Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Уфимский государственный авиационный технический университет»

Кафедра технической кибернетики

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОГРАММЫХ ПРИЛОЖЕНИЯХ

Лабораторный практикум
по дисциплинам «Методы искусственного интеллекта в управлении», «Интеллектуальное управление сложными объектами», «Интеллектуальное управление сложными техническими объектами», «Методы искусственного интеллекта в управлении техническими объектами», «Программные системы и комплексы в управлении качеством»

Уфа 2021

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Уфимский государственный авиационный технический университет»

Кафедра технической кибернетики

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОГРАММЫХ ПРИЛОЖЕНИЯХ

Лабораторный практикум
по дисциплинам «Методы искусственного интеллекта в управлении», «Интеллектуальное управление сложными объектами», «Интеллектуальное управление сложными техническими объектами», «Методы искусственного интеллекта в управлении техническими объектами», «Программные системы и комплексы в управлении качеством»

Учебное электронное издание сетевого доступа

Уфа 2021

Авторы-составители: Б. Г. Ильясов, Е. А. Макарова, Е. Ш. Закиева,
Э. Р. Габдуллина

Методы искусственного интеллекта в программных приложениях : лабораторный практикум по дисциплинам «Методы искусственного интеллекта в управлении», «Интеллектуальное управление сложными объектами», «Интеллектуальное управление сложными техническими объектами», «Методы искусственного интеллекта в управлении техническими объектами», «Программные системы и комплексы в управлении качеством» [Электронный ресурс] / Уфимск. гос. авиац. техн. ун-т ; [авт.-сост. : Б. Г. Ильясов, Е. А. Макарова, и др.]. – Уфа : УГАТУ, 2021. – URL: https://www.ugatu.su/media/uploads/MainSite/Ob%20universitete/Izdateli/El_izd/2021-52.pdf

Цель лабораторного практикума – закрепление и совершенствование знаний студентов по дисциплинам «Методы искусственного интеллекта в управлении», «Интеллектуальное управление сложными объектами», «Интеллектуальное управление сложными техническими объектами», «Методы искусственного интеллекта в управлении техническими объектами», «Программные системы и комплексы в управлении качеством», формирования умений и навыков решения задач управления с использованием методов искусственного интеллекта в современных программных приложениях.

Предназначен для бакалавров по направлениям подготовки 27.03.03 Системный анализ и управление, 27.03.04 Управление в технических системах, для магистров по направлениям подготовки 27.04.02 Управление качеством.

Рецензент канд. техн. наук, доцент С. В. Сильнова

При подготовке электронного издания использовались следующие программные средства:

- Adobe Acrobat – текстовый редактор;
- Microsoft Word – текстовый редактор.

Авторы-составители: *Ильясов Барый Галеевич,*
Макарова Елена Анатольевна,
Закиева Елена Шавкатовна,
Габдуллина Эльвира Риятовна

Компьютерная верстка: *О. А. Соколова*
Программирование и компьютерный дизайн: *А. П. Меркулова*

ФГБОУ ВО «Уфимский государственный авиационный технический университет»
450008, Уфа, ул. К. Маркса, 12.
Тел.: +7-908-35-05-007
e-mail: rik.ugatu@yandex.ru

ВВЕДЕНИЕ

Для решения задач контроля и управления сложными техническими и социотехническими объектами в условиях неопределённости, при большой размерности объекта управления, наличии распределённых параметров, нелинейности, неполноте оценки внешних воздействий и состояний объекта, изменчивости целей, ограничений применяют *методы искусственного интеллекта.*

Материалы данного лабораторного практикума предназначены для закрепления знаний методов искусственного интеллекта, формирования умений и навыков решения задач управления с использованием этих методов в современных программных приложениях. В лабораторном практикуме рассматривается решение следующих задач:

– разработка экспертных систем на основе нечеткой логики для управления сложным объектом в системе *FIS Matlab;*

– нейросетевое моделирование для решения задач классификации, аппроксимации функции и прогнозирования временных рядов в аналитической платформе *Deductor;*

– нейросетевая кластеризация с помощью самоорганизующихся карт Кохонена в аналитической платформе *Deductor* и системе *Matlab;*

– нейро-нечеткое моделирование в системе *Anfis Matlab;*

– поиск оптимальных решений с помощью генетических алгоритмов в редакторе *Genetic Algorithm* системы *Matlab.*

Содержание лабораторного практикума:

– соответствует основной профессиональной образовательной программе (ОПОП) 27.03.03 *Системный анализ и управление*, содержанию рабочих программ дисциплин вариативной части учебного плана «Методы искусственного интеллекта в управлении» (180/5), «Интеллектуальное управление сложными объектами» (180/5) и направлено на освоение компетенций ПК-1 – способность принимать научно-обоснованные решения на основе математики, физики, химии, информатики, экологии, методов системного анализа и теории управления, теории знаний, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности; ПК-5 – способность разрабатывать методы моделирования, анализа и технологии синтеза процессов и систем в

области техники, технологии и организационных систем (*разделы 2,3,4, 109 ч*);

– соответствует ОПОП 27.03.04 *Управление в технических системах,* содержанию рабочей программы дисциплины вариативной части учебного плана «Интеллектуальное управление сложными техническими объектами» (144/4), «Методы искусственного интеллекта в управлении техническими объектами» (144/4) и направлено на освоение компетенции ПК-5 – способность осуществлять сбор и анализ исходных данных для расчета и проектирования систем и средств автоматизации и управления (*разделы 2,3,4, 109 ч*);

– соответствует ОПОП 27.04.02 *Управление качеством*, содержанию рабочей программы дисциплины вариативной части учебного плана «Программные системы и комплексы в управлении качеством» (144/4) и направлено на освоение компетенций ОПК-5 – способность к профессиональной эксплуатации современного оборудования и приборов; ПК-6 – способность осуществлять постановку задачи исследования, формирование плана его реализации (*раздел 5, 34 ч*).

Лабораторная работа № 1

НЕЧЕТКИЕ ЭКСПЕРТНЫЕ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ СЛОЖНЫМ ОБЪЕКТОМ В СИСТЕМЕ *FIS MATLAB*

1. Цель и задачи работы

Целью работы является закрепление знаний о построении нечетких экспертных систем, приобретение умений и навыков применения алгоритмов нечеткого логического вывода в системе *FIS Matlab*.

Задачами работы являются формирование умений выполнения процедур фаззификации, нечеткого логического вывода, дефаззификации, применения алгоритма Мамдани и Ларсена в системе *FIS Matlab*.

2. Теоретические сведения

2.1. Нечеткие экспертные системы

Экспертными системами (ЭС) называют системы искусственного интеллекта, осуществляющие поддержку принятия решений при *управлении объектами*, алгоритм функционирования которых заранее не известен, путем моделирования рассуждений на основе экспертных знаний (в форме эвристических правил) квалифицированных специалистов. Пользователем ЭС может быть человек, не обладающий квалификацией эксперта.

Статические ЭС включают в себя такие ключевые элементы как факты и данные в рабочей памяти, правила в базе знаний, механизмы логического вывода в решателе, которые являются основой для подсистем приобретения знаний, объяснений и диалогового компонента. *Динамические ЭС* учитывают изменения окружающей среды, происходящие за время решения задачи, поэтому они, опираясь в своей основе на структуру статических систем, имеют подсистемы моделирования внешнего мира и связи с внешним окружением через систему датчиков и контроллеров.

На рис. 1.1 представлена архитектура статической и динамической ЭС.

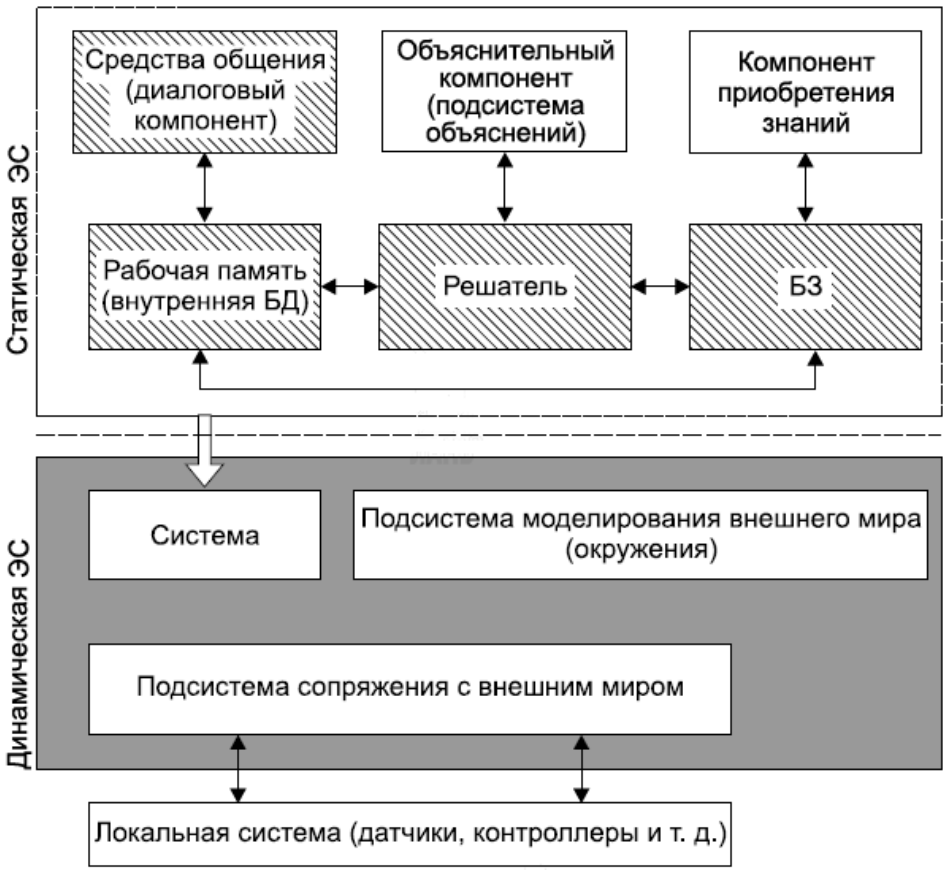


Рис. 1.1. Архитектура статической и динамической ЭС

Если *объект управления* (ОУ) относится к классу слабоформализованных систем, то информация о нем может быть фрагментарной, неполной, ненадёжной и, возможно, неточной. В этих случаях для разработки базы знаний и механизмов логического вывода ЭС применяется аппарат *нечеткой логики*.

Нечеткие экспертные системы (НЭС), или ЭС на основе нечеткой логики, используют в качестве модели представления знаний нечёткие продукционные правила *ЕСЛИ-ТО*, сформулированные на естественном языке. Ключевым преимуществом построения ЭС на основе логико-лингвистического представления нечетких экспертных знаний является компактное и адекватное представление реальных ситуаций функционирования ОУ.

Идея включения неточной информации в модели противоречит тому, что всегда требуется насколько возможно высокая точность – поэтому возникает расхождение между актуальностью и точностью. Это противоречие объясняет так называемый *принцип несовместимости*, сформулированный Л. А. Заде: «По мере усложнения системы наша способность делать абсолютные, точные

и значимые утверждения о поведении системы уменьшается. В какой-то момент будет возможен обмен между точностью и релевантностью». Следует отметить, что в обычной разговорной речи понимание собеседниками друг друга достигается достаточно точно, несмотря на то, что их рассуждения носят расплывчатый характер.

НЭС могут применяться для установления текущего состояния динамической системы как ОУ по совокупности параметров в данный момент времени; а также в системах поддержки принятия решений по управлению сложными процессами (объектами) при отсутствии математической модели и когда экспертные знания об объекте или процессе можно сформулировать в лингвистической форме. Структура НЭС представлена на рис. 1.2.

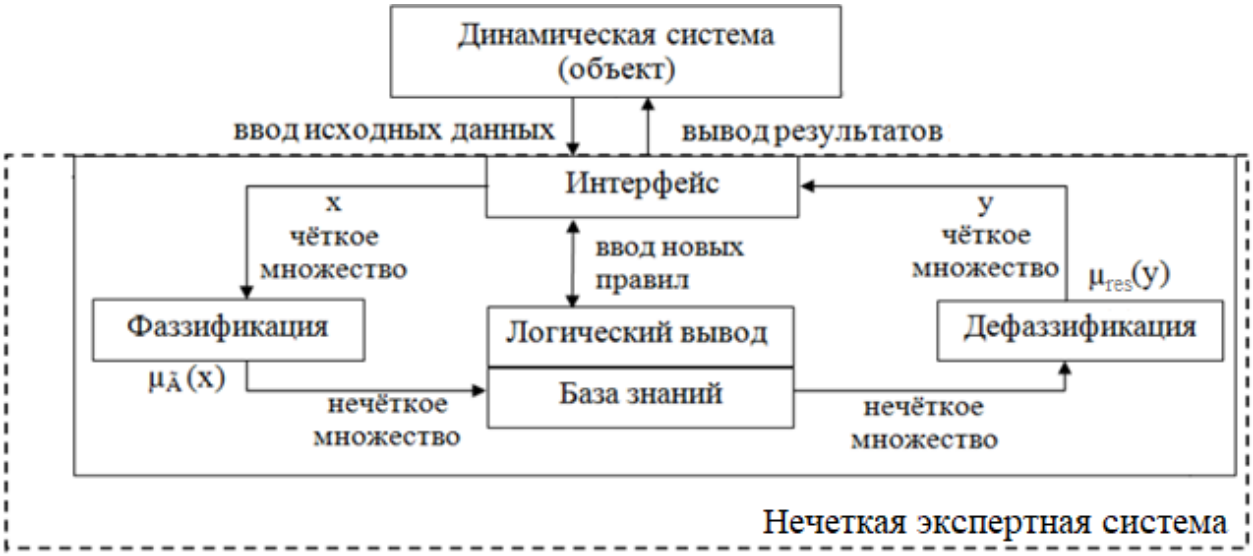


Рис. 1.2. Структура НЭС

Модуль интеллектуального интерфейса считывает данные, в т.ч. с датчиков, о состоянии динамической системы как ОУ и запускает механизм обработки, включающий этапы фаззификации, логического вывода на нечетких правилах и дефаззификации, а затем выводит результаты, которые могут быть использованы *исполнительными механизмами системы управления*, внешними по отношению к системе нечеткого вывода.

2.1.1. *Этап фаззификации.*

Фаззификацией называется преобра-зование чёткого множества входных данных в нечёткое множество, определяемое с помощью

значений функций принадлежности (ФП). Целью фаззификации является установление соответствия между конкретным численным значением отдельной входной переменной системы нечёткого вывода и значением ФП соответствующего ей терма входной лингвистической переменной.

Лингвистической переменной (ЛП) называется переменная, значениями которой могут быть слова или словосочетания естественного языка (скорость, температура, напряжение и т.д.).

Термом (лингвистическим значением) называется значение ЛП, выраженное в словесной форме (высокий, положительный, небольшой и т.д.). Термы всегда присутствуют в модели вместе со связанной ЛП (например, высокая скорость, низкая температура и т.д).

Лингвистическим терм-множеством называется множество всех возможных значений ЛП. Например, для ЛП «скорость» формируется терм-множество {малая, средняя, высокая}.

Для сокращения записей могут использоваться символические обозначения значений отдельных термов ЛП (табл. 1.1).

Таблица 1.1

Общепринятые обозначения термов ЛП

Символическое обозначение	Англоязычная нотация	Русскоязычная нотация
NB	*Negative Big*	Отрицательное большое
NM	*Negative Middle*	Отрицательное среднее
NS	*Negative Small*	Отрицательное малое
ZN	*Zero Negative*	Отрицательное близкое к нулю
Z	*Zero*	Нуль, близкое к нулю
ZP	*Zero Positive*	Положительное близкое к нулю
PS	*Positive Smal*	Положительное малое
PM	*Positive Middle*	Положительное среднее
PB	*Positive Big*	Положительное большое

Нечетким множеством A в некотором (непустом) пространстве X $(A \subseteq X)$ называется множество пар $A = \{(x,\ \mu_A(x));\ x \in X\}$, где $\mu_A(x)$: $X \to [0,1]$ – ФП нечеткого множества A, которая характеризует степень принадлежности каждого элемента x нечеткому множеству A.

Блок *фаззификации* вычисляет степени принадлежности четких числовых значений x^* входным нечетким множествам A_j. Вычисленные степени принадлежности $\mu_{A_j}(x^*)$ показывают, в какой степени входные значения x^* принадлежат конкретным нечетким множествам.

2.1.2. Этап нечеткого логического вывода

Блок *нечёткого логического вывода* (НЛВ) на основе степеней принадлежности $\mu_{A_j}(x^*)$ вычисляет результирующую ФП выходного значения модели $\mu_{res}(y)$, которая обычно имеет сложную форму и определяется посредством *логического вывода*.

База правил предназначена для формального представления эмпирических знаний экспертов о методах управления объектом в различных ситуациях, характере функционирования ОУ в различных условиях в форме нечетких продукционных правил *ЕСЛИ – ТО*.

Операция НЛВ включает в себя следующие этапы:

– *агрегирование* – процедура определения степени выполнения (истинности) условий каждого отдельного правила;

– *активизация,* или *вывод на правилах* – операция, которая заключается в определении активизированных ФП заключений отдельных правил;

– *аккумулирование* – это процедура нахождения одной результирующей ФП вывода из всей базы правил.

2.1.3. Этап дефаззификации

Дефаззификацией является обратное преобразование нечёткого множества в чёткое множество: на основе результирующей ФП $\mu_{res}(y)$ вычисляется четкое числовое значение y^* выходного параметра, являющееся результатом для входных числовых значений x^*. Существуют различные методы дефаззификации:

– *COG* (*Center of Gravity*) – метод центра тяжести, в котором определяется абсцисса центра тяжести фигуры, ограниченной графиком ФП;

– *MOM* (*Mean Of Maximums*) – метод среднего максимума;

– *LOM* (*Largest Of Maximums*) – метод наибольшего из максимумов;

– *SOM* (*Smallest Of Maximums*) – метод наименьшего из максимумов.

2.1.4. Алгоритмы нечеткого логического вывода

Различают *алгоритмы* НЛВ, отличающиеся друг от друга механизмом логического вывода.

Наибольшее применение в системах НЛВ получил *алгоритм Мамдани*, предложенный в 1975 г. английским математиком Э. Мамдани. Работу алгоритма можно проиллюстрировать на следующем примере.

Пусть x_1 и x_2 – входные переменные; y – выходная переменная; $A_1, A_2, B_1, B_2, C_1, C_2$ – некоторые заданные значения ФП.

База правил состоит из двух нечетких правила:

П 1: ЕСЛИ $(x_1 = A_1)$ И $(x_2 = B_1)$ ТО $(y = C_1)$;

П 2: ЕСЛИ $(x_1 = A_2)$ И $(x_2 = B_2)$ ТО $(y = C_2)$.

Необходимо определить результирующее четкое значение y_0 на основе входных четких значений x_1^* и x_2^* и нечетких правил П1 и П2. Этапы алгоритма Мамдани представлены в табл. 1.2.

Алгоритм Ларсена схож с алгоритмом Мамдани; отличием является то, что на этапе активизации заключений нечеткая импликация моделируется с использованием операции алгебраического произведения *PROD* вместо оператора *MIN* в алгоритме Мамдани (табл. 1.2). В случае немонотонных входных нечётких множеств алгоритм Ларсена оказывается точнее алгоритма Мамдани.

Таблица 1.2

Алгоритмы Мамдани и Ларсена

№	Шаг алг-ма	Описание
1	Фаззификация	определяются степени истинности для предпосылок каждого правила: $A_1(x_1^*)$, $A_2(x_1^*)$, $B_1(x_2^*)$, $B_2(x_2^*)$
2	Агрегирование	определяются уровни «отсечения» для предпосылок каждого из правил с использованием операции «*min*»: $\alpha_1 = A_1(x_1^*) \wedge B_1(x_2^*)$; $\alpha_2 = A_2(x_1^*) \wedge B_2(x_2^*)$
3	Активизация по Мамдани	определяются «усеченные» результирующие ФП (используется *min*-активизация): $C_1'(y) = (\alpha_1 \wedge C_1(y))$; $C_2'(y) = (\alpha_2 \wedge C_2(y))$
	Активизация по Ларсену	определяются «усеченные» результирующие ФП (используется *prod*-активизация): $C_1'(y) = (\alpha_1 \cdot C_1(y))$; $C_2'(y) = (\alpha_2 \cdot C_2(y))$
4	Аккумуляция	с использованием операции «*max*» производится объединение найденных усеченных ФП: $\mu_\Sigma(y) = C(y) = C_1'(y) \vee C_2'(y) = (\alpha_1 \wedge C_1(y)) \vee (\alpha_2 \wedge C_2(y))$
5	Дефаззификация	нахождение y_0 проводится любым из методов дефаззификации

2.2. Пример проектирования нечеткой экспертной системы для управления кондиционером воздуха в помещении

Разрабатывается НЭС для управления кондиционером воздуха в помещении. В помещении установлен бытовой кондиционер, который позволяет охлаждать или нагревать воздух. Очевидно, что наиболее комфортные условия создаются при некоторой стабильной температуре воздуха, однако из-за то, что температура среды вне помещения изменяется в течение суток, температура воздуха в помещении может колебаться. Задача состоит в том, чтобы разработать систему автоматического регулирования кондиционера для обеспечения постоянной температуры воздуха в помещении (рис. 1.3).

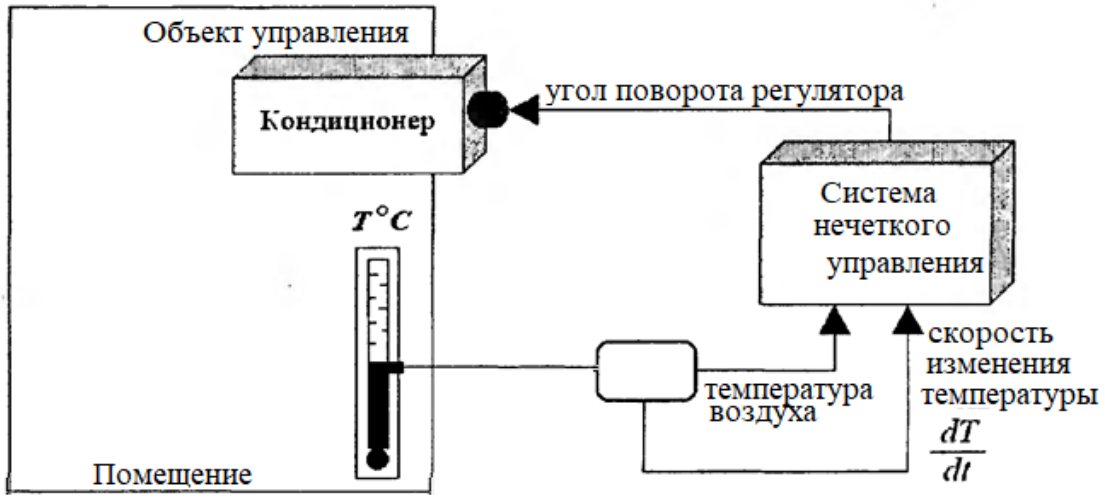


Рис. 1.3. Иллюстрация модели нечеткого управления кондиционером воздуха в помещении

Опыт использования бытовых кондиционеров показывает, что процесс охлаждения или нагревания воздуха в помещении обладает некоторой инерционностью; например, после включения режима «холод» происходит нагнетание холодного воздуха, что вызывается постепенное падение температуры, и даже в момент отключения этого режима температура все же продолжает падать в течение небольшого отрезка времени. Аналогичная ситуация наблюдается при включении и отключении режима «тепло». Поэтому необходимо рассматривать в качестве параметров объекта управления не только температуру воздуха в помещении, но и скорость изменения температуры.

Пусть в модели управления кондиционером включение режима «холод» осуществляется поворотом регулятора *влево*, а включение режима «тепло» – *вправо* относительно некоторой точки, в которой кондиционер выключен. Ручное регулирование температуры воздуха с помощью кондиционера позволило сформулировать следующие эвристические правила:

1. Если температура воздуха в помещении очень теплая, а скорость изменения температуры положительная, то следует включить режим «холод», повернув регулятор кондиционера на очень большой угол влево.

2. Если температура воздуха в помещении очень теплая, а скорость изменения температуры отрицательная, то следует включить режим «холод», повернув регулятор кондиционера на небольшой угол влево.

3. Если температура воздуха в помещении теплая, а скорость изменения температуры положительная, то следует включить режим «холод», повернув регулятор кондиционера на большой угол влево.

4. Если температура воздуха в помещении теплая, а скорость изменения температуры отрицательная, то кондиционер следует выключить.

5. Если температура воздуха в помещении очень холодная, а скорость измене-ния температуры отрицательная, то следует включить режим «тепло», повернув регулятор кондиционера на очень большой угол вправо.

6. Если температура воздуха в помещении очень холодная, а скорость измене-ния температуры положительная, то следует включить режим «тепло», повернув регулятор кондиционера на небольшой угол вправо.

7. Если температура воздуха в помещении холодная, а скорость изменения температуры отрицательная, то следует включить режим «тепло», повернув регулятор кондиционера на большой угол вправо.

8. Если температура воздуха в помещении холодная, а скорость изменения температуры положительная, то кондиционер следует выключить.

9. Если температура воздуха в помещении очень теплая, а скорость изменения температуры равна нулю, то следует включить режим «холод», повернув регулятор кондиционера на большой угол влево.

10. Если температура воздуха в помещении теплая, а скорость изменения температуры равна нулю, то следует включить режим «холод», повернув регулятор кондиционера на небольшой угол влево.

11. Если температура воздуха в помещении очень холодная, а скорость изменения температуры равна нулю, то следует включить режим «тепло», повернув регулятор кондиционера на большой угол вправо.

12. Если температура воздуха в помещении холодная, а скорость изменения температуры равна нулю, то следует включить режим «тепло», повернув регулятор кондиционера на небольшой угол вправо.

13. Если температура воздуха в помещении в пределах нормы, а скорость изменения температуры положительная, то следует включить режим «холод», повернув регулятор кондиционера на небольшой угол влево.

14. Если температура воздуха в помещении в пределах нормы, а скорость изме-нения температуры отрицательная, то следует включить режим «тепло», по-вернув регулятор кондиционера на небольшой угол вправо.

15. Если температура воздуха в помещении в пределах нормы, а скорость изменения температуры равна нулю, то кондиционер следует выключить.

Построенные правила становятся основой базы правил системы нечеткого вывода модели нечеткого управления, для формирования которой необходимо предварительно определить входные и выходные ЛП.

Очевидно, что в качестве входных ЛП следует использовать температуру воздуха в помещении (^{o}C) и скорость изменения температуры воздуха ($^{o}C/мин$).

В качестве терм-множества первой входной ЛП x_1 «Температура воздуха» используется множество A = {«очень холодная», «холодная», «в пределах нормы», «теплая», «очень теплая», которое записывается в символическом виде: A ={NB, NS, Z, PS, PB} (рис. 1.4).

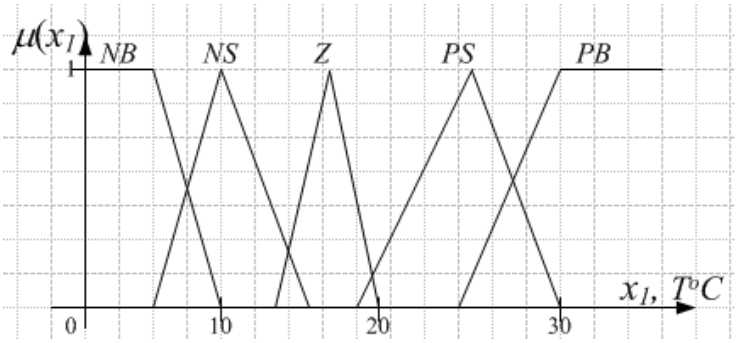


Рис. 1.4. График ФП для термов ЛП «*Температура воздуха*»

Терм-множеством второй входной ЛП x_2 «Скорость изменения температуры воздуха» является множество B ={«отрицательная», «равна нулю», «положительная»}, или B={*NS*, *Z*, *PS*} (рис. 1.5).

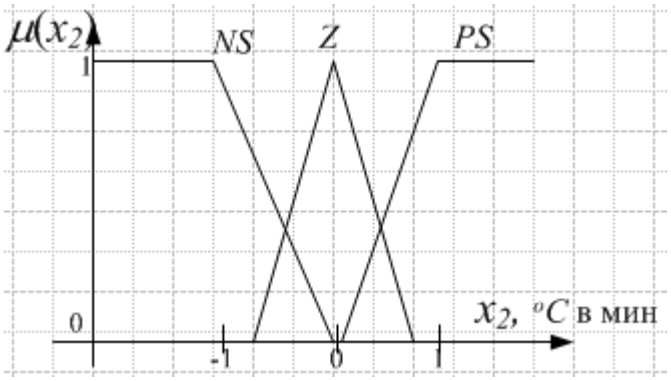


Рис. 1.5. График ФП для термов ЛП «*Скорость изменения температуры*»

В качестве выходной ЛП используется y – «угол поворота регулятора» включения режимов «холод» и «тепло» кондиционера, измеряемый в угловых градусах: поворот регулятора вправо означает включение режима «тепло» и положительное направление отсчета, а поворот влево – включение режима «холод» и отрицательное направление отсчета. Терм-множеством ЛП y является C = {«очень большой угол влево», «большой угол влево», «небольшой угол влево», «выключить кондиционер», «небольшой угол вправо», «большой угол вправо», «очень большой угол вправо»} или в символическом виде C={*NB*, *NM*, *NS*, *Z*, *PS*, *PM*, *PB*} с кусочно-линейными ФП (рис. 1.6).

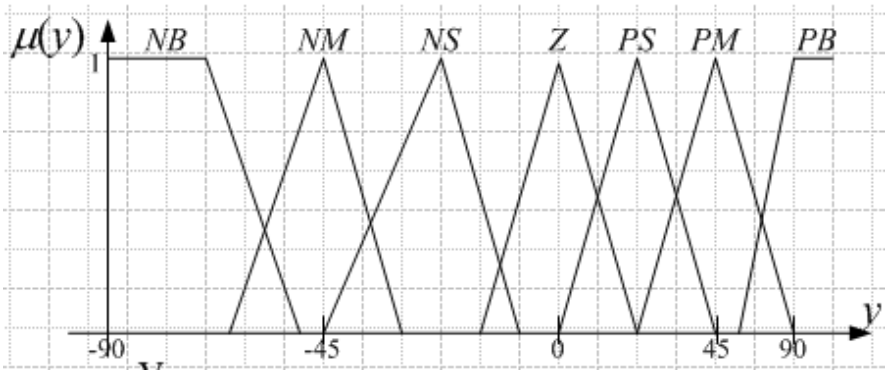


Рис. 1.6. График ФП для термов ЛП «*Угол поворота регулятора*»

База нечетких правил НЭС в матричном виде и в виде продукционных правил представлена в табл. 1.3 и 1.4.

Таблица 1.3

База нечетких правил в матричном виде

	NS	*Z*	*PS*
NB	*PB*	*PM*	*PS*
NS	*PM*	*PS*	*Z*
Z	*PS*	*Z*	*NS*
PS	*Z*	*NS*	*NM*
PB	*NS*	*NM*	*NB*

Таблица 1.4

База нечетких правил в виде продукционных правил

№	Правила
1	ЕСЛИ x_1=*PB* И x_2 = *PS*, ТО y = *NB*
2	ЕСЛИ x_1 = *PB* И x_2 = *NS*, ТО y = *NS*
3	ЕСЛИ x_1 = *PS* И x_2 = *PS*, ТО y = *NM*
4	ЕСЛИ x_1 = *PS* И x_2 = *NS*, ТО y = *Z*
5	ЕСЛИ x_1 = *NB* И x_2 = *NS*, ТО y = *PB*
6	ЕСЛИ x_1 = *NB* И x_2 = *PS*, ТО y = *PS*
7	ЕСЛИ x_1 = *NS* И x_2 = *NS*, ТО y = *PM*
8	ЕСЛИ x_1 = *NS* И x_2 = *PS,* ТО y = *Z*
9	ЕСЛИ x_1 = *PB* И x_2 = *Z*, ТО y = *NM*
10	ЕСЛИ x_1 = *PS* И x_2 = *Z*, ТО y = *NS*
11	ЕСЛИ x_1 = *NB* И x_2 = *Z*, ТО y = *PM*
12	ЕСЛИ x_1 = *NS* И x_2 = *Z*, ТО y = *PS*
13	ЕСЛИ x_1 = *Z* И x_2 = *PS*, ТО y = *NS*
14	ЕСЛИ x_1 = *Z* И x_2 = *NS*, ТО y = *PS*
15	ЕСЛИ x_1 = *Z* И x_2 = *Z*, ТО y = *Z*

На рис. 1.7 приведен пример работы алгоритма НЛВ Мамдани для случая, когда текущая температура воздуха в помещении равна $x_1^* = 19{,}5$ °С, а скорость изменения температуры положительная и равна $x_2^* = 0{,}2$ С/мин. Соответствующие подусловия используются в правилах нечетких продукций с номерами 3, 10, 13, 15. Эти правила считаются активными и используются в процессе нечеткого вывода. Результаты фаззификации переменных:

$\mu_z(x_1^* = 19{,}5)=0{,}5$ $\mu_z(x_2^* = 0{,}2)=0{,}7$

$\mu_{PS}(x_1^* = 19{,}5)=0{,}2$ $\mu_{PS}(x_2^* = 0{,}2)=0{,}3$

После агрегирования подусловий и активизации заключений в нечетких правилах продукций образуются «усеченные» ФП выходной переменной для каждого из правил, степени истинности для предпосылок каждого правила представлены ниже:

Правило 3: $\mu_{NM}(y)|_{x_1^*=19{,}5,\, x_2^*=0{,}2} = min\{min\,\{\mu_{PS}(x_1^*); \mu_{PS}(x_2^*)\}; \mu_{NM}(y)\} = 0{,}2;$

Правило 10: $\mu_{NS}(y)|_{x_1^*=19{,}5,\, x_2^*=0{,}2} = min\{min\,\{\mu_{PS}(x_1^*); \mu_Z(x_2^*)\}; \mu_{NS}(y)\} = 0{,}2;$

Правило 13: $\mu_{NS}(y)|_{x_1^*=19{,}5,\, x_2^*=0{,}2} = min\{min\,\{\mu_Z(x_1^*); \mu_{PS}(x_2^*)\}; \mu_{NS}(y)\} = 0{,}3;$

Правило 15: $\mu_Z(y)|_{x_1^*=19{,}5,\, x_2^*=0{,}2} = min\{min\,\{\mu_Z(x_1^*); \mu_Z(x_2^*)\}; \mu_Z(y)\} = 0{,}5.$

После аккумулирования заключений нечетких правил продукций с использованием операции *max*-дизъюнкции формируется нечеткое множество с ФП:

$\mu_{res}(y) = max\{\mu_{NM}(y);\ \mu_{NS}(y);\ \mu_{NS}(y);\ \mu_Z(y)\}.$

Далее проводится дефаззификация методом центра тяжести; если допустить, что результирующее нечеткое множество является дискретным, то приближенное значение управляющей переменной определяется по формуле: $y^* = \frac{\sum_{i=1}^{n} y_i \cdot \mu_{res}(y)}{\sum_{i=1}^{n} \mu_{res}(y)}$:

$\sum_{i=1}^{n} y_i \cdot \mu_{res}(y) = (-70) \cdot 0 + (-65 - 60 - 55 - 50 - 45) \cdot 0{,}2 + (-35 - 30 - 25 - 20) \cdot 0{,}3 + (-15) \cdot 0{,}4 + (-10 - 5 + 0 + 5 + 15) \cdot 0{,}5 + 15 \cdot 0 = -106;$

$\sum_{i=1}^{n} y_i \cdot \mu_{res}(y) = 0 + 0{,}15 \cdot 5 + 0{,}3 \cdot 5 + 0{,}4 + 0{,}5 \cdot 4 + 0 = 5{,}4;$

$y^* = \frac{-106}{5{,}4} = -19{,}6296 \approx -20^{\text{o}}.$

Итак, значение управляющей переменной y^* – угол поворота регулятора кондиционера влево – равен приблизительно 20°. Это значение соответствует включению режима «холод» на пятую часть своей мощности и является результатом решения задачи нечеткого вывода.

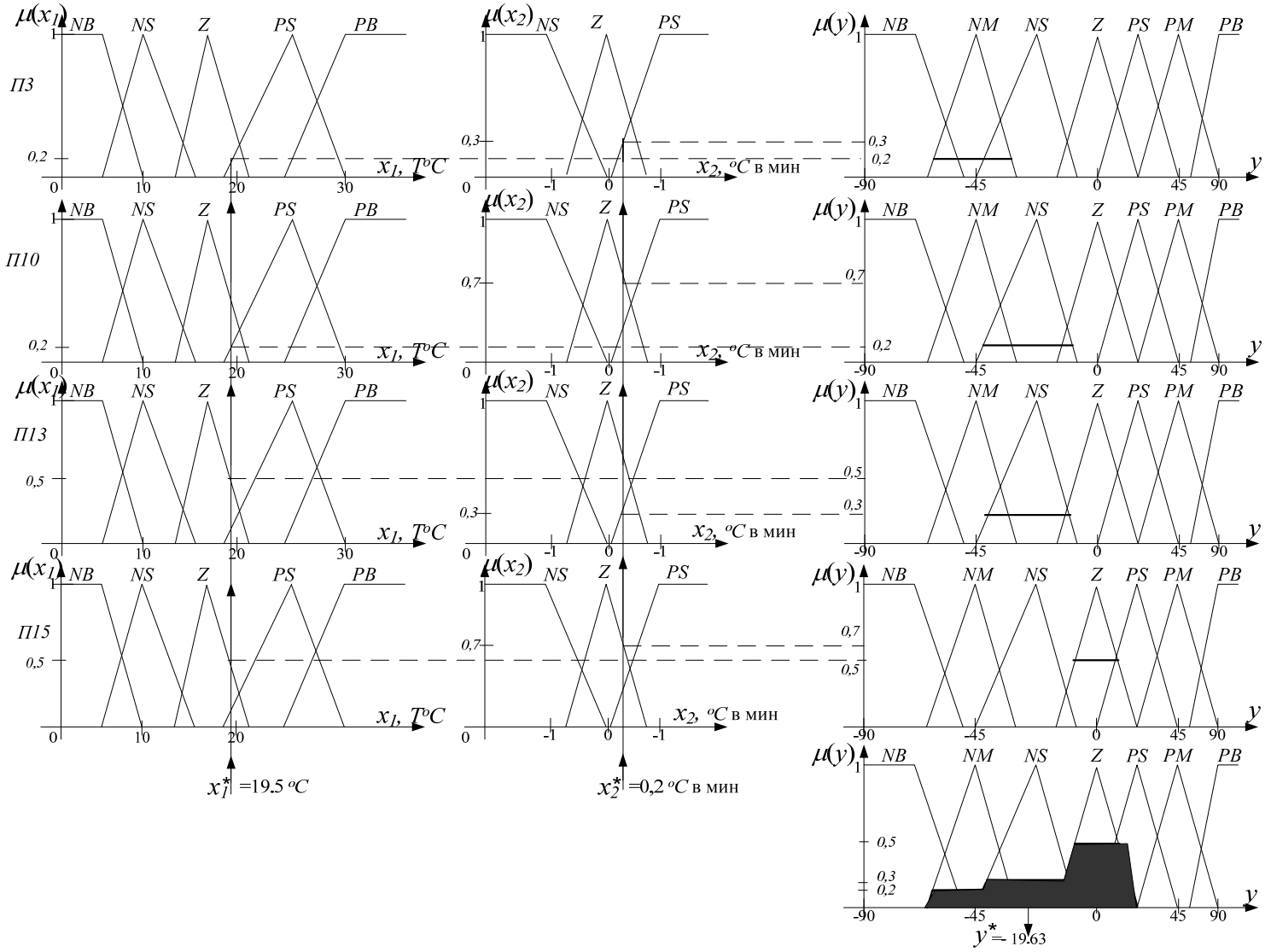


Рис. 1.7. Процедура нечеткого логического вывода на примере алгоритма Мамдани

2.3. Проектирование нечетких экспертных систем в *Matlab*

Для проектирования ЭС на основе нечеткой логики используется пакет расширения *Matlab – Fuzzy Logic Toolbox*, который поддерживает все стадии разработки нечетких систем; встроенные *GUI*-модули обеспечивают удобную среду с графическим интерфейсом.

Интерактивный режим разработки экспертных систем как систем нечеткого вывода (СНВ) осуществляется с использованием *GUI*-модулей *Fuzzy Inference System* (*FIS*) (рис. 1.8):

– редактор общих свойств *FIS Fuzzy Logic Designer* – основное средство, которое используется для создания или редактирования СНВ в графическом режиме;

– редактор функций принадлежности *Membership Function Editor* для задания и редактирования функций принадлежности;

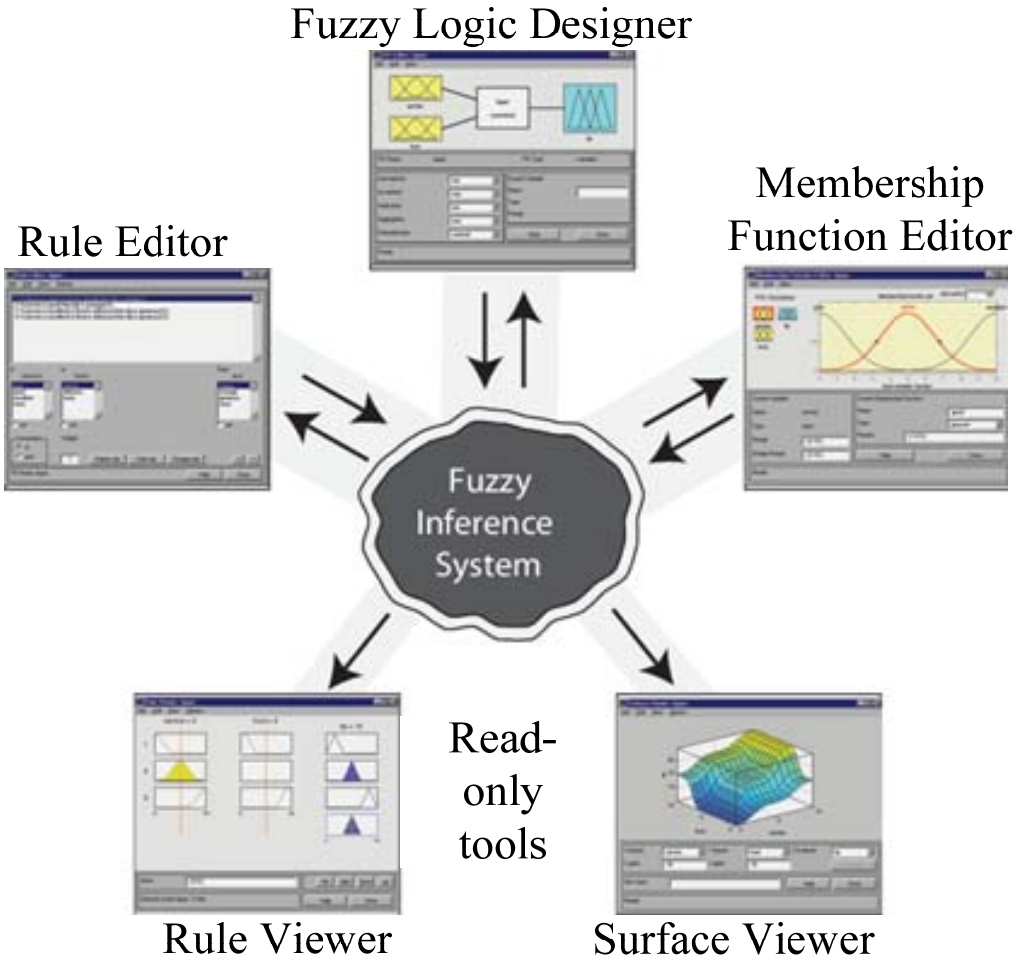


Рис. 1.8. Компоненты *Fuzzy Inference System Editor*

– редактор нечеткой базы знаний *Rule Editor* для задания и редактирования отдельных правил системы нечеткого вывода;

– программа просмотра (браузер) нечеткого вывода *Rule Viewer*;

– программа просмотра (браузер) поверхности *Surface Viewer* для визуализации графиков зависимости выходных переменных от отдельных входных переменных.

3. Методика выполнения работы

Обобщенная процедура построения нечеткой экспертной системы в *Matlab* рассматривается на примере задачи проектирования НЭС для управления кондиционером воздуха в помещении (п. 2.2).

3.1. Идентификация задачи проектирования НЭС

Первым этапом построения нечеткой экспертной системы является этап *идентификации* задачи проектирования НЭС:

1) формулируются цели построения ЭС;

2) определяются задачи, которые должна решать ЭС?;

3) устанавливается структура поддерживающих ее знаний.

Цель построения ЭС состоит в автоматическом регулировании кондиционера для обеспечения постоянной температуры воздуха в помещении.

Задачами системы является определение значений переменной «угол поворота регулятора» кондиционера воздуха, которые должны изменяться в зависимости от текущих значений переменных «Температура воздуха» и «Скорость изменения температуры воздуха». Диапазон показателей целевой переменной варьируется, начиная от «очень большого угла влево» до «очень большого угла вправо» и направлен на включение режимов «холод» и «тепло» кондиционера.

Дерево целей, представленное на рис. 1.9, является не только способом структуризации задачи, но и инструментом формализации знаний в виде правил, направленных на достижение целевой переменной.

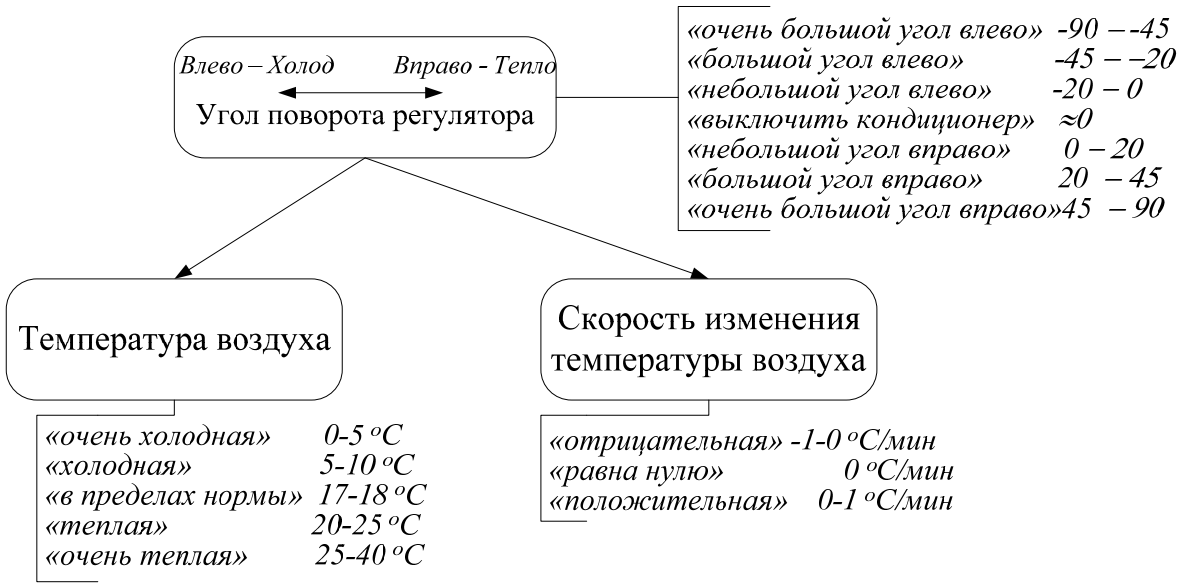


Рис. 1.9. Дерево целей системы управления кондиционером воздуха

Построение правил по дереву целей осуществляется снизу вверх, например, в виде: «ЕСЛИ *температура воздуха* = очень теплая *И скорость изменения температуры воздуха* = положительная, ТО *угол поворота регулятора* = очень большой угол влево».

3.2. Проектирование нечеткой системы

На следующем этапе осуществляется проектирование нечеткой системы. Для этого командой *fuzzy* в строке *Matlab* осуществляется загрузка основной интерфейсной программы пакета *Fuzzy Logic Toolbox – FIS*-редактор, который позволяет задать тип системы, ее имя, количество входных и выходных переменных, параметры СНВ.

Откроется новое графическое окно (рис. 1.10) с именем системы нечеткого вывода *Untitled* (расширение **.fis*).

По команде *File/New FIS…* создается новая система нечеткого выводв (СНВ) (в рассматриваемом примере – *Conditioner*); при ее вызове появляются две альтернативы *Mamdani* и *Sugeno,* задающие тип нечеткой системы (оставить тип СНВ *Mamdani* – по умолчанию).

Команда *File/Import* загружает ранее созданную и сохраненную систему либо из рабочей области *From Workspace…* или из файла *From Disk….* Команда *File/Export* сохраняет систему в рабочую область или на диск.

