Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

Абрахин С.И., Лоханов А.В., Хорьков К.С.

Методические указания для выполнения курсовой работы по дисциплине «Нечеткая логика и нейронные сети»

для бакалавров по направлению
38.03.05 «Бизнес информатика»

Владимир, 2016 г.

Составители: Абрахин Сергей Иванович, Лоханов Александр Васильевич, Хорьков Кирилл Сергеевич

Методические указания для выполнения курсовой работы по дисциплине «Нечеткая логика и нейронные сети» / Владим. гос. уни-т имени Александра Григорьевича и Николая Григорьевича Столетовых; Абрахин Сергей Иванович, Лоханов Александр Васильевич, Хорьков Кирилл Сергеевич – Владимир: Изд-во ВлГУ, 2016.- 60 с.

Методические рекомендации составлены на основе требований ФГОС ВО направления 38.03.05 «Бизнес-информатика». Приведено описание курсовой работы по дисциплине «Нечеткая логика и нейронные сети». Включают цель, теоретические положения, методику выполнения работы, требования к пояснительной записке. Для организации эффективной работы студентов использованы методические пособия ведущих вузов России.

Пособие выполнено в рамках государственного задания ВлГУ на 2014/13 на выполнение государственных работ в сфере научно-технической деятельности.

Рецензент – Давыдов Н.Н.-профессор, доктор технических наук, директор ИПМФИ

Оглавление

Введение

Дисциплина «Нечеткая логика и нейронные сети» включена в учебный план подготовки бакалавров по направлению 38.03.05 «Бизнес-информатика» в радел дисциплин по выбору вариативной части.

Целью курсовой работы по дисциплине «Нечеткая логика и нейронные сети» закрепление теоретических знаний и формирование практических навыков по умению ставить задачи по практическому применению средств и методов нечеткой логики и нейронных сетей и анализ результатов их проведения, оформление результатов.

Объектами профессиональной деятельности бизнес-аналитика являются сложные математические модели, в том числе описывающие процесс принятия решений. Главная цель этого курса состоит в том, чтобы подготовить студентов к разработке систем поддержки принятий решений. Мощь и интуитивная простота нечеткой логики и нейронных сетей как методологии разрешения проблем гарантирует ее успешное использование во встроенных системах контроля и анализа информации. В отличие от традиционной математики, требующей на каждом шаге моделирования точных и однозначных формулировок закономерностей, нечеткая логика предлагает совершенно иной уровень мышления.

Задачи дисциплины:

- изучение основных приемов и методов использования аппарата нечетких множеств в задачах управления и принятия решений;
- изучение основных приемов проектирования и обучения нейронных сетей;
- формирование навыков построения нечетких моделей и нейронных сетей, наиболее полно отвечающих требованиям поставленной задачи;
- изучение способов реализации нечетких моделей и нейронных сетей в виде программ для ЭВМ;
- отработка практических навыков, связанных с выбором и использованием программных средств с элементами нечеткой логики и нейронных сетей.

1 Основы нечеткой логики

1.1 Понятие нечеткого множества

1.1.1 Понятие принадлежности

Пусть U есть множество (универсальное множество), A – подмножество U: $A \subset U$.

Тот факт, что элемент x множества U есть элемент подмножества A, или, как еще говорят, принадлежит A, обычно обозначают с помощью символа $\in$: $x \in A$.

Для выражения этой принадлежности можно использовать и другое понятие –*характеристическую функцию* $\mu_A(x)$, значения которой указывают, является ли (да или нет) x элементом A:

$$\mu_A(x) = \begin{cases} 1, & \text{если } x \in A \\ 0, & \text{если } x \notin A \end{cases}.$$

Пример 1

Рассмотрим конечное множество из пяти элементов $U=\{x_1,x_2,x_3,x_4,x_5\}$ и пусть $A=\{x_2,x_3,x_5\}$.

Выпишем для каждого элемента из U степень его принадлежности множеству A:

$$\mu_A(x_1)=0,\ \mu_A(x_2)=1,\ \mu_A(x_3)=1,\ \mu_A(x_4)=0,\ \mu_A(x_5)=1.$$

Это позволяет представить A через все элементы множества E, сопроводив каждый из них значением его функции принадлежности:

$$A = \{(x_1, 0), (x_2, 1), (x_3, 1), (x_4, 0), (x_5, 1)\}.$$

1.1.2 Понятие нечеткого множества

Рассмотрим подмножество A множества U, определенное в примере 1. Каждый из пяти элементов U или принадлежит или не принадлежит A. Характеристическая функция принимает только значения 0 или 1.

Представим теперь, что характеристическая функция может принимать любое значение в интервале [0, 1]. В соответствии с этим элемент x_i множества U может не принадлежать A ($\mu_A=0$), может быть элементом A в небольшой степени (μ_A близко к 0), может более или менее принадлежать A (μ_A ни слишком близко к 0, ни слишком близко к 1), может в значительной степени быть элементом A (μ_A близко к 1)

или, наконец, может быть элементом A (μ_A=1). Таким образом, понятие принадлежности получает интересное обобщение, приводящее как это мы увидим, к очень полезным результатам.

Математический объект, определяемый выражением:

$$\tilde{A} = \{(x_1|0{,}2), (x_2|0), (x_3|0{,}3), (x_4|1), (x_5|0{,}8)\}$$

где x_i – элемент универсального множества U, а число после вертикальной черты дает значение характеристической функции на этом элементе, будем называть нечетким подмножеством множества U и обозначать $\tilde{A} \subset U$.

Принадлежность нечеткому подмножеству будем обозначать так:

$$x \underset{0,2}{\in} \tilde{A},\ y \underset{0}{\in} \tilde{A},\ z \underset{1}{\in} \tilde{A}.$$

1.1.3 Определение нечеткого множества

Дадим строгое определение понятия, нечеткого подмножества (нечеткого множества), введенного Заде [1].

Определение 1: Пусть E есть множество, счетное или нет, и x – элемент E. Тогда нечетким подмножеством A множества E называется множество упорядоченных пар

$$\{(x,\mu_A(x))\},\ \forall x \in E,$$

где $\mu_A(x)$ – степень принадлежности x в A.

Таким образом, если $\mu_A(x)$ принимает свои значения во множестве значений M функции принадлежности или, иначе говоря, во множестве принадлежностей, то можно сказать, что x принимает значение в M посредством функции $\mu_A(x)$. Таким образом,

$$x \underset{\mu_A}{\rightarrow} M\ .$$

Эта функция также называется *функцией принадлежности*.

Обычно множество M значений функции принадлежности это множество значений из интервала [0,1]. Учитывая это, приведем определение нечеткого подмножества данного Л. Заде [1].

Определение 2: Нечеткое подмножество A универсального множества U характеризуется *функцией принадлежности* $\mu_A: U \to [0,1]$, которая ставит в соответствие каждому элементу $x \in U$ число $\mu_A(x)$ из интервала [0,1], характеризующее *степень принадлежности* элемента u подмножеству A.

Как видно из этого определения, нечеткое множество вполне описывается своей функцией принадлежности, поэтому далее мы часто будем использовать эту функцию как обозначение нечеткого множества.

1.2 Понятие лингвистической переменной

Лингвистическая переменная отличается от числовой переменной тем, что ее значениями являются не числа, а слова или предложения из естественного или формального языка. Поскольку слова в общем менее точны, чем числа, понятие лингвистической переменной дает возможность приближенно описывать явления, которые настолько сложны, что не поддаются описанию в общепринятых количественных терминах.

В лингвистической переменной роль нечетких множеств аналогична роли, которую играют слова и предложения в естественном языке. Нечеткие множества представляют собой нечеткие ограничения, связанные со значениями лингвистической переменной, можно рассматривать как совокупную характеристику различных подклассов элементов универсального множества.

Приведем определение лингвистической переменной данного Л. Заде [1].

Определение 3: Лингвистическая переменная описывается пятеркой $<X, T(X), U, M, G>$, где

- X – имя переменной;
- T –терм-множество переменной X, т.е. множество названий лингвистических значений переменной X, причем каждое из таких значений является нечеткой переменной x со значениями из универсального множества U с базовой переменной u;

- M – семантические правила, задающие функции принадлежности нечетких термов, которое ставит в соответствие каждой нечеткой переменной (каждому терму) *x* ее смысл *M(x)*, т.е. нечеткое подмножество *M(x)* универсального множества *U;*
- *G* – синтаксическое правило (грамматика), согласно которому каждому элементу *u* из универсального множества *U* ставится в соответствие одно из значений терм-множества T, т.е. правило порождающее значение переменной *X (из терм множества).*

Задание значения переменной словами, без использования чисел, для человека более естественно. Ежедневно мы принимаем решения на основе лингвистической информации типа: "очень высокая температура"; "длительная поездка"; "быстрый ответ" и т.п.

