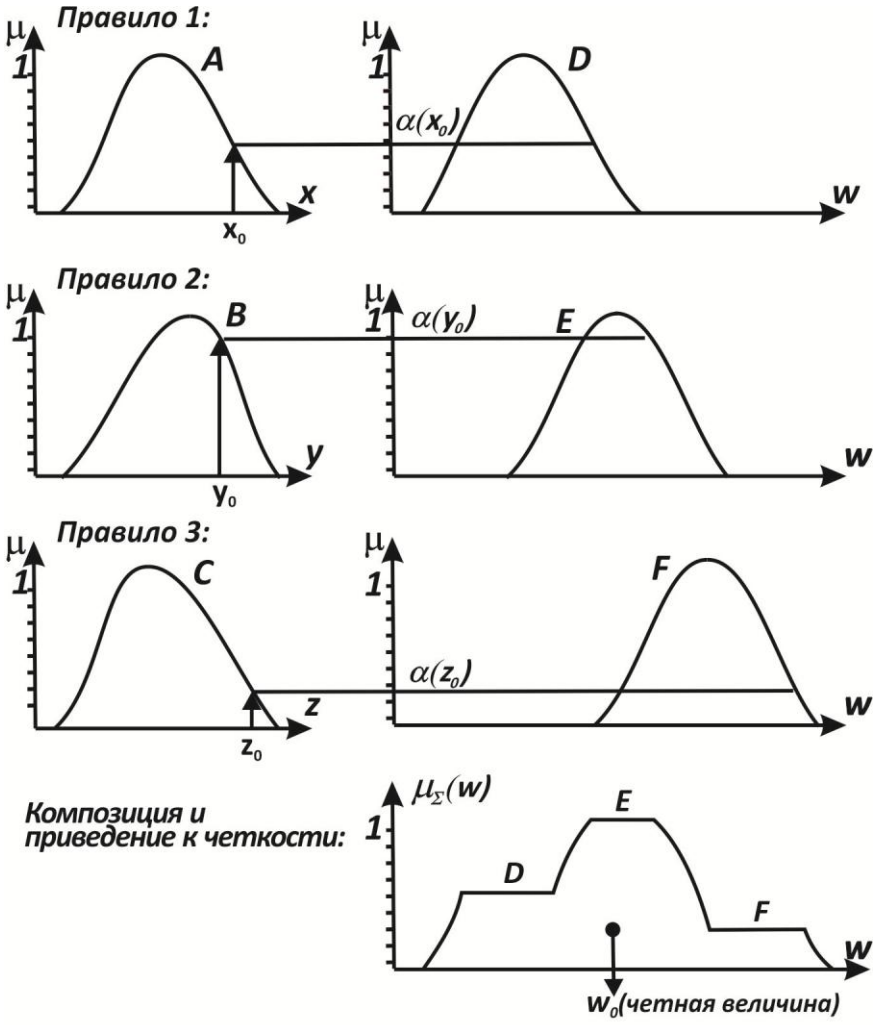


Рис. 52. Иллюстрация процедуры логического вывода

Рассмотрим следующие наиболее употребительные модификации алгоритма нечёткого вывода, полагая, для простоты, что базу знаний организуют два нечётких правила вида:

$$\text{П}_1\text{: если } x \text{ есть } A_1 \text{ и } y \text{ есть } B_1, \text{то } z \text{ есть } C_1,$$

$$\Pi_2\text{: если } x \text{ есть } A_2 \text{ и } y \text{ есть } B_2, \text{то } z \text{ есть } C_2,$$

где x и y – имена входных переменных; z – имя переменной вывода; $A_1,\ A_2,\ B_1, B_2,\ C_1, C_2$ – некоторые заданные функции принадлежности.

При этом чёткое значение z_0 необходимо определить на основе приведённой информации и чётких значений x_0 и y_0.

Алгоритм Mamdani.

Данный алгоритм соответствует рассмотренному примеру и рисунку 52. В рассматриваемой ситуации он математически может быть описан следующим образом:

1) ***Введение нечёткости.*** Находятся степени истинности для предпосылок каждого правила: $A_1(x_0), A_2(x_0), B_1(y_0), B_2(y_0)$.

2) ***Логический вывод***. Находятся уровни «отсечения» для предпосылок каждого из правил (с использованием операции МИНИМУМ):

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0),$$
$$\alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

где через « ∧ » обозначается операция логического минимума (min).

Затем находятся «усеченные» функции принадлежности:

$$C'_1(z) = \big(\alpha_1 \wedge C_1(z)\big),$$
$$C'_2(z) = \big(\alpha_2 \wedge C_2(z)\big).$$

3) ***Композиция.*** Производится объединение найденных усеченных функций с использованием операции МАКСИМУМ (max, далее обозначаемой как «∨»), что приводит к получению итогового нечёткого подмножества для переменной выхода с функцией принадлежности:

$$\mu_\Sigma(z) = C(z) = C'_1(z) \vee C'_2(z) = \big(\alpha_1 \wedge C_1(z)\big) \vee \big(\alpha_2 \wedge C_2(z)\big).$$

4) ***Приведение к нечеткости***. Проводится для нахождения z_0, например, центроидным методом.

Алгоритм Tsukamoto.

Исходные посылки — как у предыдущего алгоритма, но здесь предполагается, что функции $C_1(z)$, $C_2(z)$ являются монотонными.

1) ***Введение нечёткости*** (как в алгоритме *Mamdani*).

2) ***Нечёткий вывод.*** Сначала находятся уровни «отсечения» α_1 и α_2 (как в алгоритме *Mamdani*), а затем решением уравнений:

$$\alpha_1 = C_1(z_1),$$
$$\alpha_2 = C_2(z_2)$$

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

3) Определяется чёткое значение переменной вывода (как взвешенное

среднее z_1 и z_2):

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}; \tag{2.5}$$

в общем случае (дискретный вариант центроидного метода):

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i}. \tag{2.6}$$

Пример 2.11. Пусть заданы $A_1(x_0) = 0{,}7;\ A_2(x_0) = 0{,}6;\ B_1(y_0) = 0{,}3;$ $B_2(y_0) = 0{,}8$; соответствующие уровни отсечения:

$$\alpha_1 = min\{A_1(x_0);\ B_1(y_0)\} = min\{0{,}7; 0{,}3\} = 0{,}3;$$

$$\alpha_2 = min\{A_2(x_0);\ B_2(y_0)\} = min\{0{,}6; 0{,}8\} = 0{,}6;$$

и значения $z_1 = 8$ и $z_2 = 4$, найденные в результате решения уравнений (рис.53):

$$C_1(z_1) = 0{,}3;$$

$$C_2(z_2) = 0{,}6.$$

При этом четкое значение переменной выхода:

$$z_0 = \frac{(8 \cdot 0{,}3 + 4 \cdot 0{,}6)}{(0{,}3 + 0{,}6)} = 6.$$

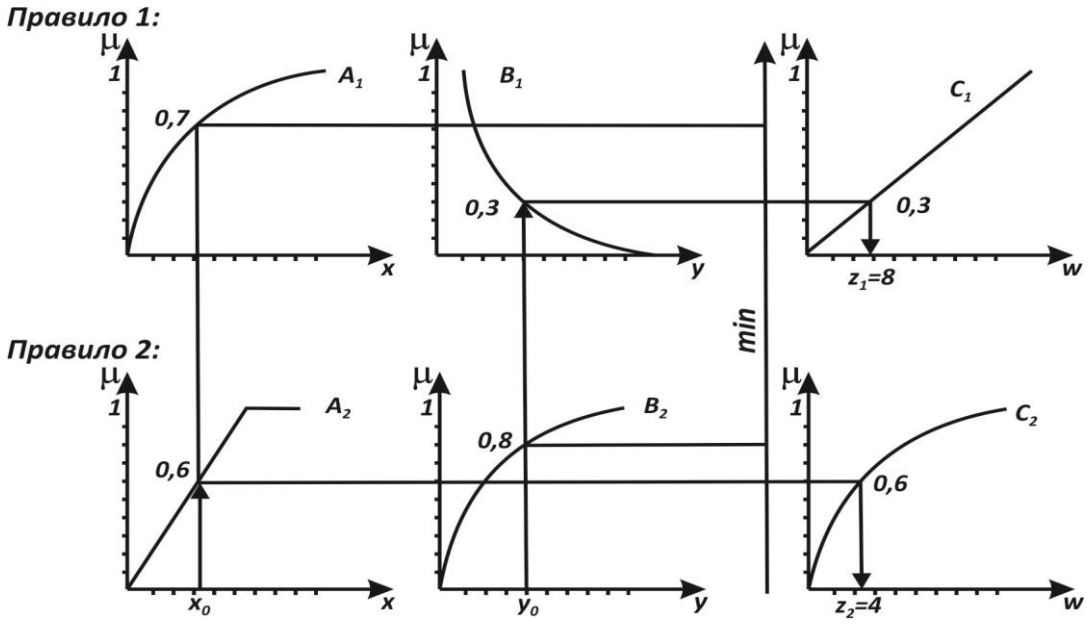


Рис. 53. Иллюстрация к алгоритму *Tsukamoto*

Алгоритм Sugeno.

Sugeno и *Takagi* использовали набор правил в следующей форме (как и ранее, приведём пример двух правил):

$$\text{П}_1\text{: если } x \text{ есть } A_1 \text{ и } y \text{ есть } B_1, \text{ то } z_1 = a_1 x + b_1 y,$$

$$П_2: \text{если } x \text{ есть } A_2 \text{и } y \text{ есть } B_2, \text{то } z_2 = a_2x + b_2y.$$

Представление алгоритма (рис. 54).

1) ***Введение нечеткости*** (как в алгоритме *Mamdani*).

2) ***Нечёткий вывод.*** Находятся $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ и индивидуальные выходы правил:

$$z_1^* = a_1x_0 + b_1y_0,\ z_2^* = a_2x_0 + b_2y_0.$$

3) Определяется четкое значение переменной вывода:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}. \qquad (2.7)$$

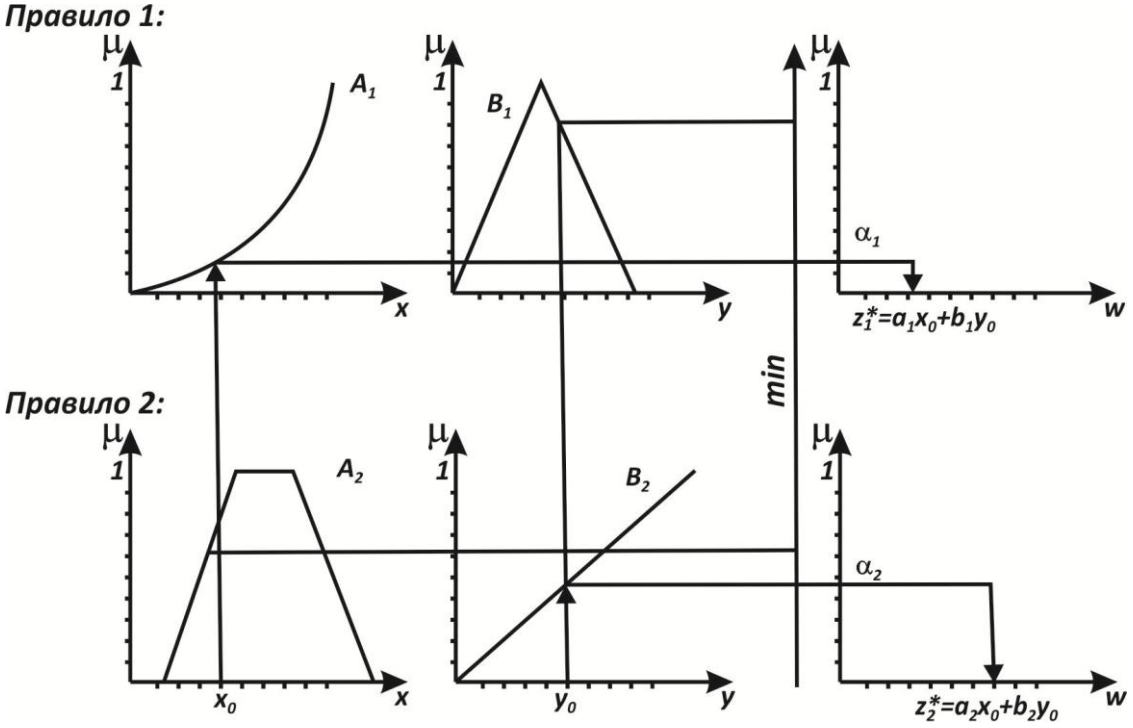


Рис. 54. Иллюстрация к алгоритму *Sugeno*

Алгоритм Larsen.

В алгоритме *Larsen* нечёткая импликация моделируется с использованием оператора умножения.

Описание алгоритма (рис. 55).

1) ***Введение нечёткости*** (как в алгоритме *Mamdani*).

2) ***Нечёткий вывод.*** Сначала, как в алгоритме *Mamdani*, находятся значения $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$, а затем определяются частные нечеткие подмножества:

$$\alpha_1 C_1(z),\ \alpha_2 C_2(z).$$

3) Находится итоговое нечеткое подмножество:

$$\mu_\Sigma(z) = C(z) = \big(\alpha_1 C_1(z)\big) \vee \big(\ \alpha_2 C_2(z)\big) \qquad (2.8)$$

в общем случае n правил:

$$\mu_{\Sigma}(z) = C(z) = V_{i=1}^{n}\left(\alpha_i C_i(z)\right) \tag{2.9}$$

4) При необходимости производится приведение к четкости (как в ранее рассмотренных алгоритмах).

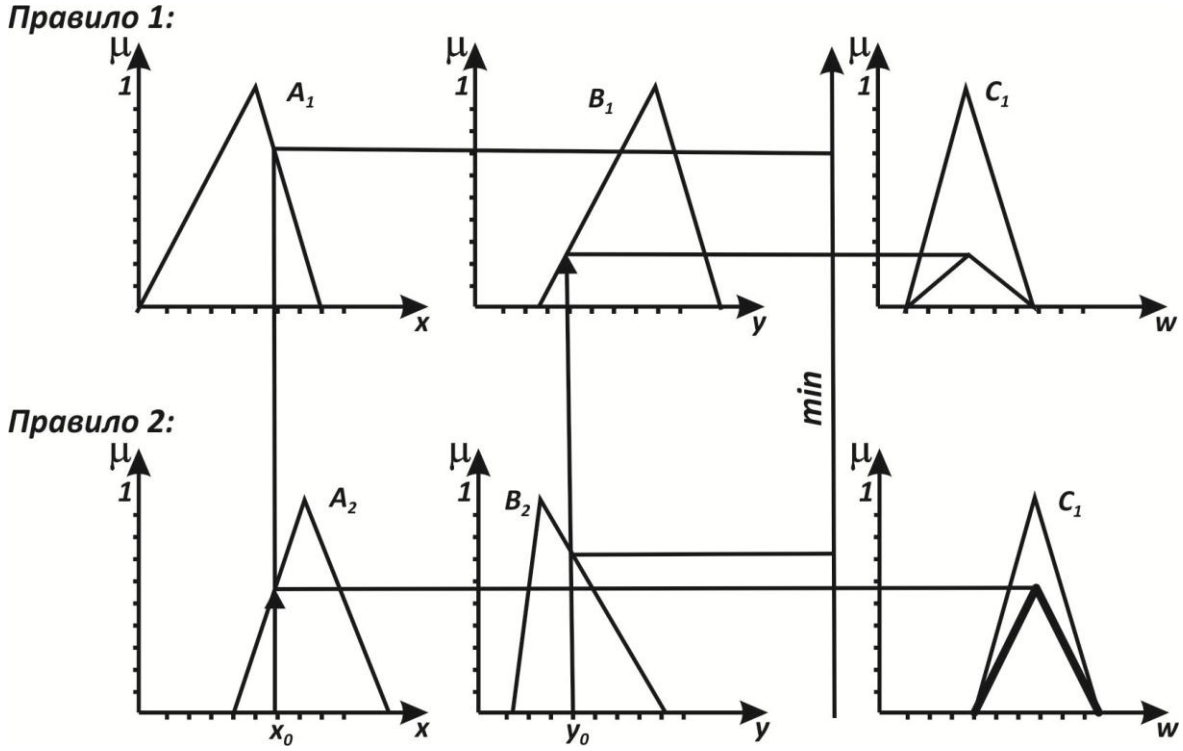


Рис. 55. Иллюстрация к алгоритму *Sugeno*

Упрощённый алгоритм нечёткого вывода.

Исходные правила в данном случае задаются в виде:

$$\text{П}_1\text{: если } x \text{ есть } A_1 \text{ и } y \text{ есть } B_1, \text{то } z_1 = c_1,$$

$$\text{П}_2\text{: если } x \text{ есть } A_2 \text{ и } y \text{ есть } B_2, \text{то } z_2 = c_2.$$

где c_1 и c_2 - некоторые четкие числа.

Описание алгоритма (рис. 56).

1) ***Введение нечёткости*** (как в алгоритме *Mamdani*).

2) ***Нечёткий вывод.*** Находятся числа $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$.

3) Определяется четкое значение выходной переменной:

$$z_0 = \frac{\alpha_1 c_1 + \alpha_2 c_2}{\alpha_1 + \alpha_2}, \tag{2.10}$$

в общем случае n правил:

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i c_i}{\sum_{i=1}^{n} \alpha_i}. \tag{2.11}$$

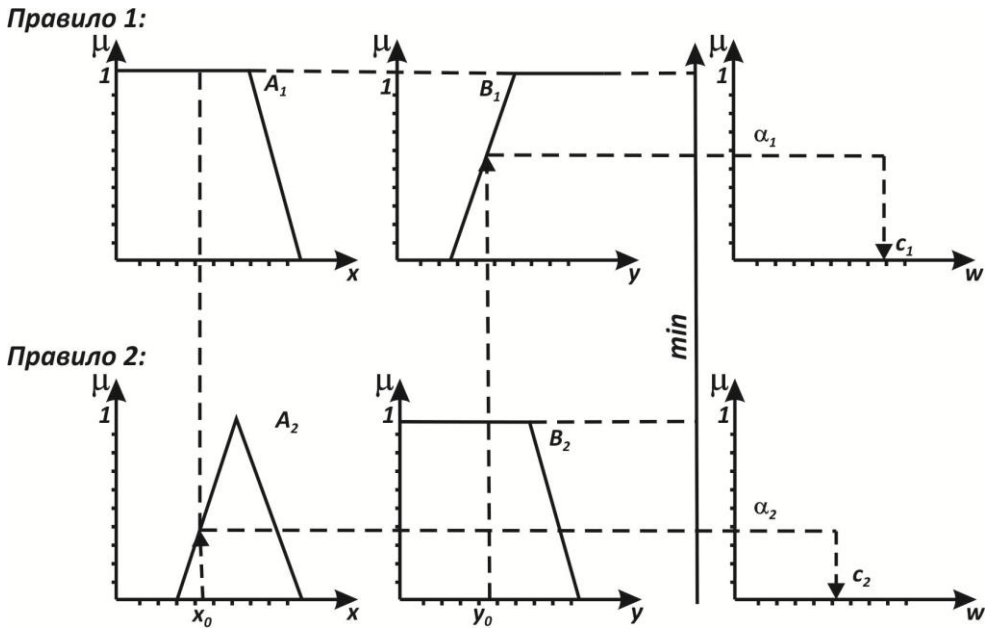


Рис. 56. Иллюстрация к упрощенному алгоритму
нечеткого логического вывода

Методы приведения к чёткости.

1) Выше уже был рассмотрен один из данных методов — *центроидный.* Приведём соответствующие формулы ещё раз. В общем случае:

$$z_0 = \frac{\int_\Omega z \cdot C(z)dz}{\int_\Omega C(z)dz}, \tag{2.12}$$

для дискретного варианта:

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i}. \tag{2.13}$$

2) *Первый максимум* (*First-of-Maxima*). Четкая величина вывода находится как наименьшее значение, при котором достигается максимум итогового нечёткого множества (рис. 57а).

3) *Средний максимум* (*Middle-of-Maxima*). Четкое значение находится по формуле:

$$z_0 = \frac{\int_G zdz}{\int_G dz}, \tag{2.14}$$

где G – подмножество элементов, максимизирующих C (рис. 57б).

Для дискретного варианта (C – дискретно):

$$z_0 = \frac{1}{n}\sum_{j=1}^{n} z_j. \tag{2.15}$$

$$z_0 = min\left\{z \middle| C(z) = \max_u C(u)\right\}. \quad (2.16)$$

4) *Критерий максимума* (*Max-Criterion*). Чёткое значение выбирается произвольно среди множества элементов, для которых C достигает максимума:

$$z_0 \in min\left\{z \middle| C(z) = \max_u C(u)\right\} \quad (2.17)$$

5) *Высотная дефаззификация* (*Height defuzzification*). Элементы области определения Ω, для которых значения функции принадлежности меньше, чем некоторый уровень α, в расчет не принимаются, и чёткое значение рассчитывается в соответствии с выражением:

$$z_0 = \frac{\int_{C\alpha} z \cdot C(z)dz}{\int_{C\alpha} C(z)dz}, \quad (2.18)$$

где $C\alpha$ - нечеткое множество α – уровня (см. выше).

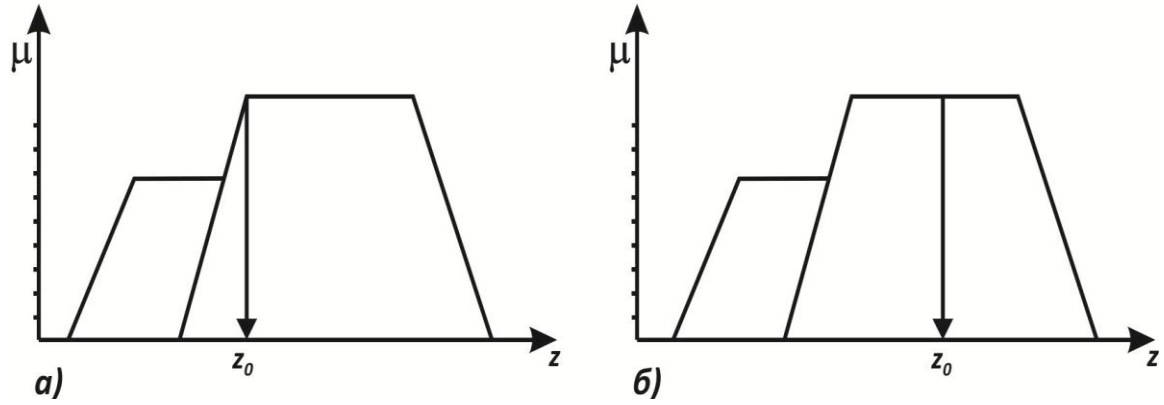


Рис. 57. Иллюстрация к методам приведения к чёткости:
а) – первый максимум; ***б***) – средний максимум

Нисходящие нечёткие логические выводы.

Рассмотренные нечёткие логические выводы являются восходящими выводами от предпосылок к заключениям. В диагностических нечётких системах это нисходящие выводы. Рассмотрим механизм подобного вывода.

Пусть задано полное пространство предпосылок $X = \{x_1, \dots, x_m\}$ и полное пространство заключений $Y = \{y_1, \dots, y_m\}$.

Между x_i и y_i существуют нечёткие причинные отношения $x_i \to y_i$, которые можно представить в виде некоторой матрицы R с элементами $r_{ij} \in [0,1],\ i = 1 \dots m,\ j = 1 \dots n.$ Предпосылки и заключения можно рассматривать как нечеткие множества A и B на пространствах X и Y, отношения которых можно представить в виде:

$$B = A \cdot R,$$

где « · », как и раньше, обозначает правило композиции нечетких выводов,

например $(max - min)$ – композицию.

В данном случае направление выводов является обратным для правил, т.е. задана матрица R (знания эксперта), наблюдаются выходы B (заключения) и определяются входы A (предпосылки).

Пусть задана медаль диагностики системы, состоящая из двух предпосылок и трёх заключений: $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3\}$, а матрица нечетких отношений имеет вид:

$$R = \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}3 \\ 0{,}7 & 0{,}4 & 0{,}5 \end{bmatrix}.$$

Допустим в результате диагностики системы были получены следующие заключения:

$$B = {}^{0{,}8}/_{y_1} + {}^{0{,}2}/_{y_2} + {}^{0{,}3}/_{y_3}.$$

Необходимо найти приведшие к этому предпосылки:

$$A = {}^{a_1}/_{x_1} + {}^{a_2}/_{x_2}.$$

С учетом конкретных данных отношения между предпосылками и заключениями будут представлены следующим образом:

$$[0{,}8 \quad 0{,}2 \quad 0{,}3] = [a_1 \quad a_2] \cdot \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}3 \\ 0{,}7 & 0{,}4 & 0{,}5 \end{bmatrix},$$

либо в транспонированном виде:

$$\begin{bmatrix} 0{,}8 \\ 0{,}2 \\ 0{,}3 \end{bmatrix} = \begin{bmatrix} 0{,}8 & 0{,}7 \\ 0{,}2 & 0{,}4 \\ 0{,}3 & 0{,}5 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

При использовании $(max - min)$ – композиции последнее соотношение преобразуется к виду:

$$0{,}8 = (0{,}8 \wedge a_1) \vee (0{,}7 \wedge a_2),$$

$$0{,}2 = (0{,}2 \wedge a_1) \vee (0{,}4 \wedge a_2),$$

$$0{,}3 = (0{,}3 \wedge a_1) \vee (0{,}5 \wedge a_2).$$

При решении данной системы заметим, что в первом уравнении второй член правой части не влияет на левую часть, поэтому:

$$0{,}8 = 0{,}8 \wedge a_1, \qquad a_1 \geq 0{,}8.$$

Из второго уравнения получим:

$$0{,}2 \geq 0{,}4 \wedge a_2, \qquad a_1 \leq 0{,}2.$$

Полученное решение удовлетворяет третьему уравнению. Таким образом:

$$0{,}8 \leq a_1 \leq 1, \qquad 0 \leq a_2 \leq 0{,}2.$$

При решении практических задач могут одновременно использоваться различные правила композиции нечётких выводов, сама схема выводов может быть многокаскадной. В настоящее время общих методов решения подобных задач, по-видимому, не существует.

2.7. Эффективность нечётких систем принятия решений

Возможность использования аппарата нечёткой логики базируется на следующих результатах.

1) В 1992 г. *Wang* показал, что нечёткая система является универсальным аппроксиматором, т.е. может аппроксимировать любую непрерывную функцию на компакте U с произвольной точностью, если использует набор n $(n \to \infty)$ правил:

Π_i: если x_i, есть A_i и y_i, есть B_i, тогда z_i есть $C_i, i = 1, \dots n$, при условиях:

- гауссовых функциях принадлежности:

$$A_i(x) = exp\left[-\frac{1}{2}\left(\frac{x-\alpha_{i1}}{\beta_{i1}}\right)^2\right], \qquad B_i(y) = exp\left[-\frac{1}{2}\left(\frac{y-\alpha_{i2}}{\beta_{i2}}\right)^2\right],$$

$$C_i(z) = exp\left[-\frac{1}{2}\left(\frac{z-\alpha_{i3}}{\beta_{i3}}\right)^2\right];$$

- композиции в виде произведения:

$$[A_i(x)\ and\ B_i(y)] = A_i(x)\ B_i(y);$$

- импликации в форме (*Larsen*):

$$[A_i(x)\ and\ B_i(y) \to C_i(z)] = A_i(x)\ B_i(y) C_i(z);$$

- центроидном методе приведения к чёткости:

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_{i3} A_i B_i}{\sum_{i=1}^{n} A_i B_i};$$

где α_{i3} - центры C_i.

Иначе говоря, *Wang* доказал теорему: для каждой вещественной непрерывной функции q, заданной на компакте U, и для произвольного $\varepsilon > 0$ существует нечёткая система, формирующая выходную функцию $f(x)$ такую, что

$$sup_{x \in U} \|q(x) - f(x)\| \le \varepsilon,$$

где $\|q(x) - f(x)\|$ - символ принятого расстояния между функциями.

2) В 1995 г. *Castro* показал, что логический контроллер *Mamdani* также является универсальным аппроксиматором, при:

- симметричных треугольных функциях принадлежности:

$$A_i(x) = \begin{cases} 1 - \dfrac{|a_i - x|}{a_i}, \text{если } |a_i - x| \le \alpha_i, \\ 0, \text{если } |a_i - x| > \alpha_i, \end{cases}$$

$$B_i(y) = \begin{cases} 1 - \dfrac{|b_i - y|}{\beta_i}, \text{если } |b_i - y| \le \beta_i, \\ 0, \text{если } |b_i - x| > \beta_i, \end{cases}$$

$$C_i(z) = \begin{cases} 1 - \dfrac{|c_i - z|}{\gamma_i}, \text{если } |c_i - z| \le \gamma_i, \\ 0, \text{если } |c_i - z| > \gamma_i, \end{cases}$$

- композиции с использованием операции min:

$$[A_i(x)\ and\ B_i(y)] = min\{A_i(x)\ B_i(y)\};$$

- импликации в форме Mamdani и центроидного метода приведения к чёткости:

$$z_0 = \frac{\sum_{i=1}^{n} c_i\ min\{A_i(x)\ B_i(y)\}}{\sum_{i=1}^{n} min\{A_i(x)\ B_i(y)\}};$$

где c_i - центры C_i.