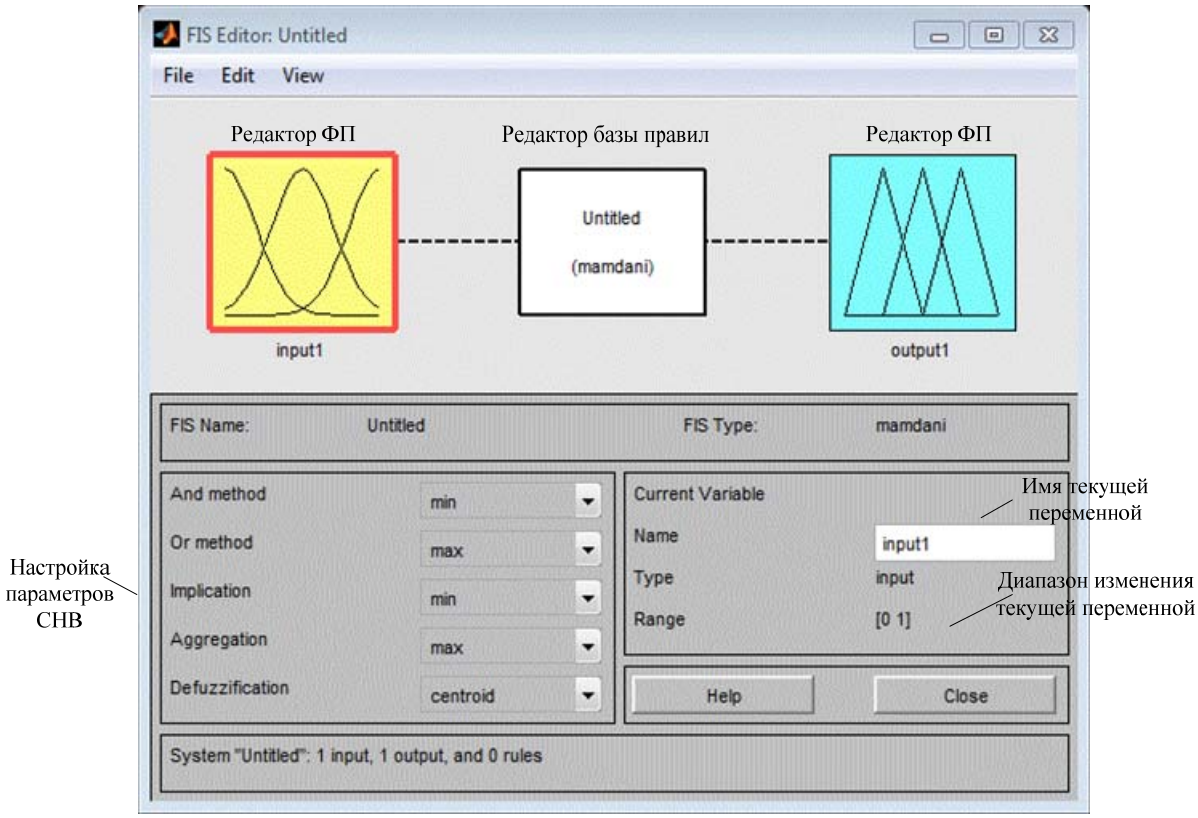


Рис. 1.10. Графический интерфейс редактора *FIS*

3.2.1. Описание входных и выходных переменных

Для построения СНВ требуются две входные и одна выходная переменная (рис. 1.11).

Первую входную переменную *input1*, заданную по умолчанию, необходимо переименовать, щелкнув левой кнопкой мыши на названии переменной и введя новое имя *Температура* в поле редактирования имени текущей переменной.

Аналогичным образом переименовывается выходная переменная *output1* на *Угол.*

Вторая входная переменная *Скорость* добавляется в СНВ по команде *File/Add Variable.*

Для определения лингвистических терм-множеств входных и выходных переменных вызывается редактор ФП *Membership Function Editor* – либо двойным щелчком на значке прямоугольника с именем соответствующей переменной либо командой *Edit / Membership Functions...* (предварительно должен быть выделен значок соответствующей ЛП).

В редакторе ФП переменной *Температура* вносятся изменения в соответствии с графиком ФП ЛП «*Температура воздуха*» (рис. 1.4, п. 2.2) следующим образом.

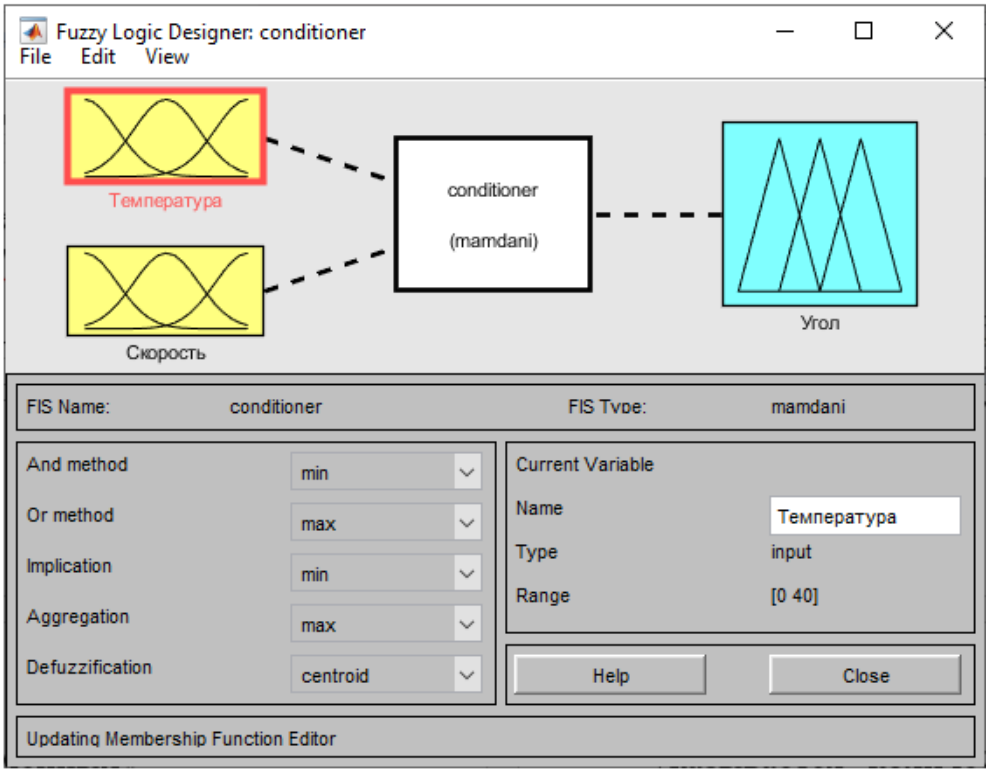


Рис. 1.11. Определение входных и выходных переменных СНВ *Conditioner*

Сначала устанавливается новый диапазон определения значений переменной: в полях ввода *Range* и *Display Range* изменяются значения с $0\,^{o}C$ до $40\,^{o}C$. Редактор ФП по умолчанию предлагает три терма с треугольными ФП, однако терм-множество ЛП «*Температура воздуха*» состоит из 5 термов {*NB*, *NS*, *Z*, *PS*, *PB*}.

Добавление термов в терм-множество текущей переменной осуществляется командой *Edit/Add MFs…*; в появившемся окне указывается количество добавляемых термов и тип их ФП. Для равномерного ввода термов ЛП рекомендуется перед добавлением новых термов удалить все имеющиеся термы, выделяя их указателем мыши и нажимая *Delete*.

Редактирование формы ФП с изменением границ и диапазонов осуществляется с помощью указателя мыши непосредственно на графике. Тип ФП указывается в поле *Type* – выбирается из списка; *Fuzzy Logic Toolbox* включает 11 встроенных типов ФП (рис. 1.12).

Имя терма вводится в поле *Name*, а поле *Params* автоматически заполняется параметрами ФП при изменении границ – в случае необходимости можно установить вручную значения ФП в квадратных скобках в поле *Params*.

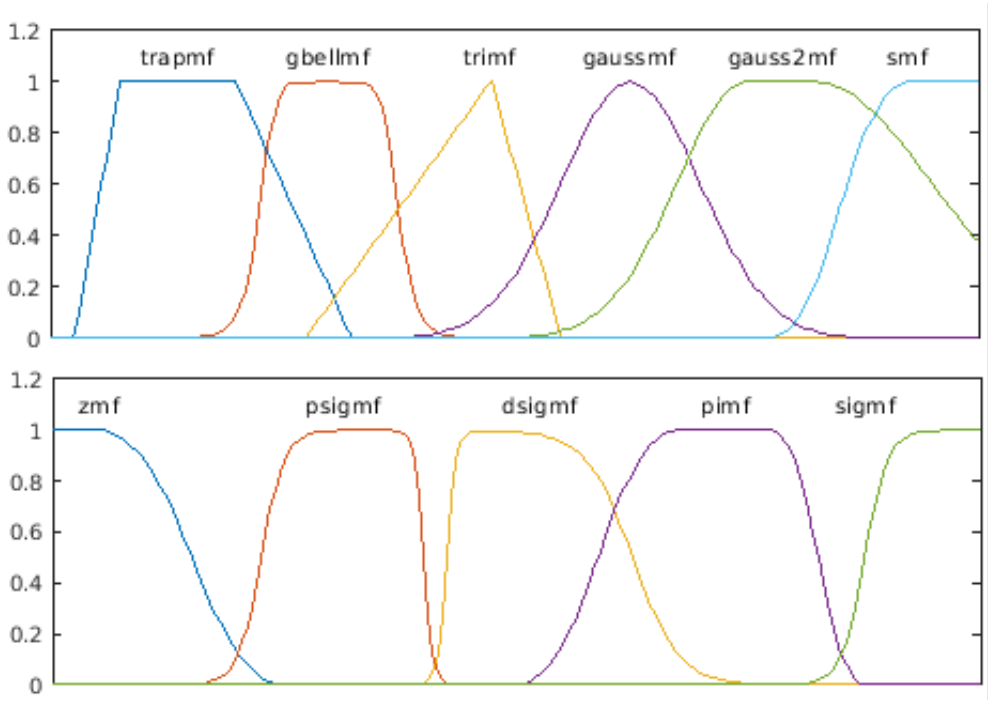


Рис. 1.12. Типы ФП в *Fuzzy Logic Toolbox*

Для ЛП *Температура* устанавливаются следующие параметры ФП: терм *NB* трапециевидной формы (*trapmf*) [-1 0 5 10]; терм *NS* треугольной формы (*trimf*) [5 10 15]; *Z* (*trimf*) [13,5 17 20,45], *PS* (*trimf*) [19 24 29], *PB* (*trapmf*) [25 30 42 43]. Построенные ФП для входной переменной «*Температура*» в редакторе *Membership Function Editor* представлены на рис. 1.13.

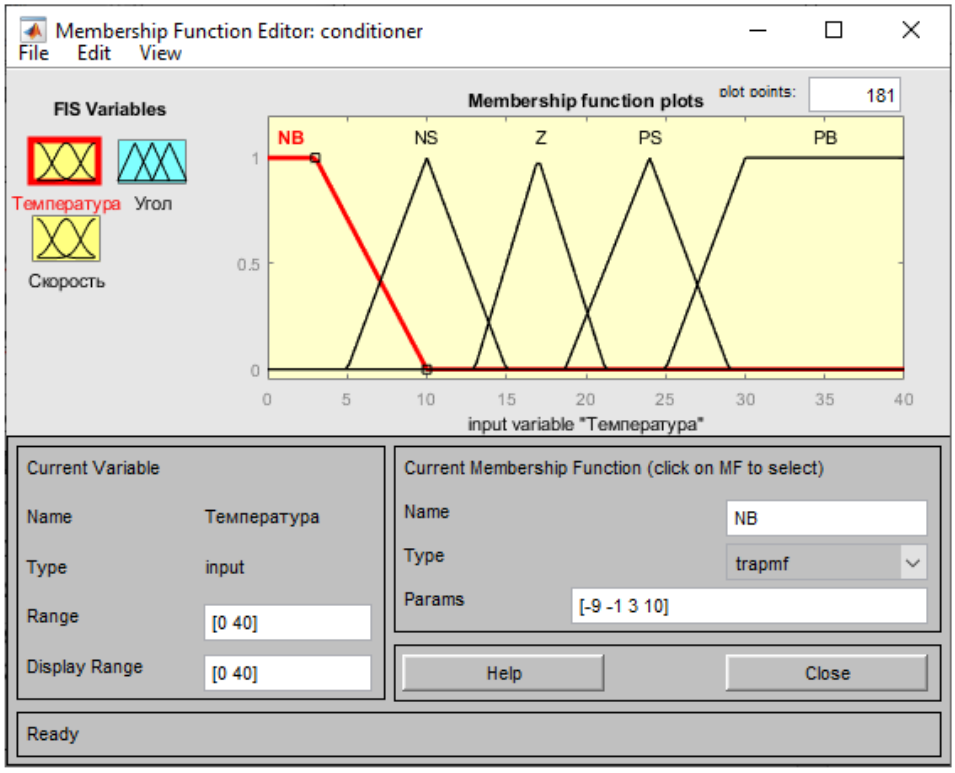


Рис. 1.13. ФП для входной переменной «*Температура*»

Аналогичным образом строятся ФП для входной переменной *Скорость*, в которую вносятся изменения в соответствии с графиком ФП ЛП «*Скорость изменения температуры воздуха*» (рис. 1.5, п. 2.2). Диапазон значений переменной устанавливается от –2 до 2 ^{o}C/мин. Терм-множество состоит из 3 термов {*NS*, *Z*, *PS*} (рис. 1.14): *NS* (*trapmf*) [-3 -2 -1 0]; *Z* (*trimf*) [-0,7 0 0,7], *PS* (*trapmf*) [0 1 2 3].

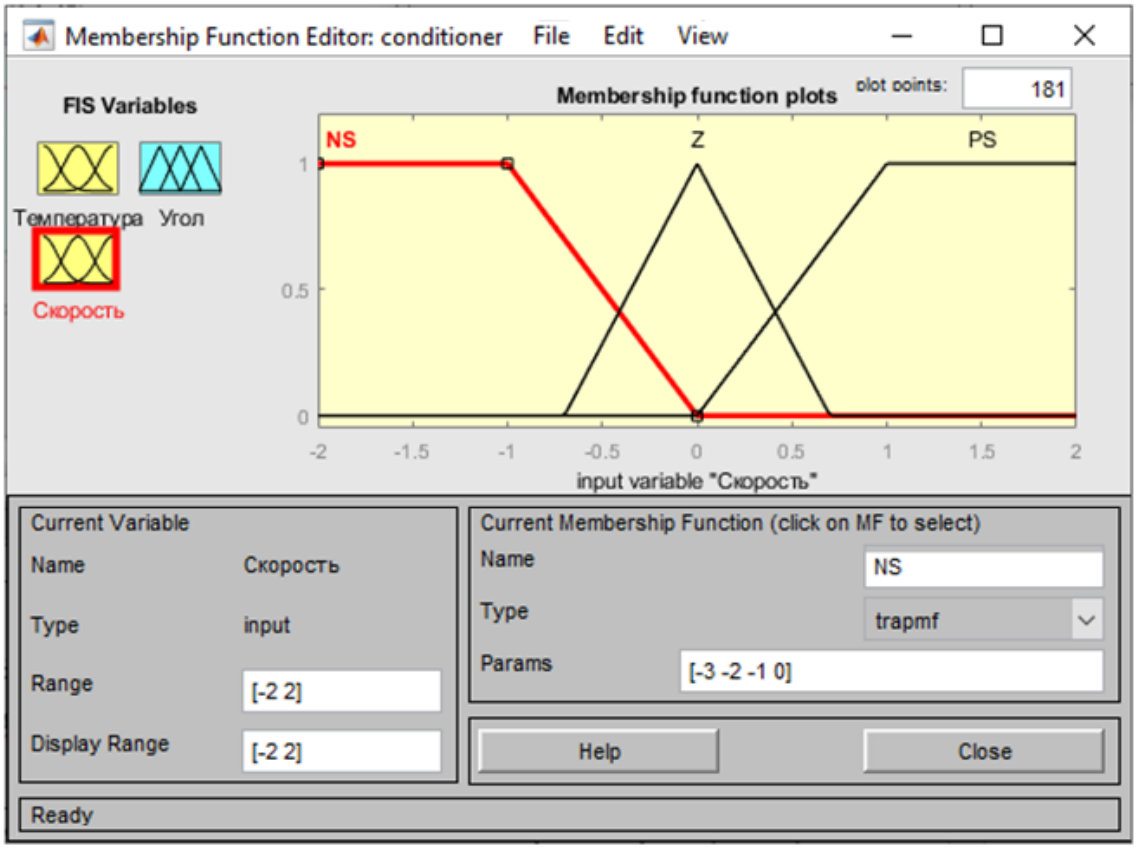


Рис. 1.14. ФП для входной переменной «*Скорость*»

И, наконец, для выходной переменной *Угол* в редакторе *Membership Function Editor* строятся ФП в соответствии с графиком ФП ЛП «*Угол поворота регулятора*» (рис. 1.6, п. 2.2). Диапазон значений переменной устанавливается от –90 до 90 угловых градусов. Терм-множество ЛП разделено на 7 термов {*NB*, *NM*, *NS*, *Z*, *PS*, *PM*, *PB*} (рис. 1.15): *NB* (*trapmf*) [-106 -100 -75 -50]; *NM* (*trimf*) [-70 -45 -25]; *NS* (*trimf*) [-45 -25 -5]; *Z* (*trimf*) [-15 0 15], *PS* (*trimf*) [5 25 45], *PM* (*trimf*) [25 45 70], *PB* [50 75 100 105].

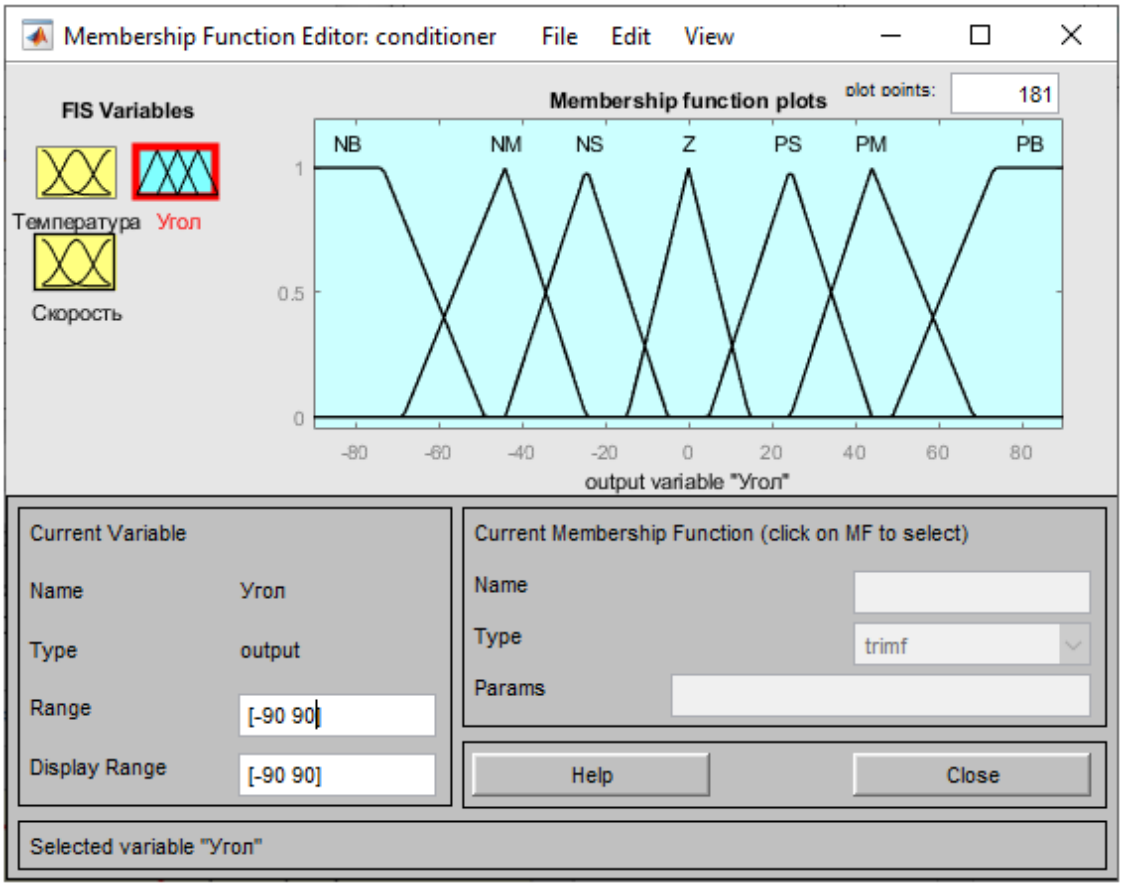


Рис. 1.15. ФП для выходной переменной «*Угол*»

3.2.2. Ввод и редактирование правил. Настройка параметров СНВ

Разработка правил нечеткой экспертной системы проводится в редакторе правил *Rule Editor*, запускаемого командой *Edit/Rules* (рис. 1.16). Для ввода разработанных ранее правил НЭС (см. табл. 1.3, 1.4, п. 2.2) в редакторе правил нужно выбрать соответствующую комбинацию лингвистических термов входных и выходных переменных, выбрать тип их логической связки (*И* или *ИЛИ*) и нажать кнопу *Add Rule*. Если переменная в правиле не используется, то вместо ее терма нужно выбрать *none*. Для удаления правила применяется команда *Delete Rule*, для редактирования – *Edit Rule*.

Под графическим окном главного окна редактора *FIS Matlab* (см. рис. 1.10) расположена область настройки параметров СНВ:

– метод нечеткого логического *И* (*And method*) устанавливает реализацию логического *И*: *MIN* или *PROD*;

– метод нечеткого логического *ИЛИ* (*Or method*) устанавливает реализацию логического *ИЛИ*: *MAX* или *PROBOR*;

– метод импликации (*Implication*) устанавливает реализацию импликации *MIN* (в методе Мамдани) или *PROD* (в методе Ларсена) на этапе активизации заключений;

– метод агрегирования (*Aggregation*) – устанавливает в нечетком выводе Мамдани следующие типы агрегации: *MAX*, *SUM*, *PROBOR*

– метод дефаззификации (*Defuzzification*) – для системы Мамдани используются методы: *centoid* – центр тяжести, *bi*sector – медиана, *lom* – наибольший из максимумов.

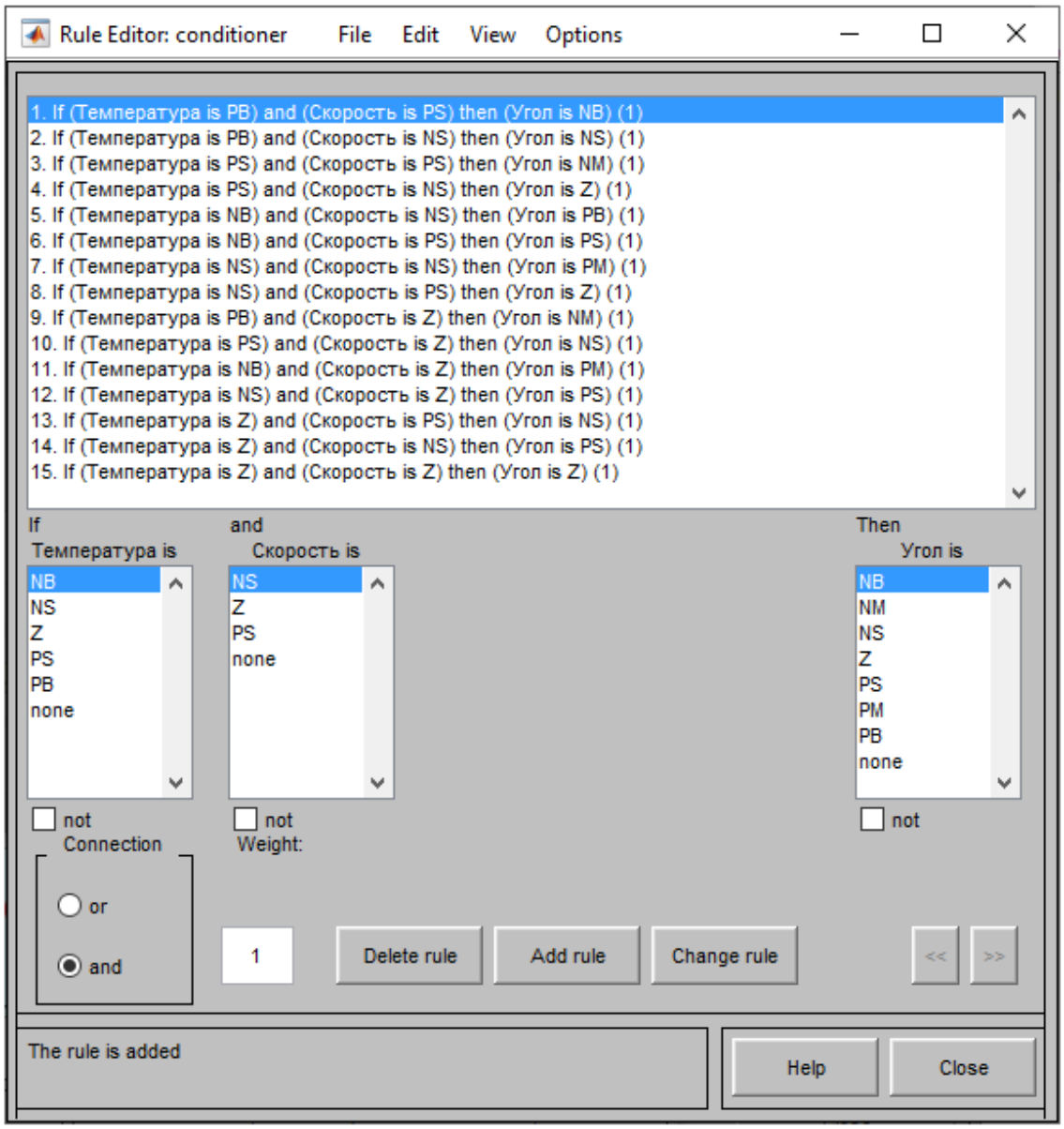


Рис. 1.16. Правила в редакторе *Rule Editor* для СНВ *Conditioner*

3.3. Просмотр результатов работы СНВ

Командой *View/Rules* запускается редактор просмотра срабатывания правил *Rule Viewer* построенной СНВ. Редактор позволяет проводить оценку работы СНВ в режиме «*что-если*»: в поле *Input* или сдвиганием положения красной линии-курсора устанавливаются значения входных переменных; затем на основе процедуры нечеткого вывода вычисляется значение результирующей переменной.

Например, при инициализации входных значений для системы автоматического управления кондиционером в помещении, когда текущая температура воздуха в помещении равна 19,5 °*C*, а скорость ее изменения положительная и составляет 0,2°*C/мин*, процедура нечеткого вывода возвращает значение выходной переменной «угол», равное –19,7° (рис. 1.17).

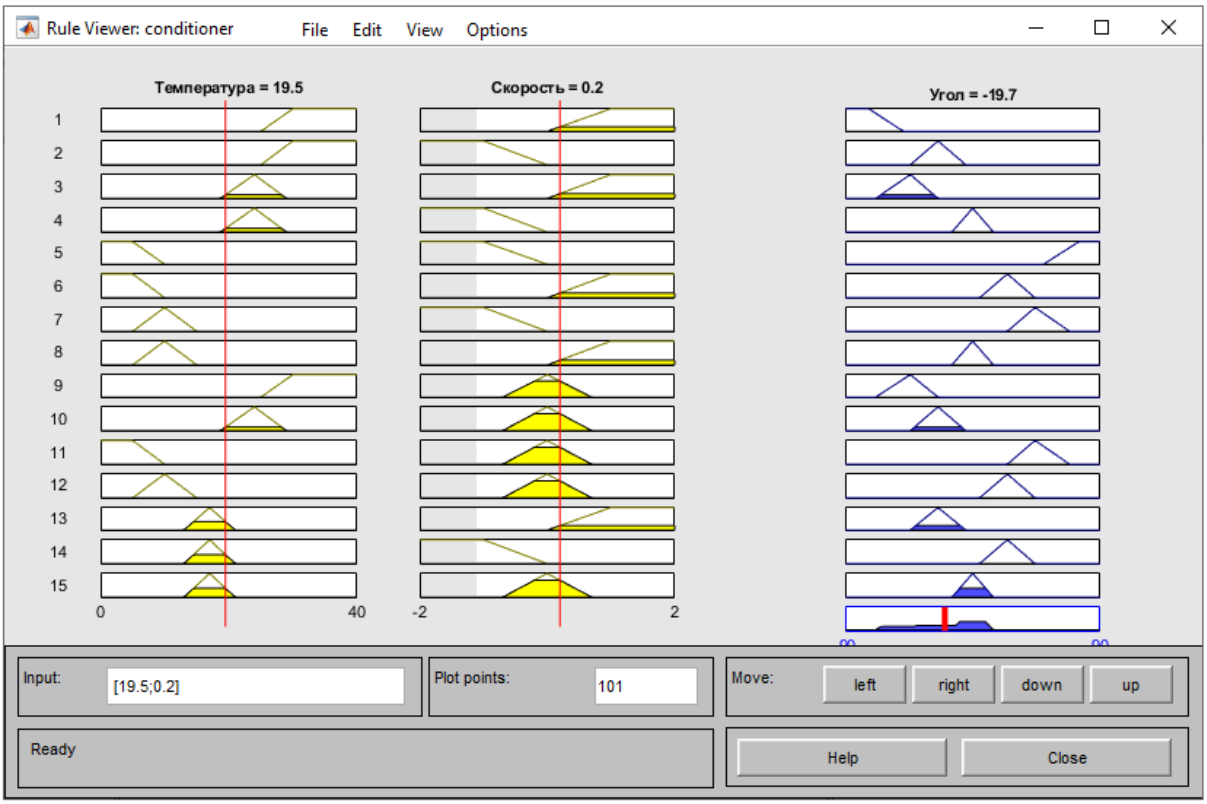


Рис. 1.17. Результаты работы СНВ в редакторе *Rule Viewer* для СНВ *Conditioner*

Сравнение результатов нечеткого вывода для этих значений входных переменных, полученные на основе численных расчетов (≈20° влево) и с помощью разработанной нечеткой модели *Matlab*, показывает согласованность модели.

Для проверки адекватности модели необходим процесс исследования, включающего группу экспериментов с выполнением нечеткого вывода для различных значений входных переменных и оценки полученных результатов.

Визуальная проверка адекватности СНВ осуществляется с помощью программы просмотра *Surface Viewer* (рис. 1.18), запускаемой командой *View/Surface*.

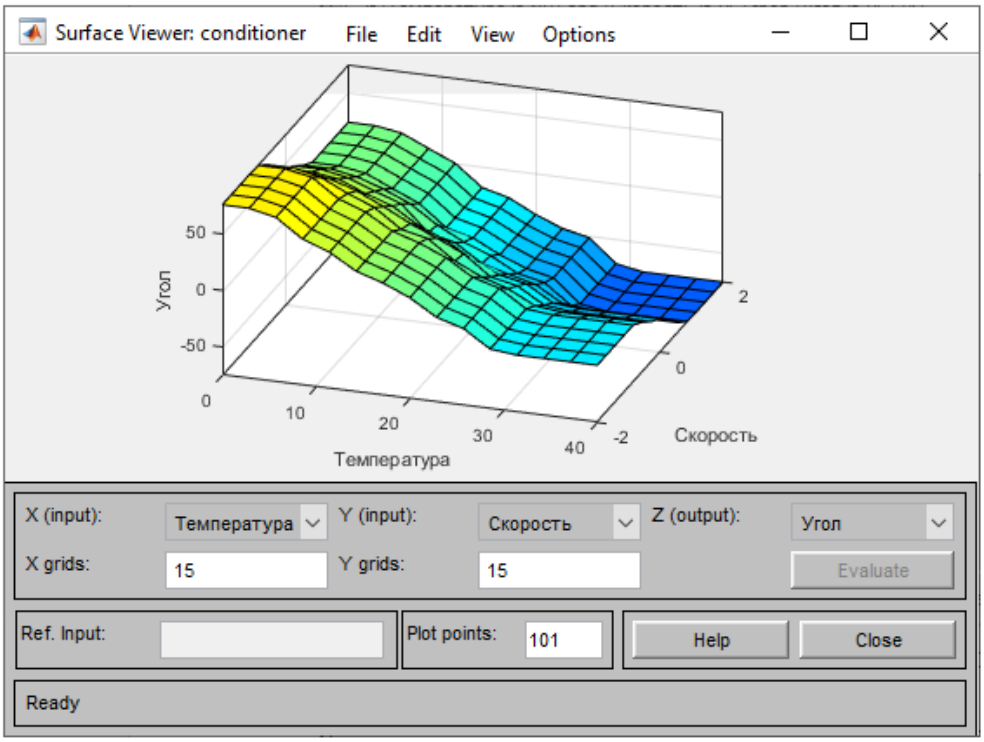


Рис. 1.18. Визуализация поверхности СНВ *Conditioner*

Для анализа разработанной нечеткой модели полезной является визуализация трехмерной поверхности нечеткого вывода, которая позволяет установить зависимость значений выходной переменной от значений входных переменных и служить основой для внесения изменений в СНВ.

3.4. Построение подсистемы нечеткого регулирования системы управления в *Simulink Matlab*

При проектировании интеллектуальных систем управления нечеткие модели встраивают в контур сложных систем управления: например, при построении моделей динамических систем в пакете *Simulink Matlab* управляющими подсистемами выступают нечеткие регуляторы (контроллеры) верхнего уровня, представленные в виде *FIS*-модели.

На рис. 1.19 изображена подсистема нечеткого регулирования системы управления кондиционером *Conditioner* в среде структурного моделирования *Simulink Matlab*.

Входами модели являются структурные элементы *Температура* и *Скорость изменения температуры* вида *Constant,* которые через мультиплексор *MUX* подаются на блок *Fuzzy Logic Controller* с именем *Fuzzy регулятор*.

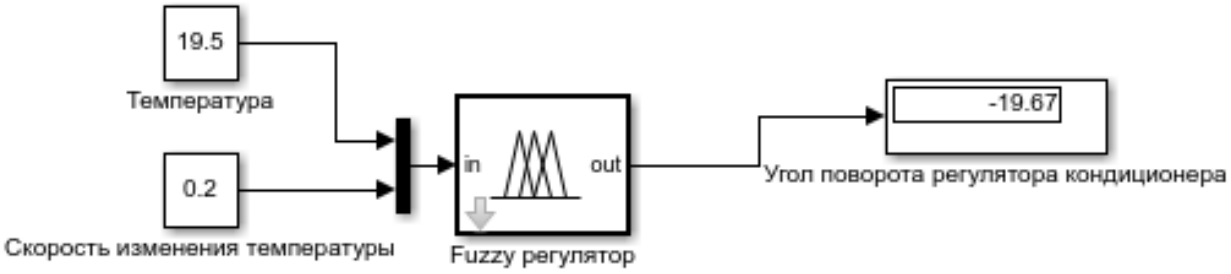


Рис. 1.19. Подсистема нечеткого регулирования системы управления кондиционером в *Simulink Matlab*

В окне параметров блока *Fuzzy Logic Controller* указывается имя разработанной ранее СНВ *Conditioner* как **.fis*-файла:

При запуске модели командой *Run* в блоке *Fuzzy регулятор* осуществляется нечеткий логический вывод на основе информации, поступающей с датчиков о температуре и скорости изменения температуры, и определяется результирующая переменная *Угол поворота регулятора кондиционера,* которая из *FIS*-файла передается на элемент *Display*.

Следует отметить, что динамическая модель может не сработать, если *FIS*-файл не будет предварительно загружен в рабочую область *WorkSpace Matlab*. Для автоматической загрузки файла в рабочую область необходимо в окне динамической модели из контекстного меню выбрать пункт *Model Properties*, в открывшемся окне выбрать вкладку *CallBacks* и в поле *Model PreloadFcn* ввести команду *conditioner=readfis('conditioner.fis')*, как показано на рис. 1.20. Эта команда каждый раз при открытии файла модели будет помещать файл *'conditioner.fis'* в рабочее пространство *WorkSpace Matlab*.

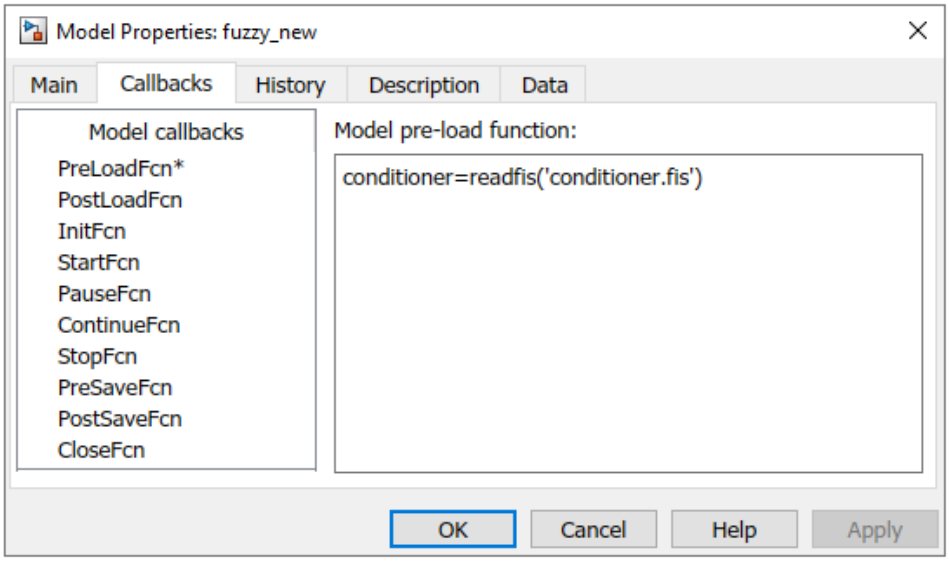


Рис. 1.20. Экранная форма со свойствами динамической модели

Следует отметить, что для настройки подсистемы нечеткого регулирования необходим пересмотр используемых лингвистических значений, а также правил и методов логического вывода.

Порядок выполнения работы

1. Изучить методику построения нечетких экспертных систем в системе *FIS Matlab*.

2. Провести этап идентификации задачи проектирования НЭС:

– сформулировать цели и задачи построения НЭС;

– разработать дерево целей с целевыми переменными на 1–2 уровнях и общим количеством переменных, равным 3-5;

– указать возможные значения каждой переменной в виде термов; количество термов – от 3 до 5;

– определить предварительную структуру правил базы знаний.

3. С использованием пакета расширения *Fuzzy Logic Toolbox* системы *Matlab* в интерактивном режиме разработки экспертных систем с использованием *GUI*-модулей *Fuzzy Inference System* (*FIS*) спроектировать нечеткую экспертную систему как СНВ.

3.1. Описать входные и выходные переменные как лингвистические переменные; построить функции принадлежности для каждого терма.

3.2. Разработать нечеткие правила базы знаний НЭС, общее количество правил должно быть не менее 10. Задать в параметрах СНВ метод нечеткого логического вывода и метод дефаззификации.

4. Оценить адекватность построенной модели нечеткой экспертной системы на основе визуальной проверки поверхности СНВ и по результатам экспериментальных исследований.

5. Провести экспериментальные исследования построенной модели НЭС для одной комбинации точных исходных значений входных переменных x^* при которой срабатывают несколько правил одновременно, выполняя нечеткий логический вывод для них. Результаты экспериментов занести в табл. 1.5.

Провести сравнительный анализ результатов и сделать выводы.

6. Подтвердить полученный результат выполнением вручную процедуры нечеткого логического вывода на основе алгоритмов Мамдани или Ларсена, построением графика результирующей функции принадлежности для одного эксперимента и определением точного значения результирующей переменной y^*.

Таблица 1.5

Результаты экспериментальных исследований

№	Метод нечеткого логического вывода	Метод дефаззификации	Результаты y^*
Исходные данные: значения входных переменных x_i^*			
1	Алгоритм Мамдани	Метод центра тяжести	
2		Левого максимума	
3		Правого максимума	
4	Алгоритм Ларсена	Метод центра тяжести	
5		Левого максимума	
6		Правого максимума	

7. Построить модель фрагмента нечеткой системы управления в *Simulink Matlab*. Провести экспериментальные исследования и убедиться в правильности работы модели.

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

1. Название и цель работы.
2. Цели и задачи построения нечеткой ЭС для заданной предметной области; дерево целей с указанием термов.
3. Графики функций принадлежности входных и выходных лингвистических переменных.
4. Список нечетких правил базы знаний НЭС.

5. Результаты оценки адекватности построенной модели НЭС (график поверхности СНВ, результаты нечеткого вывода с различными входными данными и выходными результатами).

6. Результаты экспериментальных исследований построенной модели НЭС для одной комбинации входных переменных и различных методов нечеткого логического вывода и метода дефаззификации. Выводы, сделанные по результатам сравнительного анализа исследований.

7. Выполненная вручную процедура нечеткого логического вывода на основе алгоритмов Мамдани или Ларсена.

8. Выводы.

Варианты заданий

Варианты заданий могут быть предложены студентами самостоятельно или выбраны из табл. 1.6, содержащей основные типы решаемых ЭС задач и примеры ЭС.

Таблица 1.6

Тип задачи и ее характеристика	Пример ЭС
1	*2*
Интерпретации данных – определение смысла данных (например, результатов исследования)	ЭС интерпретации геофизических данных ЭС определения основных свойств личности по результатам психодиагностического тестирования
Диагностики – соотнесение объекта с некоторым классом объектов и/или обнаружение неисправностей в системе	ЭС диагностики неисправностей, дефектов технического оборудования ЭС финансово-хозяйственного состояния предприятия ЭС диагностики ошибок в аппаратуре и математическом обеспечении ЭВМ
Мониторинга – непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы	ЭС предупреждения опасных ситуаций на производственном участке ЭС экологического мониторинга ЭС мониторинга состояния здоровья человека ЭС информационного мониторинга технологических процессов

Окончание табл. 1.6

1	*2*
Проектирование – создание ранее не существовавшего объекта и подготовка спецификаций на создание объектов с заранее определенными характеристиками	ЭС проектирования трубопроводов ЭС инвестиционного проектирования ЭС синтеза электрических цепей ЭС проектирования конфигураций ЭВМ
Прогнозирование – предсказание последствий некоторых событий или явлений на основе анализа имеющихся данных	ЭС прогноза в экономике ЭС прогноза состояния конструкции под влиянием внешних воздействий ЭС прогноза оценки будущего урожая
Планирование – построение планов действий субъектов (объектов), способных выполнять некоторые функции	ЭС планирования работ и ресурсов ЭС планирования маршрутов ЭС планирование поведения робота ЭС планирование эксперимента
Обучение – использование компьютера для обучения какой-либо дисциплине или предмету.	ЭС диагностики ошибок и подсказывания правильных решений; ЭС обучения языку программирования
Управление – функция организованной системы, поддерживающая определенный режим ее деятельности	ЭС помощи в управлении газовой котельной. ЭС управления системой календарного планирования
Поддержка принятия решений – совокупность процедур, обеспечивающая лицо, принимающее решения, рекомендациями, облегчающими процесс принятия решения	ЭС выбора стратегии выхода фирмы из кризисной ситуации. ЭС помощи в выборе инвестора ЭС для интеллектуальной поддержки оператора при управлении техническим объектом

Контрольные вопросы и задания

1. Какие программные модули входят в пакет *Fuzzy Logic Toolbox*?

2. Какое предназначение редактора систем нечёткого вывода?

3. Какие установки можно выполнить в окне редактора функций принадлежности этом окне?

4. Каким образом задаются правила работы системы нечеткого вывода?

5. Как осуществляется проверка адекватности нечеткой экспертной системы?

Список литературы

1. Пегат А. Нечеткое моделирование и управление / А. Пегат; под редакцией Ю. В. Тюменцева; перевод с английского А. Г. Подвесовского, Ю. В. Тюменцева. 3-е изд. М.: Лаборатория знаний, 2015. 801 с.
2. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. СПб.: БХВ-Петербург, 2003. 736 с.
3. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. М.: Горячая линия-Телеком, 2004. 452 с.
4. Ярушкина Н. Г. Основы теории нечетких и гибридных систем: учеб. пособие. М.: Финансы и статистика, 2004. 320 с.
5. Флегонтов А. В. Моделирование задач принятия решений при нечетких исходных данных: монография / А. В. Флегонтов, В. Б. Вилков, А. К. Черных. СПб.: Лань, 2020. 332 с.
6. Чернов В. Г. Нечеткие множества. Основы теории и применения: учеб. пособие / В. Г. Чернов; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. Владимир: Изд-во ВлГУ, 2018. 156 с.
7. Рыбина, Г.В. Основы построения интеллектуальных систем: учеб. пособие / Г.В. Рыбина. М.: Финансы и статистика, 2010. 432 с.
8. Васильев В. И., Ильясов Б. Г. Интеллектуальные системы управления. Теория и практика: учебное пособие. М.: Радиотехника, 2009. 392 с.

Лабораторная работа № 2

НЕЙРОСЕТЕВОЕ МОДЕЛИРОВАНИЕ В АНАЛИТИЧЕСКОЙ ПЛАТФОРМЕ *DEDUCTOR*. РЕШЕНИЕ ЗАДАЧ КЛАССИФИКАЦИИ, АППРОКСИМАЦИИ ФУНКЦИИ И ПРОГНОЗИРОВАНИЯ

1. Цель и задачи работы

Целью работы является изучение особенностей применения нейронных сетей для решения задач классификации, аппроксимации функции и прогнозирования на основе временных рядов в аналитической платформе *Deductor*.

Задачами работы являются: формирование умений подготовки обучающей выборки, разработки структуры нейронных сетей, закрепление навыков обучения искусственных нейронных сетей на основе парадигмы обучения с учителем.

2. Теоретические сведения

2.1. Обучение искусственных нейронных сетей

Под искусственными нейронными сетями понимаются вычислительные системы, которые преобразуют информацию по образу биологических процессов, протекающих в нервной системе человека и животных. Единицей обработки информации в нейронной сети является *нейрон*.

На рис. 2.1 показана модель искусственного нейрона.

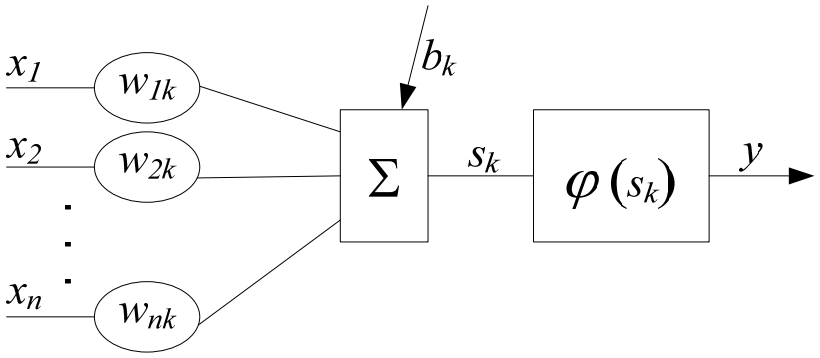


Рис. 2.1. Модель искусственного нейрона

В модели искусственного нейрона выделяют следующие элементы:

– вектор *входных* сигналов $X = \{x_1, x_2, \dots, x_n\}$, поступающих от других нейронов;

– вектор коэффициентов *синаптических связей* (*weight*) между нейронами $W = \{w_{1k}, w_{2k}, \dots, w_{nk}\}$;

– *сумматор* s_k, который складывает входные сигналы *X*, взвешенные относительно соответствующих синапсов *W* нейрона; на выходе сумматора формируется линейная комбинация входных воздействий;

– *функция активации* φ(·), ограничивающая амплитуду выходного сигнала нейрона;

– *выходной* сигнал нейрона y_k;

– *пороговый* элемент b_k (*bias*), который является внешним параметром искусственного нейрона.

Математически функционирование *k*-го искусственного нейрона описывается парой уравнений:

$$s_k = \sum_{i=1}^{n} w_{ik} \cdot x_i;$$
$$y_k = \varphi(s_k + b_k). \qquad (2.1)$$

Формальное представление нейрона демонстрирует *двухтактную* работу нейрона: на первом такте вычисляется скалярное произведение векторов *X* и *W*; на втором – рассчитывается выходной сигнал нейрона *Y* в соответствии с заданными порогом и функцией активации.

Функции активации φ(·), которые также называют нелинейным преобразователем, определяют выходной сигнал нейрона: в результате нормализованный диапазон амплитуд выхода нейрона должен находиться в интервале $[0, 1]$ или $[-1, 1]$. Выделяют следующие типы функций активации:

– *функция единичного скачка, или пороговая функция*, которая используется в модели формального нейрона У. МакКалока и У. Питтса для обработки двоичных сигналов: недостатком этой функции является то, что функция не позволяет моделировать сети с непрерывными сигналами:

$$y = \varphi(s_k) = \begin{cases} 1, & \text{если } s_k \geq \Theta \\ 0, & \text{если } s_k < \Theta \end{cases};$$

– *сигмоидальная функция*, которая часто применяется для многослойных персептронов и других нейронных сетей с непрерывными сигналами:

$$y = \frac{1}{1 + e^{-a \cdot s_k}}.$$

Формальные нейроны объединяются, образуя искусственные нейронные сети (ИНС). Нейросети – множество искусственных нейронов, соединенных определенным образом между собой синаптическими соединениями. ИНС может рассматриваться как направленный граф со взвешенными связями, в котором искусственные нейроны являются узлами. Как правило, активационные функции всех нейронов ИНС фиксированы, а синаптические веса являются параметрами ИНС и могут изменяться.

