МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«25» 03 2021 г.

НАУЧНЫЕ ИССЛЕДОВАНИЯ В ОБЛАСТИ БИОТЕХНИЧЕСКИХ СИСТЕМ И ТЕХНОЛОГИЙ

Методические указания к самостоятельной работе по дисциплине «Методы и средства исследований в области биотехнических систем и технологий»

Курск 2021

УДК 004.93:61

Составитель: О.В. Шаталова.

Рецензент
Кандидат технических наук, доцент Т.Н. Конаныхина

Научные исследования в области биотехнических систем и технологий: методические указания к самостоятельной работе / Юго-Зап. гос. ун-т; сост.: О.В. Шаталова, К.Д.А. Кассим. Курск, 2021. 52 с.

Предназначено для студентов направления подготовки 12.04.04 «Биотехнические системы и технологии» по дисциплине «Методы и средства исследований в области биотехнических систем и технологий». Может быть использована аспирантами, обучающимися по направленностям 05.11.13 – Системный анализ, управление и обработка информации и 05.11.17 – Приборы, системы и изделия медицинского назначения.

Текст печатается в авторской редакции

Подписано в печать 25.03.21 . Формат 60×84 1/16. Бумага офсетная.
Усл. печ. л. 3,02. Уч.-изд. л. 2,74. Тираж 100 экз. Заказ 521 .
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

1 Реализация нейронных сетей в пакете Matlab. Графический интерфейс Toolbox NNTool

Теоретические сведения

NNTool – графический интерфейс для работы с нейронными сетями пакета Matlab позволяет выбирать структуры НС из обширного перечня и предоставляет множество алгоритмов обучения для каждого типа сети.

В лабораторной работе рассмотрены следующие вопросы, относящиеся к работе с NNTool:

- назначение графических управляющих элементов;
- подготовка данных;
- создание нейронной сети;
- обучение сети;
- прогон сети.

Методика и порядок выполнения работы

Чтобы запустить NNTool, необходимо выполнить одноимённую команду в командном окне MATLAB:

```
>> nntool
```

После этого появится главное окно NNTool, именуемое "Окном управления сетями и данными" (Network/Data Manager) (рисунок 1.1).

Панель "Сети и данные" (Networks and Data) имеет функциональные клавиши со следующими назначениями:

- Помощь (Help) − краткое описание управляющих элементов данного окна;
- Новые данные (New Data…) − вызов окна, позволяющего создавать новые наборы данных;
- Новая сеть (New Network…) − вызов окна создания новой сети;
- Импорт (Import…) − импорт данных из рабочего пространства MATLAB в пространство переменных NNTool;
- Экспорт (Export…) − экспорт данных из пространства переменных NNTool в рабочее пространство MATLAB;

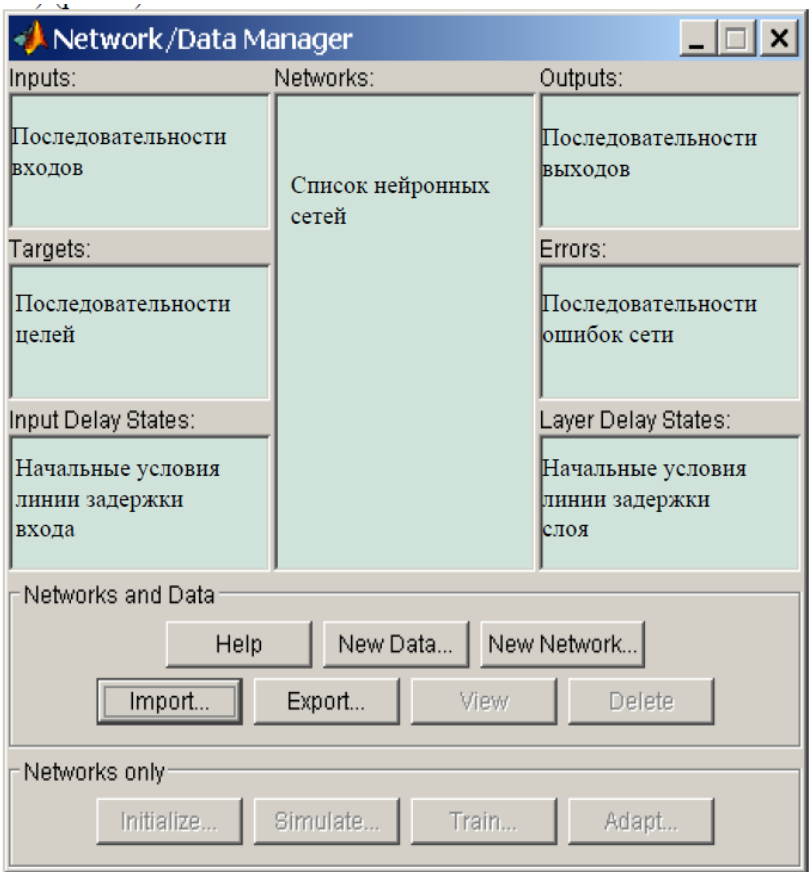

Рисунок 1.1 – Главное окно NNTool

- Вид (View) − графическое отображение архитектуры выбранной сети;
- Удалить (Delete) − удаление выбранного объекта.

На панели "Только сети" (Networks only) расположены клавиши для работы исключительно с сетями. При выборе указателем мыши объекта любого другого типа, эти кнопки становятся неактивными.

При работе с NNTool важно помнить, что клавиши View, Delete, Initialize, Simulate, Train и Adapt (изображены на рис. 3.1 как неактивные) действуют применительно к тому объекту, который отмечен в данный момент выделением. Если такого объекта нет, либо над выделенным объектом

невозможно произвести указанное действие, соответствующая клавиша неактивна.

Рассмотрим создание нейронной сети с помощью NNTool на примере.

Пусть требуется создать нейронную сеть, выполняющую логическую функцию "И".

Создание сети

Выберем сеть, состоящую из одного персептрона с двумя входами. В процессе обучения сети на её входы подаются входные данные и производится сопоставление значения, полученного на выходе, с целевым (желаемым). На основании результата сравнения (отклонения полученного значения от желаемого) вычисляются величины изменения весов и смещения, уменьшающие это отклонение.

Итак, перед созданием сети необходимо заготовить набор обучающих и целевых данных. Составим таблицу истинности для логической функции "И", где P1 и P2 - входы, а A – желаемый выход (таблица 1.1).

Таблица 1.1 - Таблица истинности логической функции "И"

P_1	P_2	A
0	0	0
0	1	0
1	0	0
1	1	1

Чтобы задать матрицу, состоящую из четырёх векторов-строк, как входную, воспользуемся кнопкой New Data. В появившемся окне следует произвести изменения (рисунок 1.2), и нажать клавишу "Создать" (Create).

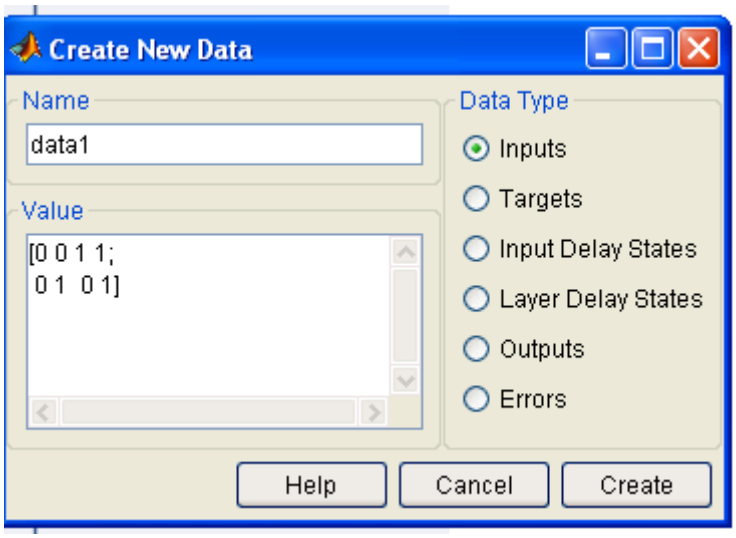

Рисунок 1.2 – Задание входных векторов

После этого в окне управления появится вектор data1 в разделе Inputs. Вектор целей задаётся схожим образом (рисунок 1.3).

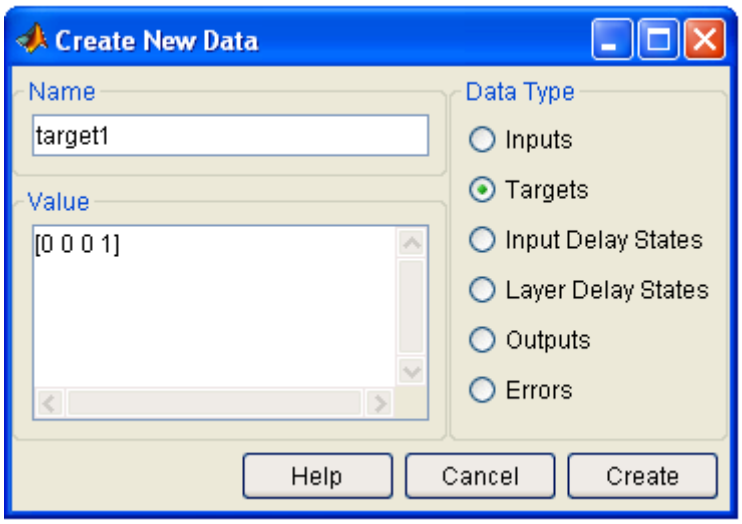

Рисунок 1.3 – Задание целевого вектора

После нажатия на Create в разделе Targets появится вектор target1. Данные в поле "Значение" (Value) могут быть представлены любым понятным MATLAB выражением. К примеру, предыдущее определение вектора целей можно эквивалентно заменить строкой вида

```
bitand([0 0 1 1], [0 1 0 1]).
```

Теперь следует приступить к созданию нейронной сети. Выбираем кнопку New Network и заполняем форму (рисунок 1.4).

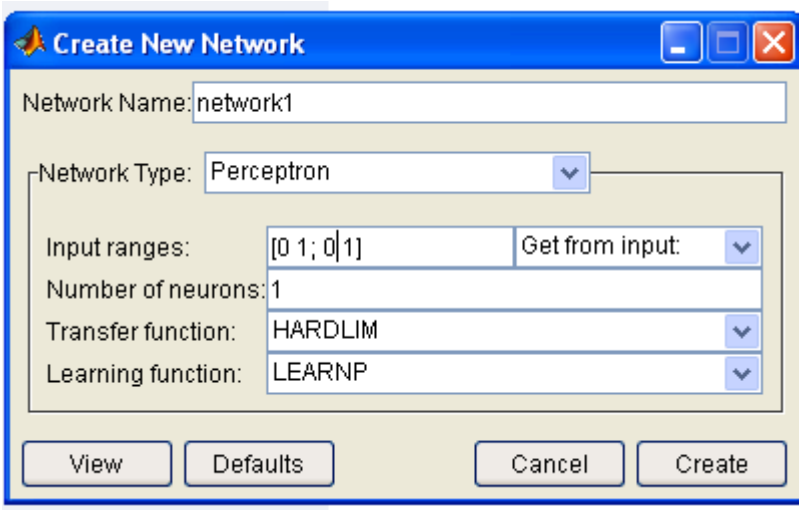

Рисунок 1.4 – Окно «Создание нейронной сети»

При этом поля несут следующие смысловые нагрузки:

- Имя сети (Network Name) − это имя объекта создаваемой сети.
- Тип сети (Network Type) − определяет тип сети и в контексте выбранного типа представляет для ввода различные параметры в части окна, расположенной ниже этого пункта. Таким образом, для разных типов сетей окно изменяет своё содержание. Для удобства список НС повторен в таблице 1.2. Интерфейс NNTool позволяет создавать нейронные сети только с одним или двумя слоями.
- Входные диапазоны (Input ranges) − матрица с числом строк, равным числу входов сети. Каждая строка представляет собой вектор с двумя элементами: первый – минимальное значение сигнала, которое будет подано на соответствующий вход сети при обучении, второй − максимальное. Для упрощения ввода этих значений предусмотрен выпадающий список "Получить из входа" (Get from input), позволяющий автоматически сформировать необходимые данные, указав имя входной переменной.
- Количество нейронов (Number of neurons) − число нейронов в слое.
- Передаточная функция (Transfer function) − в этом пункте выбирается передаточная функция (функция активации) нейронов.
- Функция обучения (Learning function) − функция, отвечающая за обновление весов и смещений сети в процессе обучения.

С помощью клавиши "Вид" (View) можно посмотреть архитектуру создаваемой сети (рисунок 1.5). Так, мы имеем возможность удостовериться, все ли действия были произведены верно. На рисунке 1.5 изображена персептронная сеть с выходным блоком, реализующим передаточную функцию с жёстким ограничением. Количество нейронов в слое равно одному, что символически отображается размерностью вектора-столбца на выходе слоя и указывается числом непосредственно под блоком передаточной функции. Рассматриваемая сеть имеет два входа, так как размерность входного вектора-столбца равна двум.

Таблица 1.2 - Типы нейронных сетей NNTool

№ п/п	Тип сети	Название сети	Число слоев	Обучаемые параметры
1	2	3	4	5
1	Competitive	Конкурирующая сеть	1	IW{1, 1}, b{ 1}
2	Cascade-forward backprop	Каскадная сеть с прямым распространением сигнала и обратным распространением ошибки	2	IW{1, 1}, b{1} LW{2, 1} IW{2, 1}, b{2}
3	Elman backprop	Сеть Элмана с обратным распространением ошибки	2	IW{1, 1}, b{1} LW{2, 1}, b{2} LW{2, 1},
4	Feed-forward backprop	Сеть с прямым распространением сигнала и обратным распространением ошибки	2	IW{1, 1}, b{1} LW{2, 1}, b{2}
5	Time delay backprop	Сеть с запаздыванием и обратным распространением ошибки	2	IW{1, 1}, b{1}, LW{2, l},b{2}
6	Generalized regression	Обобщенная регрессионная сеть	2	IW{1, 1}, b{1}, LW{2, l}
7	Hopfield	Сеть Хопфилда	1	LW{l, l},b{l}
8	Linear layer (design)	Линейный слой (создание)	1	IW{1, 1}, b{1}
9	Linear layer (train)	Линейный слой (обучение)	1	IW{1, 1}, b{1}
10	LVQ	Сеть для классификации входных векторов	2	IW{1, 1}, LW{2, 1}
11	Perceptron	Персептрон	1	IW{1, 1), b{l}

Продолжение таблицы 1.2

1	2	3	4	5
12	Probabalistic	Вероятностная сеть	2	IW{1, 1}, b{l}, LW{2, 1}
13	Radial basis (exact fit)	Радиальная базисная сеть с нулевой ошибкой	2	IW{ 1, 1}, b{ 1}, LW{2, 1}
14	Radial basis (fewer neurons)	Радиальная базисная сеть с минимальным числом нейронов	2	IW{1, 1}, b{ 1}, LW{2, 1}, b{2}
15	Self-organizing map	Самоорганизующаяся карта Кохонена	1	IW{1, 1}

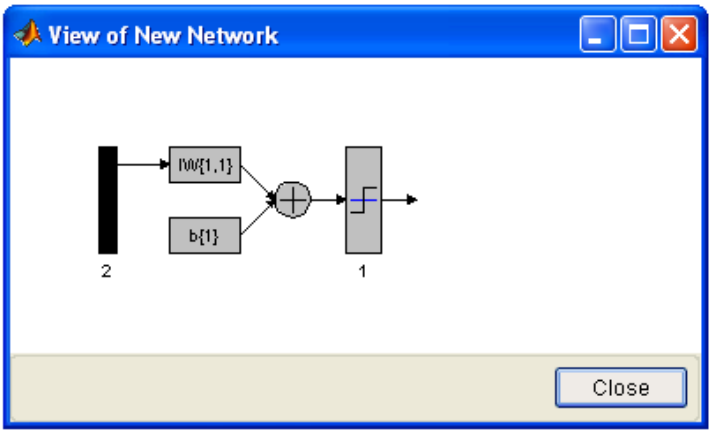

Рисунок 1.5 – Предварительный просмотр создаваемой сети

Итак, структура сети соответствует нашему заданию. Теперь можно закрыть окно предварительного просмотра, нажав клавишу "Закрыть" (Close), и подтвердить намерение создать сеть, нажав "Создать" (Create) в окне создания сети. В результате проделанных операций в разделе "Сети" (Networks) главного окна NNTool появится объект с именем network1.