Вообще говоря, системы с нечёткой логикой *целесообразно* применять в следующих случаях:

- для сложных процессов, когда нет простой математической модели;
- если экспертные знания об объекте или о процессе можно сформулировать только в лингвистической форме.

Системы, базирующиеся на нечёткой логике, применять *нецелесообразно:*

- если требуемый результат может быть получен каким-либо другим (стандартным) путём;
- когда для объекта или процесса уже найдена адекватная и легко исследуемая математическая модель.

Отметим, что *основными недостатками* систем с нечёткой логикой являются:

- исходный набор постулируемых нечётких правил формулируется экспертом-человеком и может оказаться неполным или противоречивым;
- вид и параметры функций принадлежности, описывающих входные и выходные переменные системы, выбираются субъективно и могут оказаться не вполне отражающими реальную действительность.

3. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Как известно, аппарат нечетких множеств и нечеткой логики уже давно (более десяти лет) с успехом применяется для решения задач, в которых исходные данные являются ненадежными и слабо формализованными. Сильные стороны такого подхода:

- описание условий и метода решения задачи на языке, близком к естественному языку;
- универсальность: согласно знаменитой теореме FAT (Fuzzy Approximation Theorem), доказанной Б.Коско (B/Kosko) в 1993г., любая математическая система может быть аппроксимирована системой, основанной на нечеткой логике;
- эффективность (связана с универсальностью), поясняемая рядом теорем, аналогичных теоремам о полноте для искусственных нейронных сетей, например, теоремой вида: для каждой вещественной непрерывной функции q, задается на компакте U и для произвольного $\varepsilon > 0$ существует некоторая нечеткая экспертная система, формирующая выходную функцию $f(x)$ такую, что $sup_U \|q(x) - f(x)\| \leq \varepsilon$.

Вместе с тем для некоторых нечетких экспертных и управляющих систем характерны и определенные недостатки:

- исходный набор постулирующих нечетких правил формулируется экспертом-человеком и может оказаться неполным и противоречивым;
- вид и параметры функций принадлежности, описывающие входные и выходные параметры системы, выбираются субъективно и могут оказаться не вполне отражающими реальную действительность.

Для устранения, по крайней мере, частично, указанных недостатков рядом авторов было предложено выполнять нечеткие экспертные и управляющие системы адаптивными элементами – корректируя по мере работы системы, и правила и параметры функции принадлежности. Среди нескольких вариантов такой адаптации одним из самых удачных, по-видимому, являются так называемые искусственные нейронные сети.

Искусственные нейронные сети состоят из нейроноподобных элементов, соединенных между собой в сеть. Существуют статические и динамические нейронные сети. В статических нейронных сетях изменение параметров системы происходит по некоторому алгоритму в процессе обучения. После обучения параметры сети не меняются. В динамических нейронных сетях отображение внешней информации, и ее обработка осуществляется в виде некоторого динамического процесса, то есть процесса зависящего от времени.

В связи с этим данные лабораторные работы направлены на вырабатывание практических навыков построения искусственных нейронных сетей, реализации аппарата нечеткой логики применительно к задачам

управления и поддержки принятия решений в условиях неопределенности.

Особое внимание уделено программной реализации моделей указанных подходов инструментальными средствами математической среды MATLAB.

3.1. Лабораторная работа №1 «Аппроксимация функции одной переменной»

Цель работы: научиться работать с сетью прямой передачи данных, разобраться с функцией $newff$ [27]. Разобраться с алгоритмом обратного распространения ошибки.

Краткие теоретические сведения.

Рассматривается нейронная сеть с прямой передачей данных (с прямой связью) [20], то есть сеть, в которой сигналы передаются только в направлении от входного слоя к выходному, и элементы одного слоя связаны со всеми элементами следующего слоя. Важным для реализации нейронных сетей является определение алгоритма обучения сети.

В настоящее время одним из эффективных и обоснованных методов обучения нейронных сетей является ***алгоритм обратного распространения ошибки***, который применим к ***однонаправленным многослойным сетям***. В многослойных нейронных сетях имеется множество скрытых нейронов, входы и выходы которых не являются входами и выходами нейронной сети, а соединяют нейроны внутри сети, то есть ***скрытые нейроны***. Занумеруем выходы нейронной сети индексом $j = 1,2,\dots,n,$ а обучающие примеры индексом $M = 1,2,\dots,M_0$. Тогда в качестве целевой функции можно выбрать функцию ошибки как сумму квадратов расстояний между реальными выходными состояниями y_{jM} нейронной сети, выдаваемых сетью на входных данных примеров, и правильными значениями функции d_{jM}, соответствующими этим примерам. Пусть $x = \{x_i\}$ – столбец входных значений, где $i = 1,2,\dots,n$. Тогда $y = \{y_j\}$ – выходные значения, где $j = 1,2,\dots,m$. В общем случае $n \neq m$. Рассмотрим разность $y_{jM} - d_{jM}$, где d_{ji} – точное (правильное) значение из примера. Эта разность должна быть минимальна. Введем расстояния согласно евклидовой метрике, определив норму

$$\|y - d\| = \sqrt{(y - d, y - d)^2}. \tag{3.1}$$

Пусть целевая функция имеет вид:

$$E = \frac{1}{2}\sum_{jM}\left(y_{jM} - d_{jM}\right)^2. \tag{3.2}$$

Коэффициент $^1/_2$ выбран из соображений более короткой записи последующих формул. Задача обучения нейронной сети состоит в том, чтобы

найти такие коэффициенты $w_{\beta k}$, при которых достигается минимум $E(w)(E \geq 0)$.

На рисунке 58 показана нейронная сеть с прямой передачей данных.

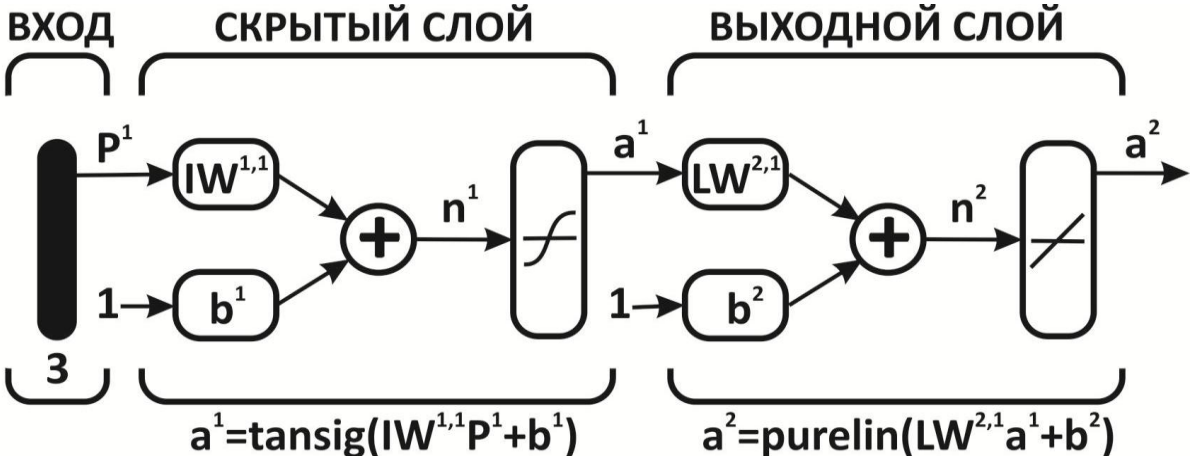


Рис. 58. Схема архитектуры нейронной сети с прямой передачей данных

Здесь приняты обозначения, используемые в [27], а именно, P^1 - вектор входа, IW^{ij}, LW^{ij} - матрицы весов входа и выхода, b^i - смещение, a^i - выход слоя, y - выход сети, *tansig* (гиперболическая тангенциальная), *purelin* (линейная) соответствующие функции активации.

Веса и смещения определяются с помощью алгоритма обратного распространения ошибок [44].

Обучение сети обратного распространения требует выполнения следующих операций:

1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Скорректировать веса сети так, чтобы минимизировать ошибку.
5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет минимума.

Пример решения типовой задачи.

Выполнение лабораторной работы состоит из следующих этапов: прежде всего, необходимо оцифровать график функции $y = f(x)$, то есть получить ряд соответствующих значений по горизонтальной и вертикальной осям.

В примере, показанном на рисунке 59 были получены два массива, каждый из которых состоит из 15 значений.

По горизонтальной оси –

[0.10 0.31 0.51 0.72 0.93 1.14 1.34 1.55 1.76 1.96 2.17 2.38 2.59 2.79 3.00].

По вертикальной оси –

[0.1010 0.3365 0.6551 1.1159 1.7632 2.5847 3.4686 4.2115
4.6152 4.6095 4.2887 3.8349 3.4160 3.1388 3.0603].

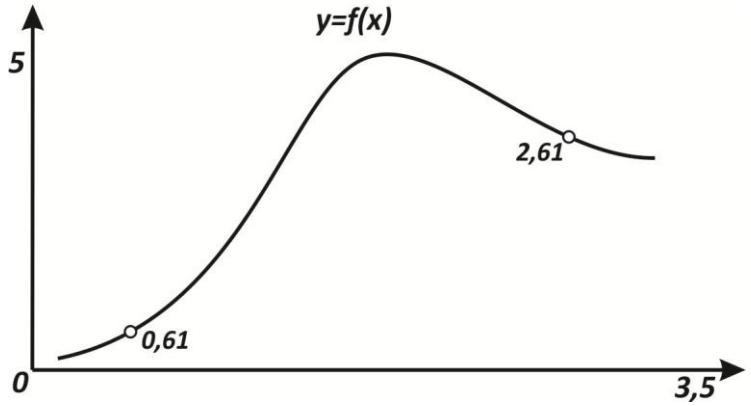


Рис. 59. Пример зависимости для функции одной переменной

Ниже приводится программа создания, обучения нейронной сети и вывода результатов.

```
x=[0.10 0.31 0.51 0.72 0.93 1.14 1.34 1.55 1.76 1.96 2.17 2.38 2.59 2.79
   3.00];
y=[0.1010 0.3365 0.6551 1.1159 1.7632 2.5847 3.4686 4.2115 4.6152 4.6095
   4.2887 3.8349 3.4160 3.1388 3.0603];
net=newff ( [0 3], [5 1], {'tansig', 'purelin'}, 'trainbfg');
net.trainParam.epochs = 300;
net.trainParam.show = 50;
net.trainParam.goal = 1.37e-2;
[net, tr]=train (net, x, y);
an = sim (net, x);
plot (x, y, '+r', x, an, '-g');
hold on;
xx = [0.61  2.61];
v = sim (net, xx);
plot (xx, v, 'ob', 'MarkerSize', 5, 'LineWidth', 2);
```

Описание приведенной программы.

Ввод одномерной матрицы:

x = [0.10 0.31 0.51 0.72 0.93 1.14 1.34 1.55 1.76 1.96 2.17 2.38 2.59 2.79 3.00];

Внешний вид матрицы:	Синтаксис:
$A = \begin{Bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{Bmatrix}$	A = [$x11$ $x12$ $x13$; $x21$ $x22$ $x23$; $x31$ $x32$ $x33$];

Создаем двухслойную однонаправленную сеть. Диапазон входных

значений от 0 до 3, первый слой состоит из 5 нейронов с функциями активации $TANSIG$, второй слой содержит один нейрон с функцией активации $PURELIN$:

$$Net = newff\ ([0\ 3],[5\ 1],\{'tansig'\ \ 'purelin'\},\ \ 'trainbfg');$$

Синтаксис:

$$newff(pr,[s1 \dots sni],\{tf1 \dots tfni\},btf,blf,pf)$$

pr - $R \times 2$ матрица минимальных и максимальных значений строк входной матрицы с размерностью $R \times Q$.

s_i - количество нейронов в $i-$ *ом* слое, N_i – количество слоев.

tf_i - функция активации $i\ -$ го слоя:

$tansig$ - сигмоидная функция в виде гиперболического тангенса;

$purelin$ - линейная функция активации;

$logsig$ - логарифмическая сигмоидная функция активации;

btf - обучающая функция обратного распространения:

$trainlm$ - функция тренировки сети, которая модифицирует значения весов и смещений в соответствии с методом оптимизации Левенберга-Маркара. Обеспечивает наиболее быстрое обучение. Но она требует много памяти.

$trainbfg$ - обучает сеть путем модификации весов и смещений в пакетном режиме, которая работает медленнее, но требует меньше памяти.

$trainrp$ - функция обучения сети, которая модифицирует веса и смещения в соответствии с алгоритмом упругого обратного распространения, которая работает еще медленнее.

$net.trainParam.epochs = 300;$ - максимальное количество эпох тренировки;

$net.trainParam.show = 50;$- количество эпох между графиками;

$net.trainParam.goal = 1.37e-2;$ - условие остановки по отклонению от эталона;

$[net,tr]\ =\ train\ (net,x,y);$ - тренировка нейронной сети:

1. В качестве входных параметров использует:
   - net – сеть; x – входные значения сети; y – целевые значения сети.
2. В качестве выходных параметров использует:
   - net – сеть; tr - результат тренировки (количество эпох и функция выполнения).

$an = sim(net,x);$ - оператор, который моделирует нейронную сеть;

$$plot\ (x,y,'+r',x,an,'-g');$$

Команда $plot(x1,y1,s1,x2,y2,,\dots)$ позволяет объединить на одном графике несколько функций $y1(x1),y2(x2),\dots$, определив для каждой из них свой способ отображения $s1$ или $s2$, может включать до трех символов,

характеристики которых показаны в таблице 3.1.

Таблица 3.1.

Способы отображения графиков

Тип линии		Тип точки		Цвет	
Непрерывная	-	Точка	.	Желтый	y
Штриховая	--	Плюс	+	Фиолетовый	m
Двойной пунктир	:	Звездочка	*	Голубой	c
Штрих-пунктирная	-.	Кружок	o	Красный	r
		Крестик	x	Зеленый	g
				Синий	b
				Белый	w
				Черный	k

hold on; - включает режим сохранения текущего графика и свойств объекта, так, что последующие команды приведут к добавлению новых графиков в этом же графическом окне.

В результате выполнения программы получаются результаты, отражённые на рисунках 60 - 61.

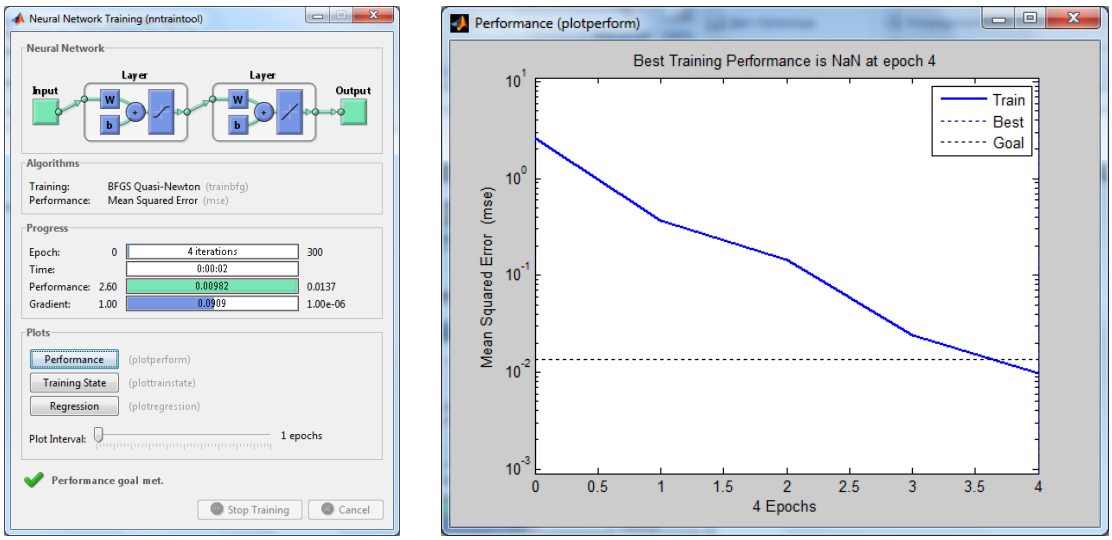


Рис. 60. Характеристика нейронной сети и ее точность обучения в зависимости от числа эпох обучения

В массиве v содержатся приближённые значения для двух контрольных точек, указанных на графике (рис. 59) $xx = [0.61\ \ 2.61]$. При данных параметрах сети получены значения: $v = [1.05\ \ 3.35]$. Сравнив эти приближённые значения с точными значениями $[0.85\ \ 3.37]$, можно сделать вывод о корректности построения нейронной сети.

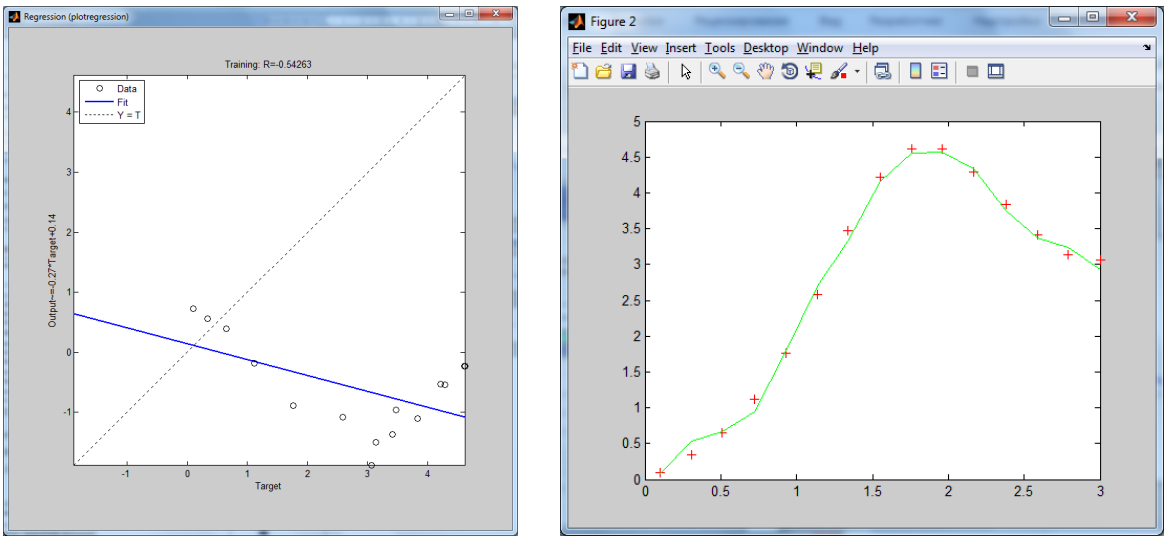


Рис. 61. Результаты моделирования сети: + - исходные данные; сплошная линия и символ «о» - результаты моделирования всей зависимости и в контрольных точках

Отчет о выполнении лабораторной работы №1 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Исходные данные (рисунок 59).
2. Текст программы с подробными комментариями.
3. Характеристику точности обучения (рисунок 60).
4. Результаты моделирования (рисунок 61).
5. Сопоставление результатов в контрольных точках.
6. Краткие выводы о результатах работы.

3.2. Лабораторная работа №2 «Аппроксимация функции двух переменных»

Цель работы: научиться работать с радиальной базисной сетью, изучить функции $newrbe$ и $newrb$.

Краткие теоретические сведения.

Радиальные базисные сети предназначены для аппроксимации функций. Возьмем произвольную непрерывную функцию и представим ее с помощью суммы колоколообразных функций. Аналитически это означает представление $f(x)$ в виде разложения по стандартному набору пространственно локализованных функций:

$$f(x) = \sum_i w_i \varphi(\|x - c_i\|), \tag{3.3}$$

где w_i - веса суммирования отдельных откликов, c_i - центры базисных радиальных функций. Это формула нейронной сети на основе радиальной базисной функции. Расстояние $\|x - c\|$ определяется как расстояние в

евклидовом пространстве:

$$\|x - c\| = AB = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + \cdots + (x_n - c_n)^2} \qquad (3.4)$$

Функция $newrbe$ формирует радиальную базисную сеть с нулевой ошибкой. Сеть с радиальными базисными функциями представляет собой, как правило, сеть с тремя слоями: обычным входным слоем, скрытым радиальным базисным слоем и выходным линейным слоем. Функция $newrb$ формирует радиальную базисную сеть с ненулевой ошибкой в отличие от функции $newrbe$. На рисунке 62 показана архитектура радиальной базисной сети.

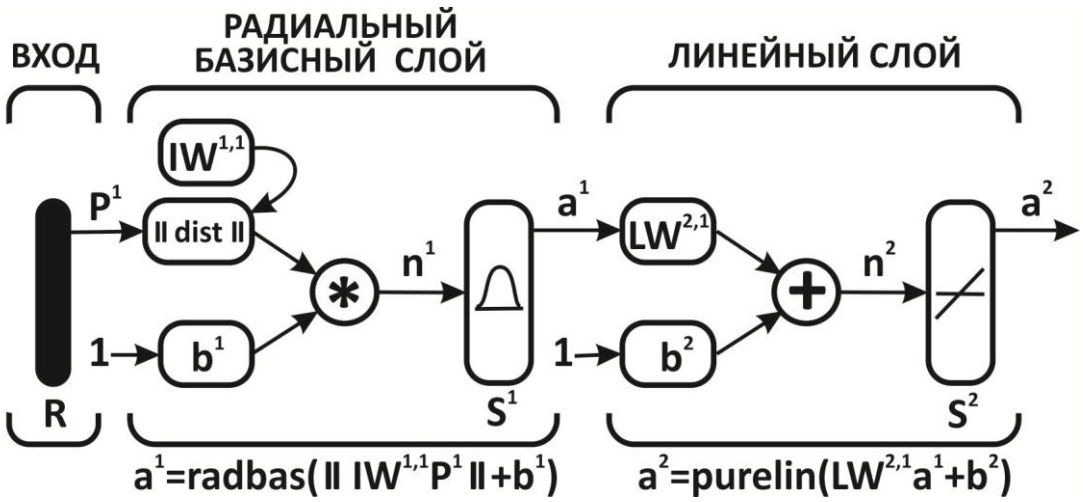


Рис. 62. Схема архитектуры радиальной базисной сети

Пример решения типовой задачи.

Пусть функция $z = e^{-x^2} \cdot e^{-y^2}$ задана на промежутках $x \in [-1,1]$, $y \in [-1{,}5; 1{,}5]$; количество точек разбиений по x есть nx, а по y есть ny. Тогда, используя алгоритм построения радиальной базисной сети, можно построить график функции $z = f(x, y)$:

```
x1=-1.0;  x2=+1.0;  y1=-1.5; y2=+1.5;
nx=7;  ny=9;
step_x=(x2-x1)/(nx-1);   step_y=(y2-y1)/(ny-1);
step_min = min(step_x,step_y);
[x,y]=meshgrid([x1:step_x:x2], [y1:step_y:y2]);
z=exp(-x.^2).*exp(-y.^2);
surf(x,y,z),  title('PS. Press');
pause;
xx=reshape(x,1,nx*ny);
yy=reshape(y,1,nx*ny);
zz=exp(-xx.^2).*exp(-yy.^2);
p=[xx;  yy];
t=zz;
```

```
goal = 0.0371;
spread = 1.0*step_min;
net = newrb(p,t, goal,spread);
net.layers{1}.size
smlt=sim(net,p);
[zz' smlt']
smltr=reshape(smlt,ny,nx);
surf(x,y,smltr), title('AS. Press');
```

Описание приведенной программы.

$$step\_min = min(step\_x, step\_y);$$

Данная функция возвращает массив тех же размеров, какие имеют массивы $step\_x$ и $step\_y$, каждый элемент, которого есть минимальный из элементов этих массивов.

$$[x, y] = meshgrid\ ([x1: step\_x: x2], [y1: step\_y: y2]);$$

Функция $Meshgrid$ – задает прямоугольную сетку на плоскости.

Функция $[X, Y] = meshgrid(x, y)$ формирует массивы X и Y, которые определяют координаты узлов прямоугольника, задаваемого векторами x и y. Этот прямоугольник задает область определения функции от двух переменных, которую можно построить в виде $3D$-поверхности.

$$z = exp(-x.\hat{}2).* exp(-y.\hat{}2);$$

где $(-x.\hat{}2)$ – точка после X определяет возведение каждого элемента массива в степень.

$$surf(x, y, z), title\ ('PS. Press < enter > ');$$

Функция $\boldsymbol{surf}$ строит каркасную поверхность графика функции и заливает каждую клетку поверхности определенным цветом, зависящим от значений функции в точках, соответствующих углам клетки. В пределах каждой клетки цвет постоянный. Команда $title$(‘текст’); - размещает текст над графиком.

Функция $pause$; - устанавливает паузу между отдельными шагами алгоритма, например, при выводе графиков. Команда $pause$ < без параметров >; останавливает выполнение до тех, пока не будет нажата какая-нибудь клавиша. Чтобы реализовать паузу в n секунд, необходимо применить оператор $pause(n)$.

$$xx = reshape(x, 1, nx * ny);$$

Функция $B = reshape(A, m, n)$ возвращает массив размером $m \times n$, сформированный из элементов массива A путем их последовательной выборки по столбцам. Если число элементов массива A не равно произведению $m * n$, выводится сообщение об ошибке.

$net = newrb(p, t, goal, spread);$ – функция конструирования сети с радиальным базисом.

$net.layers\{1\}.size$ - свойство определяющее значения весов и смещений сети, а также количество задержек, связанных с каждым весом

$smlt = sim(net, p);$ - функция SIM моделирует нейронную сеть.

В результате выполнения программы формируется график исходной функции двух переменных, приведенный на рисунке 63.

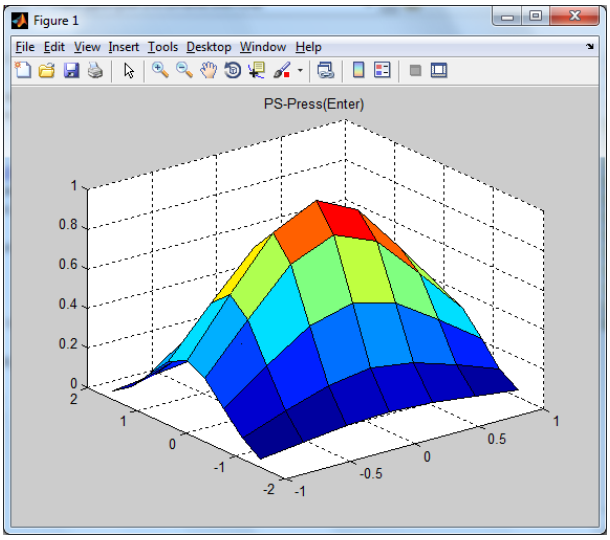


Рис. 63. График исходной функции двух переменных

На рисунке 64 показана характеристика точности обучения радиальной базисной сети и допустимая среднеквадратичная ошибка сети $Goal = 0.0371$.

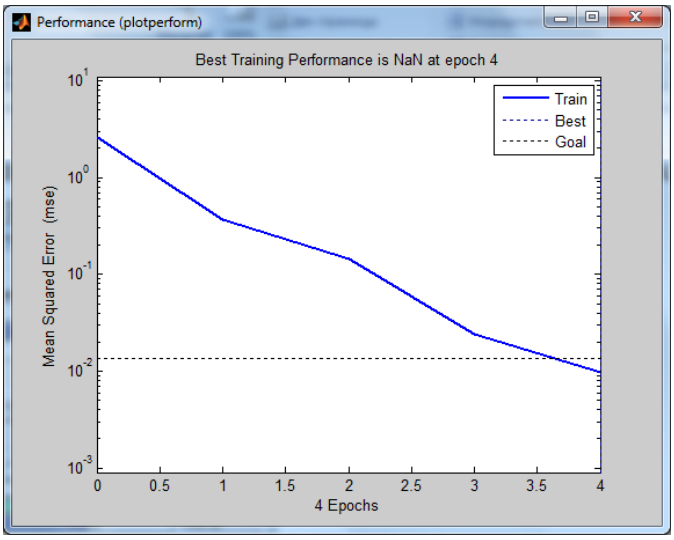


Рис. 64. Характеристика точности обучения в зависимости от количества эпох обучения

На рисунке 65 отображен полученный результат аппроксимации

нелинейной зависимости, построенный с помощью радиальной базисной функции.

