ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Кафедра информационных технологий и моделирования

Л.Ю. Мельник

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Методические указания
по выполнению лабораторно-практического цикла
для студентов очной, очно-заочной и заочной форм обучения
по направлению 080801 «Прикладная информатика»

Екатеринбург
2009

Печатается по рекомендации методической комиссии ФЭУ.
Протокол № 20 от 21 сентября 2007г.

Рецензент – старший преподаватель Г.Л. Нохрина

Редактор Н.А. Майер
Оператор Г.И. Романова

Подписано в печать 06.02.09		Поз. 114
Плоская печать	Формат 60×84 1/16	Тираж 50 экз.
Заказ №	Печ. л. 1,86	Цена 6 руб. 20 коп.

Редакционно-издательский отдел УГЛТУ
Отдел оперативной полиграфии УГЛТУ

ЛАБОРАТОРНАЯ РАБОТА № 1

Построение нечеткой аппроксимирующей системы для решения задачи интерполяции

Цель работы. Знакомство с графическим интерфейсом Fuzzy Logic Toolbox.

В состав программных средств Fuzzy Logic Toolbox входят следующие основные программы, позволяющие работать в режиме графического интерфейса:

- редактор нечеткой системы вывода Fuzzy Inference System Editor (FIS Editor или FIS-редактор) вместе со вспомогательными программами — редактором функций принадлежности (Membership Function Editor), редактором правил (Rule Editor), просмотрщиком правил (Rule Viewer) и просмотрщиком поверхности отклика (Surface Viewer);
- редактор гибридных систем (ANFIS Editor, ANFIS-редактор);
- программа нахождения центров кластеров (программа Clustering — кластеризация).

Набор данных программ предоставляет пользователю максимальные удобства для создания, редактирования и использования различных систем нечеткого вывода.

Задание к лабораторной работе 1

Командой (функцией) **Fuzzy** из режима командной строки запускается редактор нечеткой системы вывода. Вид открывающегося при этом окна приведен на рис. 1.

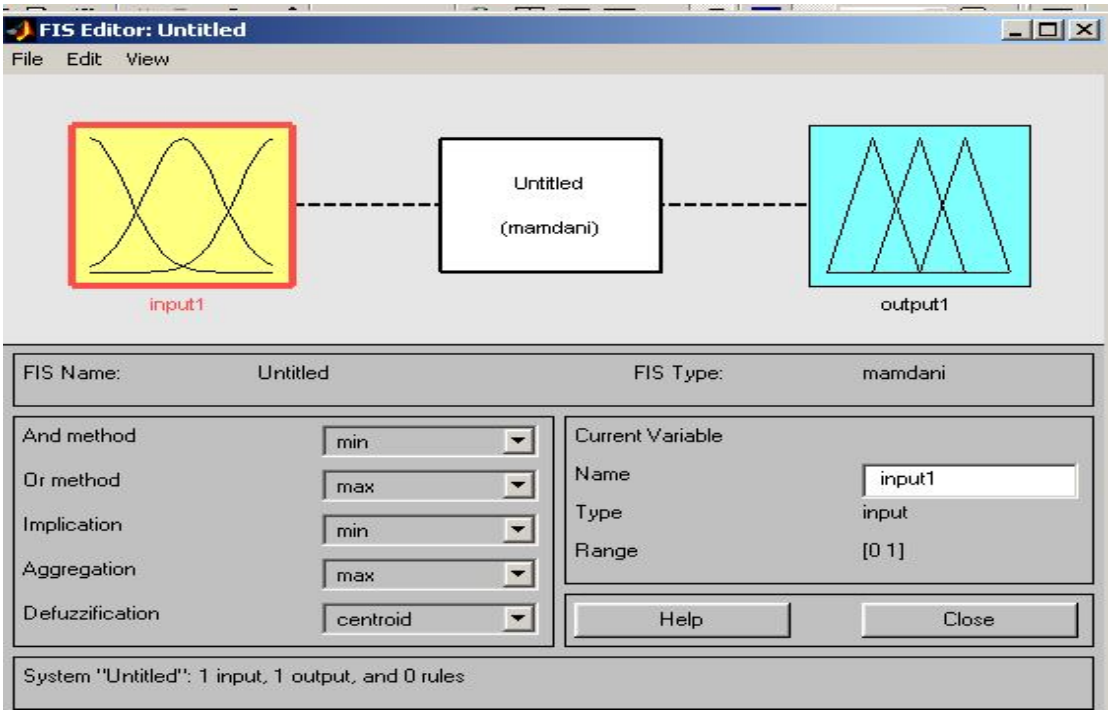


Рис. 1. Вид окна FIS Editor

Строка меню редактора содержит следующие позиции:

• **File** — работа с файлами моделей (их создание, сохранение, считывание и печать);

• **Edit** — операции редактирования (добавление и исключение входных и выходных переменных);

• **View** — переход к дополнительному инструментарию.

Попробуем сконструировать нечеткую систему, отображающую зависимость между переменными *x* и *y,* заданную с помощью табл. 1, используя алгоритм (**Sugeno**), на примере зависимости $y = x^2$.

Таблица 1

Значения x и y

x	-1	-0,6	0	0,4	1
y	1	0,36	0	0,16	1

1. В меню **File** выбираем команду New **Sugeno** FIS (Новая система типа **Sugeno**), должна появиться надпись Untitted 2 (**Sugeno**).

2. Щелкнем на блоке, озаглавленном **Input 1** (Вход 1). В правой части редактора в поле **Name** (Имя) вместо **Input 1** введем обозначение нашего аргумента ***x***. Можно щелкнуть в любом месте блока или нажатием клавиши Enter после ввода.

3. Дважды щелкнем на этом блоке. Перед нами откроется окно редактора функций принадлежности — Membership Function Editor. Откроем меню **Edit** данного редактора и выберем в нем команду **Add MFs** (Add Membership Functions — Добавить функции принадлежности). В диалоговом окне добавить недостающее количество функций принадлежности к входному сигналу к переменной ***x***. Выберем гауссовы функции принадлежности (gaussmf), их количество должно быть равным пяти — по числу значений аргумента в табл. 1. Подтвердим ввод информации нажатием кнопки ОК.

4. В поле **Range** (Диапазон) установим диапазон изменения *x* от –1 до 1, т. е. диапазон, соответствующий значениям табл. 1. Затем щелкнем левой кнопкой мыши где-нибудь в поле редактора (или нажмем клавишу **Enter**).

5. В графиках заданных нами функций принадлежности необходимо, чтобы ординаты максимумов этих функций совпадали с заданными значениями аргумента ***x***. Для этого подводим указатель к нужной кривой и щелкаем левой кнопкой мыши, она окрашивается в красный цвет, после чего с помощью мыши ее можно подвинуть в нужную сторону (более точную установку можно провести, изменяя числовые значения в поле **Params** (Параметры). Каждой функции принадлежности соответствуют два параметра, при этом первый определяет размах кривой, а второй — положение ее центра.

В поле **Name** можно изменить имя для выбранной кривой (завершая ввод каждого имени нажатием клавиши Enter) пяти кривым, например, так:

- самой левой — x1,
- следующей — x2,
- центральной — x3,
- следующей за ней справа — x4,
- самой правой — x5.

Нажмем кнопку **Close** и выйдем из редактора функций принадлежности, возвратившись в окно редактора нечеткой системы (**FIS Editor**).

6. Сделаем однократный щелчок на голубом квадрате (блоке), озаглавленном **Output l** (Выход 1). В поле **Name** заменим имя **Output l** на ***y*** (как в п. 2).

7. Дважды щелкнув на выделенном блоке, перейдем к редактору функций принадлежности. В меню **Edit** выберем команду **Add MFs.** Появляющееся затем диалоговое окно позволяет теперь задать в качестве функций принадлежности только линейные (**linear**) или постоянные (**constant**) — в зависимости от того, какой алгоритм **Sugeno** (1-го или 0-го порядка) мы выбираем. Выбираем постоянные функции принадлежности с общим числом 5 (по числу различных значений *y* в табл. 1). Подтвердим введенные данные нажатием кнопки ОК.

8. Диапазон изменения (**Range**), устанавливаемый по умолчанию [0,1], менять в данном случае не нужно. Изменим лишь имена функций принадлежности (их графики при использовании алгоритма **Sugeno** для выходных переменных не приводятся), например, задав их как соответствующие числовые значения ***y***, т.е. {1, 0.36, 0, 0.16, 1}. Одновременно эти же числовые значения введем в поле **Params.** Затем закроем окно нажатием кнопки **Close** и вернемся в окно **FIS**-редактора.

9. Дважды щелкнем на среднем (белом) блоке, при этом раскроется окно еще одной программы — редактора правил (**Rule Editor**). Введем соответствующие правила. При вводе каждого правила необходимо обозначить соответствие между каждой функцией принадлежности аргумента ***x*** и числовым значением ***y***. Кривая, обозначенная нами ***x1,*** соответствует ***x*** -1, т.е. ***y*** = 1. Выберем поэтому в левом поле (с заголовком ***x*** is) вариант ***x1***, а в правом ***y1*** и нажмем кнопку **Add rule** (Добавить правило). Введенное правило появится в окне правил и будет представлять собой такую запись:

1. If (x is x1) then (y is y1). (1)

Аналогично поступим для всех других значений ***x,*** в результате чего сформируется набор из пяти правил. Закроем окно редактора правил и возвратимся в окно **FIS**-редактора. Построение системы закончено, и можно начать эксперименты по ее исследованию. Заметим, что для большинства опций были сохранены значения по умолчанию.

