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

Кафедра технической кибернетики

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по дисциплине
«Методы искусственного интеллекта 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

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по дисциплине
«Методы искусственного интеллекта в управлении качеством»

Учебное электронное издание сетевого доступа

© УГАТУ, 2021

Уфа 2021

Авторы-составители: Л. Р. Черняховская, Г. И. Рыжов, А. Н. Павлова,
Р. И. Мухаметьянова

Методические рекомендации по дисциплине «Методы искусственного интеллекта в управлении качеством» [Электронный ресурс] / Уфимск. гос. авиац. техн. ун-т ; [авт.-сост. : Л. Р. Черняховская и др.]. – Уфа : УГАТУ, 2021. – URL: https://www.ugatu.su/media/uploads/MainSite/Ob%20universitete/Izdateli/El_izd/2021-90.pdf

Представлены теоретические сведения, модели и методы искусственного интеллекта в управлении качеством, которые необходимы для формирования у студентов компетенций в сфере интеллектуального анализа данных, разработки баз знаний и оценки рисков в процессах управления качеством, а также применения знаний в управлении организационными системами. Содержат порядок выполнения лабораторных работ и контрольные вопросы.

Предназначены для студентов направлений подготовки магистров 27.04.02 Управление качеством.

Рецензент канд. техн. наук, доцент Н. О. Никулина

При подготовке электронного издания использовались следующие программные средства:

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

Авторы-составители: *Черняховская Лилия Рашитовна,*
Рыжов Геннадий Иванович,
Павлова Анастасия Николаевна,
Мухаметьянова Регина Ильфатовна

Редактирование и верстка: *А. А. Шарипова*
Программирование и компьютерный дизайн: *А. П. Меркулова*

Подписано к использованию: 25.06.2021
Объем: 1,60 Мб.

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

Все права на размножение, распространение в любой форме остаются за разработчиком.
Нелегальное копирование, использование данного продукта запрещено.

Введение

Интеллектуальные информационные системы успешно применяются для решения практических задач и таких областях, как управление технологическими процессами, финансовый менеджмент, медицинская и техническая диагностика, биржевое прогнозирование, распознавание образов и других. Эффективное функционирование интеллектуальной системы в основном зависит от полноты и точности представления знаний в базе знаний. В методических рекомендациях рассмотрены методики разработки баз знаний интеллектуальных информационных систем для различных моделей представления знаний, алгоритмы обнаружения знаний в данных на основе построения деревьев решений, а также принципы функционирования и обучения нейро-нечеткой сети в составе системы нечеткого вывода.

Базы знаний разрабатываются с применением инструментальных средств разработки интеллектуальных систем: система обнаружения знаний в данных и построения деревьев решений *See5*, программная среда *MatLab* с подсистемой *Fuzzy Logic Toolbox*.

Методы искусственного интеллекта применяются в задачах управления качеством, а именно:

интеллектуальный анализ характеристик процесса управления качества позволяет выявить несоответствия признаков (продукции, услуг) требованиям качеству;

разработка баз знаний и применение правил по оценке качества и при выборе мероприятий по управлению рисками.

оценка рисков реализации производственных процессов в условиях неопределенности с применением теории вероятности и нечетких множеств.

Методические рекомендации содержат порядок выполнения лабораторных работ, контрольные вопросы и предназначен для студентов направлений подготовки 27.04.02 «Управление качеством».

ЛАБОРАТОРНАЯ РАБОТА № 1
ИССЛЕДОВАНИЕ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ С ПРИМЕНЕНИЕМ ПРОЦЕДУР НЕЧЕТКОГО КЛАСТЕРНОГО АНАЛИЗА

Цель работы

Целью работы является изучение интеллектуального анализа данных делового процесса с применением процедур нечеткого кластерного анализа.

Краткие теоретические сведения

Кластерный анализ – это общее название множества вычислительных процедур, используемых при создании классификации. В общей постановке проблема автоматической классификации объектов заключается в том, чтобы всю анализируемую совокупность объектов, статистически представленную в виде матрицы, разбить на сравнительно небольшое число однородных, в определенном смысле, групп, причем, число этих групп может быть неизвестно заранее.

1.1. Кластерный анализ данных с применением методов нечетких множеств

Нечеткий кластерный анализ позволяет приписать одни и те же объекты к разным группам с соответствующими степенями принадлежности. Алгоритм нечеткого кластерного анализа (англ. *Fuzzy C-Means*, *FCM*) применяется для нечеткого группирования данных, формируя группы, прототипы которых представляются точками в пространстве данных. Пусть данные, подлежащие группированию, представляются n-мерными векторами $x_k = [x_{k1}, x_{k2}, \dots, x_{kn}]^T, \quad x_k \in R_n, k = 1, \dots, M,$ состоящими из численных значений, характеризующих объекты. Множество M векторов образует матрицу X размерностью $n \times M$. Для заданного множества n входных векторов x_k и c выделяемых кластеров с центром v_i предполагается, что любой x_k принадлежит любому c_i с принадлежностью $\mu_{ik} \in [0, 1]$, где i – номер кластера, а k – номер

входного вектора. Идея алгоритма *FCM* заключается в минимизации критерия:

$$J(X,U,V)=\sum_{i=1}^{c}\sum_{k=1}^{M}(\mu_{ik})^{m}\cdot\|x_k-\nu_i\|_A^2$$

причем $U=[\mu_{ik}]\in Z2$ представляет собой матрицу декомпозиции множества *X*; матрица $V=[\nu_1,\dots,\nu_c]$ представляет центры кластеров, которые должны быть определены в результате выполнения алгоритма; $\mu_k\in R_n, i=1,\dots,c;$ $m\in(1,\infty)$ – коэффициент, характеризующий степень нечеткости образованных групп данных. Удаленность вектора x_k от центра группы ν_i определяется через стандартную Евклидову норму $D_{ik}^2\cdot A=\|x_k-\nu_i\|_A^2=(x_k-\nu_i)^T\cdot A\cdot(x_k-\nu_i)$. Для выполнения алгоритма на некотором множестве данных *X* необходимо выбрать количество групп *c*, степень нечеткости *m*, параметр ε в критерии остановки алгоритма, а также случайным образом инициировать матрицу $U(0)\in Z2$ и вектор прототипов группы $V(0)$.

Различные приложения кластерного анализа можно свести к четырем основным задачам:

1) разработка типологии или классификации;

2) исследование полезных концептуальных схем группирования объектов;

3) порождение гипотез на основе исследования данных;

4) проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

1.2. Программное обеспечение кластерного анализа в пакете *MatLab*

Для начала работы в программной системе *MatLab* данные с *Excel* необходимо представить в формате *CSV* (разделители запятые, **.csv*), а затем с применением инструмента «Блокнот» разделители в числовых данных заменить с «запятой» на «точку» и создать новый файл данных в *MatLab* (**.dat*).

Программа нечеткого кластерного анализа входит в подсистему нечеткой логики *Fuzzy Logic Toolbox* в программной системе *MatLab*.

Пример решения задач нечеткого кластерного анализа методом *FCM* в среде *MatLab* с использованием графического интерфейса пользователя показан на рис. 1.1. Вычисления производились с помощью встроенной функции *fcm*.

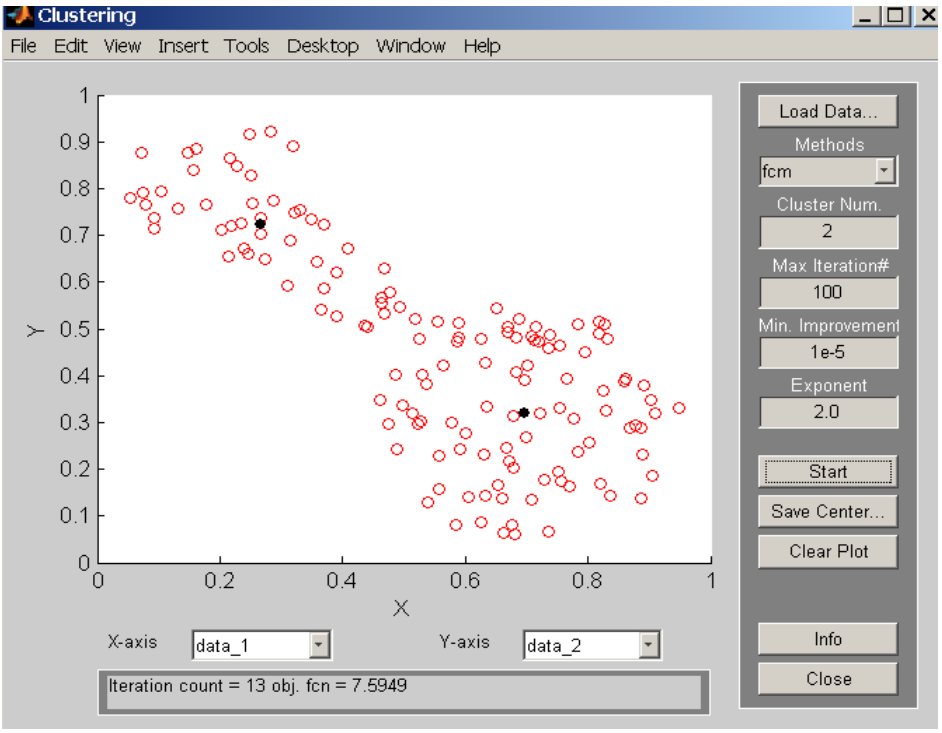

Рис. 1.1. Результаты кластерного анализа состояний технического объекта (критических и нормальных) методом *FCM* в среде *MATLAB*

На рис. 1.1 видно, что данные сгруппированы в два кластера. Проведенные исследования показывают, что необходим анализ возможностей применяемых нечетких моделей для решения конкретных задач в рассматриваемой проблемной области. Подобный анализ применяется для группирования данных, обладающих общими свойствами. Методы группирования данных представляют собой очень важный исследовательский инструмент искусственного интеллекта и имеют многочисленные приложения.

Задание на лабораторную работу

1. Подготовить исходные данные в *Excel*.
2. Загрузить *SPSS*, сформировать файл данных.
3. Провести кластерный анализ характеристик деловых процессов с применением статистического метода классификации.

4. Загрузить *MatLab*, запустить алгоритм нечеткого кластерного анализа.

5. Провести анализ и сравнение результатов кластерного анализа с применением статистического метода и метода нечеткого кластерного анализа.

6. Оформить полученные результаты.

Требования к отчету

Отчет должен содержать:

1. Титульный лист.
2. Задание.
3. Цель работы.
4. Исходные данные для выполнения работы, в соответствии с заданным вариантом.
5. Основные результаты выполнения работы.
6. Выводы по работе.

Контрольные вопросы и задания

1. Дайте определение кластерного анализа.
2. Дайте определение евклидовой метрики.
3. В чем отличие матрицы степеней принадлежности четкого разбиения от нечеткого?
4. Перечислите этапы алгоритма *FCM*.
5. Какой параметр определяет «размазанность» кластеров?
6. Как определяются центры кластеров?
7. Как рассчитываются расстояния между объектами из X и центрами кластеров?
8. Назовите существующие формальные подходы к выбору числа кластеров.
9. Как происходит останов алгоритма К-средних?
10. Сравните результаты кластерного анализа, полученные статистическим методом и методом нечеткого кластерного анализа. Поясните сходства и различия.

ЛАБОРАТОРНАЯ РАБОТА № 2
ОБНАРУЖЕНИЕ ЗНАНИЙ В ДАННЫХ С ПОМОЩЬЮ ПОСТРОЕНИЯ ДЕРЕВЬЕВ РЕШЕНИЙ

Цель работы

Целью работы является изучение методики обнаружения знаний в данных на основе построения деревьев решений в заданной предметной области с применением системы *See5*.

Краткие теоретические сведения

Деревья решений являются достаточно распространенным в настоящее время подходом к обнаружению знаний в данных. Каждому узлу сопоставлен некоторый признак, а ветвям – либо конкретные значения качественных признаков, либо области значений количественных признаков. Дерево решений позволяет строить модель зависимости множества исходов от множества характеристических признаков.

Для построения дерева решений требуется совокупность объектов в заданной предметной области. Каждый объект описывается набором характеристических признаков $X=(x_1, x_2, \ldots, x_n)$ и классифицирующим признаком (решением) D, который задает принадлежность объекта к одному их диагностических классов. Деревья решений разбивают объекты на группы на основе значений тех или иных классифицирующих признаков. В результате получается иерархия продукционных правил, которые классифицируют данные. Корню дерева соответствует самый информативный характеристический признак. Далее, в вершинах располагаются признаки в порядке уменьшения значений прироста информативности.

При генерации дерева решений используются свойства атрибутов, которые можно сформулировать в следующем виде:

$$\forall t_i, t_j \, (t_i(D) \neq t_j(D) \rightarrow t_i(X) \neq t_j(X)),$$
$$\forall t_i, t_j \, (t_i(X) = t_j(X) \rightarrow t_i(D) = t_j(D)) \qquad (2.1)$$

где t_i и t_j – i-ая и j-ая строки таблицы соответственно;

$t_i(D)$ – значение решения для строки t_i;

$t_i(X)$ – набор значений множества характеристических признаков X для строки t_i.

На каждом шаге атрибут можно выбирать произвольно, но существует наилучший выбор. Критерием такого выбора является формула (2.1). Из всех возможных атрибутов в первую очередь выбирается тот, который имеет максимальное число пар строк таблицы, для которых при неравных решениях также не равны значения самого признака. После генерации дерева решений инженер знаний может минимизировать список признаков, удалив те признаки, которые не использовались при построении дерева решений.