В общем случае выделяют три фундаментальных класса нейросетевых архитектур:

1) *однослойные* ИНС прямого распространения, состоящие из входного и выходного слоя; под единственным (выходным) слоем подразумевается слой вычислительных элементов (нейронов);

2) *многослойные* ИНС прямого распространения, имеющие несколько *скрытых слоев нейронов*, которые не являются частью входа или выхода сети. Нейроны скрытого слоя позволяют ИНС обучаться решению сложных задач, последовательно извлекая важные признаки из входного вектора;

3) *рекуррентные* ИНС – обладающие, по крайней мере, одной обратной связью.

Важнейшим свойством нейросетей является их способность обучаться на основе априорной информации (фактов), наблюдений, измерений (примеров). Под *обучением* понимается процесс, в котором параметры ИНС (весовые коэффициенты связей между нейронами, пороговые уровни и др.) в соответствии с тем или иным алгоритмом настраиваются при предъявлении *обучающей выборки*.

Обучающая выборка представляет собой информацию об исследуемом объекте, процессе или явлении; выборка может содержать как пары входных и соответствующих им выходных значений (*X, Y*), так и лишь входные значения *X*.

На основе обучающей выборки нейронная сеть в соответствии с заданным алгоритмом корректирует весовые коэффициенты

синаптических связей или другие параметры для того, чтобы обеспечивать требуемую реакцию *Y* на входные образы *X*.

Выделяют три парадигмы обучения: «с учителем», «без учителя» (самообучение) и смешанная.

В случае обучения «*с учителем*» (*supervised learning*) роль учителя рассматривается как наличие знаний об окружающей среде, представленных в виде пар входных и выходных значений (*X, Y*). При подаче обучающего вектора на вход сети учитель передает нейросети желаемый отклик, соответствующий данному входному вектору. Параметры ИНС корректируются с учетом обучающего вектора и сигнала ошибки (разности между желаемым сигналом и текущим откликом сети).

Обучение *«без учителя»* (*unsupervised learning*) не требует знания правильных ответов для каждого примера обучающей выборки. В этом случае решается задача раскрытия внутренней структуры данных или нахождения корреляций между примерами в данных, что позволяет распределить входные примеры по группам.

При *смешанном* обучении часть синаптических весов определяется при обучении «с учителем», в то время как остальная формируется с помощью самообучения.

2.2. Обучение нейронных сетей «с учителем»

Форма обучения ИНС «с учителем» рассматривается как задача минимизации эмпирической ошибки между желаемым выходом и фактической реакцией нейросети и называется обучением на основе коррекции ошибок.

Обучение ИНС проводится на основе *обучающей выборки*, представляющей совокупность векторов входных и желаемых выходных значений. Вектор входных сигналов имеет вид: $X = [x_1, x_2, \dots, x_m]^p$, где m – число признаков, p – количество примеров обучающего множества. Выходной вектор желаемого отклика сети $D = [d_1, d_2, \dots, d_r]^p$ имеет размерность r, соответствующую количеству выходных координат p.

На рис. 2.2 представлена структура *многослойного персептрона* – многослойной нейронной сети прямого распространения с последовательными связями, в которых входной сигнал распространяется по сети от слоя к слою. Количество слоев нейросети равно *N*; в каждом слое разное число нейронов n_k,

$k = 1, \ldots, N$. На вход нейросети подаются внешние сигналы $x_i, i = \overline{1, n_0}$. Все выходы нейронов k-го слоя подаются на каждый нейрон (k+1)-го слоя.

В обозначении выходных сигналов нейронов y_i^k верхний индекс k – номер слоя, а нижний индекс i – номер нейрона в этом слое. Весовой коэффициент синаптической связи w_{ij}^k соединяет i-й нейрон (k –1)-го слоя с j-м нейроном слоя (k).

В многослойных сетях желаемые выходные значения нейронов всех слоев, кроме последнего, неизвестны, и поэтому трех- или более слойный персептрон невозможно обучить, руководствуясь только величинами ошибок на выходах ИНС.

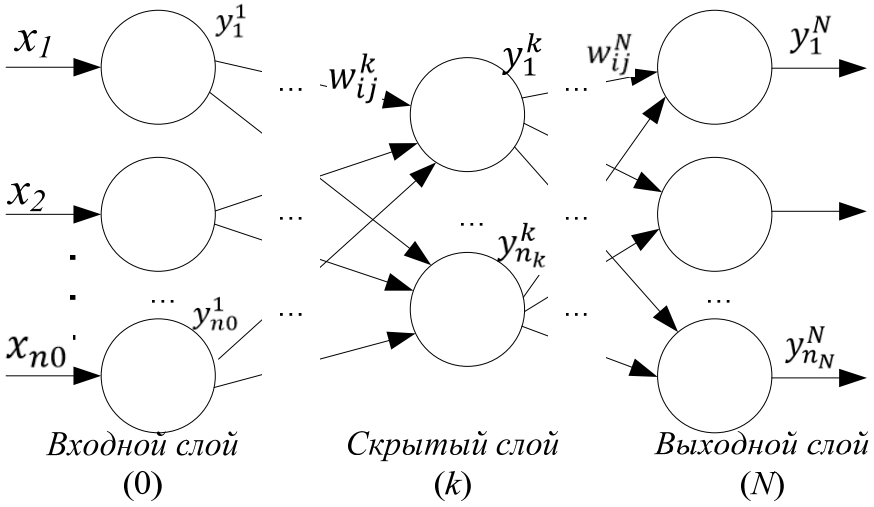


Рис. 2.2. Структура многослойного персептрона

В многослойных ИНС желаемые выходные значения нейронов всех слоев, кроме последнего, неизвестны, и поэтому трех- или более слойный персептрон невозможно обучить, руководствуясь только величинами ошибок на выходах ИНС.

Наиболее известным и чаще всего применяемым для многослойных ИНС алгоритмом обучения с учителем является *метод обратного распространения ошибки* – итеративный градиентный алгоритм обучения, который используется с целью минимизации функции ошибки многослойных ИНС, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Функция ошибки представляет собой сумму квадратов рассогласования (ошибки) желаемого и фактического выхода ИНС.

При вычислении элементов вектора-градиента используется активационная функция сигмоидального типа.

Алгоритм обучения с учителем работает итеративно, и его циклы принято называть *эпохами*. Каждая эпоха представляет собой последовательное предъявление всех обучающих примеров в ИНС. Выходные значения сети сравниваются с целевыми значениями и вычисляется суммарная ошибка, значение которой используется для корректировки синаптических весов. Процесс обучения прекращается, либо когда пройдено определенное количество эпох, либо когда ошибка достигнет определенного уровня малости, либо когда ошибка перестанет уменьшаться.

Хорошо обученная ИНС порождает правильный отклик не только для большинства вводимых примеров из набора тестовых данных. Говорят, что ИНС обладает свойствами *обобщения*, если она будет осуществлять корректное отображение выхода даже тогда, когда входные сигналы немного отличаются от примеров, использованных для обучения сети.

Если ИНС обучается на слишком большом количестве примеров, то нейросеть может только запоминать данные обучающей выборки. Такое явление называют избыточным обучением, или переобучением. Если нейросеть «переучена», то она теряет способность к обобщению.

Алгоритм обучения ИНС с учителем

Шаг 1. Инициализация весов ИНС w_{ij} малыми случайными значениями.

Шаг 2. Прямой проход по нейронной сети.

Выбирается очередной обучающий пример (*X, D*) из обучающего множества. Вектор *X* предъявляется на вход нейросети.

Каждый входной нейрон принимает входной сигнал и распространяет его ко всем нейронам следующего (скрытого) слоя, в которых осуществляется суммирование взвешенных входных сигналов и применение функции активации и формируется выходной сигнал:

$$y_j^k = f(\sum_{i=1}^{n_k} w_{ij}^k \cdot x_i^k), \qquad (2.2)$$
$$y_j^k = x_j^{k+1},$$

где n_k – количество нейронов в -м слое. Вычисляются выходы нейронной сети $y_1^N \ldots y_{nN}^N$.

Шаг 3. Обратный проход по нейронной сети.

Рассчитывается суммарная средняя квадратичная ошибка *E* по всем нейронам выходного слоя нейросети как разность между требуемыми (целевым, желаемым *D*) и реальным *Y* выходом нейросети:

$$E = \frac{1}{2}\sum_{j=1}^{n_N}(y_j^N - d_j)^2. \quad (2.3)$$

Веса нейронной сети $w_{ij}^k(t)$ корректируются так, чтобы в дальнейшем минимизировать ошибку. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов по формуле:

$$w_{ij}^k(t+1) = w_{ij}^k(t) - \eta \cdot \frac{\partial E}{\partial w_{ij}^k}, \quad (2.4)$$

где η – коэффициент скорости обучения (величина шага коррекции), $0 < \eta < 1$; *t* – номер итерации обучения.

Шаг 4. Шаги 2–3 повторяются для каждой пары обучающего множества до тех пор, пока ошибка не станет минимальной и веса сети не стабилизируются. Такая сеть считается обученной (натренированной) и готовой к применению.

2.3. Задачи, решаемые с помощью искусственных нейросетей

Искусственные нейронные сети используются в практических приложениях в качестве компонентов системы управления либо модуля принятия решений, передающих результирующий сигнал на другие элементы, не связанные непосредственно с ИНС.

К задачам, решаемым ИНС, относятся следующие:

1. *Классификация и распознавание образов* – задача состоит в указании принадлежности входного образа (примера), представленного вектором признаков *X* (в виде изображения, вектора чисел и т.д.), одному или нескольким предварительно определенным классам *Y*. В процессе обучения ИНС выделяются признаки, отличающие образы друг от друга, которые и составляют основу для принятия решений об отнесении образов к соответствующим классам. Примерами задачи классификации являются задачи распознавания диагностики и т.д.

2. *Кластеризация* – это задача группировки объектов (наблюдений, событий) по данным, описывающим свойства объектов.

Объекты внутри кластера должны быть похожими друг на друга и отличаться от других, которые вошли в другие кластеры. В задачах кластеризации не требуется указание выходной переменной, поэтому эта задача называется классификацией «без учителя», а число кластеров, в которые необходимо сгруппировать данные, может быть любым.

3. *Аппроксимация функций* – задача построения моделей трудно формализуемых зависимостей. Если имеется обучающая выборка представленная парами данных $((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$, которая генерируется неизвестной функцией $y = f(x)$, то задача аппроксимации состоит в нахождении этой неизвестной функции $f(x)$. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования. Одним из самых замечательных свойств ИНС является их способность аппроксимировать и, более того, быть *универсальными аппроксиматорами*. С помощью ИНС можно аппроксимировать сколь угодно точно непрерывные функции многих переменных.

3. *Прогнозирование* – задача состоит в предсказании будущей реакции некоторой системы по ее предшествующему поведению. Обладая информацией о значениях переменной x в моменты времени t_i, предшествующие прогнозированию $\{x(t_1), x(t_2), \dots, x(t_n)\}$, ИНС вырабатывает решение о том, каким будет наиболее вероятное значение переменной x в некоторый будущий момент времени t_{n+1} (задача прогнозирования на временных рядах).

4. *Оптимизация* – задача нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию.

2.4. Особенности работы в аналитической платформе *Deductor*

Для решения перечисленных задач с применением ИНС используются информационные интеллектуальные системы. Одной из известных и широко используемы систем является аналитическая платформа *Deductor* (*BaseGroup Labs*), в которую включены средства проектирования, моделирования и обучения ИНС.

Для загрузки данных в *Deductor* необходимо *импортировать* их из источника данных при помощи *Мастера импорта.* Вначале указываются тип данных, путь к источнику данных. Затем проводится запуск процесса импорта и осуществляется настройка

параметров столбцов получившейся таблицы: указывается назначение столбцов, необходимое для дальнейшей обработки.

Далее запускается *Мастер обработки;* выбирается метод обработки данных – *Нейронная сеть*, который позволяет сконструировать ИНС с заданной структурой, определить ее параметры и обучить с помощью одного из доступных в системе алгоритмов обучения. Настройка и обучение ИНС осуществляется в несколько этапов.

2.4.1. *Настройка назначения столбцов*. На этом этапе указывается, какие столбцы исходной таблицы являются входными, выходными, информационными, непригодными, неиспользуемыми.

Значения *входных столбцов* являются исходными данными для обучения ИНС. Значения *выходных столбцов* являются эталонными в процессе обучения нейросети; они содержат результаты обработки входных данных.

Если в поле *назначение* столбца указывается *информационное*, то этот столбец не используется при обучении ИНС.

Возникают ситуации, когда при импортировании данных столбцам присваивается статус *непригодное.* Этот статус устанавливается автоматически в трех случаях: 1) если столбец с дискретными данными содержит всего одно уникальное значение; 2) если столбец содержит пропущенные значения; 3) если столбец с непрерывными данными имеет нулевую дисперсию. *Непригодный* столбец не может быть использован при построении и работе алгоритма.

Информационное и непригодное столбцы помещаются в результирующий набор в исходном состоянии; а *неиспользуемый* столбец в работе ИНС не участвует и исключается из результирующего набора.

При настройке назначений столбцов, если выбранный столбец содержит непрерывные (числовые) данные, в правой нижней части окна появляется секция «Статистика», в которой отображаются максимальное и минимальное значения данных столбцов, его среднее значение и стандартное отклонение. Если же выбранный столбец содержит дискретные (строковые) данные, то открывается секция «Уникальные значения» со списком уникальных данных в столбце.

2.4.2. *Настройка обучающей выборки.* На данном этапе обучающая выборка разбивается на два множества – обучающее и тестовое (рис. 2.3). Способ разбиения исходного множества данных по умолчанию задан как «случайно». *Обучающее множество* включает записи (примеры), которые будут использоваться в качестве исходных данных для обучения ИНС. *Тестовое множество* используется для проверки результатов обучения ИНС.

Размер обучающего и тестового множеств могут быть изменены пользователем. В поле «*Количество строк (всего)*» отображается общее количество записей в исходной выборке данных, которое может быть задействовано для формирования множеств.

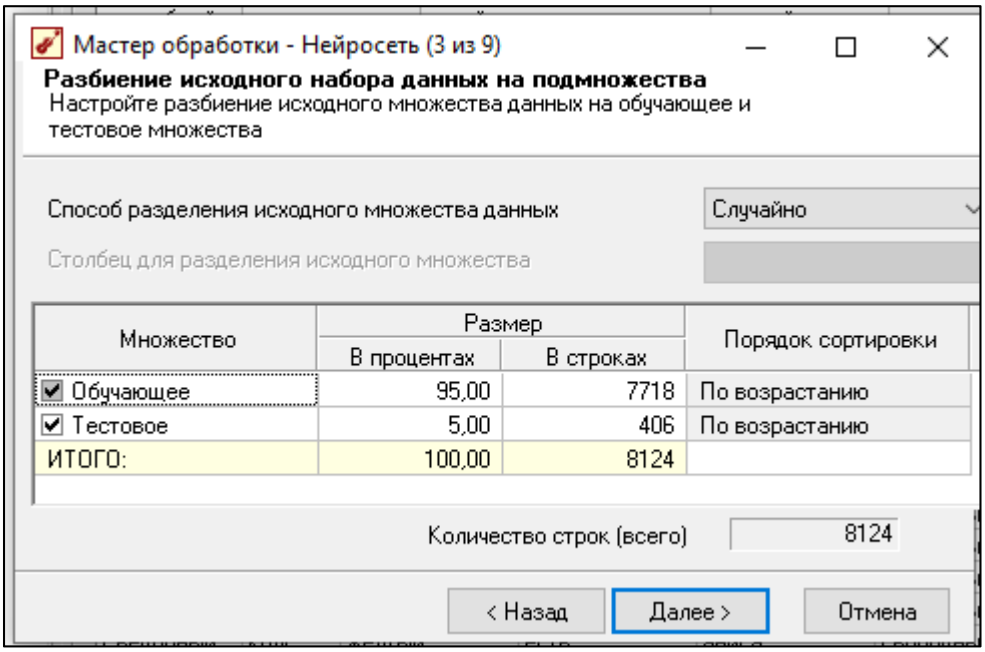


Рис. 2.3. Разбиение исходного набора данных на подмножества

2.4.3. *Настройка структуры нейросети.* На этом этапе задаются параметры, определяющие структуру ИНС, количество скрытых слоев и нейронов в слоях, а также типы активационных функций нейронов.

В разделе «*Нейроны в слоях*» пользователю предлагается указать число скрытых слоев, т.е. слоев ИНС, расположенных между входным и выходным слоями, а также изменить количество нейронов в каждом слое. Число нейронов во входном и выходном слоях устанавливается автоматически в соответствии с числом входных и выходных столбцов обучающей выборки, и изменить его нельзя.

К выбору оптимальной архитектуры нейронной сети, определению количества скрытых слоев и количества нейронов для каждого скрытого слоя нужно подходить осторожно. Хотя до сих пор не выработаны четкие критерии выбора, дать некоторые общие рекомендации все же возможно. Считается, что задачу любой сложности можно решить при помощи двухслойной ИНС, поэтому конфигурация с количеством скрытых слоев больше двух вряд ли оправдана. Для решения многих задач вполне подойдет однослойная ИНС. При выборе количества нейронов следует руководствоваться следующим правилом: «количество связей между нейронами должно быть примерно на порядок меньше количества примеров в обучающем множестве». Количество связей рассчитывается как связь каждого нейрона со всеми нейронами соседних слоев, включая связи на входном и выходном слоях. Слишком большое количество нейронов может привести к «переобучению» сети, когда ИНС выдает хорошие результаты на примерах, входящих в обучающую выборку, но практически не работает на других примерах.

Для определения числа нейронов в скрытых слоях ИНС можно воспользоваться эмпирической формулой для оценки необходимого числа синаптических весов L_w в многослойной сети с сигмоидальными передаточными функциями:

$$\frac{mN}{1+\log_2 N} \le L_w \le m\left(\frac{N}{m}+1\right)(n+m+1)+m, \qquad (2.5)$$

где N – число элементов обучающей выборки; n – размерность входного сигнала; m – размерность выходного сигнала.

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

$$L = \frac{L_w}{n+m}. \qquad (2.6)$$

Известны и другие формулы для оценки, например:

$$2 \cdot (n+L+m) \le N \le 10 \cdot (n+L+m),$$
$$\frac{N}{10} - n - m \le L \le \frac{N}{2} - n - m. \qquad (2.7)$$

В разделе «*Активационная функция*» необходимо определить тип функции активации нейронов и ее крутизну. В нижней части окна отображается график выбранной функции в соответствии с установленной крутизной.

2.4.4. *Настройка параметров обучения нейронной сети*. На этом этапе осуществляется выбор алгоритма обучения и устанавливаются параметры обучения.

Для алгоритма *обратного распространения ошибки* (*Back-Propagation*) задаются:

– *скорость обучения* – коэффициент, который определяет величину шага при итерационной коррекции весов в нейросети (рекомендуется задавать в интервале 0…1);

– *момент* – учитывает величину последнего изменения веса при коррекции весов (задается в интервале 0…1).

Для алгоритма *Resilient Propagation* указываются *шаг спуска* и *шаг подъема* – коэффициенты, которые определяют шаг увеличения и уменьшения скорости обучения соответственно при недостижении либо пропуске алгоритмом оптимального результата.

2.4.5. *Настройка параметров остановки обучения.* Устанавливаются условия, при выполнении которых обучение будет прекращено:

«*Считать пример распознанным, если ошибка меньше*» – критерием останова является условие рассогласования между эталонным и реальным выходом сети меньше заданного значения.

«*По достижении эпохи*» – задается число эпох (циклов обучения), по достижении которого обучение останавливается независимо от величины ошибки. Если флажок сброшен, то обучение будет продолжаться, пока ошибка не станет меньше заданного значения.

Для обучающего и тестового множества в соответствующих секциях окна могут независимо устанавливаться следующие критерии останова обучения:

«Средняя ошибка меньше» – когда средняя квадратичная ошибка на обучающем или тестовом множествах меньше заданного значения.

«Максимальная ошибка меньше» – когда максимальная квадратичная ошибка на обучающем или тестовом множествах меньше заданного значения.

«Распознано примеров (*%*)*»* – когда количество распознанных примеров на обучающем или тестовом множествах больше заданного процента.

При выборе нескольких условий останова процесса обучения происходит по достижении хотя бы одного из них.

2.4.6. *Обучение нейронной сети*. Для управления процессом обучения нейронной сети используются следующие команды:

«*Пуск*» – запуск процесса обучения или возобновление после паузы.

«*Пауза*» – временное приостановление обработки, необходимое либо для оценки текущих результатов процесса обучения, например, просмотра графиков динамики ошибок обучения, либо для освобождения ресурсов процессора для других приложений.

«*Стоп*» – принудительная остановка процесса обучения нейросети, которая имеет смысл, если либо значения ошибок длительное время не уменьшаются, либо процент распознанных примеров не увеличивается.

Обучение может считаться успешным, если процент распознанных примеров на обучающем и тестовом множествах достаточно велик (близок к 100 %).

В процессе обучения в секциях «*Обучающее множество*» и «*Тестовое множество*» отображаются максимальная квадратичная ошибка и средняя квадратичная ошибка на обучающем множестве и тестовом множестве соответственно, а также процент распознанных примеров. Отображаются графики хода обучения для обучающего (синяя линия) и тестового (красная линия) множеств; сплошной линией показана максимальная квадратичная ошибка на обучающем множестве и тестовом множестве, пунктирной – средняя квадратичная ошибка на обучающем множестве и тестовом множестве.

В поле «*Темп обновления*» можно задать число эпох обучения сети, через которое будет происходить обновление графика.

Флажок «*Рестарт*» позволяет включить режим инициализации начальных весов ИНС случайными значениями. Если флажок сброшен, то при повторном запуске обучения после остановки будет иметь место так называемое *дообучение сети*, когда обучение будет начато с текущими весами.

2.4.7. *Определение способа отображения*. На данном этапе устанавливаются необходимые визуализаторы результаты обработки данных, также как:

1. *Обучающий набор* с включением обучающего и тестового множеств

2. *Граф нейросети*. В зависимости от величины значений синаптические веса нейронов на графе отображаются определенным цветом согласно тепловой шкале (высокие значения весов изображаются красным цветом, а низкие значения – синим). По графу можно определить значения весов синаптических связей между нейронами, используя цветовую шкалу, расположенную внизу окна.

3. Анализ по методу «*Что-если*» – визуализатор, показывающий, как будет работать построенная модель при подаче на ее вход тех или иных данных. С помощью данного визуализатора проводится эксперимент, в котором, изменяя значения входных полей обучающей или рабочей выборки ИНС, аналитик наблюдает за изменением значений выхода нейросети.

Таким образом проводится исследование поведения ИНС, которое позволяет оценить достоверность полученных результатов, определить область *устойчивости* системы. Под устойчивостью понимается то, насколько снижается достоверность полученных результатов при попадании на вход системы нетипичных данных – выбросов, пропусков данных и т.д. Очевидно, что если подать на вход значения, существенно выходящие за диапазон входных данных, гарантировать правильную реакцию системы нельзя и достоверность полученных данных может быть снижена. Если значение входных данных выходит за границы диапазона, то это поле окрашивается в красный цвет.

4. *Статистика*, представляющая в табличном виде набор основных статистических характеристик. В верхней части окна статистики отображается общее количество записей в наборе данных. В столбце *Минимум* отображаются минимальные, а в столбце *Максимум* – максимальные значения поля. *Сумма* – столбец сумм всех значений поля.

5. *Таблица сопряженности,* или матрица классификации, – визуализатор, применяемый для оценки качества классификационных моделей. Так, для каждого примера обучающей выборки классификационная модель формирует метку класса, к которому

относится объект по своему набору признаков, указанных в примере. Если сформированная моделью метка класса совпадает с заданной меткой класса примера, то такой пример (объект) является верно распознанным, иначе – неверно распознанным. Соотношение числа верно и неверно распознанных объектов служит критерием качества модели.

6. *Диаграмма рассеяния* служит для визуальной оценки качества обучения нейросетевой модели путем сравнения заданных (по оси *OX*) и рассчитанных обученной нейросетью (по оси *OY*) значений выходного поля для каждого из примеров обучающей выборки. Прямая диагональная линия представляет собой линию идеальных значений. Чем ближе точка к этой линии, тем меньше ошибка модели. Ширина доверительного интервала определяется допустимой ошибкой, которая вводится в поле «*Ошибка*». Если ошибка ИНС (*<Имя_поля>_ERR*) меньше допустимой, то точка попадает в доверительный интервал. С помощью доверительного интервала можно оценить, в каких точках отклонение рассчитанного выхода от эталона является недопустимым.

3. Методика выполнения работы

3.1. Решение задачи классификации «Съедобный ли гриб?».

В качестве обучающего набора данных выступает база данных *Mushroom* (файл «*Грибы.txt*»), включающая записи с описанием характеристик грибов семейства *Agaricus* и *Lepiota*; часть грибов являются съедобными, другая часть – ядовитыми. Характеристиками грибов являются: «*Форма шапки*», «*Поверхность шляпки*», «*Цвет шляпки*», «*Синие пятна*», «*Запах*» и т.д. На основе этих данных необходимо построить нейросетевую модель, которая сможет дать ответ, к какому классу относится гриб. Пользователь должен получить ответ на вопрос «съедобный или ядовитый гриб?». Задача относится к группе задач классификации, т.е. обучения с учителем.

Перед построением нейросетевой модели проводится процедура импортирования (п. 2.4). Необходимо обратить внимание на то, что в процессе импортирования в столбце с номером *класса* нужно указать *дискретный вид данных*.

Затем запускается *Мастер обработки;* выбирается метод обработки данных – *Нейронная сеть*.

На первом этапе построения модели проводится настройка назначения столбцов. На рис. 2.4 показана настройка назначений исходных столбцов данных. Выходной столбец – столбец «Класс» с дискретным типом данных, все остальные столбцы – входные.

На этапе разбиения способ разбиения исходного набора данных на обучающее и тестовое множество оставить заданным по умолчанию.

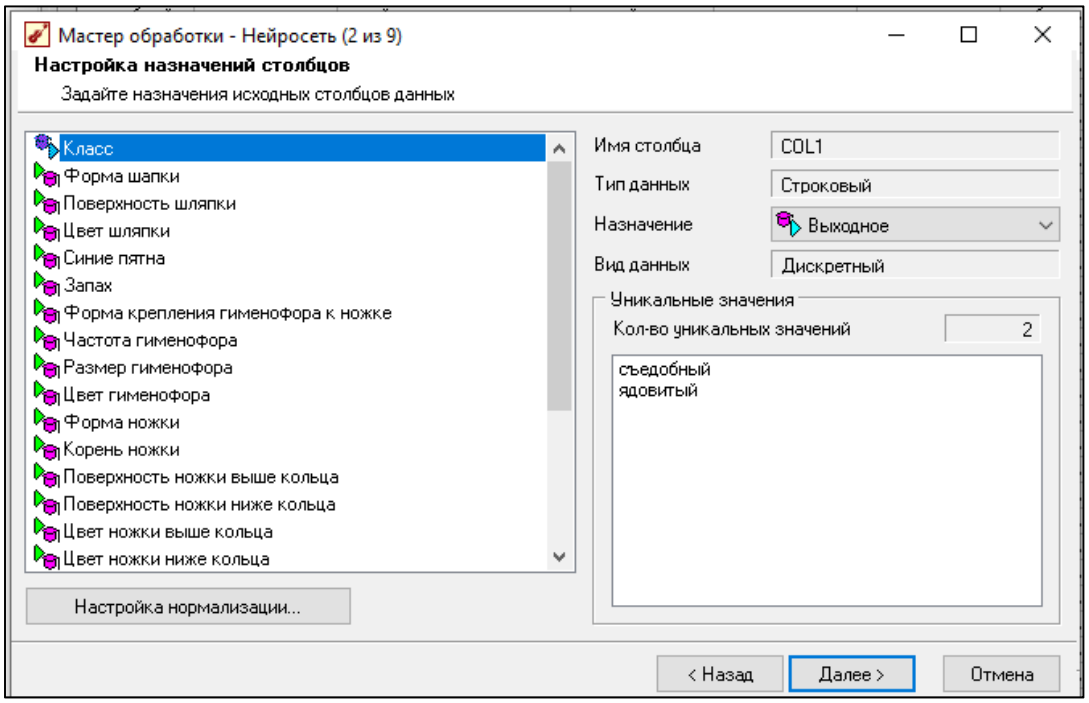


Рис. 2.4. Настройка назначений столбцов

На следующем этапе настраивается структура ИНС (рис. 2.5): количество нейронов входного слоя указывается равным 53 (количество входных переменных), выходного слоя – 1 (количество выходных переменных). Устанавливается 2 скрытых слоя по 4 нейрона в каждом слое, активационная функция – сигмоида с крутизной, равной единице.

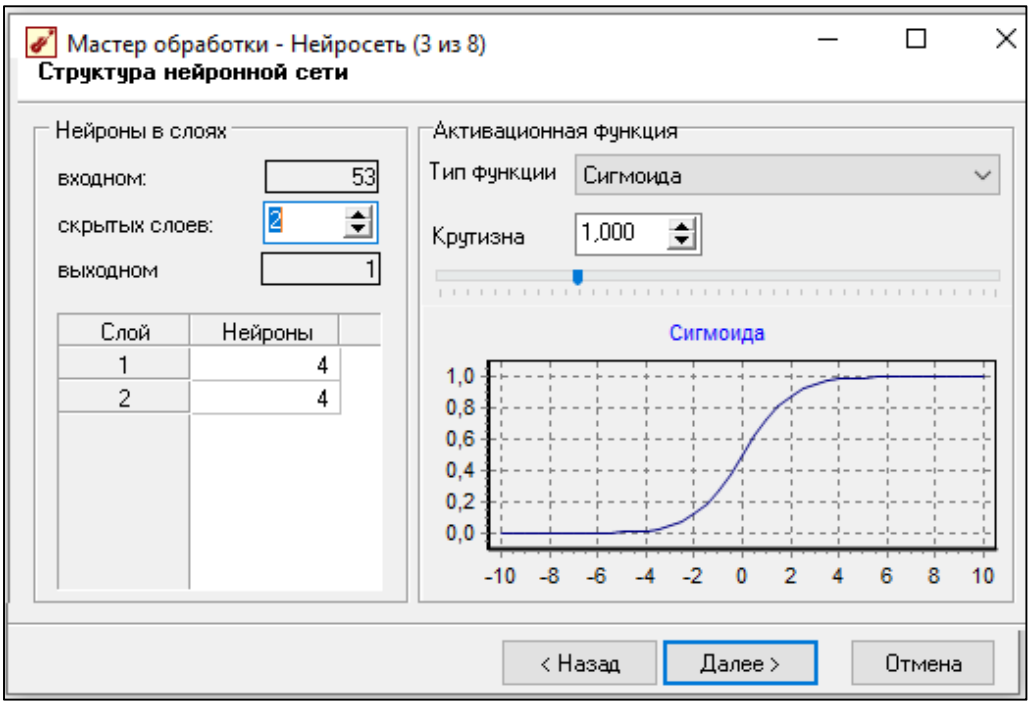


Рис. 2.5. Структура нейронной сети

На следующем этапе проводится настройка параметров обучения нейронной сети (рис. 2.6). В качестве алгоритма обучения выбирается алгоритм обратного распространения ошибки *Back-Propagation*.

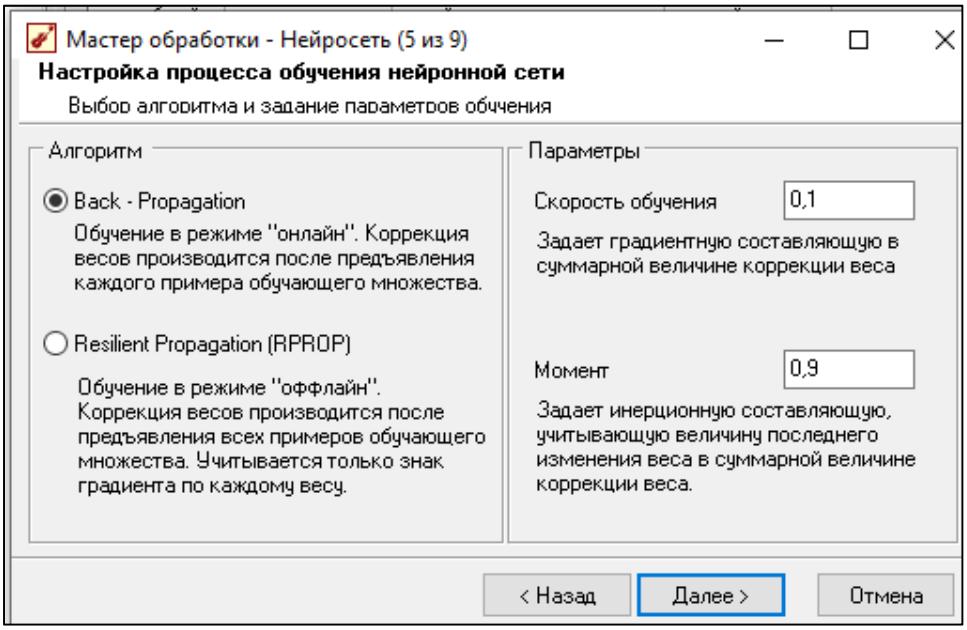


Рис. 2.6. Настройка процесса обучения нейронной сети

При настройке параметров остановки обучения устанавливается, что пример будет считаться распознанным, если ошибка меньше 0,05, а условие остановки обучения – при достижении *эпохи*, равной 1000.

Запускается процесс обучения, фрагмент которого проиллюстрирован на рис. 2.7. Показано, что на *эпохе* №1000 на обучающем и на тестовом множествах распознано 100 % примеров.

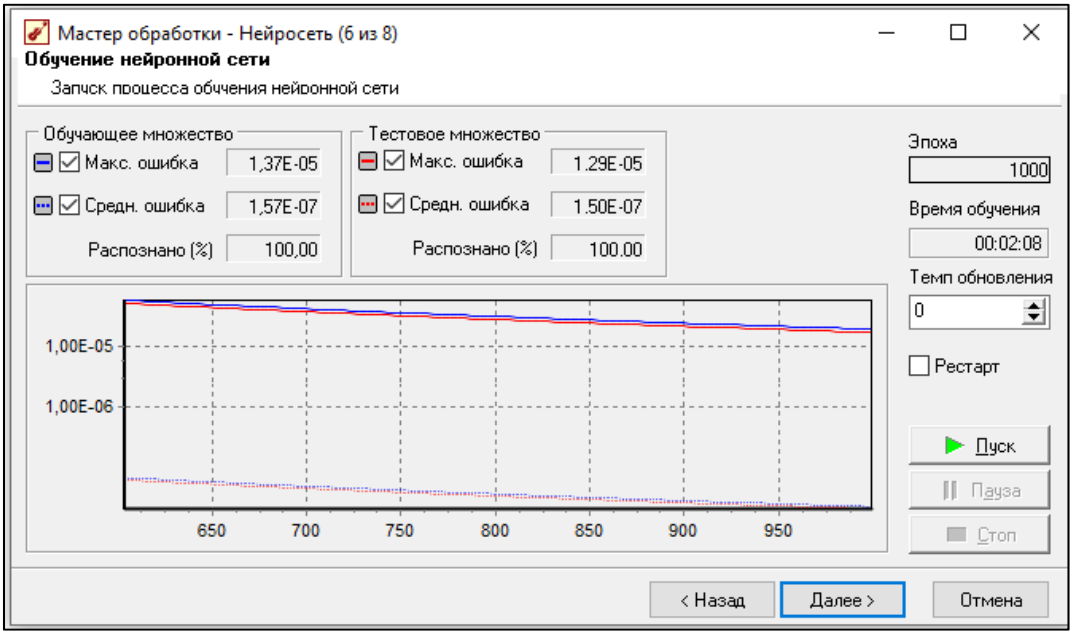


Рис. 2.7. Обучение нейронной сети

При выборе способа отображения обработки данных для решаемой задачи устанавливаются следующие визуализаторы: «Граф нейронной сети», анализ «Что-если», «Таблица сопряженности».

На рис. 2.8 представлен граф нейронной сети, построенной в соответствии с ранее заданной структурой (см. рис. 2.5): входной слой (53 нейрона), два скрытых слоя (по 4 нейрона в каждом слое) и выходной слой (1 нейрон).

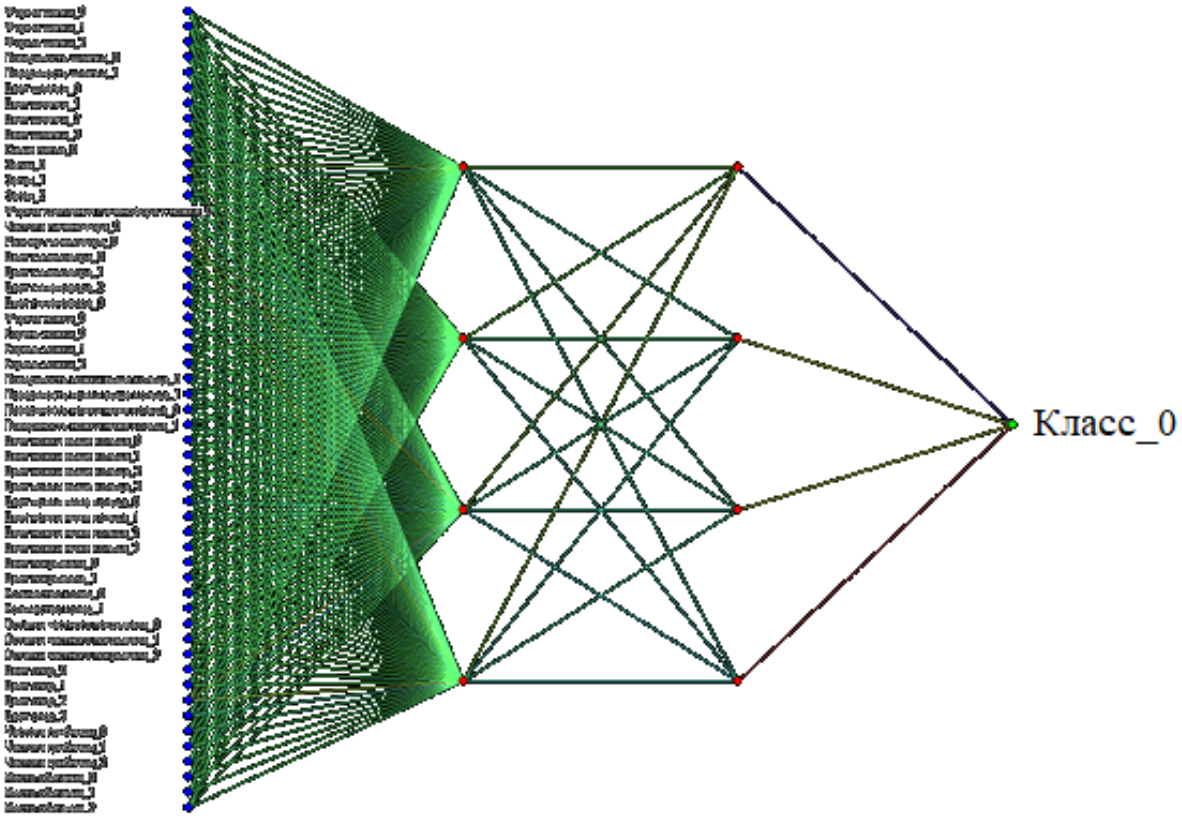


Рис. 2.8. Фрагмент графа нейросети

Качество построенной классификационной модели отражается в *таблице сопряженности*, или матрице классификации (рис. 2.9). В ячейки главной диагонали таблицы сопряженности записываются значения числа примеров, которые были правильно распознаны для разных меток классов: 4208 грибов, которые являются съедобными (класс «съедобный»), и 3916 грибов, которые являются несъедобными (класс «ядовитый»). Неверно классифицированные примеры отсутствуют.

Суммарная оценка качества классификации, содержащая общее соотношение верно и неверно классифицированных примеров (кнопка Σ), представлена на круговой диаграмме.

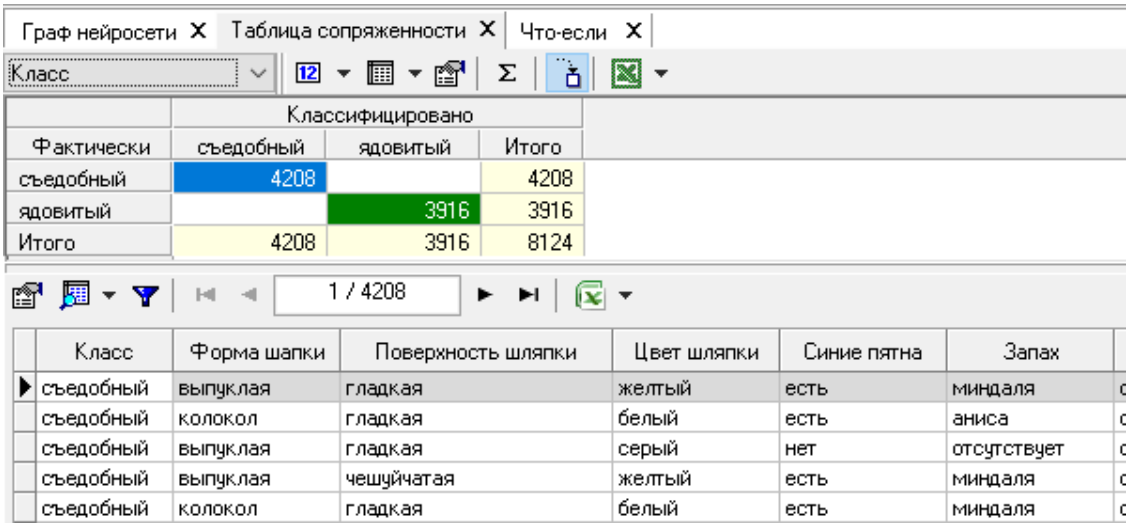


Рис. 2.9. Таблица сопряженности

Исследование построенной классификационной нейросетевой модели проводится в визуализаторе «*что-если*». Исходные данные о некотором объекте вводятся в соответствующие входные поля, а выходное поле рассчитывается.

На рис. 2.10 представлен фрагмент эксперимента, при котором исходные характеристики грибов формируют в нейросетевой модели результаты в виде рекомендаций «съедобный»/«ядовитый».

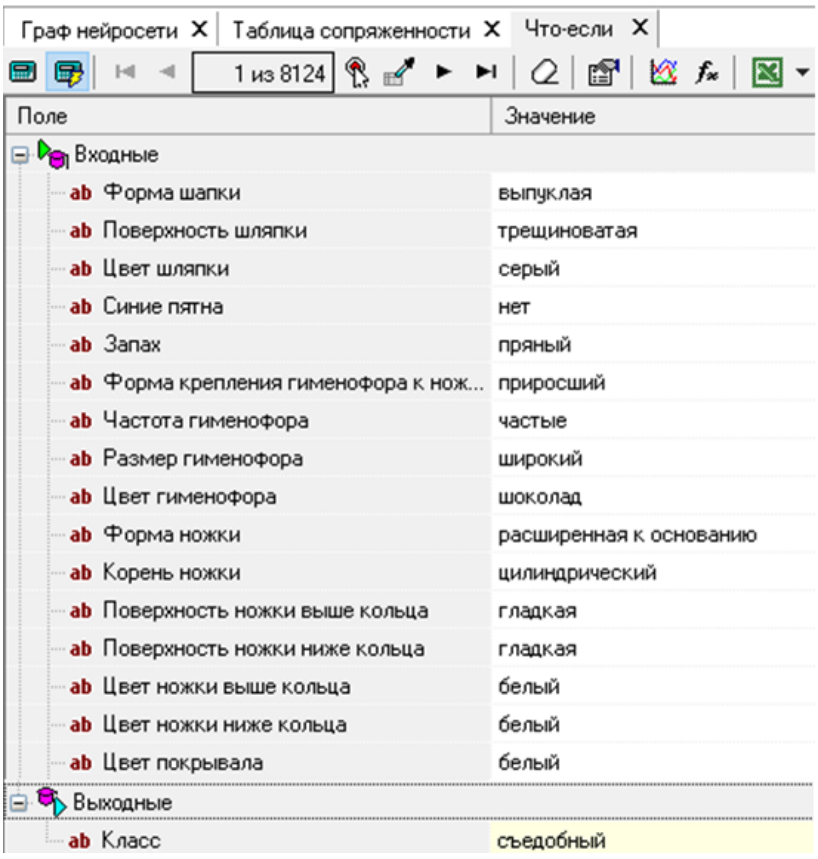


Рис. 2.10. Фрагмент анализа «*что-если*»

Многократное проведение экспериментов с различными исходными данными позволяет, во-первых, выявить значимые и незначимые признаки во входных полях (изменения значений незначимых признаков оказывает очень слабое влияние на результат классификации, либо не влияет на него вовсе), а во-вторых, найти диапазоны изменения набора значений входных признаков, приводящих к тому или иному результату классификации.

Построенная нейросетевая классификационная модель в неявном виде содержит правила классификации объектов к заданным классам, а при появлении нового объекта сформирует решение об отнесении его к тому или иному классу.

3.2. Задача аппроксимации функции

Задача обучения с учителем часто рассматривается как задача аппроксимации экспериментальных данных, состоящая в нахождении функции $f(x)$, наилучшим образом приближающей желаемую функцию $F(x)$. Для иллюстрации решения данной задачи рассматривается пример аппроксимации функции двух переменных:

$$f(x_1, x_2) = (1 - {x_1}^2) + 2 \cdot (1 - x_2)^2, \qquad (2.8)$$

в которой диапазон изменения аргументов функции x_1 и x_2 ограничен интервалом [–1;1].

3.2.1. Вначале для обучения ИНС строится обучающая выборка по заданной функции (2.8). Значения аргументов функции используются как входной вектор, а значения функции – как эталонные значения. В табл. 2.1 приведен фрагмент обучающей выборки.

Таблица 2.1

Фрагмент обучающей выборки

№ п/п	x_1	x_2	$F(x_1, x_2)$	№ п/п	x_1	x_2	$F(x_1, x_2)$
1	-1,0	-1,0	8	11	0,0	0,0	3
2	-0,8	-1,0	8,36	12	0,1	0,0	2,99
3	-1,0	-0,8	6,48	13	0,1	0,2	2,27
4	-0,6	-0,5	5,14	14	0,2	0,3	1,94
5	-0,7	-0,6	5,63	15	0,4	0,5	1,34
6	-0,4	-0,5	5,34	16	0,6	0,3	1,62
7	-0,5	-0,3	4,13	17	0,6	0,6	0,96
8	-0,3	-0,2	3,79	18	0,7	0,8	0,59
9	-0,2	-0,1	3,38	19	0,9	1,0	0,19
10	-0,1	0,0	2,99	20	1,0	1,0	0

3.2.2. Для решения задачи аппроксимации функции обучающая выборка загружается в аналитическую платформу *Deductor*. Запускается *Мастер обработки,* выбирается метод обработки данных – *Нейронная сеть*. При настройке назначения полей x_1, x_2 указываются как входные переменные, а функция $f(x_1, x_2)$ – как выходная переменная.

3.2.3. В процессе настройки структуры ИНС устанавливается число нейронов входного слоя, равное 2 (число входных переменных x_1 и x_2), а также число нейронов выходного слоя, равное 1. Определение структуры ИНС предполагает введение скрытых слоев и указание числа нейронов в каждом из них. Проектирование оптимальной топологии ИНС осуществляется путем *поиска* (перебора) такой архитектуры, которая обеспечивает наилучшее (относительно некоторого выбранного критерия) решение конкретной задачи.

Для установления первоначальных значений параметров ИНС используются формулы (2.5–2.7). Для рассматриваемой обучающей выборки с количеством примеров N=20, числом входов n=2, а числом выходов m=1, необходимо построение ИНС с общим числом синаптических связей между нейронами L_w (2.5) в диапазоне $17 \leq L_w \leq 88$ и общим числом нейронов (2.6) от 6 до 30: например, можно ввести 8 нейронов в двух скрытых слоях, т.е. по 4 нейрона в каждом слое. Граф нейросети представлен на рис. 2.11.