10. Предварительно сохраним на диске (используя команды меню **File** > **Export to disk**) созданную систему под каким-либо именем, например *Proba*.

Раскроем меню **View.** С помощью его команд **Edit membership functions** и **Edit rules** можно совершить переход к двум рассмотренным ранее программам — редакторам функций принадлежности и правил (то же можно сделать нажатием клавиш Ctrl+2 и Ctrl+3). Рассмотрим две другие команды — **View rules** (Просмотр правил) и **View Surface** (Просмотр поверхности). Выбираем эти команды.

11. В правой части окна в графической форме представлены функции принадлежности аргумента *x,* в левой — функции принадлежности переменной выхода *y* с пояснением механизма принятия решения. Когда красная вертикальная черта перемещается с помощью мыши, она изменяет значения переменной входа (можно задавать числовые значения в поле **Input** (Вход)), при этом соответственно изменяются значения *y* в правой верхней части окна. Зададим, например, $\boldsymbol{x} = 0{,}5$ в поле **Input** и нажмем затем клавишу Enter. Значение $\boldsymbol{y}$ сразу изменится и станет равным 0,202.

Таким образом, с помощью построенной модели и окна просмотра правил можно решать задачу интерполяции, то есть задачу, решение которой и требовалось найти. Изменение аргумента путем перемещения красной вертикальной линии очень наглядно демонстрирует, как система определяет значения выхода.

12. Смоделированное системой по таблице данных (см. табл. 1) отображение не очень-то напоминает функцию $\boldsymbol{x^2}$, так как число экспериментальных точек невелико, да и параметры функций принадлежности (для $\boldsymbol{x}$) выбраны, скорее всего, неоптимальным образом. В следующей работе рассмотрим возможность улучшения качества подобной модели.

В заключение рассмотрения примера отметим, что на любом этапе проектирования нечеткой модели в нее можно внести необходимые коррективы вплоть до задания какой-либо особенной пользовательской функции принадлежности. Из опций, устанавливаемых в **FIS**-редакторе по умолчанию при использовании алгоритма **Sugeno**, можно отметить следующие:

- логический вывод организуется с помощью операции умножения (prod);
- композиция организуется с помощью операции логической суммы (вероятностного ИЛИ, probor);
- приведение к четкости организуется дискретным вариантом центроидного метода (взвешенным средним, wtaver).

При использовании соответствующих полей в левой нижней части окна FIS-редактора, данные опции можно при желании изменить.

Построить нечеткую аппроксимирующую систему, выбрав зависимость по своему варианту.

ЛАБОРАТОРНАЯ РАБОТА № 2

Графический интерфейс гибридных систем

Цель работы. Создать гибридную систему по ранее выполненной зависимости $y = x^2$.

Графический интерфейс гибридных (нечетких) нейронных систем вызывается функцией ***anfisedit*** (из режима командной строки). Использование функции приводит к появлению окна редактора гибридных систем (ANFIS Editor, ANFIS-редактор).

С помощью данного редактора осуществляются создание или загрузка структуры гибридной системы, просмотр структуры, настройка ее параметров, проверка качества функционирования такой системы. Создание структуры, настройка параметров и проверка осуществляются по выборкам (наборам данных) — обучающей (Training data), проверочной (Checking data) и тестирующей (Testing data). Предварительно они должны быть представлены в виде текстовых файлов (с расширением dat и разделителями-табуляциями), первые столбцы которых соответствуют входным переменным, а последний (правый) — единственной выходной переменной; количество строк в таких файлах равно количеству образцов (примеров). Так, обучающая выборка, сформированная по табл. 1, представляется в виде Proba. dat

```
-1      1
-0.6    0.36
0.00    0.00
0.4     0.16
1       1
```

Строгих рекомендаций по объемам указанных выборок не существует; по-видимому, лучше всего исходить из принципа «чем больше, тем лучше».

Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров гибридной сети. Проверочная — для выяснения ситуации, не происходит ли так называемого *переобучения* сети, при котором ошибка для обучающей последовательности стремится к нулю. Для проверочной — возрастает; впрочем, наличие проверочной выборки не является строго необходимым, оно лишь крайне желательно. Тестовая (или тестирующая) выборка применяется для проверки качества функционирования настроенной (обученной) сети. Поясним пункты меню и опции редактора.

Меню **File** и **View** в общем идентичны аналогичным меню **FIS**-редактора за тем исключением, что здесь работа может происходить только с алгоритмом нечеткого вывода **Sugeno**. Меню **Edit** содержит единственную команду — **Undo** (Отменить выполненное действие). Набор опций **Load data** (Загрузка данных) в нижней левой части окна редактора включает в себя:

• тип (Type) загружаемых данных (для обучения — Training, для тестирования — Testing, для проверки — Checking, демонстрационные — Demo);

• место, откуда должны загружаться данные: с диска (disk) или из рабочей области MATLAB (workspace).

К данным опциям относятся две кнопки, нажатие на которые приводит к требуемым действиям - **Load Data** (Загрузить данные) и **Clear Data** (Стереть данные).

Следующая группа опций (в середине нижней части окна **ANFIS**-редактора) объединена под именем **Generate FIS** (Создание нечеткой системы вывода). Данная группа включает в себя следующие опции:

• загрузку структуры системы с диска (from disk);

• загрузку структуры системы из рабочей области MATLAB (from worksp.);

• разбиение (деление) областей определения входных переменных (аргументов) на подобласти — независимо для каждого аргумента (**Grid partition);**

• разбиение всей области определения аргументов (входных переменных) на подобласти — в комплексе для всех аргументов (**Sub.clustering**),

Кроме того, имеется также кнопка **Generate FIS**, нажатие которой приводит к процессу создания гибридной системы с точностью до ряда параметров.

Следующая группа опций — **Train FIS** (Обучение нечеткой системы вывода) — позволяет определить метод «обучения» (Optim.Method) системы (т. е. метод настройки ее параметров) — гибридный (hybrid) или обратного распространения ошибки (backproba), а также установить уровень текущей суммарной (по всем образцам) ошибки обучения (Error To1erance), при достижении которого процесс обучения заканчивается и количество циклов обучения (Epochs), т.е. количество «прогонов» всех образцов (или примеров) обучающей выборки. Процесс обучения, таким образом, заканчивается либо при достижении отмеченного уровня ошибки обучения, либо после проведения заданного количества циклов.

Кнопка **Train Now** (Начать обучение) запускает процесс обучения, т.е. процесс настройки параметров гибридной сети.

В правом верхнем углу окна ANFIS-редактора выдается информация (ANFIS Info) о проектируемой системе: количество входов, выходов, функций принадлежности входов. Нажатие кнопки **Structure** (Структура) позволяет увидеть структуру сети. Кнопка **C1ear** (Очистить) позволяет стереть все результаты.

Опции **Test FIS** в правом нижнем углу окна позволяют провести проверку и тестирование созданной и обученной системы с выводом результатов в виде графиков. Соответствующие графики для обучающей выборки — **Training data**, тестирующей выборки — **Testing data** и проверочной выборки — **Checking data**. Кнопка **Test Now** позволяет запустить указанные процессы.

Задание к лабораторной работе 2

Работу с редактором рассмотрим на примере восстановления зависимости $y = x^2$ по данным табл. 1. Предположим, что эти данные подготовлены в файле Proba. dat. Создание и проверку системы, как и раньше, проведем по этапам:

1. В окне ANFIS-редактора выберем тип загружаемых данных Training и нажмем кнопку Load data. В последующем стандартном окне диалога укажем местоположение и имя файла. Его открытие приводит к появлению в графической части окна редактора набора точек, соответствующих введенным данным.

2. В группе опций **Generate FIS** по умолчанию активизирован вариант **Grid partition**. Не будем изменять и нажмем кнопку **Generate FIS**, после чего появится диалоговое окно для задания числа и типов функций принадлежности. Сохраним все установки по умолчанию, согласившись с ними нажатием кнопки ОК. Произойдет возврат в основное окно ANFIS - редактора. Теперь структура гибридной сети создана, и ее графический вид можно просмотреть с помощью кнопки **Structure**.

3. Перейдем к опциям **Train FIS**. Не будем менять задаваемые по умолчанию метод настройки параметров (hybrid — гибридный) и уровень ошибки (0), но количество циклов обучения изменим на 40, после чего нажмем кнопку запуска процесса обучения (**Train Now**). Получился результат в виде графика ошибки сети в зависимости от числа проведенных циклов обучения (из которого следует, что фактически обучение закончилось после пятого цикла).

4. Теперь нажатием кнопки **Test Now** можно начать процесс тестирования обученной сети, но, поскольку использовалась только одна (обучающая) выборка, ничего особенно интересного ожидать не приходится. Действительно, выход обученной системы практически совпадает с точками обучающей выборки.