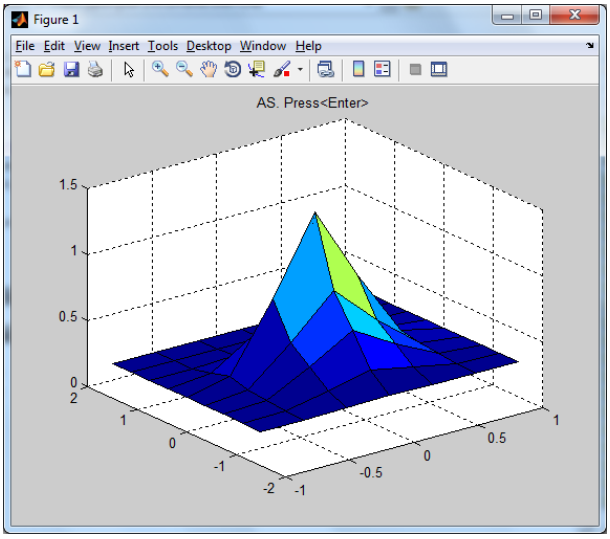


Рис. 65. Результат моделирования исходной функции

Сравнивая результаты, полученные на рисунках 64 и 65, можно сделать вывод об удовлетворительности полученных результатов. Лучший результат можно получить, варьируя параметры $goal$ и $spread$.

Отчет о выполнении лабораторной работы №2 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Исходные данные – выбор функции двух переменных и области определения функции, построение графика функции (рис. 63).
2. Текст программы с подробными комментариями.
3. Результаты моделирования (рис. 64, 65).
4. Контрольный пример.
5. Объяснение результатов проделанной работы.

3.3. Лабораторная работа №3
«Сеть Кохонена, самоорганизующаяся нейронная сеть»

Цель работы: изучить функционирование и процедуру обучения сети Кохонена с помощью функции $newc$.

Краткие теоретические сведения.

Сети, называемые картами Кохонена, - это одна из разновидностей нейронных сетей, однако они принципиально отличаются от рассмотренных выше, поскольку используют неконтролируемое обучение. Напомним, что при таком обучении обучающее множество состоит лишь из значений входных переменных, в процессе обучения нет сравнивания выходов нейронов с эталонными значениями. Можно сказать, что такая сеть учится понимать структуру данных.

В основе идеи сети Кохонена лежит аналогия со свойствами человеческого мозга. Кора головного мозга человека представляет собой плоский лист и свернута складками. Таким образом, можно сказать, что она обладает определенными топологическими свойствами (участки, ответственные за близкие части тела, примыкают друг к другу и все изображение человеческого тела отображается на эту двумерную поверхность).

Самоорганизующиеся карты могут использоваться для решения таких задач, как моделирование, прогнозирование, поиск закономерностей в больших массивах данных, выявление наборов независимых признаков и сжатие информации. Наиболее распространенное применение сетей Кохонена - решение задачи классификации без учителя, т.е. кластеризации. Вот два из распространенных применений карт Кохонена: разведочный анализ данных и обнаружение новых явлений [3].

Разведочный анализ данных. Сеть Кохонена способна распознавать кластеры в данных, а также устанавливать близость классов. Таким образом, пользователь может улучшить свое понимание структуры данных, чтобы затем уточнить нейросетевую модель. Если в данных распознаны классы, то их можно обозначить, после чего сеть сможет решать задачи классификации. Сети Кохонена можно использовать и в тех задачах классификации, где классы уже заданы, - тогда преимущество будет в том, что сеть сможет выявить сходство между различными классами.

Обнаружение новых явлений. Сеть Кохонена распознает кластеры в обучающих данных и относит все данные к тем или иным кластерам. Если после этого сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну.

На вход сети подаются последовательно значения векторов $x_L = (x_1, x_2, \dots, x_n)_L$, представляющих отдельные последовательные наборы данных для поиска кластеров, то есть различных классов образов, причем число этих кластеров заранее неизвестно. На стадии обучения (точнее самообучения) сети входной вектор x попарно сравнивается со всеми векторами w_j всех нейронов сети Кохонена. Вводится некоторая функция близости d (например, в виде евклидова расстояния). Активный нейрон с номером c слоя Кохонена, для которого значение функции близости $d(x, w_c)$ между входным вектором x, характеризующим некоторый образ, к векторам w_c максимально, объявляется «победителем». При этом образ, характеризующийся вектором x, будет отнесен к классу, который представляется нейроном - «победителем».

Рассмотрим алгоритм самообучения сетей Кохонена. Обозначим функцию близости $z = \|x - w\|$. Выигрывает нейрон с

$$\|x - w_c\| = min\|x - w_j\|. \qquad (3.5)$$

Близость x' и w' можно переделить, пользуясь скалярным произведением, как

$$z_j = 1 - \left(x^{'}, w^{'}{}_j\right) = 1 - \sum_{i=1}^{n} x^{'}{}_i w_{ij}\,. \qquad (3.6)$$

На стадии самообучения сети Кохонена осуществляется коррекция весового вектора не только нейрона - «победителя», но и весовых векторов остальных активных нейронов слоя Кохонена, однако, естественно, в значительно меньшей степени – в зависимости от удаления от нейрона - «победителя». При этом форма и величина окрестности вокруг нейрона - «победителя», весовые коэффициенты нейронов которой также корректируются, в процессе обучения изменяются. Сначала начинают с очень большой области – она, в частности, может включать все нейроны слоя Кохонена. Изменение весовых векторов осуществляется по правилу

$$w_j(t+1) = w_j(t) + \eta(t) d_{cj}(t)\left[x(t) - w_j(t)\right], j = 1, 2, \ldots, n, \qquad (3.7)$$

где $w_j(t)$ – значение весового вектора на шаге t самообучения сети, $d_{cj}(t)$ – функция близости между нейронами слоя Кохонена и $\eta(t)$ – изменяемый во времени коэффициент шага коррекции. В качестве $\eta(t)$ обычно выбирается монотонно убывающая функция ($0 < \eta(t) < 1$), то есть алгоритм самообучения начинается сравнительно большими шагами адаптации и заканчивается относительно малыми изменениями.

В результате n – мерное входное пространство R_n отобразится на m – мерную сетку (слой Кохонена). Это отображение реализуется в результате рекуррентной (итеративной) процедуры самообучения (*unsupervised learning*). Отличительная особенность этого отображения – формирование кластеров (*clusters*) или классов. По завершении процесса самообучения на стадии реального использования сети Кохонена неизвестные входные образы относятся к одному из выявленных кластеров (классов) по близости к некоторому весу, принадлежащему определенному кластеру, выявленному на стадии самообучения.

На рисунке 66 показана архитектура слоя Кохонена.

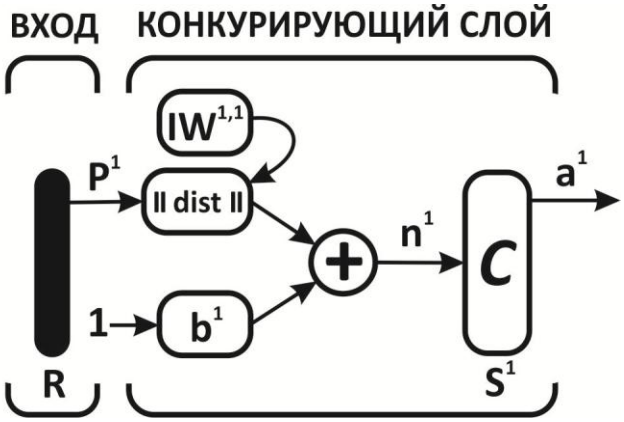


Рис. 66. Схема архитектуры слоя Кохонена

Пример решения типовой задачи.

Пусть заданы 28 случайных векторов, изображённых на графике крестами (рис. 67). Оцифровав данный график, можно получить массив входных данных $P(1,:), P(2,:)$ (табл. 3.2).

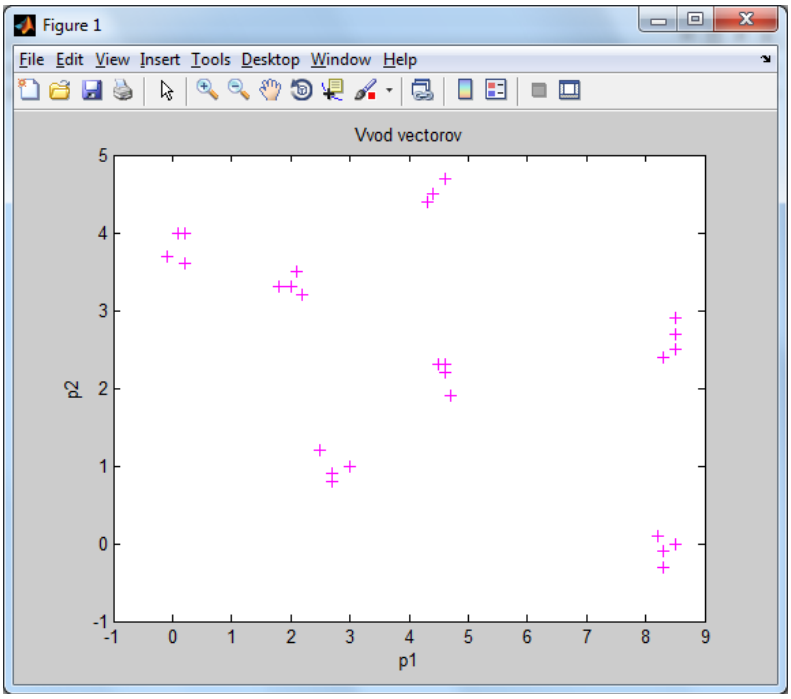


Рис. 67. Распределение входных векторов

Таблица 3.2.

Массив входных данных

	$P(1,:)$	$P(2,:)$
1.	3.0	1.0
2.	-0.1	3.7
3.	4.3	4.4
4.	4.6	2.3
5.	8.5	2.7
6.	1.8	3.3
7.	8.3	-0.1
8.	2.5	1.2
9.	0.2	3.6
10.	4.6	4.7
11.	4.5	2.9
12.	8.5	2.3
13.	2.2	3.2
14.	8.2	0.1
15.	2.7	0.8
16.	0.2	4.0
17.	4.4	4.5
18.	4.6	2.2
19.	8.3	2.4
20.	2.0	3.3
21.	8.5	0.0
22.	2.7	0.9
23.	0.1	4.0
24.	4.3	4.4
25.	4.7	1.9
26.	8.5	2.5
27.	2.1	3.5
28.	8.3	-0.3

Следующий алгоритм демонстрирует процедуру обучения самоорганизующейся нейронной сети Кохонена.

```
plot(P(1,:), P(2,:), '+m');
title('Vvod vectorov');
xlabel('P(1,:)');  ylabel('P(2,:)');
hold on;
nclusters = 7;
nvectors = 4;
a1 = -10;   a2 = +10;
b1 = -5;    b2 = +5;
net = newc([a1 a2; b1 b2], nclusters, 0.1, 0.0005);
wo = net.IW{1};  bo = net.b{1};  co = exp(1)./bo;
net.trainParam.epochs=49;
net.trainParam.show=7;
net = train(net,P);
w = net.IW{1};   bn = net.b{1};   cn = exp(1)./bn;
plot(w(:,1),w(:,2),'kp');
```

Описание приведенной программы.

$$xlabel\ ('p1'); \quad ylabel\ ('p2');$$

Функция $xlabel('текст')$ помещает текст для двумерного графика вдоль оси x, для трехмерного графика - вдоль оси x либо под графиком.

Функция $ylabel('текст')$ помещает текст для двумерного графика вдоль оси y, для трехмерного графика - вдоль оси y либо под графиком.

$$nclusters = 7; \quad nvectors = 4;$$

Данная функция задает количество кластеров и количество векторов в кластере соответственно.

$$net = newc([a1\ a2;\ b1\ b2], nclusters, 0.1, 0.0005);$$

Функция создания конкурентного слоя нейронной сети.

$$wo = net.IW\{1\}; \quad bo = net.b\{1\};$$

Функции установки весов и смещений соответственно.

На рисунке 68 представлены исходные данные (кресты) и полученные центры кластеризации (звёзды).

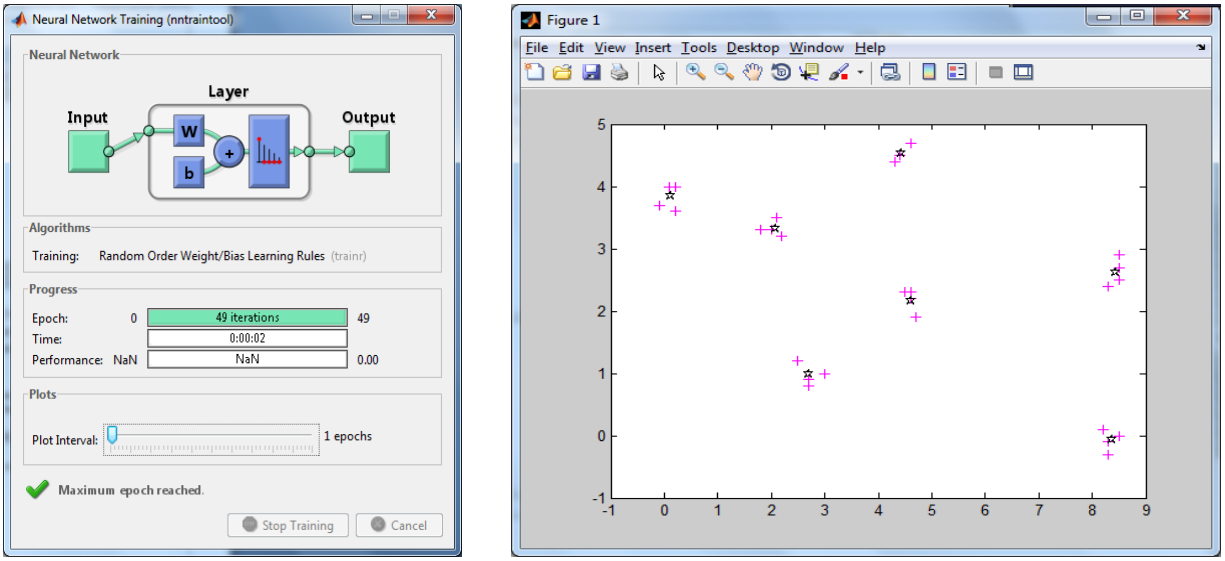


Рис. 68. Распределение исходных данных (кресты) и положение центров кластеризации (звезды)

Отчет о выполнении лабораторной работы №3 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Исходные данные (рис. 67).
2. Текст программы с подробными комментариями.
3. Результаты моделирования (рис. 68).
4. Краткое описание расположения центров кластеров.
5. Список исходных точек с указанием принадлежности к кластеру.
6. Контрольный пример.

3.4. Лабораторная работа №4
«Сеть Хопфилда»

Цель работы: научиться работать с сетью Хопфилда, изучить функцию $newhop$, исследовать устойчивость сети и её сходимость.

Краткие теоретические сведения.

Американский исследователь Хопфилд в 80-х годах 20-го века предложил специальный тип нейросетей. Названные в его честь сети Хопфилда являются рекуррентными или сетями с обратными связями и предназначены для распознавания образов. Обобщенная структура этой сети представляется, как правило, в виде системы с обратной связью выхода с входом.

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.

Сеть Хопфилда состоит из $\boldsymbol{N}$ искусственных нейронов. Граница *ёмкости памяти для сети* (то есть количество образов, которое она может запомнить) составляет приблизительно $\mathbf{15}\%$ от числа нейронов в слое Хопфилда ($N * 0{,}15$). При этом запоминаемые образы не должны быть сильно коррелированны.

В сети Хопфилда входные сигналы нейронов являются одновременно и выходными сигналами сети: $x_i(k) = y_i(k-1)$, при этом возбуждающий вектор особо не выделяется. В классической системе Хопфилда отсутствует связь нейрона с собственным выходом, что соответствует $w_{ij} = 0$, а вся матрица весов является симметричной: $w_{ij} = w_{ji}$.

$$\widehat{W} = \widehat{W}^T. \tag{3.8}$$

Симметричность матрицы весов гарантирует сходимость процесса обучения. Процесс обучения сети формирует зоны притяжения некоторых точек равновесия, соответствующих обучающим данным. При использовании ассоциативной памяти мы имеем дело с обучающим вектором $x = (x_1, x_2, \dots, x_n)$, либо с множеством этих векторов, которые в результате проводимого обучения определяют расположение конкретных точек притяжения (аттракторов). Каждый нейрон имеет функцию активации сигнум со значениями ± 1:

$$sign(a) = \begin{cases} 1, \text{если } a \geq 0 \\ -1, \text{если } a < 0. \end{cases} \tag{3.9}$$

Благодаря своей биполярной природе нейроны сети Хопфилда иногда

называют «***спинами***».

Выходной сигнал $i-$ го нейрона определяется функцией:

$$y_i = sign\left(\sum_{j=1}^{N} w_{ij}\, x_j + b_i\right), \tag{3.10}$$

где N - обозначает количество нейронов, $N = n$. Часто постоянная составляющая b_i , определяющая порог срабатывания отдельных нейронов, равна 0. Тогда циклическое прохождение сигнала в сети Хопфилда можно представить соотношением:

$$y_i(k) = sign\left(\sum_{j=1}^{N} w_{ij}\, y_j\,(k-1)\right) \tag{3.11}$$

с начальным условием $y_i(0) = x_i$.

В процессе функционирования сети Хопфилда можно выделить два режима: обучения и классификации. В режиме обучения на основе известных обучающих выборок x_i подбираются весовые коэффициенты w_{ij}. В режиме классификации при зафиксированных значениях весов и вводе конкретного начального состояния нейронов $y(0) = x$ возникает переходный процесс, протекающий в соответствии с выражением (3.9) и заканчивающийся в одном из локальных устойчивых положений, задаваемом биполярным вектором со значениями $y_i = \pm 1$, для которого $y(k) = y(k-1)$.

Обучение не носит рекуррентного характера. Достаточно ввести значения (правило Хебба) весов, выразив их через проекции вектора точки притяжения эталонного образа:

$$w_{ij} = \frac{1}{N} x_i x_j, \tag{3.12}$$

В соответствии с этим правилом сеть дает правильный результат при входном примере, совпадающим с эталонным образцом, поскольку:

$$\sum_{j=1}^{N} w_{ij}\, x_j = \frac{1}{N}\sum_{j=1}^{N} x_i(\, x_j\, x_j) = x_i \sum_{j=1}^{N} 1 \cdot \frac{1}{N} = x_i, \tag{3.13}$$

так как вследствие биполярности значений элементов вектора x всегда $x_j^2 = 1$.

При вводе большого количества обучающих выборок $x^{(k)}$ для $k = 1,2, \dots p$ веса w_{ij} подбираются согласно обобщенному правилу Хебба в соответствии с которым:

$$w_{ij} = \frac{1}{N}\sum_{k=1}^{L} x_i{}^{(k)} x_j{}^{(k)}. \tag{3.14}$$

Благодаря такому режиму обучения веса принимают значения, определяемые усреднением множества обучаемых выборок. В случае

множества обучаемых выборок актуальным становится вопрос о стабильности ассоциативной памяти.

Сеть Хопфилда [20] является автоассоциативной сетью (рис. 69). Дискретная сеть Хопфилда имеет следующие характеристики: она содержит один слой элементов; каждый элемент связывается со всеми другими элементами, но не связан с самим собой; за один шаг работы обновляется только один элемент сети; элементы обновляются в случайном порядке; выход элемента ограничен значениями 0 или 1.

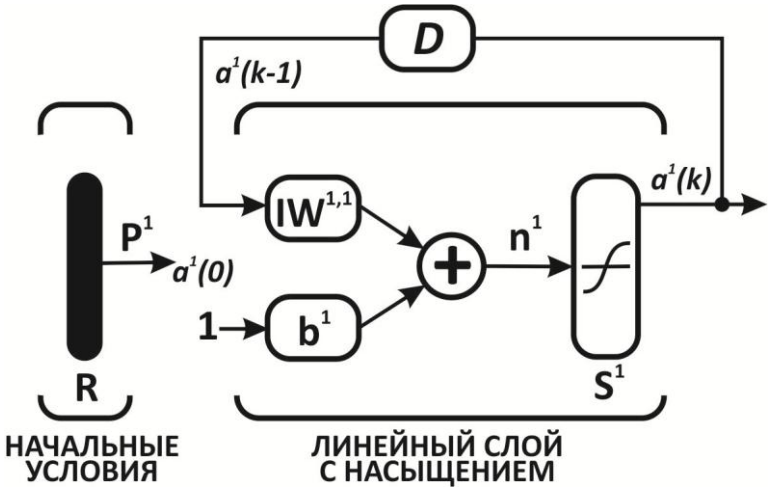


Рис. 69. Схема архитектуры модифицированной сети Хопфилда

Пример решения типовой задачи.

Рассмотрим сеть Хопфилда [27] с четырьмя нейронами и определим четыре точки равновесия:

$T = [+1\ -1;\ -1\ +1;\ +1\ +1;\ -1\ -1];$

$T = T';$

$plot(T(1,:),T(2,:),'rh',\ 'MarkerSize',13),\ \ hold\ on;$

$axis([-1.1\ \ 1.1\ \ -1.1\ \ 1.1]);$

$title('Hopfield\ \ Network\ \ State\ \ Space');$

$xlabel('a(1)');\ \ ylabel('a(2)');$

$net\ =\ newhop(T);$

$[Y,Pf,Af] =\ sim(net,4,[\,],T);$

Y

Pf

Af

$pause;$

$color\ =\ \ 'rgbmy';$

$for\ i = 1{:}25$

$a\ =\ \{rands(2,1)\};$

$[y, Pf, Af] = sim(net, \{1\ 20\}, \{\}, a);$
$record = [cell2mat(a) cell2mat(y)];$
$start = cell2mat(a);$
$plot(start(1,1), start(2,1), 'kx', record(1,:), record(2,:),$
$color(rem(i,5) + 1), 'LineWidth', 5);$
$end;$

Описание приведенной программы.

$$axis\ ([-1.1 \quad 1.1 \quad -1.1 \quad 1.1]);$$

Функция $axis([xmin\ xmax\ ymin\ ymax])$ устанавливает масштаб по осям x, y для активного графического окна.

$color = 'rgbmy';$ - свойство, задающее цветовую палитру для выводимых графиков.

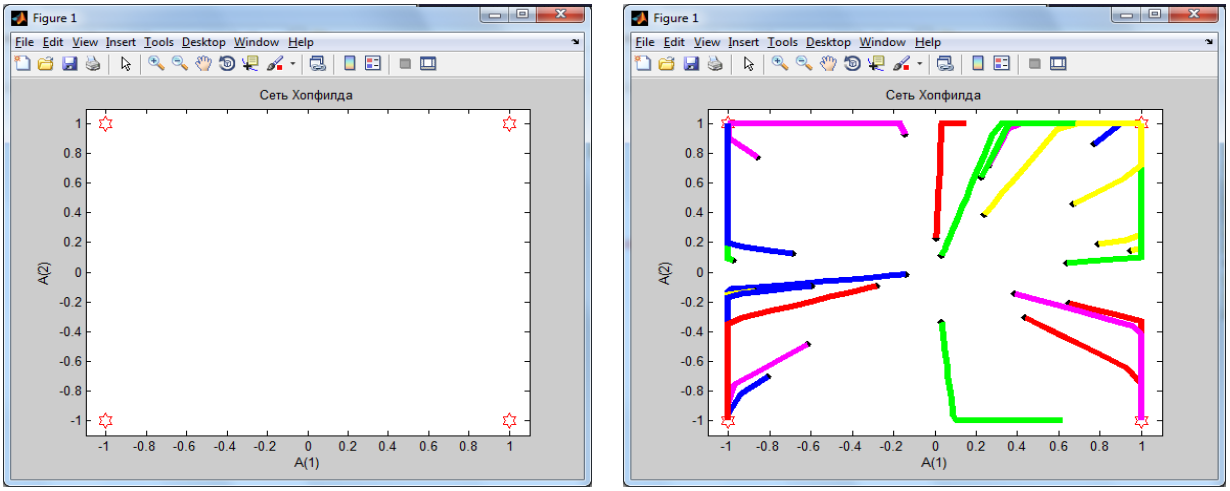


Рис. 70. Поведение обученной сети при случайных начальных условиях a

```
for i=1 : 25
        a = {rands(2,1)};
        [y,Pf,Af] = sim (net, {1 20}, {}, a);
        record=[cell2mat(a)  cell2mat(y)];
        start=cell2mat(a);
        plot(start(1,1),   start(2,1),   'kx',
             record(1,:),   record (2,:),
             color(rem(i,5)+1),   'LineWidth',  5);
end;
```

Синтаксис:

$for\ \ count = start{:}\ step{:}\ final$
операторы end ;

- оператор цикла: $count$ – переменная цикла, $start$ – ее начальное значение, $final$ – ее конечное значение, $step$ – шаг, на который увеличивается $count$ при каждом следующем заходе в цикл, цикл заканчивается, как только значение $count$ становится больше $final$.

На рисунке 70 показано поведение обученной сети при случайных начальных условиях a.

Отчет о выполнении лабораторной работы №4 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Исходные данные.
2. Текст программы с подробными комментариями.
3. Результаты моделирования (рис. 70).
4. Контрольный пример.
5. Краткие **письменные** ответы на контрольные вопросы, содержащиеся в приложении.

3.5. Лабораторная работа №5
«Формирование нечетких множеств и проведение операций с ними»

Цель работы: изучить методы построения нечетких множеств с использованием различных типов функций принадлежности. Ознакомиться с наиболее распространенными логическими операциями над нечеткими множествами.

Краткие теоретические сведения.

Функции принадлежности. Инструменты нечеткой логики в составе пакета *Matlab* содержат 11 встроенных типов функций принадлежности, формируемых на основе кусочно-линейных функций, распределения Гаусса, сигмоидной кривой, квадратических и кубических полиномиальных кривых. К наиболее простым функциям принадлежности можно отнести треугольную и трапециевидную.

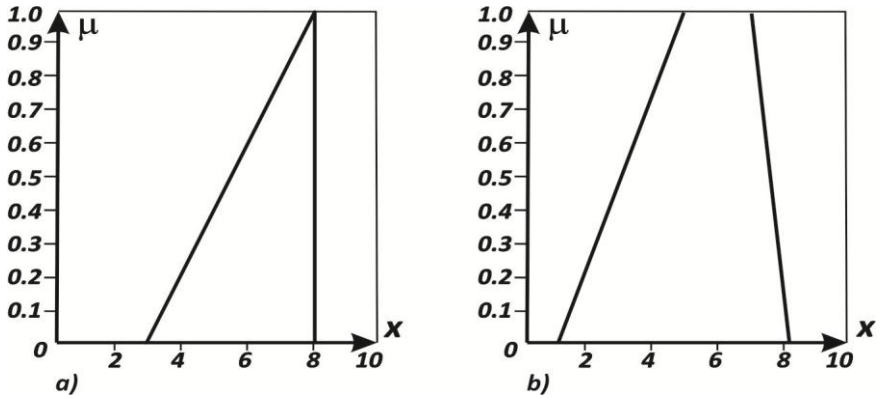


Рис. 71. Треугольная (*а*) и трапециевидная (*б*) функции принадлежности

Треугольная функция принадлежности — $trimf$ *(triangle membership function)* в параметрическом виде представляет собой набор трех точек, образующих треугольник.

$$y = trimf(x, [a\ b\ c]),$$

где вектор x — базовое множество, на котором определяется функций принадлежности. Величины a и c задают основание треугольника, b — его вершину.

В аналитическом виде треугольная функций принадлежности может быть задана следующим образом (рис.71,*а*):

$$f(x,a,b,c)=\begin{cases}0, \text{если } x<a\\ \dfrac{x-a}{b-a}, \text{если } a\le x\le b\\ \dfrac{c-x}{c-b}, \text{если } b\le x\le c\\ 0, \text{если } x>c\end{cases} \quad (3.15)$$

Рассмотрим примеры использования различных функций принадлежности в системе *Matlab.*

Примеры решения типовой задачи.

Пример 3.1. Программа использования функций принадлежности $\boldsymbol{trimf}$.

$x = 0{:}\,0{,}1{:}\,10;$ - задается базовое множество.

$y = trimf(x, [3\ 6\ 8]);$ - определяется треугольная функция принадлежности.

$plot(x, y);$ - выводится график функции.

$xlabel('trimf(x, [3\ 6\ 8])');$ - подписывается график под осью абцис.

Трапециевидная функция принадлежности — $trapmf$ (*trapezoid membership function*) — отличается от предыдущей функции лишь тем, что имеет верхнее основание.

$$y = trapmf(x, [a\ b\ c\ d]),$$

где параметры a и d — нижнее основание трапеции; b и с — верхнее основание трапеции (рис. 71, *б*).