В использовании деревьев решений существуют следующие ограничения: подобные модели неприемлемы при обработке большого количества непрерывных величин; кроме того, если список условий длинный и сложный, дерево решений получается необозримым и непонятным.

Построение дерева решений – распространенный подход к выявлению и изображению логических закономерностей в данных. Разработанное дерево решений предназначено для обучения базы знаний новым продукционным правилам. В настоящее время известны несколько десятков компьютерных программ для построения деревьев решений. К ним относятся *CHAID* (*Chi Square Automatic Interaction Detection*), *CART* (*Classification and Regression Trees*), *ID3* (*Interaction Dichotomizer*), *See5/C5.0* и *WizWhy*.

2.1. Описание системы *See5/C5.0*

Система *See5/C5.0* компании *RuleQuest* предназначена для анализа больших баз данных, содержащих до сотни тысяч записей и до сотни числовых или номинальных полей. Результат работы *See5* выражается в виде деревьев решений и множества *if-then*-правил. Система проста в обращении и не требует от пользователя специфических знаний в области прикладной статистики. Некоммерческая версия для обучения ограничена количеством анализируемых записей (до 200).

2.2. Построение дерева решений в системе *See5*

Основные этапы обработки и анализа данных при построении дерева решений в системе *See5* проиллюстрируем на конкретном примере. Рассмотрим задачу оценки кредитного риска на основе следующих критериев: возраста, трудового стажа, дохода, категории работодателя, кода консультанта и кредитной истории. Выборка содержит 198 объектов. Фрагмент выборки представлен в табл. 2.1.

Таблица 2.1

Данные о кредитном риске

№№	Возраст, год	Трудовой стаж, месяц	Доход, ×1000 руб.	Категория работодателя	Код консультанта	Кредитная история	Значения
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
1	21–50	3-12	5–10	ИП	да	Отрицательная	отказ
2	21–50	3–12	5–10	ИП	нет	отсутствует	необходима доп. проверка
3	21–50	3–12	5-10	ЧК	да	Положительная	отказ
4	21–50	3–12	5–10	ГК	да	Отрицательная	отказ
5	21–50	3–12	> 20	ИП	да	Отрицательная	отказ
6	21–50	3–12	> 20	ИП	нет	отсутствует	необходима доп. проверка
7	21-50	3–12	> 20	ЧК	нет	Положительная	пол. решение
8	21–50	> 12	5–10	ИП	нет	Положительная	необходима доп. проверка
9	21–50	> 12	5-10	ЧК	да	Положительная	отказ
10	21–50	> 12	5–10	ГК	нет	Положительная	отказ
11	21–50	> 12	5–10	ГК	нет	отсутствует	необходима доп. проверка
12	21–50	> 12	5–10	ГК	нет	Положительная	положительное решение

Окончание табл. 2.1

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
13	21–50	> 12	> 20	ИП	нет	отсутствует	необходима доп. проверка
14	21–50	> 12	> 20	ИП	нет	Положи-тельная	необходима доп. проверка
15	21–50	> 12	> 20	ЧК	да	Отрица-тельная	отказ
16	21–50	> 12	> 20	ГК	нет	Отрица-тельная	отказ
17	51–65	3–12	5–10	ИП	нет	отсутствует	необходима доп. проверка
18	51–65	3–12	5–10	ЧК	нет	положи-тельная	необходима доп. проверка
19	51–65	3–12	5–10	ГК	да	отрица-тельная	отказ
20	51–65	3–12	10–20	ЧК	да	отрица-тельная	отказ
21	51–65	> 12	5–10	ГК	нет	положи-тельная	положитель-ное решение
22	51–65	> 12	10–20	ИП	да	отрица-тельная	отказ
23	51–65	> 12	10–20	ИП	да	отсутствует	отказ

В табл. 2.1 приняты следующие обозначения:
– ИП – индивидуальный предприниматель;
– ЧК – частная компания;
– ГК – государственная компания.

В табл. 2.2 приведены обозначения и описание, используемых в работе переменных.

Таблица 2.2

Обозначение и описание используемых переменных

Воз-раст, год	Тру-довой стаж, месяц	Доход, ×1000 руб.	Катего-рия рабо-тодателя	Код консуль-танта	Кредитная история	Решение
1	*2*	*3*	*4*	*5*	*6*	*7*
age	*stag*	*income*	*employer type*	*consultant ID*	*credit history*	*credit decision*
21-50	< 3	< 5	ИП	да	отрицательная	отказ

Окончание табл. 2.2

1	*2*	*3*	*4*	*5*	*6*	*7*
51-65	3-12	5-10	ЧК	нет	отсутствует	необходима доп. проверка
	> 12	10-20	ГК		положитель-ная	положитель-ное решение
		> 20				

Построение дерева решений включает следующие этапы.

Первый этап – подготовка данных для анализа.

Подготовка данных для анализа в системе *See5* предполагает создание двух обязательных файлов: файла имен переменных и файла данных.

Файл имен переменных содержит перечисление имен разделяющих признаков и указание классифицирующего признака. Файл создается в любом текстовом редакторе и сохраняется с расширением **.names*.

Файл имен переменных *exam.names* в нашей задаче выглядит следующим образом (рис. 2.1).

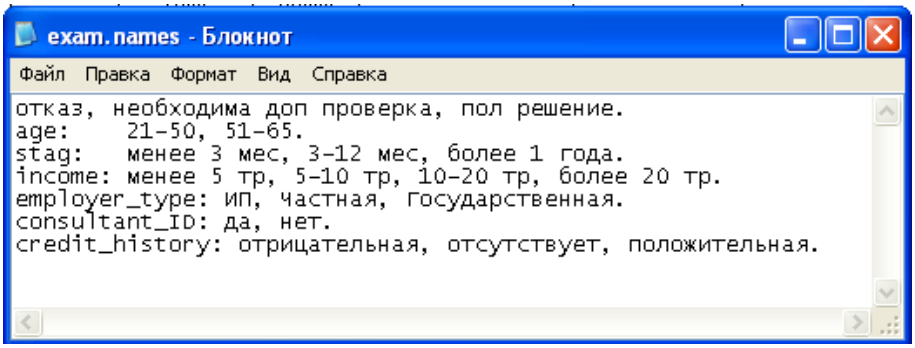

Рис. 2.1. Файл имен переменных

Среди признаков различают две важные подгруппы:

1) номинальные признаки (*discrete attribute*), количественные признаки (*continuous attribut*e) и метки;

2) явно определенные признаки, значения которых берутся непосредственно из файла данных, и неявно определенные признаки, задаваемые формулами (чаще всего употребляются явно определенные признаки).

При подготовке файла имен переменных следует иметь в виду, что пробелы, пустые строки и знаки табуляции игнорируются системой (кроме, конечно, случаев, когда они применяются в именах

переменных). Вертикальная черта «|» предназначена для записи напоминаний или комментариев.

После имени каждой явно определенной переменной вставляется двоеточие «:», а затем следует характеристика этой переменной. Возможны следующие характеристики:

– *continuous* – количественный признак;

– список значений переменной, разделенных запятой (для дискретной, номинальной переменной);

– максимальное значение *N* для дискретной переменной (эту характеристику рекомендуется применять очень осторожно, так как здесь исключается дополнительная проверка данных при их вводе в анализ);

– *ignore* – для признака, исключаемого из анализа;

– *label* – метка для идентификации отдельного объекта.

После имени каждой неявно определенной переменной также следует двоеточие и далее записывается формула. В формуле используются, где необходимо, скобки, а дискретные признаки ограничиваются кавычками. Ниже приведены доступные операторы:

– «+», «-», «*», «/», «%» (*mod)*, «^» (возведение в степень);

– «>», «>=», «<», «<=», «=», «<>» или «!=» (не равно);

– *and*, *or*;

– sin(...), cos(...), tan(...), log(...), exp(...), int(...).

В зависимости от применяемой формулы конечный результат может быть, как количественным, так и давать логическое значение *true/false*.

Файл данных содержит сведения об объектах. В файле по строкам располагаются объекты, а по столбцам признаки, причем в том порядке, в котором они заданы в файле имен переменных. Если значение целевой переменной находится вверху файла имен переменных, строка начинается со значения этой целевой переменной. Затем через запятую следуют значения всех остальных признаков. Файл создается в любом текстовом редакторе и сохраняется с расширением **.data*.

Файл данных *exam.data* в нашей задаче выглядит следующим образом (рис. 2.2).

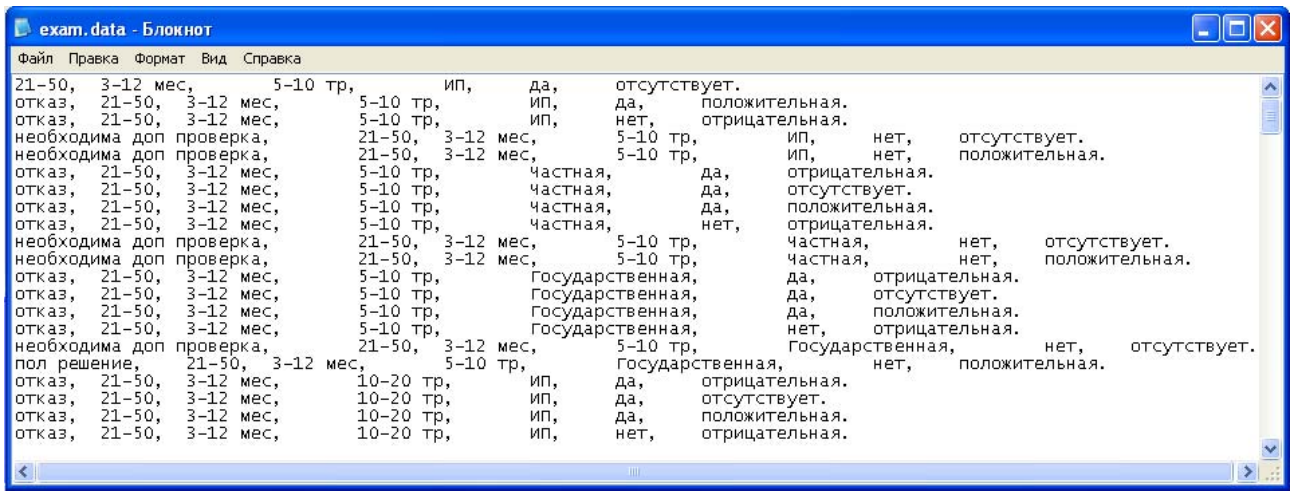

Рис. 2.2. Файл данных

Второй этап – задание начальных параметров и построение дерева решений.

В главном окне *See5* располагаются кнопки (рис. 2.3), предназначение которых представлено в табл. 2.3.

Все перечисленные функции доступны также из меню *File*. В свою очередь, в меню *Edit* предоставляется возможность редактирования файла имен данных и файла стоимости ошибок классификации. Для построения дерева решений необходимо загрузить данные из файла данных в систему *See5* (*Locate Data*) и построить классификатор (*Construct Classifier*).

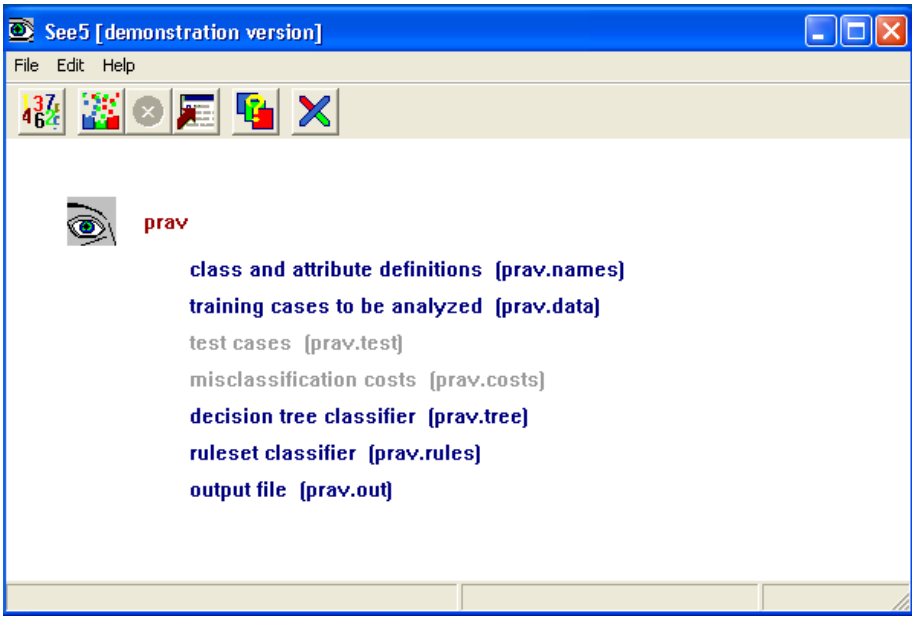

Рис. 2.3. Главное окно *See5*

Таблица 2.3

Назначение кнопок меню

Кнопка	Назначение
Locate Data	Вызов окна для просмотра доступных файлов данных и их загрузки в систему
Construct Classifier	Обращение к окну диалога для выбора типа классификатора и установки его параметров
Stop	Останов процесса построения дерева решений
Use Classifier	Запуск процесса интерактивной классификации одного или более объектов
Cross-Reference	Вызов окна, в котором наглядно раскрываются связи между объектами обучающей выборки и найденными правилами их классификации

Третий этап – анализ полученного дерева решений. Анализ полученного дерева решений производится на основе сформированного в *See5* отчета.

```
See5 [Release 1.14]     Tue May 25 00:28:41 2010

    Options:
      Generating rules

Read 198 cases (6 attributes) from exam.data

Decision tree:

consultant_ID = да: отказ (99)
consultant_ID = нет:
:...credit history = отрицательная: отказ (33)
    credit_history = отсутствует: необходима доп проверка (33/1)
    credit_history = пол решение:
    :...employer_type = ИП: проверка (11/1)
        employer_type = Государственная: пол решение (11)
        employer_type = Частная:
        :...stag = более 1 года: пол решение (5)
            stag = 3-12 мес:
            :...age = 21-50: пол решение (3/1)
                age = 51-65: проверка (3)

Evaluation on training data (198 cases):

          Decision Tree          Rules
        ----------------    ----------------
        Size      Errors      No      Errors

          8    3( 1.5%)       6    3( 1.5%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
         132                 (a): class отказ
               45     1     (b): class необходима доп проверка
                2    18     (c): class пол решение


Time: 0.2 secs
```

В первой строке отчета о результатах дается информация об используемой версии системы *See5* и текущее время. Затем в следующих двух строках говорится о том, что прочтенный файл данных *exam.data* содержит 198 объектов, каждый из которых описан шестью признаками.