5. Сохраним разработанную систему на диске в файле с именем Proba1 (с расширением *fis*) и для исследования разработанной системы средствами FIS-редактора из командной строки MATLAB выполним команду **Fuzzy**, а затем через пункты меню **File | Open FIS from disk** откроем созданный файл. С созданной системой можно теперь выполнять все приемы редактирования (изменение имен переменных и т. п.) и исследования, которые были рассмотрены выше. В результате убеждаемся, что качество аппроксимации существенно не улучшилось – слишком мало данных.

Что можно сказать про эффективность использования гибридных систем и ANFIS-редактора?

В данном случае используется только один алгоритм нечеткого вывода — **Sugeno** (нулевого или первого порядков), может быть только одна выходная переменная, всем правилам приписывается один и тот же единичный вес. Вообще говоря, возникают значительные проблемы при

большом (более 5-6) количестве входных переменных. Это — ограничения и недостатки подхода.

Его несомненные достоинства: практически полная автоматизация процесса создания нечеткой (гибридной) системы, возможность просмотра сформированных правил и придания им содержательной (лингвистической) интерпретации, что позволяет, кстати говоря, рассматривать аппарат гибридных сетей как средство извлечения знаний из баз данных и существенно отличает данные сети от классических нейронных.

Рекомендуемая область применения: построение аппроксиматоров зависимостей по экспериментальным данным, построение систем классификации (в случае бинарной или дискретной выходной переменной), изучение механизма явлений.

Затем работу с редактором рассмотрим на примере зависимости y от x, заданной по вариантам.

ЛАБОРАТОРНАЯ РАБОТА № 3

Графический интерфейс программы кластеризации

Цель работы. Использовать программу **Clustering** (Кластеризация) для выявления центров кластеров, т.е. точек в многомерном пространстве данных, около которых группируются (скапливаются) экспериментальные данные. Выявление подобных центров является значимым этапом при предварительной обработке данных, поскольку позволяет сопоставить с этими центрами функции принадлежности переменных при последующем проектировании системы нечеткого вывода.

Запуск программы **Clustering** осуществляется командой (функцией) **findcluster**. В появляющемся окне программы имеется (вверху) главное меню, содержащее достаточно стандартный набор пунктов (File, Edit, Windows, Help) и набор управляющих кнопок и опций (справа). К этим кнопкам относятся:

- кнопка загрузки файла данных – Load Data;
- кнопка выбора алгоритма кластеризации – Method;
- четыре расположенные ниже кнопки опций алгоритма (их названия меняются в зависимости от выбранного алгоритма);
- кнопка начала итеративного процесса нахождения центров кластеров (кластеризации) – Start;
- кнопка сохранения результатов кластеризации – Save Center;
- кнопка очистки (стирания) графиков – C1ear P1ot;
- кнопка справочной информации – Info;
- кнопка завершения работы с программой – C1ose.

В программе используются два алгоритма выявления центров кластеров: **Fuzzy c-means** (Алгоритм нечетких центров) и **Subtractive clustering** (Вычитающая кластеризация). Ограничившись выявлением различий на уровне пользователя, можно отметить, что алгоритм **Fuzzy c-means** является, пожалуй, более точным (если понятие точности вообще здесь применимо). Для своей работы требуется задание таких опций, как число кластеров (кнопка **Cluster num**) и число итераций (кнопка **Max Iteration#**). Если число итераций еще можно задать как-то наугад, то ошибка в задании числа кластеров может в будущем привести к неприятным последствиям. Алгоритм **Subtractive clustering** менее точен, но и менее требователен к априорной информации; при работе с ним можно сохранить опции, заданные в программе по умолчанию.

Далее приведен пример использования программы для файла данных clusterdemo.dat из директории Matlab/toolbox/fuzzy/fuzdemos/ при использовании алгоритма Subtractive clustering. Заметим, что выводится только двумерное поле рассеяния, но, изменяя переменные в соответствующих полях (X-axis и Y-axis), можно «просмотреть» все многомерное пространство переменных.

Задание к лабораторной работе 3

1. Подготовить данные и сохранить их в файле **Proba.dat** в виде

-1 1
-0.6 0.36
0.0 0.00
0.4 0.16
1 1

2. При запуске программы **Clustering**, используя приведенный выше материал, выявить центры кластеров, то есть точки, около которых группируются (скапливаются) экспериментальные данные.

3. Подготовить данные для своей зависимости y от x, заданной по варианту, выявить центры кластеров.

4. Сохранить результаты кластеризации на своем диске.

ЛАБОРАТОРНАЯ РАБОТА № 4

Использование графического интерфейса для построения нечеткой аппроксимирующей системы

Цель работы. Построение нечеткой экспертной системы.

Рассмотрим методику построения нечеткой экспертной системы, которая должна помочь пользователю с ответом на вопрос: сколько дать на чай официанту за обслуживание в ресторане?

На основании каких-то устоявшихся обычаев и интуитивных представлений предположим, что задача о чаевых может быть описана следующими предложениями.

1. Если обслуживание плохое или еда подгоревшая, то чаевые – маленькие.

2. Если обслуживание хорошее, то чаевые – средние.

3. Если обслуживание отличное или еда превосходная, то чаевые - щедрые.

Качество обслуживания и еды будем оценивать по 10-балльной системе (0 – наихудшая оценка, 10 - наилучшая).

Будем предполагать далее, что малые чаевые составляют около 5% от стоимости обеда, средние – около 15 % и щедрые – примерно 25 %.

Заметим, что представленной информации, в принципе, достаточно для проектирования нечеткой экспертной системы. Такая система будет иметь 2 входа (которые можно условно назвать «сервис» и «еда»), один выход («чаевые»), три правила типа если….то (в соответствии с тремя приведенными предложениями) и по три значения (соответственно 0 баллов, 5 баллов, 10 баллов и 5%, 15%, 25%) для центров функций принадлежности входов и выходов. Построим данную систему, используя алгоритм **Мамдани**, описывая требуемые действия по пунктам.

Задание к лабораторной работе 4

1. Командой **Fuzzy** запускаем FIS-редактор. По умолчанию предлагается алгоритм **Мамдани** (о чем говорит надпись в центральном белом блоке), здесь никаких изменений не требуется, но в системе должно быть два входа, поэтому через пункт меню **Edit> Add Variable Input** добавляем в системе этот второй вход (в окне редактора появляется второй желтый блок с именем **Input 2**). Делая далее однократный щелчок на блоке **Input 1**, меняем его имя на «сервис», завершая ввод нового имени нажатием клавиши. Аналогичным образом устанавливаем имя «еда» блоку **Input 2** и «чаевые» - выходному блоку **Output 1**(справа вверху). Присвоим сразу же и имя всей системе, например TIP_ FIO**.** Зададим теперь функции принадлежности переменных. Программу – редактор функции принадлежности – можно открыть тремя способами: выберем первый через пункт меню **View> Edit membership function**. Задание и редактирование функций принадлежности начнем с переменной «сервис». Сначала в полях **Range** и **Display Range** установим диапазон изменения и отображения этой переменной – от 0 до 10 (баллов), подтверждая ввод нажатием клавиши Enter. Затем через пункт меню Edit/Add MFs перейдем к диалоговому окну на рис. 2.

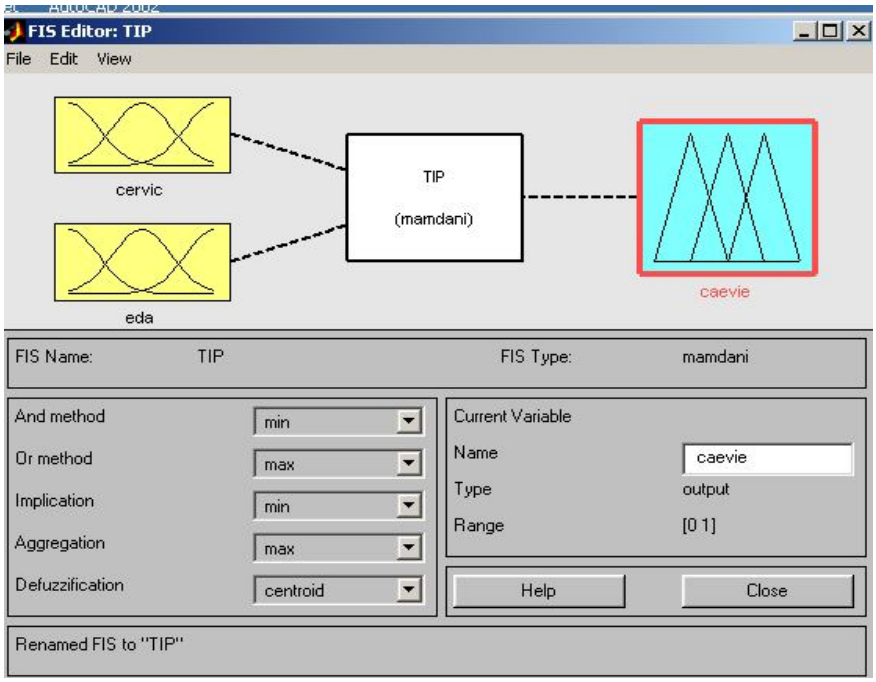


Рис. 2. Вид окна FIS Editor

Как показано на рис. 3, зададим три функции принадлежности гауссова типа (**gaussmf**). Нажмем кнопку ОК и возвратимся в окно редактора функций принадлежности. Не изменяя размах и положение заданных функций, заменим только их имена на «плохой», «хороший» и «отличный». Щелчком на значке «еда» войдем в окно редактирования функций принадлежности для этой переменной. Зададим сначала диапазон ее изменения от 0 до 10, а затем, поступая как ранее, зададим две функции принадлежности трапецеидальной формы с параметрами соответственно [0 0 1 3] и [7 9 10 10] и именами «подгоревшая» и «превосходная».