Аналитическая запись трапециевидной функции имеет вид:

$$f(x,a,b,c,d)=\begin{cases}0, \text{если } x<a\\ \dfrac{x-a}{b-a}, \text{если } a\le x\le b\\ 1, \text{если } b<x\le c\\ \dfrac{d-x}{d-c}, \text{если } c<x\le d\\ 0, \text{если } x>d\end{cases} \quad (3.16)$$

Одно из основных достоинств треугольных и трапециевидных функций принадлежности — их простота. На основе функции распределения Гаусса можно построить функции принадлежности двух видов: простую функцию принадлежности Гаусса и двухстороннюю, образованную с помощью различных функций распределения Гаусса. Первая из них обозначается $gaussmf$ а вторая — $gauss2mf$.

$$y = gaussmf(x, [a\ c]).$$

Симметричная функция Гаусса зависит от двух параметров a и c (рис.72 – *левый рисунок*):

$$f(x, \delta, c) = e^{\frac{-(x-c)^2}{2\delta^2}} \quad (3.17)$$

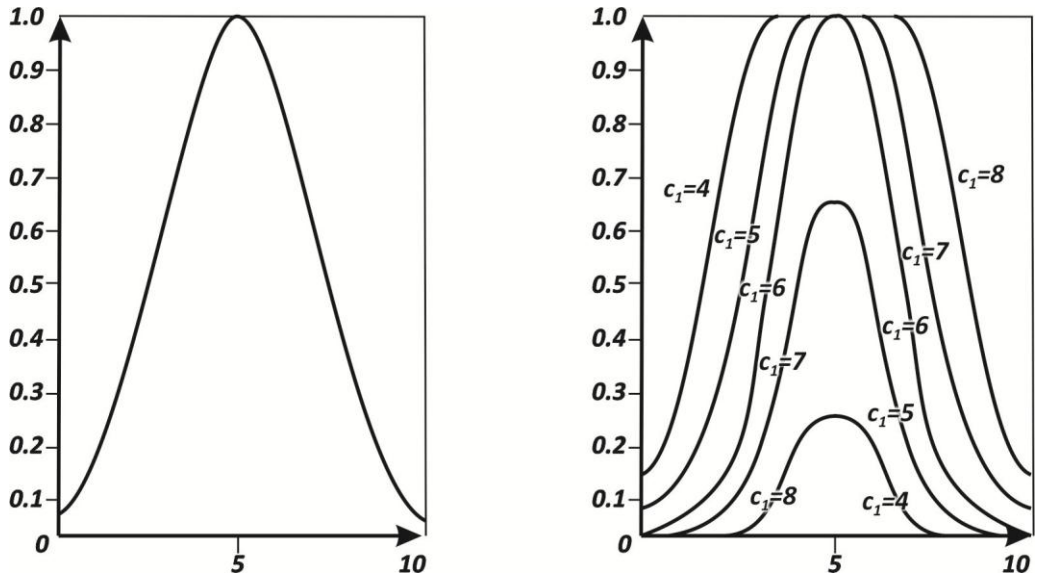


Рис. 72. Простая и двухсторонняя функции принадлежности Гаусса

Пример 3.2. Программа использования функции принадлежности $gaussmf$.

$X = 0{:}\,0{,}1{:}\,10;$

$Y = gaussmf(x, [2\ 5]);$

$plot(x, y);$

Описание функции: $y = gauss2mf(x, [a1, c1, a2, c2])$. Выражение является комбинацией двух различных функций распределения Гаусса. Первая определяется параметрами $a1$ и $c1$ и задает форму левой стороны, а вторая (параметры $a2$, $c2$,) — правой стороны функции принадлежности. Если $q < c2$, то функция $gauss2mf$ достигает своего максимального значения на уровне 1. Иначе — максимальное значение функции меньше 1 (рис. 72–*правый рисунок*).

Пример 3.3. Программа использования функции принадлежности $gauss2mf$.

$x = (0 : 0{,}1 : 10)';$

$y1 = gauss2mf(x, [2\ 4\ 18]);$

$y2 = gauss2mf(x, [2\ 5\ 17]);$

$y3 = gauss2mf\ (x, [2\ 6\ 16]);$

$y4 = gauss2mf(x, [2\ 7\ 15]);$

$y5 = gauss2mf(x, [2\ 8\ 14]);$

$plot\ (x, [y1\ y2\ y3\ y4\ y5]);$

Символ « $'$ » в строке определения базового множества x показывает транспонированность базового множества.

Следующей функцией, которая позволяет представлять нечеткие субъективные предпочтения, является функцией принадлежности «*обобщенный колокол*» и обозначается $gbellmf$ *{generalized bell shape membership function).* Ее отличие от рассмотренных ранее функций принадлежности заключается в добавлении третьего параметра, что позволяет осуществлять плавный переход между нечеткими множествами.

$$y = gbellmf(x, [a\ b\ c]).$$

Функция «обобщенный колокол» зависит от трех параметров и имеет следующую аналитическую запись:

$$f(x, a, b, c) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}} \tag{3.18}$$

где c определяет расположение центра функции принадлежности; a и b влияют на форму кривой (рис.73).

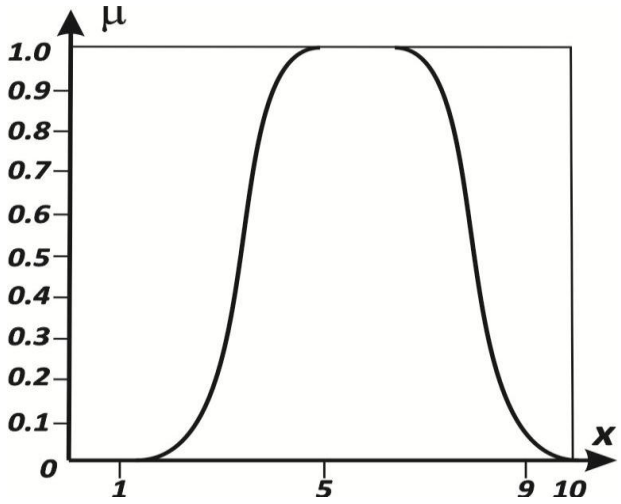


Рис. 73. Функция принадлежности «обобщенный колокол» $gbellmf, P = [2\ 4\ 6]$

Пример 3.4. Программа использования $gbellmf$.

$x = 0 : 0{,}1 : 10;$

$\mathrm{y} = gbellmf\ (x, [2\ 4\ 6]);$

$plot(x, \mathrm{y});$

$xlabel('gbellmf, P = [2\ 4\ 6]');$

Функции принадлежности на основе функции распределения Гаусса и функции принадлежности «*обобщенный колокол*» отличаются гладкостью и простотой записи. Они являются наиболее используемыми при описании

нечетких множеств. Несмотря на то, что гауссовы и колоколообразные функции принадлежности обладают свойством гладкости, они не позволяют формировать асимметричные функции принадлежности. Для этих целей предусмотрен набор сигмоидальных функций, которые могут быть открыты либо слева, либо справа в зависимости от типа функции. Симметричные и закрытые функции синтезируют с использованием двух дополнительных сигмоид. Основная сигмоидальная функция принадлежности обозначается $sigmf$, а дополнительные — $dsigmf$ и $psigmf$.

$$y = sigmf(x, [a\ c]).$$

В аналитической форме сигмоидальная функция $sigmf$ записывается следующим образом:

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}} \tag{3.19}$$

В зависимости от знака параметра a рассматриваемая функция принадлежности будет открыта или справа или слева (рис.74, *а*), что позволит применять ее при описании таких нечетких понятий, как «*очень большой*», «*крайне отрицательно*» и др.

Описание дополнительной сигмоидальной функции: $y = dsigmf(x, [a1, c1, a2, c2])$. Функция принадлежности $dsigmf$ зависит от четырех параметров $a1$, $c1$, $a2$ и $c2$ и определяется как разность двух сигмоидальных функций: $f1(x, a1, c1) - f2(x, a2, c2)$ (рис. 74, *б)*.

Описание дополнительной сигмоидальной функции: $y = psigmf\ (x, [\ a1, c1, a2, c2])$. Функция принадлежности $psigmf$, также, как и предыдущая функция, зависит от четырех параметров $a1$, $c1$, $a2$, $c2$ и определяется как произведение двух сигмоидальных функций: $f1(x, a1, c1)\ f2\{x, a2, c2)$ (рис.74, *в*).

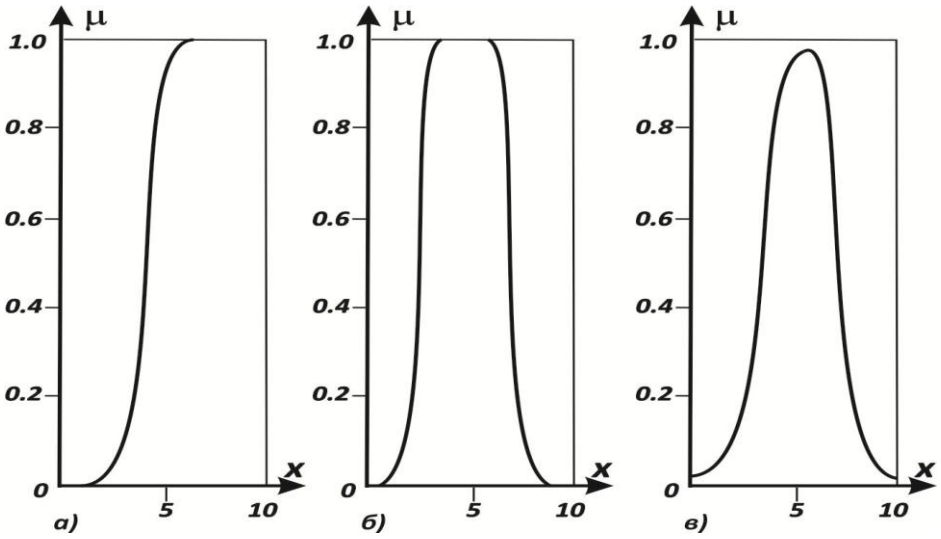


Рис. 74. Сигмоидальные функции принадлежности:
а — основная односторонняя; *б* — дополнительная двухсторонняя;
в — дополнительная несимметричная

Пример 3.5. Программа использования сигмоидных функций.

x = **0**: **0**, **1**: **10**; — определяется базовое множество

subplot (**1**, **3**, **1**); — формируется матрица графиков (**3** x **1**) — первый элемент — текущий.

$\boldsymbol{y = sigmf}\ (\boldsymbol{x}, [\mathbf{2}\ \ \mathbf{4}])$;

plot $(\boldsymbol{x}, \boldsymbol{y})$; — выводится график в первый элемент матрицы.

$\boldsymbol{xlabel}(' \boldsymbol{sigmf}, \boldsymbol{P} = [\mathbf{2}\ \ \mathbf{4}]')$;

subplot (**1**, **3**, **2**); — выбирается второй текущий элемент.

y = $\boldsymbol{dsigmf}\ (\boldsymbol{x}, [\mathbf{5}\ \ \mathbf{2}\ \ \mathbf{5}\ \ \mathbf{7}])$;

$\boldsymbol{plot}\ (\boldsymbol{x}, \boldsymbol{y})$; — выводится график во второй элемент матрицы.

$xlabel(' dsigmf, P = [5\ \ 2\ \ 5\ \ 7]')$;

$subplot(1, 3, 3)$; — выбирается третий текущий элемент.

y = $psigmf(x, [2\ \ 3\ \ -5\ \ 8])$;

$plot(x, y)$; — выводится график в третий элемент матрицы.

$xlabel\ (' psigmf, P = [2\ \ 3\ \ -5\ \ 8]')$;

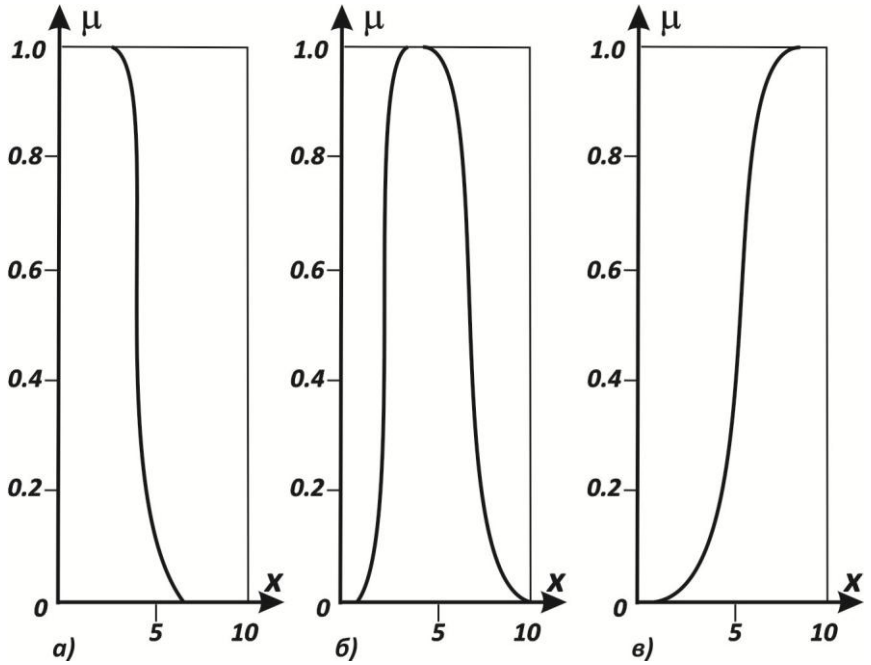


Рис. 75. Полиномиальные функции принадлежности:
а — Z - функция; *б* — PI - функция; *в* — S - функция

Инструментарий нечеткой логики (*fuzzy logic toolbox*) в составе *Matlab* предоставляет возможность формирования функции принадлежности на основе полиномиальных кривых. Соответствующие функции называются Z - функции (zmf), PI -функции ($pimf$) и 5-функции (smf). Функция ***zmf*** представляет

собой асимметричную полиномиальную кривую, открытую слева (рис.75,*а*), функция ***smf*** — зеркальное отображение функции ***zmf*** (рис.75,*б*). Соответственно функция ***pimf*** равна нулю в правом и левом пределах и принимает значение, равное единице, в середине некоторого отрезка (рис.75,*в).*

$$y = zmf\ (x, [a\ \ b]).$$

Параметры a и b определяют экстремальные значения кривой (рис.75,*а*).

$$y = pimf\ (x, [a\ \ b\ \ \text{c}\ \ d]).$$

Параметры a и d задают переход функции в нулевое значение, а параметры b и c — в единичное (рис. 75,*б*).

$$y = smf\ (x, [a\ \ b]).$$

Параметры a и b определяют экстремальные значения кривой (рис. 75,*в*).

Пример 3.6. Программа использования полиномиальных кривых.

$x = 0{:}\ 0{,}1{:}\ 10;$

$subplot(1, 3, 1);$

$y = zmf\ (x, [3\ \ 7]);\ \ plot(x, y);\ xlabel\ ('zmf, P = [3\ \ 7]');$

$subplot\ (1, 3, 2);$

$y = pimf\ (x, [1\ 4\ 5\ 10]);\ \ plot(x, y);\ xlabel\ ('pimf, P = [1\ 4\ 5\ 10]');$

$subplot\ (1, 3, 3);$

$y = smf(x, [1\ 8\ ]);\ \ plot\ (x, y);\ xlabel\ ('smf, P = [1\ \ 8]');$

Помимо рассмотренных выше функций, позволяющих представлять нечеткие множества, в *Matlab* имеется возможность формировать собственные функции принадлежности или модифицировать встроенные.

Операции с нечеткими множествами.

Выделяют три основные логические операции с нечеткими множествами: конъюнкцию, дизъюнкцию и логическое отрицание. В среде *Matlab* существует возможность определять конъюнктивные и дизъюнктивные операторы с точки зрения минимаксной и вероятностной интерпретаций.

Рассмотрим минимаксную интерпретацию логических операторов, в которой конъюнктивный оператор представляет нахождение минимума — min (рис. 76,*а*), а дизъюнктивный — нахождение максимума — max (рис. 76,*б*).

$$y = min\ ([y1;\ y2]),$$

$$y = max([y1;\ y2]).$$

Параметры $y1$ и $y2$ представляют собой исходные функции принадлежности. Функция min работает со списком функций принадлежности. В *Matlab* список оформляется квадратными скобками, а элементы списка разделяются точкой с запятой.

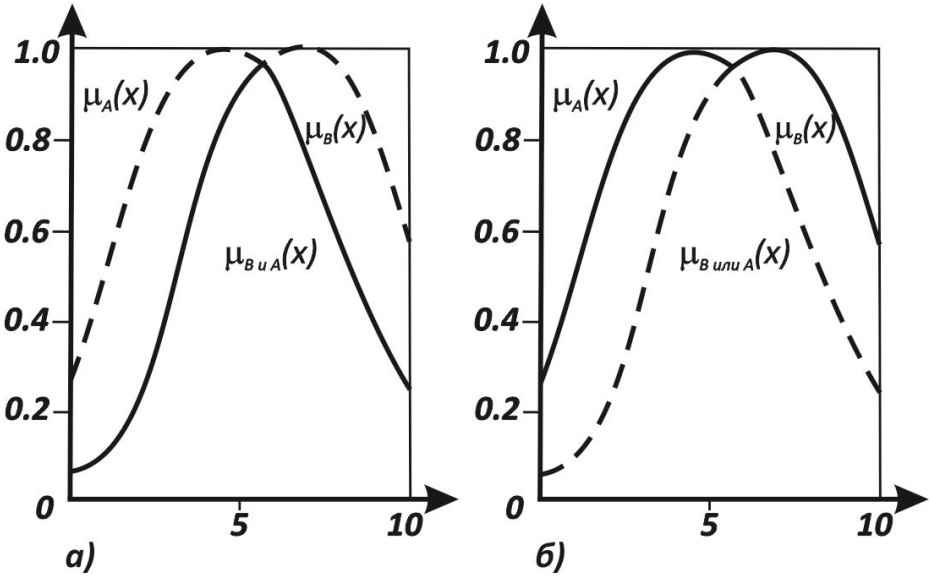


Рис. 76. Пересечение (*а*) и объединение (*б*) нечетких множеств (минимаксная интерпретация)

Пример 3.7. Программа использования операций min и max.

$x = 0{:}\ 0{,}1{:}\ 10;$

$subplot(1, 2, 1);$

$y1 = gaussmf(x, [3\ \ 5]);\ \ y2 = gaussmf\ (x, [3\ \ 7]);$

$y3 = min\ (\ [y1;\ y2]);\ \ plot\ (x, [y1;\ y2], ':')$

$hold\ on;\ \ plot\ (x, y3);\ \ hold\ off;$

$subplot(1, 2, 2);$

$y4\ =\ max([y1;\ y2]);\ \ plot\ (x, [y1;\ y2]\ , ':')\ ;$

$hold\ on;\ \ plot\ (x, y4);\ \ hold\ off;$

Пунктирной линией на графиках изображены исходные функции принадлежности, а сплошной линией — результат действия логических операторов.

Минимаксная интерпретация является наиболее распространенной при построении нечетких систем. Тем не менее, на практике довольно часто используется альтернативная вероятностная интерпретация конъюнктивных и дизъюнктивных операторов. *Matlab* содержит соответствующие функции.

В рамках данной интерпретации конъюнктивный оператор представляет собой оператор вычисления алгебраического произведения — $prod$ (рис.77,*а*), а дизъюнктивный оператор — оператор вычисления алгебраической суммы — $prober$ (рис.77,*б*).

$$y\ =\ prod\ ([y1\ ;\ y2]),$$

$$y\ =\ probor\ ([y1;\ y2]).$$

Параметры $y1$ и $y2$ представляют собой исходные функции принадлежности.

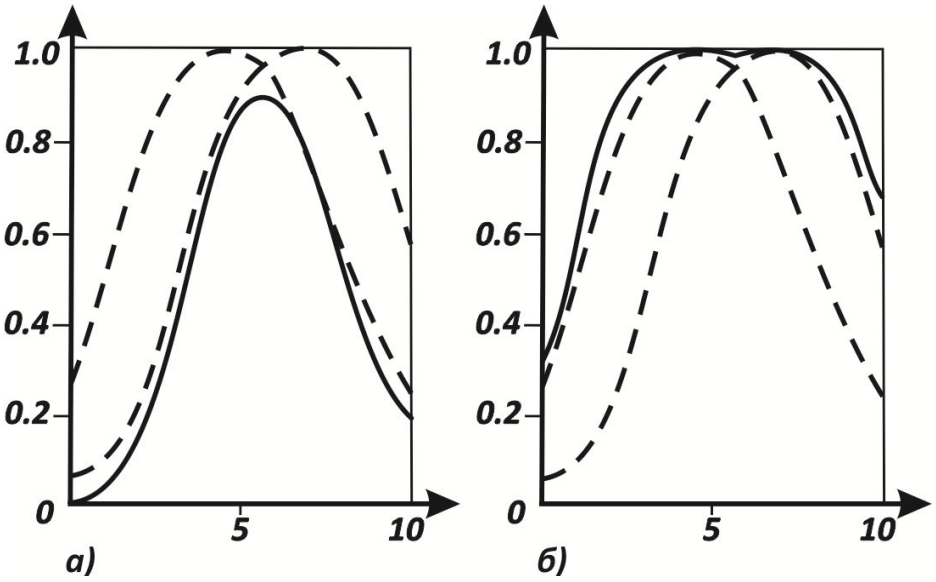


Рис. 77. Пересечение (*а*) и объединение (*б*) нечетких множеств (вероятностная интерпретация)

Пример 3.8. Программа использования вероятностных операторов конъюнкции и дизъюнкции.

$x = 0{:}\ 0{,}1{:}\ 10;$

$subplot(1, 2, 1);$

$y1 = gaussmf(x, [3\ \ 5]);\ \ y2 = gaussmf(x, [3\ \ 7]);$

$y3 = prod([y1;\ y2]);\ \ plot(x, [y1;\ y2], ':');$

$hold\ on;\ \ plot(x, y3);\ \ hold\ off;$

$subplot(1, 2, 2);$

$y4 = probor([y1;\ y2]);\ \ plot(x, [y1;\ y2], ';');$

$hold\ on;\ \ plot\ (x, y4);\ \ hold\ off;$

Дополнение нечеткого множества есть не что иное, как математическое представление вербального выражения «*НЕ А*» (рис. 78), где A — нечеткое множество, описывающее некоторое размытое суждение.

Описание функции дополнения: $y = 1 - y^*$, где y^* — исходная функции принадлежности.

Пример 3.9. Программа использования операции дополнения.

$\mathrm{x} = 0 : 0{,}1 : 10;\ \ \ y1 = gaussmf(x, [3\ \ 5]);$

$y = 1 - y1;\ \ plot\ (x, y1, ':')\ ;$

$hold\ on;\ \ plot\ (x, y);\ \ hold\ off;$

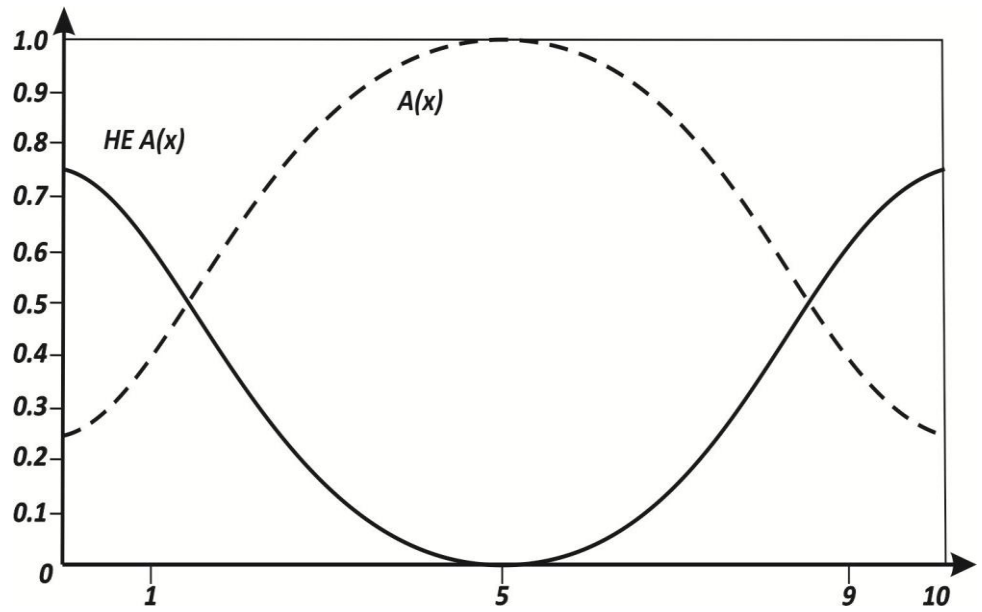


Рис. 78. Дополнение нечеткого множества

Отчет о выполнении лабораторной работы №5 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Текст программы с подробными комментариями.
2. Результаты моделирования (рис. 71 - 78).
3. Контрольный пример.

3.6. Лабораторная работа №6
«Моделирование нечеткой системы инструментами нечеткой логики»

Цель работы: изучить метод построения нечеткой системы инструментами нечеткой логики.

Краткие теоретические сведения.

В составе *Matlab* присутствуют пять основных средств графического интерфейса пользователя (ГИП), которые обеспечивают доступ к инструментами нечеткой логики: редакторы системы нечеткого вывода (СНВ), функции принадлежности, правил вывода, а также средства просмотра правил и поверхности вывода. Эти средства связаны между собой динамически и производимые изменения в одном из них влекут изменения в других.

Редактор СНВ предоставляет возможность формирования проектируемой системы на высоком уровне абстракции: количество входных и выходных переменных, наименование переменных.

Редактор функций принадлежности используется для определения формы функций принадлежности, ассоциированных с каждой переменной.

Редактор правил вывода применяется для редактирования списка правил, которые определяют поведение проектируемой системы.

Средство просмотра правил вывода используется в целях диагностики и может показывать, например, активность правил или форму влияния отдельных

функций принадлежности на результат нечеткого вывода.

Средство просмотра поверхности вывода используется для отображения зависимости выхода системы от одного или двух входов. Другими словами, оно генерирует и выводит карту поверхности вывода разработанной СНВ.

Построение нечетких систем по методу Мамдани. Для построения создаваемой системы в командной строке основного окна *Matlab* необходимо набрать команду $fuzzy$**.** Окно редактора новой СНВ содержит входную, обозначенную $input\ 1$ и выходную — $output\ 1$ переменные. По умолчанию инструмент нечеткой логики предлагает создавать СНВ типа Мамдани.

Для того чтобы добавить новую переменную, необходимо выбрать в меню $Edit$ соответствующий пункт (для входной переменной — $Add\ input$**,** для выходной — $Add\ output$**).** Изменение наименования переменной происходит по шагам.

Шаг 1. Отмечается переменная, которую необходимо переименовать.

Шаг 2. В поле редактирования изменяется наименование переменной по умолчанию на имя, предложенное пользователем.

Сохранение проектируемой системы в рабочее пространство среды *MATLAB* (в переменную) производится с помощью пункта меню $File \rightarrow Save\ to\ workspace\ as\ldots$ В этом случае данные сохраняются до окончания сеанса работы с *Matlab.* Для сохранения данных на диске после окончания сеанса работы применяется соответствующий пункт того же меню — $Save\ to\ disk\ as\ldots$

Редактор функций принадлежности. Следующим шагом в построении нечеткой модели средствами инструментов нечеткой логики является ассоциирование набора функций принадлежности с каждой входной и выходной переменной. Данная операция производится в редакторе функций принадлежности тремя способами, активизировать который можно:

- выбором в меню $View$ пункта $Edit\ Membership\ Functions\ldots$;
- двойным щелчком мыши на изображении соответствующей переменной (входной или выходной);
- набором в командной строке оператора $mfedit$.

С помощью редактора функций принадлежности можно отображать и редактировать любые функции принадлежности, ассоциированные (связанные) со всеми входными и выходными переменными разрабатываемой СНВ.

Связывание функции принадлежности с именем переменной происходит следующим образом:

- выбирается переменная по имени из набора графических объектов окна редактора функции принадлежности;
- указывается диапазон изменения значений для базовой и видимый диапазон для текущей переменных;
- в меню $Edit$ выбирается пункт $Add\ MFs\ldots$ В появившемся окне выбирают

вид функций принадлежности и их количество.

Редактируют функции принадлежности текущей переменной двумя способами: используя графическое окно функций принадлежности или изменяя характеристики функции принадлежности (наименование, тип и числовые параметры). При выборе необходимой функции принадлежности в графическом окне допускается плавное изменение кривой с помощью мыши.

Таким образом, при построении СНВ необходимо с помощью редактора функций принадлежности определить соответствующие функции для каждой из входных и выходных переменных.

Редактор правил вывода. После того как указано количество входных и выходных переменных, определены их наименования и построены соответствующие функции принадлежности, в СНВ необходимо включить правила вывода. Для этого в меню $View$ выбирается пункт $Edit\ Rules\ldots$ или в командной строке $Matlab$ набирается команда $ruleedit$.

Основываясь на описаниях входных и выходных переменных, определенных в редакторе функций принадлежности, редактор правил вывода формирует структуру правила автоматически. От пользователя требуется лишь связать значения входных и выходных переменных, выбирая из списка заданных ранее функций принадлежности и определить логические связки между ними. Также допускается использование логического отрицания («*HE*») и изменение весов правил в диапазоне от 0 до 1.