Пример 2

Рассмотрим *лингвистическую переменную* с именем X="Температура в аудитории". Тогда оставшуюся четверку <*T, U, M, G*>, можно определить так:

- универсальное множество U=[5,35] с базовой переменной *t*;
- терм-множество T={"Холодно", "Комфортно", "Жарко"}
- семантика *M* будет являться процедурой, ставящей каждому терму в соответствие нечеткое множество из *U*. Функции принадлежности:

$$\mu_{\text{Холодно}}(t) = \frac{1}{1 + \left(\frac{t-10}{7}\right)^{12}};$$

$$\mu_{\text{Комфртно}}(t) = \frac{1}{1 + \left(\frac{t-20}{3}\right)^{6}};$$

$$\mu_{\text{Жарко}}(t) = \frac{1}{1 + \left(\frac{t-30}{7}\right)^{12}};$$

- синтаксическое правило G (грамматика), присваивающее значение лингвистической переменной (выбор терма из терм множества):

$$G(t) = max\{\mu_{\text{Холодно}}(t), \mu_{\text{Комфртно}}(t), \mu_{\text{Жарко}}(t)\}.$$

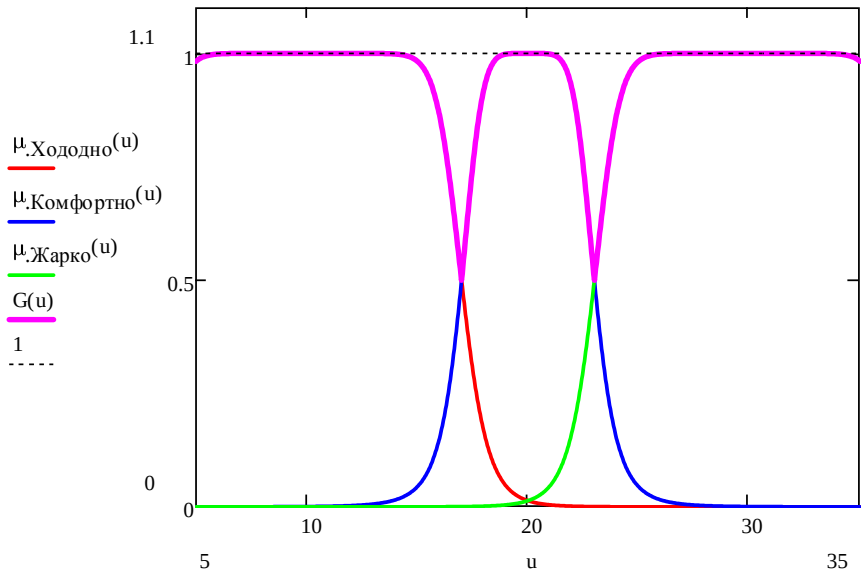


Рисунок 1. Графическое представление лингвистической переменной «Температура в аудитории»

В рассмотренном примере терм-множество состояло лишь из небольшого числа термов, так что целесообразно было просто перечислить элементы терм-множества *T(X)* и установить прямое соответствие между каждым элементом и его смыслом.

В более общем случае, число элементов *T(X)* в может быть бесконечным, и тогда как для порождения элементов множества *T(X)*, так и для вычисления их смысла необходимо применять алгоритм, а не просто процедуру перечисления.

1.3 Логико-лингвистическое описание систем, нечеткие модели

Логико-лингвистические методы описания систем основаны на том, что поведение исследуемой системы описывается на естественном (или близком к естественному) языке в терминах лингвистических переменных.

Определяются входные и выходные параметры системы, которые рассматриваются как лингвистические переменные, а качественное описание процесса задается совокупностью высказываний L_1 ,..., L_k . Например, если исследуемая система имеет один «вход» ($A=\cup <A_i>$, $i=1,...,k$) и один «выход» ($B=\cup <B_i>$, $i=1,...,k$), то высказывания будут следующего вида:

L_1 : если A_1 то B_1,
L_2 : если A_2 то B_2,
.....
L_k : если A_k то B_k,

где $A_1,A_2,..,A_k$ – термы входной лингвистической переменной (предпосылки), заданные как нечеткие множества, а $B_1, B_2, ..., B_k$ - термы выходной лингвистической переменной (заключения), заданные как нечеткие множества. Здесь, каждое из таких высказываний L_i представляется импликацией $A_i \rightarrow B_i$.

Совокупность импликаций $\{L_1, L_2, ..., L_k\}$ отражает функциональную взаимосвязь входной и выходной переменных и является основой построения нечеткого отношения $X\ R\ Y$, заданного на произведении $X \times Y$ универсальных множеств входных и выходных переменных.

Если система имеет более одного входа и/или выхода, то с помощью правил преобразования дизъюнктивной (*и*) и конъюнктивной (*или*) формы описание системы можно привести к виду:

$$L_i: \text{ если } A_{i_1}^1 \text{ и/или } A_{i_2}^2 \text{ и/или ... и/или } A_{i_n}^n \text{ то } B_{j_1}^1 \text{ и/или } B_{j_2}^2 \text{ и/или ... и/или } B_{j_m}^m, \quad i=1,...,k$$

где: n и m количество входных и выходных переменных соответственно; A^t, $t=1,...,n$ лингвистические переменные описывающие входные параметры системы; B^r, $r=1,...,m$ лингвистические переменные описывающие выходные параметры системы; $i_t, t=1,...,n$ и j_r, $r=1,...,m$ номера термов соответствующих входных и выходных параметров системы.

1.4 Этапы нечеткого логического вывода).

Используемый в различного рода экспертных и управляющих системах механизм нечетких выводов в своей основе имеет нечеткую базу знаний, состоящую из нечетких правил, формируемую специалистами предметной области в виде логико-лингвистического описания системы [2].

В общем случае общий нечеткий логический вывод осуществляется за следующие четыре этапа:

1. Нечеткость (введение нечеткости, фаззификация, fuzzification). Функции принадлежности, определенные на входных переменных применяются к фактическим значениям для определения степени истинности каждой предпосылки каждого правила.

2. Логический вывод. Логический вывод включает два под этапа:

2.1. Агрегация. Определяется степень истинности всех предпосылок в каждом правиле. Агрегация необходима если в правилах больше чем одна предпосылка.

2.2 Активация Вычисленное значение истинности для всех предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к нечетким множествам, которые будут назначено каждой переменной вывода для каждого правила. На этапе активации происходит обращение к нечеткой базе знаний.

3. Композиция. Все нечеткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы формировать одно нечеткое подмножество для каждой переменной вывода. В результате композиции получается нечеткий вывод. Композиция необходима, если нечеткая база знаний включает больше одного правила.

4. Приведение к четкости (дефаззификация, defuzzification) используется, когда необходимо преобразовать нечеткий набор выводов к четкому аналогу.

Пример 3. Пусть некоторая система описывается следующими нечеткими правилами:

$П_1$: если x есть A, тогда w есть D,
$П_2$: если y есть B, тогда w есть E,
$П_3$: если z есть C, тогда w есть F,

где *x, y и z* – имена входных переменных, *w* – имя переменной вывода, а *A, B, C, D, E, F* – заданные функции принадлежности.

Процедура получения логического вывода иллюстрируется рис. 2. Предполагается, что исходные переменные приняли некоторые конкретные (четкие) значения – x_0 , y_0 и z_0.

В соответствии с приведенными этапами, на этапе 1 для данных значений и исходя из функций принадлежности *A, B, C,* находятся

степени истинности $\alpha(x_0)$, $\alpha(y_0)$, $\alpha(z_0)$ предпосылок каждого из трех приведенных правил (см. рис. 2).