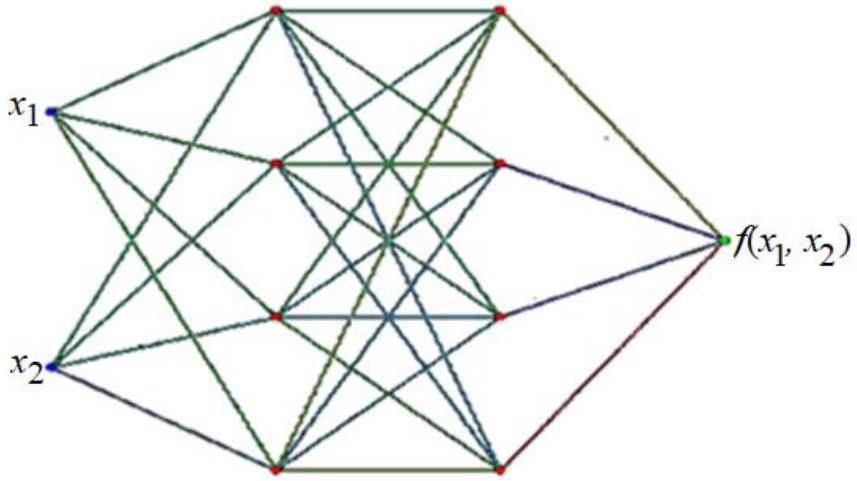


Рис. 2.11. Граф нейросети

3.2.4. На следующем шаге мастера обработки при настройке параметров остановки обучения ИНС устанавливается значение поля «*Считать пример распознанным, если ошибка меньше*», равное 0,05.

3.2.5. Запустить обучение нейросети при решении задачи аппроксимации.

3.2.6. Для оценки качества обучения нейросети с выходной переменной непрерывного типа используется специальный визуализатор «*Работа с диаграммой рассеяния*». Необходимо подчеркнуть, что если выходные переменные имеют дискретный вид, данный визуализатор недоступен, а используется *таблица сопряженности*.

По *диаграмме рассеяния* (рис. 2.12) оценивается качество обучения построенного аппроксиматора: на ней изображены отклонения результатов обработки данных относительно эталонных. Анализ диаграммы показывает, что объекты находятся в границах доверительного интервала и близки к эталонным значениям, однако имеются есть участки, не заполненные данными, в которых невозможно определить поведение аппроксимируемой функции – поэтому для повышения качества нейросетевого аппроксиматора необходимо увеличить количество примеров обучающей выборки.

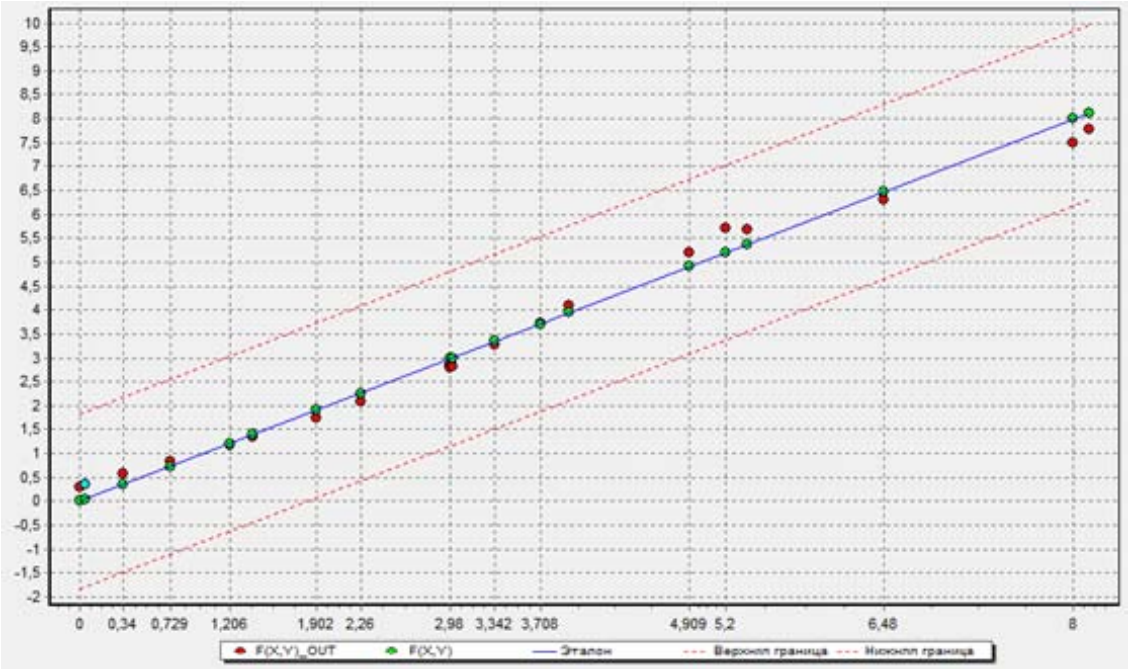


Рис. 2.12. Диаграмма рассеяния

3.2.7. Для оценки результатов аппроксимации проводится опрос ИНС с использованием инструмента «*Что-если*». В табл. 2.2 представлены результаты опроса ИНС $f(x_1, x_2)$, а также их отклонения ε от эталонных(желаемых) значений $F(x_1, x_2)$.

Отметим, что хотя заданные для опроса нейросети пары входных значений x_1 и x_2 были заданы не из обучающей выборки, результаты опроса сети все же очень близки к желаемым значениям, что свидетельствует об аппроксимационной способности нейросети.

Таблица 2.2

Результаты опроса ИНС

	x_1	x_2	«что-если» $f(x_1, x_2)$	$F(x_1, x_2)$	$\varepsilon = f(x_1, x_2) - F(x_1, x_2)$
1	0,3	0,3	1,76	1,89	0,13
2	-0,9	-1	7,99	8,19	0,2
3	-0,2	0,9	1,68	0,98	-0,7
4	0	0	3,007	3	-0,007
5	0,1	-0,4	3,15	4,91	1,76
6	-0,9	-0,3	3,54	3,57	0,03
7	0	0,2	2,36	2,28	-0,08

Однако не все результаты опроса нейронной сети удовлетворяют требованиям точности аппроксимации. Так, в данных, близких к примерам обучающей выборки, например, когда оба значения x_1 и x_2 являются либо большими, либо малыми, ошибка аппроксимации незначительна. Данные, в которых примеры состоят либо из больших значений x_1 и малых значений x_2, либо наоборот, порождают значительное расхождение ε между желаемым и фактическим значениями функции.

Таким образом, можно сделать вывод о том, что для улучшения качества нейросетевого аппроксиматора, т.е. для повышения точности аппроксимации, необходимо хорошее качество обучающей выборки. Кроме того, требуется иметь достаточный объем обучающего множества – например, для ошибки в 10 % количество примеров обучения должно в 10 раз превосходить количество свободных параметров сети (синаптических весов и порогов). И наконец, качество аппроксимации зависит от структуры нейронной сети и параметров обучения.

3.3. Задача прогнозирования

Прогнозирование является одной из самых востребованных задач анализа и управления в сложных системах. Многие задачи прогнозирования связаны с обработкой данных, зависящих от времени, называемых *временными рядами*. Временной ряд состоит из последовательности наблюдений за состоянием признаков исследуемых объектов или процессов.

Если наблюдения содержат один признак, то временной ряд является одномерным $X = \{x_1, x_2, \dots, x_n\}$, а если два или более – многомерным; например, трехмерный временной ряд процесса F можно записать в виде: $F = \{(x_1, y_1, z_1); (x_2, y_2, z_2), \dots, (x_n, y_n, z_n)\}$. Значения временного ряда определены только в фиксированные моменты времени (отсчеты), при этом отсчеты полагаются равноотстоящими друг от друга.

Целью прогнозирования значений временного ряда является *определение значения x* на основе предыдущих значений признака на несколько шагов вперед. Для решения этой задачи необходимо преобразовать (трансформировать) временной ряд методом *скользящего окна* в таблицу, содержащую множество *обучающих примеров*, т.е. представляющую собой обучающую выборку.

Под *окном* понимается временной интервал, содержащий набор значений, который используется для формирования каждого примера; при этом задаются три параметра:

– *интервал прогноза* – временной интервал, на который будет осуществляться прогнозирование: день, неделя, месяц, квартал, год;

– *горизонт прогноза* h – на какое количество интервалов (дней, недель и т.д.) нужно получить прогноз;

– *глубина погружения* d – количество значений интервалов прогноза в прошлом, которое будет использоваться для прогнозирования определенных значений интервалов в будущем.

Важно, что при решении задачи прогнозирования глубина погружения d должна в несколько раз превышать горизонт прогноза h. Чем большее число значений из прошлого используется для прогнозирования, тем выше степень обобщения и тем достовернее результаты предсказания. И чем дальше простирается горизонт прогноза, тем ниже достоверность результатов.

Формирование обучающих примеров основано на параметрах метода скользящего окна d и h: входные значения обучающих

примеров являются значениями временного ряда, начиная с текущего значения и далее в обратном направлении с ретроспективой, равной глубине погружения d: $x(n), x(n-1), \ldots, x(n-d)$, а выходные значения примеров в соответствии с горизонтом планирования h равны $x(n+1) \ldots x(n+h)$. Обобщенная модель прогноза представлена на рис. 2.13.

Временной ряд

№	Значение признака
t(1)	*x*(1)
t(2)	*x*(2)
...	...
t(*n*-2)	*x*(*n*-2)
t(*n*-1)	*x*(*n*-1)
t(*n*)	*x*(*n*)

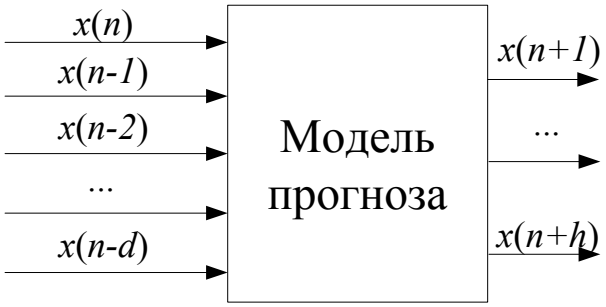


Рис. 2.13. Обобщенная модель прогноза

Для граничных положений окна будут формироваться *неполные* записи, т.е. записи, содержащиеся пустые значения для отсутствующих прошлых или будущих отсчетов.

В табл. 2.3, *а* представлена обучающая выборка с *неполными* записями, полученная путем преобразования временного ряда с n=5 записями, глубиной погружения d=2 и горизонтом прогноза h=1; общее число неполных записей рассчитывается как сумма параметров $(n + d + h)$, и в данном примере равно 8.

Метод скользящего окна позволяет исключить неполные записи из обучающего множества (табл. 2.3, *б*); число записей обучающего множества с *полными* записями равно $(n - d - h)$.

Таблица 2.3,а

Обучающая выборка с неполными записями

№	d_2 *x*(*n*-2)	d_1 *x*(*n*-1)	*x*(*n*)	h_1 *x*(*n*+1)
				x(*1*)
t(*1*)			*x*(*1*)	*x*(2)
t(*2*)		*x*(*1*)	*x*(2)	*x*(*3*)
t(*3*)	*x*(*1*)	*x*(2)	*x*(*3*)	*x*(4)
t(*4*)	*x*(2)	*x*(*3*)	*x*(4)	*x*(*5*)
t(*5*)	*x*(*3*)	*x*(4)	*x*(*5*)	
	x(4)	*x*(*5*)		
	x(*5*)			

Таблица 2.3,б

Обучающая выборка с полными записями

№	d_2 $x(n-2)$	d_1 $x(n-1)$	$x(n)$	h_1 $x(n+1)$
$t(3)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$
$t(4)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$

Пример. Имеются ретроспективные данные о потреблении электроэнергии (база данных «*Потребление электрической энергии*»), включающие 109 записей со следующими характеристиками: *Дата, Объект, Потребление эл. энергии, кВт·час* (табл. 2.4).

Таблица 2.4

Исходный набор данных (временной ряд)

№	*Дата*	*Объект*	*Потребление*
$x(0)$	01.05.2010	Объект 2	185127,5
$x(1)$	01.06.2010	Объект 2	225127,5
$x(2)$	01.07.2010	Объект 2	284265
$x(3)$	01.08.2010	Объект 2	292792,5
..	…	…	…
$x(24)$	01.01.2008	Объект 1	25668
..	…	…	…
$x(109)$	01.04.2012	Объект 3	154102,5

3.3.1. Предварительная обработка и анализ временного ряда.

Файл «*Потребление электрической энергии.txt*» импортируется в аналитическую платформу *Deductor*.

При предварительном просмотре исходного файла можно заметить, что наблюдения о расходе электроэнергии указаны для разных объектов: 1, 2 и 3. Поскольку прогнозирование может проводиться только для одной группы объектов, необходима предварительная процедура преобразования временного ряда с выделением однотипных объектов, т.е. *фильтрация*. Для этого запускается *Мастер обработки*, затем в разделе *Очистка данных* указывается *Фильтрация*; затем в поле *Выберите столбец* устанавливается *Объект=* , а затем из появившегося списка значений данного поля выбирается, например, *Объект 1*. В конце процедуры фильтрации при определении способов отображения указываются визуализаторы «*Таблица*» и «*Диаграмма*»; которые демонстрируют временной ряд изменения потребления электроэнергии по месяцам в табличном и графическом виде для *Объекта 1* (количество объектов равно 49) (рис. 2.14).

Анализ построенной диаграммы показывает, что данные временного ряда содержат аномалии (выбросы) и шумы, из-за которых сложно понять тенденции изменения потребления электроэнергии. С целью *сглаживания* данных для узла *Фильтр*([Объект] = ’Объект 1’) запускается *Мастер обработки*, выбирается вид обработки упорядоченных данных *Спектральная обработка*. В процедуре сглаживания при настройке полей выделяется поле «*Потребление эл. энергии, кВт·час*», устанавливается назначение *Используемое*, а затем для него выбирается *Вычитание шума, Фурье-преобразование* (степень вычитания – *малая*).

Дата	Объект	Потребление эл энергия,кВт час
01.04.2008	Объект 1	24228
01.05.2008	Объект 1	21510
01.06.2008	Объект 1	18513
01.08.2008	Объект 1	20079
01.09.2008	Объект 1	21951
01.10.2008	Объект 1	31212
01.11.2008	Объект 1	32688
01.12.2008	Объект 1	30438
01.01.2009	Объект 1	28764
01.02.2009	Объект 1	27387
01.03.2009	Объект 1	27171
01.04.2009	Объект 1	23037,669
01.06.2009	Объект 1	20361,525
01.07.2009	Объект 1	21972,3
01.08.2009	Объект 1	20659,725
01.09.2009	Объект 1	22602,975
01.10.2009	Объект 1	26337,15
01.11.2009	Объект 1	26279,25
01.12.2009	Объект 1	36532,425
01.01.2010	Объект 1	29880,45

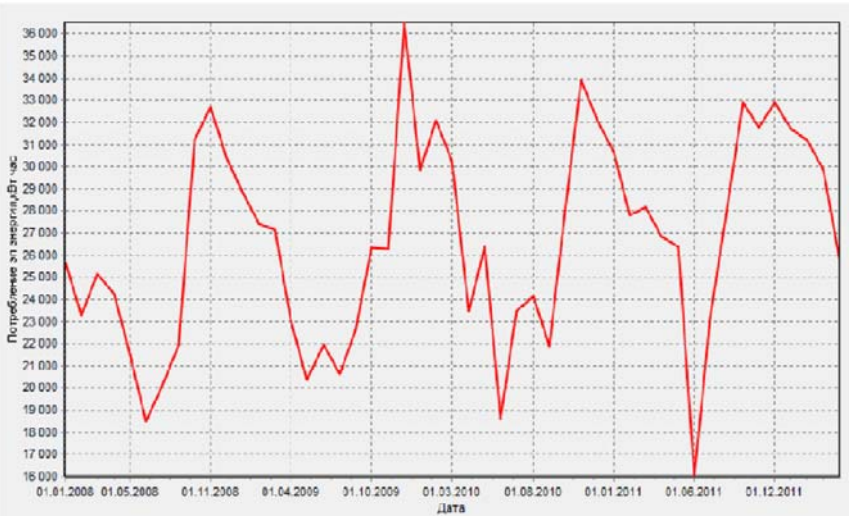


Рис. 2.14. Таблица и диаграмма потребления электроэнергии для *Объекта 1*

Для отображения результатов сглаживания указывается визуализатор *Диаграмма*. На рис. 2.15 представлены таблица с измененными данными и соответствующая диаграмма, на которой исчезли аномалии и шумы. Детальный анализ построенной диаграммы показывает, что в данных имеется наличие годовой сезонности, равное 12 месяцам: в зимнее время наблюдаются пики, а в летнее – спады электрической энергии.

Дата	Объект	Потребление эл энергия,кВт час
01.01.2008	Объект 1	30391,538959252
01.02.2008	Объект 1	29287,4922343733
01.03.2008	Объект 1	27221,3139809438
01.04.2008	Объект 1	24791,1840836275
01.05.2008	Объект 1	22754,3123957082
01.06.2008	Объект 1	21786,3291659132
01.08.2008	Объект 1	22256,1352575945
01.09.2008	Объект 1	24091,492346674
01.10.2008	Объект 1	26783,0349545217
01.11.2008	Объект 1	29530,1546740396
01.12.2008	Объект 1	31485,6782855231
01.01.2009	Объект 1	32022,8137668868
01.02.2009	Объект 1	30938,78037157
01.03.2009	Объект 1	28528,2905795555
01.04.2009	Объект 1	25500,9039562672
01.06.2009	Объект 1	22766,1682528576
01.07.2009	Объект 1	21153,2282595215
01.08.2009	Объект 1	21153,2282595215
01.09.2009	Объект 1	22766,1682528576
01.10.2009	Объект 1	25500,9039562672
01.11.2009	Объект 1	28528,2905795555
01.12.2009	Объект 1	30938,78037157

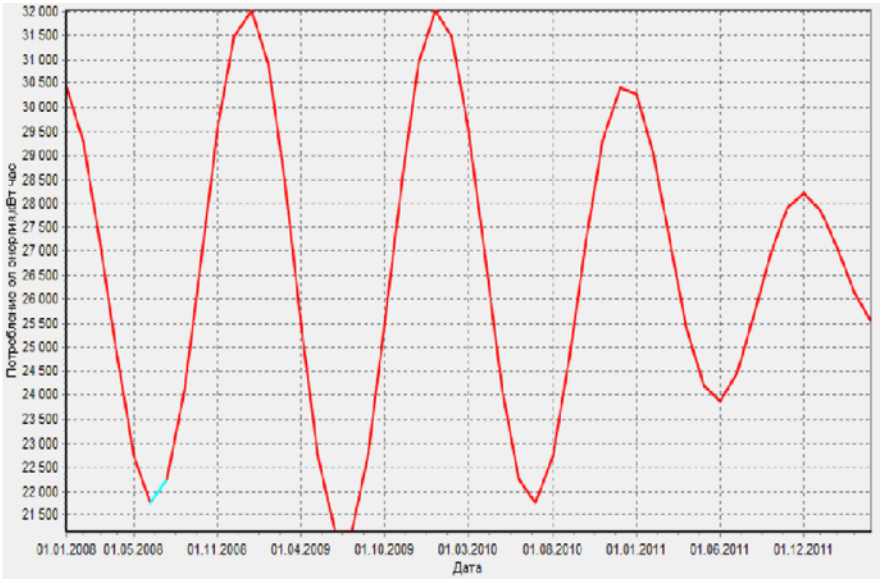


Рис. 2.15. Таблица и диаграмма потребления электроэнергии для объекта 1 после спектральной обработки

3.3.2. Формирование обучающего множества для процедуры прогнозирования.

Временной ряд методом *скользящего окна* преобразуется в таблицу с обучающими примерами. Для этого в текущем узле с результатами спектральной обработки запускается *Мастер обработки,* выбирается метод обработки данных – *Скользящее окно* (рис. 2.16).

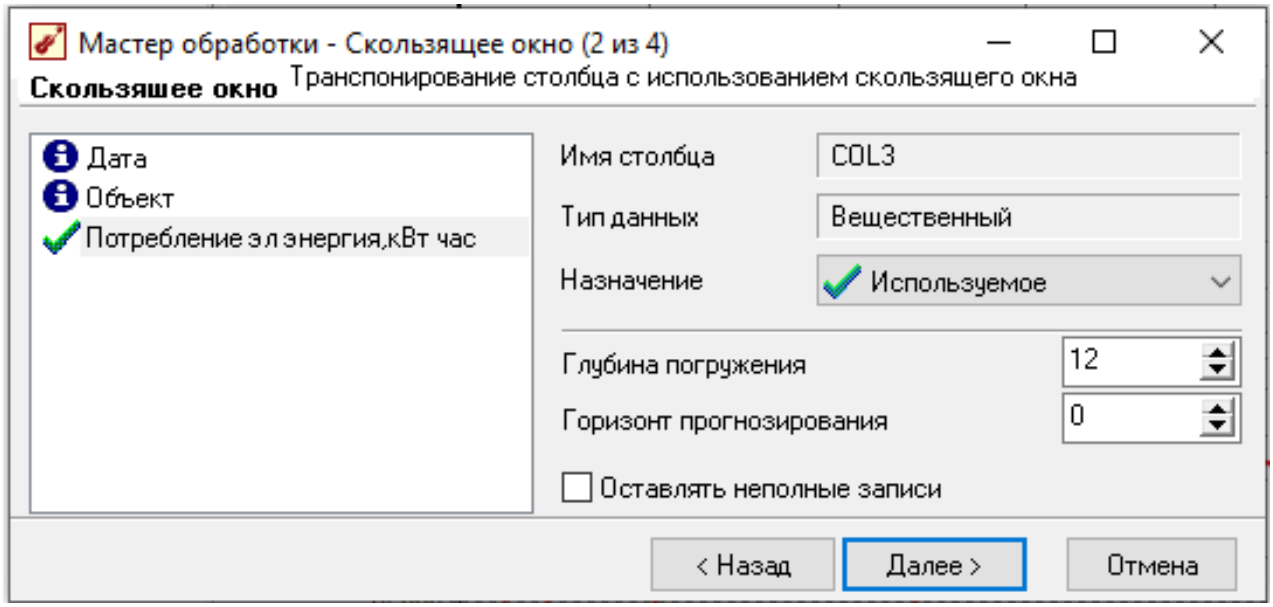


Рис. 2.16. Настройка параметров скользящего окна

Для столбца, содержащего результаты наблюдений, указывается назначение *Используемое,* а затем устанавливается *глубина погружения* и *горизонт прогнозирования*. Поскольку в данных потребления электроэнергии была обнаружена тенденция годовой сезонности, то *глубина погружения* должна быть равна 12. Для построения обучающего множества *горизонт планирования* принимается равным 0.

Таблица с обучающими примерами формируется методом скользящего окна и будет содержать в начале и в конце неполные записи – их нужно *исключить* из рассмотрения (не ставить флажок в поле «*Оставлять неполные записи*»).

В табл. 2.5 представлен результат построения таблицы с обучающими данными о потреблении электроэнергии, полученной в результате преобразования временного ряда методом скользящего окна, с полными записями.

Таблица 2.5

Фрагмент таблицы о потреблении (*П*) электроэнергии

Дата	*Объект*	*П-12*	*П-11*		*П-1*	*П*
01.02.2009	Объект 1	30391,54	29287,49	…	32022,81	30938,78
01.03.2009	Объект 1	29287,49	27221,31	…	30938,78	28528,29
01.04.2009	Объект 1	27221,31	24791,18	…	28528,29	25500,902
01.05.2009	Объект 1	24791,18	22754,31	…	25500,902	22766,17
…	…	…	…	…	…	…

3.3.3. Построение нейросетевой модели.

На основе построенной таблицы с обучающими примерами о потреблении электроэнергии строится *нейросетевая модель* для решения задачи прогнозирования. Для узла *Скользящее окно* открывается *Мастер обработки,* выбирается метод *Data Mining* – Нейросеть.

Принимается решение о построении прогноза на 1 месяц вперед, основываясь на данных 12, 11 месяцев назад, 2 месяца назад и 1 месяц назад. В соответствии этим решением, в качестве входных данных нейросети назначаются следующие столбцы построенной методом скользящего окна таблицы с обучающими примерами: «*Потребление–12*», «*Потребление–11*», «*Потребление–2*», «*Потребление–1*», а в качестве выхода ИНС – «*Потребление*» (рис. 2.17). Остальные столбцы – информационные.

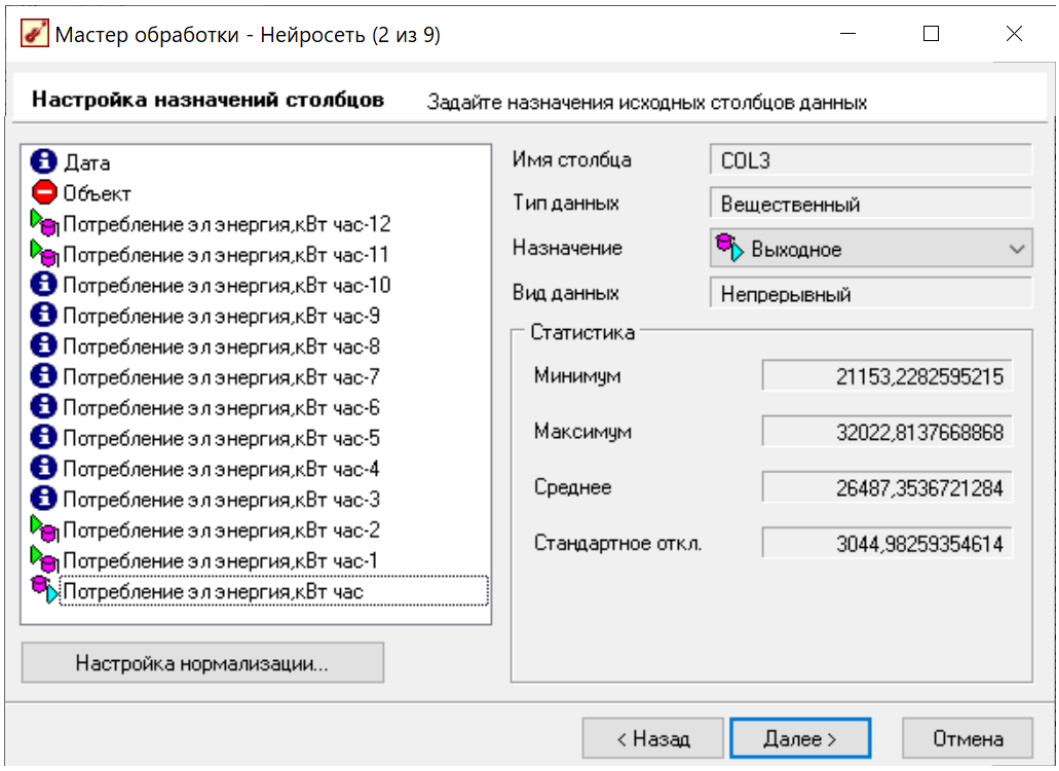


Рис. 2.17. Настройка назначений входов и выходов нейросети

На следующем шаге мастера обработки определяется структура ИНС. Очевидно, число входных нейронов соответствует числу входных столбцов и равно 4, число выходных нейронов – 1. Пусть количество скрытых слоев равно 1, число нейронов в скрытом слое – 5; все остальные параметры оставляются без изменения.

При настройке процесса обучения нейросети выбирается метод обратного распространения ошибки – *Back – Propagation*, параметры обучения и параметры остановки обучения остаются без изменения.

Для просмотра результатов обучения при определении спообов отображения данных выбираются визуализаторы *Диаграмма* и *Диаграмма рассеяния*. Необходимо обратить внимание на то, что при настройке параметров *Диаграммы* выбирается реальное и спрогнозированное значение выхода нейросети – *«Потребление»* и *«Потребление_OUT»* (рис. 2.18).

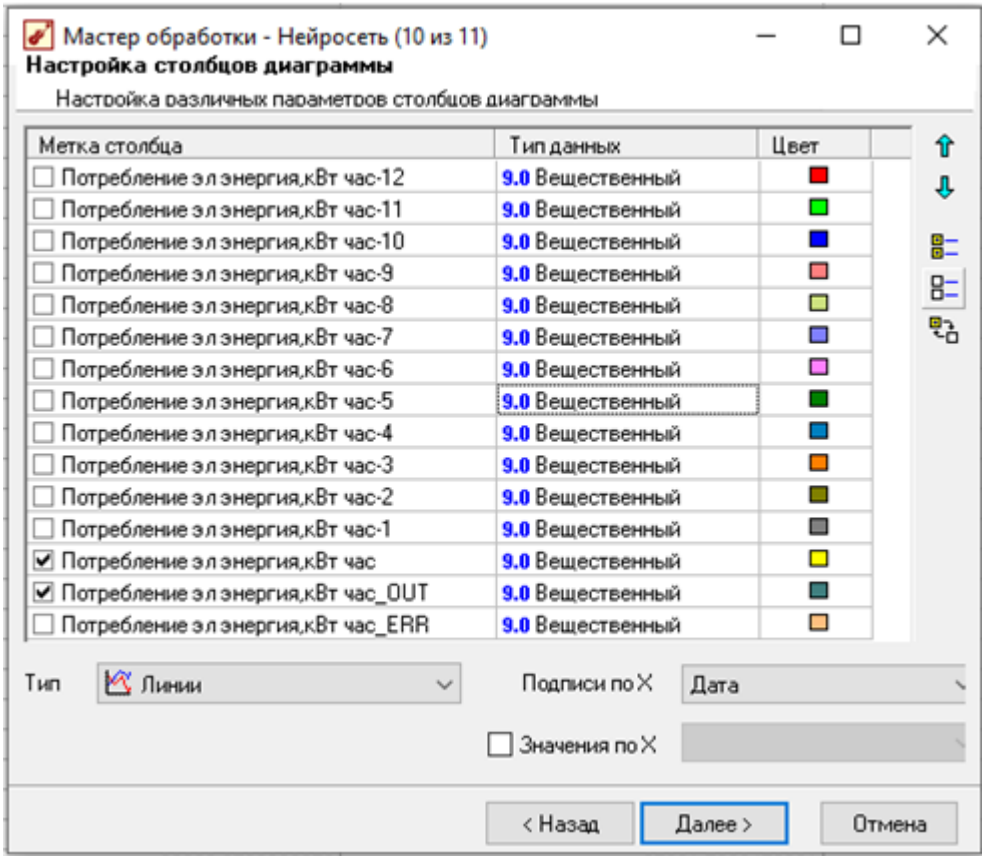


Рис. 2.18. Настройка параметров *диаграммы*

На рис. 2.19 представлены результаты обучения нейронной сети – почти для всех объектов разница между реальным и спрогнозированным значением выхода нейросети очень мала, что свидетельствует о хороших результатах обучения.

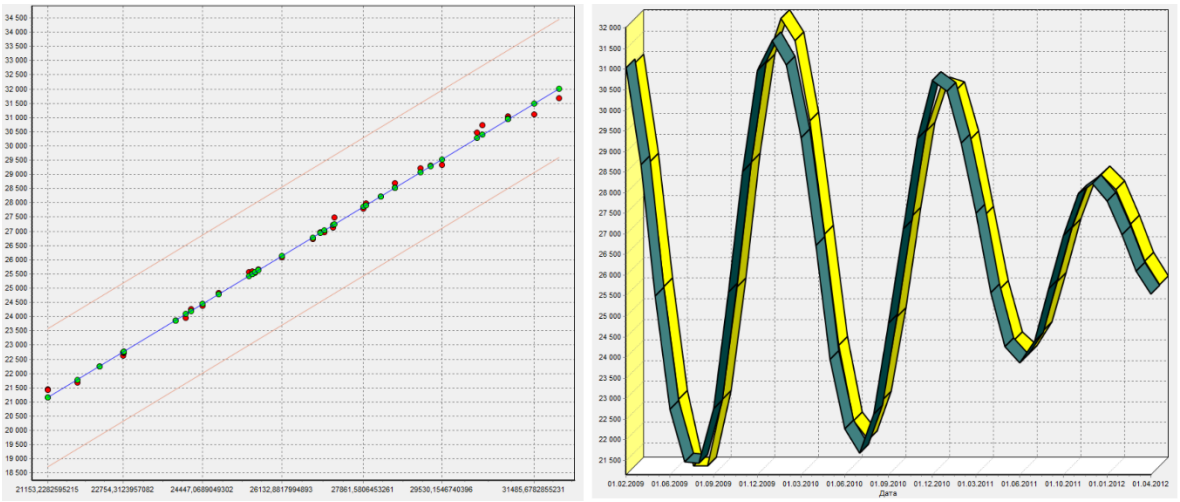


Рис. 2.19. Результаты обучения ИНС на *диаграмме рассеяния* и *диаграмме* реального и прогнозируемого выхода нейросети

3.3.4. Решение задачи прогнозирования на основе нейросетевой модели.

Для построения прогноза потребления электроэнергии на следующие периоды для узла *Нейросеть* открывается *Мастер обработки,* выбирается *Прогнозирование*.

При настройке прогнозирования временного ряда (рис. 2.20) устанавливаются связи столбцов: для каждого столбца указывается поле, которое на очередном шаге нужно использовать для расчета прогнозируемого значения.

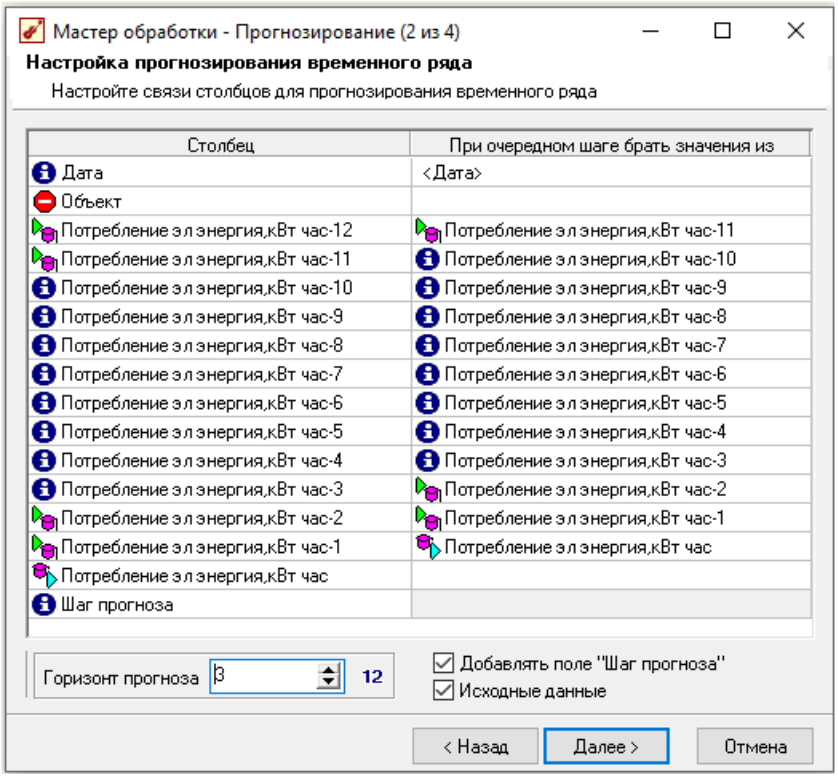


Рис. 2.20. Настройка прогнозирования временного ряда

В окне настройки прогнозирования связи между столбцами можно установить по умолчанию; а в поле *Горизонт прогноза* вносятся будущий период, на который необходимо сделать прогноз – например, на 3 месяца вперед. Для удобства работы к результирующей выборке можно добавить исходные данные, установив в окне настройки соответствующий флажок.

При выборе способов отображения результатов прогнозирования данных о потреблении электроэнергии в качестве визуализатора выбирается *Диаграмма прогноза*, в которой необходим вывод на экран выходного столбца «*Потребление*» (рис. 2.21).

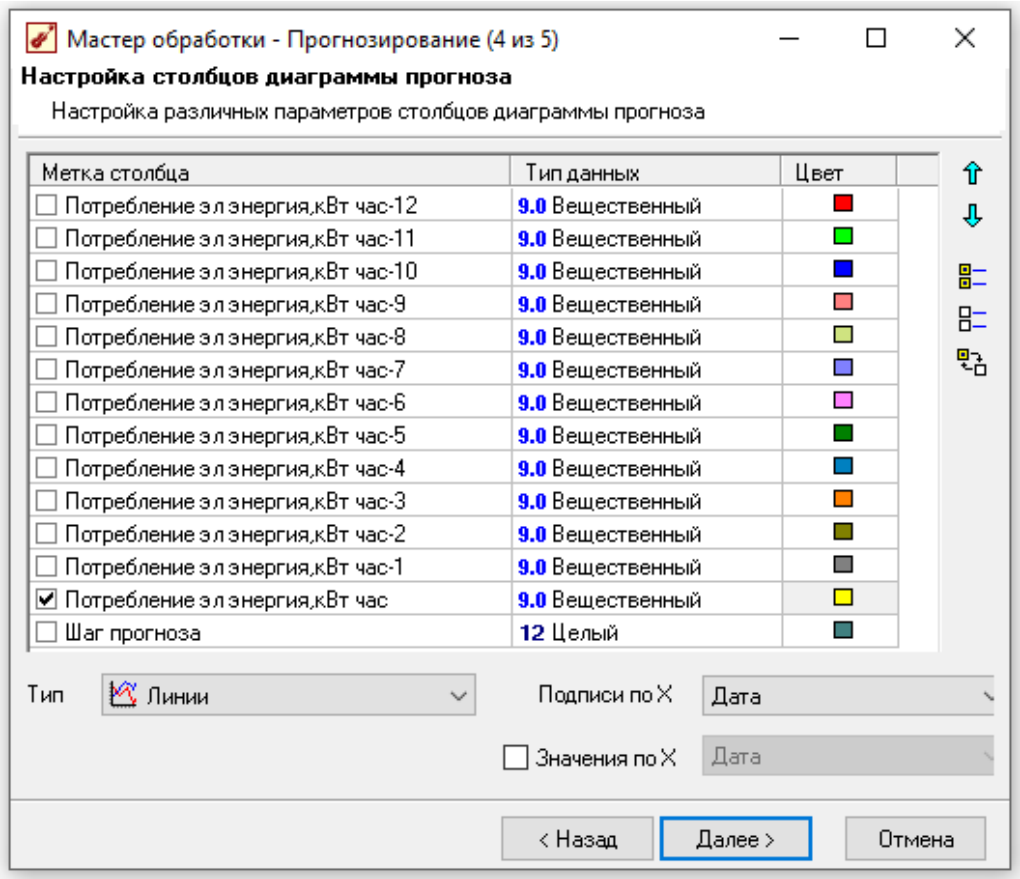


Рис. 2.21. Настройка отображения *Диаграммы прогноза*

Итак, результаты прогнозирования временного ряда в виде *диаграммы прогноза* с расчетными значениями (меню *Тип меток – Значение*) и в трехмерном виде (меню *Трехмерный вид*) представлены на рис. 2.22. Модель прогноза демонстрирует, что в следующие 3 месяца будет увеличение потребления электроэнергии до 31960,018 кВт·час.

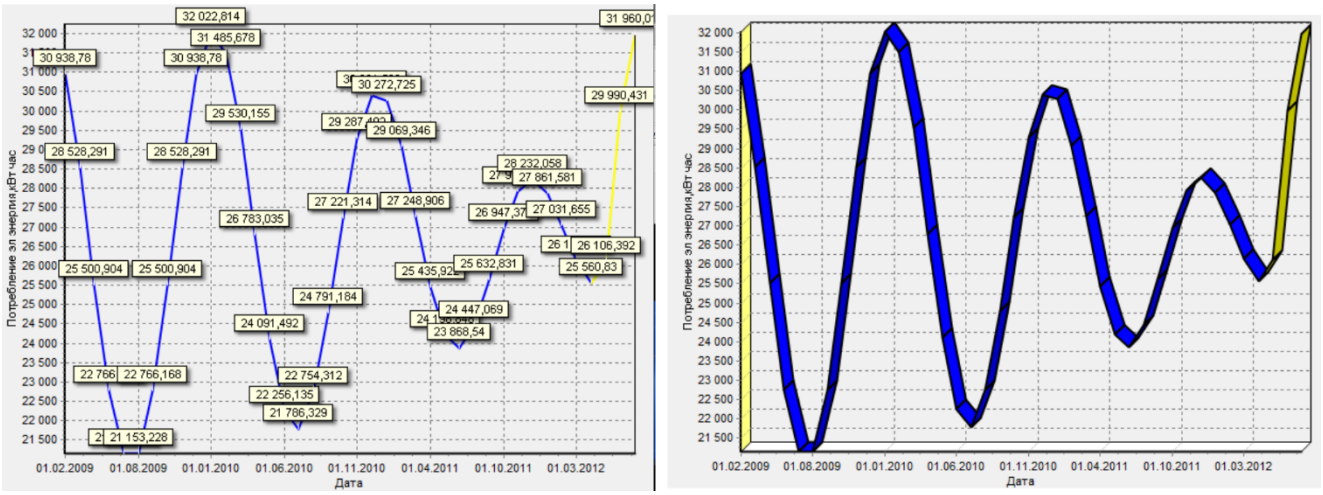


Рис. 2.22. Диаграмма прогноза с расчетными значениями и в трехмерном виде

Порядок выполнения работы

1. Изучить методику нейросетевого моделирования и методы решения задач классификации, аппроксимации функции и прогнозирования в пакете *Deductor*.

2. Решить задачу классификации с целью ознакомления с основными приемами работы в среде *Deductor*. Рассмотреть пример «*Грибы*» (п. 3.1).

Построить нейросетевую модель, решающей задачу классификации грибов к классу съедобных или ядовитых: студент должен получить ответ на вопрос «съедобный ли гриб?» и представить результаты опроса нейросети при добавлении новых видов грибов.

Выявить значимые и незначимые признаки в модели; для значимых признаков определить диапазоны значений, которые влияют на принятие решения об отнесении гриба к классу съедобных / несъедобных.

3. Решить задачу аппроксимации функции. По варианту задания сформировать обучающую выборку по математической функции (выполнить расчеты в *Excel*, затем скопировать в текстовый файл *.*txt*) и построить нейросетевую модель.

Провести опрос сети, записать полученные результаты в таблицу (п. 3.2, табл. 2.2).

Повысить точность аппроксимации, улучшая качество обучающих примеров, расширяя размер выборки или изменяя структуру ИНС путем настраивания количества скрытых слоев, числа нейронов в слоях, вида и параметров активационной функции, а также параметров обучения. Исходные данные и результаты экспериментов занести в таблицу.

4. Ознакомиться с решением задачи прогнозирования с помощью нейронных сетей на основе примера «*Потребление электрической энергии*» для *Объекта 1* (п. 3.3). Самостоятельно решить задачу нейросетевого прогнозирования для Объекта 2 или Объекта 3.

5. Сделать выводы об особенностях задач классификации, аппроксимации и прогнозирования в нейросетевом базисе.

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

1. Название и цель работы.

2. Постановка задачи классификации в нейросетевом базисе для выборки «Грибы»; структура ИНС, параметры обучения; результаты обучения и опроса ИНС; значимые и незначимые признаки ИНС. Результаты выполнения задания.

3. Постановка задачи аппроксимации функции. Математическая функция. Содержание обучающей выборки; структура ИНС, параметры обучения; результаты обучения и опроса ИНС. Таблица экспериментальных исследований по настройке параметров ИНС.

4. Постановка задачи прогнозирования. Исходный временной ряд, результаты применения механизмов очистки данных от шумов, аномалий, выявления сезонности, сглаживания; построения скользящего окна. Структура ИНС, параметры обучения; результаты обучения и опроса ИНС.

5. Выводы об особенностях задач классификации, аппроксимации и прогнозирования в нейросетевом базисе.

Варианты заданий

№	Математическая функция	Диапазон изменения x_1 и x_2
1	$f(x_1, x_2) = (1 - {x_1}^2) + 2(1 - x_2)^2$	[–1;1]
2	$f(x_1, x_2) = 2(1 + {x_1}^2) + 3(x_2 - 2)^2$	[–1;1]
3	$f(x_1, x_2) = (x_1 - 1)^2 + (2 - {x_2}^2)$	[–1;1]
4	$f(x_1, x_2) = 2(x_1 - 1)^2 + (1 + {x_2}^2)$	[–1;1]
5	$f(x_1, x_2) = (3 - {x_1}^2) + 2(1 + x_2)^2$	[–1;1]
6	$f(x_1, x_2) = (2 + {x_1}^2) + (1 - 2x_2)^2$	[–1;1]
7	$f(x_1, x_2) = (1 + {x_1}^2) + (1 - x_2)^2$	[–1;1]
8	$f(x_1, x_2) = (3 - 2{x_1}^2) + (1 - 2x_2)^2$	[–1;1]
9	$f(x_1, x_2) = (2 - {x_1}^2) + (1 + x_2)^2$	[–1;1]
10	$f(x_1, x_2) = (1 - 2{x_1}^2) + (1 - x_2)^2$	[–1;1]

Контрольные вопросы

1. Что представляет собой модель искусственного нейрона? Какие задачи решаются нейросетями? Назовите области применения нейронных сетей.
2. Какие можно выделить классы нейросетевых архитектур?
3. Назовите парадигмы обучения искусственных нейронных сетей. Что представляет собой алгоритм обучения с учителем?
4. Как проявляется в нейронной сети свойство обобщения? Что такое эпоха в обучении нейросетей?
5. Как повысить качество обучения нейронных сетей? Назовите основные проблемы обучения нейросетей.

Список литературы

1. Хайкин С. Нейронные сети: полный курс, 2-е изд.: пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.
2. Васильев В. И., Ильясов Б. Г. Интеллектуальные системы управления. Теория и практика: учебное пособие. М.: Радиотехника, 2009. – 392 с.
3. Паклин Н. Б., Орешков В. И. Бизнес-аналитика: от данных к знаниям: учеб. пособие. СПб.: Питер, 2010. 704 с.
4. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский; пер. с польск. И. Д. Рудинского. М.: Горячая линия-Телеком, 2008. 383 с.
5. Круглов В. Искусственные нейронные сети. Теория и практика / В. В. Круглов, В. В. Борисов. 2-е изд. М.: Горячая линия-Телеком, 2002. 382 с.
6. Круглов В. В., Дли М. И., Голунов Р. Ю. Нечёткая логика и искусственные нейронные сети: учеб. пособие. М.: Физматлит, 2001. 224 с.
7. Матвеев М. Г. Модели и методы искусственного интеллекта. Применение в экономике: учеб. пособие / М. Г. Матвеев, А. С. Свиридов, Н. А. Алейникова. М.: Финансы и статистика: ИНФРА-М, 2008. 446 с.

Лабораторная работа № 3

НЕРОСЕТЕВАЯ КЛАСТЕРИЗАЦИЯ С ПОМОЩЬЮ СЕТЕЙ КОХОНЕНА В АНАЛИТИЧЕСКОЙ ПЛАТФОРМЕ *DEDUCTOR* И СИСТЕМЕ *MATLAB*

1. Цель и задачи работы

Целью работы является изучение алгоритма нейросетевой кластеризации с использованием сетей Кохонена в аналитическом пакете *Deductor* и подсистеме *Neural Networks Toolbox* системы *Matlab.*

Задачи работы: формирование умений и навыков обучения искусственных нейронных сетей на основе парадигмы обучения без учителя; закрепление навыков построения и интерпретации самоорганизующихся карт Кохонена.

2. Теоретические сведения

2.1. Нейронные сети Кохонена

Кластерный анализ, или автоматическая классификация, занимает одно из центральных мест среди методов анализа данных и представляет собой совокупность подходов, методов, алгоритмов, предназначенных для формирования однородных классов. Задача кластерного анализа заключается в нахождении некоторого теоретико-множественного разбиения исходного множества объектов любой предметной области на непересекающиеся подмножества (*кластеры*) таким образом, чтобы элементы, относимые к одному подмножеству, отличались между собой в значительно меньшей степени, чем элементы из разных подмножеств.

На сегодняшний день предложено несколько десятков алгоритмов кластеризации и их разновидностей. Одним из наиболее известных методов *нейросетевой кластеризации* являются *самоорганизующиеся карты* (СОК) *Кохонена* – особый класс *искусственных нейронных сетей,* введенных финским ученым Т. Кохоненом в 1982 г.

Нейронные сети Кохонена – это прямонаправленные нейронные сети, основанные на *конкурентом* обучении. Целью используемых

в них алгоритмов обучения является выявление из множества входных данных существенных (значимых) образов или признаков без участия внешнего учителя. Таким образом, алгоритмы обучения основаны на подобии образов и размещают близкие образы в один кластер.

Пусть имеются исходные данные, содержащие n объектов, каждый из которых характеризуется p признаками; каждый q-й входной вектор имеет вид $X=[x_1, x_2,\ldots,x_p]^q$, где $q \in (1 \div n)$. Строится нейронная сеть Кохонена, состоящая из m нейронов, образующих решетку на плоскости (рис. 3.1). Сеть Кохонена не имеет скрытых слоев: данные с входного слоя передаются непосредственно на выходной слой, нейроны которого упорядочены в одномерную или двумерную решетку прямоугольной или шестиугольной формы.