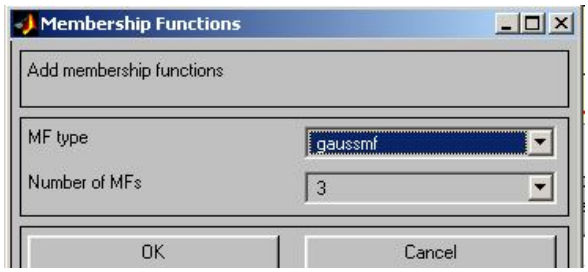


Рис. 3. Окно задания функции принадлежности пользователя

Для выходной переменной «чаевые» укажем сначала диапазон изменения (от 0 до 30), потом зададим три функции принадлежности треугольной формы с именами «малые», «средние» и « щедрые» так, как представлено на рис.4. Заметим, что можно, разумеется, задать и какие-либо другие функции или выбрать другие параметры.

Перейдем к конструированию правил. Для этого выберем пункт меню **View>Edit rules**. Далее ввод правил производится в соответствии с предложениями, описывающими задачу. Заметим, что в первом и третьем правилах в качестве «связки» в предпосылках правила необходимо использовать не «И» (and), а «ИЛИ» (or); при вводе второго правила, где отсутствует переменная «еда», для нее выбирается опция none. В результате формируется итоговый набор правил.

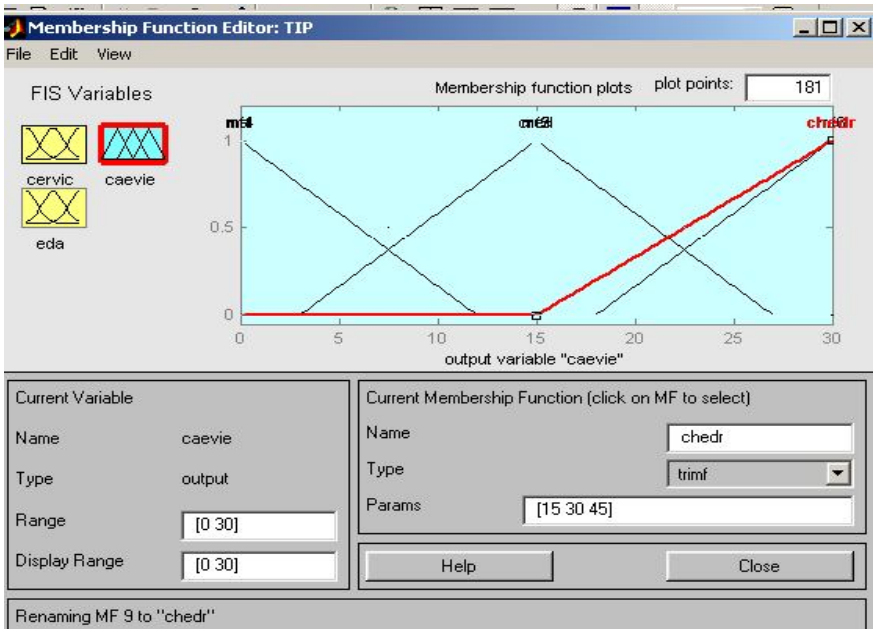


Рис. 4. Функции принадлежности переменной «чаевые»

Такая подробная (verbose) запись представляется достаточно понятной; единица в скобках после каждого правила указывает его «вес» (Weight), т.е. значимость правила. Данный вес можно менять, используя соответствующее поле в левой нижней части окна редактора правил. Правила представлены и в других формах – символической (symbolic) и индексной (indexed), при этом переход от одной формы к другой происходит с помощью меню Options> Format редактора правил. Вот как выглядят рассмотренные правила в символической форме:

1) (сервис==плохой)I(еда==подгоревшая)=>(чаевые==малые)(1);

2) (сервис==хороший)=>(чаевые==средние)(1);

3) (сервис==отличный)I(еда==превосходная)=>(чаевые==щедрые)(1).

Правила понятны.

Наконец, самый сжатый формат представления правил – индексный – является тем форматом, который в действительности используется программой. В этом формате приведенные правила выглядят так:

- o 1 1, 1 (1):2
- o 2 0, 2 (1):2
- o 3 2, 3 (1):2

Здесь первая колонка относится к первой входной переменной (соответственно первое, второе или третье возможное значение), вторая – ко второй, третья (после запятой) – к выходной переменной, цифра (после двоеточия) указывает тип «связки» (1 для «И», 2 для «ИЛИ»).

На этом соответственно конструирование экспертной системы закончено. Сохраним ее на диске под выбранным именем TIP.

Теперь самое время проверить систему в действии. Откроем (через пункт меню **View > View rules**) окно просмотра правил и установим значения переменных: сервис = 0 (то есть никуда не годный), еда = 10 (то есть превосходная). Увидим ответ: чаевые = 15 (то есть средние). Ну что ж, с системой не поспоришь, надо платить.

Можно проверить и другие варианты. В частности (может быть, не без удивления), выяснится, что нашей системой обслуживание ценится больше, чем качество еды: при наборе «сервис = 10, еда = 3» система советует определить размер чаевых в 23,9 %, в то время как набору «сервис = 3, еда = 10» размер чаевых по рекомендации системы — 16,6 % (от стоимости обеда). Впрочем, ничего удивительного здесь нет: это мы сами (не особенно подозревая об этом) заложили в систему соответствующие знания в виде совокупности приведенных правил.

Подтверждением отмеченной зависимости выходной переменной от входных может служить вид поверхности отклика, который представляется при выборе пункта меню View/View surface; обратите внимание, что с помощью мышки график можно поворачивать во все стороны.

В открывшемся окне, меняя имена переменных в полях ввода (X (input) и Y (input)), можно задать и просмотр одномерных зависимостей, например «чаевых» от «еды».

По приведенному выше примеру подготовим **свою экспертную систему**, протестируем ее и сохраним под именем *ECFio*, оформив отчет в виде, представленном в начале лабораторной работы 4.

ЛАБОРАТОРНАЯ РАБОТА № 5

Создание пользовательских функций принадлежности

Цель работы. Работа с **Fuzzy Logic Toolbox** в режиме командной строки. Возможности работы в режиме командной строки. Функции систем нечеткого вывода. Функции сохранения, открытия и использования созданной системы.

Пакет **Fuzzy Logic Toolbox** располагает большим набором функций, исполняемых из командной строки MATLAB и позволяющих не использовать при работе с системами нечеткого вывода рассмотренные программы графического интерфейса. Все функции делятся на следующие группы:

1) вызов программ графического интерфейса;
2) задания функций принадлежности;
3) создание, редактирование, просмотр, открытие и сохранение систем нечеткого вывода;
4) дополнительные;
5) сервисные;
6) вызов диалоговых окон интерфейса;
7) блоки Simulink;
8) демонстрации возможностей пакета.

Функции вызова программ графического интерфейса

К этой группе относятся следующие функции:

• **fuzzy** — вызов FIS-редактора;
• **mfedit** — вызов редактора функций принадлежности;
• **ruleedit** — вызов редактора правил;
• **ruleview** — вызов программы просмотра правил;
• **surfview** — вызов программы просмотра поверхности отклика;
• **anfisedit** — вызов FIS-редактора (только для систем, использующих алгоритм Sugeno и имеющих одну выходную переменную);
• **findcluster** — вызов программы кластеризации.

Использование первых шести функций с аргументом (например, fuzzy(a), где **a** — имя переменной рабочего пространства, присвоенное системе нечеткого вывода) открывает соответствующую программу с одновременной загрузкой в нее рассматриваемой системы. Функция **findcluster** (имя_файла) открывает программу кластеризации с одновременной загрузкой указанного файла данных.

Чтение, использование и сохранение на диск созданной системы нечеткого вывода в режиме командной строки осуществляется следующими функциями:

readfis (‘имя_файла’)
evalfis (вектор_параметров, имя)
writefis (имя) или writefis (имя.'имя_файла')

Здесь имя_файла — наименование файла с написанной системой (без указания расширения), имя — идентификатор, который дан системе в рабочей среде MATLAB. Вектор_параметров — набор значений входов, для которых требуется рассчитать выход (возможна и матрица параметров, тогда результат расчетов — вектор в случае одной выходной переменной или матрица при нескольких таких переменных).