Обучение

Наша цель – построить нейронную сеть, которая выполняет функцию логического "И". Очевидно, нельзя рассчитывать на то, что сразу после этапа создания сети последняя будет обеспечивать правильный результат (правильное соотношение "вход/выход"). Для достижения цели сеть необходимо должным образом обучить, то есть подобрать подходящие значения параметров. В MATLAB реализовано большинство известных алгоритмов обучения нейронных сетей, среди которых представлено два для персептронных сетей рассматриваемого вида. Создавая сеть, мы указали LEARNP в качестве функции, реализующей алгоритм обучения (рисунок 1.4).

Вернёмся в главное окно NNTool. На данном этапе интерес представляет нижняя панель "Только сети" (Networks only). Нажатие любой из клавиш на этой панели вызовет окно, на множестве вкладок которого представлены параметры сети, необходимые для её обучения и прогона, а также отражающие текущее состояние сети.

Отметив указателем мыши объект сети network1, вызовем окно управления сетью нажатием кнопки Train. Перед нами возникнет вкладка "Train" окна свойств сети, содержащая, в свою очередь, ещё одну панель вкладок (рисунок 1.6). Их главное назначение - управление процессом обучения. На вкладке "Информация обучения" (Training info) требуется указать набор обучающих данных в поле "Входы" (Inputs) и набор целевых данных в поле "Цели" (Targets). Поля "Выходы" (Outputs) и "Ошибки" (Errors) NNTool заполняет автоматически. При этом результаты обучения, к которым относятся выходы и ошибки, будут сохраняться в переменных с указанными именами.

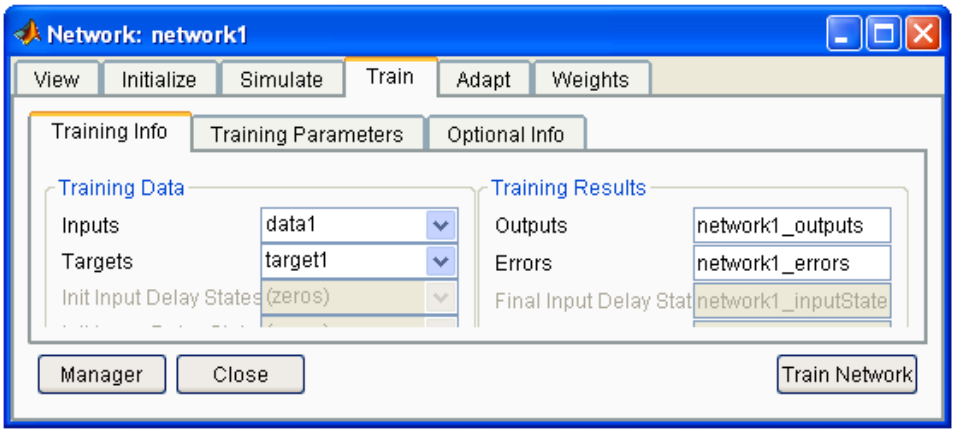

Рисунок 1.6 – Окно параметров сети, открытое на вкладке "обучение" (Train)

Завершить процесс обучения можно, руководствуясь разными критериями. Возможны ситуации, когда предпочтительно остановить обучение, полагая достаточным некоторый интервал времени. С другой стороны, объективным критерием является уровень ошибки. На вкладке "Параметры обучения" (Training parame-ters) для нашей сети (рисунок 1.7) можно установить следующие поля:

- Количество эпох (epochs) – определяет число эпох (интервал времени), по прошествии которых обучение будет прекращено. Эпохой называют однократное представление всех обучающих входных данных на входы
- сети.
- Достижение цели или попадание (goal) – здесь задаётся абсолютная величина функции ошибки, при которой цель будет считаться достигнутой.
- Период обновления (show) – период обновления графика кривой обучения, выраженный числом эпох.
- Время обучения (time) – по истечении указанного здесь временного интервала, выраженного в секундах, обучение прекращается.

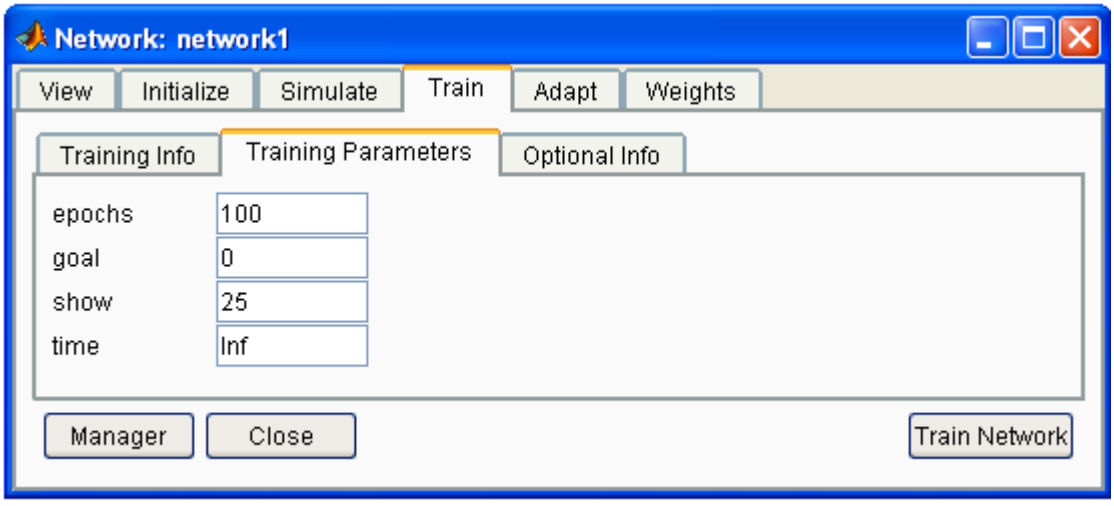

Рисунок 1.7 – Вкладка параметров обучения

Принимая во внимание тот факт, что для задач с линейно отделимыми множествами (а наша задача относится к этому классу) всегда существует точное решение, установим порог достижения цели, равный нулю. Значения остальных параметров оставим по умолчанию. Заметим только, что поле времени обучения содержит запись Inf,

которая определяет бесконечный интервал времени (от английского Infinite – бесконечный).

Следующая вкладка "Необязательная информация" (Optional Info) показана на рисунке 1.8.

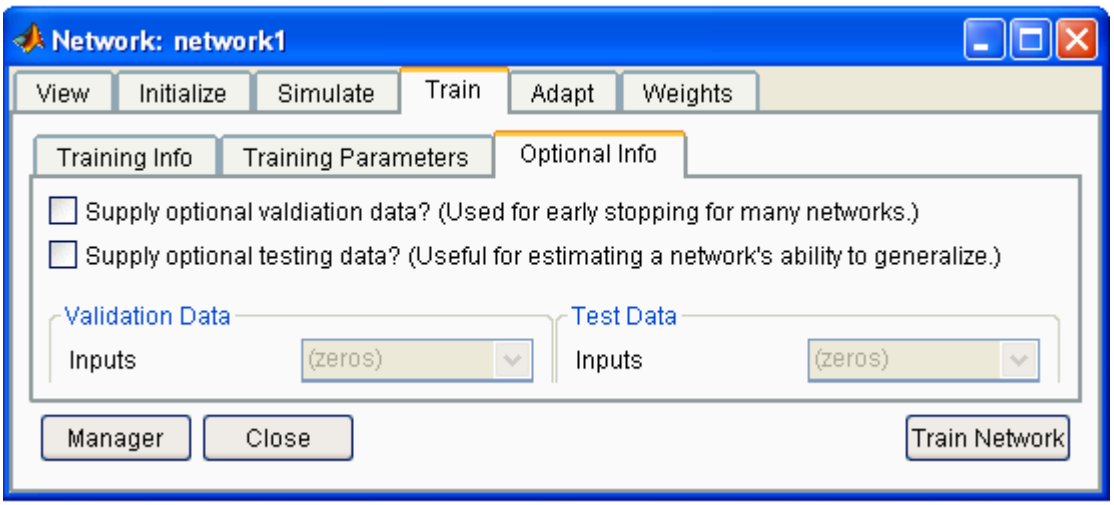

Рисунок 1.8 – Вкладка необязательной информации

Рассмотрим вкладку обучения (Train). Чтобы начать обучение, нужно нажать кнопку "Обучить сеть" (Train Network). После этого, если в текущий момент сеть не удовлетворяет ни одному из условий, указанных в разделе параметров обучения (Training Parameters), появится окно, иллюстрирующее динамику целевой функции - кривую обучения. В нашем случае график может выглядеть так, как показано на рисунке 1.9.

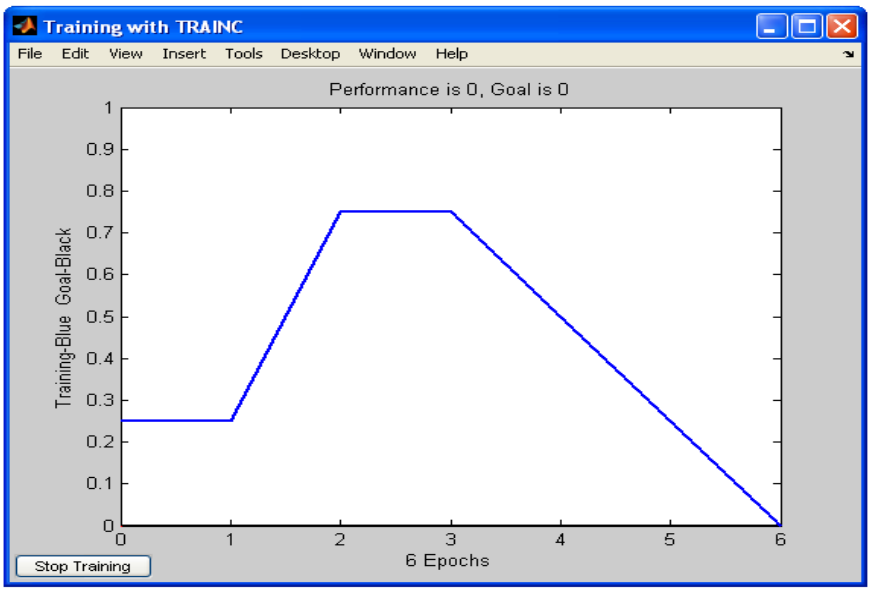

Рисунок 1.9 – Кривая обучения

Кнопкой "Остановить обучение" (Stop Training) можно прекратить этот процесс. Из рисунка видно, что обучение было остановлено, когда функция цели достигла установленной величины (goal=0).

Следует отметить, что для персептронов, имеющих функцию активации с жёстким ограничением, ошибка рассчитывается как разница между целью и полученным выходом. Итак, алгоритм обучения нашёл точное решение задачи. В методических целях убедимся в правильности решения задачи путём прогона обученной сети. Для этого необходимо открыть вкладку "Прогон" (Simulate) и выбрать в выпадающем списке "Входы" (Inputs) заготовленные данные. В данной задаче естественно использовать тот же набор данных, что и при обучении data1. При желании можно установить флажок "Задать цели" (Supply Targets). Тогда в результате прогона дополнительно будут рассчитаны значения ошибки. Нажатие кнопки "Прогон сети" (Simulate Network) запишет результаты прогона в переменную, имя которой указано в поле "Выходы" (Outputs). Теперь можно вернуться в основное окно NNTool и, выделив мышью выходную переменную network1, нажать кнопку "Просмотр" (View). Содержимое окна просмотра совпадает со значением вектора целей − сеть работает правильно.

Следует заметить, что сеть создаётся инициализированной, то есть значения весов и смещений задаются определённым образом. Перед каждым следующим опытом обучения обычно начальные условия обновляются, для чего на вкладке "Инициализация" (Initialize) предусмотрена функция инициализации. Так, если требуется провести несколько независимых опытов обучения, инициализация весов и смещений перед каждым из них осуществляется нажатием кнопки "Инициализировать веса" (Initialize Weights).

Вернёмся к вкладке "Необязательная информация" (Optional Info) (рисунок 1.8). Чтобы понять, какой цели служат представленные здесь параметры, необходимо обсудить два понятия: переобучение и обобщение.

При выборе нейронной сети для решения конкретной задачи трудно предсказать её порядок. Если выбрать неоправданно большой порядок, сеть может оказаться слишком гибкой и может представить простую зависимость сложным образом. Это явление называется переобучением. В случае сети с недостаточным количеством нейронов, напротив, необходимый уровень ошибки никогда не будет достигнут. Здесь налицо чрезмерное обобщение. Для предупреждения переобучения применяется следующая техника. Данные делятся на два множества: обучающее

(Training Data) и контрольное (Validation Data). Контрольное множество в обучении не используется. В начале работы ошибки сети на обучающем и контрольном множествах будут одинаковыми. По мере того, как сеть обучается, ошибка обучения убывает, и, пока обучение уменьшает действительную функцию ошибки, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, это указывает на то, что обучение следует закончить. Остановка на этом этапе называется ранней остановкой (Early stopping). Таким образом, необходимо провести серию экспериментов с различными сетями, прежде чем будет получена подходящая. При этом чтобы не быть введённым в заблуждение локальными минимумами функции ошибки, следует несколько раз обучать каждую сеть. Если в результате последовательных шагов обучения и контроля ошибка остаётся недопустимо большой, целесообразно изменить модель нейронной сети (например, усложнить сеть, увеличив число нейронов, или использовать сеть другого вида). В такой ситуации рекомендуется применять ещё одно множество − тестовое множество наблюдений (Test Data), которое представляет собой независимую выборку из входных данных. Итоговая модель тестируется на этом множестве, что даёт дополнительную возможность убедиться в достоверности полученных результатов. Очевидно, чтобы сыграть свою роль, тестовое множество должно быть использовано только один раз. Если его использовать для корректировки сети, оно фактически превратится в контрольное множество. Установка верхнего флажка (рисунок 1.8) позволит задать контрольное множество и соответствующий вектор целей (возможно, тот же, что при обучении). Установка нижнего флажка позволяет задать тестовое множество и вектор целей для него.

Обучение сети можно проводить в разных режимах. В связи с этим, в NNTool предусмотрено две вкладки, представляющие обучающие функции: рассмотренная ранее вкладка Train и "Адаптация" (Adapt). Adapt вмещает вкладку информация адаптации (Adaption Info), на которой содержатся поля, схожие по своему назначению с полями вкладки Training Info и выполняющие те же функции и вкладку параметры адаптации (Adaption Parameters). Последняя содержит единственное поле "Проходы" (passes). Значение, указанное в этом поле, определяет, сколько раз все входные векторы будут представлены сети в процессе обучения. Параметры вкладок "Train" и "Adapt" в MATLAB используются функциями train и adapt, соответственно.

Выявление показателей, влияющих на валовую прибыль предприятия

Пусть требуется на основе экспертных данных выявить факторы, наиболее сильно влияющие на ежемесячную прибыль медицинского предприятия (таблица 1.3).

Для начала проведем предварительный анализ задачи, выделив наименее важные показатели. Это нужно потому, что количество наборов параметров близко к количеству самих параметров, и в данном представлении задача может не иметь видимого решения. В результате ряд следующих показателей может быть исключен из дальнейшего рассмотрения:

– объем реализации по линии бюджета – поскольку данные являются неполными, содержат пропуски;

– затраты – поскольку мы уже рассматриваем их по частям: на материалы и зарплату;

– объем реализованной продукции, рентабельность, так как они связаны жесткой аналитической зависимостью с другими показателями;

– численность – практически постоянная величина;

– цена единицы продукции – никак не связана с прибылью и другими показателями.