Правила вывода могут отображаться в окне в различных форматах, которые определяются путем выбора соответствующего пункта подменю $Format$ меню $Options$**.** По умолчанию используется расширенный формат отображения правил вывода ($verbose\ form$):

$$If\ (input_1\ is\ [not]\ mf\_lj_1)\ <and, or>\ldots$$

$$(input_i\ \ is\ [not\ ]\ mf_{ij})\ldots<and, or>$$

$$(input_n\ is\ [not]\ mf\_njn)\ then$$

$$(output_1\ is\ [not]\ mf\_n\ +\ 1j_n\ +\ 1<and, or>\ldots$$

$$(output_k\ is\ [not]\ mf\_k\ +\ nj_k\ +\ n)\ <and, or>\ldots$$

$$(output_m\ is[not]\ mf_m\ +\ nj_m\ +\ n)(w);$$

где i — номер входной переменной; ij — номер функции принадлежности i-й переменной; k — номер выходной переменной; n — количество входных переменных; m— количество выходных переменных; w — вес правила.

Круглые скобки заключают в себе обязательные параметры, квадратные — необязательные, а угловые — альтернативные параметры (один на выбор).

Кроме формата по умолчанию, существуют еще два вида форматов отображения правил: *символьный* ($symbolic\ form$) и *индексный* ($indexed\ form$).

Символьный формат имеет следующий вид:

$(input_1 < \sim =, ==> mf\_lj_1) \ < \&, | \ >\ldots$

$(input_i < \sim =, ==> mf\_ij_1)\ldots < \&, | >$

$(input_n < \sim =, ==> mf\_nj_n) \ =>$

$(output_1 < \sim =, ==> mf\_n \ + \ lj_n \ + 1) \ldots < \&, | \ >$

$(output_k < \sim = , ==> mf\_\_k \ + \ nj_k \ + \ n) < \&, | >\ldots$

$(output_m < \sim =, ==> mf\_m \ + \ nj_m \neq n)\ (w)$

Отличие символьного формата от расширенного состоит в том, что вместо словесной интерпретации связок используется символьная (символы «&» и «|» — соответственно определяют логическое «*И*» и логическое «*ИЛИ*», символ «~» — логическое отрицание, а символ «⇒» является разделителем условной и заключительной частей правила (антецендента и консеквента).

Здесь порядок следования чисел соответствует очередности вводимых переменных, причем символ «,» разделяет правило на условную и заключительную части. До двоеточия записывается порядковый номер соответствующей функции принадлежности, после двоеточия — вид логической связки («1» — логическое «*И*», «2» — логическое «*ИЛИ*»). Логическое отрицание задается символом «—».

После определения правил вывода в одноименном редакторе можно утверждать, что СНВ полностью создана.

Пример 3.10. Создание СНВ. Рассмотрим следующую ситуацию. Необходимо оценить степень инвестиционной привлекательности конкретного бизнес-проекта на основании данных о ставке дисконтирования и периоде окупаемости.

Шаг 1. Вызываем редактор для создания СНВ, набирая в командной строке $fuzzy$**.** Добавляем входную переменную посредством выбора в меню $Edit$ пункта $Add\ input$**.** В результате получаем следующую структуру СНВ: два входа, механизм нечеткого вывода по *методу Мамдани*, один выход. Объявляем первую переменную как $discont$, а вторую — $period$, которые соответственно будут представлять ставку дисконтирования и период окупаемости бизнес-проекта. Наименование выходной переменной, на основании которой принимается решение о степени инвестиционной привлекательности бизнес-проекта, задается как $rate$. Сохраним создаваемую модель под именем $Invest$. На рис. 79 представлено текущее состояние окна редактора СНВ.

Шаг 2. Каждой входной и выходной переменной поставим в соответствие набор функций принадлежности. Данная процедура реализуется в редакторе функций принадлежности. Для $discont$ определяем диапазон базовой переменной от 5 до 50 (единица измерения — проценты). Такой же диапазон выбираем для ее отображения. Добавим три функции принадлежности, тип которых — $trimf$ Последовательно выделяя мышью, отдельные функции

принадлежности, присвоим наименования — $small, middle, big$ соответственно небольшой, средней и большой ставке дисконтирования. Окно редактора функций принадлежности в текущем состоянии показано на рис.80. Переменной $period$ диапазон базовой переменной определен равным $[3, 36]$ (единица измерения — месяцы), поставлены в соответствие три функции принадлежности типа $gaussmf$ с наименованиями — $short, normal, long$.

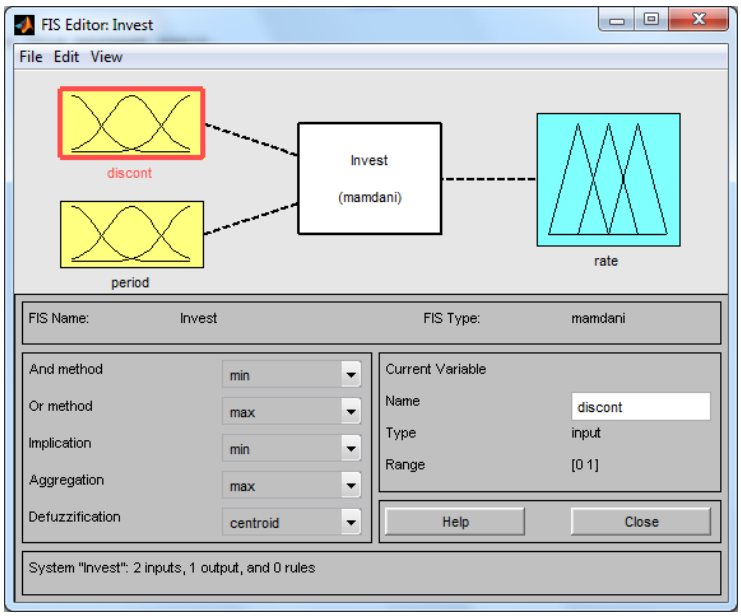


Рис. 79. Окно редактора системы нечеткого вывода

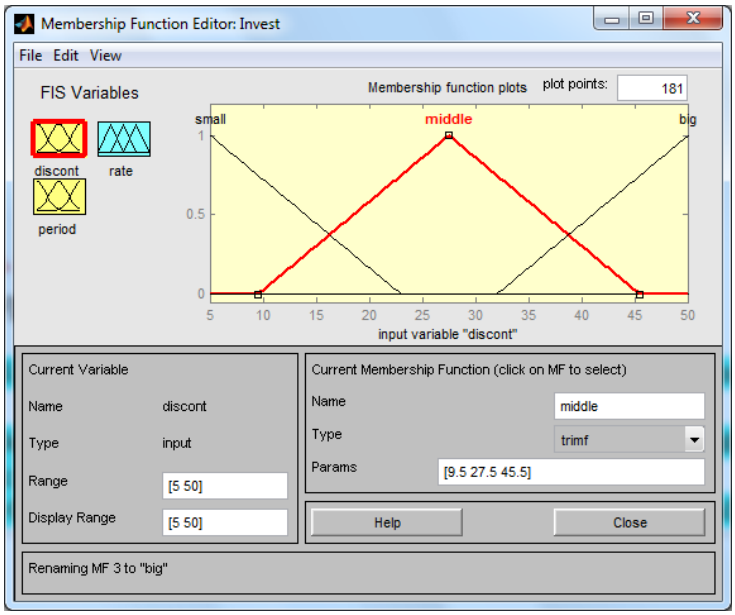


Рис. 80. Окно редактора функций принадлежности

Таким образом, переменная срока окупаемости бизнес-проекта будет принимать следующие значения: короткий, обычный и длительный срок окупаемости. Наконец, для переменной $rate$ определяем: базовая переменная изменяет значение в диапазоне $[0, 1]$, семантика описывается тремя функций

принадлежности типа ***trimf*** с наименованиями: $bad, normal, good.$

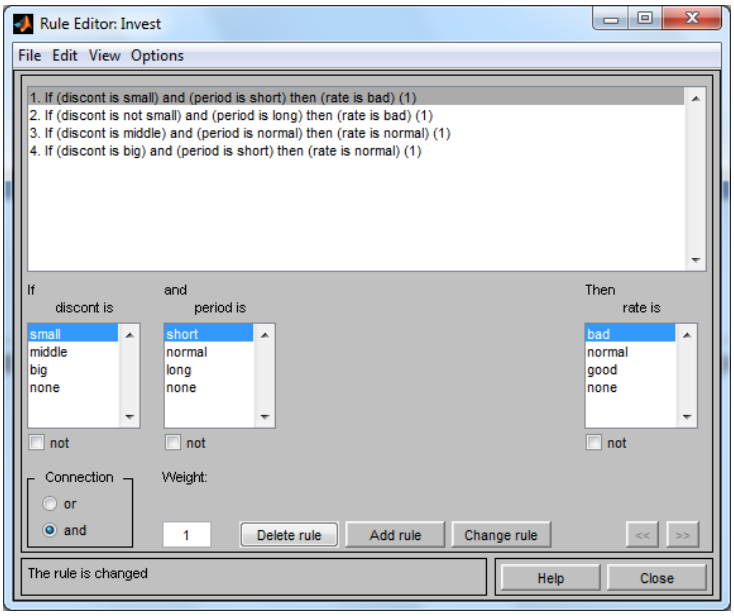


Рис. 81. Окно редактора правил вывода

Ш а г 3. Заключительным этапом построения СНВ является определение набора правил, которые задают связь входных переменных с выходными. Для этого в редакторе правил вывода определим:

ЕСЛИ $discont = small$ И $period = short$ ТО $rate = good$

ЕСЛИ $discont$ — $HE\ small$ И $period = long\ TO\ rate = bad$

ЕСЛИ $discont = middle$ И $period = normal\ TO\ rate = normal$

ЕСЛИ $discont$ — big И $period = short\ TO\ rate = normal$

Текущее состояние окна редактора правил вывода показано на рис. 81. В расширенном формате отображения указанные правила вывода представляются следующим образом:

$if\,(discont\ is\ small)\ and\ (period\ is\ short)\ then\ (rate\ is\ good)\ (1)$

$if\ (discont\ is\ not\ small)\ and\ (period\ is\ long)\ then\ (rate\ is\ bad)\ (1)$

$if\,(discont\ is\ middle)\ and\ (period\ is\ normal)\ then\ (rate\ is\ normal)\ (1)$

$if\,(discont\ is\ big)\ and\ (period\ is\ short)\ then\ (rate\ is\ normal)\ (1)$

При изменении формата на символьный правила вывода будут иметь вид:

$(discont\ ==\ small)\ \&\ (period\ ==\ short)\ =>\ (rate\ ==\ good)\ (1)$

$(discont\sim =\ small)\ \&\ (period\ —=\ long)\ =>\ (rate\ ==\ bad)\ (1)$

$(discont\ ==\ middle)\&\ (period\ ==\ normal)\ =>$
$=>\ (rate\ ==\ normal)(1)$

$(discont\ ==\ big)\ \&\ (period\ ==\ short)\ =>\ (rate\ ==\ normal)\ (1)$

Наконец, то же самое, но в индексном формате:

$$11{,}3(1) : 1 \; -13{,}1(1) : 1 \; 22{,}2(1) : 1 \; 31{,}2(1) : 1.$$

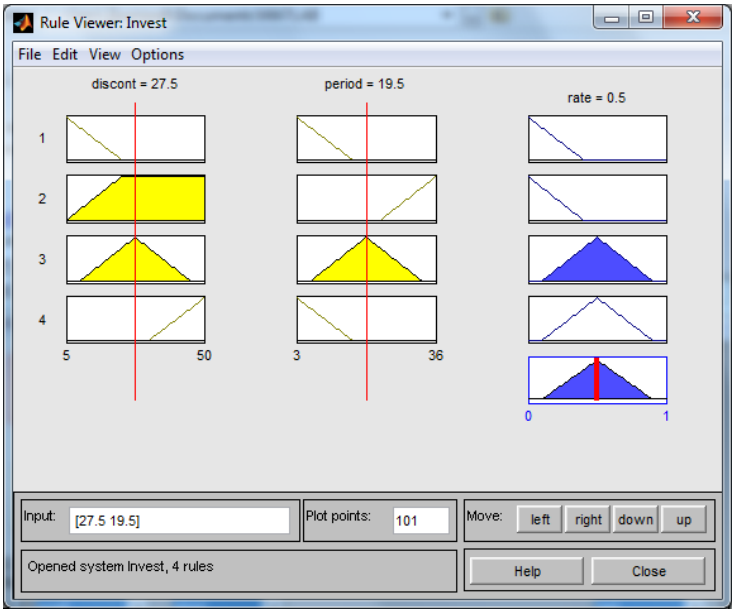


Рис. 82. Окно средства просмотра правил вывода

Средство просмотра правил вывода. Данное средство просмотра правил вывода позволяет отобразить процесс нечеткого вывода и получить результат. Главное окно средства просмотра состоит из нескольких графических окон, располагаемых по строкам и столбцам. Количество строк соответствует числу правил нечеткого вывода, а количество столбцов — числу входных и выходных переменных, заданных в разрабатываемой СНВ. Дополнительное графическое окно служит для отображения результата нечеткого вывода и операции дефазификации. В каждом окне отображается соответствующая функция принадлежности, уровень ее среза (для входных переменных) и вклад отдельной функции принадлежности в общий результат (для выходных переменных).

В нижней части главного окна можно отобразить номера правил вывода в различных форматах вывода, отмечая их мышью. Для изменения формата в меню *Options* выбирается пункт *Rule display format*.

Изменение значений входных переменных допустимо двумя способами:

- путем ввода в поле Input записи входного вектора, размерность которого равна количеству входных переменных;
- щелчком мыши в любом графическом окне, которое относится к входной переменной.

Входной вектор в каждом из этих вариантов определения исходных данных будет задавать набор красных вертикальных прямых.

Для СНВ, рассмотренной в примере 3.10, при входном векторе [15 10] (ставка дисконтирования — 15 %, период окупаемости бизнес-проекта — 10

месяцев) результат (степень инвестиционной привлекательности) будет составлять 0,585 (рис. 82).

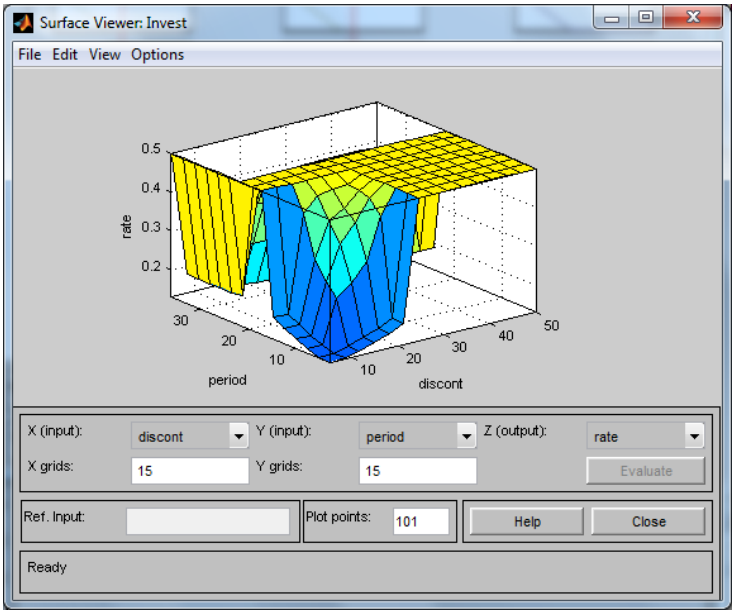


Рис. 83. Окно просмотра поверхности решений

Средство просмотра поверхности вывода. Средство просмотра поверхности вывода позволяет строить трехмерную поверхность как зависимость одной из выходных переменных от двух входных. Выбор входных и выходных переменных осуществляется посредством ниспадающих меню главного окна рассматриваемого программного средства. Количество выводимых линий по осям X и Y определяется в полях ввода $Xgrids, Ygrids$. Поверхность вывода, соответствующая правилам вывода примера 3.10 показана на рис. 83.

Построение нечетких систем методом Суджено. Рассмотрим построение СНВ двумя редакторами — СНВ и функций принадлежности. Для построения СНВ типа Суджено необходимо в меню ***File*** выбрать пункт ***New Sugeno FIS.*** Количество входных и выходных переменных определяется так же, как и при построении СНВ типа Мамдани.

Редактор функций принадлежности. Для СНВ типа Суджено изменения касаются только схемы определения функций принадлежности для выходных переменных. Инструмент нечеткой логики в среде $Matlab$ позволяет разрабатывать два вида нечетких моделей. Первая модель — это нечеткая модель Суджено нулевого порядка. Нечеткое правило вывода имеет следующий вид:

$$if\ x\ is\ A\ and\ \text{y}\ is\ B\ then\ z — k,$$

где A и B — нечеткие множества антецедента; k — четко заданная константа консеквента.

Для построения такой модели при добавлении функции принадлежности

необходимо выбрать тип — константа $(constant)$ и задать в качестве параметра функции принадлежности численное значение соответствующей константы. Вторая модель — нечеткая модель Суджено первого порядка. Для нее нечеткое правило вывода записывается следующим образом:

$$if\ x\ as\ A\ and\ \text{y}\ is\ B\ then\ z\ =\ p\ \bullet\ x\ +\ q\ \bullet\ y\ +\ r,$$

где $p,\ q$ и r — константы.

В данном случае тип функции принадлежности — линейная зависимость $(linear)$. Для определения параметров функции принадлежности необходимо ввести вектор, элементы которого соответствуют численным значениям констант консеквента.

Работа с редактором правил вывода, а также со средствами просмотра правил и поверхности вывода выполняется аналогично случаю построения СНВ по Мамдани.

Пример нечеткого вывода по Суджено с использованием нечеткой модели нулевого порядка и правил вывода, определенных выше, представлен на рис. 84 (выходная переменная имеет три значения: bad, $normal$, $good$, которые задаются соответственно тремя константами — $0,\ 0{,}5,\ 1$).

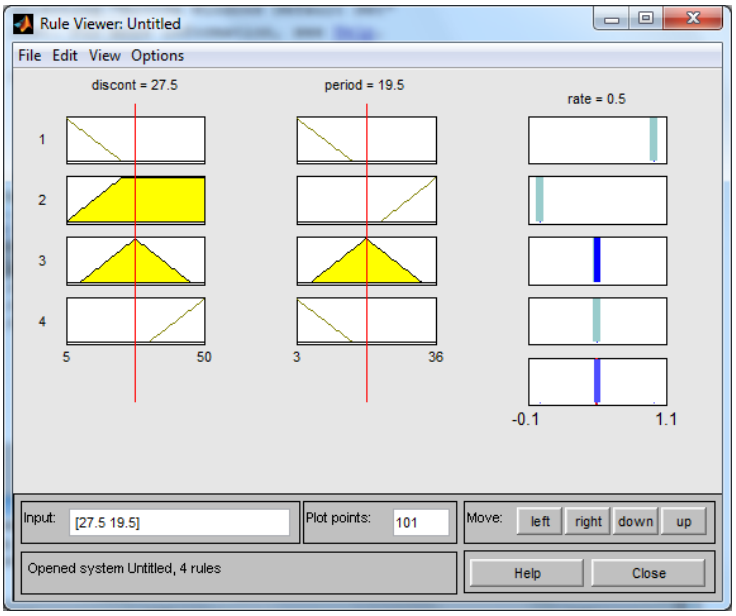


Рис. 84. Окно просмотра правил вывода (вывод по Суджено)

Отчет о выполнении лабораторной работы №6 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Текст программы с подробными комментариями.
2. Результаты моделирования (все полученные рисунки, с объяснением).
3. Контрольный пример.

3.7. Лабораторная работа №7
«Исследование алгоритма нечеткой кластеризации»

Цель работы: изучить алгоритм нечеткой кластеризации, получить практические навыки решения задач кластеризации методами нечеткой логики.

Краткие теоретические сведения.

Алгоритм нечеткой кластеризации называют FCM - алгоритмом (*Fuzzy Classifier Means, Fuzzy C-Means*). Целью FCM - алгоритма кластеризации является автоматическая классификация множества объектов, которые задаются векторами признаков в пространстве признаков. Другими словами, такой алгоритм определяет кластеры и соответственно классифицирует объекты. Кластеры представляются нечеткими множествами, и, кроме того, границы между кластерами также являются нечеткими.

FCM - алгоритм кластеризации предполагает, что объекты принадлежат всем кластерам с определенной функции принадлежности. Степень принадлежности определяется расстоянием от объекта до соответствующих кластерных центров. Данный алгоритм итерационно вычисляет центры кластеров и новые степени принадлежности объектов.

Для заданного множества K входных векторов x_k и N выделяемых кластеров c_j предполагается, что любой x_k принадлежит любому c_j с принадлежностью $\mu_{jk} \in [0,1]$, где j — номер кластера, а k — входного вектора. Принимаются во внимание следующие *условия нормирования* для μ_{jk}:

$$\sum_{j=1}^{N} \mu_{jk} = 1, \forall k = 1, \dots, K; \qquad 0 < \sum_{k=1}^{K} \mu_{jk} \leq \mathrm{K}, \forall j = 1, \dots, N.$$

Цель алгоритма — минимизация суммы всех взвешенных расстояний $\left|\left|x_k - c_j\right|\right|$:

$$\sum_{j=1}^{N} \sum_{k=1}^{K} (\mu_{jk})^q \left|\left|x_k - c_j\right|\right| \rightarrow min,$$

где q - фиксированный параметр, задаваемый перед итерациями.

Для достижения вышеуказанной цели необходимо решить следующую систему уравнений:

$$\partial / \partial \mu_{jk} \left(\sum_{j=1}^{N} \sum_{k=1}^{K} (\mu_{jk})^q \left|\left|x_k - c_j\right|\right|\right) = 0,$$

$$\partial / \partial c_j \left(\sum_{j=1}^{N} \sum_{k=1}^{K} (\mu_{jk})^q \left|\left|x_k - c_j\right|\right|\right) = 0.$$

Совместно с условиями нормирования (μ_{jk} данная система дифференциальных уравнений имеет следующее решение:

$$c_j = \frac{\sum_{j=1}^{N}(\mu_{jk})^q \cdot x_k}{\sum_{k=1}^{K}(\mu_{jk})^q}$$

(взвешенный центр гравитации) и

$$\mu_{jk} = \frac{1/\left|\left|x_k - c_j\right|\right|^{1/(q-1)}}{\sum_{j=1}^{N}(1/\left|\left|x_k - c_j\right|\right|^{\frac{1}{q-1}})}.$$

Алгоритм нечеткой кластеризации выполняется по шагам.

Ш а г 1. Инициализация. Выбираются следующие параметры:

- необходимое количество кластеров $N, [2 \le N \le K]$;
- мера расстояний, как Евклидово расстояние;
- фиксированный параметр q (обычно ~1,5);
- начальная (на нулевой итерации) матрица принадлежности $U^{(0)} = \left(\mu_{jk}\right)^{(0)}$;
- объектов x_k с учетом заданных начальных центров кластеров c_j.

Ш а г 2. Регулирование позиций центров кластеров. На $i-$ом итерационном шаге при известной матрице вычисляется $c_j^{(t)}$ в соответствии с вышеприведенным решением системы дифференциальных уравнений.

Ш а г 3. Корректировка значений принадлежности μ_{jk}. Учитывая известные $c_j^{(t)}$, вычисляются $\mu_{jk}^{(t)}$, если $x_k \neq c_j$, в противном случае:

$$\mu_{jk}^{(t+1)} = \begin{cases} 1, & l = j, \\ 0, & \text{иначе.} \end{cases}$$

Ш а г 4. Остановка алгоритма. Алгоритм нечеткой кластеризации останавливается при выполнении следующего условия:

$$\left|\left|U^{(t+1)} - U^{(t)}\right|\right| \le \varepsilon,$$

где $\|\ \|$ — матричная норма (например, Евклидова норма); ε — заранее задаваемый уровень точности.

Решение задач кластеризации. Существуют два способа решения задач кластеризации в $Matlab$: с использованием командной строки или графического интерфейса пользователя.

Рассмотрим **первый** из указанных способов.

Для нахождения центров кластеров в $Matlab$ имеется встроенная функция fcm, описание которой представлено ниже.

$$[center, U, obj\_fcn] = fcm(data, cluster\_n).$$

где $data$ — множество данных, подлежащих кластеризации, каждая строка описывает точку в многомерном пространстве характеристик; $cluster\_n$ — количество кластеров (более одного); $center$ — матрица центров кластеров, каждая строка которой содержит координаты центра отдельного кластера;

U — результирующая матрица функций принадлежности; obj_fcn — значение целевой функции на каждой итерации.

Пример 3.11. Программа нечеткой кластеризации.

- загрузка данных, подлежащих кластеризации, из файла:

$$load\ fcmdata.dat;$$

- определение центров кластеризации (два кластера):

$$[center, U, obj\_fcm] = fcm(fcmdata, 2);$$

- определение максимальной степени принадлежности отдельного элемента данных кластеру:

$$maxU = max\ (U);$$

- распределение строк матрицы данных между соответствующими кластерами:

$$index1 = find(U(1,:) == maxU);$$

$$index2 = find\ (U(2,:) == maxU);$$

- построение данных, соответствующих первому кластеру:

$$plot(fcmdata\ (index1, 1), fcmdata(index1, 2), 'ko', 'markersize', 5, 'LineWidth', 1);\ hold\ on;$$

- построение данных, соответствующих второму кластеру:

$$plot(fcmdata\ (index2, 1), fcmdata(index2,), 'kx', 'markersize', 5, 'LineWidth', 1);$$

- построение кластерных центров:

$$plot\ (center\ (1, 1)\ , center(1, 2), 'ko', 'markersize', 15, 'LineWidth', 2);$$

$$plot(center\ (2, 1), center\ (2, 2), 'kx', 'markersize', 15, 'LineWidth', 2);$$

На рисунке 85 представлено множество данных, подлежащих кластеризации и найденные центры кластеров для примера 3.11).

Функция fcm выполняется итерационно до тех пор, пока изменения целевой функции превышают некоторый заданный порог. На каждом шаге в командном окне $Matlab$ выводятся порядковый номер итерации и соответствующее текущее значение целевой функции.

Таблица 3.3

Целевая функция

Номер итерации	1	2	3	4	5	6	7	8	9	10	11	12
Целевая функция	8,94	7,31	6,90	5,41	4,08	3,83	3,81	3,80	3,79	3,79	3,79	3,78

Для оценки динамики изменения значений целевой функции используется команда построения графика $plot(obj\_fcm)$.

Результаты примера 3.11 показаны на рисунке 85.

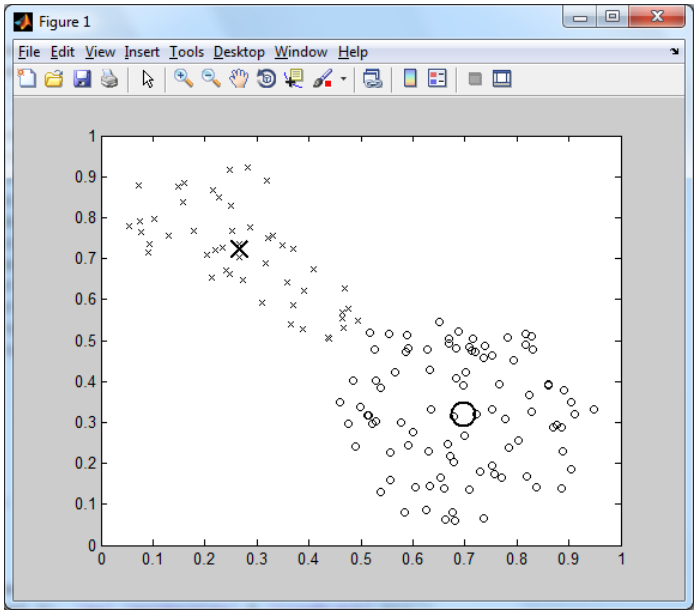


Рис. 85. Множество анализируемых данных и центры кластеров

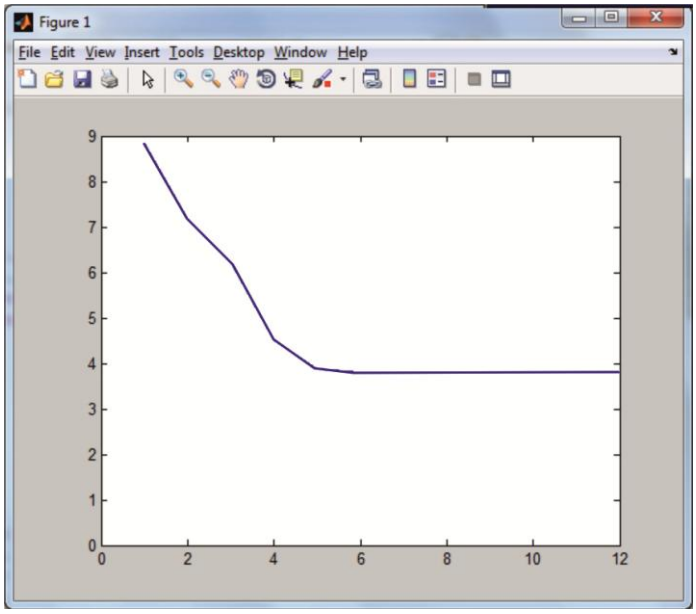


Рис. 86. График изменения значений целевой функции

Функцию кластеризации можно вызвать с дополнительным набором параметров: $fcm(data, cluster\_n, options)$. Дополнительные аргументы используются для управления процессом кластеризации:

- $options\ (1)$ — показатель степени для матрицы U (по умолчанию — 2.0);
- $options\ (2)$ — max количество итераций (по умолчанию — 100);
- $options\ (3)$ — предельное изменение значений целевой функции (по умолчанию — $1e-5$);
- $options\ (4)$ — информация на каждом шаге (по умолчанию — 1).