Задания к лабораторной работе 5

1. На примере ранее созданной и сохраненной на диске в виде файла с именем **Tip** экспертной системы для задачи о чаевых использовать системы нечеткого вывода с помощью графического интерфейса, приведенного выше:

```
>>a = readfis ('Tip');
>>out = evalfis ([1 2],a)
   out =
          <результат>
>>writefis (a, 'Tip')
```

Выбрать следующие функции использования графического окна, которые позволяют использовать элементы графических изображений вне программ с графическим интерфейсом.

Команда (функция) **plotfis (a)** вызовет появление графического окна MATLAB с мнемоническим представлением разработанной системы (рис. 5.1).

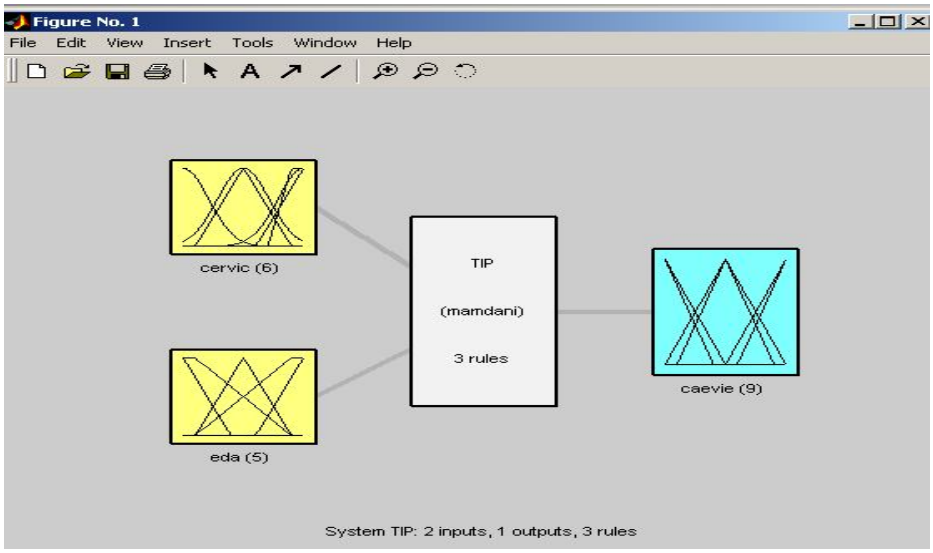


Рис 5.1. Мнемоническое представление системы нечеткого вывода

Команда (функция) **gensurf (a)** делает то же самое, но применительно к поверхности отклика (рис. 5.2).

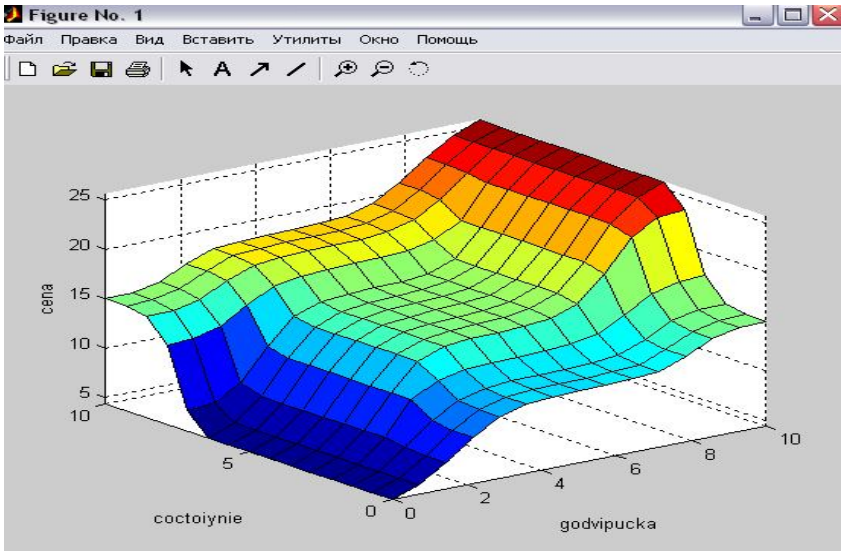
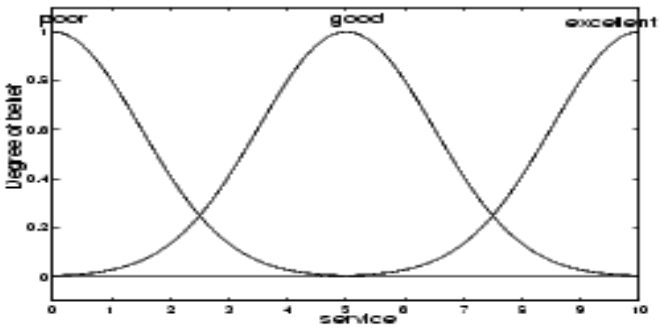


Рис.5.2. Результат выполнения функции gensurf (имя)

2. Далее выполнить пример построения экспертной системы для задачи о чаевых, используя функции создания, просмотра структуры и редактирования систем нечеткого вывода: функции **newfis** (новая система), **addvar** (добавить переменную), **addmf** (добавить функцию принадлежности) и **addrule** (добавить правило).

3. Сконструировать систему нечеткого вывода целиком в режиме командной строки MATLAB.

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
plotmf(a,'input',1)
```



Затем точно так же указать все для «еды» и для «чаевых», только заменить ‘input’ на ‘output’.

```
ruleList=[
  1 1 1 1 1
  1 2 2 1 1];
```

a = addrule(a,ruleList).

Далее при необходимости просмотра элементов структуры (системы *c* идентификатором *a*) в режиме командной строки следует ввести команду вида *a*. Параметр, например

>> a.type

Получим ответ:

ans =

mamdani.

Получение информации по всем элементам структуры обеспечивается функцией **getfis** (a).

Функция **setfis** (a) в определенном смысле противоположна функции **getfis** (a) и позволяет изменять параметры.

Полный просмотр структуры системы нечеткого вывода осуществляется функцией **showfis** (a).

Просмотр правил, включенных в систему нечеткого вывода, осуществляется с помощью функции **showrule** (a).

```
a = readfis ('tipper');
showrule (a,1)
ans =
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
showrule (a,2)
ans =
2. If (service is good) then (tip is average) (1)
Showrule (a,[3 1],'symbolic')
ans =
3. (service==excellent) | (food==delicious) => (tip=generous) (1)
1.    (service==poor) | (food==rancid) => (tip=cheap) (1)
Showrule (a,1:3,'indexed')
ans =
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2.
```

4. Повторить весь процесс с применением указанных программ для своей экспертной системы, созданной в лабораторной работе 4, на примере построения системы в задаче о чаевых.

ЛАБОРАТОРНАЯ РАБОТА № 6

Средства для проектирования, моделирования, обучения и макроязык программирования искусственных нейронных сетей

Цель работы. Использование пакета Neural Networks Toolbox, который позволяет создавать, обучать нейронные сети.

В состав пакета **Neural Networks Toolbox** (Нейронные сети) входят более 150 различных функций, что позволяет пользователю создавать, обучать и использовать самые различные искусственные нейронные сети (ИНС). Пакет может быть использован для решения множества разнообразных задач, таких как обработка сигналов, нелинейное управление, финансовое моделирование и т.п.

Для каждого типа архитектуры и обучающего алгоритма ИНС имеются функции инициализации, обучения, адаптации, создания, моделирования, демонстрации, а так же, примеры применения. Искусственные многослойные нейронные сети конструируются по принципам построения их биологических аналогов. Они уже сейчас способны решать широкий круг задач распознавания образов, идентификации, управления сложными нелинейными объектами, роботами и т.п.

Представим некоторые проблемы, решаемые в контексте ИНС, и представляющие интерес для пользователей.

- **Классификация образов.** Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, к одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови.

- **Кластеризация/категоризация**. При решении задачи кластеризации, которая известна так же, как классификация образов «без учителя», отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и помещает близкие образы в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

- **Аппроксимация функций.** Предположим, что имеется обучающая выборка $((x^1,y^1), (x^2,y^2)..., (x^N, y^n))$ (пары данных «вход—выход»), которая генерируется неизвестной функцией $F(x)$, искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции $F(x)$. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

- **Предсказание/прогноз.** Пусть заданы n дискретных отсчетов $\mathbf{y(t_1)}$, $\mathbf{y(t_2)}$, $y(t_k)\}$ в последовательные моменты времени $t_1, t_2, ...,t_k$. Задача состоит в предсказании значения $\mathbf{y(t_{k-1})}$ в некоторый будущий момент времени $\mathbf{t_{k-1}}$. Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

- **Оптимизация.** Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Известная задача коммивояжера является классическим примером задачи оптимизации.

- **Память, адресуемая по содержимому.** В модели вычислений фон Неймана – обращение к памяти доступно только посредством адреса, который не зависит от содержимого памяти. Более того, если допущена

ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержимому, доступна по указанию заданного содержимого. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержимому. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.

▪ **Управление.** Рассмотрим динамическую систему, заданную совокупностью ***{u(t), y(t)},*** где ***u(t)*** является входным управляющим воздействием, а ***y(t)*** — выходом системы. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия ***u(t),*** при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление двигателем.

Эти и подобные задачи успешно решаются средствами пакета Neural Networks Toolbox.

Net=newgrnn – (P,T, speread) – функция создания обобщенно-регрессионной сети.

Задания к лабораторной работе 6

1. Выполнить пример создания и использования нейронных сетей для аппроксимации функции на примере своей заданной функции по своему варианту, используя указанный ниже пример. Создадим обобщенно-регрессионную НС с именем ***A*** для аппроксимации функции вида $y=x^2$ на отрезке [-1, 1], используя следующие экспериментальные данные (10 точек), со знака **%** вводятся комментарии.

X = [-1 -0.8 -0.5 0.2 0 0.1 0.3 0.6 0.9 1]
Y = [1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1]