В результате оставлены факторы: затраты на материалы, объем заработной платы, производительность, курс доллара.

Таблица 1.3 − Показатели, характеризующие деятельность предприятия

Фактор	ед. изм.	1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12	13	14
Объем реализации (без НДС)	тыс. руб.	354	310	277	302	327	211	263	168	278	292	305	326
в том числе бюджет	тыс. руб.		20		37		27			30	18	10	19
Затраты, в том числе	тыс. руб.	255	281	319	251	215	203	208	172	323	262	239	475
материалы	тыс. руб.	53	58	44	63	38	32	41	33	45	50	39	58
заработная плата	тыс. руб.	122	126	126	104	112	74	102	76	123	117	107	218
Численность	чел.	59	62	62	63	62	62	62	63	63	62	62	62

Продолжение таблицы 1.3

1	2	3	4	5	6	7	8	9	10	11	12	13	14
Производительность	руб./ чел.	6003	5002	4474	4798	5273	3410	4237	2673	4406	4711	4833	5253
Цена ед. продукции	руб.	0,08	0,08	0,08	0,06	0,07	0,06	0,08	0,08	0,12	0,10	0,15	0,15
Рентабельность	%	38,9	10,4		20,4	52,0	4,1	26,3			11,5	27,4	
Курс $	руб.	6,0	6,1	6,1	6,1	6,2	6,2	6,2	7,9	16,1	16,0	17,9	20,1
Прибыль валовая	тыс. руб.	99	29	-42	51	112	8	55	-4	-45	30	66	

Задание

1. Реализовать с помощью NNTool нейронную сеть прямого распространения.
2. Обучить сеть, основываясь на важных показателях из таблицы 1.3.
3. Спрогнозировать валовую прибыль за 12 отчетный период.
4. Сделать выводы, объяснить положительный или отрицательный результат.

Контрольные вопросы

1. В какой форме принимает и выдает данные пакет Matlab?
2. По каким критериям завершается обучение в пакете Matlab?
3. Как использовать обученную в Matlab сеть?
4. Как готовятся тренировочные и тестовые данные для пакета Matlab?
5. Что включает в себя нейропроект в Matlab?
6. Как происходит процесс обучения сети и ее использования в Matlab?
7. Какие функции активации нейронов реализованы в данном пакете?
8. Какие методы упрощения сети реализованы в Matlab?

2 Экспертные системы и нейронные сети. Реализация в программном продукте Matlab

Экспертные системы

В середине семидесятых годов в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название экспертные системы. Цель исследований по экспертным системам состоит в разработке программ (устройств), которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. В большинстве случаев экспертные системы решают трудно формализуемые задачи или задачи, не имеющие алгоритмического решения.

Экспертная система - программно-техническое средство, позволяющее пользователю в диалоговом режиме получать от компьютера консультационную помощь в конкретной предметной области, где сконцентрированы опыт и знания людей-экспертов (специалистов в данной области).

Экспертные системы – программы для компьютера, которые могут воспроизводить процесс решения проблемы человеком-экспертом.

Экспертная система - программа, которая использует знания специалистов (экспертов) о некоторой конкретной узкоспециализированной предметной области и в пределах этой области способна принимать решения на уровне эксперта-профессионала.

Экспертные системы - прикладные программы ИИ, в которых база знаний представляет собой формализованные эмпирические знания высококвалифицированных специалистов (экспертов) в какой-либо узкой предметной области.

Экспертная система - программа для компьютера, которая оперирует со знаниями в определенной предметной области с целью выработки рекомендаций или решения проблем.

В основе функционирования ЭС лежит использование знаний, а манипулирование ими осуществляется на базе эвристических правил, сформулированных экспертами. ЭС выдают советы, проводят анализ, выполняют классификацию, дают консультации и ставят диагноз. Они ориентированы на решение задач, обычно требующих проведения экспертизы человеком-специалистом. В отличие от машинных программ, использующий процедурный анализ, ЭС решают задачи в

узкой предметной области (конкретной области экспертизы) на основе дедуктивных рассуждений. Главное достоинство экспертных систем - возможность накапливать знания, сохранять их длительное время, обновлять и тем самым обеспечивать относительную независимость конкретной организации от наличия в ней квалифицированных специалистов.

Классификация и виды экспертных систем

Для классификации ЭС используют следующие признаки:
- способ формирования решения;
- способ учета временного признака;
- вид используемых данных;
- число используемых источников решения знаний.

По способу формирования решения ЭС можно разделить на анализирующие и синтезирующие. В системах первого типа осуществляется выбор решения из множества известных решений на основе анализа знаний, в системах второго типа решение синтезируется из отдельных фрагментов знаний.

В зависимости от способа учета временного признака ЭС делят на статические и динамические. Статические ЭС предназначены для решения задач с неизменяемыми в процессе решения данными и знаниями, а динамические ЭС допускают такие изменения.

По видам используемых данных и знаний различают ЭС с детерминированными и неопределенными знаниями. Под неопределенностью знаний и данных понимаются их неполнота, ненадежность, нечеткость.

ЭС могут создаваться с использованием одного или нескольких источников знаний.

Структура экспертной системы

Типичная статическая ЭС состоит из следующих основных компонентов (рисунок 1):
- решателя (интерпретатора);
- рабочей памяти (РП), называемой также базой данных (БД);
- базы знаний (БЗ);
- компонентов приобретения знаний;
- объяснительного компонента;

- диалогового компонента.

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

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

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

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

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

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

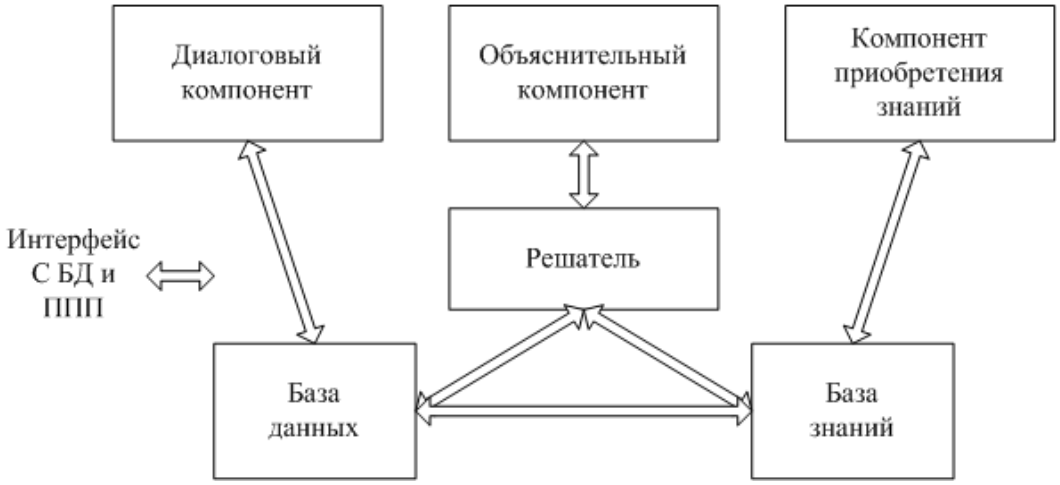

Рисунок 1 – Структура экспертной системы

Базы знаний и модели представления знаний

База знаний - важная компонента экспертной системы, она предназначена для хранения долгосрочных данных, описывающих рассматриваемую предметную область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

В качестве предметной области выбирается узкая (специальная) прикладная область. Далее для создания ЭС в выбранной области собираются факты и правила, которые помещаются в базу знаний вместе с механизмами вывода и упрощения. В отличие от всех остальных компонент ЭС, база знаний - "переменная " часть системы, которая может пополняться и модифицироваться инженерами знаний и опыта использование ЭС, между консультациями (а в некоторых системах и в процессе консультации).

Существует несколько способов представления знаний в ЭС, однако общим для всех них является то, что знания представлены в символьной форме (элементарными компонентами представления знаний являются тексты, списки и другие символьные структуры). Тем самым, в ЭС реализуется принцип символьной природы рассуждений, который заключается в том, что процесс рассуждения представляется как последовательность символьных преобразований.

Существуют динамические и статические базы знаний. Динамическая база знаний изменяется со временем. Ее содержимое зависит и от состояния окружающей. Новые факты, добавляемые в базу знаний, являются результатом вывода, который состоит в применении правил к имеющимся фактам. В системах с монотонным выводом факты, хранимые в базе знаний, статичны, то есть не изменяются в процессе решения задачи. В системах с немонотонным выводом допускается изменение или удаление фактов из базы знаний.

Одной из наиболее важных проблем, характерных для систем, основанных на знаниях, является проблема представления знаний. Это объясняется тем, что форма представления знаний оказывает существенное влияние на характеристики и свойства системы. Для того чтобы манипулировать всевозможными знаниями из реального мира с помощью компьютера, необходимо осуществлять их моделирование. В таких случаях необходимо отличать знания, предназначенные для обработки компьютером, от знаний, используемых человеком.

При проектировании модели представления знаний следует учитывать такие факторы, как однородность представления и простота

понимания. Однородное представление приводит к упрощению механизма управления логическим выводом и упрощению управления знаниями. Представление знаний должно быть понятным экспертам и пользователям системы. В противном случае затрудняются приобретение знаний и их оценка. Однако выполнить это требование в равной степени, как для простых, так и для сложных задач довольно трудно. Обычно, для несложных задач останавливаются на некотором среднем (компромиссном) представлении, но для решения сложных и больших задач необходимы структурирование и модульное представление.

Типичными моделями представления знаний являются:
- продукционная модель;
- модель, основанная на использовании фреймов;
- модель семантической сети;
- логическая модель.

Продукционная модель - модель, основанная на правилах, позволяющая представить знания в виде предложений типа:

Если (условие), то (действие)

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

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

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

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

Одним из способов представления знаний является *семантическая сеть*. Изначально семантическая сеть была задумана как модель преставления структуры долговременной памяти в психологии, но в последствии стала одним из основных способов представления знаний в инженерии знаний.

В основе сетевых моделей представления знаний лежит идея о том, что любые знания можно представить в виде совокупности объектов (понятий) и связей (отношений) между ними. В отличие от продукционных эти модели более наглядны, поскольку любой пример можно представить в виде ориентированного (направленного) графа, вершины которого - понятия, а дуги - отношения между ними.

Понятиями обычно выступают абстрактные или конкретные объекты, а отношения - это связи типа: "это" ("is"), "имеет частью" ("has part"), "принадлежит", "любит" и т.п. Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

- класс - элемент класса.
- свойство – значение.
- пример элемента класса.

Традиционно в представлении знаний выделяют *логические модели,* основанные на классическом исчислении предикатов первого порядка, когда предметная область или задача описывается в виде набора аксиом. Основное преимущество использования логики предикатов для представления знаний заключается в том, что обладающий хорошо понятными математическими свойствами мощный механизм вывода может быть непосредственно запрограммирован. С помощью этих программ из известных ранее знаний могут быть получены новые знания.

Механизмы логического вывода

При использовании продукционной модели база знаний состоит из набора правил. Программа, управляющая перебором правил, называется машиной вывода. Механизм выводов связывает знания.

Два способа использования продукционных правил:

- прямая цепочка рассуждений;
- обратная цепочка рассуждений.

Первый предполагает обработку информации в прямом направлении (метод сопоставления), когда образцом для поиска служит левая часть продукционного правила — условие, то есть задача решается в направлении от исходного состояния к целевому. Это соответствует стратегии «от данных к цели» или стратегии управления данными.

При втором подходе обработка информации осуществляется в обратном направлении — метод «генерации» или выдвижения гипотезы и ее проверки (стратегия «от цели к данным»).

Пример:Имеется фрагмент БЗ из двух правил:

П1: ЕСЛИ «отдых - летом» и «человек - активный», ТО «ехать в горы».

П2: ЕСЛИ «любит солнце»,«отдых летом».

Предположим в систему поступили данные: «человек - активный» и «любит солнце». Прямой вывод: исходя из данных, получить ответ:

Шаг 1. Пробуем П1 не работает - не хватает данных «отдых - летом».

Шаг 2. Пробуем П2 , работает, в базу поступает факт «отдых - летом». 2-й проход:

Шаг 3. Пробуем П1 , работает, активируя цель «ехать в горы», которая и выступает, например, как совет, который дает система.

Обратный вывод: подтвердить выбранную цель при помощи имеющихся правил и данных:

Шаг 1. Цель - «ехать в горы»:
становятся новой целью, и имеется правило, где она в правой части.

Шаг 2. Цель «отдых летом»:
правило П2 подтверждает цель и активизирует ее. 2-й проход:

Шаг 3. Пробуем П1 , подтверждается искомая цель.

Формализованные экспертный систем являются одним из наглядных методов решения задачи прогнозирования результатов сдачи сессии, на основе анализа текущей успеваемости. Поэтому, построение адекватных моделей, а также разработка методов и алгоритмов,

позволяющих установить соответствие между параметрами математических моделей и реальными системами, является актуальной научно-прикладной задачей.

Нейронные сети

Свойства и назначение нейронных сетей

Под методами *искусственного интеллекта* понимают все методы и математические модели, которые имитируют мыслительную (интеллектуальную) деятельность человека. В этом отношении можно выделить два подхода:

- моделирование *процесса* (алгоритма) мышления при принятии какого-либо решения,
- моделирование *работы мозга* на основе изучения биологических принципов его функционирования.

Первый подход реализуется в виде нечетких логических функций (Fuzzy Logic), экспертных систем, естественно-языковых систем. Второй подход представлен нейронными сетями, которые фактически являются математическими моделями мозга.

Основные свойства НС:

- массивно-параллельная обработка,
- высокая отказоустойчивость,
- использование неалгоритмических вычислений,
- способность к обобщению и классификации данных,
- способность к обучению с учителем или без учителя,
- возможность использования слабоформализованной информации.

НС получили распространение благодаря тому, что они:

- дают стандартный способ решения многих нестандартных задач;
- обладают способностью к обучению - необходимо только формировать учебные задачники, то есть труд программиста замещается трудом учителя;
- особенно эффективны там, где нужен аналог человеческой интуиции;
- позволяют создавать эффективное программное обеспечение для компьютеров с высокой степенью параллельной обработки.

В литературе встречается значительное число признаков, которыми должна обладать задача, чтобы применение НС было оправдано и НС могла бы ее решить:

- отсутствует алгоритм или не известны принципы решения задач, но накоплено достаточное число примеров;

- проблема характеризуется большими объемами входной информации;

- данные неполны или избыточны, зашумлены, частично противоречивы.

Таким образом, НС хорошо подходят для распознавания образов и решения задач классификации, оптимизации и прогнозирования. В нефтяной и химической промышленности НС могут применяться для анализа геологической информации, идентификация неисправностей оборудования, разведки залежей минералов по данным аэрофотосъемок, анализа составов примесей, при управлении процессами. Кроме того, с помощью НС может быть реализовано управление манипуляторами, управление качеством, обнаружение неисправностей, адаптивная робототехника, управление голосом и т.д.

Принцип функционирования нейронных сетей

Нейронная сеть представляет собой совокупность нейронов, связанных между собой соответствующим образом.

Нейрон – преобразовательный элемент, имеющий некоторое количество входов (синапсов), на которые поступают входные сигналы x_i и один выход (аксон), с которого снимается выходной сигнал y. Каждый синапс имеет вес w_i, на который умножается входной сигнал x_i.

Структура нейрона представлена на рисунке 2. Внутри нейрона можно выделить блок суммирования, определяющий взвешенную сумму всех входных сигналов

$$U = \sum_{i=1}^{n} w_i \cdot x_i$$

и блок функции активации Y=F(U). Таким образом, нейрон функционирует за два такта: 1) суммирование входных сигналов и 2)вычисление Y по функции активации.