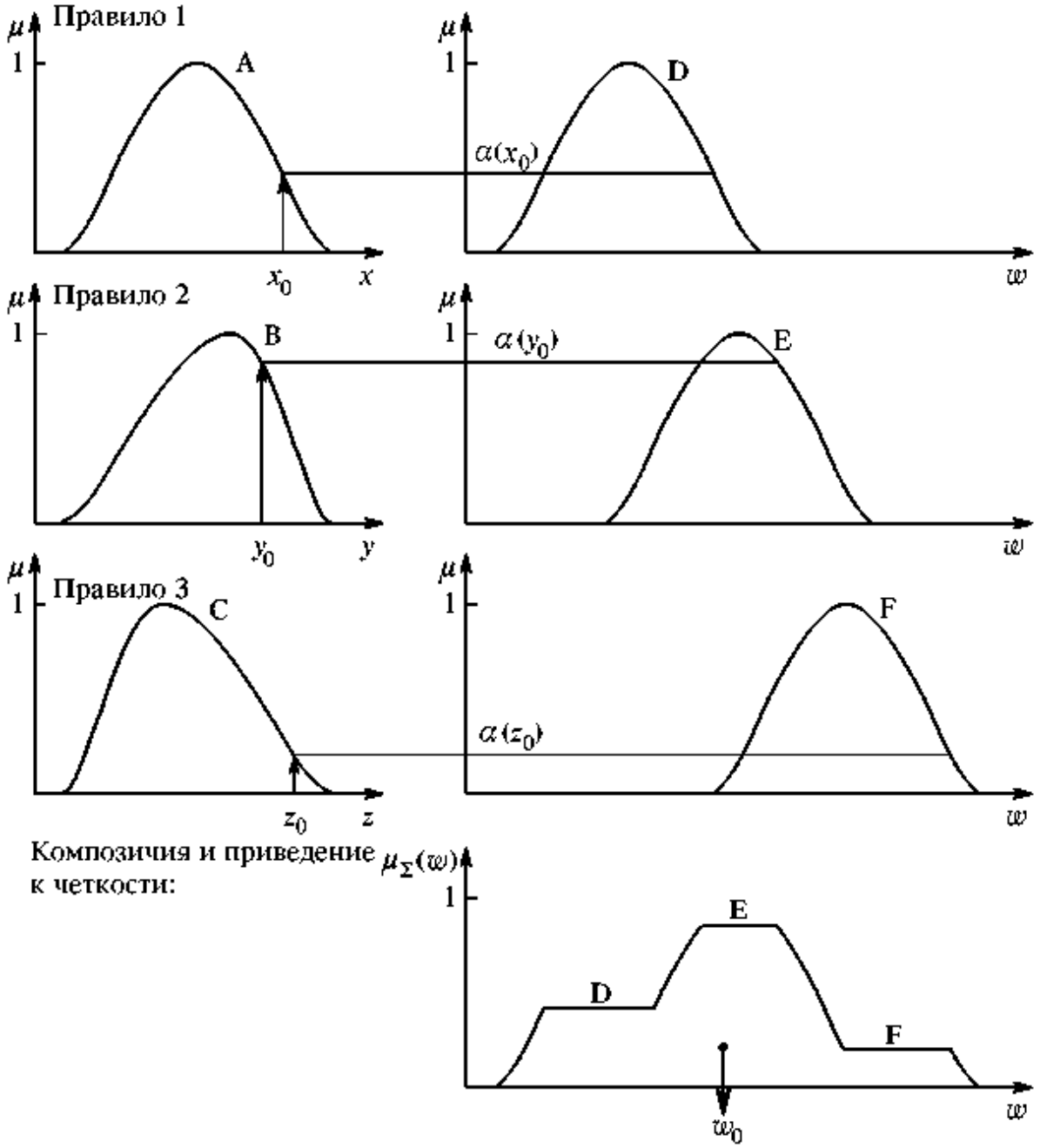


Рисунок 2. Процедура получения логического вывода

На этапе 2 происходит «отсекание» функций принадлежности заключений правил (т.е. *D, E, F*) на уровнях $\alpha(x_0)$, $\alpha(y_0)$ и $\alpha(z_0)$. Так каждое правило додержит лишь одну предпосылку под этап агрегации в данном примере не требуется.

На этапе 3 рассматривается усеченные на втором этапе функции принадлежности и производится их объединение с использованием операции max, в результате чего получается комбинированное нечет-

кое подмножество, описываемое функцией принадлежности $\mu_{\sum}(w)$ и соответствующее логическому выводу для выходной переменной w.

Наконец, на 4-м этапе – при необходимости – находится четкое значение выходной переменной, например, с применением центроидного метода: четкое значение выходной переменной определяется как центр тяжести для кривой $\mu_{\sum}(w)$, т. е.:

$$w_0 = \frac{\int_{\Omega} w\mu_{\sum}(w)dw}{\int_{\Omega} \mu_{\sum}(w)dw}.$$

1.5 Алгоритмы нечеткого логического вывода

Рассмотрим следующие наиболее часто используемые модификации алгоритма нечеткого вывода, полагая, для простоты, что базу знаний организуют два нечетких правила вида:

$П_1$: если x есть A_1 и y есть B_1, тогда z есть C_1,

$П_2$: если x есть A_2 и y есть B_2, тогда z есть C_2.

где x и y – имена входных переменных, z – имя переменной вывода, A_1, A_2, B_1, B_2, C_1 и C_2 – некоторые заданные функции принадлежности, при этом четкое значение z_0 необходимо определить на основе приведенной информации и четких значений x_0 и y_0 [2].

1. Алгоритм Mamdani. Данный алгоритм соответствует рассмотренному примеру и рис. 2. В рассматриваемой ситуации он математически может быть описан следующим образом.

1. Нечеткость: находятся степени истинности для предпосылок каждого правила: $A_1(x_0)$, $A_2(x_0)$, $B_1(y_0)$, $B_2(y_0)$.

2.Нечеткий вывод: На этапе агрегаци находятся уровни «отсечения» для предпосылок каждого из правил (с использованием операции МИНИМУМ)

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

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

где через «$\wedge$» обозначена операция логического минимума (min), затем, на этапе активации, находятся «усеченные» функции принадлежности:

$$C'_1 = \alpha_1 \wedge C_1(z),$$

$$C'_2 = \alpha_2 \wedge C_2(z).$$

3. Композиция: использование операции МАКСИМУМ (max, далее обозначаемой как «$\vee$») производится объединение найденных усеченных функций, что приводит к получению итогового нечеткого подмножества для переменной выхода с функцией принадлежности:

$$\mu_{\sum}(z) = C(z) = C'_1(z) \vee C'_2(z) = (\alpha_1 \wedge C_1(z)) \vee (\alpha_2 \wedge C_2(z)).$$

4. Наконец, приведение к четкости (для нахождения z_0) проводится, например, центроидным методом.

2. Алгоритм Larsen. В алгоритме Larsen логическое «и» и нечеткая импликация моделируется с использованием оператора умножения.

1. Первый этап – как в алгоритме Mamdani.

2. На этапе агрегации, находятся значения истинности предпосылок каждого правила:

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

а затем, на этапе активации, – частные нечеткие подмножества $\alpha_1 C_1(z)$, $\alpha_2 C_2(z)$

3. Находится итоговое нечеткое подмножество с функцией принадлежности:

$$\mu_{\sum}(z) = C(z) = \alpha_1 C_1(z) \vee \alpha_2 C_2(z)$$

4. При необходимости производится приведение к четкости (как в ранее рассмотренном алгоритме).

3. Алгоритм Tsukamoto. Исходные посылки – как у предыдущих алгоритмов, но в данном случае предполагается, что функции $C_1(z)$, $C_2(z)$ являются монотонными.

1. Первый этап - такой же, как в алгоритме Mamdani.

2. На этапе агрегации, находятся (как в алгоритме Mamdani) уровни «отсечения» α_1 и α_2, а затем, на этапе активации, – посредством решения уравнений:

$$\alpha_1 = C_1(z), \quad \alpha_{2=} C_2(z)$$

– четкие значения (z_1 и z_2) для каждого из исходных правил.

3. Определяется четкое значение переменной вывода (как взвешенное среднее z_1 и z_2):

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}.$$

4. Алгоритм Sugeno. Sugeno и Takagi использовали набор правил в следующей форме (как и раньше, приводим пример двух правил):

$П_1$: если x есть A_1 и y есть B_1, тогда $z_1=a_1x+b_1y$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z_2=a_2x+b_2y$.

1. Первый этап – как в алгоритме Mamdani.

2. На этапе активации находится $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ и на этапе активации индивидуальные выходы

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

3. На третьем этапе определяется четкое значение переменной вывода:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}.$$

5. Упрощенный алгоритм нечеткого вывода. Исходные правила в данном случае задаются в виде:

$П_1$: если x есть A_1 и y есть B_1, тогда $z_1=c_1$,

$П_2$: если x есть A_2 и y есть B_2, тогда $z_2=c_2$,

где c_1 и c_2 – некоторые обычные (четкие) числа.