```
>> % Задание входных значений
>> P = [-1 -0.8 -0.5 0.2 0 0.1 0.3 0.6 0.9 1];
>> % Задание выходных параметров
>> T = [1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1];
>> a=newgrnn(P,T,0.01);    % Создание НС
>> P1 = [-0.9 -0.7 -0.3 0.4 0.8]
P1 =
  - 0.9000   -0.7000   -0.3000    0.4000    0.8000
>> Y = sim(a,P1) % Опрос НС
Y =
   0.8200    0.6400    0.2500    0.0900    0.8100.
```

Как видно, точность аппроксимации в данном случае получилась не очень высокой.

Можно попытаться улучшить качество аппроксимации за счет подбора величины отклонения, но в условиях примера приемлемый результат легко достигается при использовании сети с радиальными базисными элементами.

a=newrbe(P,T);
>> P1 = [-0.9 -0.7 -0.3 0.4 0.8];
>> Y = sim(a,P1) % Опрос НС
Y = 0.8099 0.4901 0.0900 0.1600 0.6400

Созданную сеть можно сохранить для последующего использования набором в командной строке команды **save ('a');** при этом будет создан файл **a.mat**, то есть файл с именем НС и расширением **.mat.** В последующих сеансах работы можно загрузить сохраненную сеть, используя функцию **load('a')**.

2. Создать сеть для своей функции, сохранить на своем диске.

3. Рассмотреть аналогичную задачу, но с использованием линейной НС.

Пусть экспериментальная информация задана значениями

X = 1 1.5 3 -1.2
Y = 0.5 1.1 3 -1

Процесс создания, обучения и использования линейной НС с именем **b** приведен ниже и иллюстрируется приведенным ниже листингом и рис. 6.

>>P = [1 1.5 3 -1.2];
>> T = [0.5 1.1 3 -1];
>> % Определение величины коэффициента обучения
>> maxlr=maxlinlr(P,'bias');
>> b=newlin([-2 2],1,[0],maxlr);
>> b.trainParam.epochs=15; % Задание количества циклов обучения
>> b = train(b,P,T); % Обучение НС
TRAINB, Epoch 0/15, MSE 2.865/0.
TRAINB, Epoch 15/15, MSE 0.0730734/0.
TRAINB, Maximum epoch reached.

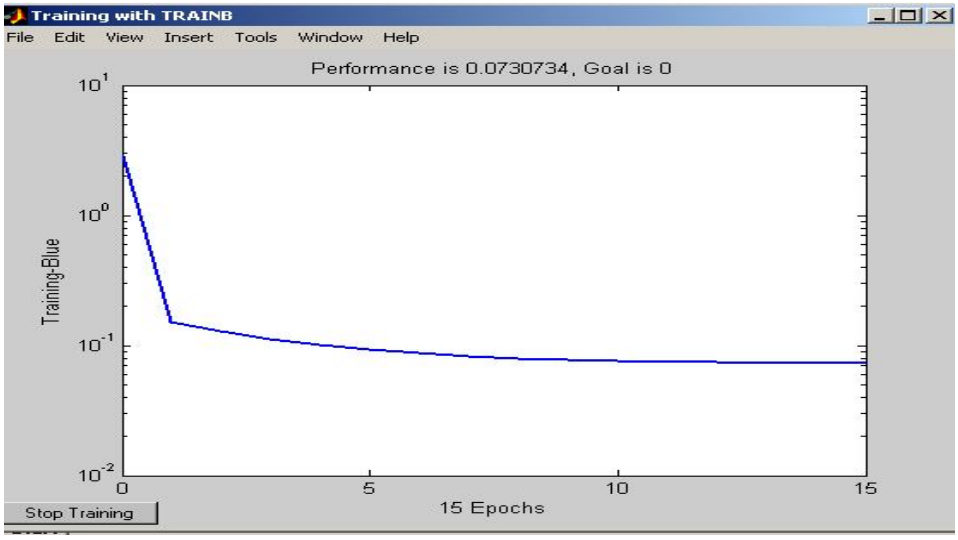


Рис. 6. Изменение ошибки сети в процессе ее обучения

>> p = -1.2;
% Опрос сети
>> y = sim(b,p)
y =
-1.1803

4. Использование слоя Кохонена. Рассмотрим задачу автоматического выявления (в режиме обучения без учителя) центров кластеров входов для двумерного случая с использованием слоя Кохонена (слоя «соревнующихся» нейронов). Решение данной задачи приведено ниже.

```
» % Задание диапазонов возможного положения центров кластеров:
» X = [0  1;    0  1];
» % Задание параметров для моделирования исходных данных.
» % принадлежащих 8 классам (кластерам)):
» clusters = 8;
» points = 10;
» std_dev = 0.05;
» % Моделирование входных данных
» P = nngenc(X,clusters,points,std_dev);
» % Создание слоя Кохонена
» net = newc([0 1; 0 1],8,1);
» % Задание количества циклов обучения
» net.trainParam.epochos=500;
» % Инициализация сети
» net=init(net);
» % Обучение сети
» net=train(net,P);
» w=net.IW{1};
» % Вывод графика исходных данных и выявленных центров кластеров
» plot(P(1,:),P(2,:),'+r');
» hold on;
» plot(w(:,1),w(:,2),'ob');
» xlabel('p(1)');
» ylabel('p(2)');
» % Задание нового входного вектора
» p = [0; 0.2];
»y = sim(net,p)   % Опрос сети
y =   (8,1)        1
```

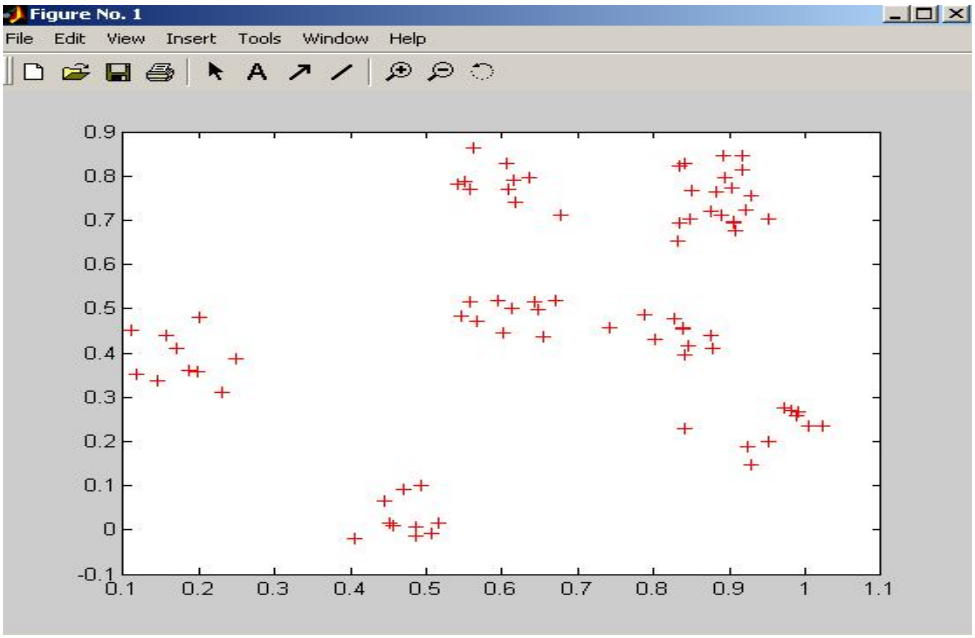


Рис. 7. Исходные данные

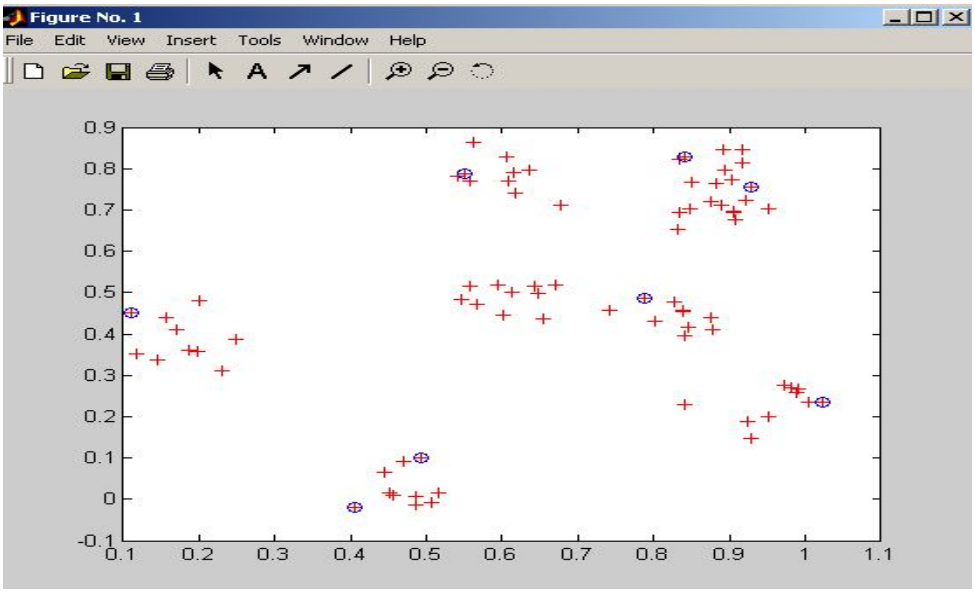


Рис. 8. Исходные данные и выявленные центры кластеров

Работу обученной сети и результат ее опроса иллюстрирует рис. 8. (матрица y в конце приведенного листинга, которая выдается в форме разреженной матрицы). В условиях примера предъявленный вектор отнесен к третьему классу (кластеру). Сохранить сеть на диске.

5. **Сеть Хопфилда с двумя нейронами**. Рассмотрим сеть Хопфилда, имеющую два нейрона и два устойчивых состояния, отображаемых векторами [1 -1] и [-1 1]. Результат представлен на рис. 9. Эти векторы выведены следующей программой:

```
» T = [1 -1; -1 1];
» plot(T(1,:),T(2,:),'r*')
```

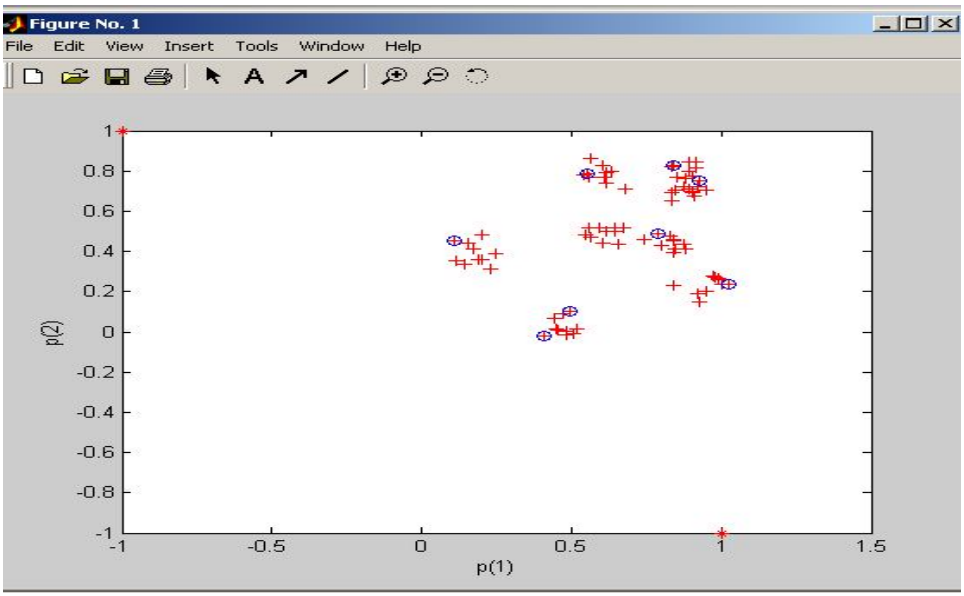


Рис. 9. Исходные данные и выявленные центры кластеризации

```
>> axis([-1.1 1.1 -1.1 1.1]);
>> title('Пространство векторов НС Хопфилда');
>> xlabel('a(1)');
>> ylabel('a(2)').
```

Создадим сеть Хопфилда (с именем H) и проверим ее работу, подав на вход векторы, соответствующие устойчивым точкам. Если сеть работает правильно, она должна выдать эти же векторы без каких-либо изменений.

```
>> H = newhop(T);                    %Создание НС Хопфилда
>> [Y,Pf,Af] = sim(H,2,{},T) ; Y     % Опрос сети Хопфилда
Y =
     1    -1
    -1     1
```

Как видно из результатов опроса, сеть работает правильно. Подадим теперь на ее вход произвольный вектор

```
a = {rands(2,1)};  % Задание случайного вектора
>> [Y,Pf,Af] = sim(H,{1 50},{},a);
>> plot(T(1,:),T(2,:),'r*')
>> axis([-1.1 1.1 -1.1 1.1]);
>> record=[cell2mat(a) cell2mat(Y)];
>> start=cell2mat(a);
>> hold on;
>> plot(start(1,1),start(2,1),'bx',record(1,:),record(2,:))
>> xlabel('a(1)');
>> ylabel('a(2)');
>> title(‘Результат работы сети Хопфилда’).
```

Результат иллюстрируется на рис. 10. Сохранить сеть на диске.

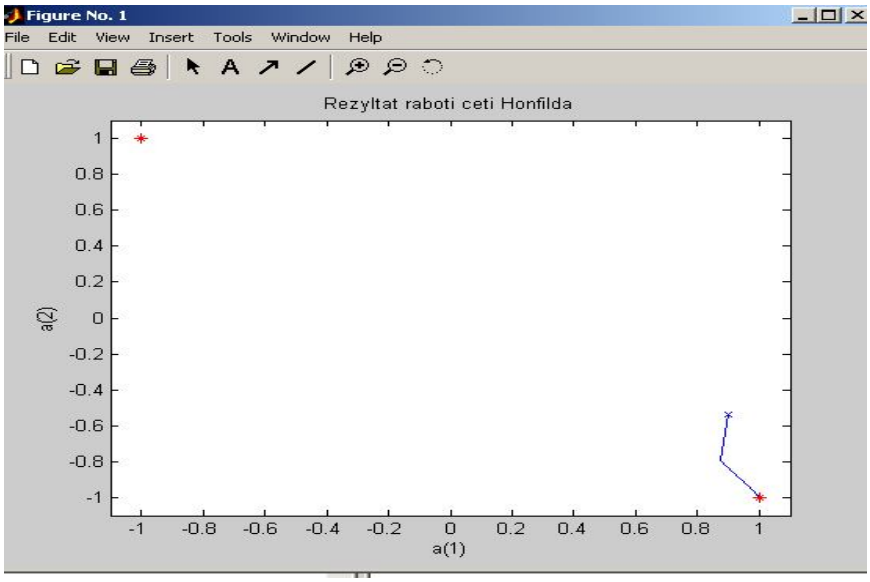


Рис. 10. Результат работы сети Хопфилда

6. **Классификация с помощью персептрона.** Следующий пример иллюстрирует решение задачи классификации с помощью персептрона. Исходные входные векторы '(с указанием их принадлежности к одному из двух классов) и результат настройки персептрона (с именем My_net,) представлены на рис. 11.

```
» %Задание входных векторов с указанием их принадлежности
» %одному из двух классов
>>P = [-0.5 -0.5 0.3 -0.1;-0.5 0.5 -0.5 1];
>> T = [1 1 0 0];
» % Графическое представление исходных векторов
>> plotpv (P,T);
» % Создание персептрона с указанием границ изменений входов
» % и 1 нейроном
>> My_net=newp ([-1 1;-1 1],1);
>> E = 1;
» %  Инициализация персептрона
>> My_net=init (My_net);
» % Организация цикла адаптивной настройки персептрона
» % с выводом графика разделяющей линии
>> while (sse(E))
    [My_net,Y,E]=adapt (My_net,P,T);
    linehand=plotpc (My_net.IW{1},My_net.b{1});
    drawnow;          end;
```

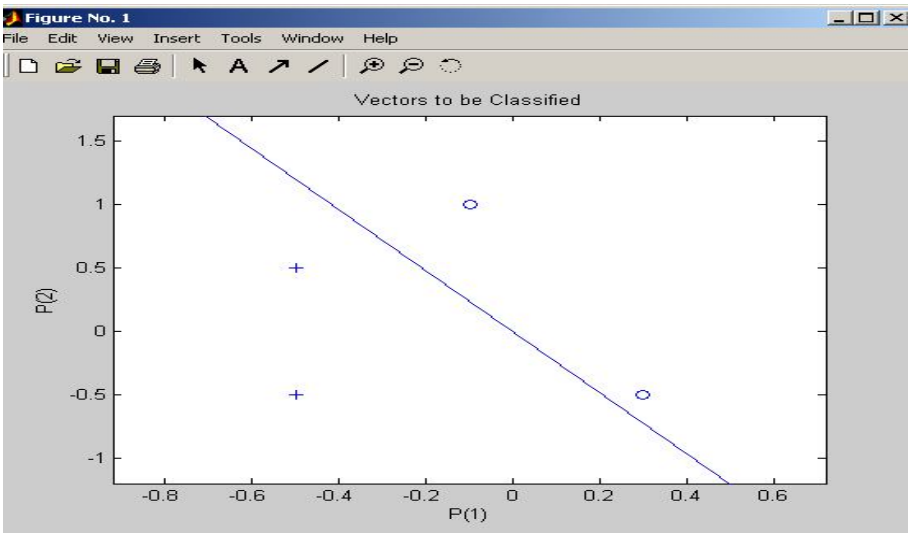


Рис. 11. Исходные входные векторы и разделяющая линия

7. Сохранить сеть на диске.

8. **Адаптивный линейный прогноз**. В предыдущем примере настройка НС производилась адаптивно. Отличие такой настройки от выполняемой, например, с помощью метода обратного распространения ошибки заключается в том, что векторы обучающей выборки поступают на вход сети не все «одновременно», а последовательно по одному. При этом после предъявления очередного вектора производится корректировка весов и смещений, и может быть произведен опрос сети, затем все повторяется. Адаптивная настройка особенно удобна при работе НС в «реальном» масштабе времени.

Рассмотрим пример задачи прогнозирования значений сигнала (по 5 предыдущим значениям) с использованием указанной настройки. Предположим, что исходный сигнал определен на интервале времени от 0 до 6 секунд, при этом при $0 < t < 4$ с он описывается соотношением $x(t) = \sin(4\pi)$, а при $4 < t < 6$ с — соотношением $x(t) = \sin(8\pi)$. График такого сигнала приведен на рис. 12.