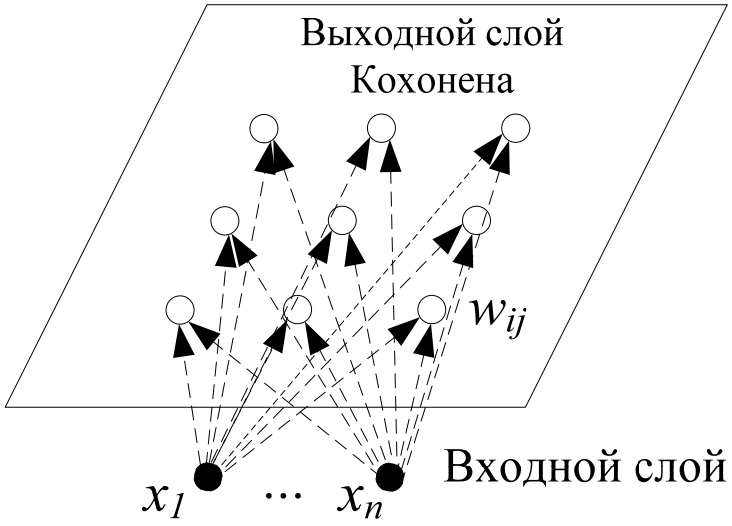


Рис. 3.1. Структура сети Кохонена

Входной (*распределительный*) *слой* нейросети содержит по одному нейрону для каждого входного признака $x_1,\ldots,x_p$, причем входные нейроны не участвуют в процессе обучения; их задача – передать значения входных признаков обучающего множества на нейроны выходного слоя.

Число *выходных нейронов* сети Кохонена соответствует числу кластеров, которые должны быть построены. Значения каждого признака объекта поступают через входные нейроны на каждый нейрон выходного слоя. С каждым j-м нейроном слоя Кохонена ассоциируется весовой вектор W_j, которому соответствует точка входного пространства: $W_j = w_{j1}, w_{j2}, \ldots, w_{jp}$, где j=1,2,…,m.

Отдельные нейроны выходного слоя сети Кохонена соревнуются (конкурируют друг с другом) за право активации – каждый *j*-й нейрон сравнивается с входным вектором *X*. Сравнение предполагает вычисление расстояния между X и W_j, так что в слое Кохонена появляется нейрон-победитель с индексом $i(x)$, веса которого имеют наименьшее расстояние до входного вектора:

$$i(x) = arg\ \min_j \|X - W_j\|.$$

В качестве метрики может выступать евклидово расстояние $d(, W_j) = \sqrt{\sum_{i=1}^{n}(x_{qi} - w_{ji})^2}$ между векторами X_q и W_j.

Если векторы X и W_j – нормализованные, то в качестве меры близости можно использовать скалярное произведение $W_j \cdot X$.

$$\min_{j=\overline{1,m}} \left(\sqrt{\sum_{i=1}^{n}(x_{qi} - w_{ji})^2}\right) = \min_{j=\overline{1,m}} \left(\sqrt{\sum_{i=1}^{n} x_{qi}^2 - 2x_{qi} \cdot w_{ji} + w_{ji}^2}\right) =$$

$$\min_{j=\overline{1,m}} \left(\sqrt{\sum_{i=1}^{n} x_{qi}^2 - \sum_{i=1}^{n}\left(2x_{qi} \cdot w_{ji}\right) + \sum_{i=1}^{n} w_{ji}^2}\right). \qquad (3.1)$$

Поскольку для нормализованных обучающих векторов X, т.е. когда выполняется равенство $\sum_{i=1}^{n} x_{qi}^2 = 1$, стремящиеся в процессе обучения к ним векторы весов W_j автоматически нормализуются, так что выполняется равенство $\sum_{i=1}^{n} w_{ji}^2 = 1$. Следовательно, минимум выражения (3.1) будет достигаться при максимуме суммы произведения $\sum_{i=1}^{n}(x_{qi} \cdot w_{ji})$. Таким образом, критерий соответствия, основанный на минимизации евклидового расстояния, математически эквивалентен максимизации скалярного произведения $y_i = W_j \cdot X$.

Итак, результатом работы выходного слоя конкурирующих нейронов при подаче на входной слой некоторого вектора *X* является определение нейрона, который имеет наибольший выходной сигнал y_i (нейрон-победитель). Этот нейрон обладает весовым вектором W_j, который наиболее близок к входному вектору. В результате определяется местоположение, которое должно стать центром топологической окрестности возбужденного нейрона.

Алгоритм обучения сети Кохонена

Шаг 1. Задается структура сети (количество нейронов *m* слоя Кохонена).

Шаг 2. Весовые коэффициенты w_0 нейронов инициализируются небольшими *случайными* значениями. Однако рандомизация весов

слоя Кохонена может породить серьезную проблему обучения: если начальная плотность весовых векторов в окрестности обучающих векторов слишком мала, то может оказаться невозможным разделение сходных классов из-за того, что не будет достаточного количества весовых векторов в интересующей окрестности для приписывания их классам входных векторов. Наилучшее решение этой проблемы заключается в распределении весовых векторов в соответствии с плотностью входных векторов; весовые коэффициенты устанавливаются равными величине $w_0 = \frac{1}{\sqrt{N}}$, где N – размерность входного вектора.

Далее производится нормализация входных образов в соответствии с формулой: $\overline{x}_{qi} = \frac{x_{qi}}{\sqrt{\sum_{j=1}^{n}(x_{qj})^2}}$. Нормализация позволяет сократить длительность процесса обучения.

Шаг 3. На входы нейронной сети подается вектор x_{qi}, и рассчитываются расстояния от входного вектора до центров всех кластеров.

Шаг 4. По наименьшему из значений расстояний выбирается нейрон, в наибольшей степени близкий по значениям к входному вектору (т.е. определяется «нейрон-победитель»), и производится корректировка его весовых коэффициентов.

Вектор весов выходных нейронов W_j должен изменяться по направлению к входному вектору X_q. Нейроны, которые являются соседями нейрона-победителя, также подстраивают свои весовые коэффициенты, используя линейную комбинацию входных векторов и текущих векторов весов:

$$w_j(t+1) = w_j(t) + \eta(t) \cdot [x_q - w_j(t)] \cdot H(j), \qquad (3.2)$$

где $\eta(t)$ – коэффициент *скорости обучения;* $W_j(t)$ и $W_j(t+1)$ – векторы синаптических весов с учетом формализации дискретного времени в моменты времени (t) и $(t+1)$.

Далее определяется топологическая окрестность $H(j)$ с центром в победившем нейроне, которая состоит из множества нейронов; окрестность плавно сходит на нет с увеличением расстояния $d(j,i)\to\infty$ между нейроном-победителем i и остальными j-ми нейронами окрестности.

Выделяют следующие функции окрестности:

единичная окрестность $-H(j)=\begin{cases}1, j=i\\0, j\neq i\end{cases}$ (3.3)

прямоугольная окрестность $-H(j)=\begin{cases}1, d(j,i)\leq\lambda\\0, d(j,i)>\lambda\end{cases}$ (3.4)

гауссова окрестность $-H(j)=e^{\frac{d^2(j,i)}{2\lambda^2}}$, (3.5)

где λ – радиус обучения, или уровень соседства, определяющий ширину ближайшей окрестности нейрона-победителя, в которой нейроны будут считаться соседями и участвовать в процессе обучения.

Радиусу обучения, или величине уровня соседства λ, присваивается достаточно большая величина λ_0, чтобы обучались все нейроны. Определяется минимальное значение уровня соседства $\lambda_{\min}$, при котором будет останавливаться обучение и шаг его изменения – $\Delta\lambda$.

При каждом представлении нового примера обучающей выборки скорректированные векторы синаптических весов в окрестности нейрона-победителя будут стремиться следовать за распределением входных векторов. Нейроны, близкие к победителю, так же, как и сам победитель, изменяют свои веса в значительной степени, в то время как удаленные от победителя нейроны испытывают меньшие изменения весов.

При продолжении процесса обучения размер области соседства от нейрона-победителя уменьшается: в начале обучения значительное число выходных нейронов будут изменяться, но затем все меньше и меньше нейронов подвергается модификации своих весов. В конце обучения только победитель регулирует свой вес. Таким образом, представленный алгоритм ведет к топологическому упорядочиванию пространства признаков во входном пространстве, то есть корректируемые в решетке нейроны будут стремиться иметь одинаковые синаптические веса.

Аналогичная процедура происходит и со *скоростью обучения*: по мере обучения коэффициент скорости обучения $\eta(t)$ уменьшается, и его снижение до нуля останавливает обучающий процесс. В первой фазе обучения, на этапе *грубой* подстройки, скорость обучения $\eta(t)$ велика, и веса нейронов корректируются значительно, что позволяет лишь примерно настроить их в соответствии с распределением значений признаков объектов в исходной выборке. Во второй фазе

обучения, на этапе *точной* подстройки, скорость обучения уменьшается, что позволяет подстраивать весовые коэффициенты нейронов более аккуратно. При этом коэффициент скорости обучения, с которым веса нейрона победителя и его соседей будут двигаться в сторону очередного экземпляра данных, изменяется в зависимости от текущей эпохи обучения по правилу: $\lambda = \lambda_{нач} \cdot \left(\frac{\lambda_{кон}}{\lambda_{нач}}\right)^{\frac{T}{Tmax}} \lambda_{нач}$, где λ – текущий радиус обучения, $\lambda_{нач}$ и $\lambda_{кон}$ – начальная и конечная скорости обучения, T_{max} – количество эпох обучения, T – текущая эпоха обучения.

Для того чтобы при обучении правильно распределить плотность ядер классов в соответствии с плотностью входных векторов, применяется *модификация* алгоритма, получившая название *метода выпуклой комбинации*. Использование этого метода позволят избежать ситуации, когда различные классы, которым соответствуют плотно распределенные входные образы, сольются или, наоборот, раздробятся на дополнительные подклассы в случае близких образов одного и того же класса. Суть метода сводится к тому, что входные образы преобразуются по формуле:

$$x_{qi} = \beta(t) \cdot x_{qi} + \frac{1-\beta(t)}{\sqrt{N}}, \qquad (3.6)$$

где x_{qi} – i-я компонента входного образа; N – число входов каждого весового вектора; $\beta(t)$ – коэффициент, изменяющийся в процессе обучения от положительного числа, близкого 0, до 1. В начале обучения $\beta(t)$ очень мало, вследствие чего все входные векторы имеют длину, близкую к $\frac{1}{\sqrt{N}}$, и почти совпадают с векторами весов. В процессе обучения сети $\beta(t)$ постепенно возрастает, приближаясь к 1, в результате чего вначале на входы нейронной сети подаются практически одинаковые образы, а с течением времени они все больше сходятся к исходным. Это позволяет разделять входные векторы и окончательно приписывает им их истинные значения. Весовые векторы отслеживают один или небольшую группу входных векторов и в конце обучения дают требуемую картину выходов. Метод выпуклой комбинации хорошо работает, но замедляет процесс обучения, так как весовые векторы подстраиваются к изменяющейся цели.

Шаг 5. Величина уровня соседства *уменьшается* по формуле $\lambda = \lambda - \Delta\lambda$, и если $\lambda > \lambda_{\min}$, то осуществляются переход к шагу 3 и повтор обучения для всех входных образов. Таким образом, чем ближе конец обучения, тем меньше величина λ и тем точнее определяется группа нейронов, отвечающих каждому классу образов.

Обучение сети завершается, если выполняются одно или несколько условий окончания: исчерпано заданное предельное количество эпох обучения; не произошло значимого изменения весовых коэффициентов в пределах заданной точности на протяжении последней эпохи обучения; исчерпано заданное предельное физическое время обучения; а также если $\lambda \leq \lambda_{\min}$.

2.2. Самоорганизующиеся карты признаков

На основе нейронной сети Кохонена строятся более сложные структуры – самоорганизующиеся карты (СОК) признаков (*Self Organizing Maps – SOM*), которые представляют собой один из вариантов кластеризации многомерных векторов в виде двумерных карт, где расстояния между объектами соответствуют расстояниям между их векторами в многомерном пространстве, а сами значения признаков отображаются различными цветами и оттенками.

При отображении многомерного пространства на плоскую карту неизбежно нарушается его топологическое подобие, т.е. существенно удаленные объекты (с малой степенью сходства) в исходном пространстве признаков могут на карте оказаться рядом друг с другом. Из-за потери информации об одном или нескольких измерениях (признаках) может быть сделан неверный вывод о схожести объектов по их свойствам, что станет причиной ошибочного отнесения объектов к одному классу. Можно добиться сохранения *топологического* подобия отображения многомерных векторов, если проецировать на карту не сами объекты, а *расстояния* между ними, вычисленные в соответствии с определенной метрикой. Таким образом, чем больше расстояние между объектами на карте, тем больше расстояние между векторами их признаков и, следовательно, тем больше различаются их свойства.

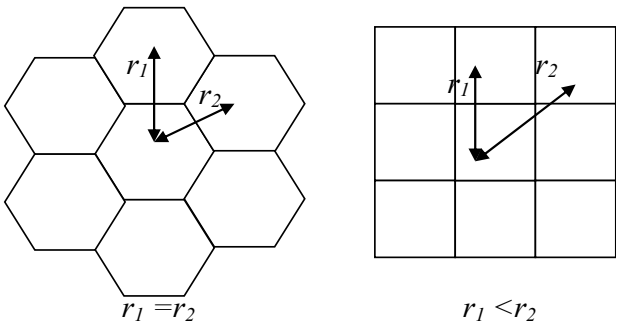
Отображение многомерного пространства данных на плоскости в виде СОК обладает богатыми выразительными возможностями. Цветовая раскраска карт по выделенным кластерам обеспечивает наглядность и удобство описания качественной характеристики

каждого кластера. СОК позволяют визуализировать результаты кластеризации в виде двумерных *тепловых карт* (*heat maps*), где каждому значению признака соответствует один из оттенков в заранее выбранной цветовой гамме. Карты выделенных признаков собираются воедино, и в результате формируется топографический атлас, дающий комплексное представление о структуре многомерных данных.

Карты Кохонена состоят из сегментов прямоугольной или шестиугольной формы, называемых ячейками. Каждая ячейка связана с определенным выходным нейроном и представляет собой «сферу влияния» данного нейрона: в нее попадают объекты, «захваченные» нейроном в процессе кластеризации. Распределение векторов весов нейронов карты происходит так же, как и в обычной сети Кохонена, т.е. на основе конкурентного обучения. Объекты, векторы которых оказываются ближе к вектору весов данного нейрона, попадают в ячейку, связанную с ним. Тогда распределение объектов на карте в целом будет соответствовать распределению векторов весов ее нейронов. Следовательно, если объекты на карте расположены близко друг к другу, то и их векторы будут близки; и наоборот, если объекты на карте находятся далеко друг от друга, то и векторы их признаков сильно различаются.

Хотя по расстоянию между объектами можно сделать выводы о степени их сходства или различия, также важна информация о том, в чем проявляется это сходство и различие, по каким признакам объекты различаются в наибольшей степени, а по каким – в наименьшей и т.д. Именно специальная раскраска помогает получить ответы на эти вопросы, выполняя функцию третьего измерения: каждой ячейке на карте назначается цвет в соответствии со значениями признаков объектов в ней.

Ячейки карты Кохонена могут быть прямоугольной и шестиугольной формы (рис. 3.2): шестиугольные ячейки более корректно отражают расстояния между объектами на карте, поскольку в этом случае расстояния между центрами смежных ячеек одинаковы; в случае же прямоугольных ячеек расстояния между центрами смежных ячеек зависит от их взаимного расположения.



а *б*

Рис. 3.2. Шестиугольные *(а)* и прямоугольные *(б)* ячейки СОК

Для выбора числа нейронов на карте Кохонена не существует строгих правил. Оптимальное количество нейронов определяется исходя из решаемой задачи и особенностей данных. Например, если разброс значений признаков в обучающей выборке сильный (т.е. векторы объектов в пространстве признаков разрежены), то, возможно, следует уменьшить число нейронов СОК, чтобы избежать большого количества пустых ячеек, а если векторы объектов расположены в пространстве признаков плотно, то для получения лучших результатов можно увеличить число нейронов (рис. 3.3).

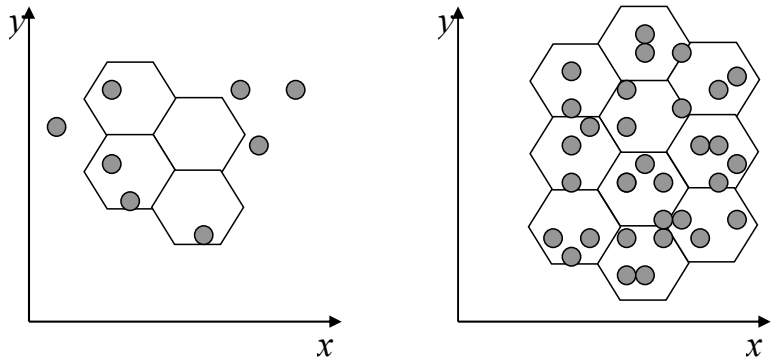


Рис. 3.3. Выбор числа нейронов карты

При построении карт Кохонена может возникнуть проблема *мертвых нейронов*, веса которых на начальном этапе были проинициализированы таким образом, что их векторы оказались в той части пространства признаков, где отсутствуют или почти отсутствуют векторы объектов. «Мертвые» нейроны в процессе конкуренции практически не имеют шансов стать победителями и принять участие в обучении и коррекции весов. Вследствие этого, обработка входных данных проводится меньшим (недостаточным) числом нейронов, что ухудшает качество кластеризации. Для решения этой проблемы используются различные подходы:

во-первых, можно инициализировать карту не случайными весами, а векторами признаков объектов обучающей выборки; во-вторых, подсчитывать количество «побед» каждого нейрона с целью учета его активности.

Область применения карт Кохонена довольно обширна; они с успехом применяются для решения задач моделирования, прогнозирования, поиска закономерностей в больших объемах данных, выявления набора независимых признаков, сжатия информации. Карты Кохонена позволяют выдвинуть *гипотезы о наличии кластерной структуры* и зависимостей в наборе данных; однако выдвинутые гипотезы *нужно* подтверждать другими методами. Особенностями карты Кохонена является то, что поскольку в большинстве случаев инициализация, или задание начальных векторов, весов нейронов сети является произвольной, происходит *потеря однозначности* результата, которая проявляется в том, что при каждом обучении сети всякий раз формируются разные карты Кохонена.

2.3. Особенности построения карт Кохонена в аналитической платформе *Deductor*

2.3.1. Загрузка файла с обучающей выборкой в аналитическую платформу *Deductor*.

Для загрузки файла вызывается  *Мастер импорта* (F6) на левой панели *Сценарии* главного окна программы *Deductor*. Из открывшегося списка типов источников данных выбирается прямой доступ к файлам типа *TXT – Импорт из TXT*, указывается имя, параметры импорта файла и запускается процесс импорта кнопкой *Далее.* Затем открывается таблица с обучающей выборкой и осуществляется настройка параметров столбцов таблицы – указывается их назначение, необходимое для дальнейшей обработки. *Вид данных* может быть *непрерывным* – значения в столбце могут принимать любое значение в рамках своего типа (числовые данные) и *дискретным* – значения могут принимать ограниченное число значений (строковые данные).

2.3.2. Построение карт Кохонена.

После процесса импортирования запускается *Мастер обработки,* в котором выбирается метод *Карта Кохонена*. Мастер обработки позволяет построить нейронную сеть с заданной структурой, определить ее параметры и обучить с помощью одного из доступных в системе алгоритмов обучения. Настройка и обучение ИНС состоит из следующих шагов.

Шаг 1. *Настройка назначения столбцов*. Этот шаг необходим для уточнения, какие столбцы таблицы с обучающей выборкой являются входными, выходными, информационными, непригодными, неиспользуемыми.

В качестве *входных* выбираются столбцы, которые будут использованы для обучения карты; эти столбцы обязательно должны иметь числовой тип; нечисловые столбцы не используются для обучения карты – их назначение должно быть *информативным* (рис. 3.4).

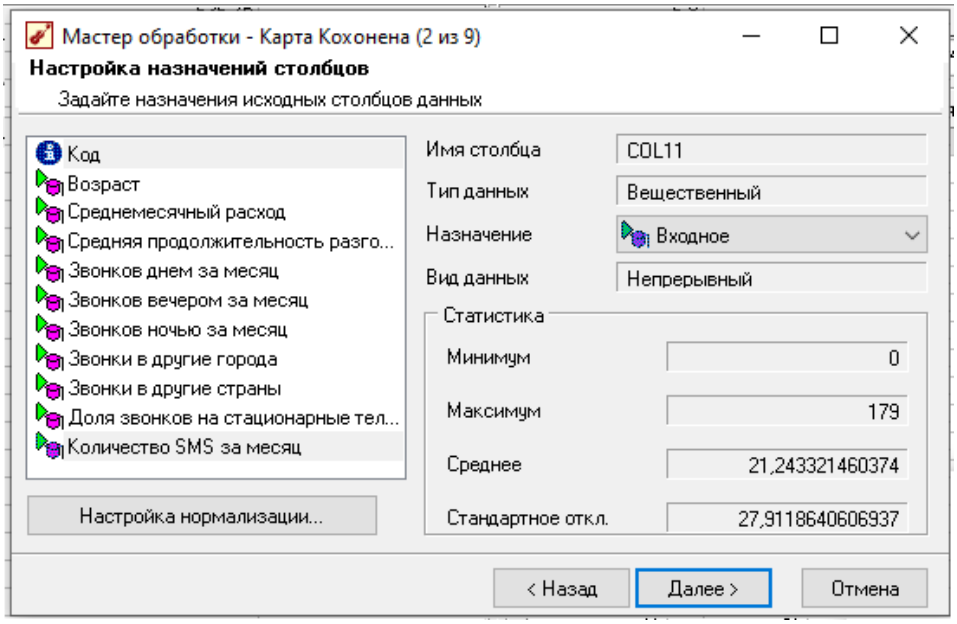


Рис. 3.4. Настройка назначений столбцов

Поскольку при решении задачи кластеризации с помощью СОК Кохонена основаны используется алгоритм обучения без учителя, то *выходные* столбцы для обучения не задаются. Однако для визуализации группировок на построенных картах и поиска закономерностей выходные столбцы могут быть заданы.

Настройка нормализации проводится с целью преобразования данных к виду, наиболее подходящему для обработки средствами пакета *Deductor*: нормализатор может преобразовать дискретные

данные к набору уникальных индексов, а данные столбца с непрерывными значениями, лежащими в произвольном диапазоне, – к заданному диапазону [0…1].

Шаг 2. *Настройка обучающей выборки*. Обучающая выборка разбивается на два множества – *обучающее* и *тестовое*. Способ разбиения исходного множества данных по умолчанию задан как *случайно*.

Обучающее множество включает записи (примеры), которые будут использоваться в качестве исходных данных для обучения ИНС. *Тестовое множество* используется для проверки результатов обучения ИНС. Поскольку любой метод кластеризации, в том числе и алгоритм Кохонена, субъективен, то смысл в выделении отдельного, тестового множества, как правило, отсутствует. Поэтому при разделении обучающей выборки в обучающем множестве необходимо оставить все 100 % записей (рис. 3.5).

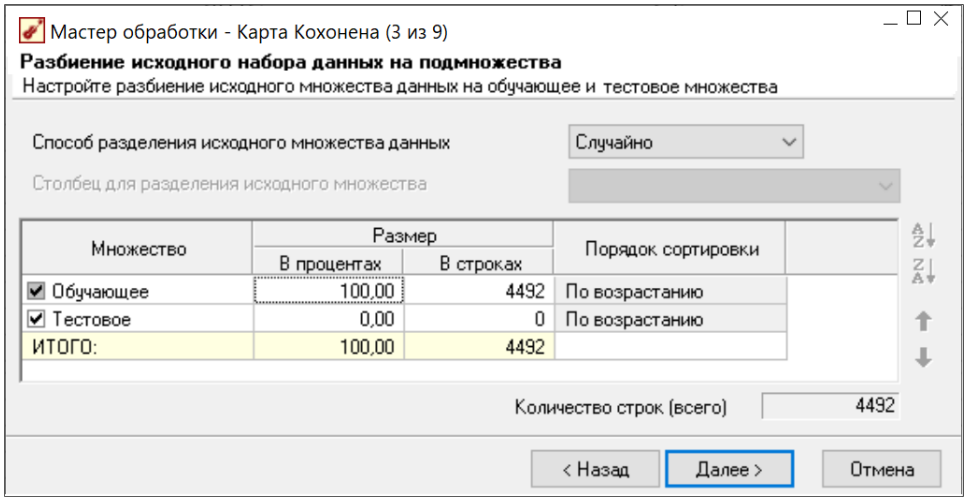


Рис. 3.5. Настройка обучающей выборки

Шаг 3. *Настройка параметров карты Кохонена* (рис. 3.6) задает размер карты, т.е. количество ячеек, из которых она будет состоять.

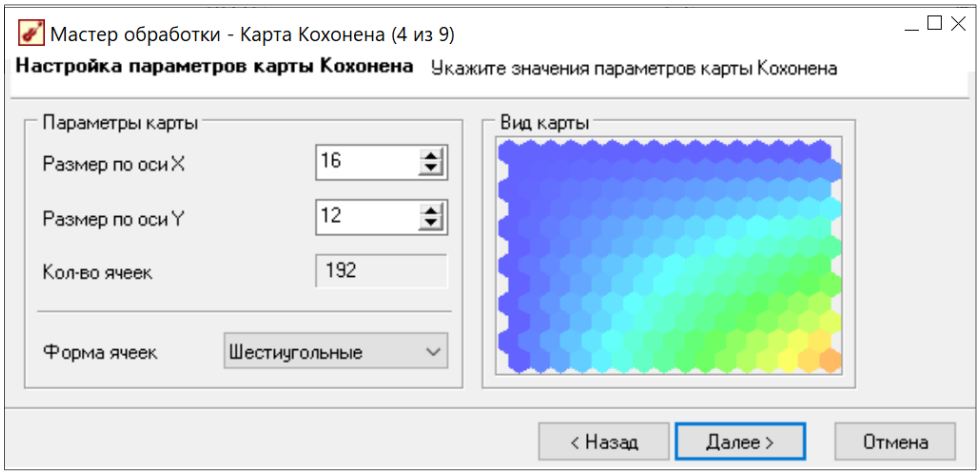


Рис. 3.6. Настройка параметров карты Кохонена

Для этого в полях «*Размер по оси X*» и «*Размер по оси Y*» указывается количество ячеек по соответствующим координатам. В поле «*Количество ячеек*» отображается общее число ячеек карты, равное произведению значений заданных полей.

При создании карты следует руководствоваться правилами:

– для поиска статистически значимых закономерностей общее число ячеек СОК должно быть на порядок меньше числа обучающих примеров;

– для исследования кластерной структуры данных число ячеек СОК должно быть сравнимо с числом примеров. При этом появятся пустые ячейки, разграничивающие области данных;

– рекомендуется выбирать такой размер карты, чтобы *на каждую ячейку приходилось хотя бы по две* записи.

В списке *«Форма ячеек»* выбирается один из вариантов конфигурации ячейки – прямоугольная или шестиугольная. При задании формы ячеек нужно учитывать, что шестиугольники дают более корректные результаты, т.к. расстояние между центрами ячеек в шестиугольниках ближе к евклидову, однако скорость обучения выше при использовании прямоугольной формы ячеек.

Шаг 4. *Настройка параметров остановки обучения.* Задаются условия, при выполнении которых обучение карты будет прекращено:

а) «*считать пример распознанным, если ошибка меньше*» – критерием останова обучения являются условия того, что рассогласование между эталонным и реальным выходом карты станет меньше заданного значения;

б) «*по достижению эпохи*» – параметр останова задает количество эпох, по достижении значения которого процесс обучения будет остановлен, т.е. даже тогда, когда не достигнута заданная точность обучения. Это условие останова позволяет избежать *зацикливания* в ситуациях, когда точность не достижима (ошибка не уменьшается).

Кроме того, для обучающего и тестового множества в соответствующих секциях окна могут независимо устанавливаться следующие критерии останова обучения: средняя квадратичная ошибка или максимальная квадратичная ошибка на обучающем множестве или тестовом множествах должна быть меньше заданного значения; количество распознанных примеров на обучающем множестве и тестовом множестве должно стать больше заданного процента.

Шаг 5. *Настройка параметров обучения карты Кохонена* (рис. 3.7). Указываются следующие параметры обучения:

а) *способ начальной инициализации* карты – это способ установления начальных весовых коэффициентов нейронов карты. Удачно выбранный способ инициализации позволяет существенно ускорить обучение и привести к получению более качественных результатов. Доступны три варианта инициализации: случайными значениями; случайными примерами из обучающего множества; из собственных векторов. При выборе способа начальной инициализации следует руководствоваться объемом обучающей выборки; количеством эпох, отведенных для обучения; размерами обучаемой карты;

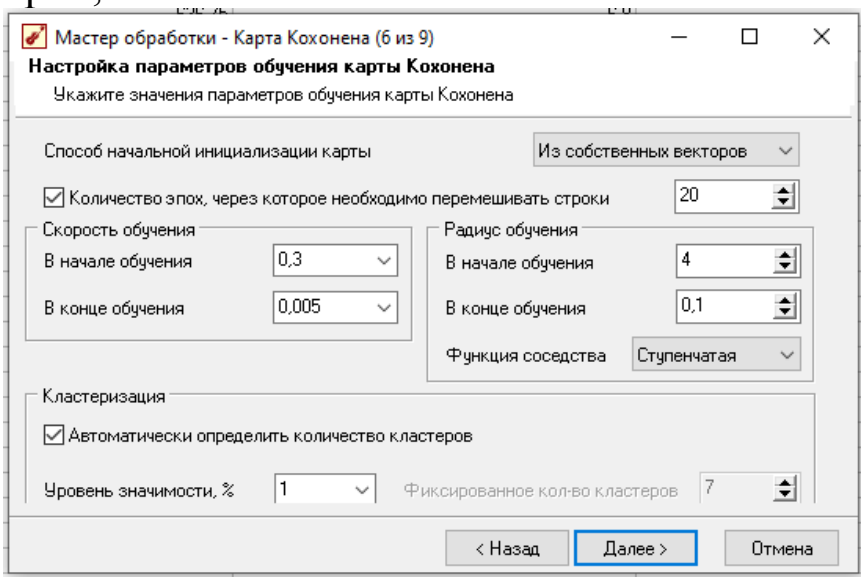


Рис. 3.7. Настройка параметров обучения карты Кохонена

– если объем обучающей выборки значительно (раз в 100) превышает количество нейронов карты и время обучения не играет первоочередную роль, то лучше выбрать инициализацию *случайными значениями*, т.к. в этом случае меньше вероятность попадания в локальный минимум ошибки кластеризации;

– если объем обучающей выборки не очень велик, ограничено время обучения или необходимо уменьшить вероятность появления «мертвых» нейронов (в которые не попадают объекты обучающей выборки), то следует использовать инициализацию примерами из *обучающего множества*;

– инициализацию *из собственных векторов* можно использовать при любом стечении обстоятельств, но в этом случае вероятность появления «мертвых» нейронов возрастает по сравнению с тем, чем если была бы использована инициализация примерами из обучающего множества. Этот способ рекомендуется выбирать при ознакомлении с данными;

б) в секции *скорость обучения* задаются скорость обучения *в начале* и *в конце обучения* карты Кохонена; рекомендуемые значения в начале обучения 0,1–0,3, в конце обучения 0,05–0,005;

в) *радиус обучения* – задается *в начале* и *в конце обучения*; в начале он должен быть достаточно большой – примерно половина или меньше размера карты (максимальное линейное расстояние от любого нейрона до другого любого нейрона), а в конце обучения – достаточно малый (примерно 1 или меньше);

г) *функция соседства* определяет, какие нейроны будут считаться соседними по отношению к нейрону-победителю (3.2–3.4). Например, если функция соседства ступенчатая, то «соседями» для нейрона-победителя будут считаться все нейроны, линейное расстояние до которых не больше текущего радиуса обучения. Выбор функции соседства влияет на скорость и качество обучения. Например, скорость обучения больше при использовании ступенчатой функции, а качество обучения выше при использование гауссовой функции соседства (обучение более плавное и равномерное);

д) *кластеризация* – в этой секции указывается способ определения количества кластеров – автоматическим способом или вручную. При автоматическом определении количества кластеров указывается *уровень значимости* – чем больше этот параметр, тем

большее количество кластеров будет получено. При ручном определении количества кластеров настраивается *фиксированное количество кластеров* – параметр, который задается пользователем.

Шаг 6. *Построение карты Кохонена.* На этом шаге запускается процесс обучения нажатием на кнопку *Пуск*; при этом можно наблюдать динамику процесса обучения сети Кохонена (рис. 3.8) с заданными параметрами. Остановка процесса обучения осуществляется кнопкой «*Стоп*».

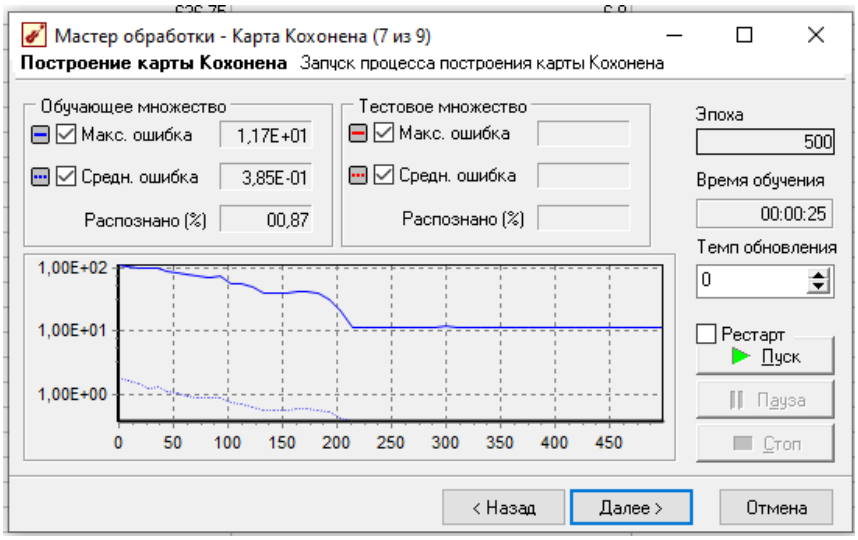


Рис. 3.8. Настройка параметров обучения карты Кохонена

Когда возникают ситуации необходимости повторной настройки обучения сети, можно продолжить обучение, начиная с текущих весовых коэффициентов – для этого нужно убрать флажок *Рестарт.* Если включить флажок *Рестарт*, то веса нейронов сети будут проинициализированы заново согласно выбранному на предыдущем шаге способу инициализации.

Шаг 7. *Настройка отображений карт Кохонена*. Для визуализации карт Кохонена необходимо отметить входные / выходные столбцов и специальные карты, которые будут показаны пользователю, и настроить параметры визуализации (рис. 3.9).

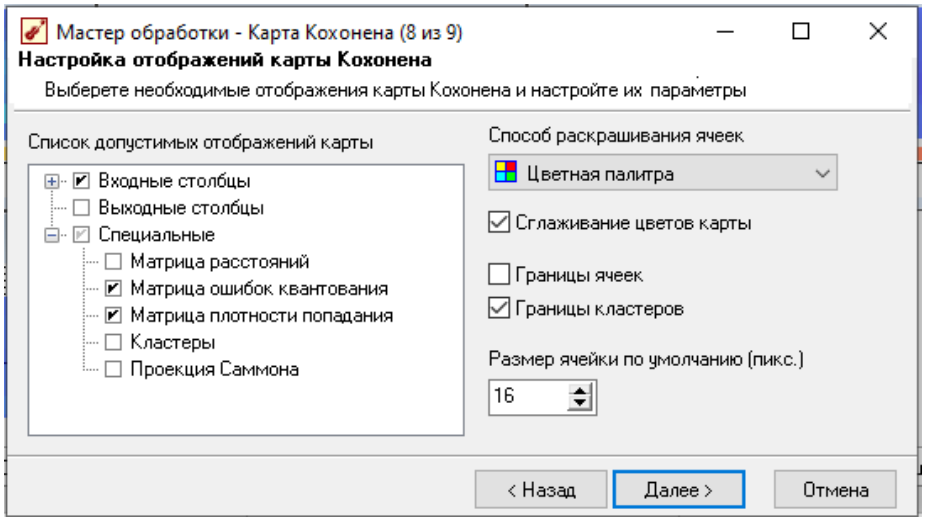


Рис. 3.9. Настройка параметров обучения карты Кохонена

В список допустимых отображений карт Кохонена входят *специальные* карты:

а) *матрица расстояний* – карта, необходимая для визуализации структуры кластеров, полученных в результате обучения. Элементы матрицы определяют расстояние между весовыми коэффициентами нейрона и его ближайшими соседями. Большое значение элемента матрицы свидетельствует о том, что данный нейрон сильно отличается от окружающих и относится к другому классу;

б) *матрица ошибок квантования* – карта, которая отображает среднее расстояние от входных объектов до центра ячейки. Она показывает качество обучения СОК Кохонена: чем меньше среднее расстояние до центра ячейки, тем ближе к ней расположены примеры, и тем лучше модель;

в) *матрица плотности попадания* – карта, отображающая количество объектов, попавших в ячейку;

г) *кластеры* – карты Кохонена, которые отображают ячейки, объединенные в кластеры алгоритмом *k-means*;

д) *проекция Саммона* – карта, являющаяся результатом проецирования многомерных данных на плоскость. Примечательно, что данные, расположенные рядом в исходной многомерной выборке, будут расположены рядом и на плоскости.

Для настройки отображения построенных карт Кохонена устанавливаются следующие параметры:

– *способ раскрашивания ячеек* – цветная палитра и или градация серого: если необходимо встраивать карту Кохонена в печатный отчет, то лучше выбрать серую цветовую схему;

– *сглаживание цветов карты* – позволяет обеспечить более плавный переход цветов в картах (без выбросов), но при этом ухудшается работа с каждой ячейкой в отдельности;

– *границы ячеек* – показывает границы ячеек на карте;

– *границы кластеров* – показывает границы кластеров на всех картах. Этот режим удобен для анализа структуры кластеров;

– *размер ячейки* на карте в пикселях (по умолчанию 16).

2.3.3. Настройка карт Кохонена.

Визуализация карт. После выполнения всех *этапов построения* СОК отображаются карты Кохонена, так что каждому входному столбцу соответствует своя карта. Текущая ячейка одновременно отображается на всех картах признаков маленькой окружностью черного цвета. Изменить текущую ячейку можно, щелкнув мышью в нужный участок карты. Внизу каждого отображения на градиентной шкале в желтом прямоугольнике показывается числовое значение признака, соответствующее ее цвету.

При работе с картами применяются операции, запускаемые либо из верхнего меню на панели инструментов визуализатора, либо из контекстного меню, вызываемого правой кнопкой мыши в любом окне карты.

В верхнем меню визуализатора карт расположен ряд кнопок:

 *границы ячеек* – показывает/скрывает границы ячеек;

 *границы кластеров* – показывает/скрывает границы кластеров;

 *сглаживание цветов карты* –плавный переход цветов;

 *настройка отображения* –окно настройки отображений карты;

 *настроить кластеры* – окно настройки количества кластеров.

Режимы работы с картами. При работе в СОК Кохонена в программе *Deductor* есть 4 режима работы, которые включаются с помощью верхнего и контекстного меню кнопкой  *Изменить режим работы с картами*:

 *простой режим* – режим, заданный по умолчанию, когда указатель мыши активизирует текущую ячейку в виде маленькой окружности одновременно на всех картах в одних и тех же координатах;

 *режим выделения* – режим, в котором указателем мыши можно выделить (или снять выделение) ретушью  одну или

несколько ячеек на карте. Если нужно выделить ячейки, значения которых входят в некоторый диапазон значений, то сначала этот диапазон необходимо задать на градиентной шкале внизу карты: сначала выбрать начало диапазона на шкале, нажать левую кнопку мыши для выделения (или правую кнопку мыши для снятия выделения), затем, не отпуская кнопки, переместить указатель в конечную точку диапазона и отжать кнопку мыши. Для очищения списка выделенных ячеек через контекстное меню, всплывающее по нажатию правой кнопки мыши, нужно выбрать *очистить выделенные ячейки;*

режим рисования контура в виде ломаной линии – режим, в котором границы некоторой обводятся с помощью ломаной линии;

режим установки меток – в режиме установления, удаления и правки текстовых меток для определенных ячеек СОК двойным нажатием мыши по ячейке вызывается диалоговое окно, в котором вводится / редактируется текст метки.

Сопоставление карт Кохонена с данными. И в верхнем, и в контекстном меню есть кнопка *Показать/скрыть окно данных* (*F4*), которая открывает (или закрывает) в нижней части визуализатора карт Кохонена таблицу с примерами обучающей выборки. При работе с этой таблицей доступны следующие операции:

найти ячейку на карте (*Ctrl+Enter*) – нахождение текущей ячейки, соответствующей значениям текущей записи таблицы. Однако данная команда недоступна, если выбран фильтр по ячейке;

способ отображения (*Ctrl+F12*) – визуализируется либо таблица с исходными примерами, либо форма, либо статистика;

фильтрация – выполняется фильтрация записей (примеров) в таблице 4 способами:

без фильтрации – все примеры исходного набора данных;

фильтр по ячейке – примеры, попавшие в выделенную ячейку;

фильтр по кластеру – примеры, попавшие в выбранный кластер;

фильтр по выделенному – примеры, попавшие в выделенную область.

3. Методика выполнения работы

3.1. Решение задачи кластеризации с помощью карт Кохонена на основе аналитической платформы *Deductor*

Пример. Проводится сегментация абонентов телекоммуникационной компании, необходимая для построения профилей абонентов путем выявления их схожего поведения в частоте, длительности, времени звонков и ежемесячных расходов, а также для оценки наиболее и наименее доходных сегментов. Задача нейросетевой кластеризации решается с помощью карт Кохонена на основе аналитической платформы *Deductor*.

В качестве обучающего набора данных выступает база данных клиентов телекоммуникационной компании *Абоненты.txt*, включающая записи с описанием 11 характеристик (признаков), включающих: *«Код»*, «*Возраст*», «*Среднемесячный расход*», «*Средняя продолжительность разговора*», «*Звонков днем за месяц*», «*Звонков вечером за месяц*», *«Звонков ночью за месяц», «Звонки в другие города», «Звонки в другие страны», «Доля звонков на стационарные телефоны», «Количество SMS за месяц»*.

3.1.1. Построение карт Кохонена для исходной выборки.

Файл с обучающей выборкой «*Абоненты.txt»* импортируется в программу *Deductor*. После импортирования запускается *Мастер обработки* (*F7*)*;* выбирается метод обработки данных – *Карта Кохонена*. Настройка и обучение карт проводится по шагам в соответствии с п. 2.3.

На шаге 1 настройки назначения столбцов таблицы с обучающей выборкой первому столбцу *«Код»* устанавливается *информационное* назначение, всем остальным 10 столбцам – *входное*.

На шаге 2 настройки обучающей выборки в обучающее множество включается 100% записей.

Остаются без изменений (т.е. со значениями, заданными по умолчанию) параметры карты Кохонена (шаг 3), параметры остановки обучения (шаг 4) и параметры обучения карты Кохонена (шаг 5).

На шаге 6 запускается процесс построения карт Кохонена, после завершения которого на шаге 7 проводится настройка отображений

карт Кохонена: в списке допустимых отображений карты отмечаются все 10 входных столбцов, а также среди специальных карт указываются матрицы ошибок квантования и плотности попадания; кроме того, устанавливается флажок в поле *Сглаживание цветов карты* и убирается флажок из поля *Границы кластеров*.

По окончании всех *этапов построения SOMap*-карт формируются карты Кохонена (рис. 3.10), в которых каждому входному столбцу таблицы с обучающей выборкой соответствует своя карта признака (всего 10 карт признаков).

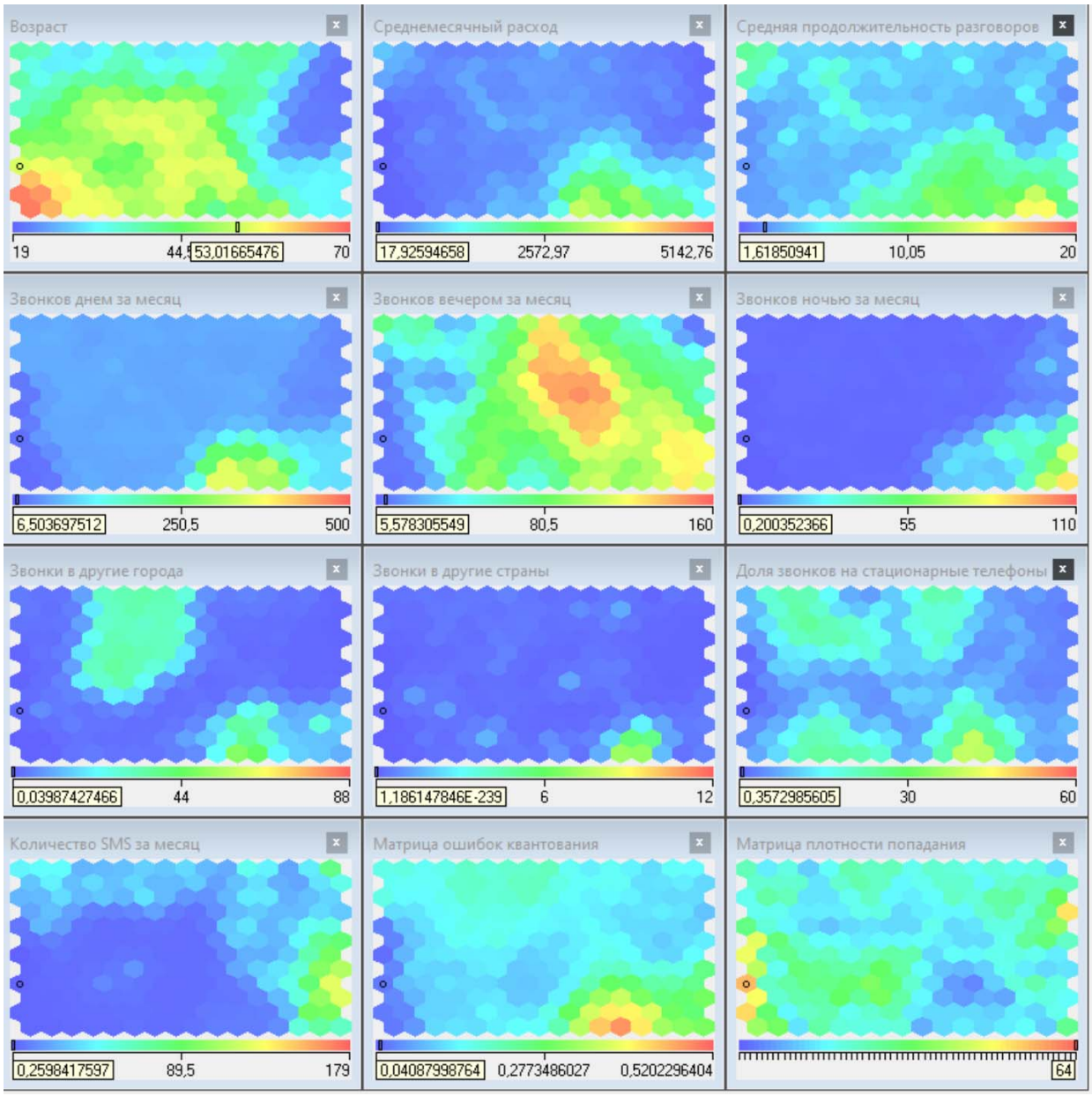


Рис. 3.10. Карты Кохонена масштаба *16×12* для сегментации абонентов

Ручная кластеризация. Если внимательно рассмотреть среди построенных карт отдельно взятую карту *Возраст*, то можно увидеть, что на ней четко выделяются области синего, желто-зеленого и также красного цвета. Принимая во внимание, что на картах признаков различными цветами показаны значения признаков с использованием тепловой шкалы, когда самые низкие значения изображаются темным цветом, а самые высокие – красным, можно сделать вывод, что на карте *Возраст* показаны три возрастные группы абонентов телекоммуникационной сети: молодежь (области синего и голубого цветов с малыми значениями), пользователи среднего возраста (наиболее обширная область желто-зеленого цвета) и абоненты старшего возраста (область оранжевого и красного цвета с высокими значениями признака).

Характеризуя значения объектов в определенных областях карт по цветам, можно построить *профили* объектов в этих областях, а для рассматриваемого примера – получить представление о профилях абонентов телекоммуникационной компании.

Например, на карте *Возраст* имеется несколько областей с малыми значениями, соответствующими группам абонентов молодого возраста: одна область темно-синего цвета, две области голубого цвета.

Если на этой карте в *режиме выделения ячеек* ретушью выделить нужные ячейки области голубого цвета с малыми значениями признаков, а затем в *режиме рисования контура* обвести эту область (рис. 3.11), т.е. *вручную* построить отдельный *кластер*, то эта же область будет отображена на всех остальных картах, что даст возможность по цвету оценить значения признаков объектов в ней.

Итак, в выделенный на картах кластер входит группа молодых людей, которые имеют средний среднемесячный расход средств на звонки (область голубовато-зеленого цвета), продолжительность разговоров выше среднего (желто-зеленый цвет), количество звонков вечером выше среднего (желто-зеленый цвет), много звонков ночью (желто-зеленый цвет), малое количество звонков в другие города и страны (сине-голубой цвет), малую долю звонков на стационарные телефоны (голубой цвет) и высокое количество SMS за месяц (желто-зеленый цвет). Эти закономерности можно также описать в виде продукционных правил *ЕСЛИ-ТО*.

3.1.2. Анализ построенных карт Кохонена.