Функция активации должна удовлетворять двум условиям: 1)|F(U)|<1 при любом U, 2) функция должна быть монотонной неубывающей.

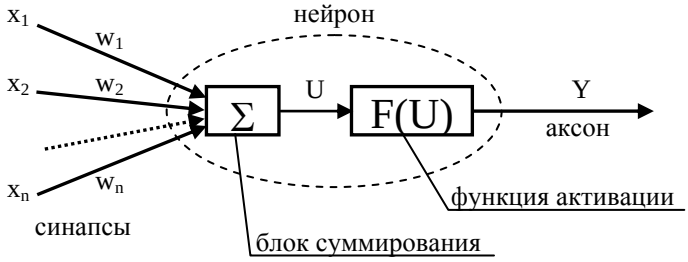

Рисунок 2 – Структура нейрона

Наиболее часто в качестве функций активации используются следующие функции.

1) Ступенчатая функция $F(U) = \begin{cases} 1, & \text{если } U \geq \alpha \\ 0, & \text{если } U < \alpha \end{cases}$;

2) Сигмоидная функция $F(U) = \dfrac{1}{1+e^{-\alpha U}}$;

3) Гиперболический тангенс $F(U) = th(U) = \dfrac{e^{U} - e^{-U}}{e^{U} + e^{-U}}$;

4) Гладкие сжимающие функции $F(U) = \dfrac{U+Q}{|U+Q|+\alpha}$,

где Q – порог (смещение), α - параметр, определяющий крутизну статической характеристики нейрона.

Нейроны образуют нейронные сети путем соединения синапсов с аксонами.

Наиболее распространенными и хорошо изученными являются трехслойные НС, состоящие из трех слоев нейронов: входного, скрытого и выходного. Нейроны входного слоя имеют только по одному синапсу. Количество нейронов входного слоя соответствует количеству входных переменных сети X. Задачей нейронов этого слоя является только распределение входных сигналов по нейронам скрытого слоя, суммирования и вычисления функции активации в них не происходит. Количество нейронов в скрытом слое может быть различным и часто подбирается экспериментально. Недостаточное или избыточное количество нейронов в скрытом слое приводит к ухудшению точности аппроксимации. Кроме того, избыточное количество усложняет сеть и уменьшает быстродействие. Нейроны выходного слоя формируют выходные сигналы, их количество соответствует количеству выходов Y. Пример НС с 3 входными, 4 скрытыми и 2 выходными нейронами приведен на рисунке 3. Такая НС для краткости обозначается как (3-4-2). H_{ij} – нейроны.

Данные сети относятся к сетям *прямого распространения*, поскольку в них входные сигналы последовательно проходят через все нейроны и после преобразований напрямую подаются на выходы. Выходной сигнал y_{ij} каждого j-го нейрона в i-м слое определяется как

$$y_{ij} = F\left(\sum_{k=1}^{n(i-1)} w_{ij}^{k} \cdot y_{i-1,k} \right),$$

где n(i) – число нейронов в i-м слое.

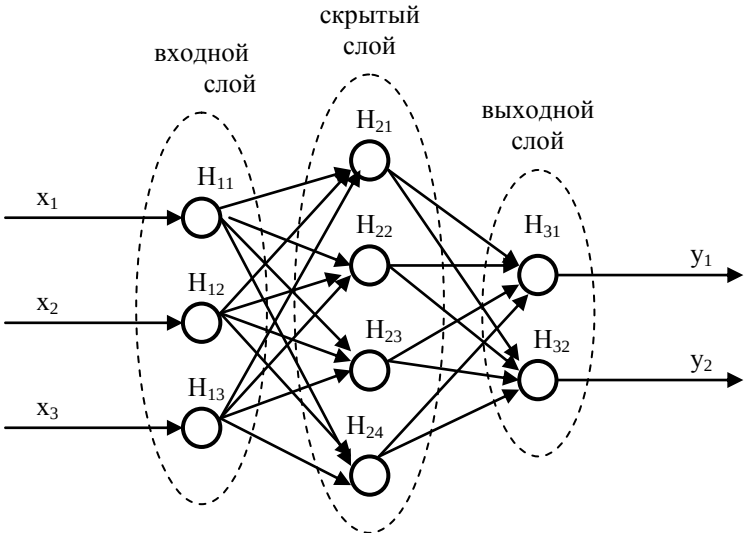

Рисунок 3 – Нейронная сеть вида (3-4-2)

Выходные сигналы НС соответствуют $y_{N_{out}\,j}$, где N_{out} – число нейронов выходного слоя.

Наиболее популярный класс многослойных сетей прямого распространения образуют многослойные персептроны (MLP), в которых каждый вычислительный элемент использует пороговую или сигмоидальную функцию активации. Многослойный перцептрон может формировать сколь угодно сложные границы принятия решения и реализовывать произвольные булевы функции.

Сети, использующие радиальные базисные функции (RBF-сети), являются частным случаем двухслойной сети прямого распространения. Каждый элемент скрытого слоя использует в качестве активационной функции радиальную базисную функцию типа гауссовой. Радиальная базисная функция (функция ядра) центрируется в точке, которая определяется весовым вектором, связанным с нейроном. Каждый

выходной элемент вычисляет линейную комбинацию этих радиальных базисных функций. С точки зрения задачи аппроксимации скрытые элементы формируют совокупность функций, которые образуют базисную систему для представления входных примеров в построенном на ней пространстве.

Методы обучения нейронных сетей

Нейронные сети относятся к классу аппроксиматоров и «черных ящиков», аппроксимирующих некоторые функции вида Y F(X), где Y – вектор выходных переменных, X – вектор входных [1, 10, 11].

Процесс аппроксимации заключается в подборе весовых коэффициентов w_{ij} и называется *обучением* НС. То есть НС может функционировать в двух режимах:

- эксплуатации, когда на вход подаются сигналы, а на выходе снимаются результаты вычислений;
- обучения, когда происходит корректировка весов таким образом, чтобы выходные сигналы наиболее точно соответствовали желаемым.

От качества обучения НС зависит точность ее работы в режиме эксплуатации.

Структура процесса обучения представлена на рисунке 4, где обозначены: $Y_{жел}$ – желаемые значения выходных сигналов, E – ошибка обучения ($E=Y_{жел}–Y$), K – корректирующие воздействия (обычно изменения весов Δw_{ij}).

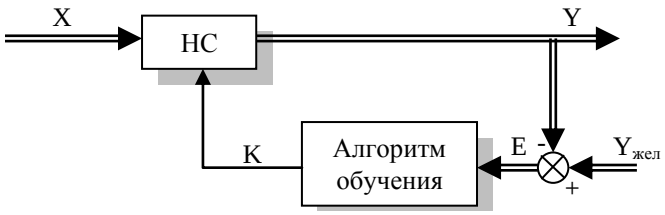

Рисунок 4 – Процесс обучения НС

Для обучения НС составляется *обучающая выборка* входных сигналов и соответствующих им выходных. Выборка может быть разделена на две части: *рабочую выборку* (на основе которой производится собственно обучение) и *тестирующую выборку* (для проверки качества обучения).

Далее определяется структура НС. Для трехслойной НС количества входных и выходных нейронов определяются по количествам входных и

выходных переменных. Количество нейронов в скрытом слое N_c может быть взято из условия:

$$N_c \le \frac{(N_p - 1) \cdot N_{out}}{N_{in} + N_{out} + 1},$$

где N_{in} и N_{out} – количества нейронов во входном и выходном слоях, N_p – количество обучающих примеров (объем выборки).

Весам синапсов необученной НС изначально присваиваются произвольные значения. Далее на вход НС подается первый вектор X из рабочей выборки, определяется вектор Y и ошибка обучения E. Исходя из значений вектора E корректируются веса синапсов. Затем подается следующий вектор X из выборки и т.д. Циклы обучения повторяются многократно, пока качество обучения не станет удовлетворительным (это можно проверить по тестирующей выборке).

Существует несколько методов обучения, которые можно классифицировать по способам использования учителя:

- обучение с учителем (коррекция весов производится исходя из сравнения текущего и желаемого выходных векторов);
- обучение с последовательным подкреплением знаний (сети не даются желаемые значения выходов, а ставится оценка «хорошо» или «плохо»);
- обучение без учителя (сеть сама вырабатывает правила обучения путем выделения особенностей из набора входных данных).

По использованию элементов случайности методы обучения подразделяются на:

- детерминистские (коррекция на основе анализа входных и выходных сигналов, а также дополнительной информации, например, желаемых выходов);
- стохастические (случайное изменение весов в ходе обучения – Больцмановское обучение).

К детерминистским правилам обучения относятся правило Хебба, дельта-правило, правило Кохонена, ART-правило, правило обратного распространения.

Наиболее распространенным правилом для сетей MLP является правило обратного распространения (back propagation).

Для обучения RBF-сетей разработаны различные алгоритмы. Основной алгоритм использует двушаговую стратегию обучения, или смешанное обучение. Он оценивает позицию и ширину ядра с

использованием алгоритма кластеризации "без учителя", а затем алгоритм минимизации среднеквадратической ошибки "с учителем" для определения весов связей между скрытым и выходным слоями. Поскольку выходные элементы линейны, применяется неитерационный алгоритм. После получения этого начального приближения используется градиентный спуск для уточнения параметров сети.

Этот смешанный алгоритм обучения RBF-сети сходится гораздо быстрее, чем алгоритм обратного распространения для обучения многослойных перцептронов. Однако RBF-сеть часто содержит слишком большое число скрытых элементов. Это влечет более медленное функционирование RBF-сети, чем многослойного персептрона. Эффективность (ошибка в зависимости от размера сети) RBF-сети и многослойного персептрона зависят от решаемой задачи.

Правило обратного распространения

Для обучения обычно используется НС с функциями активации сигмоидного типа. Целью обучения по правилу обратного распространения является минимизация ошибки обучения, которая определяется как

$$E = \frac{1}{2}\sum_{i=1}^{N_{out}} \left(Y_i - Y_{жел.i}\right)^2 .$$

Для уменьшения ошибки веса изменяются по правилу

$$w_{ij} = w_{ij} - \nu \frac{\partial E}{\partial w_{ij}},$$

где ν - константа, характеризующая скорость обучения. Данная формула описывает процесс градиентного спуска в пространстве весов.

Алгоритм обратного распространением состоит из следующих шагов.

Шаг 1. На вход НС подается вектор X из обучающей выборки и вычисляются выходы всех нейронов Y_{ij}.

Шаг 2. Определяется величина градиента ошибки EI для каждого нейрона выходного слоя:

$$EI_{Nj} = (Y_j - Y_{жел.j}) \cdot Y_j \cdot (1 - Y_j),$$

где Y_j – выход j-го нейрона выходного слоя.

Шаг 3. Двигаясь от последнего слоя к первому определяются градиенты EI_{ij} для каждого j-го нейрона каждого i-го слоя:

$$EI_{ij} = Y_{ij} \cdot (1 - Y_{ij}) \cdot \sum_{k=1}^{n(i+1)} EI_{i+1,k} \cdot w_{i+1,k}^{j},$$

где k – номер синапса, соединяющего нейрон H_{ij} с нейроном $H_{i+1,k}$ следующего слоя.

Шаг 4. Коррекция весов синапсов:

$$w_{ij}^{k} = w_{ij}^{k} - \nu \cdot EI_{ij} \cdot Y_{i-1,k}.$$

Коррекция весов для входного слоя не производится.

Шаг 5. Если обучающая выборка не закончилась, то шаги 1 – 5 повторяются.

Шаг 6. Определяется величина ошибки E. Если она не удовлетворительна, то шаги 1 – 6 повторяются.

Из описанного алгоритма видно, что процесс обучения НС включает два вложенных цикла обучения: внутренний цикл (шаги 1 – 5) повторяется соответственно количеству примеров из обучающей выборки, внешний (шаги 1 – 6) – до тех пор, пока не будет достигнуто удовлетворительное (с точки зрения ошибки E) качество обучения.

После успешного обучения НС может быть протестирована на тестовой выборке. Если ошибка обучения на каждом примере из тестовой выборки удовлетворительна, то НС можно считать обученной и приступать к ее эксплуатации.

Семантические сети и фреймы

Семантическая сеть

Семантическая сеть представляет собой направленный граф, вершины которого соответствуют объектам (понятиям, сущностям) предметной области, а дуги – отношениям (связям) между ними. И узлы,

и дуги имеют метки (имена). Имена вершин и дуг обычно совпадают с именами соответствующих объектов и отношений предметной области.

Объекты предметной области, отображаемые семантической сетью, можно условно разделить на три группы: обобщенные, индивидные и агрегатные объекты. Обобщенный объект соответствует собирательному понятию некоторой абстракции реально существующего объекта, процесса или явления предметной области. Индивидный объект – это каким-то образом выделенный единичный представитель класса. Агрегатным называется составной объект, образованный из других объектов, которые рассматриваются как его составные части.

Типы связей между объектами семантических сетей могут быть любыми. Но чаще всего применяются следующие основные связи: «род-вид», «является представителем», «является частью». Наличие связи типа «род-вид» между обобщенными объектами A и B означает, что понятие A более общее, чем понятие B. Например, понятие «животное» - это родовое понятие для объекта «птица». Связь «является представителем» существует обычно между обобщенным и индивидным объектом, когда индивидный объект выступает в роли представителя класса.

Например, индивидный объект «овчарка Альма» является представителем обобщенного объекта «овчарка». Связь «род-вид» обычно обозначают отношением ako (от англ. a-kind-of – «разновидность»), а связь «является представителем» - отношением is_a (от англ. is a – «является», «есть»).

Для того чтобы представить семантическую сеть на языке Common Lisp можно воспользоваться списком свойств атома. В этом случае объекты семантической сети будут представляться соответствующими символами языка, а отношения – свойствами символа.

Свойство символа представляется в виде двух элементов: имени свойства и значения свойства. Свойства символа записываются в хранимый вместе с символом список свойств.

Фреймы

Фрейм — (англ. frame — «каркас» или «рамка») — способ представления знаний в искусственном интеллекте, представляющий собой схему действий в реальной ситуации. Первоначально термин «фрейм» ввёл Марвин Минский в 70-е годы XX века для обозначения

структуры знаний для восприятия пространственных сцен. Фрейм — это модель абстрактного образа, минимально возможное описание сущности какого-либо объекта, явления, события, ситуации, процесса.

Фреймы используются в системах искусственного интеллекта (например, в экспертных системах) как одна из распространенных форм представления знаний.

Различают фреймы-образцы, фреймы-экземпляры, фреймы-структуры, фреймы-роли, фреймы-сценарии, фреймы-ситуации. Система связанных фреймов может образовывать семантическую сеть. Применяются фреймы в экспертных системах и других интеллектуальных системах различного назначения.

Структура фрейма

Под структурой фрейма понимается способ использования схемы, типичной последовательности действий, ситуативная модификация фрейма. Фрейм, кроме всего прочего, включает определённое знание по умолчанию, которое называется презумпцией.

Фрейм отличает наличие определённой структуры.

Фрейм состоит из имени и отдельных единиц, называемых слотами. Он имеет однородную структуру:

ИМЯ ФРЕЙМА
Имя 1-го слота: значение 1-го слота
Имя 2-го слота: значение 2-го слота
.....
Имя N-го слота: значение N-го слота.

В качестве значения слота может выступать имя другого фрейма. Таким образом фреймы объединяются в сеть. Свойства фреймов наследуются сверху вниз, т.е. от вышестоящих к нижестоящим через так называемые АКО-связи. Слот с именем АКО указывает на имя фрейма более высокого уровня иерархии.

Незаполненный фрейм называется протофреймом, а заполненный — экзофреймом. Роль протофрейма как оболочки в экзофрейме весьма важна. Эта оболочка позволяет осуществлять процедуру внутренней интерпретации, благодаря которой данные в памяти системы не безлики, а имеют вполне определенный, известный системе смысл.