1. Первый этап – как в алгоритме Mamdani.

2. На этапе агрегации находятся числа $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$, которые на этапе активации дают четкое значение выходной переменной по формуле $z_0 = \frac{\alpha_1 c_1 + \alpha_2 c_2}{\alpha_1 + \alpha_2}$.

1.6 Аппроксимация функций с применением нечеткого вывода

Возможность использования аппарата нечеткой логики для аппроксимации функций базируется на следующих результатах.

Будем считать, что функция z зависит от двух переменных x,y полагая, для простоты, что базу знаний организуют два нечетких правила вида:

П1: если x есть A1 и y есть B1, тогда z есть C1,

П2: если x есть A2 и y есть B2, тогда z есть C2.

Теорема

Нечеткая система, использующая набор правил:

Пi: если xi есть Ai и y есть Bi, тогда zi есть Ci, i=1, 2, …, n, при:

1. гауссовских функциях принадлежности

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

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

2. агрегации в виде произведения $(A_i(x) \text{ and } B_i(y)) = A_i(x) \cdot B_i(y)$;

3.активации в форме (Larsen) $(A_i(x) \text{ and } B_i(y)) \to C_i(z) = A_i(x) \cdot B_i(y) \cdot C_i(z)$;

4. центроидном методе приведения к четкости $z_0 = \sum_{i=1}^{n} \alpha_i A_i B_i \Big/ \sum_{i=1}^{n} A_i B_i$, где αi центры Ci; является универсальным аппроксиматором, т.е. может аппроксимировать любую непрерывную функцию на компакте U с произвольной точностью (естественно, при n→∞).

1.6.1 Исходные данные для аппроксимации фцнкций

В качестве исходных данных имеем таблично заданную функцию одной или многих переменных (табл. 1), каждый столбец которой представляет собой зависимость выхода (значение функции) от входов (аргументы функции).

Таблица 1. Исходные данные: таблично заданная функция многих переменных

			m – размерность экспериментальных данных					
Входы системы n– размерность вектора аргументов функции y(X)	X	x_1	x_1^1	x_1^2	…	x_1^j	…	x_1^m
		x_2	x_2^1	x_2^2	…	x_2^j	…	x_2^m
		…	…	…	…	…	…	…
		x_i	x_i^1	x_i^2	…	x_i^j	…	x_i^m
		…	…	…	…	…	…	…
		x_n	x_n^1	x_n^2	…	x_n^j	…	x_n^m
Выход системы	y		y_1	y_2	…	y_j	…	y_m

1.6.2 Этапы аппроксимации функций с применением нечеткого вывода

Для реализации аппроксимации функций с применением нечеткого вывода необходимо выполнение следующих этапов и под этапов:

1. Построение логико-лингвистического описания системы:
   - Представление входов и выходов системы в виде лингвистических переменных;
   - Построение нечетких высказываний вида «Если …., то …» характеризующих зависимость выходного параметра от входов (генерация нечеткой базы знаний).
2. Реализация механизма нечеткого логического вывода.

1.6.3 Фаззификация входных переменных

Определим входные и выходные лингвистические переменные (лп). Очевидно, в качестве входных лингвистических переменных следует использовать вектор аргументов x_i, $i=\overline{1,n}$ функции *y(X)*, а в качестве выходной – значение функции *y*.

Опишем входы системы x_i, $i=\overline{1,n}$ в соответствии с определением лингвистической переменной (описание для выхода *y* будет аналогичным):

1. Собственное имя переменной $-$ «x_i».
2. Область определения каждой переменной представляет собой диапазон $[\underline{x_i}, \overline{x_i}]$, в котором изменяется аргумент x_i, где $\underline{x_i} = \min\limits_{i=1,n} x_i$ – нижняя граница диапазона, $\overline{x_i} = \max\limits_{i=1,n} x_i$ – верхняя граница диапазона, определяемые по исходной таблице.
3. Множество значений лингвистических переменных – терм–множество *T={“Низкий”, “Ниже среднего”, “Средний”, “Выше среднего”, “Высокий”}* или в символическом виде *T={“Н”, “НС”, “С”, “ВС”, “В”}*.

Диапазон возможных значений каждой переменной $[\underline{x_i}, \overline{x_i}]$ разбиваем на пять частей, которые ассоциируются с пятью лингвистическими оценками (рис. 3).

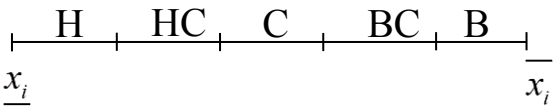


Рисунок 3. Ассоциирование разбиения диапазона возможных значений переменной x_i с лингвистическими оценками

Каждый терм из множества Т определяется функцией принадлежности Гаусса, которая задается двумя параметрами – (b,c):

$$\mu^T(x_i) = e^{-\frac{(x_i-b)^2}{2c^2}},$$

b и c – параметры настройки: b – координата максимума функции, $\mu^T(b)=1$; c – коэффициент концентрации растяжения функции.

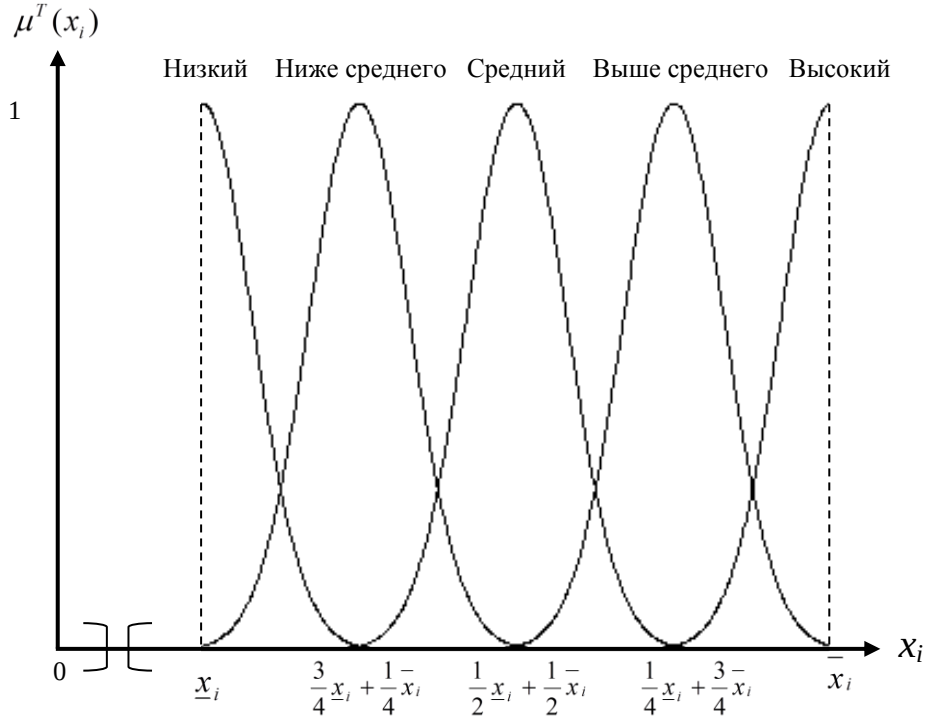


Рисунок 4. Графики функций принадлежности для термов входных лингвистических переменных x_i, $i=\overline{1,n}$

Имеем следующие функции принадлежности для пяти термов:

$$\mu^H(x_i) = e^{-\frac{(x_i-b_H)^2}{2c^2}},\ b_H = \underline{x}_i,\ c_H = \frac{\sqrt{2}}{4\ln(2)^{\left(\frac{1}{2}\right)}} \cdot \frac{\overline{x}_i - \underline{x}_i}{4},$$

$$\mu^{HC}(x_i) = e^{-\frac{(x_i-b_{HC})^2}{2c^2}},\ b_{HC} = \frac{3}{4}\underline{x}_i + \frac{1}{4}\overline{x}_i,\ c_{HC} = \frac{\sqrt{2}}{4\ln(2)^{\left(\frac{1}{2}\right)}} \cdot \frac{\overline{x}_i - \underline{x}_i}{4},$$

$$\mu^{C}(x_i)=e^{-\frac{(x_i-b_C)^2}{2c^2}},\ b_C=\frac{1}{2}\underline{x}_i+\frac{1}{2}\overline{x}_i,\ c_C=\frac{\sqrt{2}}{4\ln(2)^{\left(\frac{1}{2}\right)}}\cdot\frac{\overline{x}_i-\underline{x}_i}{4},$$

$$\mu^{BC}(x_i)=e^{-\frac{(x_i-b_{BC})^2}{2c^2}},\ b_{BC}=\frac{1}{4}\underline{x}_i+\frac{3}{4}\overline{x}_i,\ c_{BC}=\frac{\sqrt{2}}{4\ln(2)^{\left(\frac{1}{2}\right)}}\cdot\frac{\overline{x}_i-\underline{x}_i}{4},$$

$$\mu^{B}(x_i)=e^{-\frac{(x_i-b_B)^2}{2c^2}},\ b_B=\overline{x}_i,\ c_B=\frac{\sqrt{2}}{4\ln(2)^{\left(\frac{1}{2}\right)}}\cdot\frac{\overline{x}_i-\underline{x}_i}{4}.$$