В следующих строках отчета отображено построенное дерево решений.

Каждая ветка дерева заканчивается указанием номера класса, к которому она приводит. Сразу за номером следует запись вида («) или (*n/m*). Например, самая первая ветка заканчивается записью (99). Это означает, что данной ветке соответствует 99 объектов из определенного класса (отказ). Величины *n* или *m* могут оказаться дробными в случае, когда на какую-либо ветку придется некоторое число объектов с неизвестными значениями признаков.

В следующем разделе отчета приводятся характеристики сконструированного классификатора, оцениваемые на обучающей выборке. Здесь мы видим, что построенное дерево решений имеет восемь веток (*size* = 8), а ошибка классификации наблюдается на трех объектах, что составляет 1,5 %.

В завершающей части отчета дается таблица с детальным разбором результатов классификации. Исходя из данных этой таблицы, можно сказать, что из первого класса (отказ) правильно классифицируются 132 объекта; среди объектов второго класса (необходима дополнительная проверка) 45 диагностируются правильно, а два ошибочно (класс 3, положительное решение); все объекты третьего класса (положительное решение) классифицируются правильно за исключением одного ошибочно классифицированного объекта, попадающего в класс 2.

В заключение система *See5* выдает сообщение о затраченном на решение времени. Здесь надо отметить вообще очень высокую скорость работы алгоритма *See5*, позволяющую оперативно обрабатывать высоко размерные массивы информации, содержащие тысячи и десятки тысяч записей.

Четвертый этап – преобразование дерева решений в набор правил.

В ряде случаев полученное дерево решений может оказаться слишком сложным для восприятия. Например, при решении задач высокой размерности для неоднородных данных дерево нередко

получается кустистое и довольно запутанное. Вместо того чтобы «ползать» по каждой полученной ветке, в системе *See5* предусмотрена возможность преобразования дерева решений в набор правил *IF...THEN*. Для этого требуется вызвать окно диалога для заданий параметров конструируемого алгоритма (*Construct Classifier*) и поставить флажок в поле *Rulesets* (набор правил). После проведения такой операции система добавляет в окно отчета список правил, соответствующих рассчитанному дереву решений.

```
Extracted rules:

Rule 1: (99, lift 1.5)
        consultant_ID = да
        ->  class отказ  [0.990]
Rule 2: (66, lift 1.5)
        credit_history = отрицательная
        ->  class отказ  [0.985]
Rule 3: (33/1, lift 4.1)
        consultant_ID = нет
        credit_history = отсутствует
        ->  class необходима доп проверка [0.943]
Rule 4: (11/1, lift 3.6)
        employer_type = ИП
        consultant_ID = нет
        credit history = положительная
        ->  class необходима доп проверка [0.846]
Rule 5: (3, lift 3.4)
        age = 51-65
        stag = 3-12 мес
        employer_type = Частная
        consultant_ID = нет
        credit_history = положительная
        ->  class необходима доп проверка [0.800]
Rule 6: (33/14, lift 5.7)
        consultant_ID = нет
        credit_history = положительная
        ->  class пол решение [0.571]

Default class: отказ
```

Каждое правило состоит из следующих фрагментов:

– номера правила;

– количества объектов обучающей выборки, подпадающих под действие правила («*n*»);

– одного или нескольких элементарных логических событий, входящих в состав правила;

– номера класса, которому соответствует данное правило;

– величины, принимающей значение от 0 до 1, которая выражает степень доверия к правилу (характеристика точности правила).

Пятый этап – более подробный анализ результатов классификации.

Более подробный анализ результатов классификации проводится при помощи перекрестных ссылок. Для этого необходимо в главном окне *See5* нажать кнопку *Cross-Reference* (перекрестная

ссылка). Система выдаст окно, в левой половине которого нарисовано построенное дерево решений, а в правой половине перечисляются объекты, попавшие на ту или иную ветвь дерева (рис. 2.4). Чтобы выделить интересующую ветвь, нужно щелкнуть по ней левой кнопкой мыши (справа от ветви появится темный круг). Кроме того, если щелкнуть мышью по номеру какого-либо объекта из правого поля, то система выдаст еще одно окно с именем *Case*, в котором приводятся значения признаков и выделенного объекта.

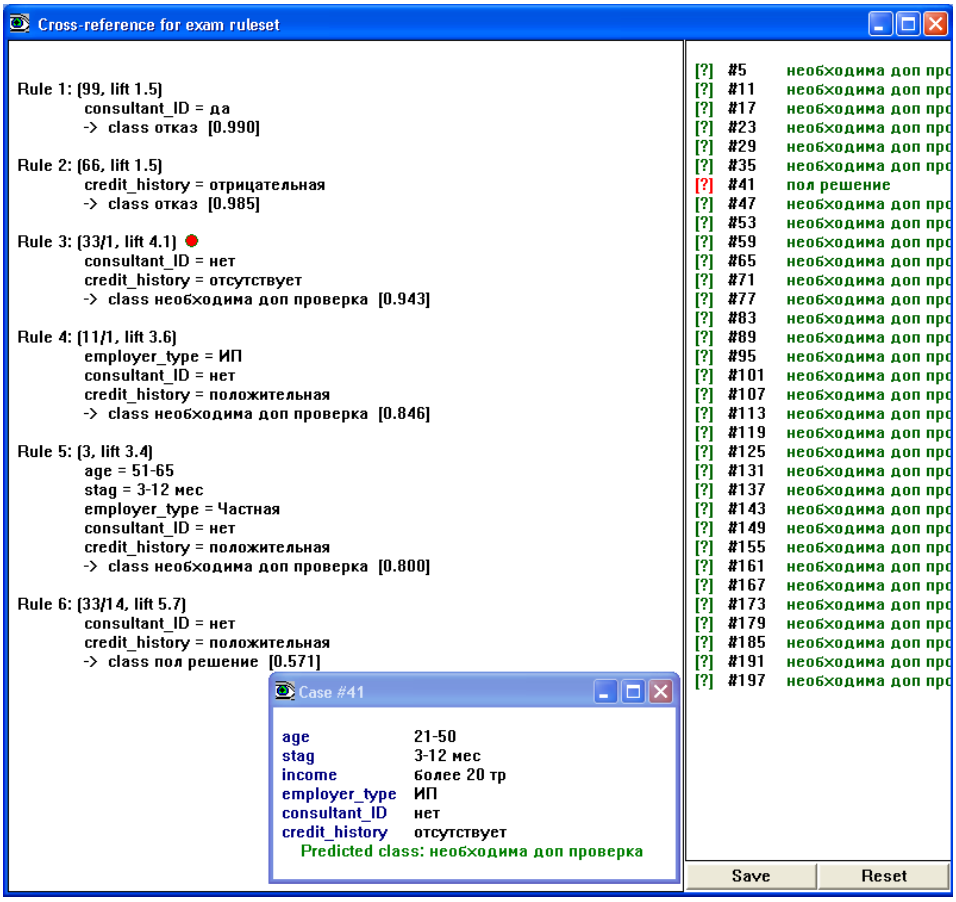

Рис. 2.4. Окно задания перекрестных ссылок

Шестой этап – усиление решения.

В системе *See5* реализована возможность усиление решения. Идея усиления решения заключается в конструировании не одного, а сразу нескольких деревьев решений. Главное требование к деревьям решений заключается в том, чтобы они как можно меньше дублировали друг друга. Для этого необходимо на первом шаге построить начальное дерево решений. При этом классификатор, построенный на основе начального дерева, дает ошибки на некоторых объектах.

На втором шаге при конструировании следующего дерева делается попытка избежать ранее сделанных ошибок. Следствием такой попытки считается существенное отличие второго дерева от начального. Полученное дерево также будет приводить к ошибочным решениям, но уже на других объектах. На следующем шаге работы алгоритма очередное дерево строится с учетом ошибок всех предыдущих деревьев решений.

Для запуска процесса усиления решения требуется установить флажок *Boost* в диалоговом окне для задания параметров работы алгоритма. Кроме того, в этом же окне нужно задать общее число строящихся деревьев решений. Это число проставляется в поле *trials*.

В результате построения такой совокупности деревьев решений значительно повышается точность классификации. Разработчики *See5* утверждают, что при использовании десяти деревьев решений ошибки классификации снижаются в среднем на 25 %.

В рассматриваемом примере после проведения усиления решения (построения трех деревьев решений) количество ошибочно классифицированных объектов снизилось с 1,5% до 0,5%. Ошибочно классифицированным оказался лишь один объект.

```
See5 [Release 1.14]     Tue May 25 09:11:43 2010

    Options:
      Boost using 3 trials

Read 198 cases (6 attributes) from exam.data

-----  Trial 0:  -----

Decision tree:
consultant_ID = да: отказ (99)
consultant_ID = нет:
:...credit_history = отрицательная: отказ (33)
    credit_history = отсутствует: необходима доп проверка (33/1)
    credit_history = положительная:
    :...employer_type = ИП: необходима доп проверка (11/1)
        employer_type = Государственная: пол решение (11)
        employer_type = Частная:
        :...stag = более 1 года: пол решение (5)
            stag = 3-12 мес:
            :...age = 21-50: пол решение (3/1)
                age = 51-65: необходима доп проверка (3)

-----  Trial 1:  -----

Decision tree:
consultant_ID = да: отказ (86.8)
consultant_ID = нет:
:...credit_history = отрицательная: отказ (28.9)
    credit_history = отсутствует:
    :...income = 5-10 тр: необходима доп проверка (10.5)
    :   income = 10-20 тр: необходима доп проверка (10.5)
    :   income = более 20 тр:
    :   :...employer_type = ИП: пол решение (10.8/1.8)
    :       employer_type = Частная: необходима доп проверка (2.6)
    :       employer_type = Государственная: необходима доп проверка (2.6)
    credit_history = положительная:
    :...income = 5-10 тр:
```

```
        :...employer_type = ИП: необходима доп проверка (3.5)
        :   employer_type = Частная: необходима доп проверка (11.6/1.8)
        :   employer_type = Государственная: пол решение (3.5)
        income = 10-20 тр:
        :...employer_type = ИП: необходима доп проверка (3.5)
        :   employer_type = Частная: пол решение (3.5/0.9)
        :   employer_type = Государственная: пол решение (3.5)
        income = более 20 тр:
        :...age = 21-50: пол решение (13.4/0.9)
            age = 51-65: необходима доп проверка (2.6/0.9)

-----  Trial 2:  -----

Decision tree:

consultant_ID = да: отказ (77.7)
consultant_ID = нет:
:...credit_history = отрицательная: отказ (25.9)
    credit_history = отсутствует: необходима доп проверка (38.9/6.9)
    credit_history = положительная:
    :...employer_type = Государственная: пол решение (12.1)
        employer_type = ИП:
        :...stag = 3-12 мес: пол решение (10.9/3.9)
        :   stag = более 1 года: необходима доп проверка (7.4)
        employer_type = Частная:
        :...stag = 3-12 мес: необходима доп проверка (14.3/1.6)
            stag = более 1 года: пол решение (10.8)

*** warning: boosting may be unhelpful

Evaluation on training data (198 cases):

Trial       Decision Tree
-----     ----------------
          Size      Errors
   0        8    3( 1.5%)
   1       15    7( 3.5%)
   2        8    8( 4.0%)
boost             1( 0.5%)   <<

          (a)   (b)   (c)    <-classified as
          ----  ----  ----
          132                (a): class отказ
                46           (b): class необходима доп проверка
                 1    19     (c): class пол решение

Time: 3.3 secs
```

Седьмой этап – выполнение проверки эффективности построенной системы.

В режиме консультации выполняется проверка эффективности построенной системы при помощи команды *File*, *Use classifier*. После выполнения команды станет доступным окно задания исходных значений переменных, результат анализа которых будет сформирован в виде рекомендуемого решения с коэффициентом уверенности, рис. 2.5.