Слот может содержать не только конкретное значение, но и имя процедуры, позволяющей вычислить его по заданному алгоритму, а также одну или несколько продукций (эвристик), с помощью которых это значение определяется. В слот может входить не одно, а несколько значений. Иногда этот слот включает компонент, называемый фасетом, который задает диапазон или перечень его возможных значений. Фасет указывает также граничные значения заполнителя слота.

Помимо конкретного значения в слоте могут храниться процедуры и правила, которые вызываются при необходимости вычисления этого значения. Среди них выделяют процедуры-демоны и процедуры-слуги. Первые запускаются автоматически при выполнении некоторого условия, а вторые активизируются только по специальному запросу. Если, например, фрейм, описывающий человека, включает слоты ДАТА РОЖДЕНИЯ и ВОЗРАСТ и в первом из них находится некоторое значение, то во втором слоте может стоять имя процедуры-демона, вычисляющей возраст по дате рождения и текущей дате и активизирующейся при каждом изменении текущей даты.

Совокупность фреймов, моделирующая какую-либо предметную область, представляет собой иерархическую структуру, в которую фреймы собираются с помощью родовидовых связей. На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов. Фреймы обладают способностью наследовать значения характеристик своих родителей, находящихся на более высоком уровне иерархии. Эти значения могут передаваться по умолчанию фреймам, находящимся ниже них в иерархии, но если последние содержат собственные значения данных характеристик, то в качестве истинных принимаются именно они. Это обстоятельство позволяет без затруднений учитывать во фреймовых системах различного рода исключения.

Различают статические и динамические системы фреймов. В системах первого типа фреймы не могут быть изменены в процессе решения задачи, а в системах второго типа это допустимо.

О системах программирования, основанных на фреймах, говорят, что они являются объектно-ориентированными. Каждый фрейм соответствует некоторому объекту предметной области, а слоты содержат описывающие этот объект данные, то есть в слотах находятся значения признаков объектов. Фрейм может быть представлен в виде списка свойств, а если использовать средства базы данных, то в виде записи.

MATLAB и системы фаззи-регулирования

Широкому распространению фаззи-систем управления в немалой степени способствует программная система MATLAB, в составе которой имеется пакет программ по фаззи-логике.

Fuzzy Logic Toolbox позволяет создавать и редактировать фаззи-системы управления с нечеткой логикой, называемые в терминах программной системы *MATLAB – Fuzzy Inference System* или FIS. Эти системы можно создавать, используя как графические инструменты, так и команды рабочего окна MATLAB.

Fuzzy Logic Toolbox также позволяет управлять созданными программами и непосредственно без Simulink. Это осуществляется с помощью автономного Fuzzy Inference Engine, который работает с нечеткими системами, сохранёнными в MATLAB. Можно также настраивать *Fuzzy Logic Toolbox*, используя его для совместной работы с другими пакетами типа *Control System* (Системы управления), *Neural Network* (Нейросети) или *Optimization Toolbox* (Оптимизация).

Общие сведения о Fuzzy Logic Toolbox

Одна из целей пакета *Fuzzy Logic Toolbox* состоит в облегчении понимания и создания фаззи-систем управления. Фаззи-систему можно создавать, используя командную строку главного окна MATLAB. Однако намного проще сделать это графически с помощью инструментов графического интерфейса пользователя (GUI или ГИП), имеющихся в *Fuzzy Logic Toolbox*. Имеются пять инструментов GUI для создания, редактирования и наблюдения фаззи-систем в среде *Fuzzy Logic Toolbox* (рисунок 5). Это *Fuzzy Inference System Editor* или *FIS Editor* – Редактор фаззи-инференционной системы, *Membership Function Editor* - Редактор функций принадлежности, *Rule Editor* - Редактор правил, *Rule Viewer* - Просмотр правил и *Surface Viewer* - Просмотр поверхности (пространства управления). Эти инструменты GUI динамически связаны между собой: используя один из них в *FIS* и производя изменения, можно затем увидеть их действия в других инструментах GUI. В дополнение к этим пяти инструментам *Fuzzy Logic Toolbox* включает в себя графический редактор GUI *ANFIS*, который используется для создания и анализа Sugeno-типа адаптивных нейросистем с нечеткой логикой – adaptive neuro-FIS (*ANFIS*).

FIS редактор решает проблемы верхнего уровня системы: сколько входных и выходных переменных и каковы их имена?

Fuzzy Logic Toolbox не имеет ограничений по числу входов фаззи-системы – все зависит от доступной памяти компьютера.

Рисунок 5 – Структура фаззи-инференционной системы – FIS

Membership Function Editor используется для того, чтобы определить виды всех функций принадлежности, связанных с каждой переменной.

Rule Editor используется для редактирования списка правил, который определяет функционирование фаззи-системы.

Rule Viewer и *Surface Viewer* используются только для просмотра результатов, но не редактирования фаззи-системы. *Rule Viewer* показывает диаграмму функционирования. Используя его в качестве диагностического средства, можно увидеть, например, какие правила активны, или как индивидуальные функции принадлежности влияют на результаты. *Surface Viewer* используется для показа зависимости выхода от одного или двух из входов системы, то есть показывается диаграмма поверхности пространства управления системы. Все пять инструментов GUI могут взаимодействовать и обмениваться информацией между собой. Любой из них может считывать и сохранять данные в рабочем пространстве *MATLAB* и на жестком диске компьютера.

Практическая часть

Разработаем проект базы знаний в следующей предметной области - классификация операционных усилителей.

Составление семантической сети и фреймов

Шаг 1. Составим для операционных усилителей семантическую сеть и фреймы. (Рисунки 6 и 7).

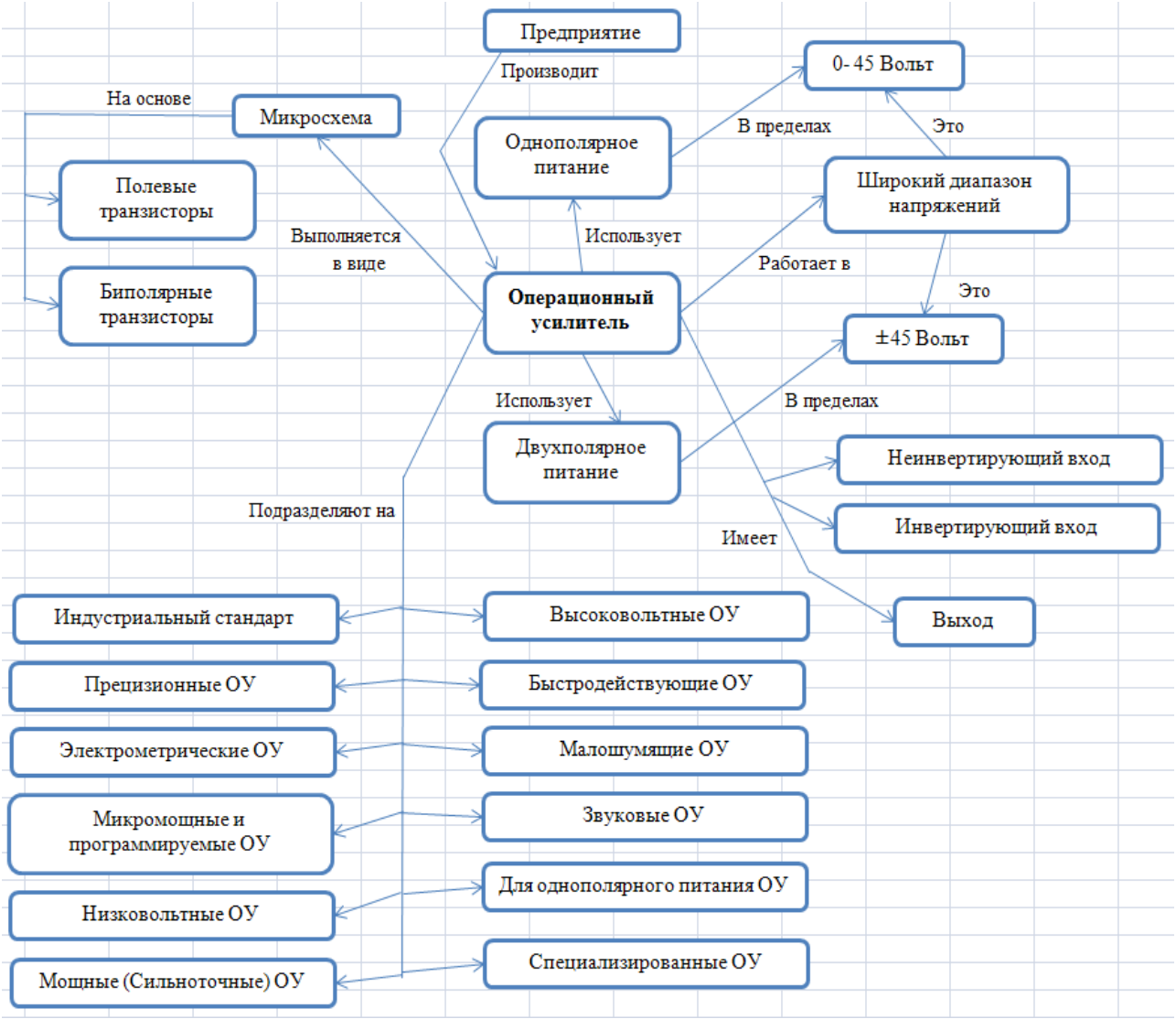

Рисунок 6 – Семантическая сеть

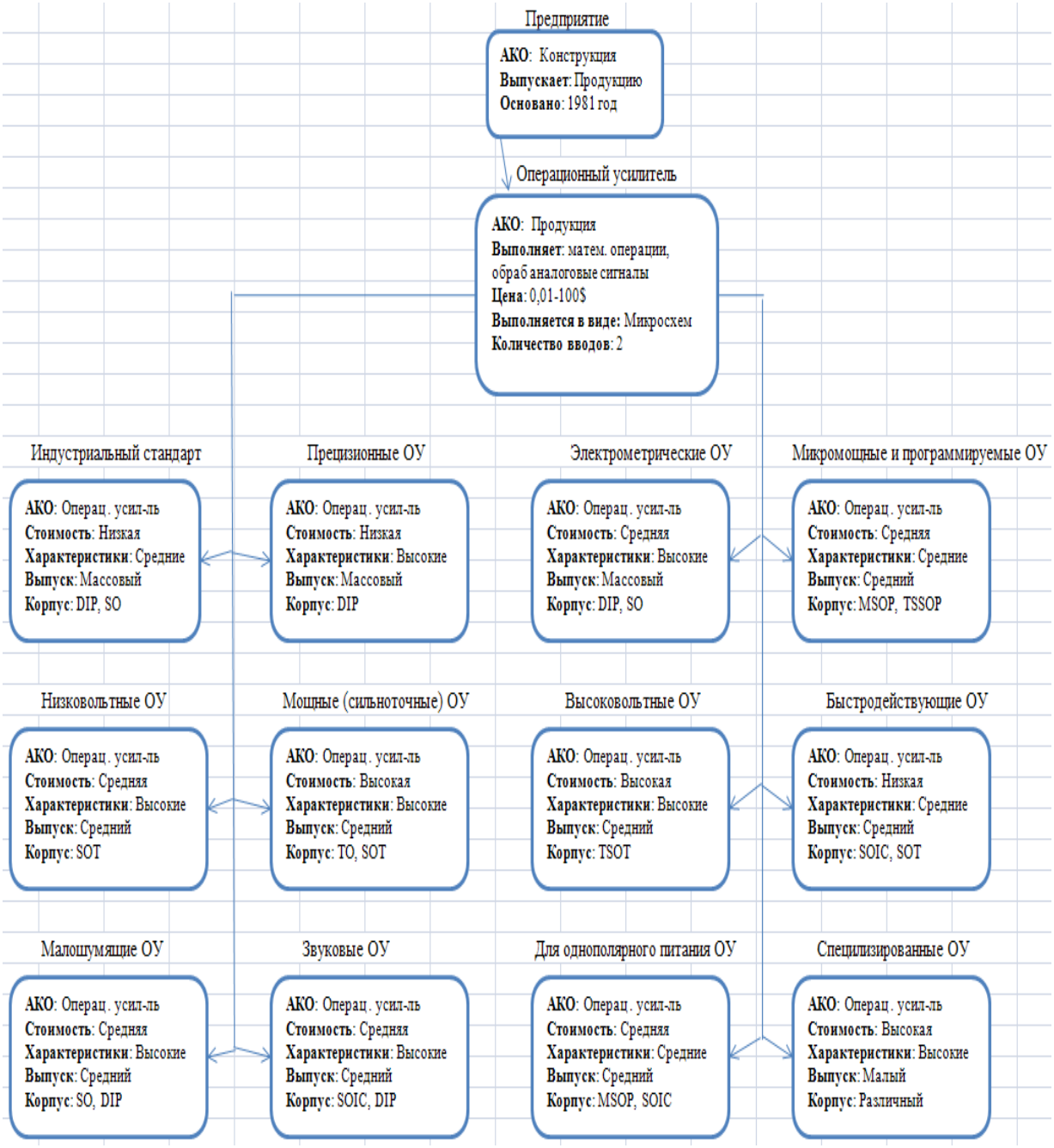

Рисунок 7 – Фреймы

Составление продукционных правил

Пусть необходимо подобрать операционный усилитель по области применения.

Определим:

Лингвистические переменные: Ток питания (I_p), напряжение питания (U_p), коэффициент усиления (K_u), полоса единичного пропускного сигнала (F), выходной ток (I_v);

Термы для переменной I_p: от 0 до 25 мкА, более 25 мкА;

Термы для переменной U_p: менее 20 В, более 20 В;

Термы для переменной K_u: меньше 3000, больше 3000;

Термы для переменной F: менее 20 МГц, более 20 МГц;

Термы для переменной I_v: менее 40 мА, более 40 мА.

Составим продукционные правила. Для этого пользователь должен ответить на 5 вопросов, на основе которых и будет выбран определенный вид операционного усилителя. Для удобства введем базу знаний в программный продукт Microsoft Excel, следовательно, при ответе на вопросы в определенном порядке: нет, да, нет, да, да – экспертная система должна выдать нам результат « Прецизионные ОУ» со 100% вероятностью (рисунок 8).

		Iпит <= 25 мкА?	Uпит max <= 20 В?	Ку<= 3000?	F<= 20 МГц?	Iвых<= 40 мА?
№	Классификация ОУ	Iпит, мкА	Uпит max, В	Ку	F, МГц	Iвых, мА
1	Индустриальный стандарт	1400-1700	8	2000	1,3 - 4	20 - 40
2	Прецизионные ОУ	350-500	7	16000	16	16
3	Электрометрические	1 до 25	18	4000	1	5 до 12
4	Микромощн и программир	1 до 25	23,6	75	0,075	1
5	Низковольтные	1 до 25	4	200	0,2	2 до 5
6	мощные(сильноточные)	10 до 25	18	2000	2	1000-1500
7	Высоковольтные	3700-8000	20	50000	>50	>70
8	Быстродействующие	725-3500	6,5	20000	20	80-90
9	Малошумящие	2500-4000	6	90000	85-95	40
10	Звуковые	1	18	3500	3,5	50
11	Для 1-полярного питания	15	36	3200	0,02	4,5
12	Специализированные	5700	44	8000	8	42
№	Классификация ОУ	Iпит, мкА	Uпит max, В	Ку	F, МГц	Iвых, мА
1	Индустриальный стандарт	нет	да	да	да	да
2	Прецизионные ОУ	нет	да	нет	да	да
3	Электрометрические	да	да	нет	да	да
4	Микромощн и программир	да	нет	да	да	да
5	Низковольтные	да	да	да	да	да
6	мощные(сильноточные)	да	да	да	да	нет
7	Высоковольтные	нет	да	нет	нет	нет
8	Быстродействующие	нет	да	нет	да	нет
9	Малошумящие	нет	да	нет	нет	да
10	Звуковые	да	да	нет	да	нет
11	Для 1-полярного питания	да	нет	нет	да	да
12	Специализированные	нет	нет	нет	да	нет

Рисунок 8 – База знаний

На основе базы знаний и предложенных вопросов мы можем составить продукционные правила. Для удобства термы переменной I_p: от 0 до 25 мкА, более 25 мкА обозначили как Yes и No соответственно. Аналогично поступили и с остальными переменными.