4. M – процедура задания на области определения $X=[\underline{x}_i,\overline{x}_i]$ нечетких подмножеств T_1= “*Низкое* x_i”, T_2= “*Ниже среднего* x_i”, T_3= “*Средний* x_i”, T_4= “*Средний* x_i”, T_5= “*Высокое* x_i”.
5. G – грамматика, в соответствии с которой генерируются термы с применением слов естественного или формального языка (рис. 5):

$$G_{x_i}=T_l,\ m.ч.:\mu^{T_l}(x_i)=\max_{\text{T=("H", "HC", "C", "BC", "B")}}\left(\mu^{T}(x_i)\right),\ i=\overline{1,n}.$$

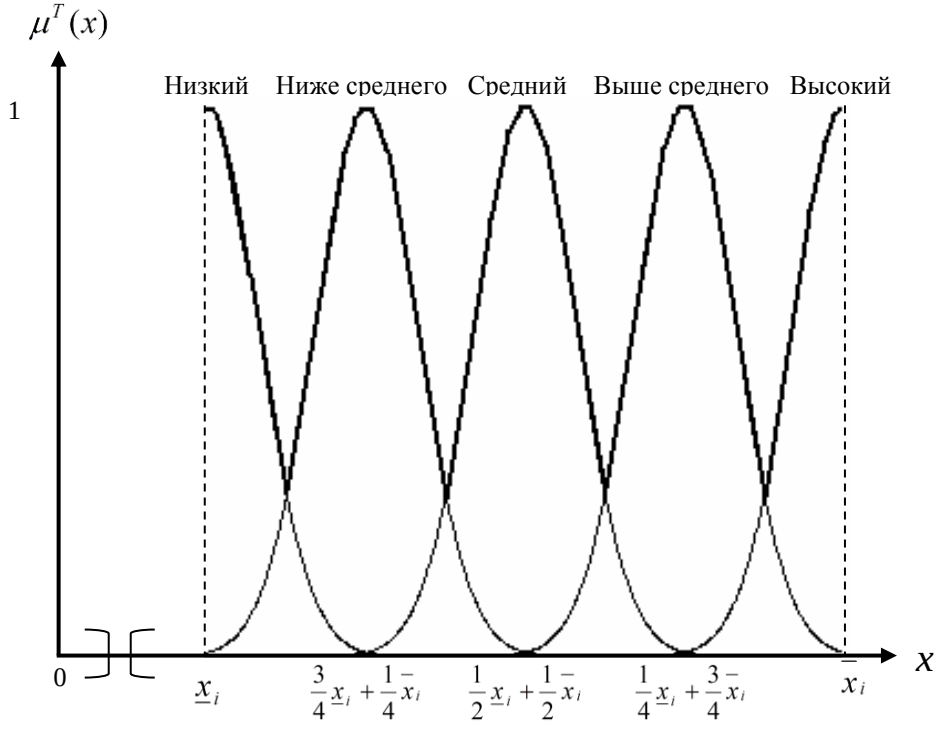


Рисунок 5. Грамматика лингвистической переменной

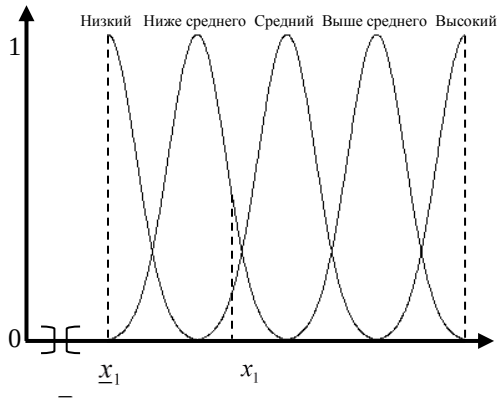
1.6.4 Генерация базы нечетких лингвистических правил

База правил систем нечеткого вывода формируется на основе предварительно определенных входных и выходной лингвистических переменных.

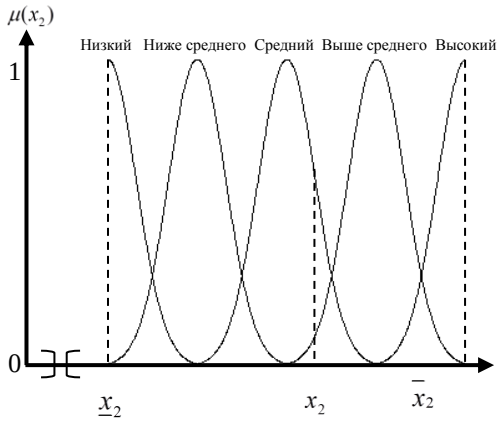
Учитывая имеющееся разбиение диапазона значений аргумента x_i, $i=\overline{1,n}$ и значений функции *y* сформулируем высказывания на естественном языке. Для составления лингвистического правила вида “ЕСЛИ .., ТО …” требуется в соответствии с грамматикой лингвистических переменных выбрать термы, соответствующие табличным значениям x_i и *y*.

Если система обладает двумя или более входами, то используем логическую связку “И”.

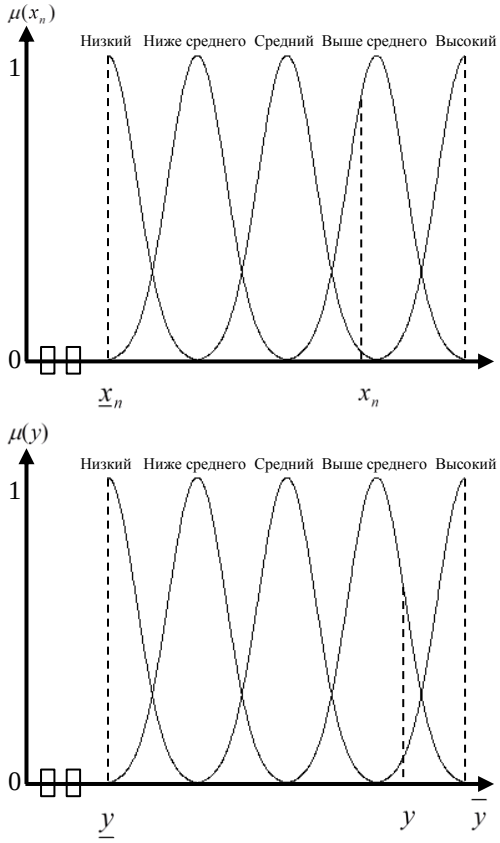
Генерация базы правил происходит для каждого столбца таблично заданной функции следующим образом: каждому значению ставим в соответствие элемент терм–множества, к которому оно относится.



$$G_{x_1} = T_2 = "Ниже\ среднего",$$
$$\mu^{T_2}(x_1) = \max\left(\mu^T(x_1)\right)$$



$$G_{x_2} = T_3 = "Средний",$$
$$\mu^{T_3}(x_2) = \max\left(\mu^T(x_2)\right)$$



$$G_{x_n} = T_4 = "Выше\ среднего",$$
$$\mu^{T_4}(x_n) = \max\left(\mu^T(x_n)\right)$$

$$G_y = T_4 = "Выше\ среднего",$$
$$\mu^{T_4}(y) = \max\left(\mu^T(y)\right)$$

Рисунок 6. Генерация нечеткой базы знаний

Для рассмотренного случая правило можно записать следующим образом:

ЕСЛИ x_1= «Ниже среднего» и x_2= «Средний» и … и x_n= «Выше среднего», ТО y= «Выше среднего».

В общем случае система нечеткого вывода будет содержать m правил нечетких продукций (табл. 2).