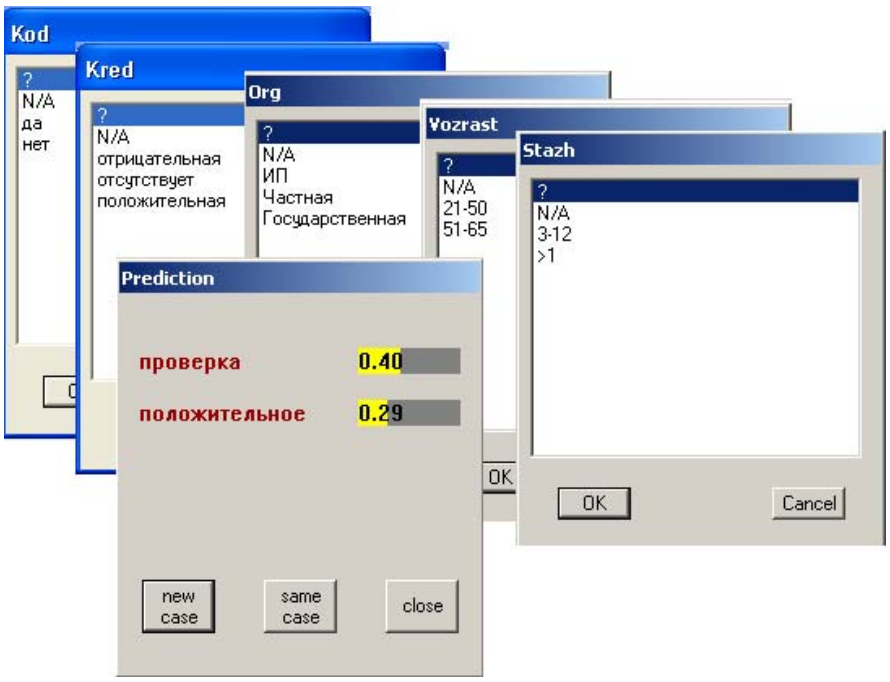

Рис. 2.5. Режим консультации

Задание на лабораторную работу

1. Изучить методику обнаружения знаний в данных на основе построения деревьев решений в заданной предметной области в системе *See5*.
2. Подготовить данные для анализа в выбранной предметной области. Сформировать файл данных и файл имен переменных.
3. Загрузить систему *See5*.
4. Построить дерево решений, выполнить его анализ.
5. Преобразовать дерево решений в набор правил. Дать характеристику правилам.
6. Выполнить подробный анализ результатов классификации при помощи перекрестных ссылок.
7. Выполнить усиление решения. Дать характеристику полученным результатам.
8. Выполнить проверку эффективности разработанной системы в режиме консультации.
9. Оформить отчет, который должен содержать: цель работы, описание задачи, файл имен переменных, файл данных, дерево решений, набор правил, раздел формирования ошибок, перекрестные ссылки, результаты работы в режиме консультации.

Требования к отчету

Отчет должен содержать:
1. Титульный лист.
2. Задание.
3. Цель работы.
4. Исходные данные для выполнения работы, в соответствии с заданным вариантом.
5. Основные результаты выполнения работы.
6. Выводы по работе.

Контрольные вопросы и задания

1. Каково назначение деревьев решений?
2. Назовите основные этапы анализа данных в системе *See5*
3. Назовите режимы работы системы *See5*.
4. Какие методы используются для интеллектуального анализа данных?
5. Как можно охарактеризовать программно-прагматическое направление искусственного интеллекта?
6. Какая модель представления знаний основана на опыте экспертов?
7. Охарактеризуйте работу системы в режиме консультации.
8. Какая модель используется для представления знаний в программе *See5*?

ЛАБОРАТОРНАЯ РАБОТА № 3
РАЗРАБОТКА БАЗЫ ЗНАНИЙ ДЛЯ СИСТЕМЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА В СРЕДЕ *MATLAB*

Цель работы

Целью работы является освоение методики разработки баз знаний для системы нечеткого вывода решений в заданной предметной области с использованием среды моделирования *MatLab*.

Краткие теоретические сведения

При разработке баз знаний интеллектуальных информационных систем, в том числе экспертной системы, в условиях неопределенности зачастую используется подход на основе нечетких знаний. В практике управления сложными динамическими объектами обычно не придерживаются границ области допустимых значений контролируемых параметров с точностью до второго – третьего знака после запятой, и многие ограничения чаще всего являются «мягкими», допускающими их незначительное нарушение. Границы области критических режимов также могут быть нечеткими. Применение нечетких («мягких») ограничений значительно расширяет возможность контроля и управления и делает их адекватными реальной обстановке в системе.

Реальные задачи содержат в себе нечеткие условия и некоторую нечеткость цели в связи с тем, что их постановку осуществляет человек. Учет фактора неопределенности при решении задач во многом изменяет методы принятия решения: меняется принцип представления исходных данных и параметров модели, становятся неоднозначными понятия решения задачи и оптимальности решения. Почти все реально работающие прикладные системы, использующие нечеткие знания – это либо системы, основанные на нечетких продукционных правилах, либо реляционные системы, использующие нечеткие отношения. Теоретически работа подобных систем основана на использовании композиционных правил нечетких выводов. Правила нечеткого управления, будучи условными высказываниями типа **«ЕСЛИ – ТО»**, представляют нечеткую импликацию:

ЕСЛИ x_1 есть A_1 И x_2 есть A_2 И …x_n есть A_n **ТО** y есть B.

Здесь A_i, B – это лингвистические значения, идентифицированные нечетким способом через соответствующие функции принадлежности $\mu A(x_i)$ и $\mu B(y)$ для переменных x и y. Переменные $x_1, x_2, \ldots, x_n$ образуют n- мерный входной вектор $x \in X$ динамического процесса, а y есть выходной сигнал. Значение функции принадлежности $\mu A(x)$, относящееся к уровню импликации (уровень активации правила), должно интерпретироваться с использованием нечетких операций. Нечетким расширением операции «И» в общей форме является T – норма. Нечетким расширением операции «ИЛИ» в общей форме является S – норма (иногда называемая T=конорма).

Приписывание единственного значения функции принадлежности, описывающей многомерное условие, будем называть агрегированием предпосылки.

Наиболее часто используемые типы нечетких операций «И» в двузначной нечеткой логике приведены в табл. 3.1, операции «ИЛИ» в табл. 3.2.

Таблица 3.1

Типы нечетких операций «И»

Интерпретация в форме логического произведения	$\mu_{A_3}(x) = \min(\mu_{A_1}(x), \mu_{A_2}(x))$
Интерпретация в форме алгебраического произведения	$\mu_{A_3}(x) = \mu_{A_1}(x) \cdot \mu_{A_2}(x)$

Таблица 3.2

Типы нечетких операций «ИЛИ»

Интерпретация в форме логической суммы	$\mu_{A_3}(x) = \max(\mu_{A_1}(x), \mu_{A_2}(x))$
Интерпретация в форме алгебраической суммы	$\mu_{A_3}(x) = \mu_{A_1}(x) + \mu_{A_2}(x) - \mu_{A_1}(x) \cdot \mu_{A_2}(x)$

Каждой импликации $A \to B$ можно приписать также единственное значение функции принадлежности $\mu A \to B(x,y)$. Приписывание единственного значения функции принадлежности всей предпосылки называется процедурой агрегирования на уровне импликации.

Использование правил осуществляется через механизм логического вывода. Логическое управление означает, что логика

управления эксперта представляется в виде нечетких правил и разнообразным предпосылкам сопоставляется некоторое действие. Формирование выходных переменных (управляющих воздействий) включает следующие этапы:

– получение вектора входных переменных $X=(X_1,\ldots, X_n)$;

– «фаззификация» этих переменных, то есть переход от четких значений переменных к их нечетким интерпретациям, то есть лингвистическим переменным;

– определение нечетких значений выходных переменных $U_1, \ldots, U_m$ (в виде функций принадлежности переменных $\mu(U)$, соответствующих нечетким подмножествам) на основе правил логического вывода;

– «дефаззификация», то есть переход от полученных функций принадлежности нечетким множествам к соответствующим единственным четким значениям выходных переменных.

Для проведения нечеткого моделирования существует довольно много инструментальных средств, например, *FuzzyTECH*, *CubiCalc*, *MatLab*.

3.1. Разработка системы нечеткого логического вывода в среде *MatLab.* Описание пакета *Fuzzy Logic Toolbox*

Система *MatLab* (сокращение от *MATrix LABoratory* – матричная лаборатория) представляет собой интерактивную компьютерную систему для выполнения инженерных и научных расчетов, ориентированную на работу с массивами данных. Нечеткое моделирование в среде *MatLab* осуществляется при помощи пакета расширения *Fuzzy Logic Toolbox*, в котором реализованы десятки функций нечеткой логики и нечеткого вывода. Пакет *Fuzzy Logic Toolbox* обладает простым и хорошо продуманным интерфейсом, позволяющим легко проектировать и диагностировать нечеткие модели, в нем обеспечивается поддержка современных методов нечеткой кластеризации и адаптивные нечеткие нейронные сети. Графические средства *Fuzzy Logic Toolbox* позволяют интерактивно отслеживать особенности поведения системы.

Базовым понятием *Fuzzy Logic Toolbox* является *FIS*-структура – система нечеткого вывода (*Fuzzy Inference System*). *FIS*-структура содержит все необходимые данные для реализации функционального

отображения «входы-выходы» на основе нечеткого логического вывода. Редактор *FIS* (*FIS Editor*) является основным средством, используемым при проектировании систем нечеткого логического вывода в интерактивном режиме. В состав пакета также входят редактор функций принадлежности (*Membership Function Editor*), редактор базы правил (*Rule Editor*), программа просмотра правил системы нечеткого вывода (*Rule Viewer*), программа просмотра поверхности системы нечеткого вывода (*Surface Viewer*).

Модуль *fuzzy* позволяет строить нечеткие системы двух типов – Мамдани и Сугэно. Основное отличие между системами Мамдани и Сугэно заключается в разных способах задания значений выходной переменной в правилах, образующих базу знаний. В системах типа Мамдани значения выходной переменной задаются нечеткими термами, в системах типа Сугэно – как линейная комбинация входных переменных. Преимущество моделей типа Мамдани состоит в том, что правила базы знаний являются прозрачными и интуитивно понятными, тогда как для моделей типа Сугэно не всегда ясно какие линейные зависимости «входы-выход» необходимо использовать.

3.2. Синтез системы нечеткого логического вывода в среде *MatLab*

Порядок выполнения работы по построению нечеткой базы знаний средствами *Fuzzy Logic Toolbox* представлен на примере принятия решений по выбору конфигурации системного блока персонального компьютера и включает следующие этапы.

Первый этап – запуск редактора *FIS*.

Редактор *FIS* может быть открыт с помощью ввода функции *fuzzy* в окне команд (рис. 3.1). Эта функция позволяет редактировать такие свойства системы нечеткого логического вывода (СНЛВ), как: число входных и выходных переменных, тип системы нечеткого вывода, метод дефаззификации и т.д.

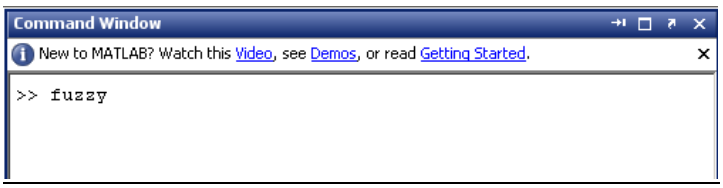

Рис. 3.1. Запуск *FIS*-редактора

В результате появляется интерактивное графическое окно (рис. 3.2).

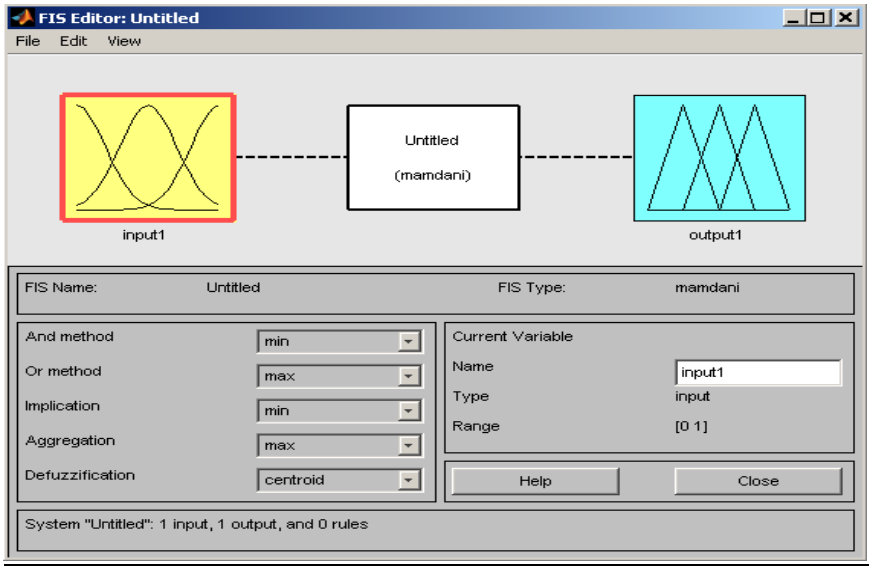

Рис. 3.2. Графический интерфейс редактора *FIS*

Главное окно *FIS*-редактора содержит восемь меню, а именно три общесистемных меню – *File*, *Edit*, *View*, и пять меню для выбора параметров нечеткого логического вывода – *And Method*, *Or Method*, *Implication*, *Aggregation* и *Defuzzification*.

Второй этап – задание типа СНЛВ, задание параметров модели, определение входных и выходных переменных.

Для решения поставленной задачи нечеткого моделирования использована СНЛВ типа Сугэно. Параметры модели: логические операции (*prod* – для нечеткого логического «И», *probor* – для нечеткого логического «ИЛИ»), метод импликации (min), метод агрегации (max) и метод дефаззификации (*wtaver*).

В рассматриваемом примере в редакторе *FIS* определяем пять входных переменных: *summa* (количество финансовых средств), *chastota_proc* (частота процессора), *OZU* (объем оперативного запоминающего устройства, ОЗУ), *HD* (объем жесткого диска), *video_card* (объем видеопамяти видеокарты) и одну выходную переменную *versiya* (конфигурация системного блока). Описание переменных приведено в табл. 3.3 и табл. 3.4.