Продукционные правила:

1) Если I_p =No и U_p=Yes и K_u=Yes и F=Yes и I_v =Yes, то P100=industr standart;

2) Если I_p =No и U_p=Yes и K_u=No и F=Yes и I_v =Yes, то P100=presizionyi;

3) Если I_p =Yes и U_p=Yes и K_u=No и F=Yes и I_v =Yes, то P100=ilektrometrich;
4) Если I_p =Yes и U_p=No и K_u=Yes и F=Yes и I_v =Yes, то P100=mikromosh i programmir;
5) Если I_p =Yes и U_p=Yes и K_u=Yes и F=Yes и I_v =Yes, то P100=nizkovoltnyi;
6) Если I_p =Yes и U_p=Yes и K_u=Yes и F=Yes и I_v =No, то P100=moshnye;
7) Если I_p =No и U_p=Yes и K_u=No и F=No и I_v =No, то P100=vysokovoltnye;
8) Если I_p =No и U_p=Yes и K_u=No и F=Yes и I_v =No, то P100=bystrodeistvuysh;
9) Если I_p = No и U_p=Yes и K_u=No и F=No и I_v =Yes, то P100=maloshumyashie;
10) Если I_p =Yes и U_p =Yes и K_u=No и F=Yes и I_v =No, то P100=zvurovye;
11) Если I_p =Yes и U_p=No и K_u=No и F=Yes и I_v =Yes, то P100=dly 1polyar;
12) Если I_p =No и U_p=No и K_u=No и F=Yes и I_v =No, то P100=spesializirov.

Для реализации лингвистической переменной необходимо определить точные физические значения ее термов. Пусть, например, переменная U_p может принимать любое значение из диапазона от 0 до 44 В. Согласно положениям теории нечетких множеств, каждому значению напряжения из диапазона в 44 В может быть поставлено в соответствие некоторое число, от нуля до единицы, которое определяет *степень принадлежности* данного физического значения тока (допустим, 30 В) к тому или иному терму лингвистической переменной U_p.

Проектирование экспертной системы в MatLab

Создадим в MatLab экспертную систему:

Для этого необходимо ввести в окне команду *Fuzzy* или же проделать операции, показанные на рисунке 9.

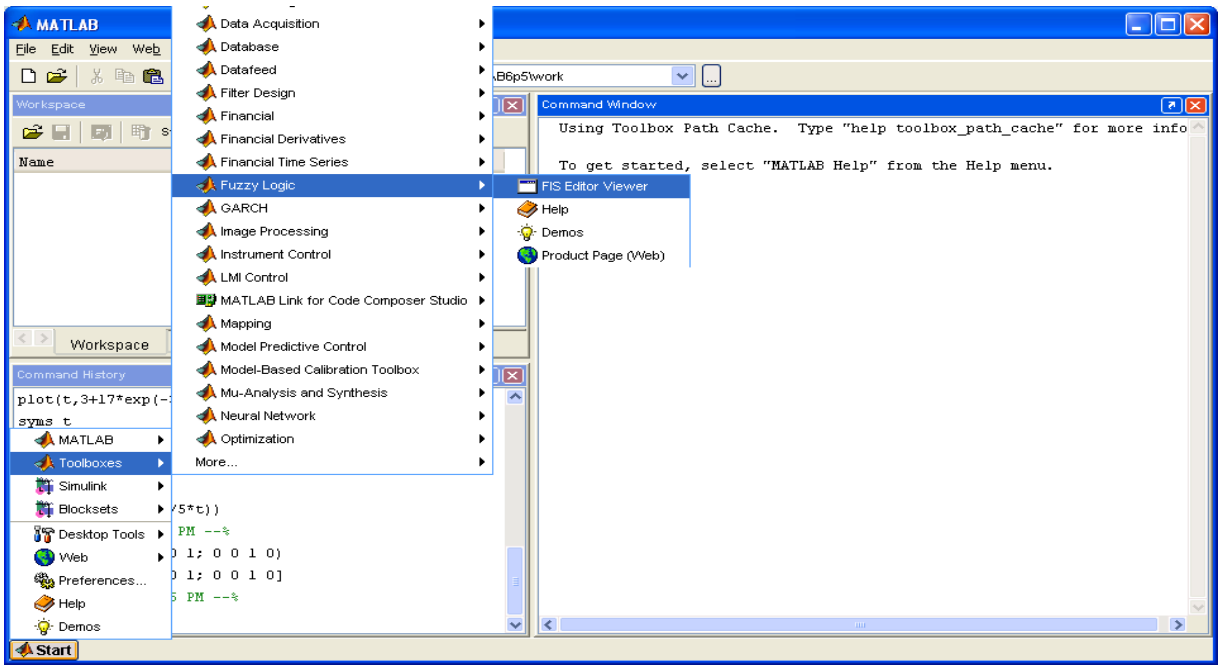

Рисунок 9 – Инструкции по запуску Fuzzy Logic Toolbox

В открывшемся окне выбираем Edit⇒FIS properties. Открывается окно FIS Editor:

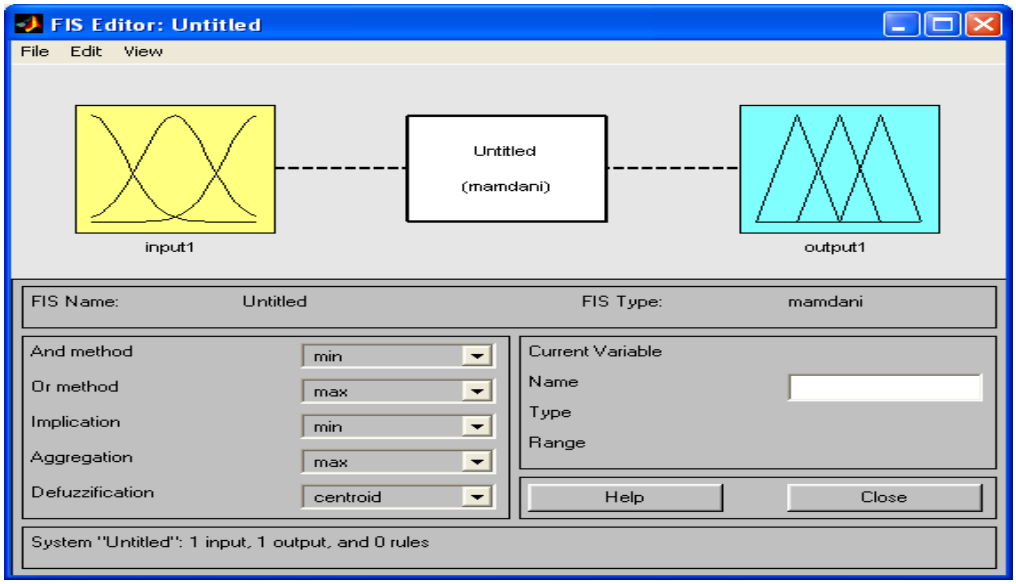

Рисунок 10 – Рабочее окно FIS Editor

Задаем 5 входных переменных I_p, U_p, K_u, F, I_v (Edit⇒Add Variable⇒Input) и одну выходную переменную p100 (Edit⇒Add Variable⇒Output). Первоначально названия входных и выходных переменных будут стандартными как на рисунке 10, поэтому необходимо для удобства переименовать их. Окно принимает вид (рисунок 11):

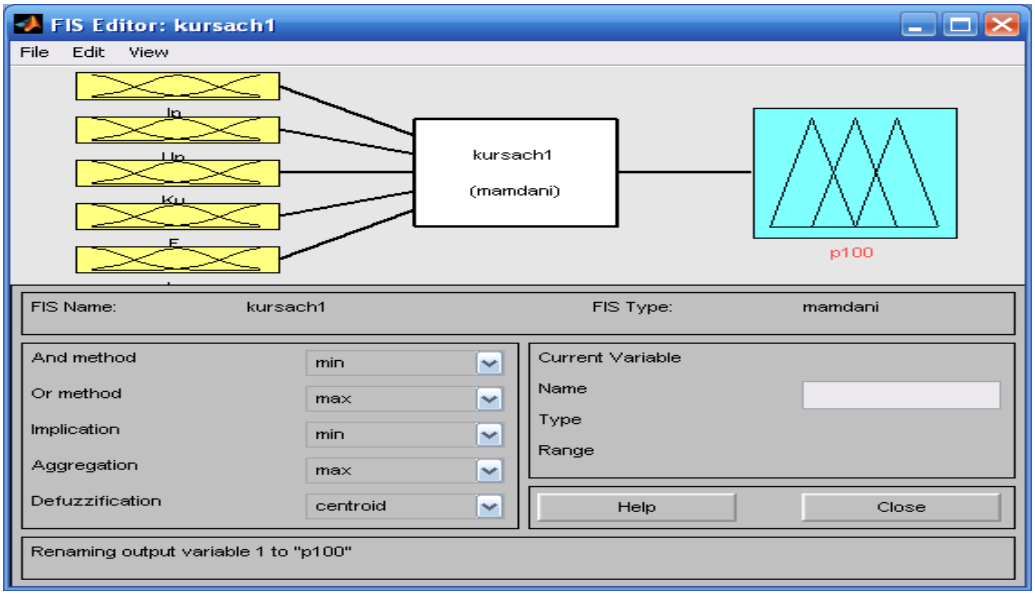

Рисунок 11 – Рабочее окно FIS Editor с переименованными переменными

Для переменной U_p, щелкнув на соответствующей иконке, задаем две функции принадлежности zmf и smf (Yes и No), так как они наиболее подходят для наших термов.

Напомним еще раз, что программу-редактор функций принадлежности можно открыть тремя способами:

- через пункт меню View -> Edit membership functions,
- двойным щелчком на значке, отображающем соответствующую переменную,
- нажатием клавиш Ctrl+2.

Откроем программу-редактор функций принадлежности 2 способом (рисунок 12).

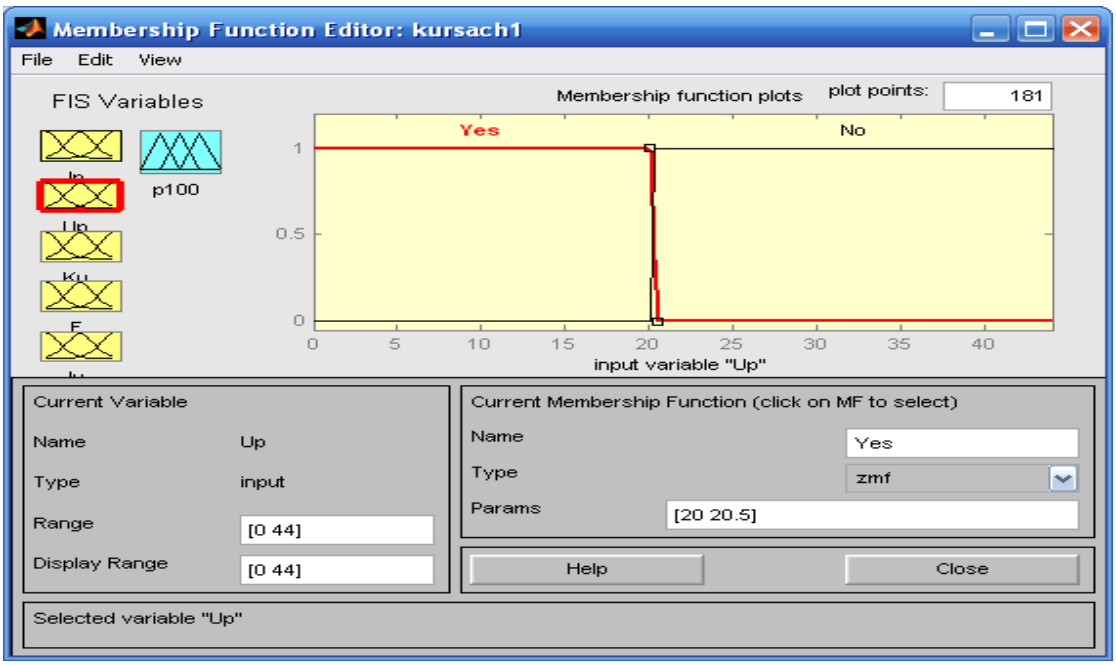

Рисунок 12 – Окно настроек функции принадлежности Yes

В параметре Range необходимо указывать интервал вводимых значений, в параметре Name даем название нашему интервалу (мы его уже указали как один из термов - менее 20 В); в параметре Params необходимо указать вероятности для функции принадлежности.

Аналогично производим настройки для второго интервала (No) (рисунок 13):

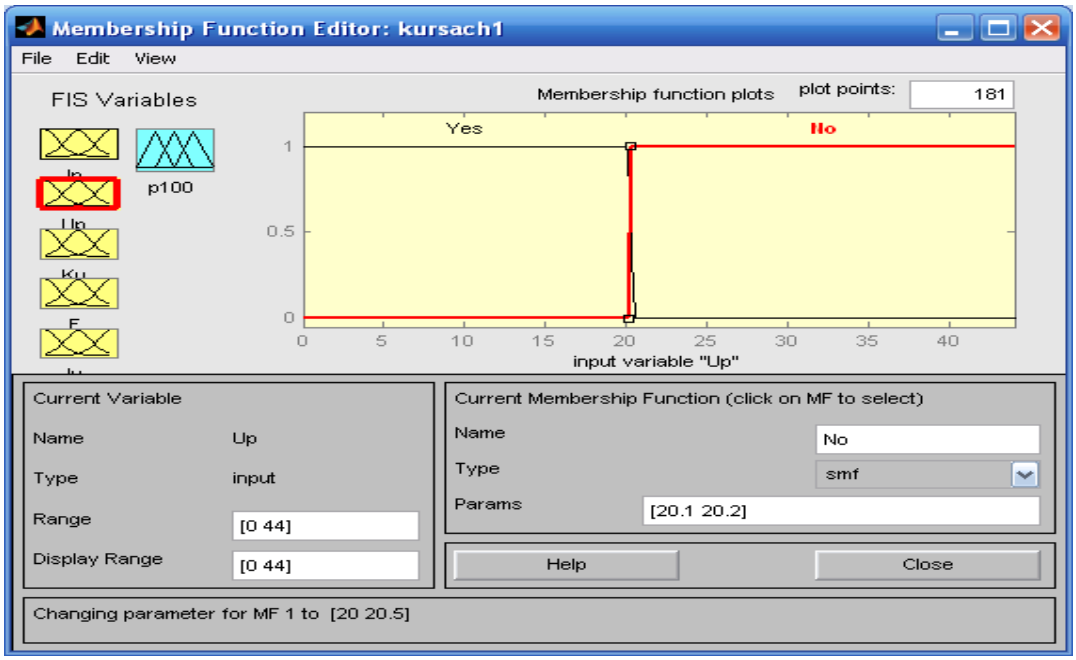

Рисунок 13 – Окно настроек функции принадлежности No

По такому же принципу необходимо настроить все 5 входных переменных.