Пример определения функции fcm с дополнительными параметрами:

$$[center, U, obj\_fcm] = fcm(fcmdata, 2, [2, 100, 1\,e-5, 1]).$$

Второй способ решения задач кластеризации в $Matlab$ основан на использовании команды $findcluster$**.** Главное окно инструмента кластеризации показано на рисунке 87. Кнопка $< Load\ Data >$ используется для загрузки подлежащих кластеризации исходных данных следующего формата: каждая строка представляет собой точку в многомерном пространстве характеристик, количество строк соответствует количеству точек (элементов данных). Графическую интерпретацию исходных данных можно наблюдать в одноименном окне главного окна инструмента.

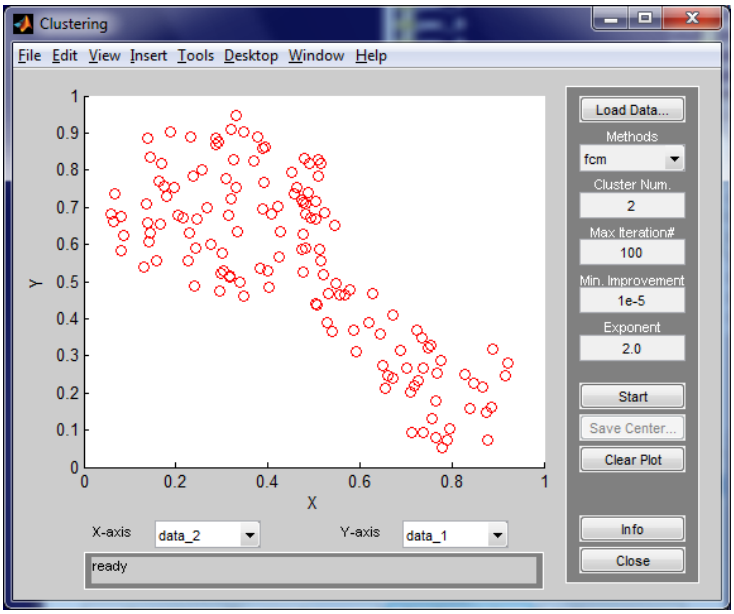


Рис. 87. Главное окно кластеризации в $MatLab$

Выбор типа алгоритма кластеризации осуществляется с использованием ниспадающего меню $Methods$ (пункт меню — fcm). Далее определяются параметры алгоритма кластеризации:

- количество кластеров (строка ввода — $Cluster\ Num$);
- максимальное количество итераций (строка ввода — $Max.Iteration$);
- минимальное значение улучшения целевой функции (строка ввода — $Min.Improvement$);
- показатель степени при матрице функций принадлежности (строка ввода — $Exponent$).

После определения необходимых значений указанных параметров осуществляется запуск алгоритма кластеризации с помощью кнопки $< Start >$. Количество произведенных итераций и значение целевой функции можно просмотреть в нижней части главного окна инструмента кластеризации.

Координаты найденных центров кластеров можно сохранить, щелкнув

мышью по кнопке $< Save\ Center\ldots >$. Каждая строка матрицы в файле представляет собой набор координат отдельного кластера. Количество строк соответствует количеству кластеров.

Отчет о выполнении лабораторной работы №7 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Текст программы с подробными комментариями.
2. Результаты моделирования (все полученные рисунки, с объяснением).
3. Контрольный пример.

3.8. Лабораторная работа №8 «Алгоритм нечеткой кластеризации *Mamdani*»

Цель работы: изучить алгоритм *Mamdani*, получить практические навыки решения задач кластеризации методами нечеткой логики.

Краткие теоретические сведения.

Одной из основной целью функционирования любой организации является получение прибыли. Для получения максимальной прибыли необходимо провести маркетинговые исследования рынка, т.е. выяснить спрос на предлагаемую продукцию, покупную способность населения и т.д.

При использовании практических задач нечеткого моделирования могут одновременно использоваться несколько алгоритмов нечеткого вывода с целью получения наиболее адекватных результатов. В предлагаемой модели нечеткого вывода используются алгоритм нечеткого вывода *Mamdani*.

Цель нашего моделирования заключается в определении ассортимента продукции, предлагаемого покупателям с целью получения максимальной прибыли. Прибыль получаемая от продаж завист от многих факторов. Основными факторами являются количество покупателей данного сектора рынка, покупная способность населения и себестоимость предлагаемой продукции. Для решения данной задачи при определении используемых факторов удобнее применить качественные показатели.

Исходя, из выше сказанного разработана система нечеткого вывода, которая имеет следующие входные параметры *«Себестоимость продукции»*, *«Количество покупателей»*, *«Покупная способность»*. Выходным параметром является *«Прибыль»*.

Перечисленные выше параметры не могут быть измерены привычным для нас способом, так как при их оценивании большую роль играет субъективный фактор (рынок): эксперт каким-либо образом оценивает значения этих параметров, используя их для принятия решений. *«Себестоимость продукции»* может быть оценена как *«Высокая»*, *«Средняя»*, *«Низкая»*. Входные параметры можно оценить и количественно, например такой параметр как *«Количество покупателей»* может быть оценен только приблизительно, хотя его и можно измерить точно, но только для

некоторого фиксированного момента времени (например, по результатам опроса населения).

Таким образом, при решении задачи определения оптимального ассортимента приходиться оперировать лишь приближенными значениями входных параметров задачи. Поэтому для решения подобной задачи логично обратиться к аппарату теории нечетких множеств и нечеткой логике.

Фаззификация входных и выходных лингвистических переменных.

В качестве терм-множества первой входной лингвистической переменной *«Себестоимость продукции»* будем использовать $T1 = \{$*«Низкая», «Средняя», «Высокая»*$\}$ (табл.3.4). *«Себестоимость продукции»* определена на интервале $[0..1]$ (абсолютная величина; чем число больше, тем себестоимость выше).

Таблица 3.4

Фаззификация лингвистической переменной *«Себестоимость продукции»*

Терм – множества лингвистической переменной	Значения параметров
«Низкая»	[0;0,2;0,4]
«Средняя»	[0,2;0,5;0,8]
«Высокая»	[0,6;0,8;1]

Ни рисунке 88 представлено формирование трех функций принадлежности для входной лингвистической переменной $input1$ – *«Себестоимость продукции»*: $mf1$ – «низкая», $mf2$ – «средняя», $mf3$ – «высокая».

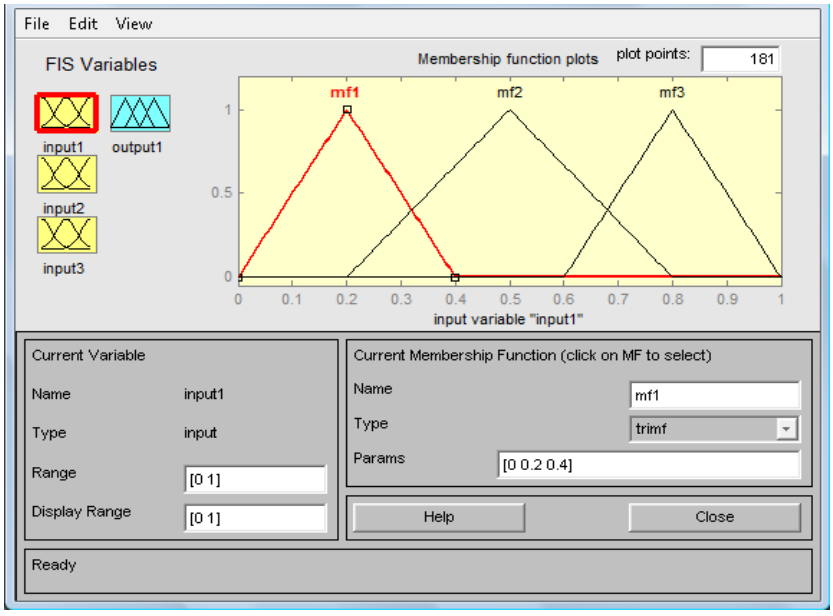


Рис. 88. Функции принадлежности входной лингвистической переменной *«Себестоимость продукции»*

В качестве терм-множества второй входной переменной *«Количество покупателей»* будем использовать (табл.3.5):

$T2 = \{$«Очень мало», «Мало», «Незначительно», «Достаточно», «Много», «Очень много»$\}$.

Лингвистическая переменная *«Количество покупателей»* определена на интервале $[0..10]$ (абсолютная величина; чем число выше, тем численность клиентов больше).

Таблица 3.5

Фаззификация лингвистической переменной *«Количество покупателей»*

Терм – множества лингвистической переменной	Значения параметров
«Очень мало»	[0;0,15;0,25]
«Мало»	[0,15;0,25;0,4]
«Незначительно»	[0,3;0,4;0,6]
«Достаточно»	[0,45;0,6;0,75]
«Много»	[0,65;0,75;0,9]
«Очень много»	[0,8;0,9;1]

Формирование шести функций принадлежности для входной лингвистической переменной $input2$ – *«количество покупателей»*: $mf1$ – очень мало, $mf2$ – мало, $mf3$ – незначительно, $mf4$ – достаточно, $mf5$ – много, $mf6$ – очень много представлено на рис. 89.

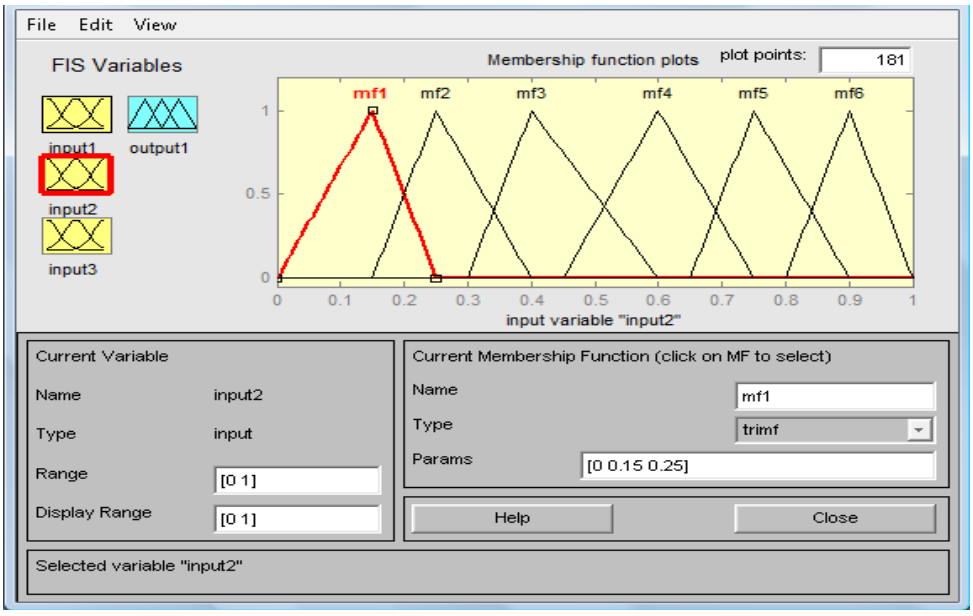


Рис. 89. Функции принадлежности входной лингвистической переменной *«Количество покупателей»*

В качестве терм-множества третьей входной переменной *«Покупная способность»* будем использовать $T3 = \{$«Очень бедные», «Бедные», «Средне обеспеченные», «Богатые», «Очень богатые»$\}$ с (табл.3.6).

Лингвистическая переменная *«Покупная способность»* определена на интервале $[0..10]$ (абсолютная величина; чем число выше, тем покупная

способность клиентов больше).

Таблица 3.6

Фаззификация лингвистической переменной *«Покупная способность»*

Терм – множества лингвистической переменной	Значения параметров
«Очень бедные»	[0;0,1;0,15]
«Бедные»	[0,1;0,35;0,5]
«Средне обеспеченные»	[0,2;0,5;0,85]
«Богатые»	[0,85;0,9;0,95]
«Очень богатые»	[0,95;0,97;1]

Формирование пяти функций принадлежности для входной лингвистической переменной $input3$ – *«Покупная способность»* (рис. 90):

$f1$ – очень бедные, $mf2$ – бедные, $mf3$ – средне обеспеченные, $mf4$ – богатые, $mf5$ – очень богатые.

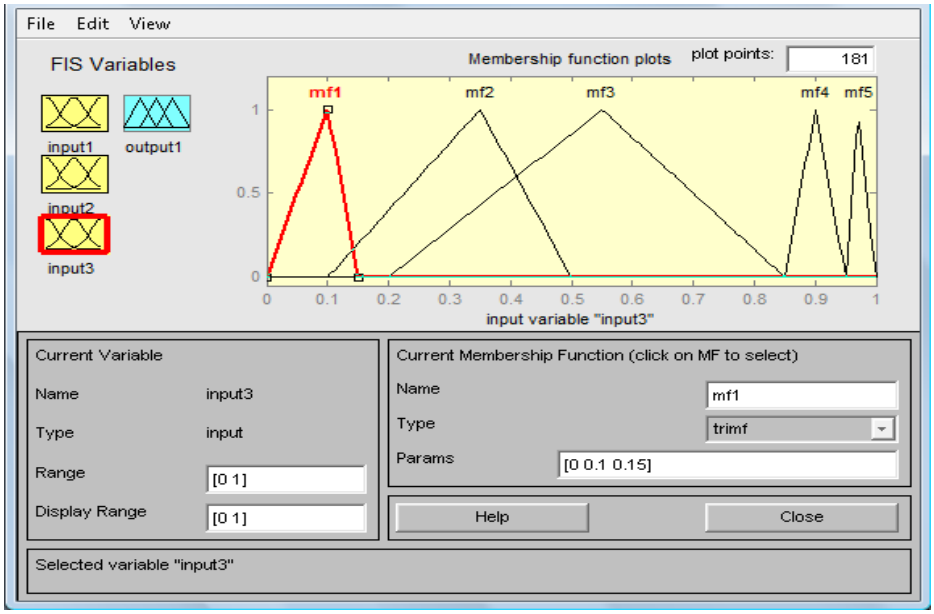


Рис. 90. Функции принадлежности лингвистической переменной *«Покупная способность»*

В качестве терм-множества выходной лингвистической переменной *«Прибыль»* будем использовать $T4 = \{«Малая», «Средняя», «Высокая»\}$ (табл.3.7). Лингвистическая переменная определена на интервале [0..10] (абсолютная величина; чем число больше, тем прибыль выше).

Таблица 3.7

Фаззификация лингвистической переменной *«Прибыль»*

Терм – множества лингвистической переменной	Значения параметров
«Малая»	[0;0,1;0,2]
«Средняя»	[0,15;0,4;0,7]
«Высокая»	[0,6;0,9;1]

Формирование трех функций принадлежности для выходной лингвистической переменной $output1$ – *«Прибыль»*: $mf1$ – малая, $mf2$ – средняя, $mf3$ – высокая представлено на рис. 91.

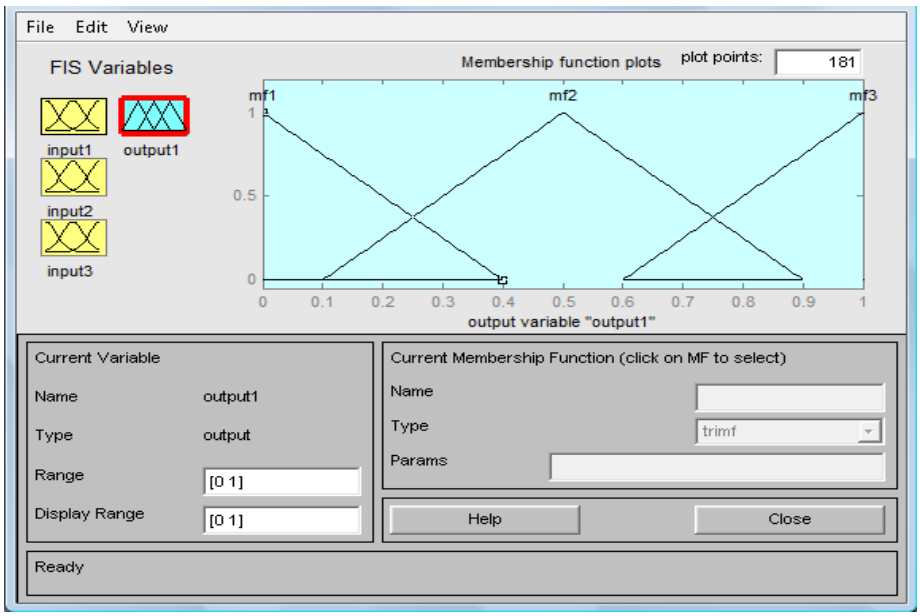


Рис. 91. Функции принадлежности лингвистической переменной *«Прибыль»*

Формирование трех входных лингвистических перменных и одной на выходе на основе алгоритма *Mamdani* представленно на рис. 92.

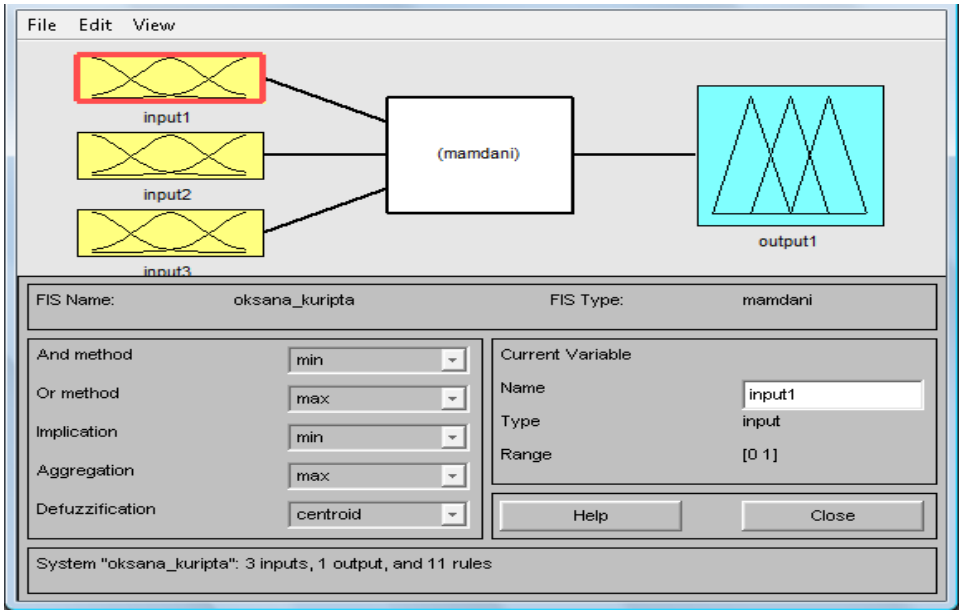


Рис. 92. Формирование лингвистических переменных на основе алгоритма *Mamdani*

Следующим этапом идет составление базы правил принятия решения системы нечеткого вывода планирования ассортимента продукции, часть которых приведена в табл.3.8 и их практическая реализация на рис. 93 – 103.

Таблица 3.8

База правил системы нечеткого вывода планирования ассортимента продукции

1. ЕСЛИ *«Себестоимость продукции» есть* «Низкая» **И** *«Количество покупателей» есть* «Очень мало» **И** *«Покупная способность» есть* «Очень бедные» **ТО** *«Прибыль» есть* «Малая»
2. ЕСЛИ *«Себестоимость продукции» есть* «Низкая» **И** *«Количество покупателей» есть* «Мало» **И** *«Покупная способность» есть* «Очень бедные» **ТО** *«Прибыль» есть* «Малая»
3. ЕСЛИ *«Себестоимость продукции» есть* «Низкая» **И** *«Количество покупателей» есть* «Незначительно» **И** *«Покупная способность» есть* «Очень бедные» **ТО** *«Прибыль» есть* «Малая»
4. ЕСЛИ *«Себестоимость продукции» есть* «Низкая» **И** *«Количество покупателей» есть* «Достаточно» **И** *«Покупная способность» есть* «Очень бедные» **ТО** *«Прибыль» есть* «Средняя»
5. ЕСЛИ *«Себестоимость продукции» есть* «Низкая» **И** *«Количество покупателей» есть* «Много» **И** *«Покупная способность» есть* «Бедные» **ТО** *«Прибыль» есть* «Средняя»
6. ЕСЛИ *«Себестоимость продукции» есть* «Низкая» **И** *«Количество покупателей» есть* «Достаточно» **И** *«Покупная способность» есть* «Средне обеспеченные» **ТО** *«Прибыль» есть* «Средняя»
7. ЕСЛИ *«Себестоимость продукции» есть* «Средняя» **И** *«Количество покупателей» есть* «Незначительно» **И** *«Покупная способность» есть* «Богатые» **ТО** *«Прибыль» есть* «Высокая»
8. ЕСЛИ *«Себестоимость продукции» есть* «Высокая» **И** *«Количество покупателей» есть* «Достаточно» **И** *«Покупная способность» есть* «Богатые» **ТО** *«Прибыль» есть* «Высокая»
9. ЕСЛИ *«Себестоимость продукции» есть* «Высокая» **И** *«Количество покупателей» есть* «Достаточно» **И** *«Покупная способность» есть* «Очень богатые» **ТО** *«Прибыль» есть* «Высокая»
10. ЕСЛИ *«Себестоимость продукции» есть* «Высокая» **И** *«Количество покупателей» есть* «Много» **И** *«Покупная способность» есть* «Очень богатые» **ТО** *«Прибыль» есть* «Высокая»
11. ЕСЛИ *«Себестоимость продукции» есть* «Высокая» **И** *«Количество покупателей» есть* «Очень много» **И** *«Покупная способность» есть* «Очень богатые» **ТО** *«Прибыль» есть* «Высокая»

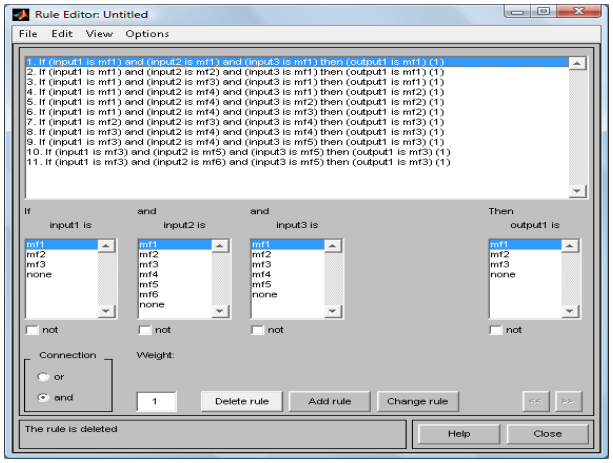


Рис. 93. Формирование первого правила принятия решения

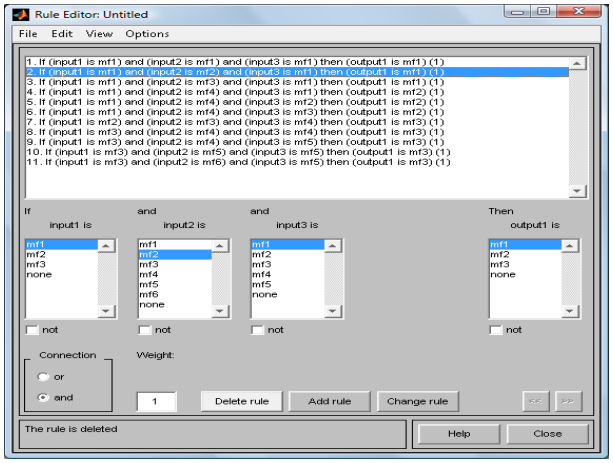


Рис. 94. Формирование второго правила принятия решения

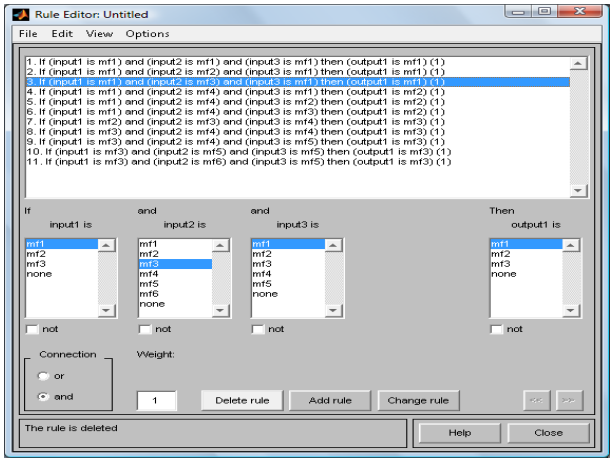


Рис. 95. Формирование третьего правила принятия решения

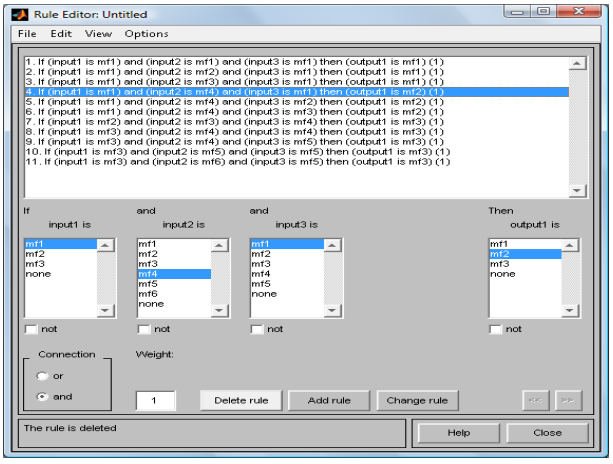


Рис. 96. Формирование четвертого правила принятия решения

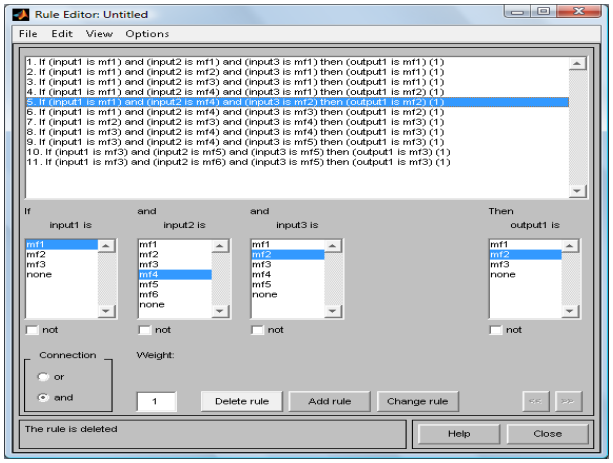


Рис. 97. Формирование пятого правила принятия решения

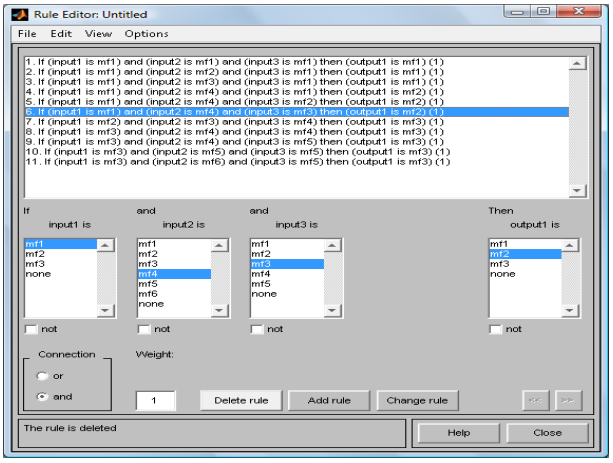


Рис. 98. Формирование второго правила принятия решения

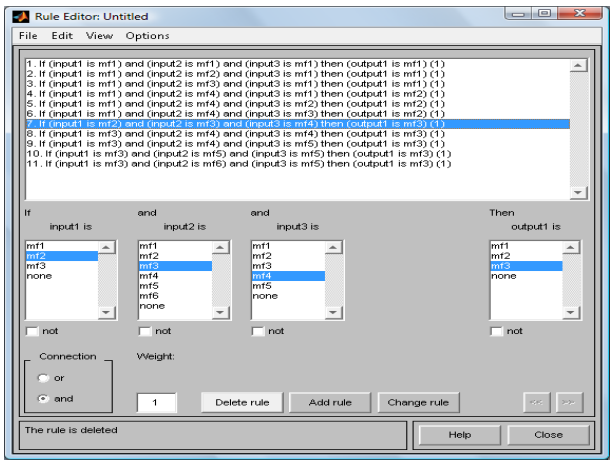


Рис. 99. Формирование седьмого правила принятия решения

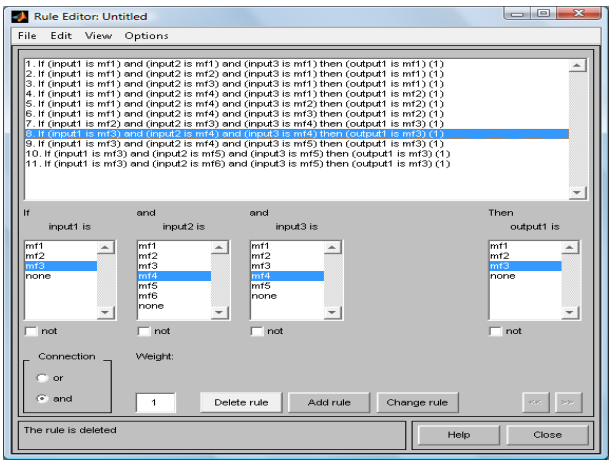


Рис. 100. Формирование восьмого правила принятия решения

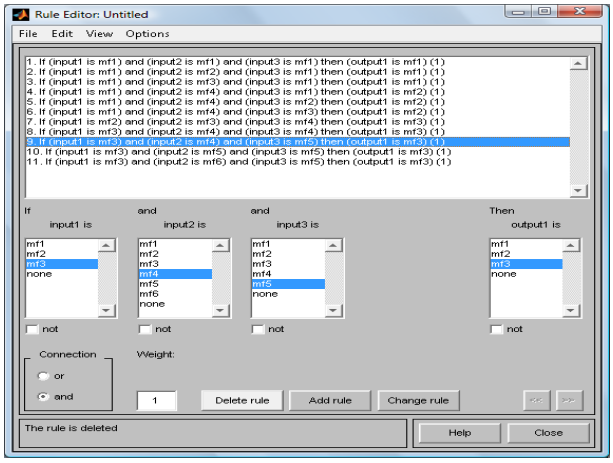


Рис. 101. Формирование девятого правила принятия решения

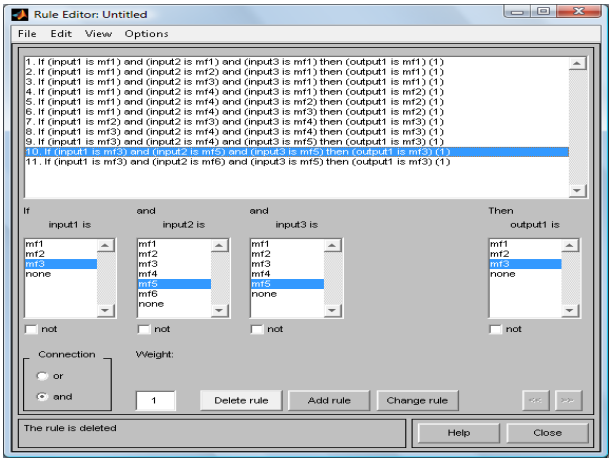


Рис. 102. Формирование десятого правила принятия решения

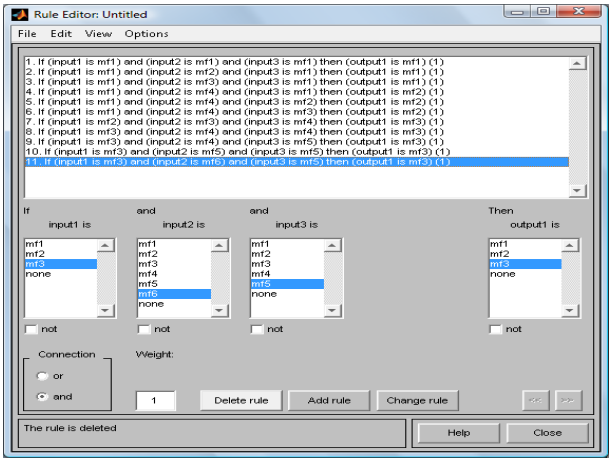


Рис. 103. Формирование одинадцатого правила принятия решения

На рис. 104 – 108 наглядно приведено отображение результатов функционирования выше приведенных правил принятия решения. На рис. 104 результат *«Прибыль»* отсутствует при оптимальном планировании, т.е.