```
>>time1=0:0.05:4;          % от 0 до 4 секунд
>> time2=4.05:0.024:6;   % от 4 до 6 секунд
>> time = [time1 time2];
>> % T  определяет исходный сигнал
>> T = con2seq([sin(time1*4*pi) sin(time2*8*pi)]);
>> % График исходного сигнала
>> plot(time,cat(2,T{:}))
>> xlabel('Время');
>> ylabel('Исходный сигнал');
>> title(‘Прогнозируемый сигнал');
```

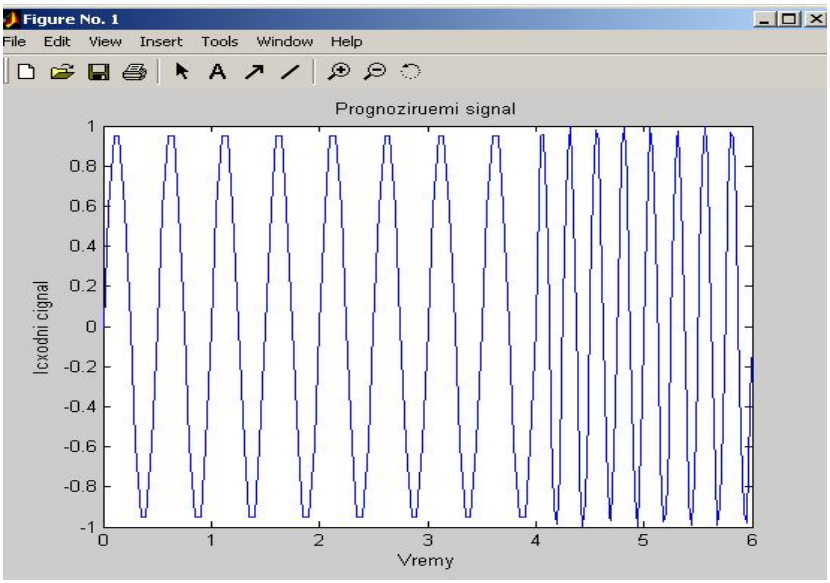


Рис. 12. График прогнозируемого сигнала

Сохранить сеть на диске.
Для прогноза значений сигнала создадим линейную НС.

```
>>% Входной и целевой прогнозируемый сигнал одинаковы
>> P = T;
>>% Задание коэффициента обучения
>> lr=0.1;
>>% Для прогноза используются 5 предыдущих значений
>> delays = [1 2 3 4 5];
>>% Создание и настройка линейной НС
>> net = newlin(minmax (cat(2,P{:})),1,delays,lr);
>> [net,y,e] = adapt (net,P,T);
>>% Графики исходного сигнала и прогноза
>> plot (time,cat (2,y{:}),time,cat(2,T{:}),'--')
```

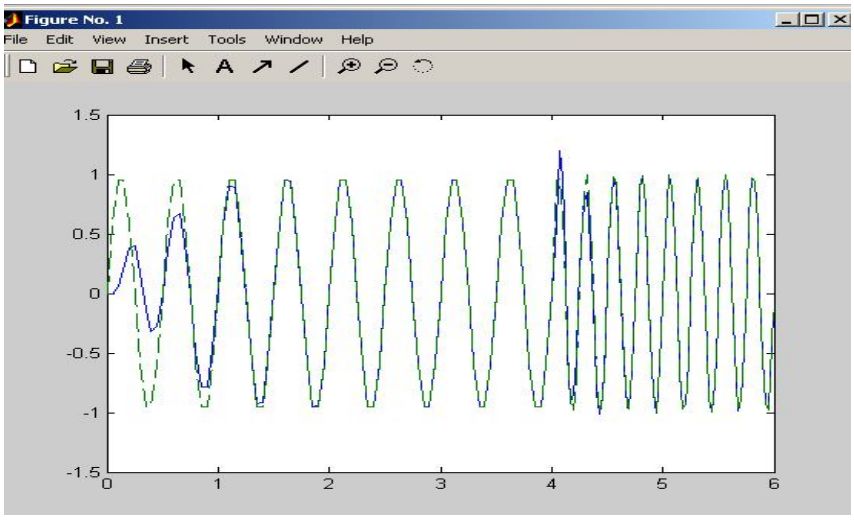


Рис. 13. Исходный сигнал и прогноз

>> plot(time,cat (2,y{:}),time, cat (2,T{:}),'--')
>> xlabel ('Время');
>> ylabel ('Исходный сигнал');
>> title ('(‘Прогнозируемый сигнал");

Исходный сигнал и прогноз приведены на рис. 13. полученный результат можно считать удовлетворительным. Сохранить сеть.

9. **Использование сети Элмана**. Рассмотрим задачу восстановления формы сигнала с использованием рекуррентной сети Элмана. Пусть имеются два синусоидальных сигнала — один с единичной амплитудой, другой — с амплитудой, равной двум:

P1 = sin (1:20);
P2 = sin (1:20)*2;

Пусть целевым сигналом будет сигнал, составленный из их амплитудных значений:

T1 = ones (1.2);
T2 = ones (1.2)*2;

При этом данные амплитуды чередуются, так что входные и целевые значения могут быть представлены в форме

p = [p1 p2 p1 p2];
t = [t1 t2 t1 t2].

Преобразуем эти значения в последовательности:

Pseq = con2seq(p);
Tseq = con2seq(t);

После этого можно непосредственно перейти к проектированию НС. В рассматриваемом случае имеем, очевидно, один вход и один выход, то есть в сети должны присутствовать один входной элемент и один выходной нейрон. Число нейронов в скрытом слое может быть, вообще говоря, любым (оно зависит от сложности задачи); примем, что этот слой содержит 10 нейронов. Дальнейшее решение задачи иллюстрируется ниже, один график представляет изменения ошибки сети в процессе ее обучения, а другой — результаты тестирования сети.

>> % Задание исходных данных
>> p1 = sin(1:20);
>> t1 = ones(1,2);
>> p2 = sin(1:20)*2;
>> t2 = ones(1,2)*2;
>> p = [p1 p2 p1 p2];
>> t = [t1 t2 t1 t2];
>> Pseq = con2seq(p);
>> Tseq = con2seq(t);
>> % Создание сети Элмана с диапозоном входа [-2, 2] 10
>> % нейронами скрытого слоя одним выходным нейроном

```
>> % функцией активации в виде гиперболического тангенса
>> % для нейронов скрытого слоя, линейной функцией активации
>> % для выходного нейрона, функцией обучения с адаптацией
>> % коэффициента обучения
>> net = newelm([-2 2],[10 1],{'tansig','purelin'},'traingdx');
>> % Задание параметров обучения
>> % Целевое значение функции ошибки
>> net.trainParam.epochs = 500;  %  Число циклов обучения
>> net.trainParam.goal = 0.01;
>> net.performFcn = 'sse';    % Задание вида функции ошибки
>> % Обучение сети. По умолчанию промежуточные результаты
>> % обучения выводятся через 25 циклов
>> [net.tr]=train(net.Pseq.Tseq);
>> % Построение графика функции ошибки
>> semilogy(tr.epoch.tr.perf);
>> title('Сумма квадратов ошибок сети Элмана');
>> xlabel('Циклы');
>> ylabel('Сумма квадратов ошибок');
>> % Тестирование сети
>>a=sim(net.Pseq);
>>time=1:length(p);
>>time=1:length(p);
>>plot(time.t.'—'.time.cat(2.a{:}));
>>title('Результаты тестирования сети');
>> xlabel('Время');
>> ylabel('Заданные значения --- Выход сети');
```

10.Сохранить сеть на диске.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

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

2. Уткин, В.Б. Информационные системы в экономике: учебник для студ. высш. учеб. заведений В.Б. Уткин. М.: Академия, 2004. – 288 с.

3. Городецкий, А.Г. Программные средства интеллектуальных систем / А.Г. Городецкий. СПб.: СПбГТУ, 2000. 171 с.

4. Джексон, П. Введение в экспертные системы: учеб. пособие / П. Джексон. М.: Вильямс, 2001. – 624 с.

5. Андрейченков, А.В. Интеллектуальные информационные системы: учебник / А.В. Андрейченков. – М.: Финансы и статистика, 2004. – 424 с.