Таблица 2. Правила нечеткой базы знаний

№ правила	x_1	x_2	…	x_n	y
1	Высокий	Выше среднего	…	Выше среднего	Выше среднего
2	Выше среднего	Низкий	…	Выше среднего	Средний
...	…	…	…	…	…
m	Средний	Выше среднего	…	Средний	Ниже среднего

Для *k*–го столбца исходной таблицы правило будет выглядеть следующим образом:

$$П_k:\ \textit{если}\ x_1^k\ \textit{есть}\ T_{x_1^k}\ \textit{и}\ x_2^k\ \textit{есть}\ T_{x_2^k}\ \textit{и}\ \ldots\ \textit{и}\ x_i^k\ \textit{есть}\ T_{x_i^k},\ \textit{то}\ y_k\ \textit{есть}\ T_{y_k}$$

где $T_{x_i^k}$ – терм, соответствующий табличному значению $x_i^k,\ i=\overline{1,n},\ k=\overline{1,m}$,

T_{y_k} – терм, соответствующий табличному значению $y_k,\ k=\overline{1,m}$.

1.6.5 Нечеткий логический вывод

Нечеткий логический вывод осуществляется по алгоритму Larsen на основе предварительно сформулированной базе нечетких правил.

Будем считать, , что базу знаний организуют два нечетких правила вида:

П1: если x_1 есть T^1 и x_2 есть T^2, тогда y есть T^4,

П2: если x_1 есть T^3 и x_2 есть T^4, тогда y есть T^5,

где x_1 и x_2 – имена входных переменных; y – имя переменной вывода; $T=\{\text{“}H\text{”},\text{“}HC\text{”},\text{“}C\text{”},\text{“}BC\text{”},\text{“}B\text{”}\}$ – терм-множество; $\mu^H(x)$, $\mu^{HC}(x)$, $\mu^C(x)$, $\mu^{BC}(x)$, $\mu^B(x)$ – заданные функции принадлежности для входных переменных x (для y функции принадлежности аналогичные); четкое значение y_0 необходимо определить на основе приведенной информации и четких значений x_{01} и x_{02}.

Фаззификация. Функции принадлежности, определенные на входных переменных, применяются к четким (конкретным) входным переменным для определения степени истинности $\alpha_{x_1}, \alpha_{x_2}$ для предпосылок каждого правила:

$$\alpha_{x_1}^1=\mu^H(x_1),\ \alpha_{x_1}^2=\mu^{HC}(x_1),\ \alpha_{x_1}^3=\mu^C(x_1),\ \alpha_{x_1}^4=\mu^{BC}(x_1),\ \alpha_{x_1}^5=\mu^B(x_1),$$
$$\alpha_{x_2}^1=\mu^H(x_2),\ \alpha_{x_2}^2=\mu^{HC}(x_2),\ \alpha_{x_2}^3=\mu^C(x_2),\ \alpha_{x_2}^4=\mu^{BC}(x_2),\ \alpha_{x_2}^5=\mu^B(x_2).$$

Нечеткий вывод. Правила с ненулевыми степенями истинности предпосылок считаются активными в текущем процессе нечеткого вывода. Вычисляются значения:

$$\alpha_1=\alpha_{x_1}^1\wedge\alpha_{x_2}^2,\ \alpha_2=\alpha_{x_1}^3\wedge\alpha_{x_2}^4.$$

Затем находятся частные нечеткие подмножества, функция принадлежности выхода при этом масштабируется на соответствующий коэффициент:

$$\alpha_1 \cdot \mu^{BC}(y),\ \alpha_2 \cdot \mu^{B}(y).$$

Композиция. Находится итоговое нечеткое подмножество с функцией принадлежности:

$$\mu_{\sum}(y) = \alpha_1 \cdot \mu^{BC}(y) \vee \alpha_2 \cdot \mu^{B}(y).$$

Приведение к четкости. Производится дефаззификация при помощи центроидного метода:

$$y^* = \frac{\int_{\underline{y}}^{\overline{y}} y \cdot \mu_{\sum}(y)dy}{\int_{\underline{y}}^{\overline{y}} \mu_{\sum}(y)dy}.$$

1.7 Построение нечетких систем в диалоговом режиме с помощью модуля MatLab – Fuzzy Logic ToolBox

Модуль fuzzy позволяет строить нечеткие системы двух типов - Мамдани и Сугэно. В системах типа Мамдани база знаний состоит из правил вида *"Если x_1=низкий и x_2=средний, то y=высокий"*. В системах типа Сугэно база знаний состоит из правил вида *"Если x_1=низкий и x_2=средний, то $y=a_0+a_1x_1+a_2x_2$"*. Таким образом, основное отличие между системами Мамдани и Сугэно заключается в разных способах задания значений выходной переменной в правилах, образующих базу знаний. В системах типа Мамдани значения выходной переменной задаются нечеткими термами, в системах типа Сугэно - как линейная комбинация входных переменных.

1.7.1 Проектирование систем типа Мамдани

Рассмотрим основные этапы проектирования систем типа Мамдани на примере создания системы нечеткого логического вывода, моделирующей зависимость $y=f(x_1,x_2)=x_1^2+x_2^2$, в области $-2 \le x_1 \le 2$, $-2 \le x_2 \le 2$. Проектирование системы нечеткого логического вывода будем проводить на основе графического изображения указанной зависимости.

Для построения трехмерного изображения функции $y=f(x_1,x_2)=x_1^2+x_2^2$, в области $-2 \le x_1 \le 2$, $-2 \le x_2 \le 2$ составим следующую программу:

```
%Построение графика функции y=x1^2+x2^2
%в области x1є[-2,2] и x2є[-2,2].
n=5;
x1=1:1:5;
x2=1:1:5;
y=zeros(n,n);
Y=zeros(25,3)
s=0;
for j=1:n
    for i=1:n
     y(j,i)=(x1(j)-3)^2+(x2(i)-3)^2;
     s=s+1;
     Y(s,1)=x1(j)-3;
     Y(s,2)=x2(i)-3;
     Y(s,3)=y(j,i);
    end

end
Y
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target');
```

В результате выполнения программы получим графическое изображение, приведенное на рис. 7. Проектирование системы нечеткого логического вывода, соответствующей приведенному графику, состоит в выполнении следующей последовательности шагов.