ЕСЛИ *«Себестоимость продукции» есть* «Низкая»

И *«Количество покупателей» есть* «Незначительно»

И *«Покупная способность» есть* «Очень богатые»

ТО *«Прибыль» есть* «Низкая».

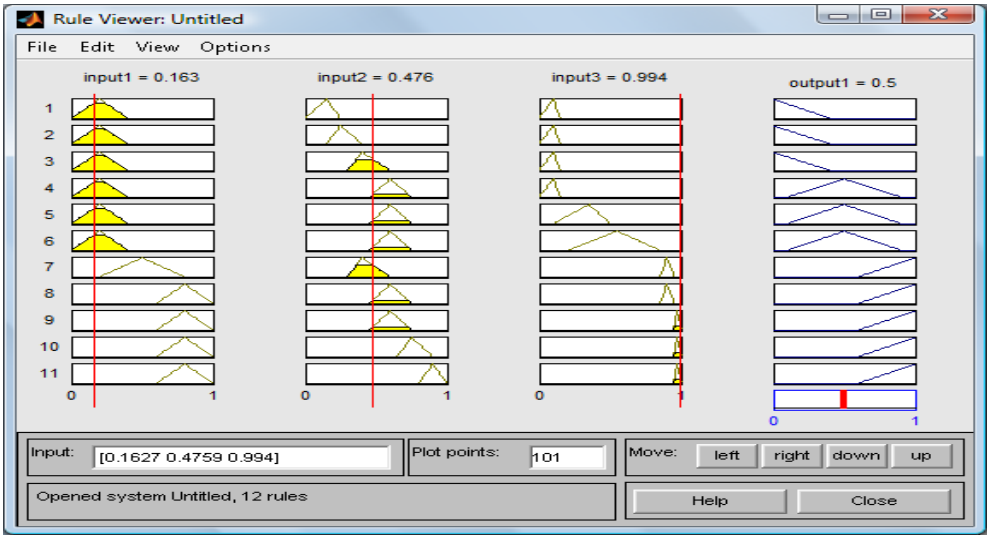


Рис. 104. Результат отсутствие прибыли

На рис. 105 результат получение прибыли незначителен при оптимальном планировании ассортимента продукции, т.е.

ЕСЛИ *«Себестоимость продукции» есть* «Низкая»

И *«Количество покупателей» есть* «Мало»

И *«Покупная способность» есть* «Очень бедные»

ТО *«Прибыль» есть* «Низкая».

Рис. 105. Результат незначительная прибыль – «очень мала»

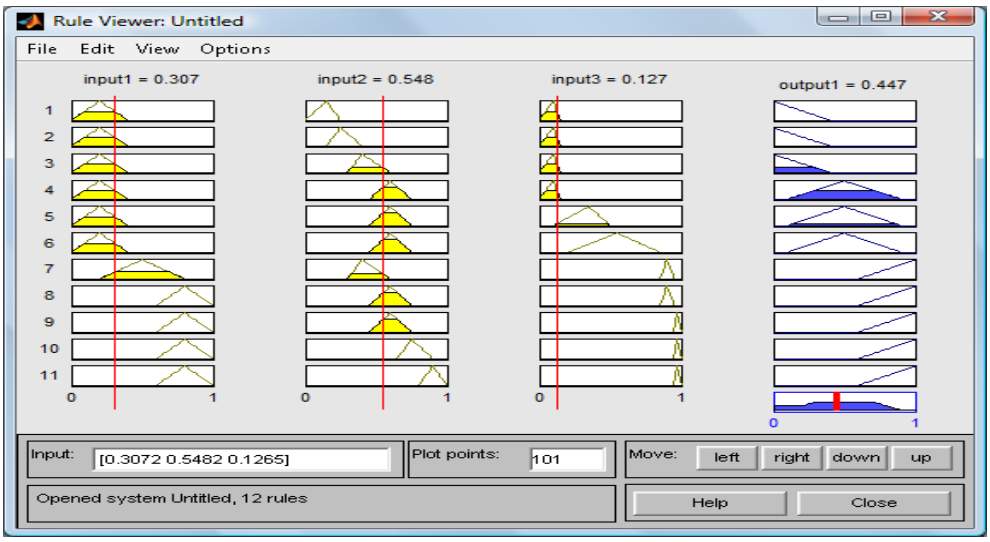


Рис. 106. Результат прибыли – «средний»

На рис. 106 результат получения средней прибыли при оптимальном планировании ассортимента продукции, т.е.

ЕСЛИ *«Себестоимость продукции» есть* «Средняя»

И *«Количество покупателей» есть* «Достаточно»

И *«Покупная способность» есть* «Бедные»

ТО *«Прибыль» есть* «Средняя».

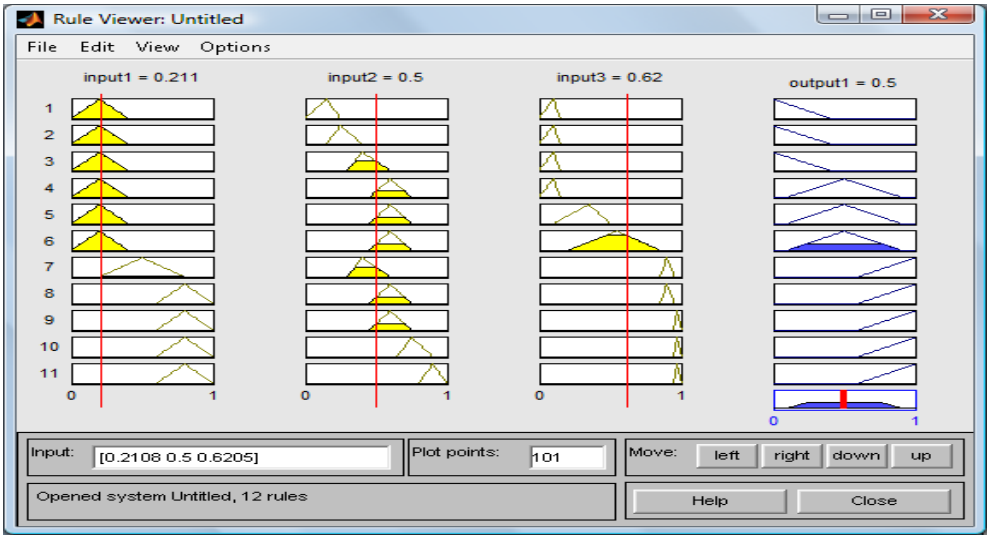


Рис. 107. Результат прибыли средней близкий к «высокому»

На рис. 107 результат получения прибыли средней близкий к высокому при оптимальном планировании ассортимента продукции, т.е.

ЕСЛИ *«Себестоимость продукции» есть* «Средняя»

И *«Количество покупателей» есть* «Достаточно»

И *«Покупная способность» есть* «Средне обеспеченные»

ТО *«Прибыль» есть* «Средняя».

На рис. 108 результат получения прибыли высокий при оптимальном планировании ассортимента продукции, т.е.

ЕСЛИ *«Себестоимость продукции» есть* «Высокая»

И *«Количество покупателей» есть* «Очень много»

И *«Покупная способность» есть* «Очень богатые»

ТО *«Прибыль» есть* «Высокая».

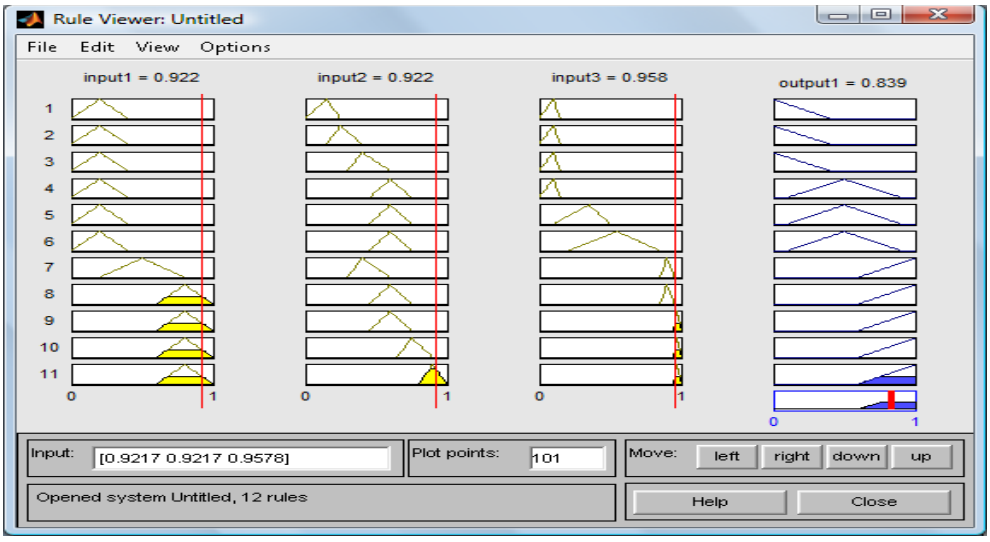


Рис. 108. Результат прибыли – «высокий»

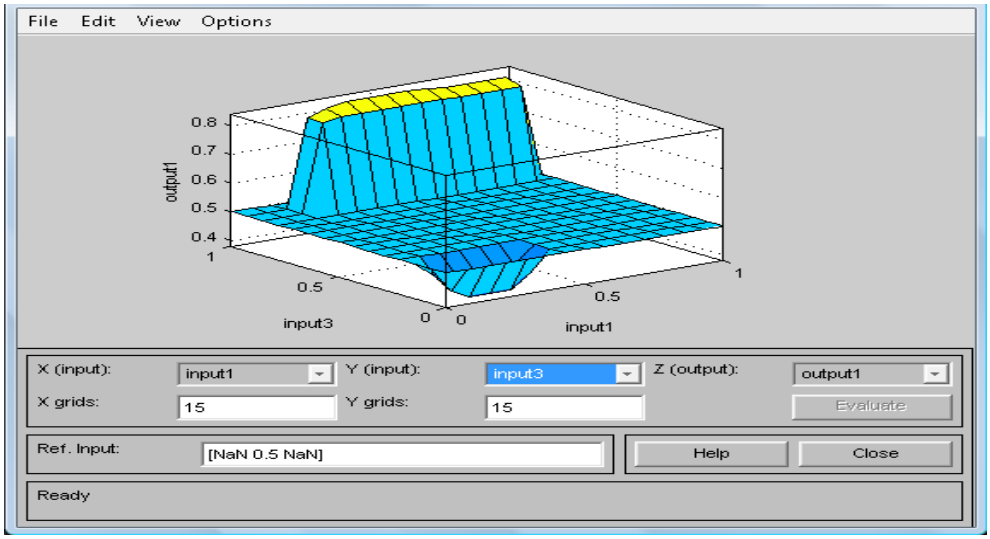


Рис. 109. Обобщенный вид получения прибыли в зависимоти от *«Себестоимости продукции»* и *«Покупной способности»*

На рис. 109 обобщенный вид получения прибыли при оптимальном планировании ассортимента продукции в зависимости от себестоимости продукции и покупной способности населения. Максимальное получение

прибыли наблюдается в двух случаях: если себестоимость низкая и покупная способность среднеобеспеченные; если себестоимость высокая и покупная способность очень богатые.

Таким образом, правила принятия решения дествуют следующим образом, прибыль организации можно получить за счет больших продаж товара с минимальной себестоимостью для мало и среднеобеспеченного населения при условии, что населения на данном секторе рынка имеется в достаточном количестве, либо при незначительных продажах товара с максимальной себестоимостью для очень богатых людей опять же при условии, что много населения в данном секторе рынка.

Отчет о выполнении лабораторной работы №8 должен быть выполнен на листах формата А4 и содержать следующие результаты:

1. Текст программы с подробными комментариями.
2. Результаты моделирования (все полученные рисунки, с объяснением).
3. Контрольный пример.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Айзерман М. А., Браверман Э. М., Розоноэр Л. И. Метод потенциальных функций в теории обучения машин. – М.: Наука, 1970. – 384 с.
2. Амамия М., Танака Ю. Архитектура ЭВМ и ИИ. — М.: Мир, 1993.
3. Андрейчиков А. В., Андрейчикова О. Н. Компьютерная поддержка изобретательства (методы, системы, примеры, применения). — М.: Машиностроение, 1998.
4. Беллман, Р. Математические методы в медицине / Р. Беллман. М. : Мир. 1987. 200 с.
5. Буров К. Обнаружение знаний в хранилищах данных // Открытые системы. - 1999. - № 5 - 6.
6. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. — СПб.: Питер, 2000.
7. Головко В.А. Нейроинтеллект: Теория и применение. Книга 1. Организация и обучение нейронных сетей с прямыми и обратными связями. – Брест: БПИ, 1999 – 264 с.
8. Головко В.А. Нейроинтеллект: Теория и применение. Книга 2. Самоорганизация, отказоустойчивость и применение нейронных сетей. – Брест: БПИ, 1999 – 228 с.
9. Горбань А. Н. Обучение нейронных сетей. – М. : ПараГраф, 1990.
10. Дебок Г. Анализ финансовых данных с помощью самоорганизующихся карт / Г. Дебок, Т. Кохонен. – М. : Альпина, 2001.
11. Дьяконов В.П., Круглов В.В. MATLAB 6.5 SP1/7/7 SP1/7 SP2 Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики. М.: СОЛОН-ПРЕСС, 2006. – 456с.
12. Ежов А.А. Нейрокомпьютинг и его применение в экономике и бизнесе / А.А. Ежов, С.А. Шумский. – М. : МИФИ, 1998. – 222 с.
13. Живоглядов В. П., Медведев А. В. Непараметрические алгоритмы адаптации. – Фрунзе: Илим, 1974. – 214 с.
14. Зайцевский И.В., Свиридов А.П., Слесарев Д.А. Нейронные сети и их приложения. М.: МЭИ, 2002. – 95с.
15. Зарипов Р. Х Машинный поиск вариантов при моделировании творческого процесса. — М.: Наука, 1983.
16. Искусственный интеллект: Справочник. В 3-х кн. Кн. 1. Системы общения и экспертные системы/Под ред. Э.В. Попова. — М.: Радио и связь, 1990.
17. Искусственный интеллект: Справочник. В 3-х кн. Кн. 2. Модели и методы/Под ред. Д.А. Поспелова. - М.: Радио и связь, 1990.
18. Искусственный интеллект: Справочник. В 3-х кн. Кн. 3. Программные и аппаратные средства/Под ред. В.А. Захарова, В.Ф. Хорошевского. — М.: Радио и связь, 1990.
19. Калан Р. Основные концепции нейронных сетей / Р. Калан. – М. :Вильямс, 2001. – 288 с.

20. Каллан Р. Основные концепции нейронных сетей. М.: Издательский дом «Вильямс», 2001. – 287с.
21. Колмогоров А.Н. Представление непрерывных функций многих переменных суперпозицией функций одной переменной и сложением // ДАН. – 1958. № 5. – 953-956 с.
22. Круглов В.В., Борисов В.В. Искусственные нейронные сети. М.: Горячая линия - Телеком, 2001. – 382с.
23. Ларичев О. И. Системы, основанные на экспертных знаниях: история, современное состояние и некоторые перспективы: Сб. науч. тр. Седьмой национальной конференции по искусственному интеллекту с международным участием. — М.: Изд-во физико-математической литературы, 2000.
24. Льюнг Л. Идентификация систем. Теория для пользователя. – М.:Наука,1991. – 432 с.
25. Масалович А. И. От нейрона к компьютеру // Журнал доктора Добба. - 1992. - № 1.
26. Медведев А. В. Адаптация в условиях непараметрической неопределенности // Адаптивные системы и их приложения. – Новосибирск: Наука, 1978. – С. 4-34.
27. Медведев В.С., Потемкин В.Г. Нейронные сети. Матлаб 6. М.: Диалог МИФИ, 2002. – 496с.
28. Минский М. Л. Персептроны / М. Л. Минский, С. Пейперт. – М. : Мир, 1971. – 360 с.
29. Минский М., Пейперт С. Персептроны: Пер. с англ. / Под ред. В.А. Ковалевского. - М.: Мир, 1971 - 261 с.
30. Надарая Э. А. О непараметрических оценках плотности вероятности и регрессии // Теория вероятностей и ее применение. – 1965. – 10. – № 1. – С. 199-203.
31. Нейронные сети. STATISTICA Neural Networks. М.: Горячая линия - Телеком, 2001. – 182с.
32. Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002 – 344 с.
33. Осовский С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2002. – 344с.
34. Прикладная статистика: Классификация и снижение размерности: Справ. Изд. /Айвазян С.А., Бухштабер В.М., Енюков И.С., Мешалкин Л.Д.; Под ред. Айвазяна С.А. – М.: Финансы и статистика. – 1989. – 607 с.
35. Применение нейронных сетей в экспериментальной физике" И.В. Кисель, В.Н. Нескромный, Г.А. Ососков Физика элементарных частиц и атомного ядра, 1993, Том 24, Выпуск 6; ОИЯИ УДК 681.322
36. Раудис Ш. Ю. Оптимизация непараметрического алгоритма классификации // Адаптивные системы и их приложения. – Новосибирск: Наука, 1978. – С. 57-61.

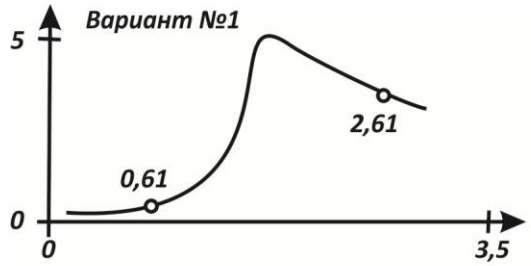
37. Розенблатт Ф. Принципы нейродинамики / Ф. Розенблатт. – М. :Мир, 1965. – 387 с.
38. Розенблатт Ф. Принципы нейродинамики. - М.: Мир, 1965. - 480 с.
39. Соколов Е.Н., Шмелев Л.А. Нейробионика. - М.: Наука, 1983.
40. Статические и динамические экспертные системы: Учеб. пособие / Э. В. Попов, И. Б. Фоминых, Е. Б. Кисель, М. Д. Шапот. — М.: Финансы и статистика, 1996.
41. Тарков М.С. Нейрокомпьютерные системы. М.: БИНОМ, 2006. –142с.
42. Тельное Ю. Ф. Интеллектуальные информационные системы в экономике: Учеб. пособие. — М.: СИНТЕГ, 1998.
43. Уоссермен Ф. Нейрокомпьютерная техника : теория и практика /Ф. Уоссермен. – М. : Мир, 1992. – 380 с.
44. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. – М.: Мир, 1992.
45. Финн В. К. Правдоподобные рассуждения в интеллектуальных системах типа ДСМ // Итоги науки и техники. Сер. «Информатика». Т. 15. «Интеллектуальные информационные системы». — М.: ВИНИТИ, 1991.
46. Цуриков В. М. Проект «Изобретающая машина» — интеллектуальная среда поддержки инженерной деятельности // Журнал ТРИЗ. - 1991.-№21.
47. Цыпкин Я. З. Основы теории обучающихся систем. – М.: Наука, 1970. – 252 с.
48. ЭлтиДж., Кумбс М. Экспертные системы: концепции и примеры: Пер. с англ. — М.: Финансы и статистика, 1987.
49. Bouchet C, Brunet C, Anjewierden A. SHELLY: An integrated workbench for KBS development // Proc. of 9th Int. Workshop Expert Syst. and their Appl. - France, Avignon, 1989. - No. 1.
50. Cichocki A., Unbehauen R. Neural Networks for Optimization and Signal Processing. – Stuttgart: Teubner, 1993. – 526 p.
51. Cover T. M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition // IEEE Trans. on Electronic Computers. – 1965. – 14. – P. 326-334.
52. Craddock R. J., Warwick K. Multi-layer radial basis function networks. An extension to the radial basis function // Proc. Int. Conf. on Neural Networks ICNN’96, Washington, DC, June 3-6, 1996. – V.2 – P. 700-704.
53. Durkin J. Expert Systems: a view of the field // IEEE Expert. - 1996. - No. 2.
54. Floreen P. The convergence of Hamming memory networks // IEEE Trans. – Neural Networks, 1991. – Vol. 2. – pp. 449-457.
55. Fukushima K., Miyake S. 1982. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern recognition 15(6): 455–69.
56. Fukushima К. 1975. Cognitron: A self–organizing mult ilayered neural network. Biological Cybernetics 20:121–36.

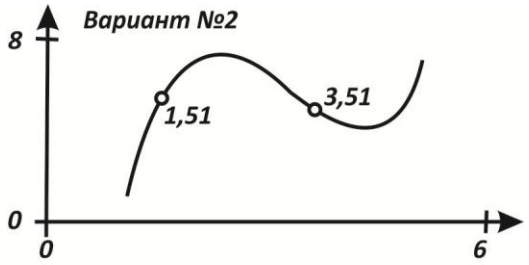
57. Fukushima K. 1980. Neocognitron: A self–organizing neural network model for a mechanism of pattern recognition uneffected by shift in position. Biological Cybernetics 36(4):193–202.
58. Fukushima K. 1987. A neural network for selective attention. In Proceedings of the IEEE First International Conference on Neural Networks, eds. M. Caudil and C. Butler, vol. 2, pp. 11–18. San Diego, CA:SOS Printing.
59. Fukushima K., Miyake S., Takayuki I. 1983. Neocog–nitron: A neural network model for a mechanism of visual pattern recognition. IEEE Transaction on Systems, Man and Cybernetics SMC–13(5):826–34.
60. Hartman E. J., Keeler J. D., Kowalski J. Layered neural networks with Gaussian hidden units as universal approximations // Neural Computation. – 1990. –2. – P. 210-215.
61. Haykin S. Neural Networks. A Comprehensive Foundation. – Upper Saddle River, N.J.: Prentice Hall, Inc., 1999. – 842 p.
62. Hinton G., McClelland J. Learning Representation by Recirculation // Proceedings of IEEE Conference on Neural Information Processing Systems. – 1989.
63. Hubel D. H., Wiesel T. N. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology 160:106–54.
64. Hubel D. H., Wiesel T. N. 1965. Reseptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cats. Journal of Neurophi–siology 28:229–89.
65. Hubel D. H., Wiesel. T. N. 1977. Functional architecture of macaque monkey visual cortex. Proceedings of the Royal Society, London. Ser. B 198, pp. 1–59.
66. Kosko B. Bidirectional associative memories // IEEE Trans. Systems, Man and Cybernetics, 1988. – Vol. 18. – pp. 49-60.
67. Kroese B. An introduction to Neural Networks. – Amsterdam: University of Amsterdam. – 1996. – 120 p.
68. McCulloch W. W., Pitts W. 1943. A logical calculus of the ideas imminent in nervous activiti. Bulletin of Mathematical Biophysics 5:115-33. (Русский перевод: Маккаллок У. С., Питтс У. Логическое исчисление идей, относящихся к нервной деятельности. Автоматы. - М.: ИЛ. - 1956.
69. Minsky M. L, Papert S. 1969. Perseptrons. Cambridge, MA: MIT Press. (Русский перевод: Минский М. Л., Пейперт С. Персептроны. - М: Мир. - 1971.)
70. Moody J. Darken C. J. Fast learning in networks of locally-tuned processing units// Neural Computation. – 1989. – 1. – P. 281-294.
71. Motta E., Rajan T., DominigueJ., Eisenstadt M. Methodological foundations of KEATS, the knowledge Engineer's Assistant // Knowledge Acquisition. - 1991. - No. 3.
72. Nelles O., Ernst S., Isermann R. Neuronale Netze zur Identification nichtlinearer, dynamischer Systeme: Ein Ueberblick //

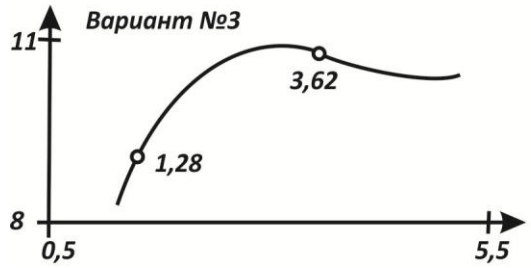
Automatisierungstechnik. – 1997. – 45. – № 6. – S. 251-262.
73. Park J., Sandberg I. W. Universal approximation using radial-basis-function networks // Neural Computation. – 1991. – 3. – P. 246-257
74. Parzen E. On the estimation of a probability density function and the mode // Ann. Math. Statist. – 1962. – 38. – P. 1065-1076.
75. Pitts W. Moculloch W. W. 1947. How we know universals. Bulletin of Mathematical Biophysics 9:127-47.
76. R. 0. Winder, Single-stage logic. Paper presented at the AIEE Fall General Meeting, 1960.
77. Rosenblatt F. 1962. Principles of Neurodinamics. New York: Spartan Books. (Русский перевод: Розенблатт Ф. Принципы нейродинамики. - М: Мир. - 1965.)
78. Schalkoff R.J. Artificial Neural Networks. – N.Y.: The McGraw–Hill Comp., Inc., 1997. – 422 p.
79. VITAL: A methodology - based workbench for KBS life cycle Support // ESPRIT - II Project 5365, 1990.
80. Wang Y. F., Cruz J., Mulligan J. Two coding strategies for bidirectional associative memory // IEEE Trans. Neural Networks, 1990. – Vol. 1. – pp. 81-92.
81. Widrow B. 1961. The speed of adaptation in adaptive control system, 1933. American Rocket Society Guidance Control and Navigation Conference.
82. Widrow B. 1963. A statistical theory of adaptation. Adaptive control systems. New York: Pergamon Press.
83. Widrow B., Angell J. B. 1962. Reliable, trainable networks for computing and control. Aerospace Engineering 21:78-123.
84. Widrow B., Hoff M. E. 1960. Adaptive switching circuits. 1960 IRE WESCON Convention Record, part 4, pp. 96-104. New York: Institute of Radio Engineers.
85. Wilamowski B. M., Jaeger R. C. Implementation of RBF type networks by MLP networks // Proc. Int. Conf. on Neural Networks ICNN'96, Washington, DC, June 3-6, 1996. – V.3 – P. 1670-1675.