Оценить качество построенных карт Кохонена можно лишь путем *сравнения* результатов построения *нескольких* карт при различных параметрах обучения. Основными инструментами анализа при этом становятся *специальные* карты – *матрица ошибок квантования*, отображающая насколько хорошо обучена сеть (средние расстояния до центра ячейки должны быть минимальны), и *матрица плотности попадания*, показывающая количество объектов, попавших в ячейку. Использование этих инструментов позволяет из ансамбля построенных карт выбрать наилучший вариант для решения задачи кластеризации объектов.

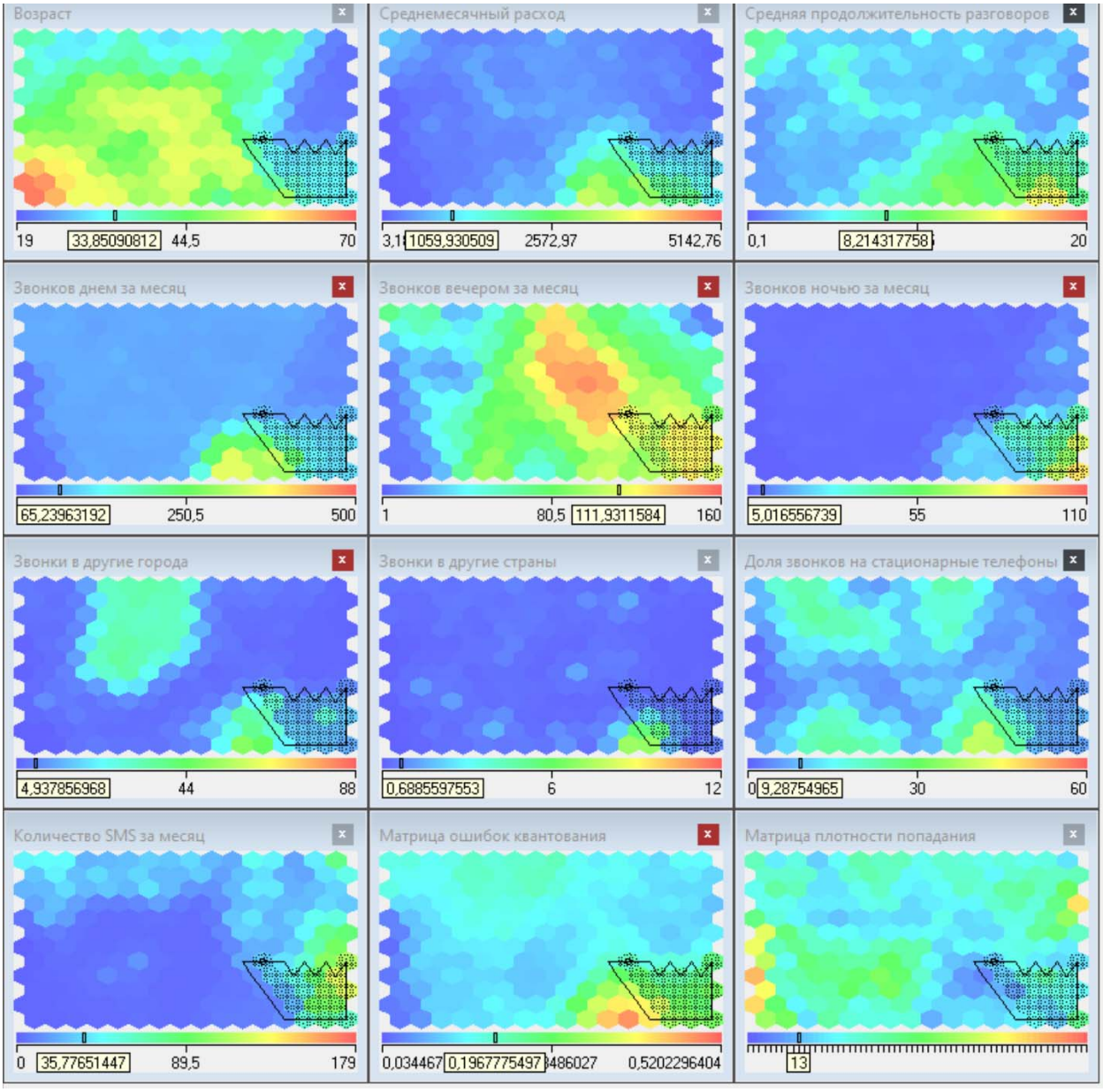


Рис. 3.11. Карты Кохонена *16×12* с выделением отдельного кластера

Для рассматриваемого примера сегментации абонентов в построенных картах Кохонена (см. рис. 3.10–3.11) с масштабом *16×12*, т.е. с 192 ячейками, максимальные значения ошибок квантования равны 0,52, а максимальные значения плотности попадания равны 64.

Строится еще одна карта Кохонена с увеличенным масштабом *24×18* (т.е. всего 432 ячеек – больше в 2,2 раза) и с измененным способом инициализации («из обучающего множества») весов нейронов для снижения вероятности образования пустых ячеек. В новой карте Кохонена (рис. 3.12) максимальное значение ошибок квантования равно 0,47, а максимальное число плотности попаданий в одну ячейку – 48; значения на обеих специальных картах снизились, что говорит об улучшении качества построенных СОК.

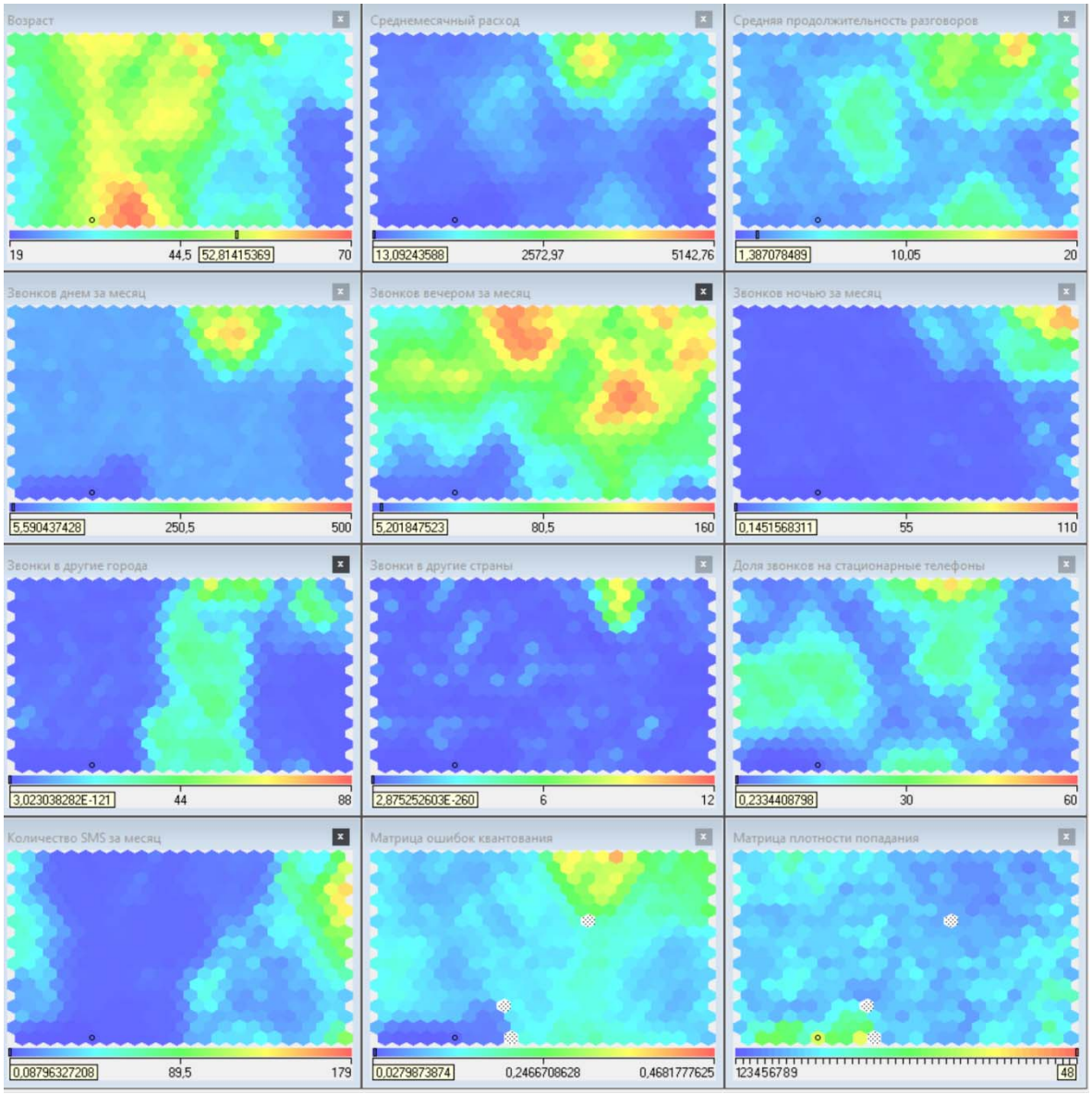


Рис. 3.12. Карты Кохонена масштаба *24×18* для сегментации абонентов

Таким образом, из двух построенных карт Кохонена для дальнейшего анализа выбирается карта с увеличенным масштабом, так как она имеет лучшие показатели качества кластеризации.

3.1.3. Описание кластеров, построенных вручную.

Исследуя построенные карты на рис. 3.12, можно выделить примерно 7 областей, в целом описывающих профили абонентов (табл. 3.1), каждый из которых характеризуется определенным стилем и активностью общения по телефону, – это три вида молодежных групп, три различные группы старшего возраста, а также одна большая группа абонентов среднего возраста.

Таблица 3.1

Описание профилей абонентов, выделенных в областях СОК Кохонена

№	Возраст	Описание профиля абонентов	Краткая характеристика
1	молодой	Высокие значения признаков «звонков вечером за месяц», «звонков ночью за месяц», «количество SMS за месяц». Абоненты тратят на разговоры больше денег, чем другая молодежь	активная молодежь
2	молодой	Малые значения признаков «звонков вечером за месяц», «звонков ночью за месяц», «количество SMS за месяц». Малые ежемесячные расходы.	молодежь, мало пользующаяся услугами связи
3	молодой	Умеренные значения признаков «звонков вечером за месяц», «звонков ночью за месяц», «количество SMS за месяц». Преимущественно вечерние разговоры.	основная молодежь
4	зрелый	Высокие значения почти всех признаков кроме SMS, высокие значения по звонкам в другие города и страны. Месячные расходы на связь этой категории абонентов самые высокие.	VIP-клиенты
5	пенсионный	Малые значения всех признаков, малые ежемесячные расходы.	«малоговорящие»
6	зрелый и пенсионный	Умеренные значения признака «звонков вечером за месяц», не используется SMS-сервис.	активная группа старшего возраста
7	средний	работающие люди среднего возраста	

Для статистического анализа данных о профилях абонентов, условно выделенных по областям, необходимо войти в режим *выделения* ячеек и отметить заданные области. Затем активизировать режим *Показать окно данных*; установить в нем *Фильтр по выделенному*, а затем в окне данных переключиться в *Способ отображения – Статистика* и выбрать параметр *Среднее*.

Для трех первых областей (молодых клиентов компании) средние значения признаков представлены в табл. 3.2.

Таблица 3.2

Статистика по молодым клиентам компании

Метка входного столбца	Среднее значение		
	1	2	3
Количество значений	370	157	890
Возраст	28,23	21,8	27,63
Среднемесячный расход	898,77	95,9	519,4
Средняя продолжительность разговоров	6,83	2,58	4,66
Звонков днем за месяц	6,83	39,13	58,7
Звонков вечером за месяц	103,67	8,48	81,02
Звонков ночью за месяц	43,04	6,78	7,71
Звонки в другие города	10,42	1,65	7,57
Звонки в другие страны	0,36	0,24	0,33
Доля звонков на стационарные телефоны	6,05	3,21	9,93
Количество SMS за месяц	90,17	44,93	29,15
Мощность кластера	*370 (8%)*	*157 (3,5%)*	*890 (20%)*
Прибыльность кластера	*281 413 (12%)*	*49 863 (2,2%)*	*458 763 (20%)*

В последних двух строках таблицы показаны *мощность* кластера, рассчитанная из количества значений в кластере по отношению к общему количеству (4492) исходных объектов, а также *прибыльность* кластера – сумма по столбцу *Среднемесячный расход*.

Анализ статистических показателей трех групп молодых клиентов показывает, что наиболее прибыльной является группа молодежи №3 со средним возрастом 27,6 лет, которая ежемесячно платит за услуги связи в среднем 519,4 руб., в основном совершает звонки днем, но и вечерних разговоров имеет также много. Наименее прибыльной является самая юная группа молодежи со средним возрастом в 21,8 лет, в целом мало пользующаяся услугами связи – с очень малым числом звонков, но достаточно большим количеством SMS за месяц; среднемесячный расход группы – 95,9 руб.

3.1.4. Автоматическая кластеризация.

В программе *Deductor* имеется возможность автоматической кластеризации, т.е. группировки объектов в кластеры. Для этого в верхнем или контекстном меню выбирается пункт *Настроить кластеры*; в появившемся диалоговом окне при установленном флажке *Автоматически определить количество кластеров* включается алгоритм *G-means*, который формирует множество кластеров.

Можно принудительно установить в поле *Фиксированное количество кластеров* и другое количество кластеров, например, 7 кластеров (как было построено вручную). Затем в меню *Настроить отображения* нужно установить флажок в поле *Границы кластеров* (рис. 3.13).

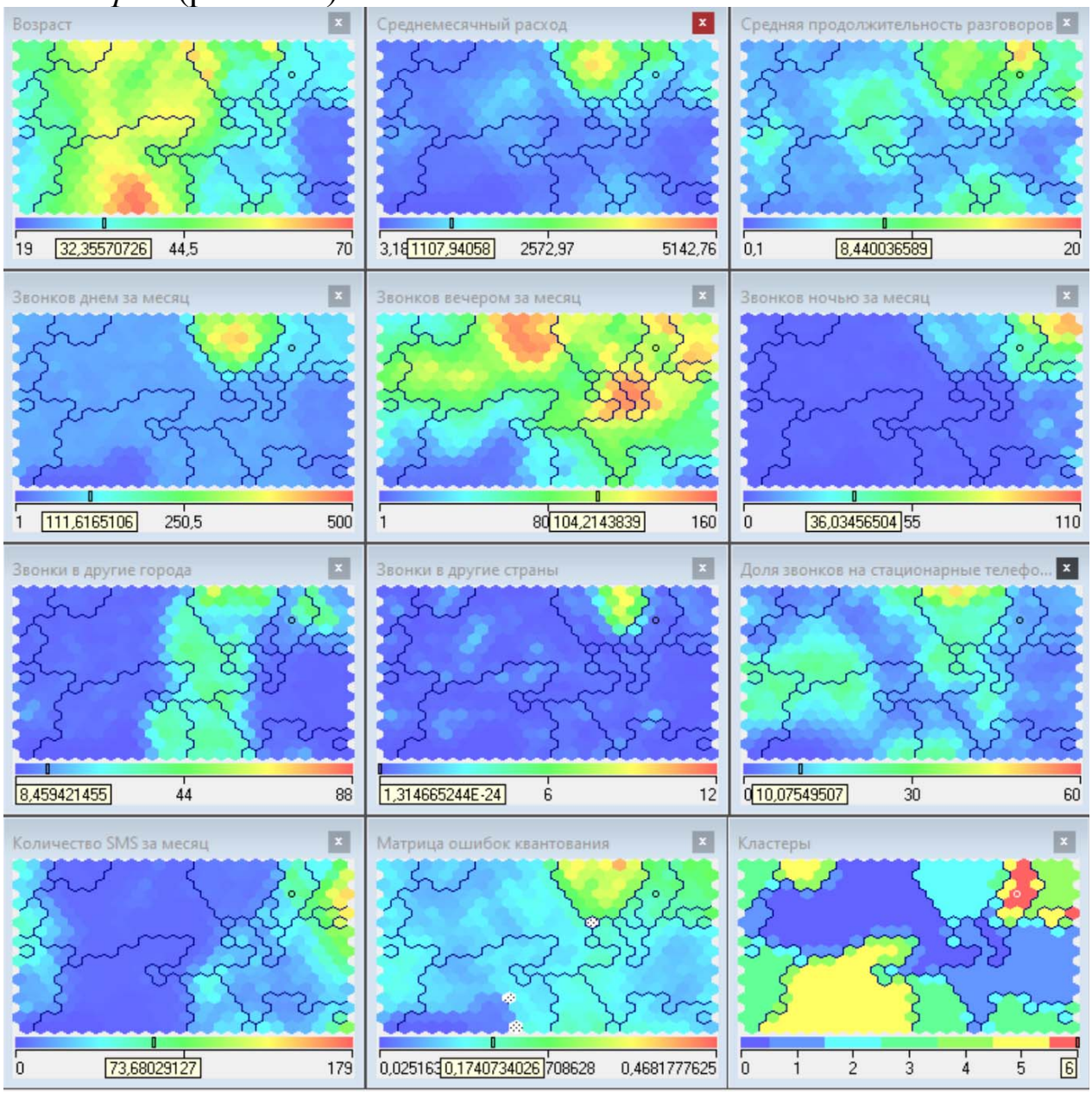


Рис. 3.13. Карты Кохонена *24×18* с автоматическим делением на 7 кластеров

Для анализа карт Кохонена с автоматической кластеризацией удобно использовать специальную карту *Кластеры,* настройка которой осуществляется с помощью Мастера визуализации (*F5*).

Результаты построения кластеров в картах Кохонена показали, что встроенный алгоритм кластеризации создал кластеры иначе, чем при ручной кластеризации. Например, объединились кластеры *Зрелый и пенсионный возраст*, по-другому сформировались группы *Молодежь* и *Люди среднего возраста*. Тем не менее, автоматическая кластеризация в *Deductor* имеет одно важное преимущество перед ручной: при автоматическом разбиении появляется расчетный столбец *Номер кластера*, который можно использовать в инструменте *Что-если* для кластеризации новых объектов (рис. 3.14).

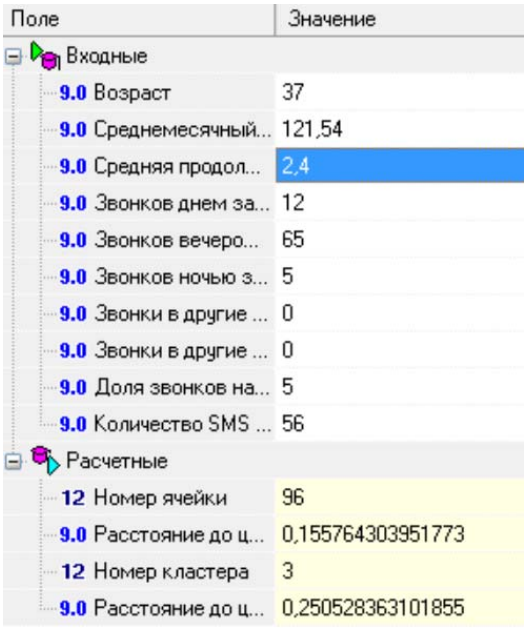


Рис. 3.14. Инструмент анализа *Что-если* пакета *Deductor*

3.1.5. Визуализаторы *Матрица сравнения* и *Связи кластеров.*

Для того чтобы определить сходство автоматически построенных кластеров используется визуализатор *Матрица сравнения* (рис. 3.15); с помощью цветовой шкалы в матрице показаны попарные сравнения близких по значению объектов кластеров в процентах. Например, степень сходства кластеров №1 и №4 составляют 47,98%. Для каждой ячейки матрицы сравнения справа указываются *Профили кластеров*, которые позволяют исследовать статистические характеристики кластеров, которые не вошли в визуализатор карт Кохонена: мощность (число записей, попавших в кластер), среднее и др.; особый интерес в этом визуализаторе

представляют две характеристики – значимость и доверительный интервал.

Визуализатор *Связи кластеров* (рис. 3.16) также показывает сходство кластеров с помощью цветовой шкалы. Кластеры №4, №5, №6 обладают наименьшим количеством схожих элементов – это показывает цвет линии связи между кластерами (синий), а №0 и №1, №6 – имеют сильные связи. При использовании данного визуализатора можно находить не только сильные или слабые связи, но и задавать требуемые критерии связи с помощью числовых диапазонов, а также менять кластеры друг с другом нажатием на кнопку  .

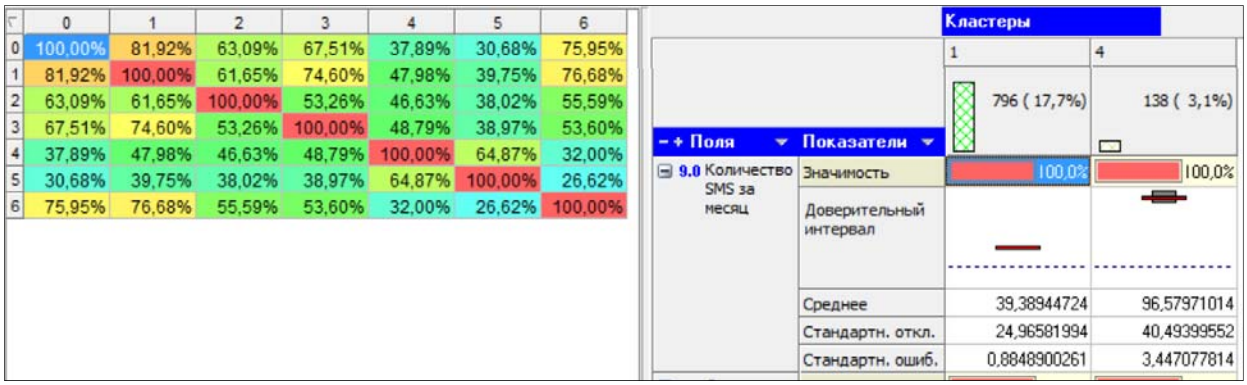


Рис. 3.15. Матрица сравнения с профилями кластеров карт Кохонена

Рис. 3.16. Связи кластеров карт Кохонена

3.1.6. Решение задачи прогнозирования с помощью карт Кохонена.

Поскольку карты Кохонена основаны на алгоритме обучения без учителя, то *выходные* столбцы для обучения не задаются, однако опционально в методе могут быть выходные поля, и в этом случае

будет решаться задача *прогнозирования*. Для решения этой задачи на шаге 1 настройки назначения столбцов таблицы с обучающей выборкой в качестве выходного столбца нужно задать столбец, который при кластеризации участвовать уже не будет. Это дает возможность после обучения карт решать задачу классификации (если выходное поле – дискретное) или регрессии (если выходное поле имеет непрерывное значение).

Например, если столбец *Возраст* указать как выходной, то после построения карты Кохонена появляется возможность определять возраст новых абонентов по остальным параметрам: числу звонков, среднемесячному расходу и т.д.

3.2. Решение задачи кластеризации с помощью карт Кохонена в пакете *Neural Networks Toolbox* системы *Matlab*

Другой способ решения задачи нейросетевой кластеризации осуществляется с использованием пакета *Neural Networks Toolbox* системы *Matlab*. Основным преимуществом использования системы *Matlab* является возможность применения различных методов обучения – например, традиционного метода и метода выпуклой комбинации (п. 2.1).

Пример. Пусть имеется исходное множество P, состоящее из n двумерных объектов, заданное в формате: $P=[a_1\ a_2\ a_3\ ...\ a_n;\ b_1\ b_2\ b_3..b_n]$, где a_i, b_i – первая и вторая координаты i-го объекта. В командной строке *Matlab* вводятся координаты объектов, например:

```
P=[.3 .25 .7 .6 .2 .15 .75 .65 .25 .2 .8 .8 .1 .2 .7 .72 .3 .3 .7 .68;
.2 .6 .6 .12 .1 .7 .8 .15 .3 .8 .65 .3 .15 .65 .75 .1 .3 .7 .8 .2];
```

Необходимо разбить объекты на 4 кластера по 5 объектов в каждом кластере (допущением является одинаковое количество объектов в каждом кластере).

3.2.1. Решение задачи нейросетевой кластеризации традиционным методом

Шаг 1. Создается слой Кохонена, проводится его инициализация и обучение.

Создание слоя Кохонена осуществляется с помощью функции *net=newc*(*bounds, clusters, m*), где *bounds* – диапазон изменения значений координат объектов (от минимального до максимального);

clusters – количество кластеров, на которые необходимо разбить объекты; *m* – коэффициент скорости обучения.

```
net= newc([0 1;0 1], 4, 0.1);
```

Затем проводится инициализация весов нейросети и обучение слоя Кохонена на основе массива входных объектов, время обучения 150 эпох:

```
net =init(net);
net.TrainParam.epochs=150;
net=train(net,P);
```

Шаг 2. Определяется номер кластера для каждого объекта множества *P*.

Сначала проводится моделирование работы слоя Кохонена, а затем выход нейронной сети конвертируется в номер класса:

```
Y = sim(net, P);
Yc = vec2ind (Y);
```

Шаг 3. Определяются координаты центров кластеров, соответствующих весовым коэффициентам нейронов.

Массив весовых коэффициентов для слоя 1:

```
w=net.iw{1};
```

Шаг 4. Для вывода на экран (рис. 3.17) объектов с разделением кластеров по цветам выполняется *подпрограмма*, где *N* – общее количество объектов, *clusters* – количество кластеров, *i* – номер объекта, *r, b, g* – цвета объектов в кластере.

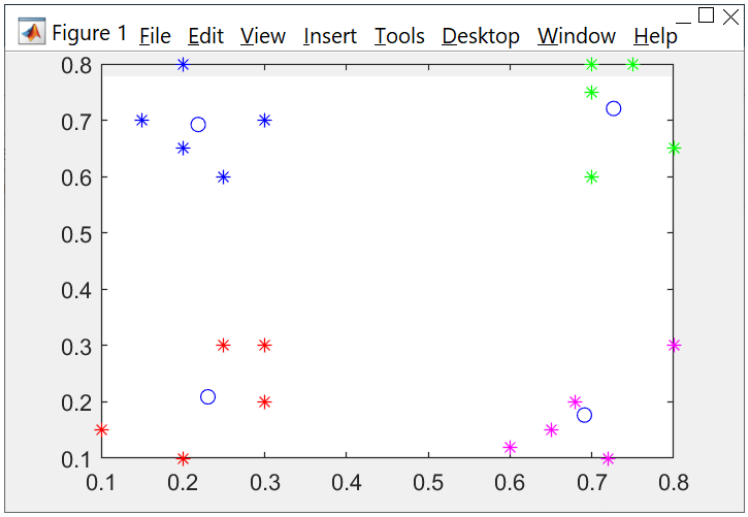


Рис. 3.17. Кластеризация традиционным методом

```
N=20;
clusters=4;
i=1;
while i<=N
    if Yc(i)==1
        plot( P(1,i),P(2,i),'*r');%звездочка красного цвета
    end;
    if   Yc(i)==2
        plot( P(1,i),P(2,i),'*b');% звездочка синего цвета
    end;
    if Yc(i)==3
        plot( P(1,i),P(2,i),'*m');% звездочка фиолетового цвета
    end;
    if Yc(i)==4
        plot( P(1,i),P(2,i),'*g');% звездочка зеленого цвета
    end;
hold on;
    i=i+1;
% Вывести на экран центры кластеров в виде кружков синего цвета:
plot( w(:,1),w(:,2),'ob');
end;
```

3.2.2. Решение задачи нейросетевой кластеризации методом выпуклой комбинации.

Создание карты Кохонена осуществляется с помощью функции:

```
Net1=newsom([0 1; 0 1],[1 4]);
```

Согласно методу выпуклой комбинации (3.6) в начале обучения коэффициент обучения очень мал, поэтому входные векторы почти совпадают с векторами весов и первоначальной топологией карты Кохонена является одна единственная точка (рис. 3.18, *а*):

```
plotsom (Net1.iw{1,1},  Net1.layers {1}.distances);
```

Затем указывается количество эпох и запускается процесс обучения:

```
Net1.trainParam.epochs = 100;
net=train(Net1,P);
```

Результаты обучения (рис. 3.18, *б*) показывают, что сформировалась одномерная карта Кохонена: точками являются центры кластеров, а линиями – кратчайшие расстояния между ними:

```
plotsom (net.iw{1,1}, net.layers {1}.distances);
```

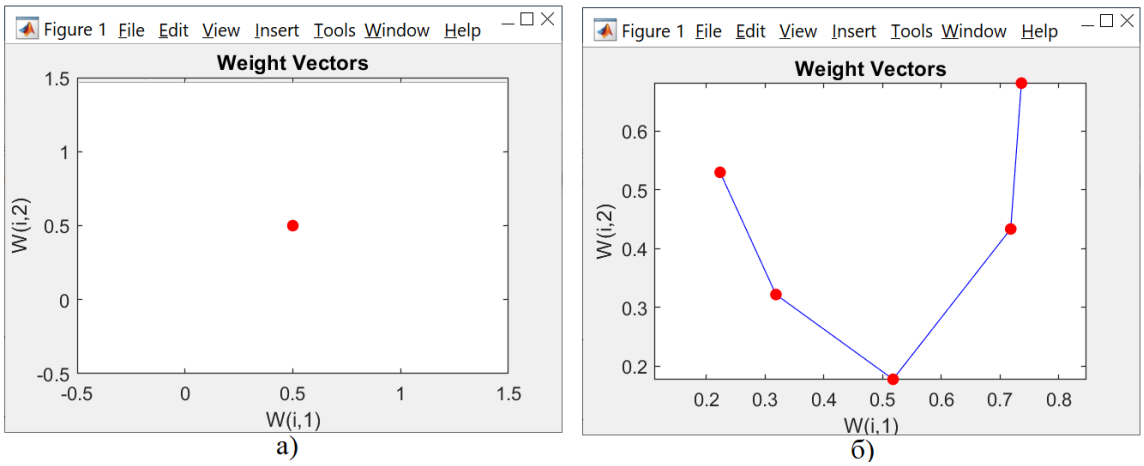


Рис. 3.18. Кластеризация методом выпуклой комбинации

Таким образом, для решения задачи нейросетевой кластеризации с помощью пакета *Neural Networks Toolbox* системы *Matlab* могут применяться различные методы обучения карт Кохонена и применяться тонкая настройка параметров обучения.

Порядок выполнения работы

1. Изучить методику нейросетевой кластеризации с помощью самоорганизующихся карт Кохонена в пакете *Deductor*.

2. Провести нейросетевой анализ данных об абонентах телекоммуникационной компании (*Абоненты.txt*) с помощью нейронной сети Кохонена в приложении *Deductor*.

Провести экспериментальные исследования результатов обучения СОК, изменяя следующие параметры:

а) вид и количество полей БД, используемых при обучении;

б) количество эпох грубой и точной подстройки;

в) начальную скорость обучения;

г) радиус обучения;

д) вид функции скорости обучения;

е) способ инициализации весов векторов;

ж) форму ячеек.

Описать профиль абонентов средневозрастной группы. Дать оценку абонентам, попавшим, например, в ячейку 48.

Решить задачу прогнозирования, построив карту Кохонена для сегментации абонентов и установив поле *Возраст* выходным. Насколько сильно изменилась карта?

Провести эксперимент в визуализаторе *Что-если*: ввести данные в поля *Количество звонков, Среднемесячный расход* и т.д. и спрогнозировать возраст.

3. Провести нейросетевой анализ в пакете *Deductor* с помощью карт Кохонена для полученного варианта задания**:** дать характеристики нескольким выбранным (наиболее представительным) кластерам и ячейкам по обучающей и рабочей выборкам. Для этого создать обучающую выборку объемом в 30 объектов.

Сделать выводы по результатам анализа сформированных кластеров.

4. Построить карты Кохонена и провести нейросетевой анализ в пакете *Neural Networks Toolbox* системы *Matlab* для варианта задания.

5. Сравнить характеристики сформированных кластеров, полученных по результатам анализа на основе пакетов *Deductor* и *Neural Networks Toolbox* системы *Matlab.*

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

1. Название и цель работы.
2. Результаты нейросетевого анализа об абонентах телекоммуникационной компании в пакете *Deductor*. Описание полученных кластеров, в том числе средневозрастной группы. Решение задачи прогнозирования.
3. Исходные данные для полученного варианта задания – описание и состав обучающей выборки, результаты нейросетевого анализа в пакете *Deductor*. Характеристика выбранного объекта (ячейки) в кластере по рассматриваемым картам для полученного варианта задания.
4. Текст программы на языке *Matlab* для полученного варианта задания. Результаты работы программы; результаты нейросетевого анализа в пакете *Neural Networks Toolbox* системы *Matlab*.
5. Выводы.

Варианты заданий

Обучающая выборка содержит информацию о работниках некоторой компании. Каждый объект характеризуется пятью признаками: *Возраст работника компании*; *Общий стаж*; *Стаж*

работы по специальности; *Квалификация*; *Профессионализм*. Каждое поле измеряется величиной от 0 до 1.

Вариант 1.

Большое количество работников старшего возраста с низкой квалификацией и умеренным профессионализмом. Большое количество работников молодых с высокой квалификацией и низким профессионализмом.

Вариант 2.

Большое количество работников среднего возраста с высокой квалификацией и умеренным профессионализмом. Большое количество работников молодых с низкой квалификацией и средним профессионализмом.

Контрольные вопросы

1. Охарактеризуйте алгоритм обучения нейронной сети без учителя.
2. Для какой задачи используется нейронная сеть Кохонена? Опишите структуру этой сети.
3. В чем заключается конкуренция при функционировании сети Кохонена?
4. Изложите алгоритм обучения сети Кохонена.
5. Какие процедуры используются для определения нейрона-победителя в сети Кохонена?
6. В чем состоит преимущество метода выпуклой комбинации?

Список литературы

1. Хайкин С. Нейронные сети: полный курс: пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.
2. Паклин Н., Орешков В. Бизнес-аналитика. От данных к знаниям. СПб.: Питер, 2013. 704 с.
3. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского. М.: Горячая линия - Телеком, 2004. 452 с.
4. Ярушкина Н.Г. Основы теории нечетких и гибридных систем: учеб. пособие. М.: Финансы и статистика, 2004. 320 с.

5. Осовский С. Нейронные сети для обработки информации / пер. с польского И. Д. Рудинского . М.: Финансы и статистика, 2002. 344 с.

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

7. Барский А. Б. Нейронные сети: распознавание, управление, принятие решений. М.: Финансы и статистика, 2004. 176 с.

8. 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. 456 с.

9. Deductor. Руководство аналитика. Версия 5.3 // https://basegroup.ru/system/files/documentation/guide_analyst_5.3.0.pdf.

10. Бураков, М.В. Нейронные сети и нейроконтроллеры: учеб. пособие. СПб.: ГУАП, 2013. 284 с.

Лабораторная работа № 4

НЕЙРО-НЕЧЕТКОЕ МОДЕЛИРОВАНИЕ В СИСТЕМЕ *ANFIS MATLAB*

1. Цель и задачи работы

Целью работы является закрепление умений и навыков моделирования нейросетевых нечетких сетей в системе *Anfis Matlab*.

Задачами работы являются формирование умений разработки структуры нейро-нечетких сетей, навыков обучения нейро-нечетких сетей на основе парадигмы обучения с учителем в *Anfis Matlab*.

2. Теоретические сведения

2.1. Нейросетевые нечеткие системы

Нейросетевые нечеткие (нейро-нечеткие) системы (ННС) – это гибридные системы, которые комбинируют методы искусственных нейронных сетей (ИНС) и нечеткой логики.

Эффективность *нейронных сетей* определяется их аппроксимирующей способностью: на основе обучения ИНС можно описать любую непрерывную функциональную зависимость без предварительной аналитической работы по выявлению правил зависимости выхода от входа; однако недостатком ИНС является невозможность в явном виде объяснить функциональную зависимость между входом и выходом исследуемого объекта.

Преимуществом *нечетких* систем является то, что знания в этих системах представляются в форме продукционных правил *ЕСЛИ – ТО*.

ННС представляют собой многослойные ИНС специальной структуры без обратных связей, в которой используются обычные сигналы, весовые коэффициенты связей и функции активации. Основная концепция модели ННС заключается в том, чтобы использовать механизм обучения нейронных сетей для определения параметров функций принадлежности (ФП) конкретных систем нечеткого вывода (рис. 4.1). ФП обычно аппроксимируются ИНС из данных обучающей выборки.

Рис. 4.1. Общая структура гибридной нейро-нечеткой системы

Система *ANFIS* (*Adaptive-Network-Based Fuzzy Inference System*) является одной из первой схем гибридных ННС; это адаптивная нейросеть, основанная на системе нечеткого вывода, реализующая нечёткую систему Такаги-Сугено. Система *ANFIS* представляет собой пятислойную нейросеть прямого распространения (рис. 4.2).

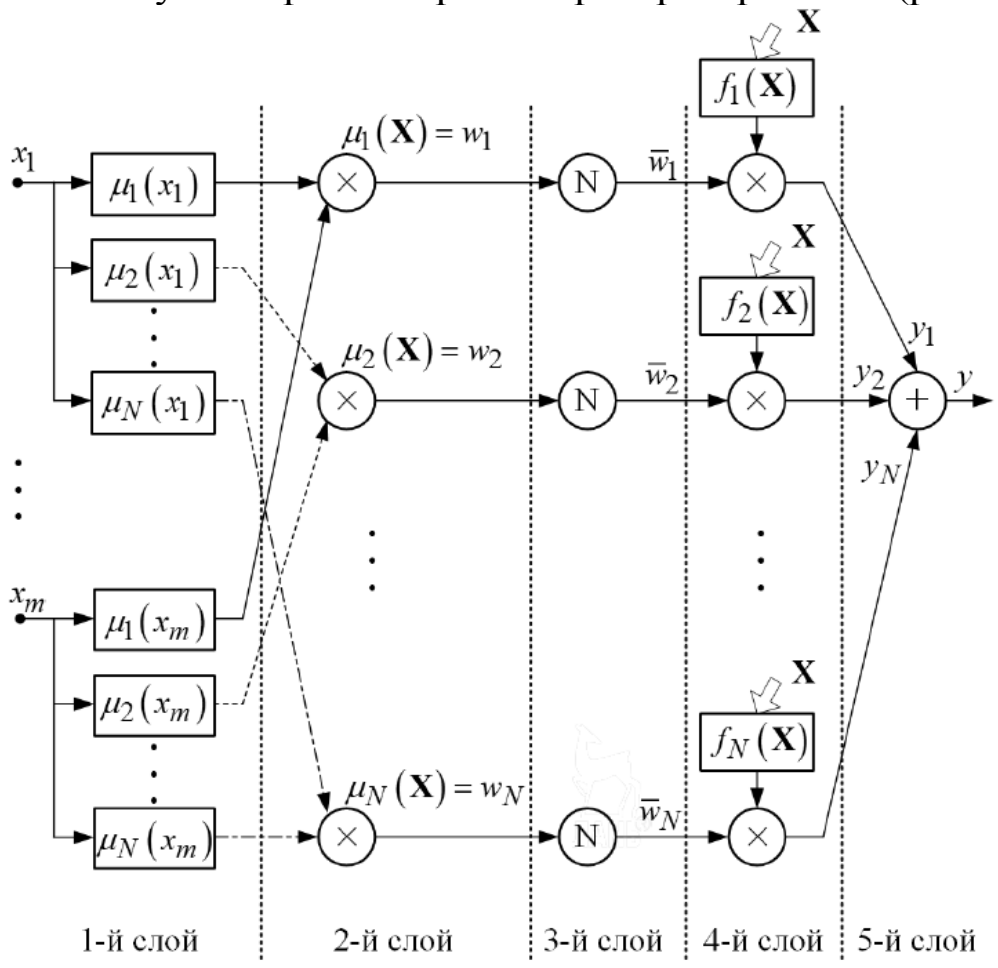


Рис. 4.2. Структура *ANFIS*-сети

Слои ННС выполняют следующие функции (заметим, что входы сети в отдельный слой не выделяются).

Слой 1 осуществляет фаззификацию каждого элемента вектора входного сигнала сети $X = [x_1, x_2, \dots, x_m]^T$. Для N нечетких правил вводятся условия:

ЕСЛИ $x_1 = A_{11}$ И $x_2 = A_{21}$ И… И $x_m = A_{m1}$

…

ЕСЛИ $x_1 = A_{1N}$ И $x_2 = A_{2N}$ И… И $x_m = A_{mN}$.

Первый слой является *параметрическим*, поскольку в нем определяются параметры ФП $\mu_r(x_i), r = 1,2, \dots, N$ при обучении сети: ФП для каждой переменной x_i и каждого правила r описывается обобщенной функцией Гаусса:

$$\mu_r(x_i) = exp\left[-\left(\frac{x_i - c_i^r}{\sigma_i^r}\right)^{2b_i^r}\right],$$

где c_i^r, σ_i^r, b_i^r – параметры функции, подлежащие адаптации в процессе обучения сети: изменение параметра c_i^r соответствует смещению ФП по горизонтальной оси; параметр σ_i^r отвечает за широту функции; параметр b_i^r – за форму кривой, которая при соответствующем подборе данного параметра может быть задана либо как функция Гаусса, либо как треугольная или трапецеидальная функция.

Слой 2 осуществляет *агрегирование* отдельных переменных x_i и определяет *предпосылки* каждого из N нечетких правил:

ЕСЛИ $x_1 = A_{11}$ И $x_2 = A_{21}$ И… И $x_m = A_{m1}$

…

ЕСЛИ $x_1 = A_{1N}$ И $x_2 = A_{2N}$ И… И $x_m = A_{mN}$.

Каждый узел этого слоя соответствует одному нечеткому правилу. Выходами узла является степень выполнения r-го нечеткого правила w_r, которая рассчитывается на основе алгебраического произведения:

$$w_r = \mu_r(X) = \prod_{i=1}^{m} exp\left[-\left(\frac{x_i - c_i^r}{\sigma_i^r}\right)^{2b_i^r}\right].$$

Слой 3 – нормализующий слой, в котором неадаптивные узлы рассчитывают относительную степень $\overline{w}_r$ выполнения нечеткого правила r:

$$\overline{w}_r = \frac{w_r}{\sum_{i=1}^{N} w_i}.$$

Слой 4 формирует *заключения* нечетких правил, т.е. определяет вклад каждого нечеткого правила в выход сети, Типовое представление *N* нечетких правил в модели Такаги-Сугено имеет вид:

ЕСЛИ $x_1 = A_{11}$ И … И $x_m = A_{m1}$ *ТО* $f_1(X) = p_{10} + \sum_{i=1}^{m} p_{1i} \cdot x_i$

…

ЕСЛИ $x_1 = A_{1N}$ И … И $x_m = A_{mN}$ *ТО* $f_N(X) = p_{N0} + \sum_{i=1}^{m} p_{Ni} \cdot x_i$.

Заключения правил модели Такаги-Сугено в большинстве случаев содержат линейные функции, хотя могут использоваться также и произвольные нелинейные функции. В четвертом слое рассчитывается произведение функций $f_r(X)$ и значений весовых коэффициентов $\overline{w}_r$, сформированных в предыдущем слое. Таким образом, определяется вклад каждого нечеткого правила в выход сети по формуле:

$$y_r = \overline{w}_r \cdot f_r(X).$$

Этот слой – *параметрический*, т.к. в нем коэффициенты $f_r(X)$ изменяются в процессе адаптации (обучения) сети.

Слой 5 выполняет агрегирование результата, полученного по разным нечетким правилам и формирует управляющий сигнал y. Агрегирование состоит в суммировании вкладов всех правил:

$$y = \sum_{r=1}^{N} y_r = \sum_{r=1}^{N} (p_{r0} + \sum_{i=1}^{m} p_{ri} \cdot x_i) \cdot \frac{\prod_{i=1}^{m} exp\left[-\left(\frac{x_i - c_i^r}{\sigma_i^r}\right)^{2b_i^r}\right]}{\sum_{l=1}^{N} \prod_{i=1}^{m} exp\left[-\left(\frac{x_i - c_i^l}{\sigma_i^l}\right)^{2b_i^l}\right]}.$$

Таким образом, сложные нелинейные поверхности можно аппроксимировать с помощью множества плоских линейных сегментов, каждый из которых задается одним нечетким правилом модели Такаги-Сугено; *ANFIS*-сеть содержит два адаптивных слоя (1 и 4), параметры которых уточняются при обучении ННС.

На рис. 4.3 представлен пример структуры сети *ANFIS* с двумя входными переменными x_1 и x_2, выходной переменной y и двумя нечеткими правилами.

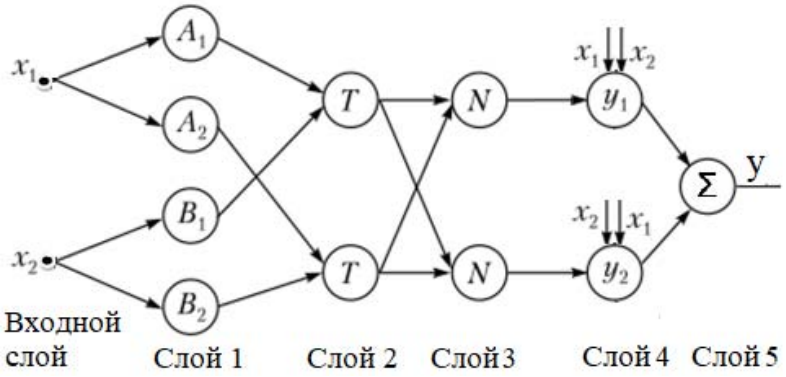


Рис. 4.3. Пример *ANFIS*-сети

Гибридные нейро-нечеткие сети позволяют разрабатывать и представлять модели систем в форме правил нечетких продукций, которые обладают наглядностью и легкостью содержательной интерпретации. Возможность построения правил нечетких продукций на основе нейросетевых методов делает процесс построения моделей более удобным и менее трудоемким.

2.2. Нейро-нечеткое моделирование в *Anfis Matlab*

Реализация адаптивных ИНС, основанных на системах нечеткого вывода (СНВ), в пакете *Matlab* проводится с помощью *Anfis*-редактора, который позволяет автоматически синтезировать ННС на основе экспериментальных данных. Редактор *Anfis* реализует создание моделей систем нейро-нечеткого вывода и обучение ННС, визуализирует структуру ННС.

Anfis-редактор в *Matlab* загружается в режиме командной строки с помощью ввода имени *anfisedit***.** В результате выполнения этой команды отображается окно редактора (рис. 4.4), которое содержит основное меню *File*, *Edit* и *View*, область визуализации, область свойств *ANFIS info*, области (панели) загрузки данных, создания исходной СНВ, обучения, тестирования, вывода текущей информации, а также кнопки *Help* и *Close*, которые позволяют вызвать окно справки и закрыть *Anfis*-редактор.

В меню *File* запуск команды *New FIS*… создает новую СНВ. При этом может быть выбран тип создаваемой системы: *Mamdani* или *Sugeno*.

В меню *Edit* открываются: *FIS*-редактор (команда *FIS Properties*); редактор ФП *Membership Functions*; редактор базы знаний *Rules*.

Через меню *View* по команде *Rules* открывается окно визуализации нечеткого логического вывода, а по команде *Surface* – окно вывода поверхности «входы-выход» СНВ.

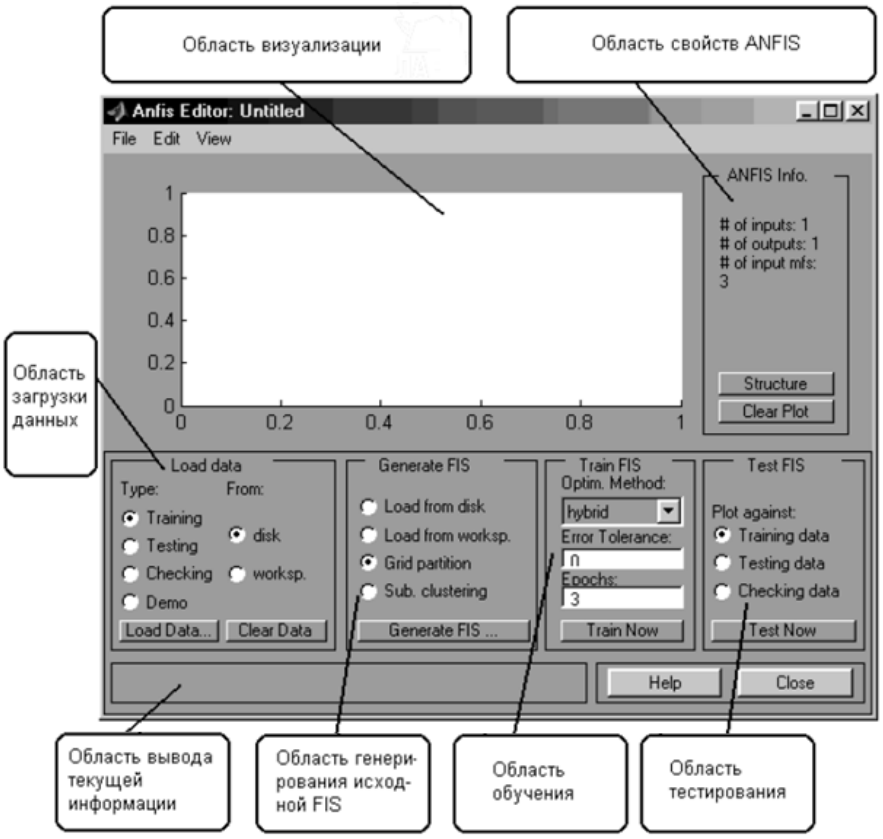


Рис. 4.4. Основное окно *Anfis*-редактора

В *области визуализации* выводятся либо кривая обучения, построенная в виде графика зависимости ошибки обучения от порядкового номера итерации; либо экспериментальные данные и результаты моделирования при загрузке данных и при тестировании системы.