Таблица 3.3

Описание входных переменных

Наименование	Диапазон изменения	Термы	Тип функций принадлежности
summa	0..30	*low* (низкая) *average* (средняя) *high* (высокая)	трапеция
chastota_proc	0..5	*low* (низкая) *average* (средняя) *high* (высокая)	трапеция
OZU	0..8	*low* (низкая) *below average* (ниже среднего) *average* (средняя) *above average* (выше среднего) *high* (высокая)	гауссова
HD	0..6000	*low* (низкая) *average* (средняя) *high* (высокая)	трапеция
video_card	0..4	*low* (низкая) *average* (средняя) *high* (высокая)	треугольная

Таблица 3.4

Описание выходных переменных

Наименование	Диапазон изменения	Термы	Значение
versiya	0..1	*budget* (бюджетная) *standard* (стандартная) *game* (игровая) *prof* (профессиональная)	0 0.3 0.7 1

Графический интерфейс редактора *FIS* после определения входных и выходных переменных, а также задания параметров СНЛВ представлен на рис. 3.3.

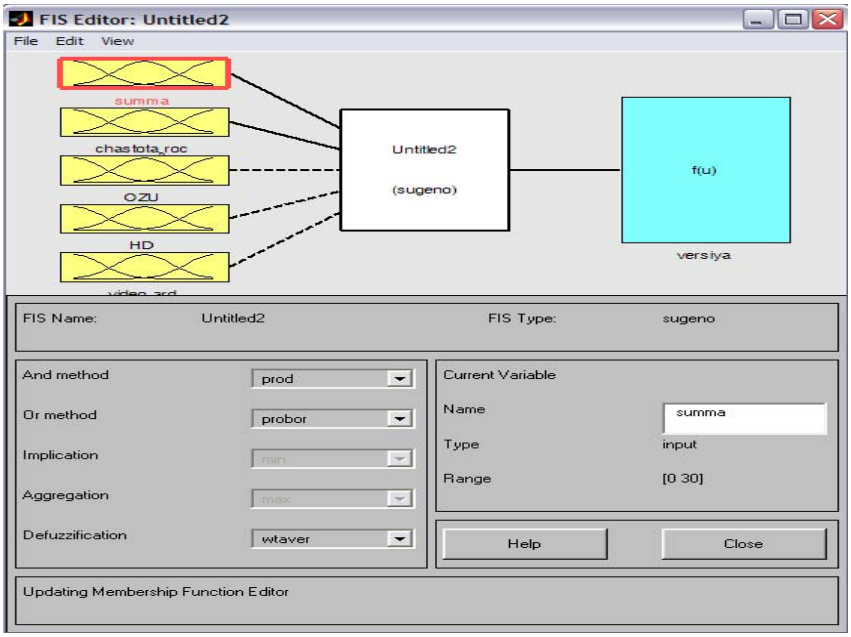

Рис. 3.3. Графический интерфейс редактора *FIS* после определения входных и выходных переменных

Третий этап – определение функций принадлежности термов для каждой входной и выходной переменных.

Для рассматриваемой СНЛВ при помощи редактора функций принадлежности для каждой из пяти входных и одной выходной переменных определяем функции принадлежности термов. Редактор может быть открыт при помощи команды *Edit ... Membership Function*. Функции принадлежности добавляются в рабочую область при помощи команды *Edit ... Add MFs*. В интерактивном графическом окне необходимо задать количество термов (*number of mfs*) и тип функций принадлежности (*mf type*). Кроме того, в редакторе необходимо указать диапазон изменения переменной (свойство *Range*), имя переменной (свойство *Name*), параметры функции принадлежности (свойство *Params*).

Графический интерфейс редактора функций принадлежности на примере входной переменной «Объем оперативного запоминающего устройства» изображен на рис. 3.4.

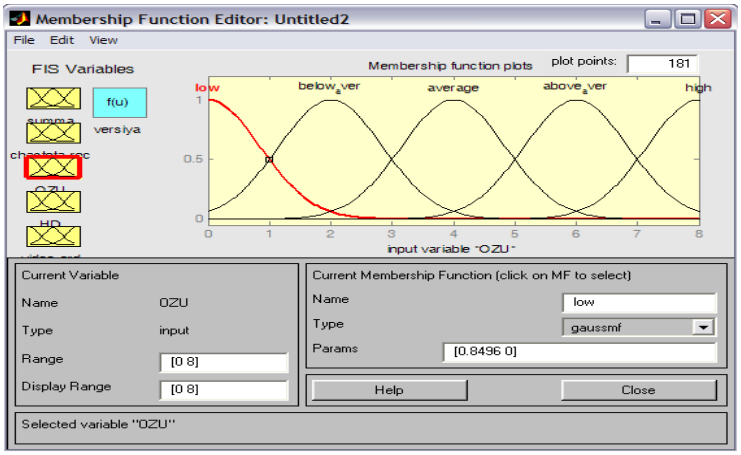

Рис. 3.4. Графический интерфейс редактора функций принадлежности для входной переменной «*OZU*

Четвертый этап – задание правил для разрабатываемой СНЛВ.

Зададим правила для разрабатываемой СНЛВ (рис. 3.5). Команда *Edit ... Rules…* открывает редактор базы знаний. Поскольку в рабочем окне отображаются не все переменные нечеткой модели, для управления режимом отображения переменных правил следует воспользоваться специальными кнопками «>>» и «<<», расположенными в нижней правой части редактора правил.

Редактор правил позволяет добавлять новое правило (*Add rule*), удалять правила (*Delete rule*) и производить редактирование уже существующих правил (*Change rule*). При формировании базы правил необходимо учитывать коэффициент уверенности каждого правила, который изменяется от 0 до 1 (свойство *Weight*).

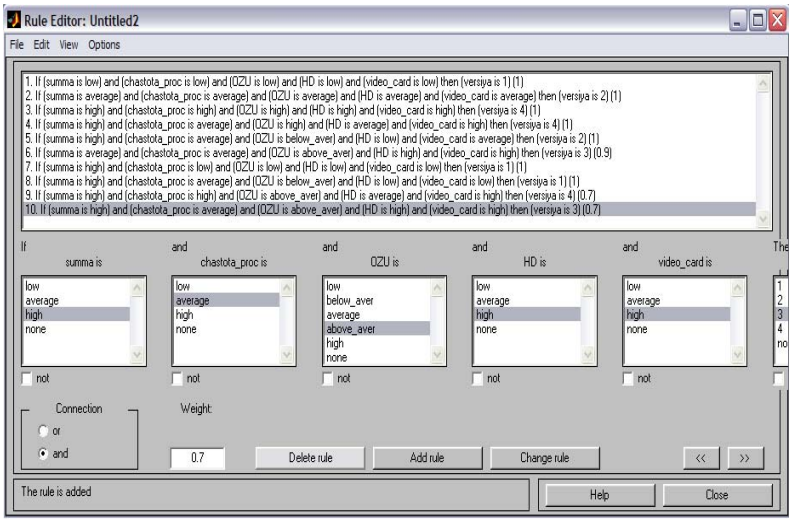

Рис. 3.5. Графический интерфейс редактора правил после задания базы правил системы нечеткого вывода

Пятый этап – анализ построенной СНЛВ.

Выполним анализ построенной СНЛВ для рассматриваемой задачи выбора конфигурации системного блока. Анализ системы выполняется при помощи программы просмотра правил, загрузка которой выполняется командой *View ... Rules*. Программа предоставляет возможность получать значения выходных переменных в зависимости от значений входных переменных. Ввести значения входных переменных можно с помощью их записи в поле *Input*. Полученное значение выходной переменной при этом отображается непосредственно после ее имени. Так, для частного случая, когда значение входной переменной количество финансовых средств равно 15; частота процессора – 2,5; объем оперативного запоминающего устройства – 4; объем жесткого диска – 3000; объем видеопамяти видеокарты – 2. Процедура нечеткого вывода, выполненная системой *MatLab* для разработанной нечеткой модели, выдает в результате значение выходной переменной «Конфигурация системного блока», равное 0,3, что соответствует стандартной конфигурации (рис. 3.6).

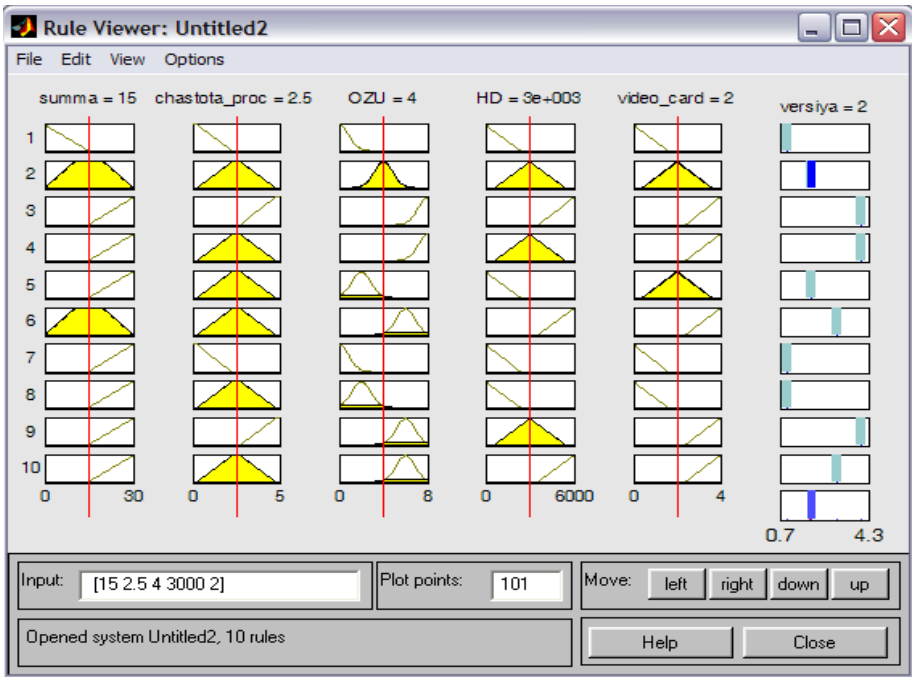

Рис. 3.6. Графический интерфейс программы просмотра правил после выполнения процедуры нечеткого вывода

Шестой этап – визуализация поверхности нечеткого вывода СНЛВ.

Загрузка программы визуализация поверхности нечеткого вывода выполняется командой *View ... Surface*. Полученная поверхность позволяет проанализировать зависимость значений выходной переменной от отдельных входных переменных. Комбинации входных переменных задаются в соответствие с их размещением на осях системы координат (рис. 3.7).

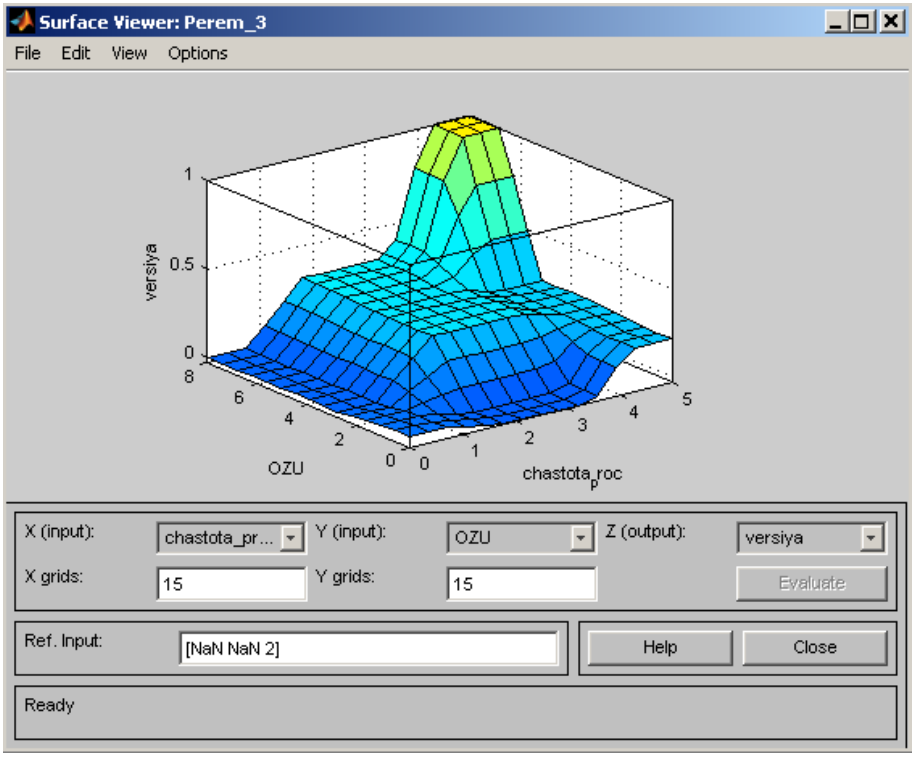

Рис. 3.7. Визуализация поверхности нечеткого вывода рассматриваемой модели для входных переменных «частота процессора» и «ОЗУ»

Проанализировав полученную поверхность нечеткого вывода, можно сделать вывод о том, что она отвечает экспертным представлениям в рассматриваемой предметной области. Так, например, можно сказать о том, что с увеличением объема оперативного запоминающего устройства и одновременным увеличением частоты процессора предлагаемая версия персонального компьютера усложняется. При максимальных объемах ОЗУ и частоты процессора система рекомендует выбрать профессиональный вариант персонального компьютера.