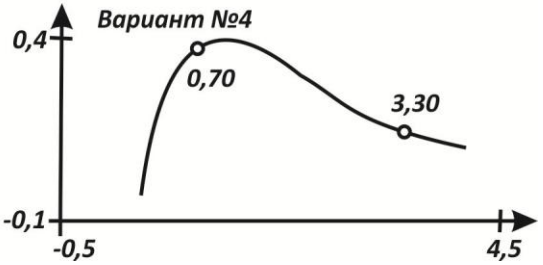
ПРИЛОЖЕНИЕ

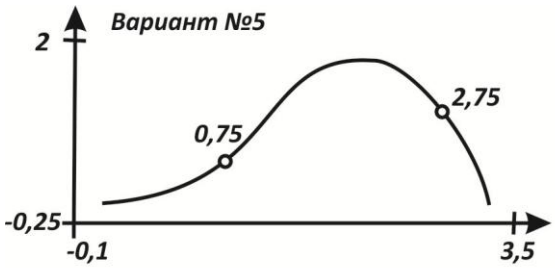
Контрольные задания для лабораторной работы №1

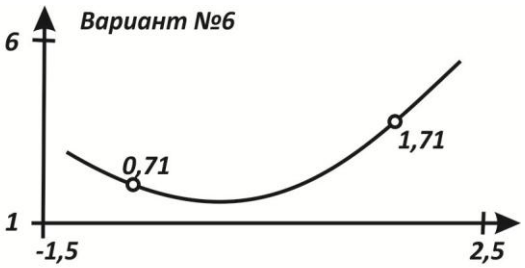


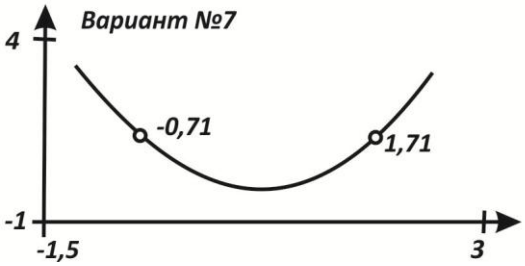


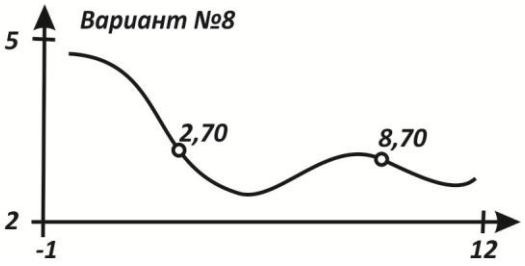


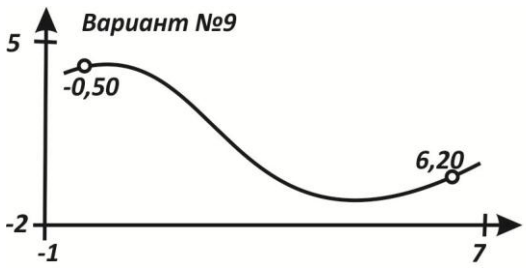


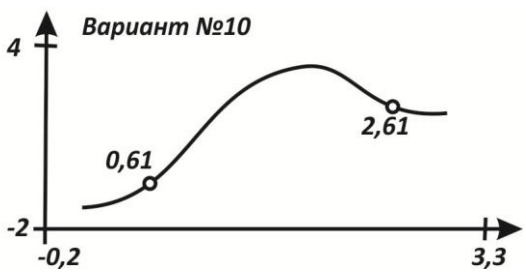












Контрольные задания для лабораторной работы №2

Вариант 1

$$z = cos(x)cos(y), x, y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

Вариант 2

$$z = \frac{sin(x)}{x} \cdot \frac{sin(y)}{y}, x, y \in [-1,1]$$

Вариант 3

$$z = x^2 \cdot y^2, x, y \in [-1.5; 1.5]$$

Вариант 4

$$z = x^3 \cdot sin(y) + 1,$$
$$x \in [-1,1], y \in [-1,5; 1,5]$$

Вариант 5

$$z = x^2 \cdot sin(y) - 1,$$
$$x \in [-1,1], y \in [-1,5; 1,5]$$

Вариант 6

$$z = e^3 + 3y, x \in [0; 1], y \in [-2; 1]$$

Вариант 7

$$z = \sqrt{y^2 + x^3}, x \in [-1; 1], y \in [-2; 2]$$

Вариант 8

$$z = \sqrt{y^3 + x^2}, x \in [-1; 1], y \in [-2; 2]$$

Вариант 9

$$z = \frac{1}{\sqrt{y^2 + x^3}}, x, y \in [-1,1]$$

Вариант 10

$$z = \frac{1}{\sqrt{y^3 + x^2}}, x, y \in [-1,1]$$

Вариант 11

$$z = x^2 \cdot cos(y) + 1, x \in [-1,1],$$
$$y \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$$

Вариант 12

$$z = x^3 \cdot cos(y) + 1, x \in [-0,5; 0,5],$$
$$y \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$$

Вариант 13

$$z = \sqrt{y^2 + x^4}, x, y \in [-1; 1]$$

Вариант 14

$$z = sin\left(\frac{y}{(1 + x^2)}\right) + 1, x \in [-1,1],$$
$$y \in [-\pi, \pi]$$

Вариант 15

$$z = sin(x^2) + y^2, x \in [-1,77; 1,77],$$
$$y \in [-0,3; 0,3]$$

Вариант 16

$$z = cos(x^2) + xy, x \in [-0,886; 0,886],$$
$$y \in [-0,3; 0,3]$$

Вариант 17

$$z = (x - 1)^2 \cdot exp(y^2), x \in [0,5; 2],$$
$$y \in [-1,1]$$

Вариант 18

$$z = \frac{cos(x)}{x^2 + 1}, x \in [-5; 5], y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

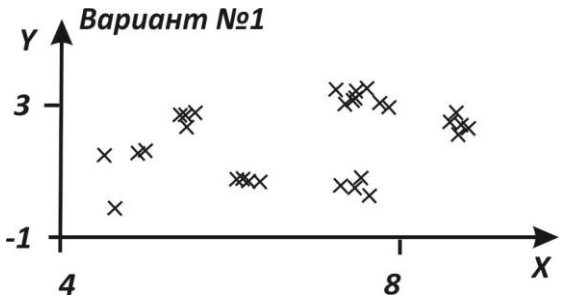
Вариант 19

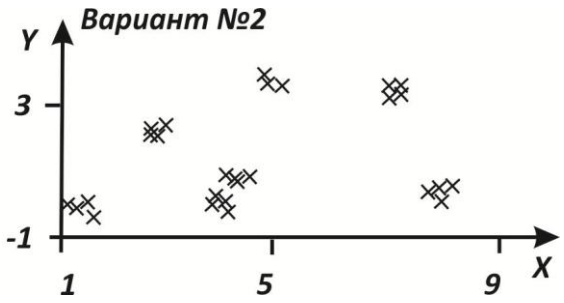
$$z = sin(x) \cdot sin(y), x, y \in [-\pi; 0]$$

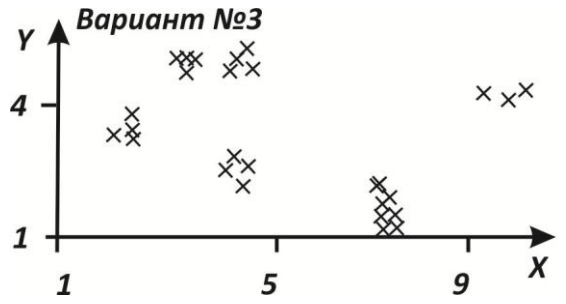
Вариант 20

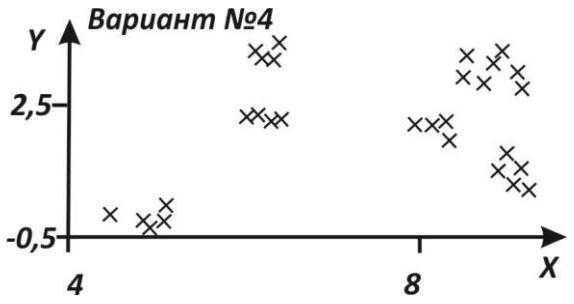
$$z = \ln(x) \cdot \ln(y), x, y \in [0,5; 2]$$

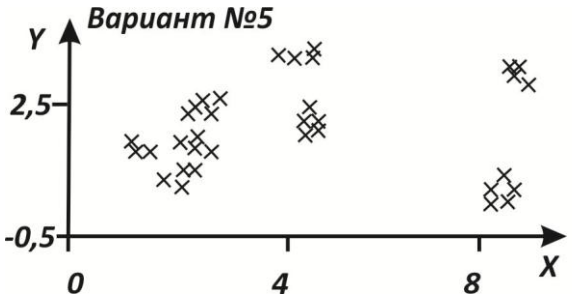
Контрольные задания для лабораторной работы №3

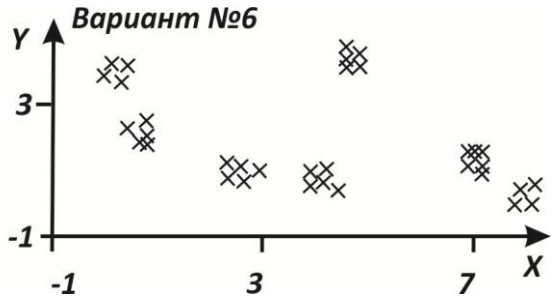


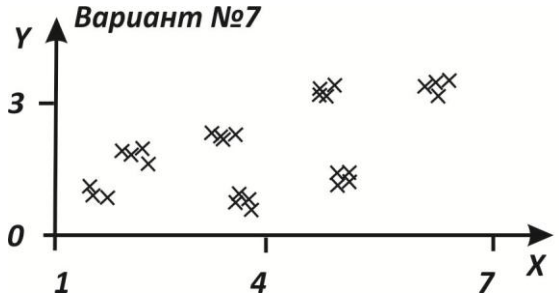


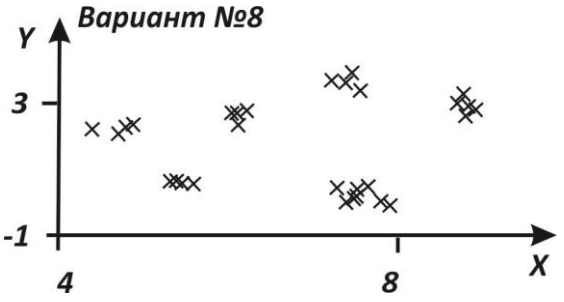


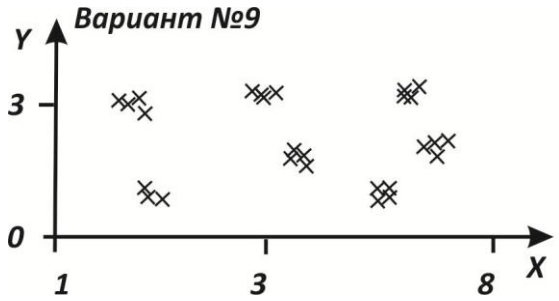


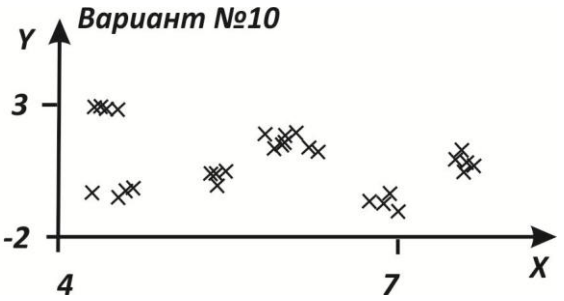












Контрольные задания для лабораторной работы №5

Исследовать функции $trimf$, $trapmf$, $gauss2mf$, $gaussmf$, $gbellmf$, $sigmf$, $dsigmf$, $psigmf$, zmf, $pimf$, smf, min, max, $prod$, $prober$, а также операцию дополнения при следующих значениях переменных:

Вариант 1
$x = 0{:}\,0{,}1{:}\,10; a = 2; b = 5; c = 9; d = 11;$
$a1 = 2; c1 = 5; a2 = 9; c2 = 11;$
$y1 = gbellmf(x, [a\ b\ c]);$
$y2 = gaussmf(x, [a\ c])$

Вариант 2
$x = 0{:}\,0{,}1{:}\,10; a = 1; b = 2; c = 8; d = 9;$
$a1 = 1; c1 = 2; a2 = 8; c2 = 9;$
$y1 = gauss2mf(x, [a1\ c1\ a2\ c2]);$
$y2 = gaussmf(x, [a\ c])$

Вариант 3
$x = 0{:}\,0{,}1{:}\,10; a = 0{,}5; b = 4; c = 7; d = 9;$
$a1 = 0{,}5; c1 = 4; a2 = 9; c2 = 10;$
$y1 = sigmf(x, [a\ c]);$
$y2 = gaussmf(x, [a\ c])$

Вариант 4
$x = 0{:}\,0{,}1{:}\,10; a = 0{,}7; b = 3{,}5;$
$c = 9; d = 9{,}5;$
$a1 = 0{,}7; c1 = 3{,}5; a2 = 9; c2 = 9{,}5;$
$y1 = trimf(x, [a\ b\ c]);$
$y2 = gaussmf(x, [a\ c])$

Вариант 5
$x = 0{:}\,0{,}1{:}\,10; a = 0; b = 2; c = 6; d = 11;$
$a1 = 0; c1 = 2; a2 = 6; c2 = 11;$
$y1 = gbellmf(x, [a\ b\ c]);$
$y2 = trimf(x, [a\ b\ c])$

Вариант 6
$x = 0{:}\,0{,}1{:}\,10; a = 2; b = 3; c = 7; d = 9;$
$a1 = 2; c1 = 3; a2 = 7; c2 = 9;$
$y1 = trimf(x, [a\ b\ c]);$
$y2 = trapmf(x, [a\ b\ c\ d])$

Вариант 7
$x = 0{:}\,0{,}1{:}\,10; a = 0; b = 1{,}5;$
$c = 6; d = 15;$
$a1 = 0; c1 = 1{,}5; a2 = 6; c2 = 15;$
$y1 = trapmf(x, [a\ b\ c\ d]);$
$y2 = gaussmf(x, [a\ c])$

Вариант 8
$x = 0{:}\,0{,}1{:}\,10; a = 2; b = 2; c = 7; d = 9;$
$a1 = 2; c1 = 2; a2 = 7; c2 = 9;$
$y1 = trimf(x, [a\ b\ c]);$
$y2 = gauss2mf(x, [a1\ c1\ a2\ c2])$

Вариант 9
$x = 0{:}\,0{,}1{:}\,10; a = 0; b = 5; c = 7; d = 11;$
$a1 = 0; c1 = 5; a2 = 7; c2 = 11;$
$y1 = zmf(x, [a\ b]);$
$y2 = pimf(x, [a\ b\ c\ d])$

Вариант 10
$x = 0{:}\,0{,}1{:}\,10; a = 2; b = 3; c = 7; d = 8;$
$a1 = 2; c1 = 3; a2 = 7; c2 = 8;$
$y1 = gbellmf(x, [a\ b\ c]);$
$y2 = pimf(x, [a\ b\ c\ d])$

Вариант 11
$x = 0{:}\,0{,}1{:}\,10; a = 1; b = 3; c = 5; d = 8;$
$a1 = 2; c1 = 3; a2 = 6; c2 = 7;$
$y1 = gbellmf(x, [a\ b\ c]);$
$y2 = pimf(x, [a\ b\ c\ d])$

Вариант 12
$x = 0{:}\,0{,}1{:}\,10; a = 0; b = 1; c = 3; d = 8;$
$a1 = 2; c1 = 4; a2 = 5; c2 = 9;$
$y1 = zmf(x, [a\ b]);$
$y2 = trimf(x, [a\ b\ c])$

Вариант 13
$x = 0{:}\,0{,}1{:}\,10; a = 1; b = 3; c = 7; d = 8;$
$a1 = 2; c1 = 5; a2 = 7; c2 = 8;$
$y1 = trimf(x, [a\ b\ c]);$
$y2 = pimf(x, [a\ b\ c\ d])$

Вариант 14
$x = 0{:}\,0{,}1{:}\,10; a = 0; b = 2; c = 7; d = 9;$
$a1 = 2; c1 = 3; a2 = 7; c2 = 8;$
$y1 = gbellmf(x, [a\ b\ c]);$
$y2 = trimf(x, [a\ b\ c])$

Контрольные задания для лабораторной работы №6

1. Оценить степень инвестиционной привлекательности предоставления кредита на основании данных о проценте по кредиту, срока кредита и суммы кредита.

2. Оценить степень привлекательности получения высшего образования на основании данных о сроке обучения, стоимости обучения и ожидаемом уровне заработной платы по выбранной специальности.

3. Оценить степень удовлетворенности клиентов работой отделения банка на основании данных о среднем количестве клиентов в день и количестве заключаемых договоров.

4. Оценить степень привлекательности приобретения недвижимого имущества в зависимости от удаленности от центра, площади, стоимости.

5. Оценить степень привлекательности приобретения автомобиля в зависимости от стоимости, объема потребляемого топлива, пробеге автомобиля.

6. Оценить автомобиль исходя из состояния кузова, данных о состоянии двигателя и лакокрасочного покрытия.

7. Оценить фильм по данным о сюжете, актерах, съемке.

8. Оценить блюдо исходя из информации о калорийности, полезности и предполагаемом времени потребления.

9. В зависимости от калорийности, содержания белковой углеводов определить будет ли диета являться нормальным питанием, способствовать снижению веса или являться спортивной.

10. Дайте оценку ПК в зависимости от частоты процессора, объема оперативной памяти и объема жесткого диска.

11. Определить спелость ананаса по значениям таких показателей как твердость корки, интенсивности аромата и цвету мякоти.

12. Оценить велосипед по значениям таких показателей как количество скоростей, ширина колес и стоимость велосипеда.

13. Подобрать одеяло в соответствии с такими характеристиками как размер одеяла, степень теплоты одеяла и состав наполнителя.

14. Подобрать пылесос в зависимости от мощности всасывания, наличия аксессуаров и стоимости.

15. Подобрать породу собаки по размеру, агрессивности и обучаемости.

Контрольные задания для лабораторной работы №7

Исходя из множества данный, подлежащих классификации ($fcmdata$) и заданного количества кластеров ($cluster\_n$), определить центры кластеризации, получить график изменения значений целевой функции.

Вариант 1

fcmdata=[0 1, 1.1 2.5, 0.4 2.2, 4.5 2.7, 3.4 1.2, 2.2 1.8, 0.5 0.4, 3 2.2, 1 1.6, 2.3 4, 1 4.7, 0.2 0.7, 0.9 4.4, 0.3 5, 0.1 2.1, 3.1 4.7, 3.2 5, 1 1, 0 1.6, 4 2.2, 1.6 2.2, 5 2.3, 4.1 4.6, 3.3 1, 2.8 2.9, 1.8 0.7, 1.9 3.4, 4 1.5, 3.8 1]

$cluster\_n = 4$

Вариант 2

fcmdata=[0.5 1.9, 1.1 2.4, 0.8 2.8, 4.5 2.7, 3.7 1.2, 2.2 1.8, 0.1 0.4, 3 2.2, 1.1 1.6, 2.3 4.2, 1 4.7, 0.2 2.7, 0.9 4.4, 0.8 5, 0 2.1, 3.1 4, 3.2 5, 3.1 1, 0 1.6, 4.5 2, 1.9 2.1, 3.5 2.3, 4.1 4.6, 3.3 1, 2.6 2.9, 1 0.7, 1.9 3.5, 4.1 5, 3.8 1]

$cluster\_n = 5$

Вариант 3

fcmdata=]0 5, 1.5 2.1, 3.4 2.1, 4.5 2.4, 3.4 1.2, 2.2 1.8, 0.9 0.2, 3.5 2.1, 1 1.6, 2.3 4, 1 4.7, 0.9 0.7, 0.1 4.9, 0.3 5, 0.1 2.1, 3.7 4.2, 3.2 5, 1 1, 0 1.6, 4 4.2, 1.9 2.2, 2.5 2.3, 2.1 4.6, 3.3 1, 3.8 2.9, 4.8 0.7, 2.9 3.4, 4.1 5, 1.8 1[

$cluster\_n = 3$

Вариант 4

fcmdata=[0.7 1.9, 1.8 2.2, 0.4 2.2, 4.5 2.7, 3.4 1.2, 2.2 1.8, 0.5 0.4, 3 2.2, 1 1.6, 2.3 4, 1 4.7, 0.2 0.7, 0.3 2.4, 0.3 5, 0.9 2.8, 3.1 4.7, 1.2 3.5, 1 1, 0.7 1.2, 4 2.2, 1.2 2.2, 5 2.3, 4.1 4.6, 1.3 1.7, 2.8 2.9, 1.8 0.2, 1.9 3.4, 4.2 1.9, 3.8 1]

$cluster\_n = 4$

Вариант 5

fcmdata=[0 1, 1.9 2.1, 3.4 4.2, 2.5 2.2, 1.4 1.2, 2.1 1.8, 0.5 0.4, 3 2.9, 1 1.6, 2.3 4, 1 4.7, 0.2 0.5, 0.9 4.4, 0.9 5, 4.1 2.9, 3.1 4.2, 3.2 1.5, 1 1, 0 1.6, 4 3.2, 1.6 2.2, 5 2.3, 4.1 2.6, 1.3 1, 1.8 2.1, 1.8 0.7, 1.9 3.4, 4 1.5, 3.3 1]

$cluster\_n = 5$

Вариант 6

fcmdata=[0 4, 1.7 3.5, 0.4 4.2, 4.5 2.7, 3.8 1.1, 2.2 1.8, 0.5 0.9, 1.3 2.2, 1 1.6, 2.3 1, 1 4.7, 4.2 2.7, 0.9 4.4, 0.3 5, 4.1 2.6, 3.1 4.7, 3.9 5, 1.7 1, 0.5 1.9, 4 2.2, 2.6 2.8, 5 2.3, 2.1 4.6, 3.2 1, 2.8 2.9, 1.8 0.7, 1.9 3.4, 4 1.5, 3.8 1]

$cluster\_n = 4$

Вариант 7

fcmdata=[4 1, 1.2 2, 0 2.2, 4.5 2.7, 3.8 2.2, 2.9 1.8, 0.5 0.9, 3 2.2, 1 1.6, 2.3 4, 1 4.7, 0.2 0.7, 0.9 4.4, 0.3 5, 0.1 2.1, 3.1 4.7, 1.2 5, 1 1, 0 3.6, 1 2.2, 1.7 2.2, 4.2 2.3, 2.1 4.6, 1.3 1, 2.8 2.9, 1.8 2.7,1.9 3.4, 4 3.5, 3.5 4]

$cluster\_n = 4$

Вариант 8

fcmdata=]0 5, 1.5 2.1, 3.4 2.1, 4.5 2.4, 3.4 1.2, 2.2 1.8, 0.9 0.2, 3.5 2.1, 1 1.6, 2.3 4, 1 4.7, 0.9 0.7, 0.1 4.9, 0.3 5, 0.1 2.1, 3.7 4.2, 3.2 5, 1 1, 0 1.6, 4 4.2, 1.9 2.2, 2.5 2.3, 2.1 4.6, 3.3 1, 3.8 2.9, 4.8 0.7, 2.9 3.4, 4.1 5, 1.8 1[

$cluster\_n = 3$

Вариант 9

fcmdata=[0.7 1.9, 1.8 2.2, 0.4 2.2, 4.5 2.7, 3.4 1.2, 2.2 1.8, 0.5 0.4, 3 2.2, 1 1.6, 2.3 4, 1 4.7, 0.2 0.7, 0.3 2.4, 0.3 5, 0.9 2.8, 3.1 4.7, 1.2 3.5, 1 1, 0.7 1.2, 4 2.2, 1.2 2.2, 5 2.3, 4.1 4.6, 1.3 1.7, 2.8 2.9, 1.8 0.2, 1.9 3.4, 4.2 1.9, 3.8 1]

$cluster\_n = 4$

Вариант 10

fcmdata=[0 1, 1.9 2.1, 3.4 4.2, 2.5 2.2, 1.4 1.2, 2.1 1.8, 0.5 0.4, 3 2.9, 1 1.6, 2.3 4, 1 4.7, 0.2 0.5, 0.9 4.4, 0.9 5, 4.1 2.9, 3.1 4.2, 3.2 1.5, 1 1, 0 1.6, 4 3.2, 1.6 2.2, 5 2.3, 4.1 2.6, 1.3 1, 1.8 2.1, 1.8 0.7, 1.9 3.4, 4 1.5, 3.3 1]

$cluster\_n = 5$

Вариант 11

fcmdata=[0 4, 1.7 3.5, 0.4 4.2, 4.5 2.7, 3.8 1.1, 2.2 1.8, 0.5 0.9, 1.3 2.2, 1 1.6, 2.3 1, 1 4.7, 4.2 2.7, 0.9 4.4, 0.3 5, 4.1 2.6, 3.1 4.7, 3.9 5, 1.7 1, 0.5 1.9, 4 2.2, 2.6 2.8, 5 2.3, 2.1 4.6, 3.2 1, 2.8 2.9, 1.8 0.7, 1.9 3.4, 4 1.5, 3.8 1]

$cluster\_n = 4$

Вариант 12

fcmdata=[4 1, 1.2 2, 0 2.2, 4.5 2.7, 3.8 2.2, 2.9 1.8, 0.5 0.9, 3 2.2, 1 1.6, 2.3 4, 1 4.7, 0.2 0.7, 0.9 4.4, 0.3 5, 0.1 2.1, 3.1 4.7, 1.2 5, 1 1, 0 3.6, 1 2.2, 1.7 2.2, 4.2 2.3, 2.1 4.6, 1.3 1, 2.8 2.9, 1.8 2.7,1.9 3.4, 4 3.5, 3.5 4]

$cluster\_n = 4$

Вариант 13

fcmdata=[0 1, 1.1 2.5, 0.4 2.2, 4.5 2.7, 3.4 1.2, 2.2 1.8, 0.5 0.4, 3 2.2, 1 1.6, 2.3 4, 1 4.7, 0.2 0.7, 0.9 4.4, 0.3 5, 0.1 2.1, 3.1 4.7, 3.2 5, 1 1, 0 1.6, 4 2.2, 1.6 2.2, 5 2.3, 4.1 4.6, 3.3 1, 2.8 2.9, 1.8 0.7, 1.9 3.4, 4 1.5, 3.8 1]

$cluster\_n = 4$

Вариант 14

fcmdata=[0.5 1.9, 1.1 2.4, 0.8 2.8, 4.5 2.7, 3.7 1.2, 2.2 1.8, 0.1 0.4, 3 2.2, 1.1 1.6, 2.3 4.2, 1 4.7, 0.2 2.7, 0.9 4.4, 0.8 5, 0 2.1, 3.1 4, 3.2 5, 3.1 1, 0 1.6, 4.5 2, 1.9 2.1, 3.5 2.3, 4.1 4.6, 3.3 1, 2.6 2.9, 1 0.7, 1.9 3.5, 4.1 5, 3.8 1]

$cluster\_n = 5$

Вариант 15

fcmdata=[0 4, 1.7 3.5, 0.4 4.2, 4.5 2.7, 3.8 1.1, 2.2 1.8, 0.5 0.9, 1.3 2.2, 1 1.6, 2.3 1, 1 4.7, 4.2 2.7, 0.9 4.4, 0.3 5, 4.1 2.6, 3.1 4.7, 3.9 5, 1.7 1, 0.5 1.9, 4 2.2, 2.6 2.8, 5 2.3, 2.1 4.6, 3.2 1, 2.8 2.9, 1.8 0.7, 1.9 3.4, 4 1.5, 3.8 1]

$cluster\_n = 4$

Учебное пособие

Сысоев Дмитрий Валериевич
Курипта Оксана Валериевна
Проскурин Дмитрий Константинович

ВВЕДЕНИЕ В ТЕОРИЮ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Отпечатано в авторской редакции

Подписано в печать _____ Формат 60 х 84 1/16 Уч.-изд.л. ____
Усл.-печ.л. _____ Бумага писчая. Тираж 500 экз. Заказ №_____

Отпечатано: отдел оперативной полиграфии
издательства учебной литературы и учебно-методических пособий
Воронежского ГАСУ
396006, Воронеж, ул. 20-летия Октября,84