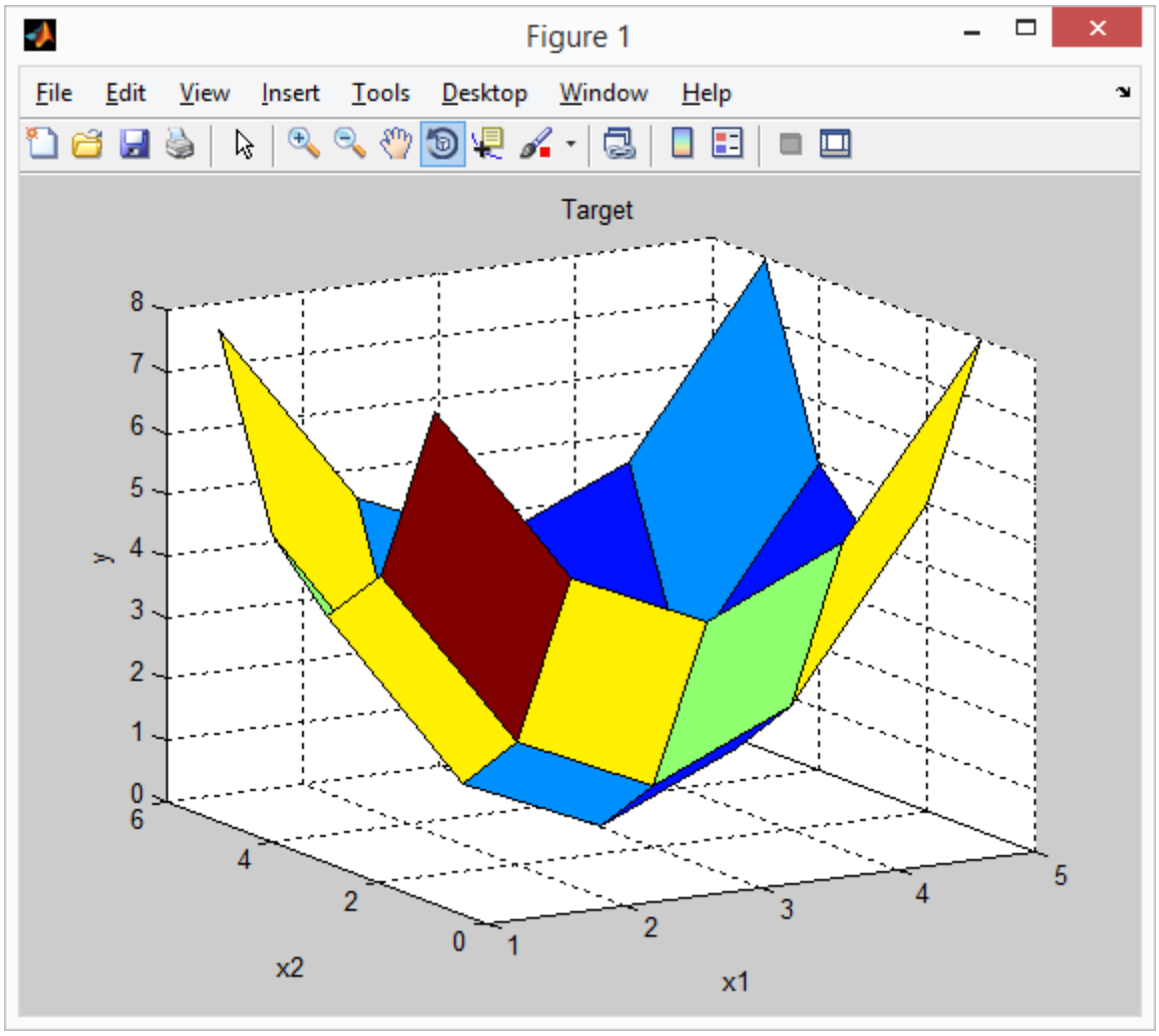


Рисунок 7. Эталонная поверхность

Также данная программа выводит таблицу значений функции в зависимости от аргументов, эти значения понадобятся для определения диапазона изменения переменной y и построения базы правил:

X1	X2	Y
-2	-2	8
-2	-1	5
-2	0	4
-2	1	5
-2	2	8
-1	-2	5
-1	-1	2
-1	0	1
-1	1	2
-1	2	5
0	-2	4
0	-1	1
0	0	0
0	1	1
0	2	4

1 -2 5
1 -1 2
1 0 1
1 1 2
1 2 5
2 -2 8
2 -1 5
2 0 4
2 1 5
2 2 8

Этапы создания нечеткой экспертной системы:

– *Шаг 1*. Для загрузки основного fis-редактора напечатаем слово **fuzzy** в командной строке. После этого откроется нового графическое окно, показанное на рис. 8.

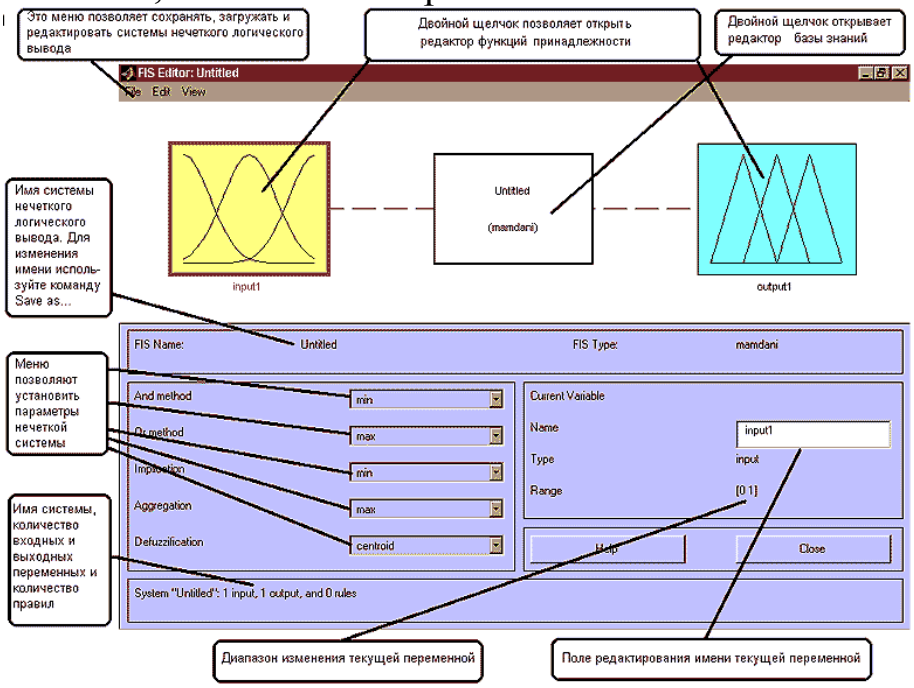


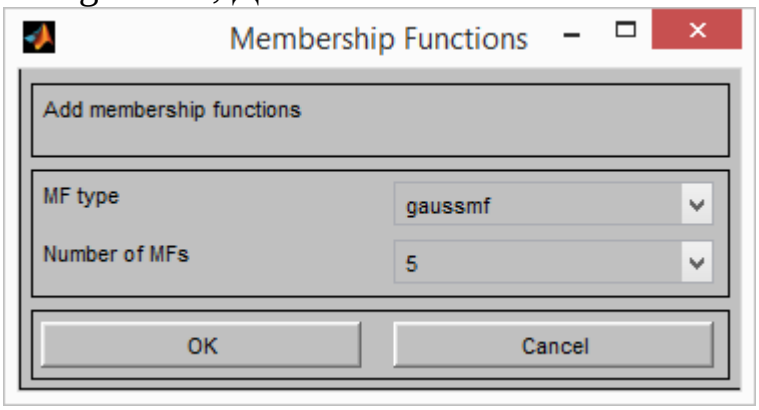
Рисунок 8. Окно редактора FIS-Editor

– *Шаг 2*. Добавим вторую входную переменную. Для этого в меню **Edit** выбираем команду **Add input**.

– *Шаг 3*. Переименуем первую входную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке **input1**, введем новое обозначение **x1** в поле редактирования имени текущей переменной и нажмем **<Enter>**.

– *Шаг 4*. Переименуем вторую входную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке **input2**, введем новое обозначение **x2** в поле редактирования имени текущей переменной и нажмем **<Enter>**.

– *Шаг 5*. Переименуем выходную переменную. Для этого сделаем один щелчок левой кнопкой мыши на блоке **output1**, введем новое обозначение **y** в поле редактирования имени текущей переменной и нажмем **<Enter>**.
– *Шаг 6*. Зададим имя системы. Для этого в меню **File** выбираем в подменю **Export** команду **To disk** и вводим имя файла, например, **first**.
– *Шаг 7*. Перейдем в редактор функций принадлежности. Для этого сделаем двойной щелчок левой кнопкой мыши на блоке **x1**.
– *Шаг 8*. Зададим диапазон изменения переменной **x1**. Для этого напечатаем **-2 2** в поле **Range** (см. рис. 7) и нажмем **<Enter>**.
– *Шаг 9*. Зададим функции принадлежности переменной **x1**. Для лингвистической оценки этой переменной будем использовать 5 термов с гауссовыми функциями принадлежности. По умолчанию для переменной создается 3 терма с треугольными функциями принадлежности. Их нужно удалить, предварительно выделив и нажав <**Del**>. Для добавления новых термов в меню **Edit** выберем команду **Add MFs...** В результате появиться диалоговое окно выбора типа и количества функций принадлежностей. Выбираем 5 и gaussmf, Далее нажимаем **<Enter>.**



– *Шаг 10*. Зададим наименования термов переменной **x1**. Для этого делаем один щелчок левой кнопкой мыши по графику первой функции принадлежности (см. рис. 9). Затем вводим наименование терма, например, **Низкий (H)**, в поле **Name** и нажмем **<Enter>** и т.д.