Далее настраиваем выходную переменную (рисунок 14):

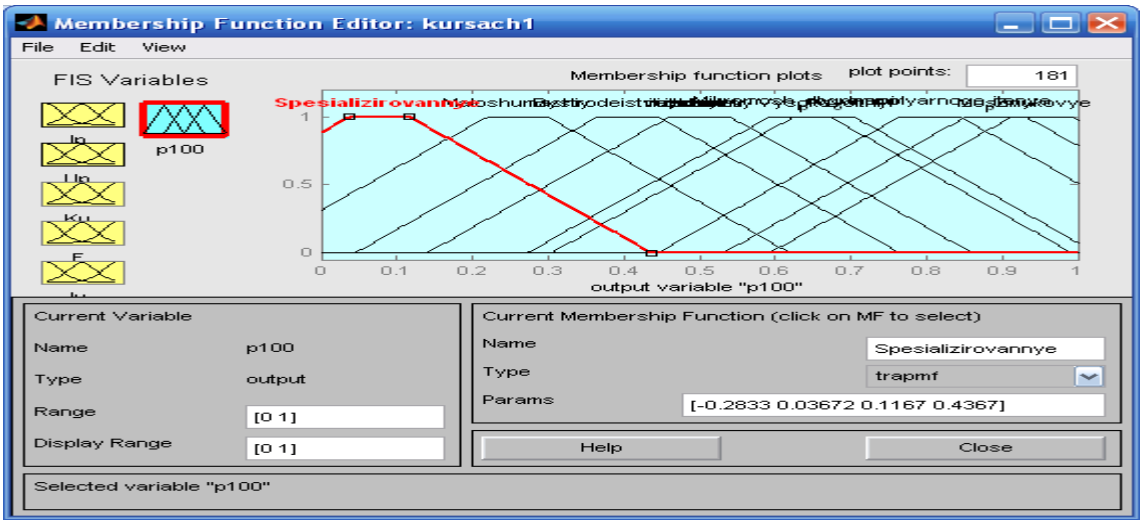

Рисунок 14 – Окно настроек функций принадлежности для выходной переменной

По рисунку видно, что в выходной переменной необходимо настроить 12 функций принадлежности (количество функций равно 12, т. к. у нас 12 видов операционных усилителей).

После настройки всех функций окно выглядит так (рисунок 15):

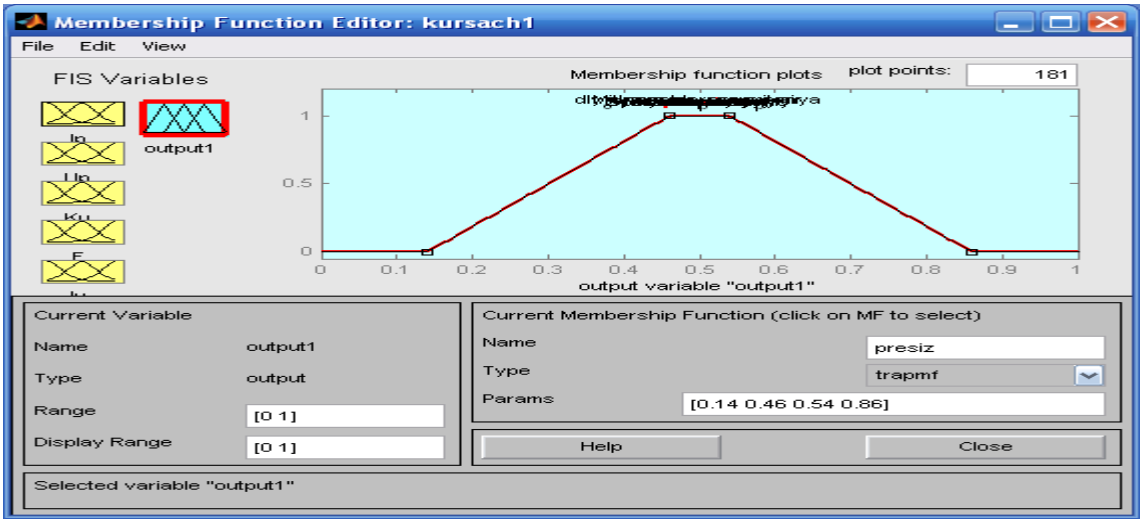

Рисунок 15 – Окно с выбранными функциями принадлежности

То есть мы выбрали одинаковую вероятность для всех функций принадлежности.

Далее необходимо вводить составленные правила. Для этого выберем пункт-меню View⇒Edit rules (рисунок 16).

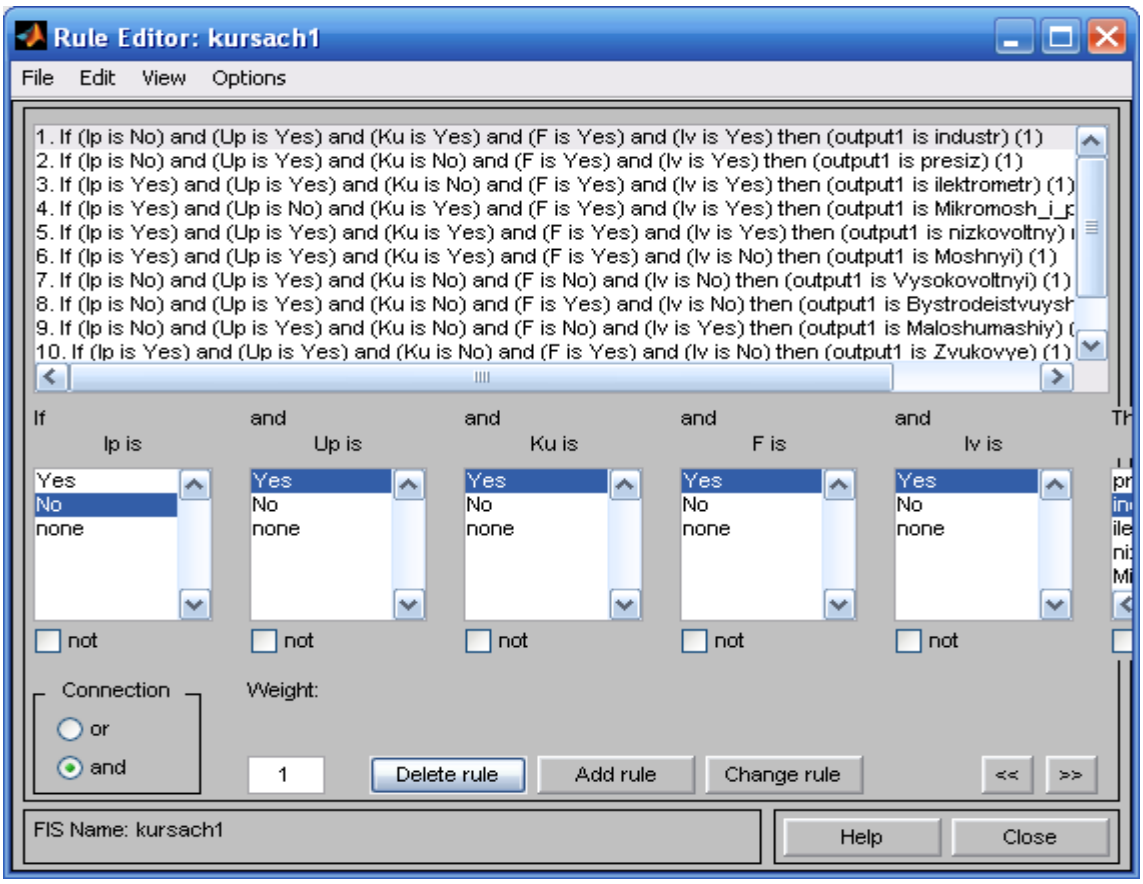

Рисунок 16 – Окно для ввода продукционных правил

Необходимо ввести все правила, выбирая ответы из представленных вариантов.

Теперь самое время проверить систему в действии. Откроем (через пункт меню View ⇒ View rules) окно просмотра правил и установим значения переменных (рисунок 17).

Переменные необходимо вводить в окно Input через пробел.

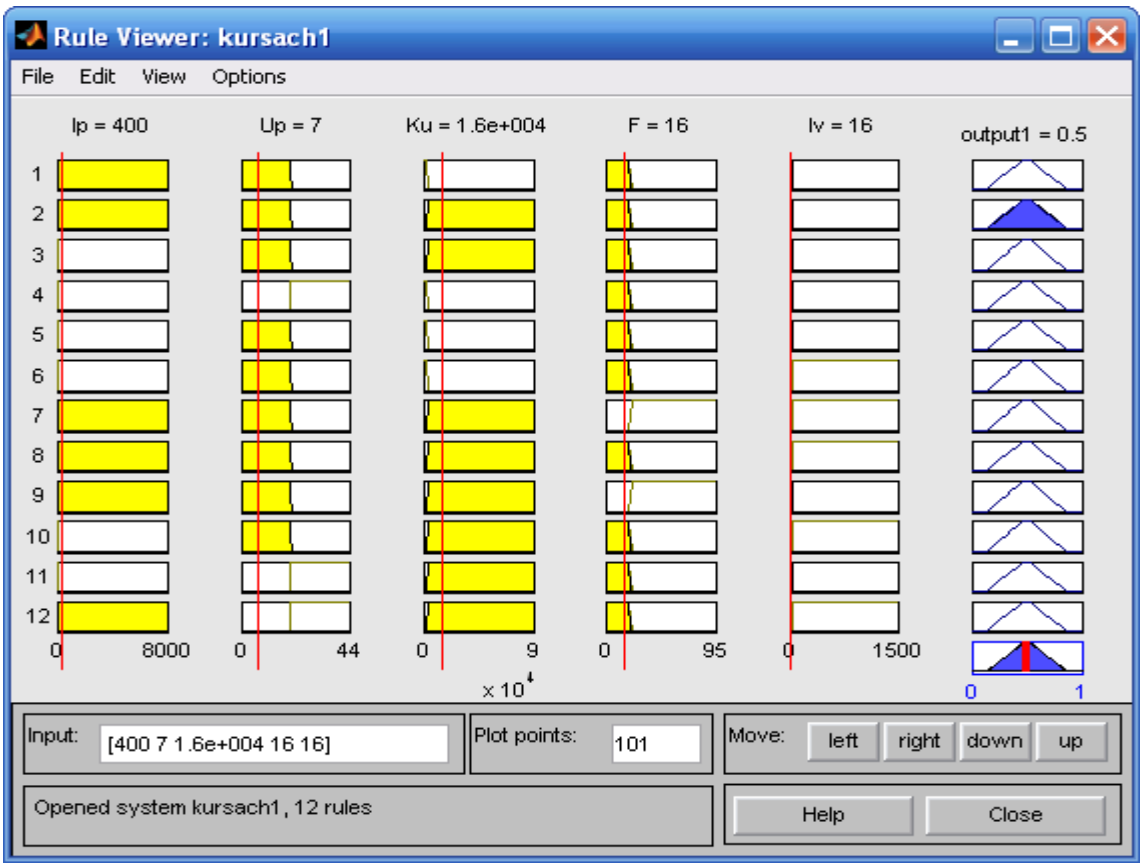

Рисунок 17 – Окно для просмотра результатов

В результате экспертная система выдает нам наиболее подходящий операционный усилитель, основываясь на введенных данных (в данном случае система выдает выполнение 2 правила, т. е. при данных значениях нам подходит Прецизионный ОУ).

Следует упомянуть, что данная система будет выдавать правильные решения лишь в том случае, если введены все переменные и они соответствуют каком-либо виду операционного усилителя. Для того чтобы система работала при любых введенных параметрах и даже при отсутствии некоторых параметров, необходимо добавлять условия для нечеткости, то есть увеличить в разы количество продукционных правил.

На примере данной экспертной системы для того, чтобы система функционировала даже при 1введенном параметре необходимо 62 правила.

И экспертная система будет выглядеть следующим образом (рисунок 18):

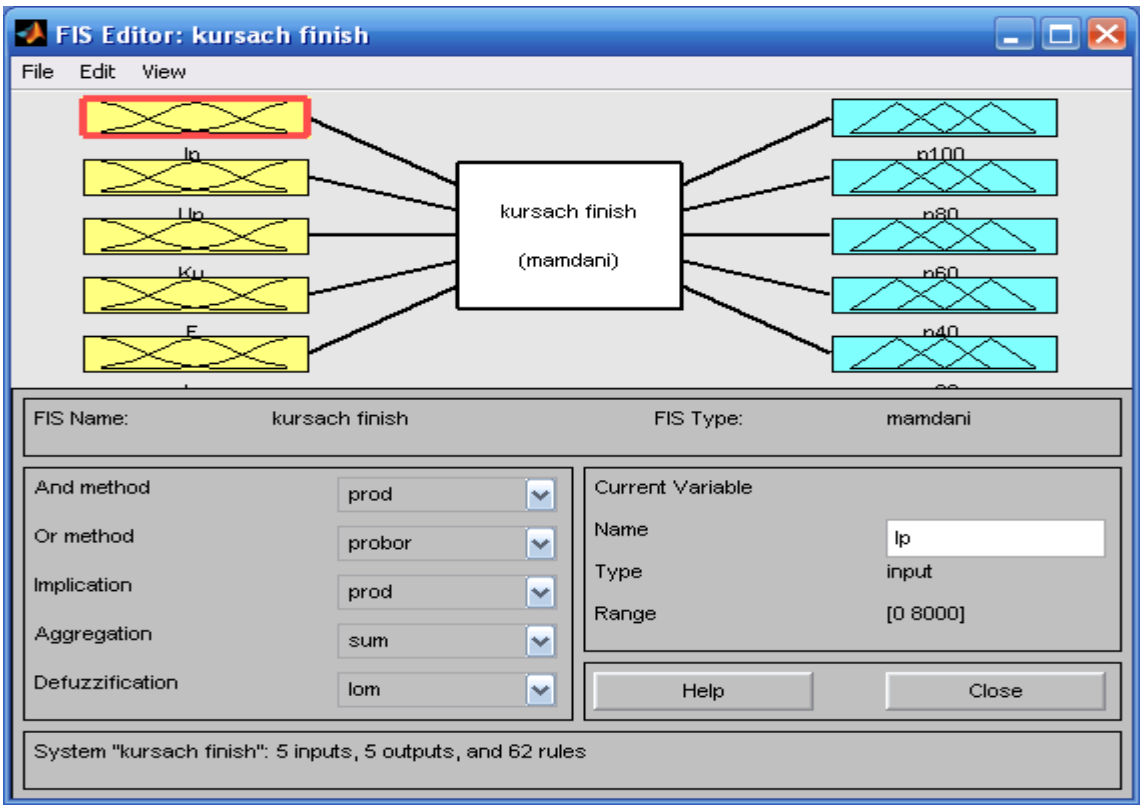

Рисунок 18 – Рабочее окно FIS Editor с добавлением выходных переменных

Как видно по рисунку мы добавили дополнительные выходы, т.е. теперь система будет нам выводить ответы со следующими вероятностями: 100, 80, 60, 40, 20 %. Для каждой выходной переменной необходимо также настроить функции принадлежности.

Необходимо дописать недостающие правила (так как в условиях нечеткости могут возникнуть коллизии). Для удобства составления правил обозначим виды операционных усилителей цифрами от 1 до 12 (рисунок 8).