Задание на лабораторную работу

1. Изучить методику разработки нечетких баз знаний экспертных систем с использованием среды визуального моделирования *MatLab*.

2. Загрузить нечеткий редактор *FIS MatLab*.

3. Провести анализ данных предметной области, указанной преподавателем или выбранной самостоятельно:

– определить входные и выходную переменные;

– определить тип системы нечеткого вывода;

– определить параметры модели: логические операции, метод импликации, метод агрегации и метод дефаззификации;

– задать функции принадлежности для каждой из рассматриваемых переменных и интервал их изменения;

– задать правила для разрабатываемой СНЛВ;

– выполнить анализ разработанной системы нечеткого вывода.

4. Построить нечеткую базу знаний, работа которой отражала бы процесс принятия решений экспертом.

5. Оформить отчет, который должен содержать: цель работы, постановку задачи, описание входных и выходной переменных в виде таблицы (наименование переменной, диапазон изменения значений, наименование термов, вид функции принадлежности), функции принадлежности для переменных, правила в виде таблицы (номер правила, часть «ЕСЛИ», часть «ТО»), визуализацию нечеткого логического вывода для набора значений входных переменных, поверхности нечеткого логического вывода для разной комбинации входных переменных.

Требования к отчету

Отчет должен содержать:

1. Титульный лист.
2. Задание.
3. Цель работы.
4. Исходные данные для выполнения работы, в соответствии с заданным вариантом.
5. Основные результаты выполнения работы.
6. Выводы по работе.

Контрольные вопросы и задания

1. В чем преимущества подхода, основанного на нечетких знаниях?
2. Каковы функции нечеткого редактора *FIS*?
3. Что такое функция принадлежности? Какие существуют виды функций принадлежности?
4. Какие типы СНЛВ поддерживает *MatLab*?
5. Какое преимущество предоставляет в инженерии знаний теория нечетких множеств?
6. По какой формуле вычисляется в нечеткой логике функция принадлежности конъюнкции двух нечетких переменных?
7. Как определяется нечеткое множество C в множестве элементов X?
8. По какой формуле вычисляется в нечеткой логике функция принадлежности дизъюнкции двух нечетких переменных?

ЛАБОРАТОРНАЯ РАБОТА № 4
НЕЙРО-НЕЧЕТКОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ *MatLab*

Цель работы

Целью работы является изучение и усвоение методов моделирования и принципов функционирования нейро-нечетких сетей (ННС), в том числе при решении задач экономического прогнозирования, а также приобретение навыков по конструированию ННС в среде *MatLab*.

Краткие теоретические сведения

Основная идея, положенная в основу модели гибридных сетей, заключается в том, чтобы использовать существующую выборку данных для определения параметров функций принадлежности, которые лучше всего соответствуют некоторой системе нечеткого вывода. При этом для нахождения параметров функции принадлежности используются известные процедуры обучения нейронных сетей.

В пакете *Fuzzy Logic Toolbox* системы *MatLab* гибридные сети реализованы в форме адаптивной системы нейро-нечеткого вывода *ANFIS*. С одной стороны, гибридная сеть *ANFIS* представляет собой нейронную сеть с единственным выходом и несколькими входами, которые представляют собой нечеткие лингвистические переменные. При этом термы входных лингвистических переменных описываются стандартными для системы *MatLab* функциями принадлежности, а термы выходной переменной представляются линейной или постоянной функцией принадлежности. С другой стороны, гибридная сеть *ANFIS* представляет собой систему нечеткого вывода *FIS* типа Сугэно нулевого или первого порядка, в которой каждое из правил нечетких продукций имеет постоянный вес, равный единице.

Для прогнозирования используем нечеткую сеть *TSK*. Обобщенную схему вывода модели *TSK* при использовании *M* правил и *N* переменных x_j можно представить в виде:

$$IF\ \left(x_1.IS.A_1^{(1)}\right).AND.\left(x_2.IS.A_2^{(1)}\right).AND.\ldots.AND.\left(x_n.IS.A_n^{(1)}\right),$$

$$THEN\ y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} \cdot x_j$$

$$\dots$$

$$IF\ \left(x_1.IS.A_1^{(M)}\right).AND.\left(x_2.IS.A_2^{(M)}\right).AND.\dots.AND.\left(x_n.IS.A_n^{(M)}\right),$$

$$THEN\ y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj} \cdot x_j$$

Условие *IF* (x_i.*IS*.A_i) реализуется функцией фаззификации, которая представляется обобщенной функцией Гаусса отдельно для каждой переменной x_i:

$$\mu_A(x_i) = \frac{1}{1+\left(\frac{x_i - c_i}{\sigma_i}\right)^{2 \cdot b_i}}, \tag{4.1}$$

где $\mu_A(x_i)$представляет оператор A_i.

В нечетких сетях целесообразно задавать это условие в форме алгебраического произведения, из которого следует, что для *k*-го правила вывода:

$$\mu_A^{(k)}(x) = \prod_{j=1}^{N} \left[\frac{1}{1+\left(\frac{x_i - c_j^{(k)}}{\sigma_j^{(k)}}\right)^{2 \cdot b_j^{(k)}}} \right], \tag{4.2}$$

При *M* правилах вывода агрегирование выходного результата сети производится по формуле:

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{N} w_j} \cdot \left(p_{i0} + \sum_{j=1}^{N} p_{ij} \cdot x_j \right), \tag{4.3}$$

которую можно представить в виде:

$$y(x) = \frac{1}{\sum_{k=1}^{N} w_k} \cdot \sum_{k=1}^{M} w_k \cdot y_k(x), \qquad (4.4)$$

где $y_k(x) = p_{k0} + \sum_{j=1}^{N} p_{kj} \cdot x_j$.

Присутствующие в этом выражении веса w_k интерпретируются как значимость компонентов $\mu_A^{(k)}(x)$, определенных формулой (4.1). При этом условии формуле (4.4) можно сопоставить многослойную структуру сети (рис. 4.1).

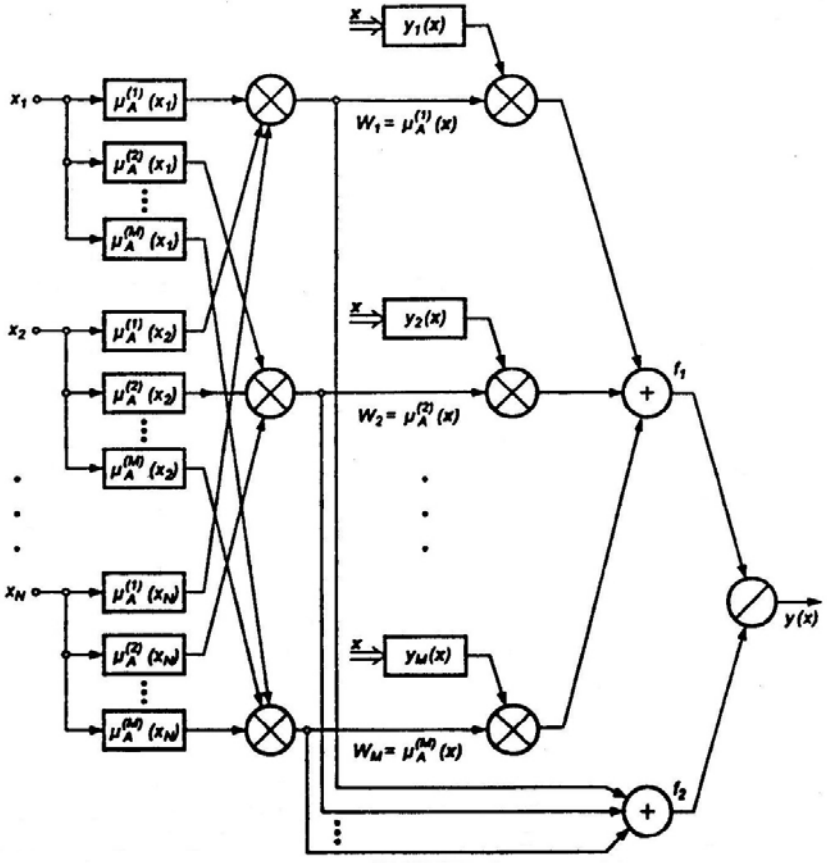

Рис. 4.1. Структура нечеткой нейронной сети *TSK*

В такой сети выделяется пять слоев.

Первый слой выполняет раздельную фаззификацию каждой переменной x_i (i=1, 2, …, N), определяя для каждого k-го правила вывода значение коэффициента принадлежности $\mu_A^{(k)}(x_i)$ в соответствии с применяемой функцией фаззификации. Это параметрический слой с параметрами $c_j(k), \sigma_j^{(k)}, b_j(k)$, подлежащими адаптации в процессе обучения.

Второй слой выполняет агрегирование отдельных переменных x_i, определяя результирующее значение коэффициента принадлежности $w_k = \mu_A^{(k)}(x)$ для вектора x (уровень активизации правила вывода) в соответствии с формулой (4.2). Это слой непараметрический.

Третий слой представляет собой генератор функции *TSK*, рассчитывающий значения:

$$y_k(x) = p_{k0} + \sum_{j=1}^{N} p_{kj} \cdot x_j$$

В этом слое также производится умножение сигналов $y_k(x)$ на значения w_k, сформированные на предыдущем слое. Это параметрический слой, в котором адаптации подлежат линейные веса p_{kj} для $k = 1, 2, \ldots, M$ и $j = 1, 2, \ldots, N$, определяющие функцию следствия модели *TSK*.

Четвертый слой составляют два нейрона-сумматора, один из которых рассчитывает взвешенную сумму сигналов $y_k(x)$, а второй определяет сумму весов $\sum_{k=1}^{M} w_k$. Это непараметрический слой.

Последний, **пятый слой**, состоящий из единственного выходного нейрона, – это нормализующий слой, в котором веса подвергаются нормализации в соответствии с формулой (4.4). Выходной сигнал $y(x)$ определяется выражением, соответствующим зависимости (4.3):

$$y(x) = f(x) = \frac{f_1}{f_2} \qquad (4.5)$$

Из приведенного описания следует, что нечеткая сеть *TSK* содержит только два параметрических слоя (первый и третий), параметры которых уточняются в процессе обучения. Параметры первого слоя будем называть нелинейными параметрами, поскольку они относятся к нелинейной функции (4.1), а параметры третьего слоя – линейными весами, т.к. они относятся к параметрам p_{kj} линейной функции *TSK*.

При уточнении функциональной зависимости (4.4) для сети *TSK* получаем:

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x)\right]} \cdot \sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right] \cdot \left[p_{k0} + \sum_{j=1}^{N} p_{kj}x_j\right]. \quad (4.6)$$

Если принять, что в конкретный момент времени параметры условия зафиксированы, то функция $y(x)$ является линейной относительно переменных x_i (i=1, 2, …, N).

При наличии N входных переменных каждое правило формирует N+1 переменных $p_j(k)$ линейной зависимости *TSK*. При M правилах вывода это дает $M\cdot(N+1)$ линейных параметров сети. В свою очередь, каждая функция принадлежности использует три параметра (c, σ, b), подлежащих адаптации. Если принять, что каждая переменная x_i характеризуется собственной функцией принадлежности, то при M правилах вывода мы получим $3\cdot M\cdot N$ нелинейных параметров. В сумме это дает $M\cdot(4\cdot N+1)$ линейных и нелинейных параметров, значения которых должны подбираться в процессе обучения сети.

На практике для уменьшения количества адаптируемых параметров оперируют меньшим количеством независимых функций принадлежности для отдельных переменных, руководствуясь правилами, в которых комбинируются типовые функции принадлежности различных переменных. Если принять, что каждая переменная x_i имеет m различных функций принадлежности, то максимальное количество правил, которые можно создать при их комбинировании, составит: $M = m^N$ (при трех функциях принадлежности, распространяющихся на две переменные, это $3^2 = 9$ правил вывода).

В пакете *Fuzzy Logic Toolbox* системы *MatLab* гибридные сети реализованы в форме адаптивных систем нейро-нечеткого вывода *ANFIS*.

Редактор *ANFIS* позволяет создавать или загружать конкретную модель адаптивной ННС, выполнять ее обучение, визуализировать ее структуру, изменять и настраивать ее параметры, а также использовать настроенную сеть для получения результатов нечеткого вывода.

4.1. Разработка нейро-нечеткой модели в среде *MatLab.* Описание *ANFIS*-редактора

ANFIS является аббревиатурой *Adaptive Neuro-Fuzzy Inference System* – (адаптивная нейро-нечеткая система). *ANFIS*-редактор позволяет автоматически синтезировать из экспериментальных данных ННС. ННС можно рассматривать как одну из разновидностей систем нечеткого логического вывода типа Сугэно. При этом функции принадлежности синтезированных систем настроены (обучены) так, чтобы минимизировать отклонения между результатами нечеткого моделирования и экспериментальными данными. Загрузка *ANFIS*-редактора осуществляется по команде *anfisedit*.

4.2. Синтез нейро-нечеткой сети в среде *MatLab*

Синтезируем ННС, реализующую принятие решений по выбору конфигурации системного блока. Выбор конфигурации осуществляется по трем критериям: частота работы процессора, объем оперативного запоминающего устройства, объем памяти видеокарты. На рис. 4.2 представлена структура рассматриваемой системы нечеткого вывода.