Экспериментальные данные и результаты моделирования выводятся в виде множества точек в двумерном пространстве. При этом по оси абсцисс откладывается порядковый номер строки данных в выборке (обучающей, тестирующей или контрольной), а по оси ординат – значение выходной переменной для данной строки выборки. Используются следующие маркеры:

- голубая точка (.) – тестируемая выборка;
- голубая окружность (o) – обучающая выборка;
- голубой плюс (+) – контрольная выборка;

– красная звездочка (*) – результаты моделирования.

В *области свойств ANFIS info* выводится информация о количестве входных и выходных переменных, функций принадлежностей для каждой входной переменной, а также о количестве строк в выборках. В этой области расположены кнопки *Structure* и *Clear Plot*.

Нажатие кнопки *Structure* открывает структуру нейро-нечеткой сети. На рис. 4.5 приведен пример ННС, содержащей четыре входных переменных и одну выходную; для оценки каждой из входных переменных используется по четыре лингвистических терма

Нажатие *Clear Plot* очищает область визуализации.

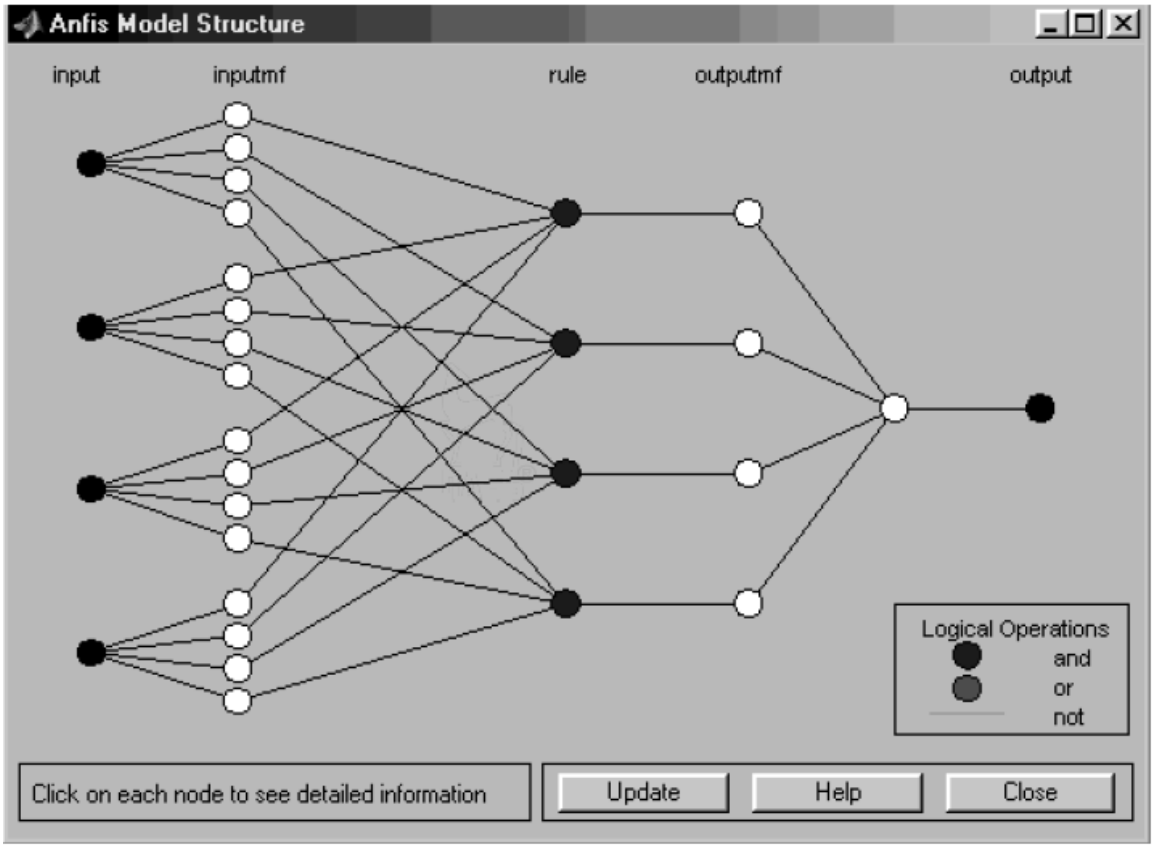


Рис. 4.5. Пример структуры нейро-нечеткой сети

Область загрузки данных (*Load data*) включает:

– меню выбора типа данных (*Type*): *Traning* – обучающая выборка; *Testing* – тестирующая выборка; *Checking* – контрольная выборка; *Demo* – демонстрационный пример;

– кнопку запуска загрузки данных *Load Data*…, при нажатии на которую открывается либо диалоговое окно выбора файла, если

загрузка данных происходит с диска, либо окно ввода идентификатора выборки, если загрузка данных происходит из рабочей области;

– кнопку запуска очистки данных *Clear Data*.

В течение одного сеанса работы *ANFIS*-редактора можно загружать данные только одного формата, т.е. количество входных переменных в выборках должно быть одинаковым.

В *области создания* исходной СНВ (*Generate FIS*) расположено меню выбора способа его создания:

– *Load from disk* – загрузка системы с диска с помощью стандартного диалогового окна открытия файла;

– *Load from worksp* – загрузка системы из рабочей области *Matlab* с помощью стандартного диалогового окна ввода идентификатора СНВ.

При генерации СНВ возможны два варианта выбора

1) *решетчатое разбиение Grid partition,* при котором функции принадлежности нечетких термов равномерно распределяются внутри универсума данных. При выборе данной опции в диалоговом окне ввода параметров указывается количество термов для каждой входной переменной и тип ФП для входных переменных.

2) *субтрактивная кластеризация Sub. clustering*, при котором число кластеров определяется во время работы на основе исходного распределения данных. При выборе данной опции в появившемся диалоговом окне вводятся параметры:

– *Range of influence* – уровни влияния входных переменных;

– *Squash factor* – коэффициент подавления;

– *Accept ratio* и *Reject ratio* – коэффициенты, устанавливающие во сколько раз потенциал данной точки должен быть выше / ниже потенциала центра первого кластера для того, чтобы при *Accept ratio* центром одного из кластеров была назначена рассматриваемая точка, а при *Reject ratio* рассматриваемая точка была исключена из возможных центров кластеров.

Оценка качества построения СНВ проводится только путем сравнения нескольких вариантов СНВ с различными параметрами. Предпочтение следует отдать тому варианту, при котором результаты тестирования обученной сети показывают меньшую ошибку.

В *области обучения Train FIS* расположено поле выбора метода оптимизации *Optim. method*: метод обратного распространения

ошибки *backpropa* и гибридный метод *hybrid*, объединяющий метод обратного распространения ошибки и метод наименьших квадратов.

Кроме того, в *Train FIS* можно изменять параметры обучения: задать требуемую точность обучения в поле *Error tolerance*, задать количество итераций обучения в поле *Epochs* и, наконец, запустить режим обучения кнопкой *Train Now*. Промежуточные результаты обучения выводятся в область визуализации, а также в рабочую область *Matlab*.

В *области тестирования Test FIS* расположены меню выбора выборки и кнопка *Test Now* запуска тестирования нечеткой системы с выводом результатов в область визуализации.

В *область вывода текущей информации* выводятся сообщения об окончании выполнении операций, отображаются значения ошибок обучения, тестирования и т.д.

3. Методика выполнения работы

Пример. Решается задачи аппроксимации функции:

$$f(x_1, x_2) = (1 - x_1^2) + 2(1 - x_2)^2,$$

в которой диапазон изменения аргумента функции ограничен интервалом [–1;1], с использованием ННС в редакторе *Anfis Editor* системы *Matlab*.

3.1. Подготовка обучающей выборки

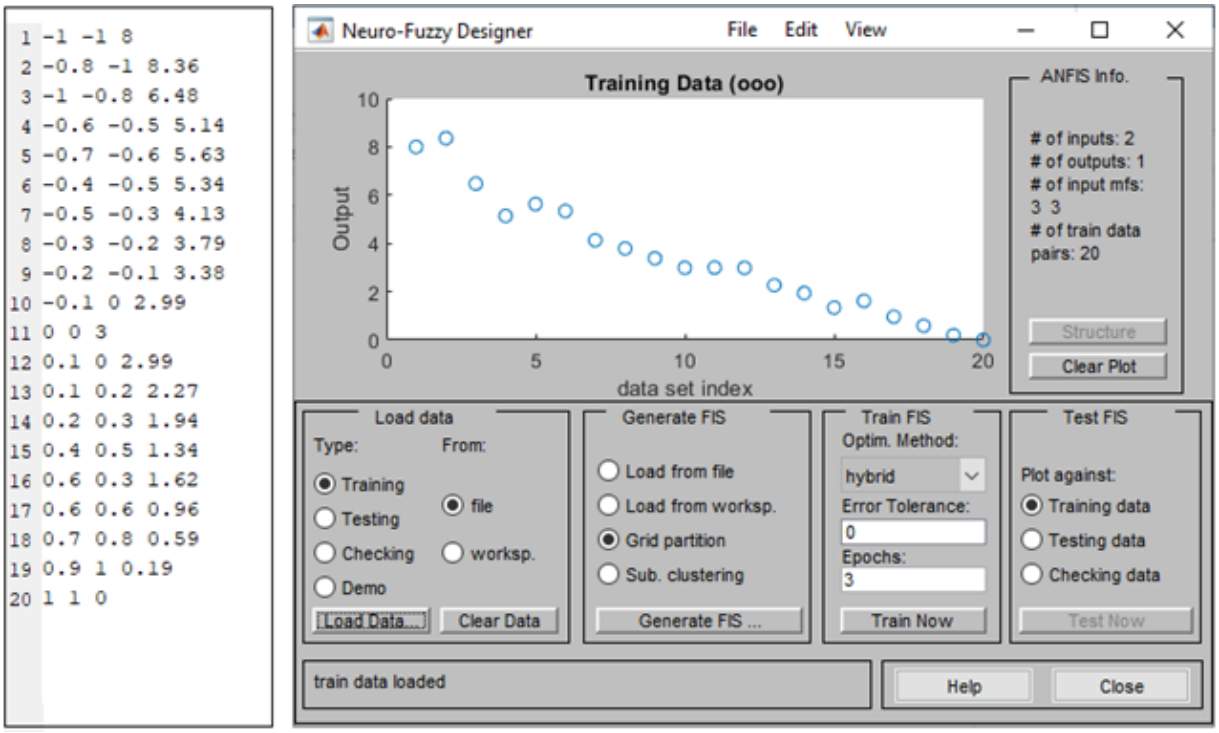
Для решения задачи аппроксимации необходимо подготовить обучающие данные в текстовом файле с расширением **.dat*. Каждая строка обучающего файла представляет собой числовые значения входов и выхода, разделенные пробелами.

Структура данных обучающего файла определяет структуру ННС: например, строка из двух чисел означает, что в ННС один вход и один выход; строка из трех чисел – два входа и один выход и т.д. Обучающая выборка для функции $f(x_1, x_2)$ показана на рис. 4.6, *а*).

3.2. Загрузка данных в редактор *Anfis Editor*

Для открытия редактора *Anfis Editor* нужно в командной строке системы *Matlab* ввести *anfisedit.*

В редактор *Anfis Editor* загружается файл с обучающими данными (тип данных *Trainig data*) – в меню *Load data,* либо с помощью менеджера файлов нужно указать имя файла с обучающими данными с расширением **.dat*. При успешной загрузке в рабочем окне редактора будет показан график, визуализирующий исходную функцию с загруженными данными (рис. 4.6, *б*).



а *б*

Рис. 4.6. Обучающие данные для функции $f(x_1, x_2)$ (*а*)
и редактор *ANFIS* после загрузки файла с обучающими данными (*б*)

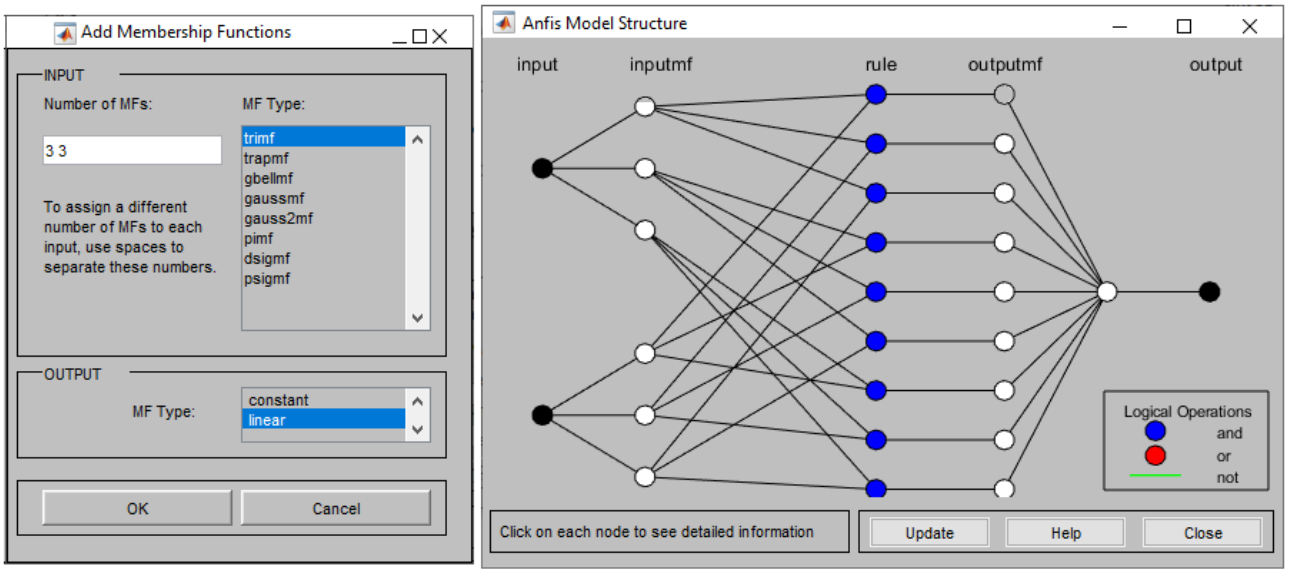
Сохранение разрабатываемой ННС в рабочее пространство (в переменную) проводится командой меню *File – Save to workspace as*.

3.3. Создание структуры ННС и формирование нечетких правил.

Для создания структуры ННС на панели редактора *Generate FIS* устанавливается форма разбиения входных переменных *по решетке* (*Grid partitions*), в которой носители нечетких термов определяются по обучающей выборке независимо для каждого входа; затем запускается команда *Generate FIS.*

В появившемся диалоговом окне указывается число и тип ФП для отдельных термов входных и выходной переменных (рис. 4.7, *а*): в поле ввода *Number of MFs* задается количество нечетких термов – например, по 3 нечетких терма для каждой входной переменной; в поле *MFType* указывается тип ФП входных переменных – например, треугольный, в поле *OUTPUT* – тип ФП выходной переменной – линейная ФП.

Просмотр структуры ННС осуществляется с помощью команды *Structure* редактора *Anfis Info* (рис. 4.7, *б*).



а *б*

Рис. 4.7. Окно выбора количества и типа ФП (*а*) и структура СНВ (*б*)

При создании структуры ННС в редакторе *Anfis Editor* системы *Matlab автоматически* формируются нечеткие правила, общее число которых соответствует количеству связей всех термов входных переменных друг с другом и рассчитывается как произведение числа термов для каждой входной переменной: так, для рассматриваемого примера с 3 нечеткими термами для обеих входных переменных формируется 9 нечетких правил.

Запуск команды меню *Edit – Rules* открывает редактор базы нечетких правил *Rule Editor* (рис. 4.8). При модификации системы нечеткого вывода может потребоваться дополнительное изменение существующих правил или изменение параметров ФП входных переменных.

3.4. Обучение ННС

Для обучения ННС на панели редактора *Train FIS* выбирается алгоритм обучения (*Hybrid* или *Backpropo*), устанавливаются целевое значение ошибки *Error Tolerance* и количество циклов обучения *Epochs*. Нажатие кнопки *Train Now* запускает процесс обучения.

Для рассматриваемого примера нужно оставить без изменения предложенные по умолчанию метод обучения (*Hybrid*) и уровень ошибки (0); количество циклов обучения *Epochs* установить, равным 30. После обучения ННС в рабочем окне редактора *ANFIS* будет показан график процесса обучения с динамическим изменением ошибки модели в ходе обучения (рис. 4.9).

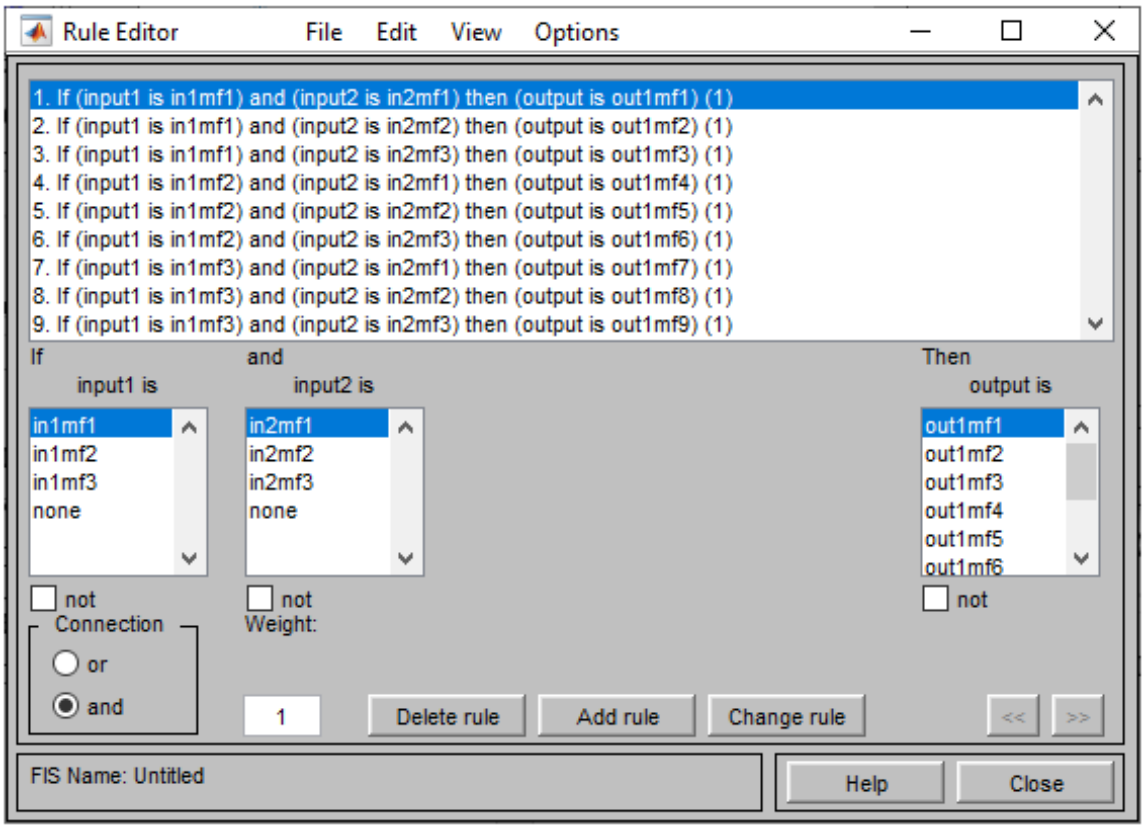


Рис. 4.8. Вид редактора правил нечеткого вывода

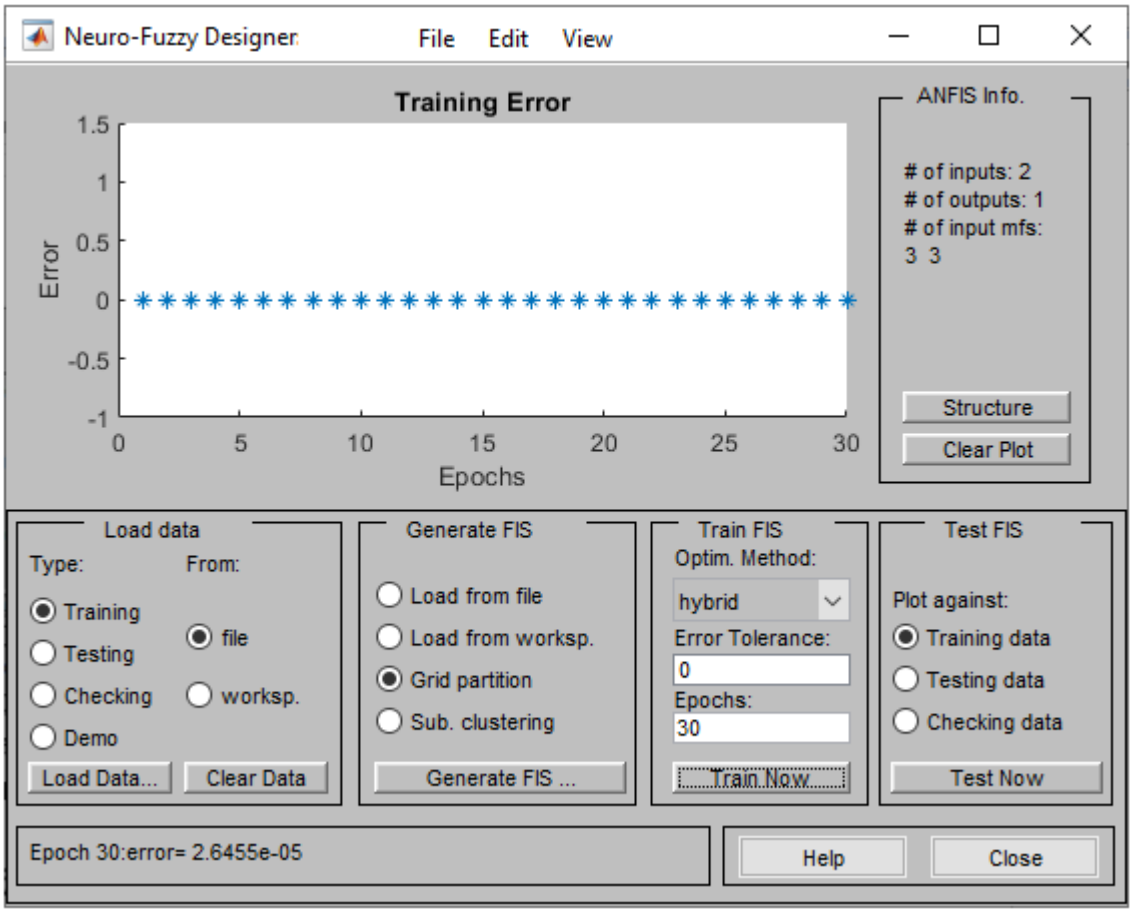


Рис. 4.9. График процесса обучения

Для проверки результатов обучения необходимо подготовить и загрузить дополнительный файл с проверочными (тестовыми) исходными данными, сформированными для пар значений рассматриваемой математической функции, в режиме *Testing data* в окне менеджера файлов, а затем запустить команду *Test Now;* результат аппроксимации тестовых данных с помощью ННС отображается в области визуализации.

Анализ точности построенной модели ННС выполняется визуально путем просмотра поверхности СНВ (команда *View/Surface*) (рис. 4.10). Сравнительный анализ изображенного графика с точным графиком функции $f(x_1, x_2)$ позволяет сделать вывод о достаточно высокой степени их совпадения, что может свидетельствовать о качестве построенной нейро-нечеткой модели.

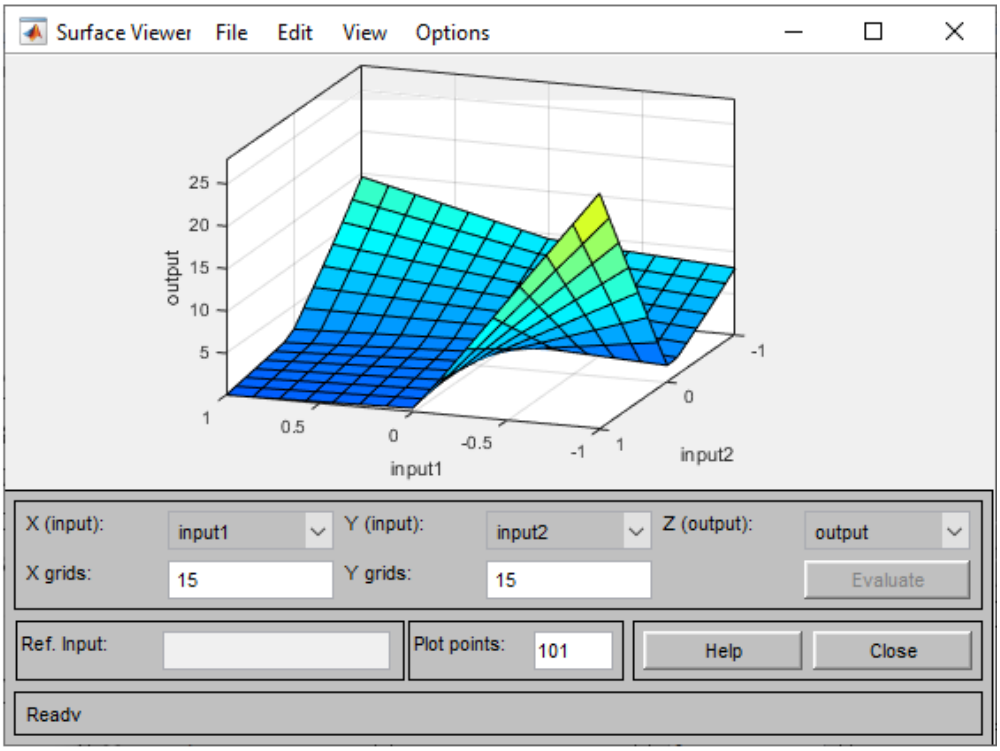


Рис. 4.10. Графический интерфейс просмотра поверхности СНВ

3.5. Анализ адекватности построенной модели можно выполнить с помощью редактора просмотра правил (команда *View/Rules*) построенной СНВ (рис. 4.11), позволяющую проводить оценку работы СНВ в режиме «*что-если*»: в поле *Input* или изменением положения красной линии-курсора устанавливаются значения входных переменных и на основе процедуры нечеткого вывода вычисляется значение результирующей переменной.

Например, если значение входной переменной устанавливается равным 1,1, то значение выхода модели становится 1,28 (рассчитанное точное значение равно 1,331). Если же входное значение равно 0,1, то выходом построенной модели является -0,09, хотя по расчетам должно равняться 0,001 – данный факт свидетельствует не в пользу адекватности построенной модели и требует ее дополнительной настройки.

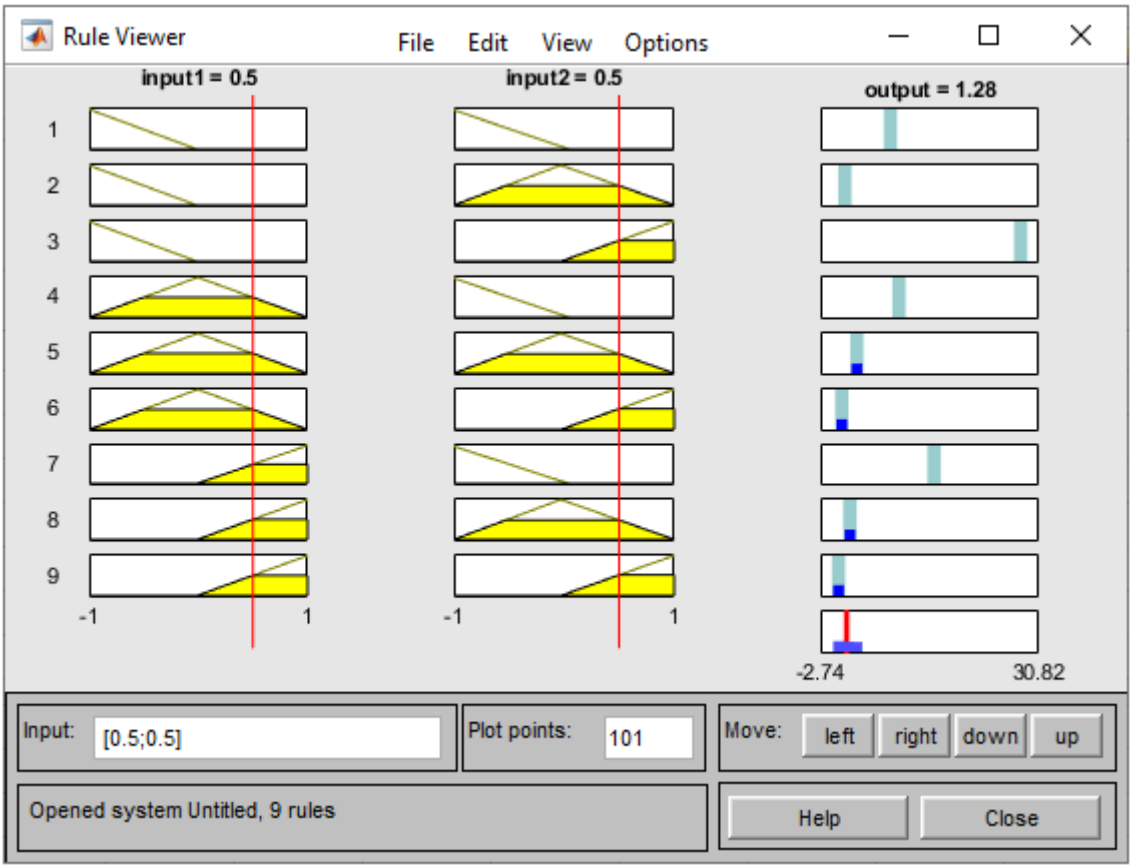


Рис. 4.11. Редактор просмотра правил СНВ

Для дополнительной настройки нейро-нечеткой модели применяются следующие способы:

1) увеличение объема выборки с обучающими данными;

2) изменение структуры ННС путем корректировки числа нечетких меток входных переменных;

3) изменение параметров обучения ННС.

Редактор *Anfis Matlab* допускает еще один способ дополнительной ручной настройки, который заключается в редактировании типов и значений параметров ФП термов входных переменных с помощью редактора ФП.

Выбор того или иного способа настройки нечетких моделей зависит не только от специфики решаемой задачи, но и от объема доступной выборки обучающих и проверочных данных.

Процесс разработки гибридных нейро-нечетких систем в интерактивном режиме является наиболее эффективным для сложных нейро-нечетких моделей с большим числом переменных и правил нечеткого вывода. Задание переменных и ФП в графическом режиме, а также визуализация правил позволяют существенно уменьшить

трудоемкость разработки нейро-нечеткой модели, а также снизить количество возможных ошибок и сократить общее время нечеткого моделирования.

Порядок выполнения работы

1. Изучить методику разработки гибридных систем с использованием среды визуального моделирования *Anfis Matlab*.

2. Исследовать процесс разработки адаптивной системы нейро-нечеткого вывода для аппроксимации зависимости, описываемой некоторой математической функцией, согласно варианту задания

2.1. Подготовить обучающие данные, сохранить их в виде **.dat* –файла.

2.2. Сгенерировать структуру ННС; обучить ННС с использованием гибридного метода.

2.3. Выполнить оценку адекватности построенной нечеткой модели гибридной сети.

Провести дополнительную настройку модели путем увеличения объема обучающей выборки; изменения структуры ННС, параметров обучения ННС; путем редактирования типов и значений параметров ФП термов входных переменных.

Заполнить таб. 4.1 результатами экспериментальных исследований над моделью: для разных наборов исходных данных x_1 и x_2 рассчитать точные значения $d = f(x_1, x_2)$; затем для разных типов ФП входных переменных определить выходные значения y и ошибки $\varepsilon = (y - d)^2$ модели.

Таблица 4.1

Результаты экспериментальных исследований

№	*Входные значения функции*		*Точное значение*	*gauss2mf*		*trimf*		*gaussmf*	
	x_1	x_2	d	y	ε	y	ε	y	ε
1									
2									
3									
4									
5									

2.4. В случае необходимости внести изменения в структуру гибридной сети, в правила нечеткого вывода; а затем провести дополнительную проверку точности построенной нечеткой модели гибридной сети.

3. Сделать выводы о качестве построенной нейро-нечеткой модели.

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

1. Название и цель работы.
2. Исходные данные варианта задания.
3. Этапы разработки адаптивной системы нейро-нечеткого вывода для аппроксимации зависимости, описываемой некоторой математической функцией, согласно варианту задания.
4. Результаты экспериментальных исследований над нейро-нечеткой моделью.
5. Выводы о качестве построенной нейро-нечеткой модели.

Варианты заданий

№	Математическая функция	Диапазон изменения x_1 и x_2
1	*2*	*3*
1	$f(x_1, x_2) = (1 - {x_1}^2) + 2(1 - x_2)^2$	$[-1; 1]$
2	$f(x_1, x_2) = 2(1 + {x_1}^2) + 3(x_2 - 2)^2$	$[-1; 1]$
3	$f(x_1, x_2) = (x_1 - 1)^2 + (2 - {x_2}^2)$	$[-1; 1]$
4	$f(x_1, x_2) = 2(x_1 - 1)^2 + (1 + {x_2}^2)$	$[-1; 1]$
5	$f(x_1, x_2) = (3 - {x_1}^2) + 2(1 + x_2)^2$	$[-1; 1]$
6	$f(x_1, x_2) = (2 + {x_1}^2) + (1 - 2x_2)^2$	$[-1; 1]$
7	$f(x_1, x_2) = (1 + {x_1}^2) + (1 - x_2)^2$	$[-1; 1]$
8	$f(x_1, x_2) = (3 - 2{x_1}^2) + (1 - 2x_2)^2$	$[-1; 1]$
9	$f(x_1, x_2) = (2 - {x_1}^2) + (1 + x_2)^2$	$[-1; 1]$
10	$f(x_1, x_2) = (3 - 2{x_1}^2) + (1 - x_2)^2$	$[-1; 1]$
11	$f(x_1, x_2) = (1 - 2{x_1}^2) + (3 - x_2)^2$	$[-1; 1]$
12	$f(x_1, x_2) = 3(1 + {x_1}^2) + 2(x_2 - 2)^2$	$[-1; 1]$
13	$f(x_1, x_2) = (x_1 - 2)^2 + (1 - {x_2}^2)$	$[-1; 1]$
14	$f(x_1, x_2) = (x_1 - 1)^2 + 2(1 + {x_2}^2)$	$[-1; 1]$
15	$f(x_1, x_2) = (1 - {x_1}^2) + 3(1 + x_2)^2$	$[-1; 1]$

Окончание табл.

1	*2*	*3*
16	$f(x_1, x_2) = (1 + {x_1}^2) + (2 - 3x_2)^2$	$[-1; 1]$
17	$f(x_1, x_2) = (1 + {x_1}^2) + (2 - x_2)^2$	$[-1; 1]$
18	$f(x_1, x_2) = (1 - {x_1}^2) + 2(3 - 2x_2)^2$	$[-1; 1]$
19	$f(x_1, x_2) = (1 - {x_1}^2) + (2 + x_2)^2$	$[-1; 1]$
20	$f(x_1, x_2) = (1 - 3{x_1}^2) + (2 - x_2)^2$	$[-1; 1]$

Контрольные вопросы

1. Что такое нейронные нечеткие системы? Охарактеризуйте структуру нейро-нечеткой сети.
2. Опишите процесс разработки ННС в системе *Anfis Matlab*.
3. Как проверить адекватность построенной ННС?
4. Какие варьируемые параметры нейро-нечеткой системы влияют на ошибку аппроксимации?
5. В чем преимущества использования нейро-нечетких сетей?

Список литературы

1. Модели и методы искусственного интеллекта. Применение в экономике: / М. Г. Матвеев, А. С. Свиридов, Н. А. Алейникова. М.: Финансы и статистика: ИНФРА–М., 2008. 446 с.
2. Леоненков А.В. Нечеткое моделирование в среде MATLAB и FuzzyTECH. СПб.: БХВ-Петербург, 2003. 736 с.
3. Бычков, Ю.А. Непрерывные и дискретные нелинейные модели динамических систем: монография / Ю. А. Бычков, Е. Б. Соловьева, С. В. Щербаков. СПб.: Лань, 2018. 420 с.
4. Ростовцев В. С. Искусственные нейронные сети: учебник / В.С. Ростовцев. СПб.: Лань, 2019. 216 с.
5. Ярушкина Н. Г. Основы теории нечетких и гибридных систем. М.: Финансы и статистика, 2004. 320 с.
6. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. М.: Радио и связь, 2000. 382 с.
7. Борисов В.В., Круглов В.В., Федулов А.С. Нечеткие модели и сети. М.: Горячая линия-Телеком, 2007. 284 с.
8. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. М.: Горячая линия-Телеком, 2013. 384 с.

9. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику. [Электронный ресурс] // http://www.Matlab.ru/fuzzylogic/book1/index.asp/.

10. Рогозин О. В. Нейро-нечеткая система поддержки принятия решений в слабо структурированных задачах // Новые информационные технологии в автоматизированных системах. 2012. №15. С. 68–77.

Лабораторная работа № 5

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ В СИСТЕМЕ *MATLAB*

1. Цель и задачи работы

Целью работы является изучение методики решения задач оптимизации с использованием генетических алгоритмов в системе *Matlab*.

Задачами работы являются: настройка параметров генетических алгоритмов; выбор исходной популяции хромосом, оценка функции приспособленности, селекция хромосом, применение генетических операторов (скрещивания и мутации), определение наилучшей хромосомы в редакторе *Genetic Algorithm* утилиты *Optimization Toolbox* системы *Matlab*.

2. Теоретические сведения

2.1. Решение задач оптимизации с помощью генетических алгоритмов

Многие задачи, возникающие в фундаментальных и прикладных науках, сводятся к задаче *оптимизации*. В оптимизационных задачах требуется найти значения входных параметров, при которых целевая функция достигает минимального (максимального) значения.

Сложность поиска оптимального решения зависит от вида целевой функции. Если целевая функция $F(X)$ *унимодальная*, т.е. имеет один экстремум (рис. 5.1, *а*), то задача оптимизации относительно проста – для решения *локальной оптимизации* предложено большое количество методов, в частности, методы спуска по градиенту целевой функции. Традиционные (градиентные) методы оптимизации дают возможность находить только локальные экстремумы.

В случае если целевая функция $F(X)$ *мультимодальная*, т.е. имеет много экстремумов (рис. 5.1, *б*), то задача оптимизации значительно усложняется, т.к. при решении задачи *глобальной оптимизации* можно столкнуться либо с проблемой

преждевременной сходимости (установление локального экстремума вместо глобального), либо с проблемой длительных вычислений.

Особенностями задач непрерывной глобальной оптимизации являются: нелинейность, недифференцируемость, овражность, многоэкстремальность (мультимодальность), отсутствие аналитического выражения (плохая формализованность) и высокая вычислительная сложность оптимизируемых функций, высокая размерность пространства поиска, сложная топология области допустимых значений и т. д.

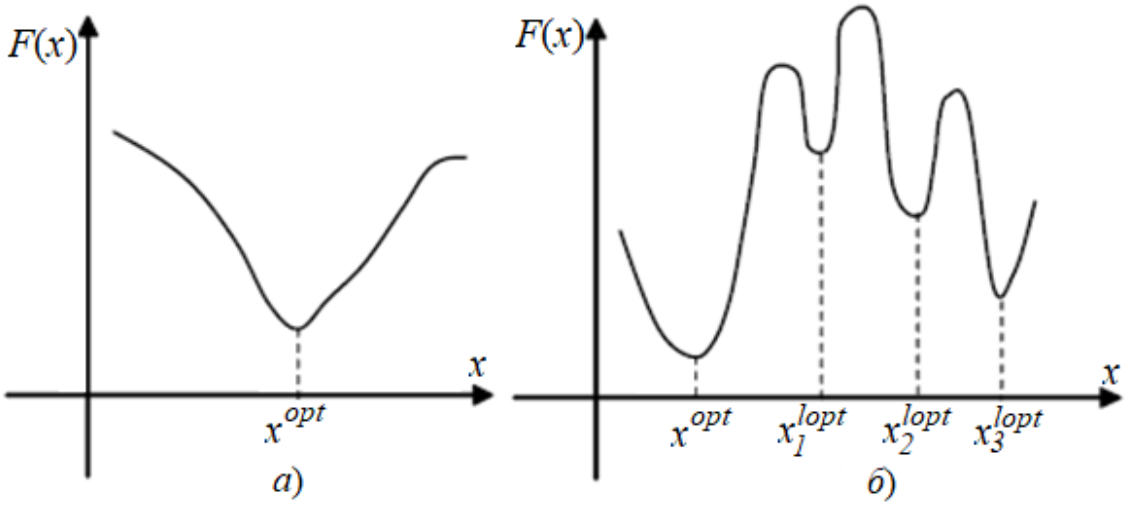


Рис. 5.1. Унимодальная (*а*) и мультимодальная (*б*) целевая функции

Для мультимодальных целевых функций, имеющих множество локальных экстремумов (рис. 5.2), таких, например, как функция Растригина:

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10 \cdot (cos2\pi x_1 + cos2\pi x_2),$$

найти глобальный экстремум традиционными методами не представляется возможным.

Для решения этих проблем в 1975 г. Дж.Холландом были предложены *генетические алгоритмы* (ГА), которые относятся к стохастическим эвристическим методам. ГА – это адаптивные методы поиска, которые в настоящее время активно используются для решения задач *оптимизации*. Они строятся на принципах *эволюции* биологических организмов: биологические популяции развиваются в течение нескольких поколений, подчиняясь законам естественного отбора по принципу «выживает наиболее приспособленный», открытому Чарльзом Дарвином.

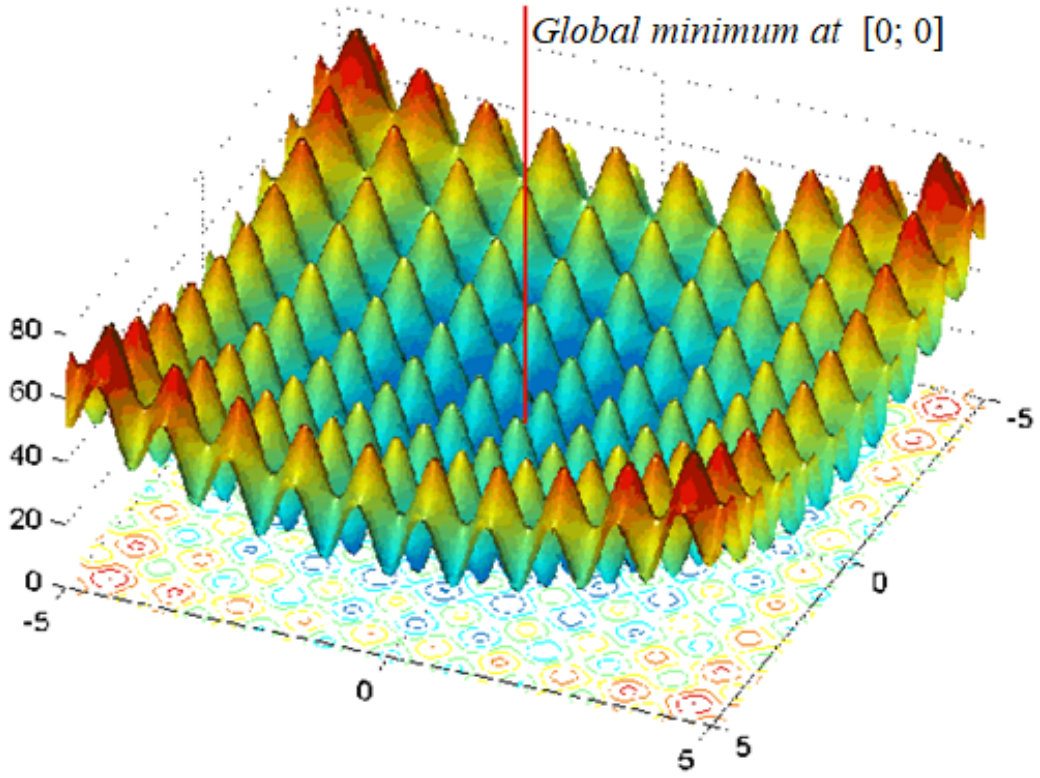


Рис. 5.2. График функции Растригина

Отбор в ГА тесно связан с принципами естественного отбора:

– приспособленность особи соответствует значению целевой функции на заданном варианте;

– выживание наиболее приспособленных особей происходит благодаря тому, что популяция каждого следующего поколения формируется на основе вычисления значения целевой функции с помощью модели отбора – чем приспособленнее особь, тем больше вероятность ее участия в кроссовере, т.е. в размножении.

Модель отбора определяет то, как строится популяция следующего поколения. Как правило, вероятность участия особи в скрещивании берется пропорционально ее приспособленности. Часто используется так называемая *стратегия элитизма*, при которой несколько лучших особей переходят в следующее поколение без изменений, не участвуя в кроссовере и отборе. В любом случае, каждое следующее поколение будет в среднем лучше предыдущего.

Когда приспособленность особей перестает заметно увеличиваться, процесс отбора останавливают, и в качестве решения задачи оптимизации выбирается наилучший из найденных вариантов.

По скорости нахождения оптимума целевой функции ГА на несколько порядков превосходят случайный поиск.

Отметим, что оптимизационная задача может быть записана как минимизационная или как максимизационная. Эти две задачи легко преобразуются в другую форму: $min\, f(x) \Leftrightarrow max\, [-f(x)]$, $max\, f(x) \Leftrightarrow min\, [-f(x)]$, где $f(x)$ – целевая функция, а вектор x – независимая переменная, или переменная решения. Число элементов в x называется размерностью задачи. При минимизации функции значение целевой функции называется *функцией стоимости*, а при максимизации – *функцией приспособленности*.

2.2. Основные понятия генетических алгоритмов

ГА используют словарь, заимствованный из естественной генетики:

– *хромосомы (цепочки, или кодовые последовательности)* – это упорядоченные последовательности *генов*. Хромосома – битовая строка 0101…101, или вектор из нулей и единиц, которая определяет точку пространства поиска и представляет потенциальное решение задачи;

– *гены* – элементы, из которых состоит хромосома;

– *популяция* – это конечное множество особей;

– *особи,* входящие в популяцию, – хромосомы с закодированным в них множествами параметров задачи, т.е. решений, которые иначе называются точками в пространстве поиска;

– *генотип,* или структура, – это набор хромосом данной особи. Следовательно, особями популяции могут быть генотипы либо единичные хромосомы (распространённый случай – когда генотип состоит из одной хромосомы);

– *фенотип* – это набор значений, соответствующих данному генотипу, т.е. декодированная структура или множество параметров задачи (решение, точка пространства поиска);

– *аллель* – значение конкретного гена, также определяемое как значение *свойства* или вариант свойства;

– *локус,* или позиция, – место размещения данного гена в хромосоме (цепочке). Множество позиций генов – это локи;

– *кроссовер* – операция скрещивания хромосом, при котором хромосомы обмениваются своими частями;

– *мутация* – случайное изменение одной или нескольких позиций в хромосоме;

– *функция приспособленности*, или функция пригодности (*fitness function*), – функция оценки, мера приспособленности особи в популяции. Именно она позволяет выбирать наиболее приспособленные особи. На каждой итерации ГА приспособленность каждой особи популяции оценивается при помощи функции приспособленности, и на этой основе создается следующая популяция особей, составляющих множество потенциальных решений, например, задачи оптимизации;

– механизм *селекции* заключается в выборе особей с наивысшей оценкой (т.е. наиболее приспособленных), которые репродуцируют чаще, чем особи с более низкой оценкой (хуже приспособленные);

– *репродукция* – создание новых хромосом в результате рекомбинации генов родительских хромосом. *Рекомбинация* – это процесс, в результате которого возникают новые комбинации генов. Для этого используется две операции: *скрещивание,* позволяющее создать две совершенно новые хромосомы потомков путем комбинирования генетического материала пары родителей, а также *мутация,* которая может вызвать изменения в отдельных хромосомах.

2.3. Этапы нахождения оптимального решения задачи с помощью генетического алгоритма

Стандартный ГА состоит из следующей последовательности шагов (рис. 5.3):

Шаг 1. *Инициализация* – формирование исходной популяции хромосом на основе случайной генерации заданного количества хромосом (особей), представляемых двоичными последовательностями фиксированной длины.

Шаг 2. *Оценивание приспособленности* хромосом – расчет *функции приспособленности* для каждой хромосомы этой популяции. Чем больше значение этой функции, тем выше «качество» хромосомы.

Шаг 3. *Проверка условия остановки генетического алгоритма*. Существует несколько условий остановки ГА:

1) в оптимизационных задачах, если известно максимальное (или минимальное) значение функции приспособленности, то остановка алгоритма осуществляется после достижения ожидаемого оптимального значения;

2) когда выполнение ГА не приводит к улучшению уже достигнутого значения;

3) алгоритм может быть остановлен либо по истечении определенного времени выполнения, либо после выполнения заданного количества итераций.