Продукционные правила:

1) Если I_p =No и U_p=Yes и K_u=Yes и F=Yes и I_v = Yes, то P100 = 1;
2) Если I_p = No и U_p = Yes и K_u = No и F = Yes и I_v = Yes, то P100 = 2;
3) Если I_p = Yes и U_p = Yes и K_u = No и F = Yes и I_v = Yes, то P100 = 3;
4) Если I_p = Yes и U_p = No и K_u = Yes и F = Yes и I_v = Yes, то P100 = 4;
5) Если I_p = Yes и U_p = Yes и K_u = Yes и F = Yes и I_v = Yes, то P100 = 5;
6) Если I_p = Yes и U_p = Yes и K_u = Yes и F = Yes и I_v = No, то P100 = 6;
7) Если I_p = No и U_p = Yes и K_u = No и F = No и I_v = No, то P100 = 7;
8) Если I_p = No и U_p = Yes и K_u = No и F = Yes и I_v = No, то P100 = 8;
9) Если I_p = No и U_p = Yes и K_u = No и F = No и I_v = Yes, то P100 = 9;
10) Если I_p = Yes и U_p = Yes и K_u = No и F = Yes и I_v = No, то P100 = 10;

11) Если I_p = Yes и U_p = No и K_u = No и F = Yes и I_v = Yes, то P100 = 11;
12) Если I_p = No и U_p = No и K_u = No и F = Yes и I_v = No, то P100 = 12;
13) Если I_p = Yes и U_p = Yes и K_u = Yes и F = No и I_v = Yes, то P80 = 5;
14) Если I_p = Yes и U_p = Yes и K_u = Yes и F = No и I_v = No, то P80 = 6;
15) Если I_p = Yes и U_p = Yes и K_u = No и F = No и I_v = Yes, то P80 = 3;
16) Если I_p = Yes и U_p = Yes и K_u = No и F = No и I_v = No, то P80 = 10;
17) Если I_p = Yes и U_p = No и K_u = Yes и F = Yes и I_v = No, то P80 = 4;
18) Если I_p = Yes и U_p = No и K_u = Yes и F = No и I_v = Yes, то P80 = 4;
19) Если I_p = Yes и U_p = No и K_u = No и F = Yes и I_v = No, то P80 = 11;
20) Если I_p = Yes и U_p = No и K_u = No и F = No и I_v = Yes, то P80 = 11;
21) Если I_p = No и U_p = Yes и K_u = Yes и F = Yes и I_v = No, то P80 = 1;
22) Если I_p = No и U_p = Yes и K_u = Yes и F = No и I_v = Yes, то P80 = 1;
23) Если I_p = No и U_p = No и K_u = Yes и F = Yes и I_v = No, то P80 = 12;
24) Если I_p = No и U_p = No и K_u = No и F = Yes и I_v = Yes, то P80 = 12;
25) Если I_p = No и U_p = No и K_u = No и F = No и I_v = No, то P80 = 12;
26) Если I_p = Yes и U_p = Yes и K_u = Yes и F = Yes, то P80 = 5, 6;
27) Если I_p = Yes и U_p = Yes и K_u = No и F = Yes, то P80 = 3, 10;
28) Если I_p = Yes и U_p = No и K_u = Yes и F = Yes, то P80 = 4;
29) Если I_p = Yes и U_p = No и K_u = No и F = Yes, то P80 = 11;
30) Если I_p = No и U_p = Yes и K_u = Yes и F = Yes, то P80 = 1;
31) Если I_p = No и U_p = Yes и K_u = No и F = Yes, то P80 = 2, 8;
32) Если I_p = No и U_p = Yes и K_u = No и F = No, то P80 = 7, 9;
33) Если I_p = No и U_p = No и K_u = No и F = Yes, то P80 = 12;
34) Если I_p = Yes и U_p = Yes и K_u = Yes и F = No, то P60 = 5, 6;
35) Если I_p = Yes и U_p = Yes и K_u = No и F = No, то P60 = 3, 10;
36) Если I_p = Yes и U_p = No и K_u = Yes и F = No, то P60 = 4;
37) Если I_p = Yes и U_p = No и K_u = No и F = No, то P60 = 11;
38) Если I_p = No и U_p = Yes и K_u = Yes и F = No, то P60 = 1;
39) Если I_p = No и U_p = No и K_u = No и F = No, то P60 = 12;
40) Если I_p = Yes и U_p = No и K_u = Yes и F = No и I_v = No, то P60 = 4;
41) Если I_p = Yes и U_p = No и K_u = No и F = No и I_v = No, то P60 = 11;
42) Если I_p = No и U_p = Yes и K_u = Yes и F = No и I_v = No, то P60 = 1;
43) Если I_p = No и U_p = No и K_u = Yes и F = Yes и I_v = Yes, то P60 = 12;
44) Если I_p = No и U_p = No и K_u = Yes и F = No и I_v = No, то P60 = 12;
45) Если I_p = No и U_p = No и K_u = No и F = No и I_v = No, то P60 = 12;
46) Если I_p = Yes и U_p = Yes и K_u = Yes, то P60 = 5, 6;
47) Если I_p = Yes и U_p = Yes и K_u = No, то P60 = 3, 10;
48) Если I_p = Yes и U_p = No и K_u = Yes, то P60 = 4;
49) Если I_p = Yes и U_p = No и K_u = No, то P60 = 11;

50) Если I_p = No и U_p = Yes и K_u = Yes, то P60 = 1;
51) Если I_p = No и U_p = Yes и K_u = No, то P60 = 2, 7, 8, 9;
52) Если I_p = No и U_p = No и K_u = No, то P60 = 12;
53) Если I_p = No и U_p = No и K_u = Yes и F = No и I_v = Yes, то P40 = 12;
54) Если I_p = No и U_p = No и K_u = Yes и F = Yes, то P40 = 12;
55) Если I_p = No и U_p = No и K_u = Yes и F = No, то P40 = 12;
56) Если I_p = No и U_p = No и K_u = Yes, то P40 = 12;
57) Если I_p = Yes и U_p = Yes, то P40 = 3, 4, 5, 6, 10;
58) Если I_p = Yes и U_p = No, то P40 = 4, 11;
59) Если I_p = No и U_p = Yes, то P40 = 1, 2, 7, 8, 9;
60) Если I_p = No и U_p = No, то P40 = 12;
61) Если I_p = Yes, то P20 = 3, 4, 5, 6, 10, 11;
62) Если I_p = No, то P20 = 1, 2, 7, 8, 9, 12.

Далее необходимо добавить недостающие правила в программу. В результате все должно быть как на рисунке 19.

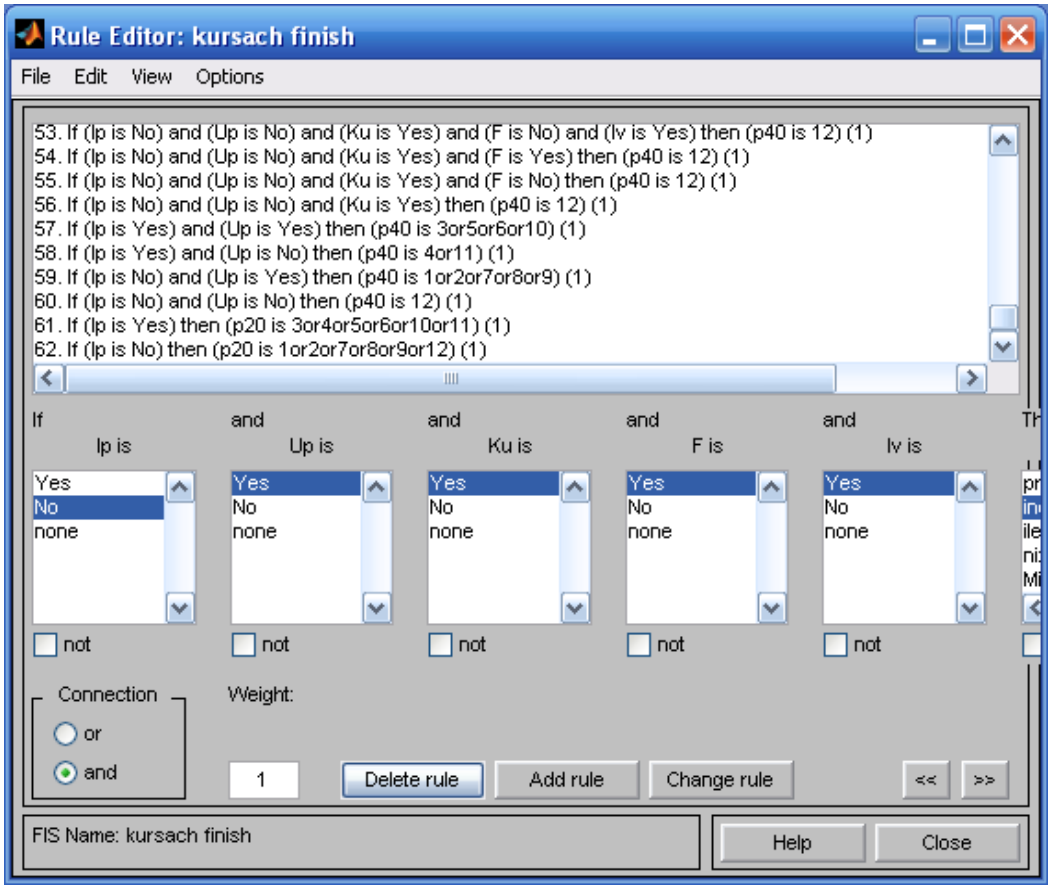

Рисунок 19 – Окно для ввода продукционных правил

Теперь самое время проверить систему в действии. Откроем (через пункт меню View ⇒ View rules) окно просмотра правил и установим значения переменных (рисунок 20).

Переменные необходимо вводить в окно Input через пробел.

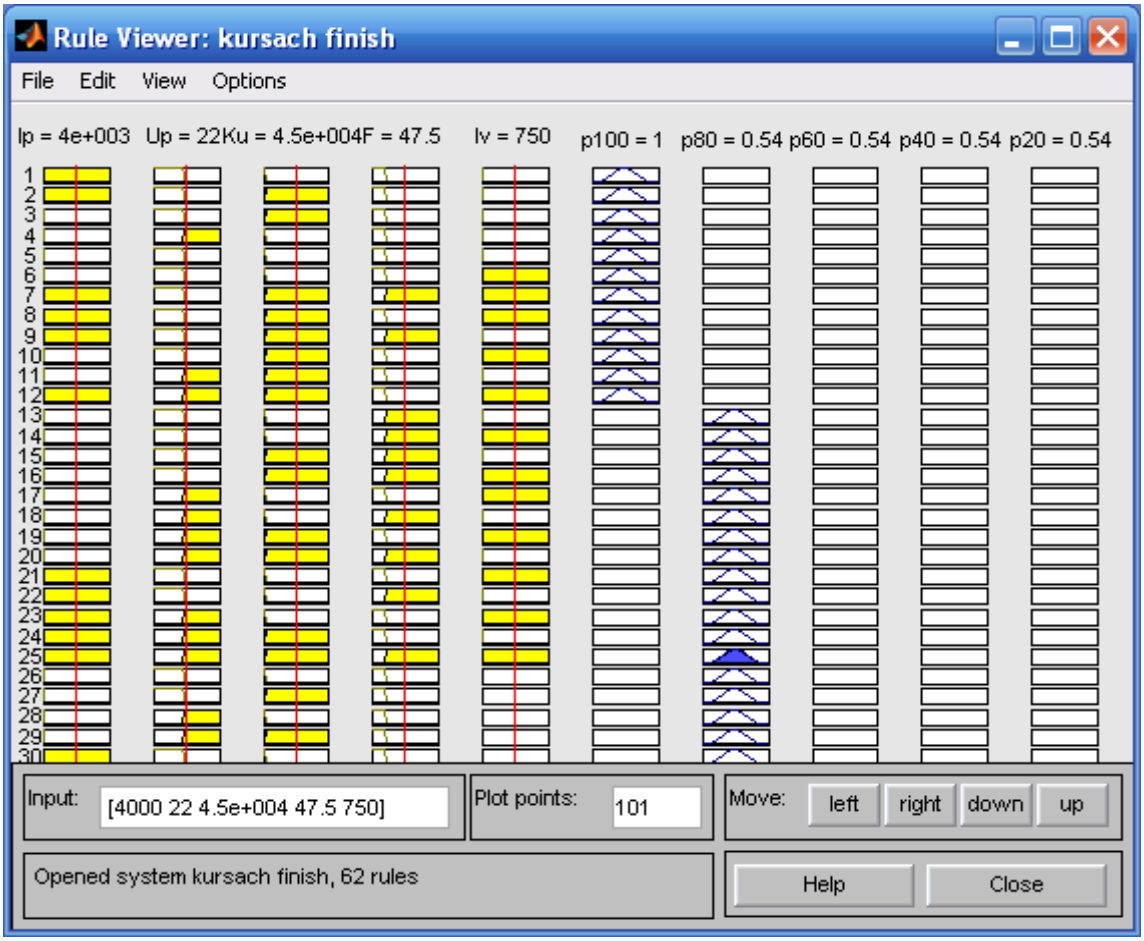

Рисунок 20 – Окно для просмотра результатов

Как видим при данных значениях параметров выполняется условие 25, которое означает, что с вероятностью 80 % нам подходит специализированный ОУ.

Варианты заданий

Разработать проект базы знаний, содержащей 50–60 правил, в следующей предметной области:

1. Диагностика аппаратных неисправностей ЭВМ
2. Диагностика программных неисправностей ЭВМ
3. Классификация датчиков температуры
4. Классификация датчиков влажности
5. Классификация датчиков давления

6. Диагностика аппаратных неисправностей ЛВС
7. Диагностика программных неисправностей ЛВС
8. Классификация сред передачи электромагнитных сигналов
9. Классификация сетевых протоколов
10. Классификация модемов
11. Классификация SCADA–систем
12. Классификация нейронных сетей
13. Выбор пакета прикладных программ для решения задачи на ЭВМ
14. Выбор программного обеспечения для компьютерной сети
15. Выбор оборудования для компьютерной сети
16. Диагностика загруженности ресурсов ЭВМ
17. Классификация микроконтроллеров семейства ATXMEGA
18. Классификация заболеваний органов дыхания
19. Классификация психических заболевании
20. Классификация сердечно-сосудистых заболеваний
21. Классификация медицинского оборудования
22. Классификация микроконтроллеров семейства AVR Tiny
23. Классификация средств разработки контроллеров AVR

Контрольные вопросы

1. Перечислите различия баз данных и знаний.
2. Перечислите этапы проектирования БЗ
3. Какие формы сообщений используются в вашей БЗ
4. Приведите пример поиска решения с использованием правил Вашей БЗ
5. Архитектура экспертных систем
6. Назначение компонент экспертных систем
7. Режимы работы экспертных систем
8. Технология построения экспертных систем
9. Условия применимости экспертных систем
10. Типы экспертных систем в зависимости от степени завершенности и особенностей использования: демонстрационные, исследовательские, промышленные, коммерческие
11. Этапы построения экспертных систем: идентификация, концептуализация, формализация, реализация, тестирование

Литература

1 Андрейчиков, А.В. Интеллектуальные информационные системы/А.В. Андрейченко, о.Н. Анрейчикова. - М. Наука, 2004. – 424 с.

2.Асаи, К. Прикладные нечеткие системы/ К. Ассаи и др./ пер. сяпон.; под. ред. Тэрано Т., Асаи К. Сугэно М. – М.: Мир, 1993. –368с.

3. База знаний – Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/%C1%E0%E7%E0_%E7%ED%E0%ED%E8%E9

4. Базы данных [Электронный ресурс] – Режим доступа: http://li.romab.ru/intro_db_6.html

5. Дьяконов В. Математические пакеты расширения MATLAB: специальный справочник/ В. Дьяконов, В.Круглов. – СПб.: Питер, 2001. – 480 с.

6. Муромцев, Д.И. Введение в технологию экспертных систем/Д.И. Муромцев. - СПб: СПб ГУ ИТМО, 2005. – 93 с.

7. Нейлор, К. Как построить свою экспертную систему/ К. Нейлор: Пер. с англ. – М.: Энергоатомиздат, 1991. – 286 с.

8.Нечеткая логика в системах управления – журнал «Компьютерра» [Электронный ресурс] – Режим доступа: http://offline.computerra.ru/2001/415/13052/

9.Попов, Э.В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ/ Э.В. Попов. - М.: Наука, 1987. – 288 с.

10.Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского/ Д. Рутковская, М. Пилинский, Л. Рутковский. - М.: Горячая линия – Телеком, 2007. – 452 с.

11 Терехов, В.А. Нейросетевые системы управления: учебное пособие для вузов/ В.А. Терехов. – М.: Высш. шк., 2002. – 183 с.

12. Ясницкий, Л.Н. Введение в искусственный интеллект: учебное пособие/ Л.Н. Ясницкий. М.: Академия, 2008. – 176 с.