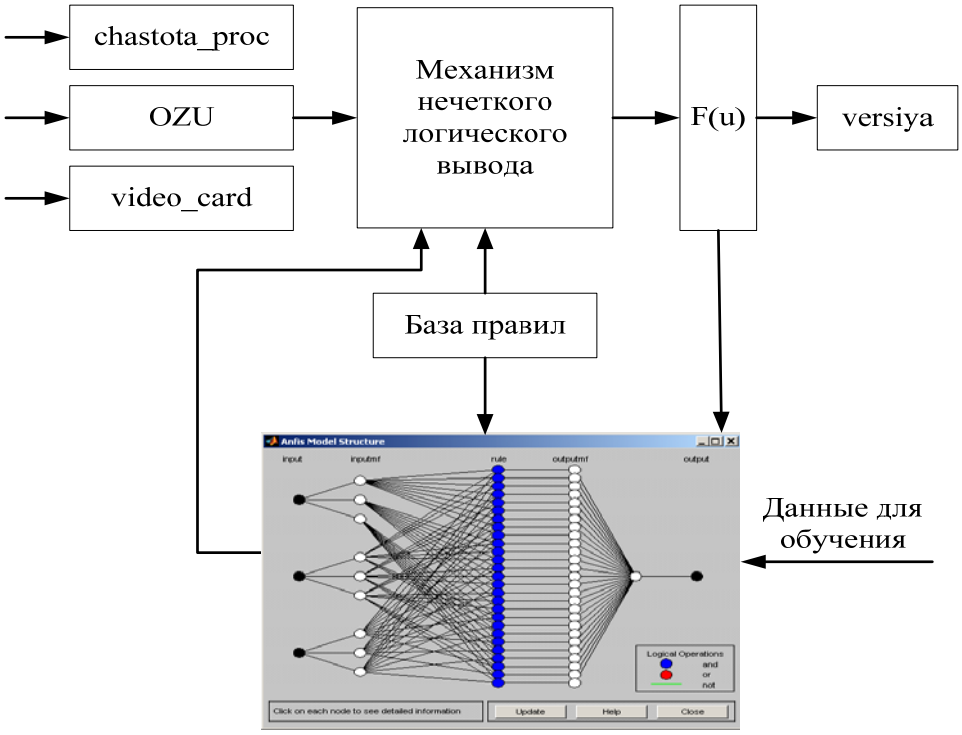

Рис. 4.2. Структура системы нечеткого вывода

Общая последовательность процесса разработки модели нейро-нечеткой сети заключается в следующем.

1. Подготовка файла с обучающими данными. Целесообразно воспользоваться текстовым редактором Блокнот или редактором электронных таблиц *Excel*. Обучающую выборку необходимо сохранить во внешнем файле с расширением **.dat*.

Фрагмент обучающей выборки для рассматриваемой предметной области представлен в табл. 4.1.

Таблица 4.1

Набор данных для обучения ННС

Первая входная переменная	Вторая входная переменная	Третья входная переменная	Выходная переменная
1	*2*	*3*	*4*
0.8	0.6	0.5	0.03
2.4	3.	1.8	0.27
4.4	5.6	3.6	0.87
1.	1.	0.5	0.10
4.3	5.4	3.7	0.85
1.8	1.9	0.7	0.20
3.8	5.2	2.6	0.40
3.	4.0	2.	0.30
3.9	5.8	3.9	0.65
3.6	5.1	3.5	0.55
4.6	7.5	3.8	0.92
3.7	5.4	3.7	0.60
0.667	0.5	0.3	0.01
0.8	0.4	0.3	0.05
4.0	6.0	4.0	0.70
1.1	1.2	0.8	0.15
3.4	3.2	4.0	0.75
3.2	4.5	2.4	0.35
4.5	6.4	3.7	0.90
4.9	7.8	3.9	0.98
5.0	8.0	4.0	1.

2. Открыть редактор *ANFIS*. Загрузить файл с обучающими данными.

Кнопка загрузки данных *Load Data*, по нажатию которой появляется диалоговое окно выбора файла, если загрузка данных

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

Внешний вид редактора *ANFIS* с загруженными обучающими данными изображен на рис. 4.3.

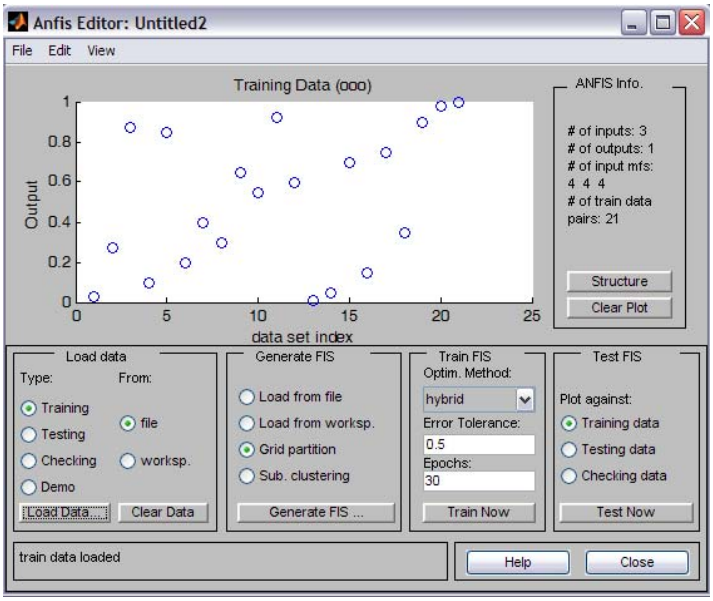

Рис. 4.3. Графический интерфейс редактора ANFIS
после загрузки обучающих данных

3. После подготовки и загрузки обучающих данных можно сгенерировать структуру системы нечеткого вывода *FIS* типа Сугэно, которая является моделью гибридной сети в системе *MatLab*. Для этой цели следует воспользоваться кнопкой *Generate FIS* в нижней части рабочего окна редактора. При этом две первые опции относятся к предварительно созданной структуре гибридной сети, а две последних – к форме разбиения входных переменных модели. Система нечеткого логического вывода сгенерирована по методу решетки.

Перед генерацией структуры системы нечеткого вывода типа Сугэно после вызова диалогового окна свойств зададим для каждой из входных переменных по три лингвистических терма, а в качестве типа их функций принадлежности выберем треугольные функции.

После нажатия кнопки *Generate FIS* вызывается диалоговое окно с указанием числа и типа функции принадлежности для отдельных термов входных переменных, и выходной переменной (рис. 4.4).

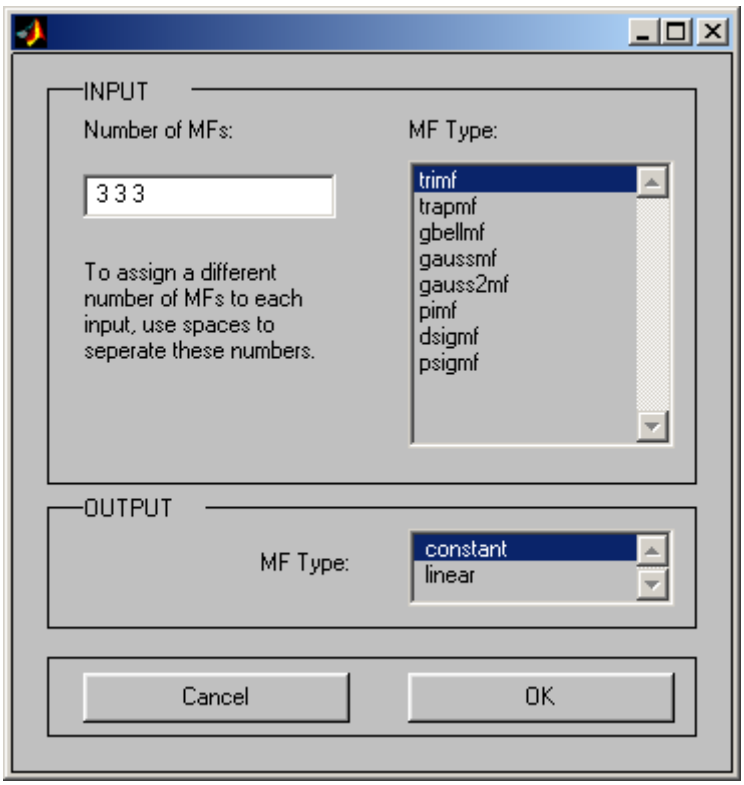

Рис. 4.4. Диалоговое окно для задания количества и типа функций принадлежности

4. После генерации структуры гибридной сети можно визуализировать ее структуру, для чего следует нажать кнопку *Structure* в правой части графического окна (рис. 4.5).

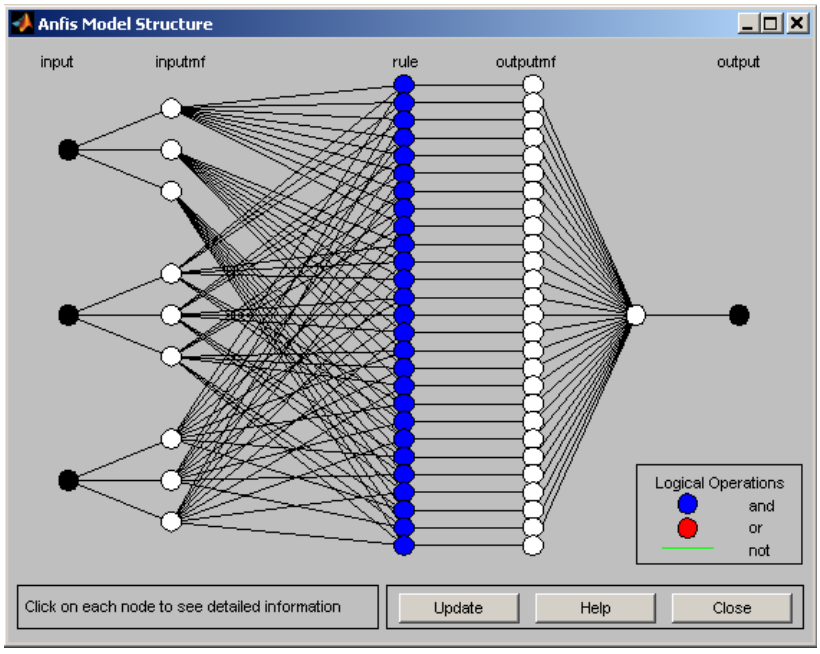

Рис. 4.5. Структура сгенерированной нейро-нечеткой сети

Для рассматриваемого примера система нечеткого вывода содержит три входных переменных с тремя термами каждая, 27 правил нечетких продукций, одну выходную переменную с 27 термами.

5. Перед обучением гибридной сети необходимо задать параметры обучения, для чего следует воспользоваться следующей группой опций в правой нижней части рабочего окна:

Выбрать метод обучения гибридной сети – обратного распространения (*backpropo*) или гибридный (*hybrid*), представляющий собой комбинацию метода наименьших квадратов и метода убывания обратного градиента.

Установить уровень ошибки обучения (*Error Tolerance*) – по умолчанию значение «0» (изменять не рекомендуется).

Задать количество циклов обучения (*Epochs*) – по умолчанию значение «3» (рекомендуется увеличить для рассматриваемого примера: задать его значение равным 30).

Для обучения сети следует нажать кнопку *Train now*. При этом ход процесса обучения иллюстрируется в окне визуализации в форме графика зависимости ошибки от количества циклов обучения (рис. 4.6).

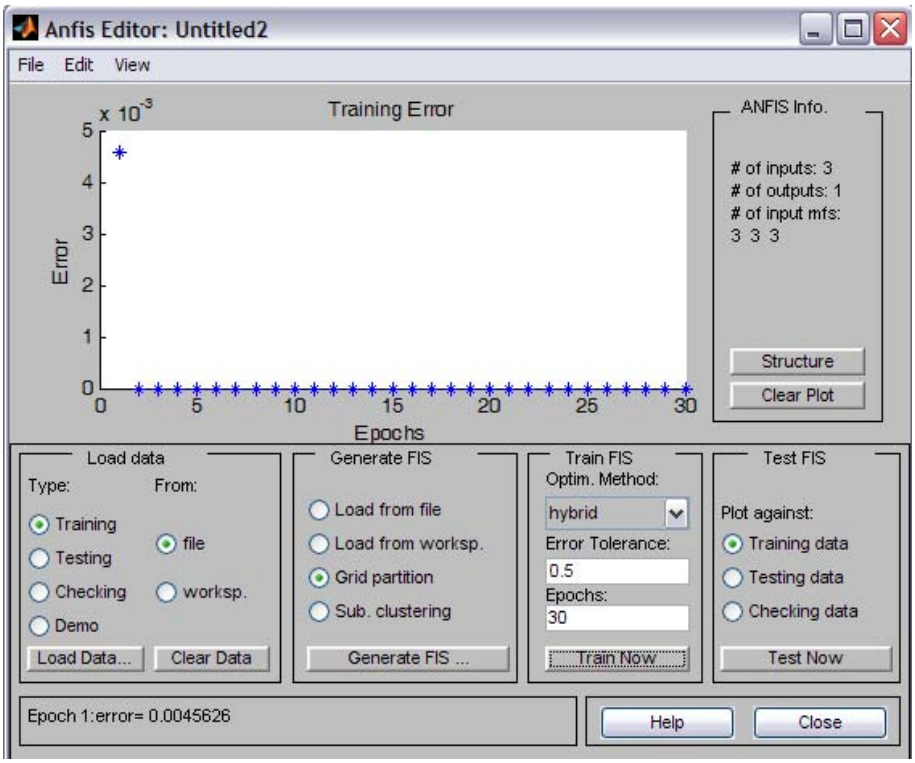

Рис. 4.6. График зависимости ошибок обучения от количества циклов обучения

Дальнейшая настройка параметров построенной и обученной гибридной сети может быть выполнена с помощью стандартных графических средств пакета *Fuzzy Logic Toolbox*. Для этого рекомендуется сохранить созданную систему нечеткого вывода во внешнем файле с расширением **.fis*, после чего следует загрузить этот файл в редактор систем нечеткого вывода *FIS*. Система нечеткого логического вывода представлена на рис. 4.7–4.11.