Если условие остановки выполнено, то алгоритм переходит к завершающему шагу 7, в котором определяется *«наилучшая»* хромосома. Если же условия остановки не соблюдаются, то осуществляется переход к следующему шагу и выполняется *селекция*.

Рис. 5.3. Блок-схема генетического алгоритма

Шаг 4. *Селекция хромосом* – отбор хромосом, которые будут участвовать в создании потомков для следующей популяции. В результате процесса селекции создается *родительская популяция*, (родительский пул) с численностью N, равной численности текущей популяции. Существует несколько видов селекции, наиболее популярным из которых считается *метод рулетки.*

Шаг 5. *Применение генетических операторов* к хромосомам родительской популяции. Применяются два основных генетических оператора: оператор скрещивания (*crossover*) и оператор мутации (*mutation*). В результате формируется новая популяция потомков: при скрещивании пары родительских хромосом создается пара потомков. *Оператор мутации* с вероятностью p_m, т.е. случайным образом, изменяет значение некоторого гена в некоторой хромосоме на противоположное и формирует новую, ранее не существующую хромосому в популяции. Таким образом, с одной стороны, мутации вводят в популяцию некоторое разнообразие и расширяют область поиска, а с другой стороны, они предупреждают потери, которые могли бы произойти вследствие исключения какого-нибудь значимого гена в результате скрещивания.

Шаг 6. *Формирование новой популяции*. Хромосомы, образованные в результате применения генетических операторов к хромосомам родительской популяции, формируют новую популяцию. Далее алгоритм возвращается к шагу 2.

Шаг 7. *Выбор «наилучшей» хромосомы*. Если условие остановки алгоритма выполнено, то следует вывести результат работы, т.е. представить искомое решение задачи. Лучшим решением считается хромосома с наибольшим значением функции приспособленности.

2.4. Генетические алгоритмы в редакторе *Genetic Algorithm* утилиты *Optimization Toolbox* системы *Matlab*

Для решения оптимизационных задач в пакете *Matlab* имеется *GUI*-утилита *Optimization Toolbox.* Одним из методов нахождения точек экстремума скалярных функций многих переменных, использующихся в этой утилите, является метод ГА.

Для загрузки редактора *Genetic Algorithm* (*GA*-редактора) необходимо в разделе *APPS* главного меню системы *Matlab* загрузить утилиту *Optimization Toolbox*; а затем в качестве решателя указать

*Solver – ga – Genetic Algorithm***.** В результате выполнения этой команды появится окно графического интерфейса *GA*-редактора (рис. 5.4).

Левая часть окна *GA*-редактора содержит поля для внесения основной информации об оптимизируемой функции нескольких переменных (табл. 5.1).

Таблица 5.1

Внесение информации об оптимизируемой функции в *GA*-редакторе

Поле	Назначение
Панель *Problem*	
Fitness function	в поле указывается оптимизируемая функция в виде *@fitnessfun*, где *fitnessfun.m* – имя заранее подготовленного *M*-файла, в котором описывается оптимизируемая функция
Number of variables	размер входного вектора для функции приспособленности (количество входных переменных)
Панель *Constraints* Задаются ограничения задачи и/или ограничивающая нелинейная функция. Если не требуется введение ограничений, то поля не заполняются	
Linear inequalities	Линейные ограничения модели записываются в виде неравенств вида: $A*x \le b$, а затем в поле A указываются значения коэффициентов $[A_1, \dots, A_n]$ и в поле b – $[b_1, \dots, b_n]$ – для n неравенств
Linear equalities	Линейные ограничения модели записываются в виде *равенств* вида: $A*x = b$, а затем в поле *Aeq* указываются значения коэффициентов уравнений $[Aeq_1, \dots, Aeq_n]$ и в поле beq – $[beq_1, \dots, beq_n]$ – для n равенств
Bounds	Задаются нижние (*Lower – lb*) и верхние (*Upper – ub*) границы n переменных в формате $[lb(x_1), \dots, lb(x_n)]$ и $[ub(x_1), \dots, ub(x_n)]$
Nonlinear constraint function	Задается произвольная нелинейная функция ограничений
Панель *Run Solver*	
Start	запуск алгоритма
Current iteration	при выполнении отображается текущая популяция
Status and Results	выводятся результаты: – сообщение "*Optimization terminated*", – значение функции приспособленности в оптимальной точке для финальной популяции; – причина остановки работы алгоритма
Final point	выводятся координаты конечной точки, при которой функция приспособленности достигает минимума

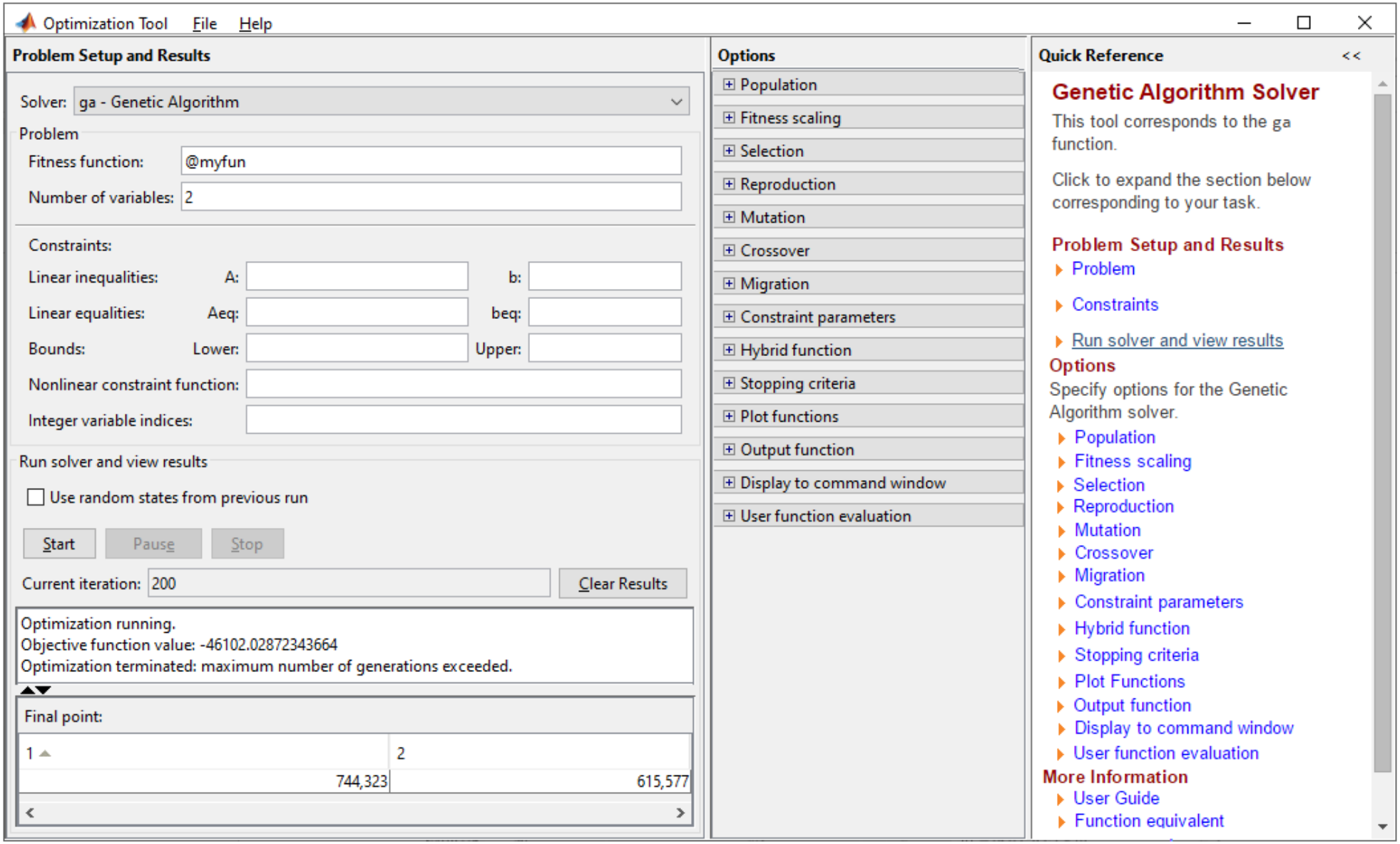


Рис. 5.4. Графический интерфейс *GA*-редактора

В правой части окна *GA*-редактора в разделе *Options* отображаются дополнительные параметры (опции), предназначенные для более точной настройки ГА при решении конкретной задачи; если не изменять параметры в этом разделе, то в работе ГА устанавливаются значения параметров по умолчанию.

2.4.1. Настройка параметров *Population* (табл. 5.2) в разделе *Options* позволяет задать тип всех объектов популяции *Population type* (двойной вектор, битовая строка или пользовательский тип); настроить размер популяции *Population size*; задать начальную популяцию *Initial population*; определить начальный рейтинг особей *Initial ranges*, ввести числовой диапазон особей начальной популяции *Initial ranges*.

Таблица 5.2

Настройка параметров опции *Population* в *GA*-редакторе

Характеристика популяций ГА *Population*	
Поле	Назначение
Population type	Задаются типы популяции: – *Double vector* – вектор, составленный из действительных чисел формата *double* (64 бита) (по умолчанию); – *Bit string* – строка двоичных символов; – *Custom* – тип данных, определяемый пользователем
Population size	Задается число особей популяции в каждом поколении. По умолчанию число особей равно 50 при числе входных переменных меньше 5, а в остальных случаях – 200
Creation function	Устанавливаются функции инициализации популяции: *Constraint depend* (по умолчанию); функция *Uniform*, создающая начальную популяцию в виде точек, распределенных случайно и равномерно; пользовательская функция *Custom*, которая будет генерировать особи в соответствии с типом *Population type*
Initial population	Установление точек начальной популяции *вручную* в формате: $[x_{11} \dots x_{1n}; \dots; x_{m1} \dots x_{mn}]$, где n – количество переменных (*Number of variables*), а m – размер популяции (*Population size*). Если данный параметр не определен, начальная популяция формируется в соответствии с опцией *Creation function*
Initial scores	Начальное распределение функции приспособленности (начальный рейтинг) особей
Initial ranges	Задание начального диапазона изменений координат точек начальной популяции – по умолчанию это интервал [-10; 10] для всех особей. Диапазон имеет вид матрицы, в которой число строк равно числу особей популяции (*Initial length*) и *Number of variables* столбцов. Каждый столбец имеет вид [*X*min*; X*max]

2.4.2. Настройка опции *Fitness scaling* – это настройка масштабирования целевой функции приспособленности, которая значительно влияет на производительность генетического алгоритма.

Если диапазон масштабируемых значений слишком широк, то особи с высокими значениями масштабируемой целевой функции воспроизводятся слишком быстро, захватывая генофонд популяции и не позволяя ГА искать другие области пространства решений. С другой стороны, при малом диапазоне все особи имеют примерно одинаковые шансы на размножение, и поиск решения будет осуществляться очень медленно.

Rank scales (по умолчанию) – масштабирует баллы на основе ранга каждой отдельной особи, а не ее оценки приспособленности. Ранг особи – это ее позиция в баллах: ранг самой лучшей особи равен 1, ранг следующей – 2 и т.д.

Top scales – наибольшее значение рейтинга присваивается лишь части особей текущей популяции с наилучшими оценками; в поле *Quantity* задается процент таких выдающихся особей (либо от 1 до значения *Population Size,* либо как доля от общего числа особей популяции в интервале от 0 до 1).

Shift linear – смещенное линейное масштибирование функции, которое достигается путем умножения на константу *Maximum survival rate* (по умолчанию равную 2).

Custom – пользовательский вид преобразования функции.

2.4.3 *Selection* – настройка способа *отбора* (селекции) родительских особей для генерации потомков. В *GA*-редакторе используется несколько методов селекции (*Selection Function*):

Roulette – метод рулетки, в котором родительские особи выбираются пропорционально значениям их функций приспособленности;

Tournament – турнирная селекция, при которой из случайно выбранного числа особей на конкурсной основе отбираются лучшие;

Uniform – метод отбора, когда родительские хромосомы отбираются в соответствии со значением приспособленности и требуемым числом потомков. Этот метод полезен при отладке и тестировании алгоритмов, но малоэффективен в реальных задачах;

Stochastic uniform (по умолчанию) – стохастическая равномерная селекция. Хромосомы отображаются в отрезки прямой, длина

которых пропорциональна функции приспособленности. В процессе селекции происходит движение по этой прямой шагами одинаковой длины. Точка, из которой делается первый шаг, выбирается случайно (в пределах длины шага). После каждого шага отбирается та хромосома, в пределы которой попала точка.

2.4.4. *Reproduction* – настройка способа формирования потомков для последующих поколений.

Elite count – положительное целое число, определяющее количество элитных особей, т.е. особей текущей популяции, которые без изменений будут скопированы в последующем поколении; по умолчанию *Elite count* равно 0,05**Population Size*.

Crossover fraction – определяет часть хромосом, которые попадут в будущую популяцию из родительского пула в результате операции скрещивания при исключении элитных особей; это число изменяется от 0 до 1 (по умолчанию равно 0,8).

2.4.5. *Mutation* – настройка параметров, задающих характер случайных мутаций в родительских особях в каждом поколении. Операция мутации препятствует преждевременной сходимости путем «выбивания» популяции из локального экстремума за счет случайного изменения гена в хромосоме Возможны следующие типы параметра мутации:

Gaussian – мутация, которая определяет случайные значения от 0 до размера популяции *Population Size*, распределенные по *Гауссовскому* закону. Виды этого распределения определяются с помощью параметров *Scale* и *Shrink*;

Uniform – стандартная мутация, которая определяет равномерно распределенные случайные значения; параметр *Rate* – вероятность мутации;

Adaptive feasible – адаптивная функция мутации, которая основывается на результатах предыдущего поколения;

Custom – пользовательская функция;

Constraint depend – по умолчанию.

2.4.6. *Crossover* – настройка параметра скрещивания, при котором генерируется случайный двоичный вектор соответствия родителей:

Scattered – распределенный кроссовер (по умолчанию);
Single point – одноточечный кроссовер;
Two point – двухточечный кроссовер;
Intermediate – промежуточный кроссовер;
Heuristic – эвристический кроссовер;
Custom – пользовательский кроссовер;
Constraint depend – по умолчанию.

2.4.7. *Migration* – настройка параметра, который используется, когда начальная популяция задана в виде нескольких субпопуляций, и необходимо определить направление движения особей между субпопуляциями и долю таких особей:

Direction – это направление движения особей вперед *Forward* (по умолчанию) и в обоих направлениях *Both*.

Fraction – параметр, который определяет долю особей, перемещающихся между субпопуляциями (по умолчанию равно 0,2);

Interval – данный параметр определяет частоту миграции: значение по умолчанию равно 20, что означает миграции особей после каждого 20-го поколения.

2.4.8. *Constraints parameters* – настройка параметров ГА для решения задач нелинейной оптимизации с линейными и нелинейными ограничениями и границами.

По умолчанию генетический алгоритм использует алгоритм *ALGA* (*Augmented Lagrangian Genetic Algorithm*), в котором определятся *Initial Penalty* – начальное значение штрафа, равное по умолчанию 10, *PenaltyFactor* – значение штрафа, равное 100.

Другой алгоритм – *Penalty Algorithm* – использует метод штрафов.

2.4.9. *Hybrid function* – настройка параметров гибридного или комбинированного поиска оптимума (по умолчанию – минимума), который запускает дополнительную функцию поиска после остановки работы ГА для уточнения положения точки минимума. Возможный выбор: *none* (по умолчанию); *fminsearch* – *Matlab*-функция поиска минимума без ограничений, реализующая метод Нелдера-Мида; *patternsearch* – функция поиска по образцу; *fminunc* – функция минимизации в утилите *Optimization Toolbox*.

2.4.10. *Stopping Criteria Options* – настройка условий остановки генетического алгоритма (табл. 5.3).

2.4.11. *Plots function* – настройка графиков, визуализирующих различные аспекты ГА по мере его выполнения (табл. 5.4).

2.4.12. *Output Function* – настройка параметров вывода результатов.

History to new window – открытие дополнительного окна, в котором графики отображаются через каждые *Interval* итераций.

Custom – задается пользовательская функция вывода выходной информации.

Таблица 5.3

Условия остановки алгоритма *Stopping Criteria*

Поле	Алгоритм останавливается ……
Generations	когда число поколений достигает заданного значения *Generations* (по умолчанию равно 100**numberOfVariables*)
Time limit	по истечении заданного времени в секундах *Time limit*
Fitness limit	когда значение функции приспособленности для наилучшей точки текущего семейства будет меньше или равно *Fitness limit*
Stall generations	если нет улучшения для целевой функции в последовательности следующих друг за другом поколений длиной *Stall generations*
Stall time limit	если нет улучшения для целевой функции в течение интервала времени (сек.), равного *Stall time limit*

Таблица 5.4

Настройка отображения результатов работы ГА

Поле	Визуализация ….
1	*2*
Best fitness	лучшей и средней функции приспособленности для каждого поколения
Expectation	ожидаемого количества потомков по сравнению с исходными баллами для каждого поколения.
Score diversity	оценки разнообразия итоговой популяции в виде гистограммы расстояний между особями
Stopping	критериев остановки
Best individual	значения хромосом особи (значения входных переменных) с лучшей функцией приспособленности для текущего поколения
Genealogy	графика, иллюстрирующего процесс образования потомков. Линии перехода от одного поколения к следующему раскрашены следующим образом: – красные линии – потомки, образованные в результате мутации; – голубые линии – потомка, образованные в результате кроссовера; – черные линии – элитные особи

Окончание табл. 5.4

1	*2*
Scores	значений целевой переменной для особей текущего поколения в виде гистограммы
Max constraint	максимального нарушения нелинейного ограничения в каждом поколении (для задач с ограничениями)
Distance	среднего расстояния между особями в каждом поколении (разница значений функции приспособленности)
Range	лучшего, худшего и среднего значения функции приспособленности в каждом поколении
Selection	количества потомков для каждой родительской особи начальной популяции

2.4.13. *Display to command window* – настройка параметров отображения работы генетического алгоритма в окне *Matlab*.

Уровень отображения (*Level of display*) принимает значения:

Off – выводится только конечный результат (по умолчанию);

Iterative – информация о поиске минимума обновляется на каждой итерации алгоритма;

Diagnose – информация о поиске минимума обновляется на каждой итерации алгоритма и выводится дополнительная информация с случае отсутствия сходимости алгоритма;

Final – информация о завершении поиска и причинах останова.

2.4.14. *User function evaluation* – настройка параметров оценки пользовательской функции (*in serial* – выключена, по умолчанию).

Задание параметра *vectorized* приводит к векторизации целевой функции, при которой при однократном обращении к ее *m*-файлу одновременно находится несколько значений функции при задании нескольких (в форме массива) значений аргументов.

3. Методика выполнения работы

Пример. Рассматривается процесс нахождения *минимума* и *максимума* функции двух переменных:

$$f(x,y) = 4sin(2\pi x) * cos(1.5\pi y)(1 - x^2) * y * (1 - y)$$

в диапазоне $-1 \le x \le 1,\ 0 \le y \le 1$ с помощью метода ГА в редакторе *Genetic Algorithm* утилиты *Optimization Toolbox* системы *Matlab.* Для этого необходимы следующие действия.

3.1. Предварительный просмотр поверхности функции.

Для того, чтобы проанализировать форму и график функции и предварительно определить точку минимума и максимума, необходимо построить трехмерный график поверхности функции двух переменных в *Matlab.*

Осуществляется разметка сетки будущей поверхности с интервалом изменения x и y от –5 до 5 с шагом 0,1:

```
[x,y]=meshgrid(-1:0.05:1,0:0.05:1)
```

Задается выражение для вычисления значений z в узлах сетки:

```
z=4*sin(2*pi*x).*cos(1.5*pi*y).*(1-x.^2).*y.*(1-y)
```

где «.*» и «.^» – операции поэлементного умножения и возведения в степень;

Строится трехмерный график поверхности функции:

```
mesh(x,y,z).
```

На построенном графике поверхности сложной функции обнаруживается несколько точек локального минимума и максимума (рис. 5.5). В точке A (-0,25; 0,65) достигается максимальное значение функции, равное $f(x_1,x_2)$=0,856, а в точке B(0,25; 0,6) достигается минимальное значение z =–0,856.

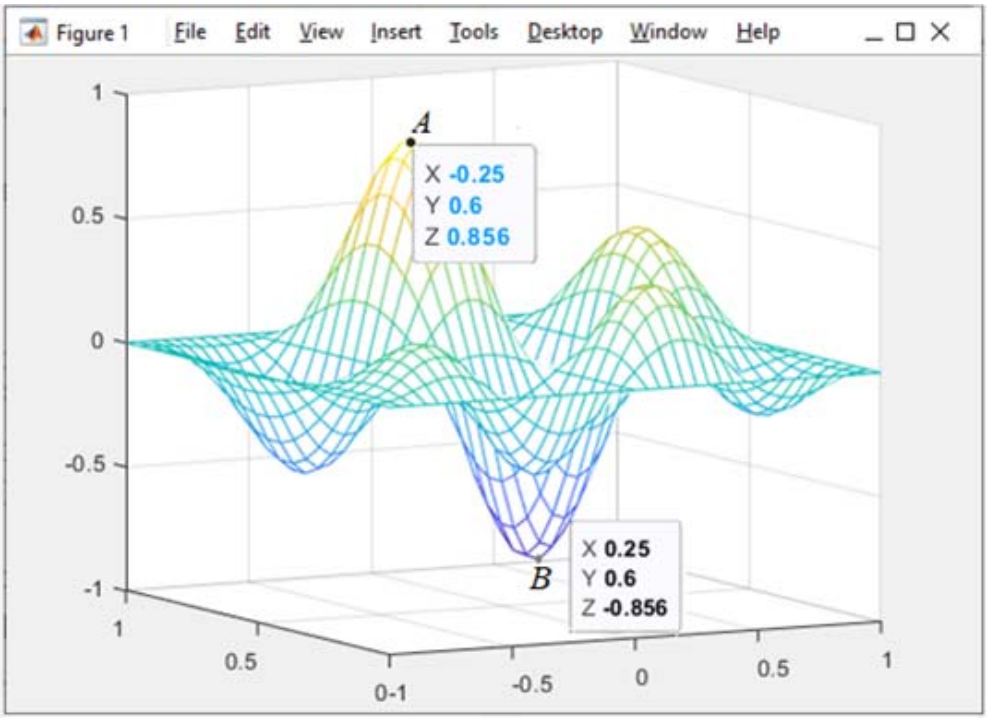


Рис. 5.5. Трехмерный график поверхности сложной функции

Строится контурный график для представления на плоскости функции двух переменных вида $z(x,y)$ с помощью линий равного уровня, образованных пересечением рядом плоскостей, расположенных параллельно друг другу:

```
contour(x,y,z).
```

Контурные графики часто используются для представления на плоскости объемной функции. Для оценки высот контурных линий используется функциональная окраска. На построенном контурном графике (рис. 5.6) отчетливо видны область *минимальных* значений глубокого фиолетового цвета и область *максимальных* значений желтого цвета.

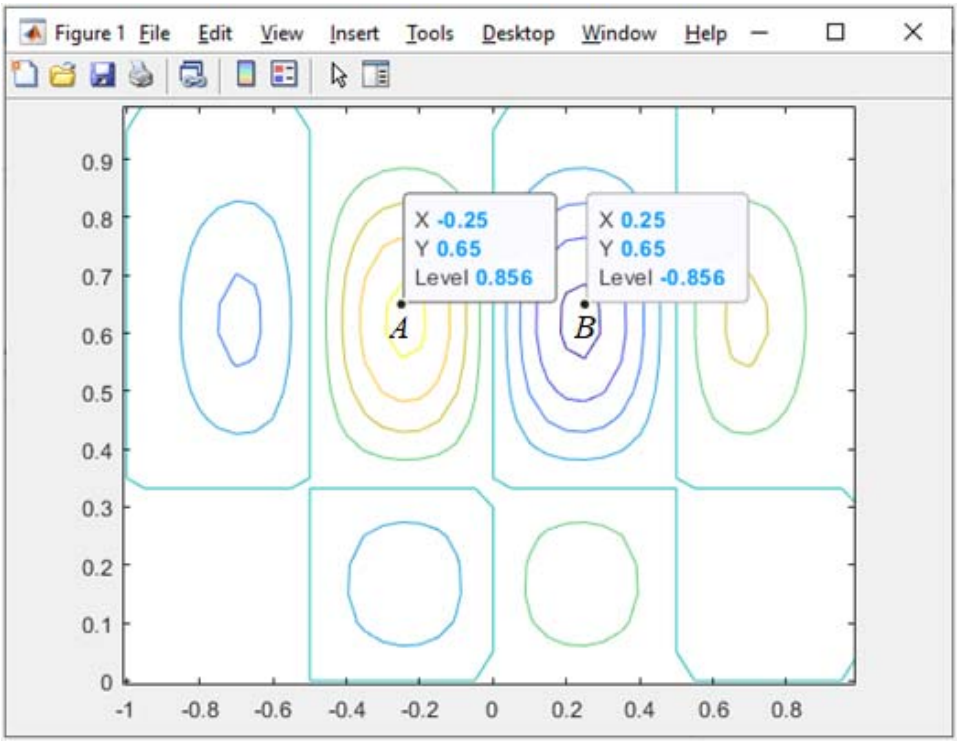


Рис. 5.6. Контурный график функции

3.2. Поиск *минимума* функции с помощью утилиты *Optimization Toolbox* системы *Matlab*

3.2.1. Создание *M*-файла и введение его в *GA*-редактор.

Целевая оптимизируемая *функция приспособленности* описывается в *M*-файле *@myfun*, который создается через меню *File – New – M-File*. Целевую функцию приспособленности при этом необходимо представить в таком виде, чтобы требовалась ее *минимизация*.

Например, для *минимизации* функции двух переменных
$f(x_1, x_2) = 4sin(2\pi x_1) * cos(1.5\pi x_2) * (1 - x_1^2) * x_2 * (1 - x_2)$
функция приспособленности в *M*-файле будет иметь вид:

```
function [z] = myfun(x)
z=4*sin(2*pi*x(:,1)).*cos(1.5*pi*x(:,2)).*(1-x(:,1).^2).*x(:,2).*(1-x(:,2))
end
```

Для корректной работы утилиты *M*-файл с целевой функцией либо должен находиться в одной из папок доступа *Matlab*, либо должен быть установлен путь к файлу в каталоге поиска *Matlab* с помощью следующей последовательности действий: *основное окно Matlab – File – Set Path – Add Folder – Save*.

Далее запускается графический интерфейс редактора ГА в утилите *Optimization Toolbox* системы *Matlab* с указанием решателя *Solver – ga-Genetic Algorithm* или выполнением команды *gatool.*

В открывшемся диалоговом окне графического интерфейса *GA*-редактора в панели *Problem* вводятся данные о целевой функции приспособленности (рис. 5.7).

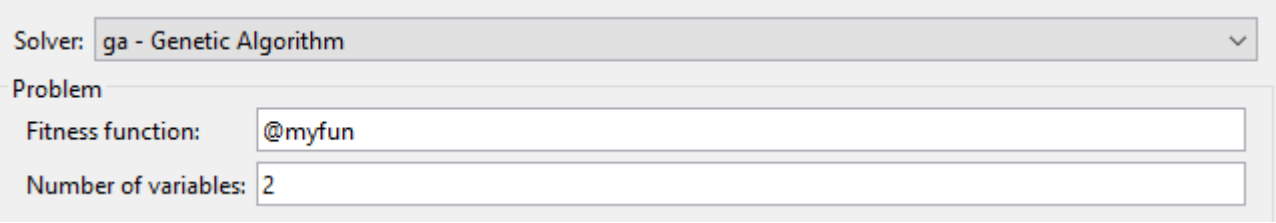


Рис. 5.7. Ввод функции приспособленности в *GA*-редакторе,

В поле *Fitness function* указывается имя оптимизируемой функции в виде *@myfun*, где *myfun.m* – имя подготовленного *M*-файла, а в поле *Number of variables* – длина входного вектора оптимизируемой функции, равная 2 для рассматриваемого примера.

3.2.2. Установление диапазона изменений аргументов функции.

Согласно начальным условиям аргументы целевой функции *x* и *y* находятся в границах диапазона [-1;1]: $-1 \le x \le 1,\ \ 0 \le y \le 1$.

Для установки этих границ *Bounds* необходимо в панели задания ограничений *Constraints GA*-редактора указать нижние *Lower* и верхние *Upper* пределы изменения границ первого и второго аргументов соответственно (рис. 5.8).

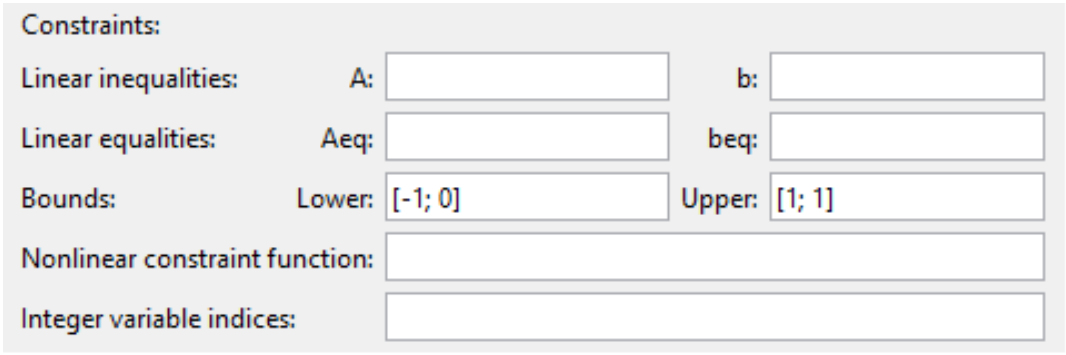


Рис. 5.8. Установка диапазонов изменения аргументов функции

3.2.3. Настройка основных опций *GA*-редактора.

При настройке параметров ГА основные параметры популяции ГА остаются заданными по умолчанию (рис. 5.9).

Параметры масштабирования, отбора, воспроизводства, мутации, скрещивания, миграции и др., а также условия остановки алгоритма (рис. 5.10) также остаются без изменений.

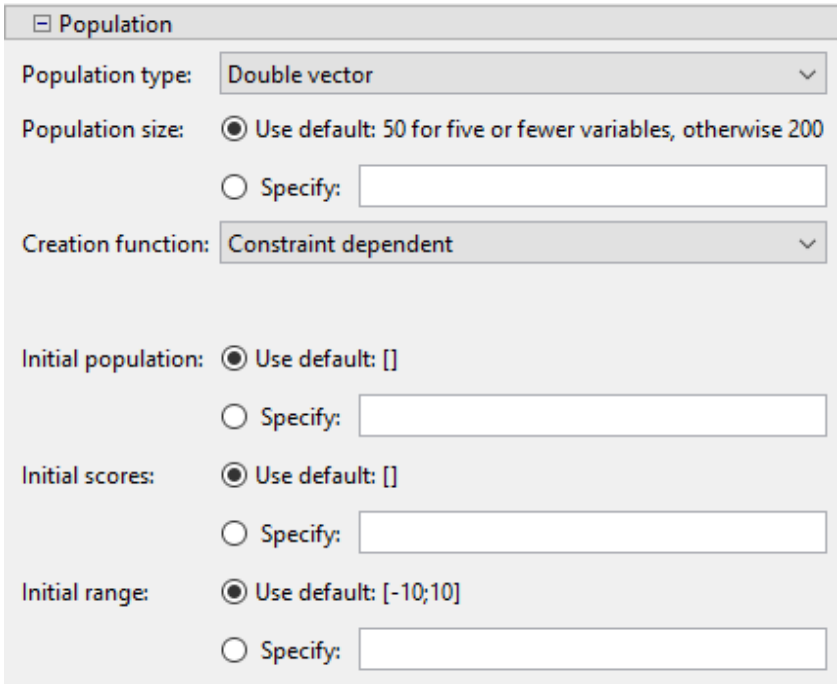


Рис. 5.9. Настройка параметров *Population*

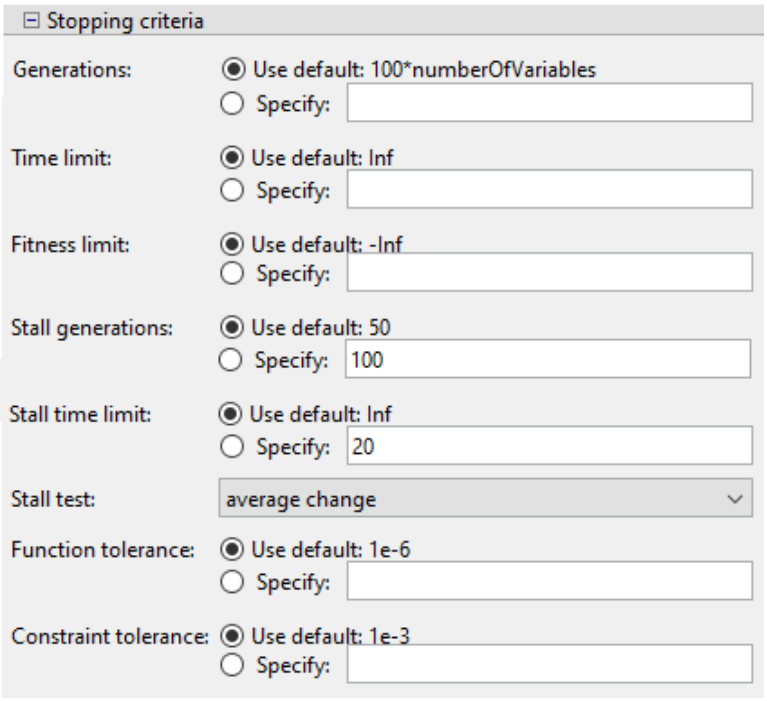


Рис. 5.10. Настройка параметров *Stopping Criteria Options*

В разделе *Plot functions* выбираются графики для отображения результатов оптимизации (рис. 5.11).

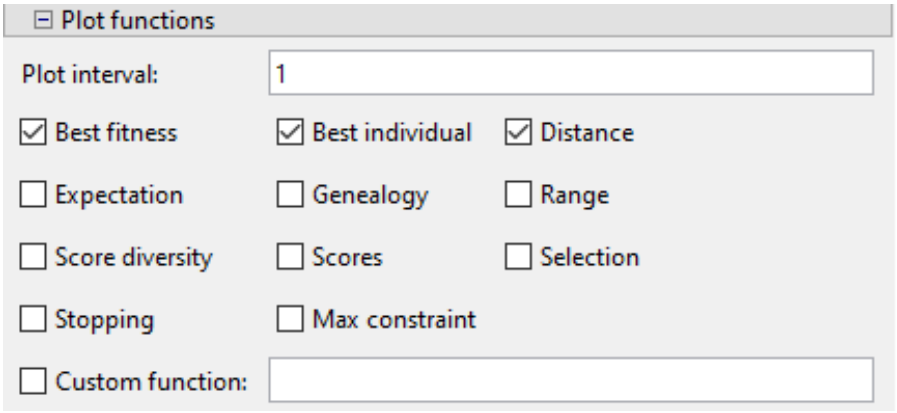


Рис. 5.11. Установка границ изменения аргументов функции

3.2.4. Запуск работы ГА.

Для запуска работы алгоритма нужно в разделе *Run solver and view results* нажать кнопку *Start*. По окончании вычислений в поле результатов появится информация (рис. 5.12), подтверждающая то, что оптимизация завершена, причина останова – среднее изменение значения приспособленности меньше, чем заданное в опции *FunctionTolerance,* которое по умолчанию равно 1e–6.

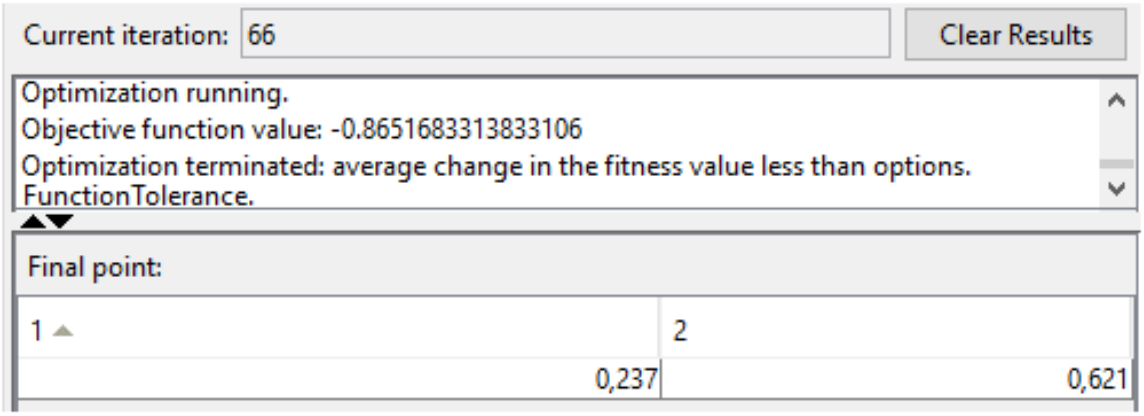


Рис. 5.12. Результаты поиска минимума функции

Результаты решения задачи оптимизации показали, что координаты найденной итоговой точки минимума (*Final point*) – [0,237 0,621]; значение целевой функции в данной точке равно –0,865.

Нетрудно видеть, что полученный результат хорошо соотносится с результатами, полученными при предварительном просмотре. Процесс вычислений демонстрируется на графиках (рис. 5.13). На графике *Fitness Value* показаны изменения значений целевой функции приспособленности: на 66-й итерации было найдено решение и ГА закончил работу.

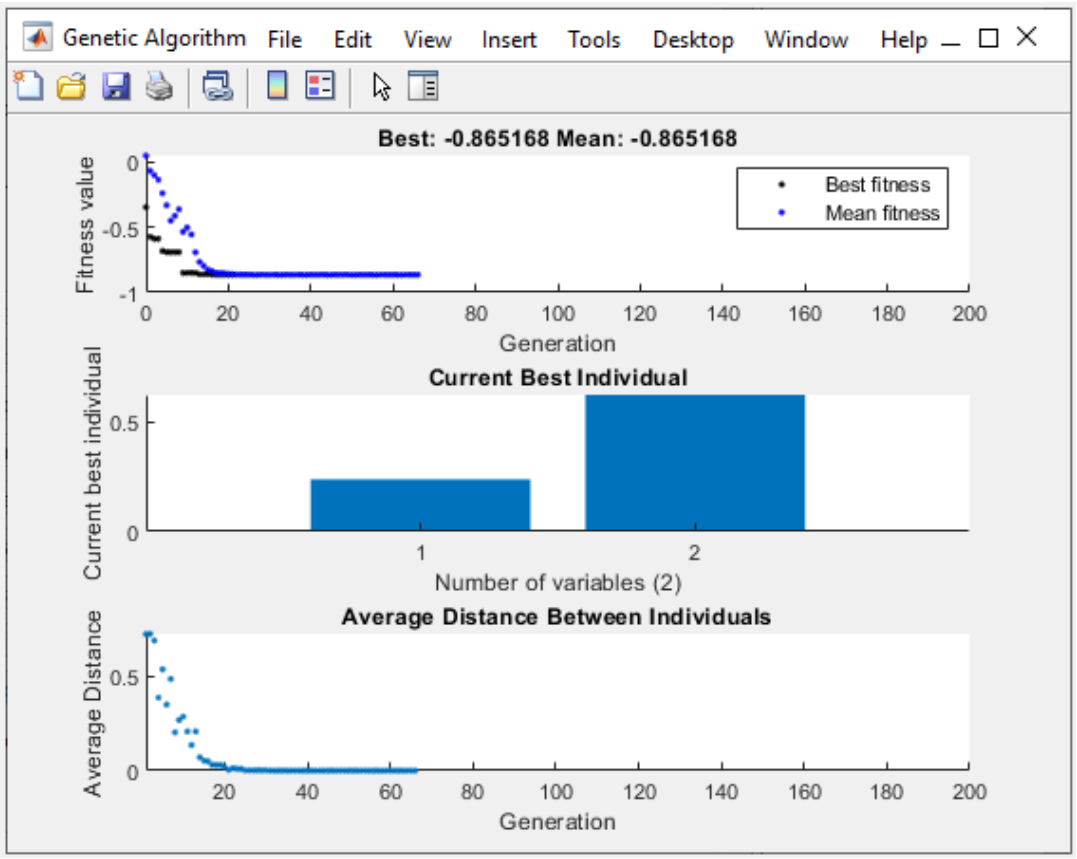


Рис. 5.13. Установка границ изменения аргументов функции

На графике *Current best individual* изображена наилучшая особь: столбиками показаны значения первого и второго аргумента. Третий график *Average Distance* показывает изменения расстояний между особями на разных итерациях: на последних итерациях изменения равны нулю.

3.3. Поиск *максимума* функции в утилите *Optimization Toolbox* системы *Matlab*

Для решения задачи нахождения максимума функции необходимо преобразовать целевую функцию по формуле $min\ f(x) \Leftrightarrow max\ [-f(x)]$:

$$f(x_1, x_2) = -(4sin(2\pi x_1) * cos(1.5\pi x_2) * (1 - x_1^2) * x_2 * (1 - x_2)),$$

т.е. в *M*-файле нужно записать:

```
function [z] = myfun(x)
z=-(4*sin(2*pi*x(:,1)).*cos(1.5*pi*x(:,2)).*(1-x(:,1).^2).*x(:,2).*(1-x(:,2)))
end
```

Если при этом оставить те же параметры ГА, что и для задачи нахождения минимума, то координаты точки максимума равны [-0,237 0,621]; значение целевой функции в данной точке после преобразования равно 0,865 (рис. 5.14).

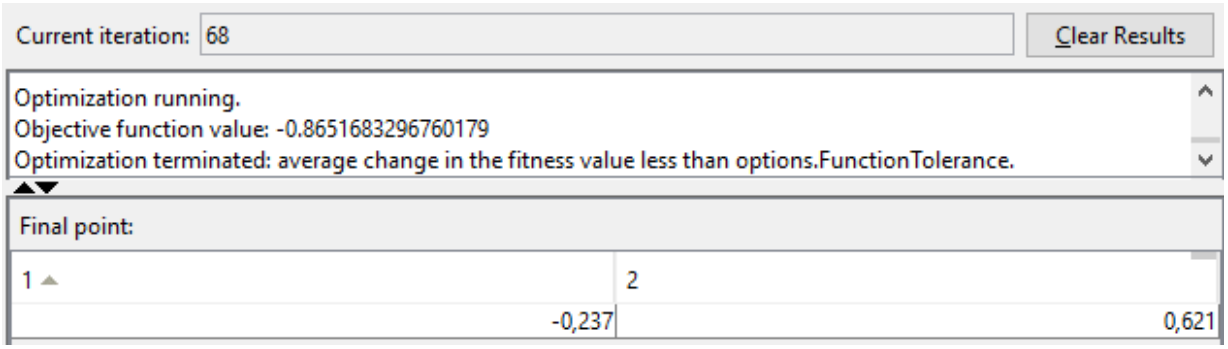


Рис. 5.14. Результаты поиска максимума функции

ГА является стохастическим, так как в алгоритме используются генераторы случайных чисел *rand* и *randn*, которые формируют случайные выборки для каждой итерации, поэтому выходные результаты повторных выполнений работы ГА могут отличаться.

Для воспроизведения результатов последнего запуска ГА следует выбрать команду *Use random states from previous run,* которая переустанавливает состояния генераторов произвольных чисел в их предыдущие значения и дает возможность проанализировать поведение алгоритма.

Порядок выполнения работы

1. Изучить методику решения задачи оптимизации с помощью генетических алгоритмов в *Matlab*.

2. Построить в *Matlab* трехмерный график поверхности и контурный график мультимодальной функции Растригина с параметрами, заданными по варианту задания (п. 3.1). Проанализировать форму функции и визуально определить точки минимума.

3. Подготовить *M*-файл с целевой функцией приспособленности.

4. Запустить графический интерфейс редактора ГА в системе *Matlab* (п. 3.2). Указать данные о целевой функции и пределы изменения границ первого и второго аргументов [-5; 5].

5. Выполнить поиск оптимального значения с помощью ГА (п. 3.1–3.2); определить координаты точки оптимума и значение целевой функции в данной точке. Вывести на графики *Plot Functions* значения наилучшей особи, изменения значений функции приспособленности и расстояний между особями на разных итерациях.

6. Провести экспериментальные исследования работы ГА, изменяя размер начальной популяции (10; 50; 100) и параметры ГА, такие как функция создания популяции *Creation function*, отбора *Selection*, воспроизведения *Reproduction*, мутации *Mutation*, скрещивания *Crossover* (табл. 5.5).

Таблица 5.5

Результаты экспериментальных исследований

№	Параметры ГА	Количество особей		
		10	*50*	*100*
1	*Creation function=Uniform* *Selection=Stochastic uniform* *Reproduction=default*	*89 iterations* x_1=0.995 x_2=0 z=0.995	*79 iterations* x_1=0 x_2=0.995 z=0.995	*69 iterations* x_1=0 x_2=0 z=0
2	*Creation function=Uniform* *Selection=Roulette*			
3	*Selection=Stochastic uniform* *Reproduction: ElitCount=5*			
4	*Reproduction: default* *Mutation: Uniform=0.02*			
5	*Mutation: default* *Crossover=single point*			
6	*Crossover=Arithmetic*			

Сделать вывод о том, как влияет размер исходной популяции на результаты, а также какое влияние вносят различные операторы отбора родительских особей.

Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

1. Название и цель работы. Постановка задачи оптимизации мультимодальной целевой функции с ограничениями согласно варианту задания.
2. Трехмерный график поверхности и контурный график функции, предварительно определенные точки оптимума.
3. Результаты поиска решения с помощью генетических алгоритмов в *Matlab*. Заполненная таблица результатов экспериментальных исследований.
4. Графики изменения значений целевой функции приспособленности на разных итерациях, значения наилучшей особи, изменения расстояний между особями на разных итерациях.
5. Выводы.

Варианты заданий

Найти *минимум* мультимодальной функции двух переменных – функции Растригина:
$f(x_1, x_2) = 20 + (x_1^2 - 10 * cos(2 * pi * x_1)) + (x_2^2 - 10 * cos(2 * pi * x_2))$,
где $-5 \le x_1 \le 5$, $-5 \le x_2 \le 5$ при изменении параметров, указанных в табл. 5.6.

Таблица 5.6

№	Функция						
$f(x_1, x_2) = a + (b * x_1^2 - c * cos(d * pi * x_1)) + (e * x_2^2 - f * cos(h * pi * x_2))$							
	a	*b*	*c*	*d*	*e*	*f*	*h*
0	*20*	*1*	*10*	*2*	*1*	*10*	*2*
1,11	25	1,1	11	2,05	1,2	12	2,1
2,12	30	0,9	12	1,9	1,1	7	2,2
3,13	22	0,8	7	1,8	0,9	14	2,3
4,14	15	1,2	14	1,5	0,8	15	2,4
5,15	14	1,1	15	2,1	1,2	9	1,7
6,16	21	0,9	9	2,2	1,1	9	1,6
7,17	19	0,8	8	2,3	0,9	8	2,05

Окончание табл. 5.6

8,18	18	1,2	13	2,4	0,8	13	1,9
9,19	23	1,1	12	1,7	1,2	12	1,8
10,20	28	0,9	10	1,6	1,1	10	1,5
11,21	25	0,8	6	2,05	0,9	6	2,1
12,22	30	1,2	10	1,9	0,8	10	2,2

Контрольные вопросы

1. Какие существуют особенности применения генетических алгоритмов при решении задач оптимизации.
2. Перечислите этапы работы ГА.
3. Назовите основные понятия ГА. Что такое функция приспособленности? Какие генетические операторы вы знаете?
4. Какой метод отбора используется в стандартном ГА?
5. Как работают операторы кроссинговера, мутации?

Список литературы

1. Карпенко, А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учеб. пособие / А.П. Карпенко. 2-е изд. М.: МГТУ им. Н.Э. Баумана, 2017. 446 с.
2. Бураков М. В. Генетический алгоритм: теория и практика: учеб. пособие. СПб.: ГУАП, 2008. 164 с.
3. Панченко Т.В. Генетические алгоритмы: учеб. пособие / под ред. Ю. Ю. Тарасевича. Астрахань: Издательский дом «Астраханский университет», 2007. 87 с.
4. Осипов Г. С. Лекции по искусственному интеллекту. М.: Книжный дом «ЛИБРОКОМ», 2016. 272 с.
5. Модели и методы искусственного интеллекта. Применение в экономике / М. Г. Матвеев, А. С. Свиридов, Н. А. Алейникова. М.: Финансы и статистика: ИНФРА-М, 2008. 446 с.
6. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия - Телеком, 2004. 452 с.
7. Ярушкина Н. Г. Основы теории нечетких и гибридных систем: учеб.пособие. М.: Финансы и статистика, 2004. 320с.
8. Саймон Д. Алгоритмы эволюционной оптимизации / Д. Саймон; перевод с английского А. В. Логунова. М.: ДМК Пресс, 2020. 940 с.