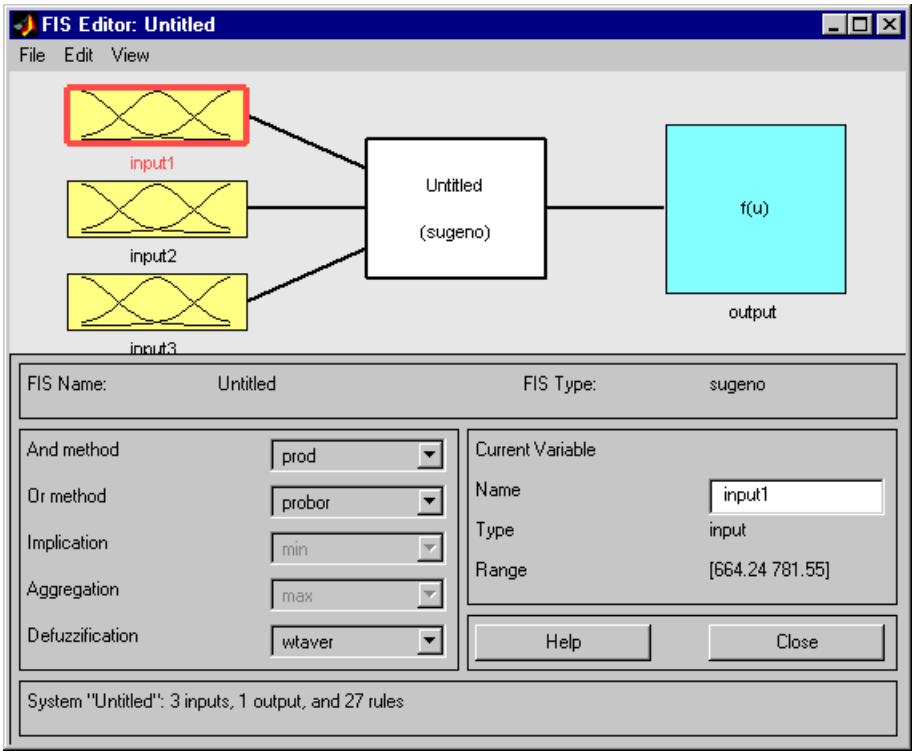

Рис. 4.7. Графический интерфейс редактора *FIS* для сгенерированной системы нечеткого вывода

6. Выполним проверку эффективности построенной нечеткой модели гибридной сети. Для этого можно определить конфигурацию системного блока. Для решения этой задачи необходимо воспользоваться функцией *evalfis*.

fis = readfis('Untitled2')

PC = evalfis([2.5 4 2], fis)

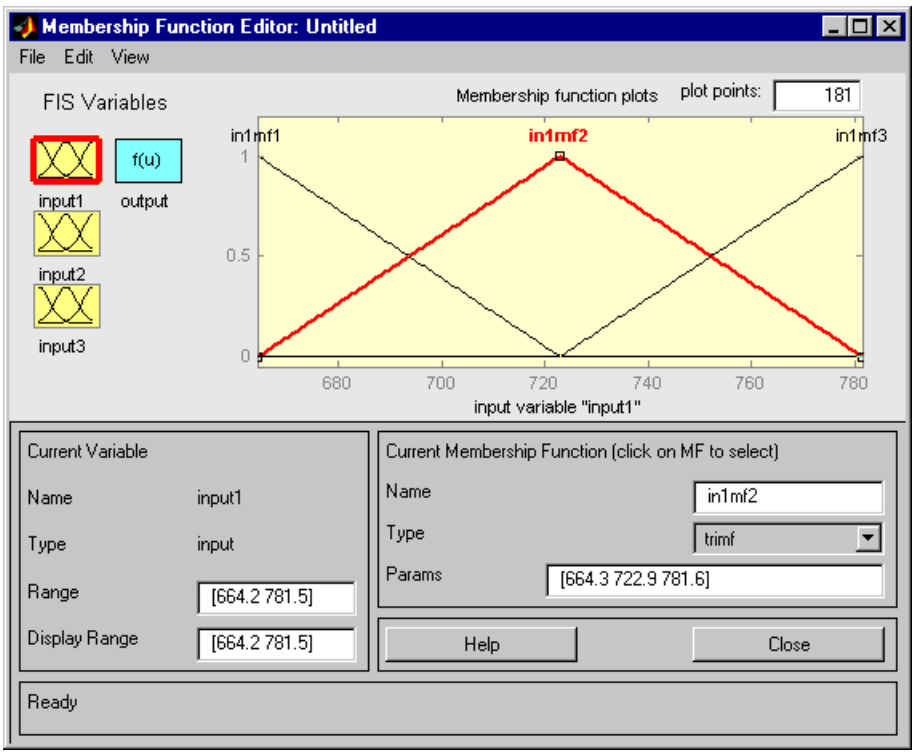

Рис. 4.8. Графический интерфейс редактора функций принадлежности построенной системы нечеткого вывода

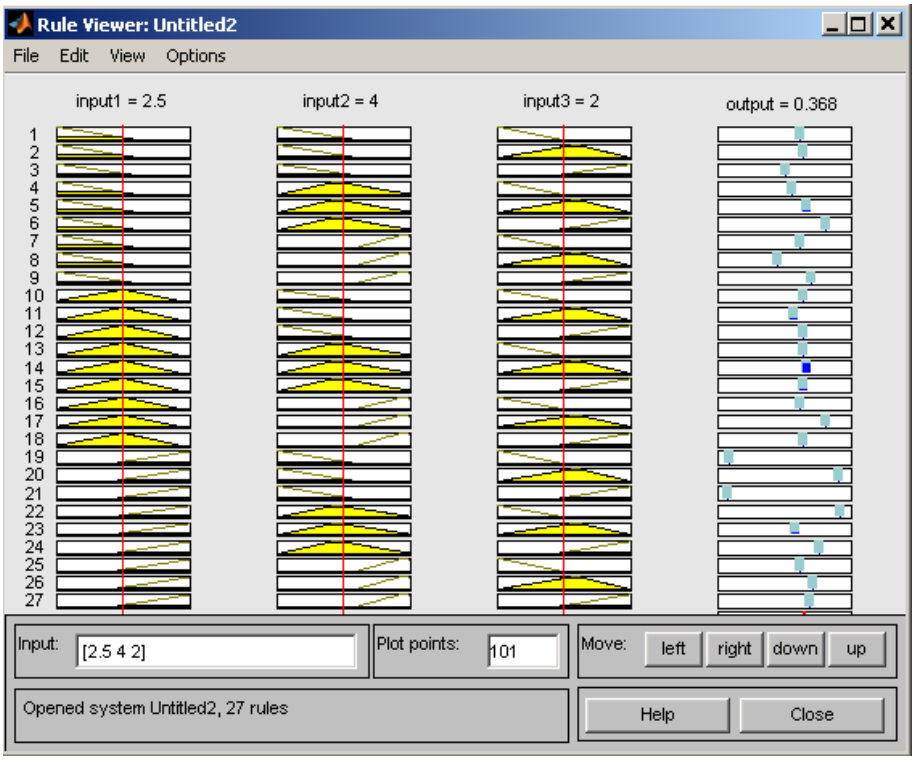

Рис. 4.9. Графический интерфейс просмотра правил сгенерированной системы нечеткого вывода

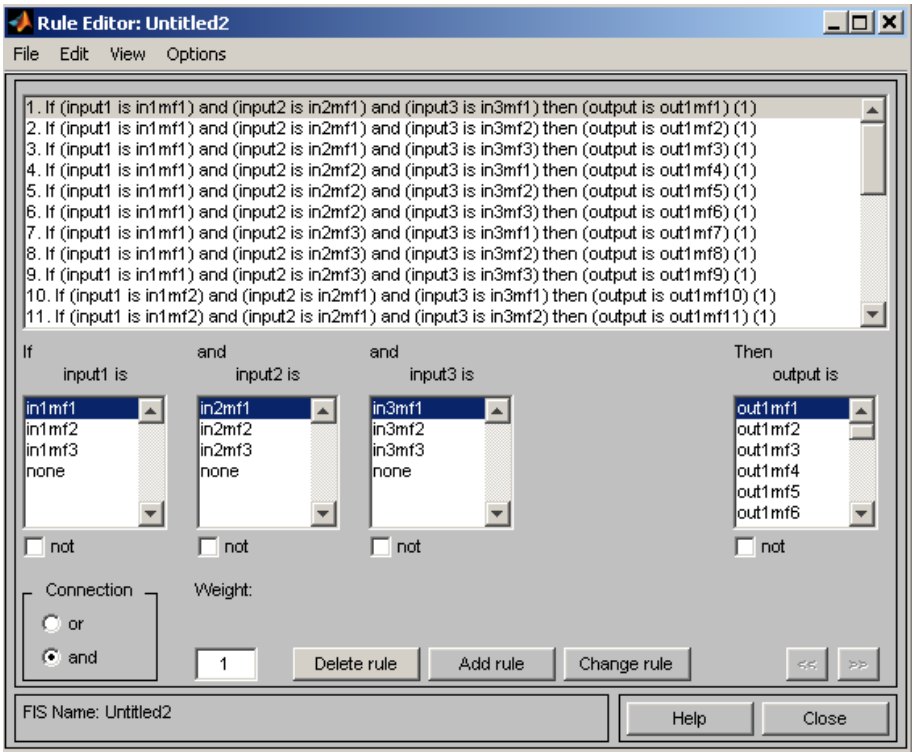

Рис. 4.10. Фрагмент базы нечетких правил

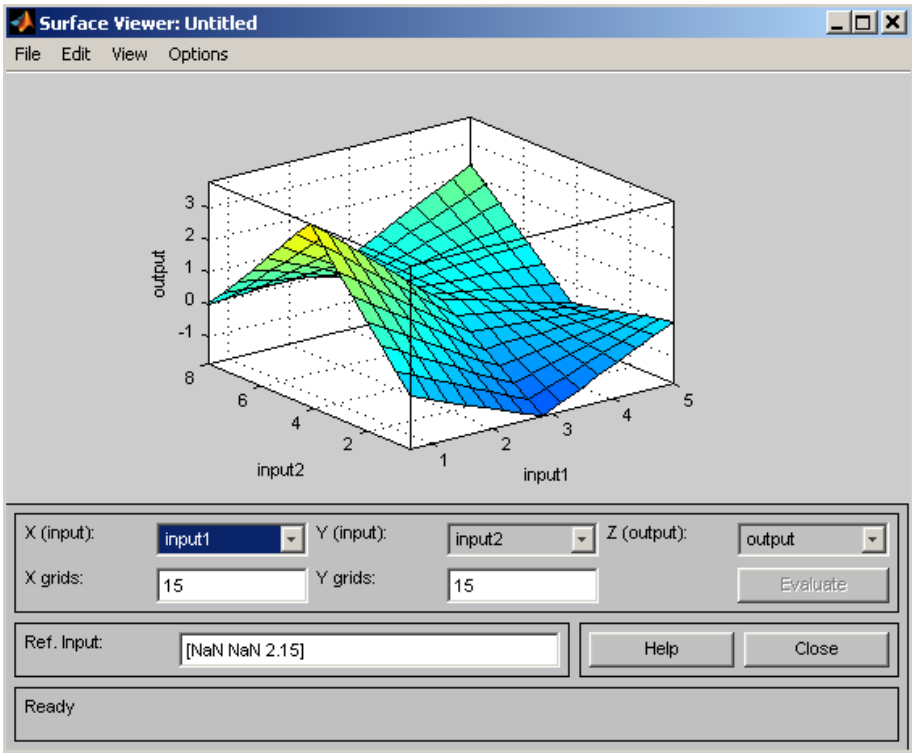

Рис. 4.11. Визуализация поверхности нечеткого вывода рассматриваемой модели для первой и второй входных переменных

Задание на лабораторную работу

1. Подготовить файл с обучающими данными с расширением **.dat*.

2. Загрузить файл с обучающими данными в редактор *ANFIS*.

3. Сгенерировать структуру системы нечеткого вывода *FI*S типа Сугэно, используя методы решетки и субкластеризации.

4. Произвести обучение ННС, предварительно задав параметры обучения.

5. Проверить эффективность построенной нейро-нечеткой модели гибридной сети.

6. Оформить отчет, который должен содержать: обучающую выборку, редактор *ANFIS* с загруженными обучающими данными, структуру нейро-нечеткой сети, функции принадлежности для входных и выходной переменных, график зависимости ошибок обучения от количества циклов, структуру ННС после обучения, поверхности нечеткого логического вывода для разного набора входных переменных, результаты проверки эффективности, построенной ННС.

Требования к отчету

Отчет должен содержать:

1. Титульный лист.
2. Задание.
3. Цель работы.
4. Исходные данные для выполнения работы, в соответствии с заданным вариантом.
5. Основные результаты выполнения работы.
6. Выводы по работе.

Контрольные вопросы и задания

1. Дайте определение ННС.
2. Каково предназначение сетей нейро-нечеткого вывода?
3. В чем преимущества использования ННС?
4. Охарактеризуйте структуру ННС.
5. Опишите процесс разработки ННС в среде *MatLab*.
6. Как проверить эффективность построенной ННС?
7. Какие возможности по визуализации результатов моделирования предоставляет система *MatLab*?