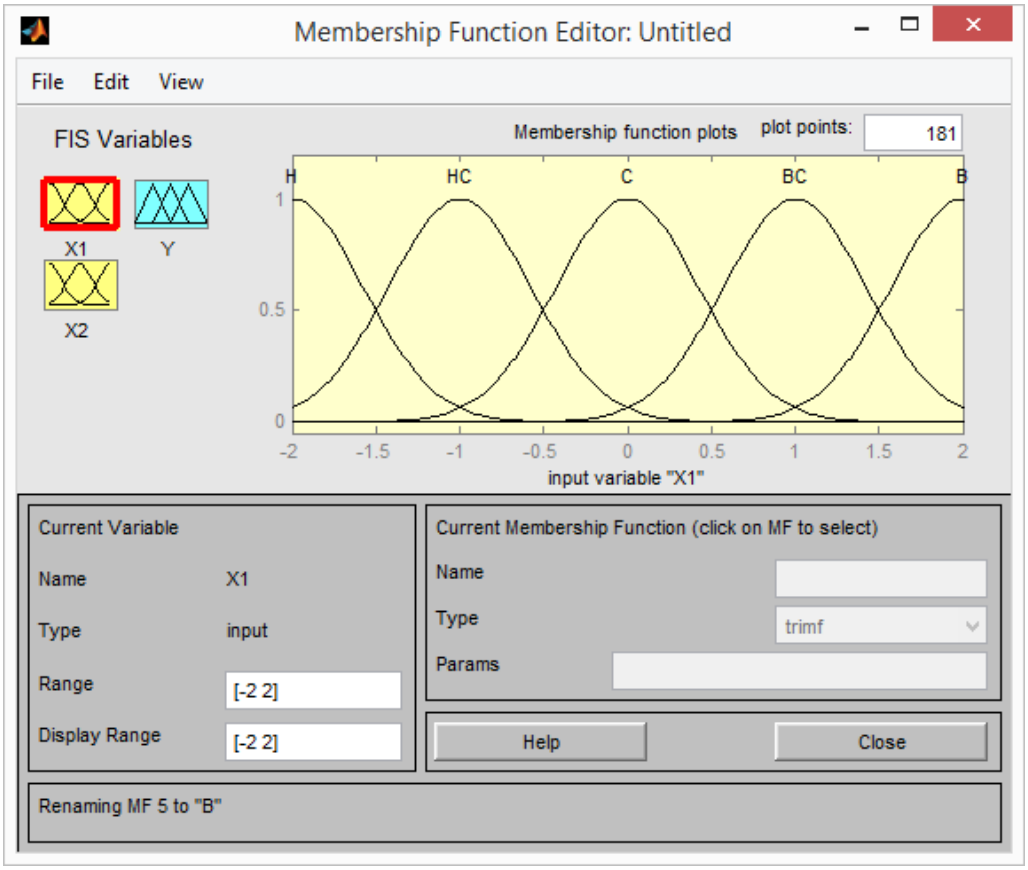


Рисунок 9. Функции принадлежности переменной x1

– *Шаг 11.* Аналогично зададим термы переменных x2 и y, при задании переменой y необходимо указывать свой диапазон изменения переменной y, в данном примере это $0 \le y \le 8$ (рис. 10 и 11).

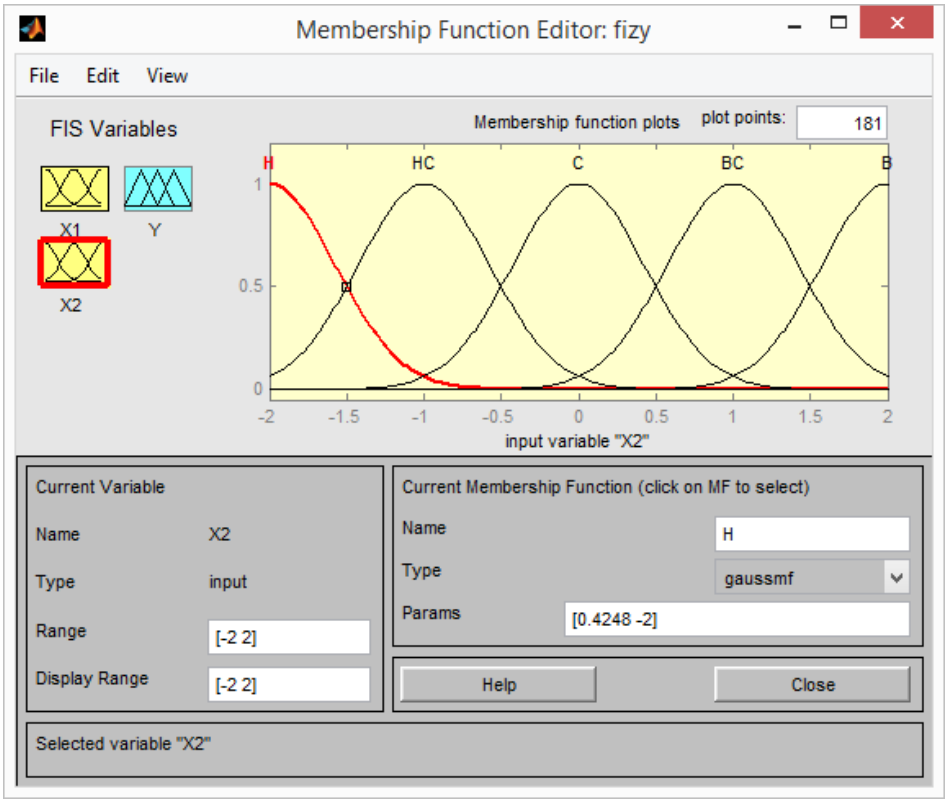


Рисунок 10. Функции принадлежности переменной x2

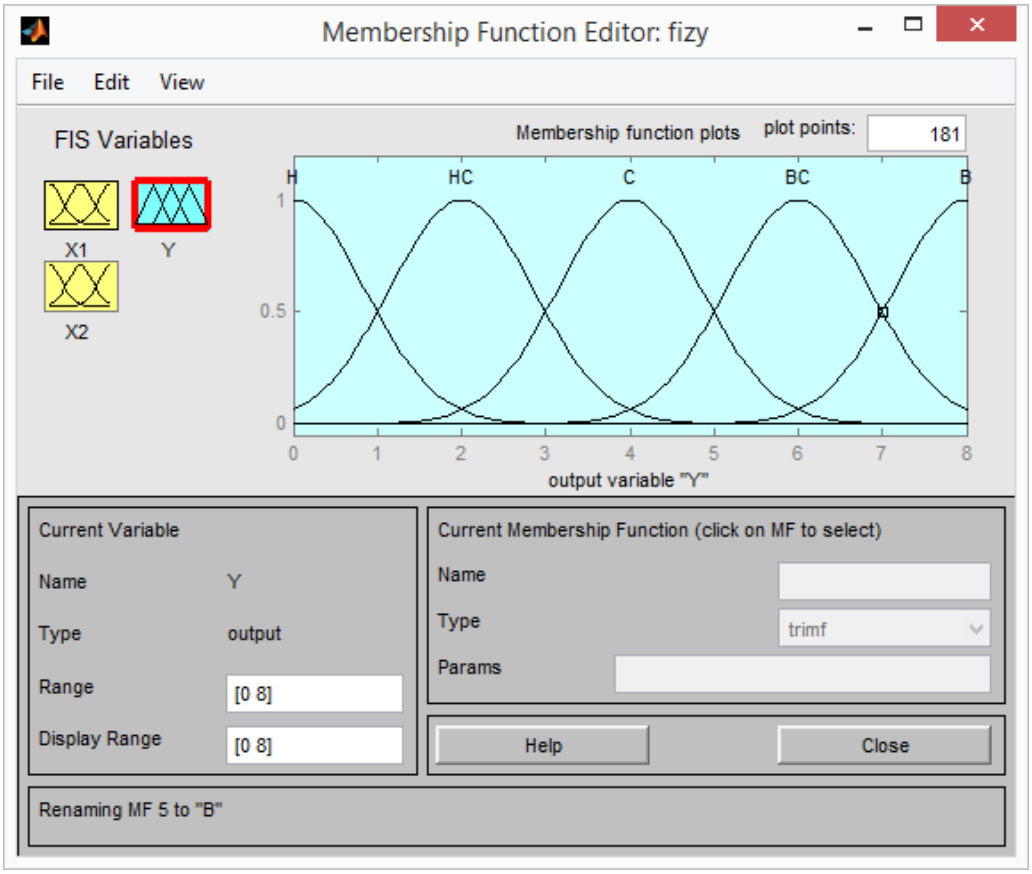


Рисунок 11. Функции принадлежности переменной y

– *Шаг 12.* Перейдем в редактор базы знаний **RuleEditor**. Для этого в меню **Edit** выберем команду **Edit rules...**.
– *Шаг 13.* На основе таблично заданный функции сформируем базу нечетких правил, в результате должно быть 25 правил, для каждой строки таблицы. Для ввода правила необходимо выбрать в меню соответствующую комбинацию термов и нажать кнопку **Add rule**. На рис. 12 изображено окно редактора базы знаний после ввода правил. Число, приведенное в скобках в конце каждого правила представляет собой весовой коэффициент соответствующего правила.

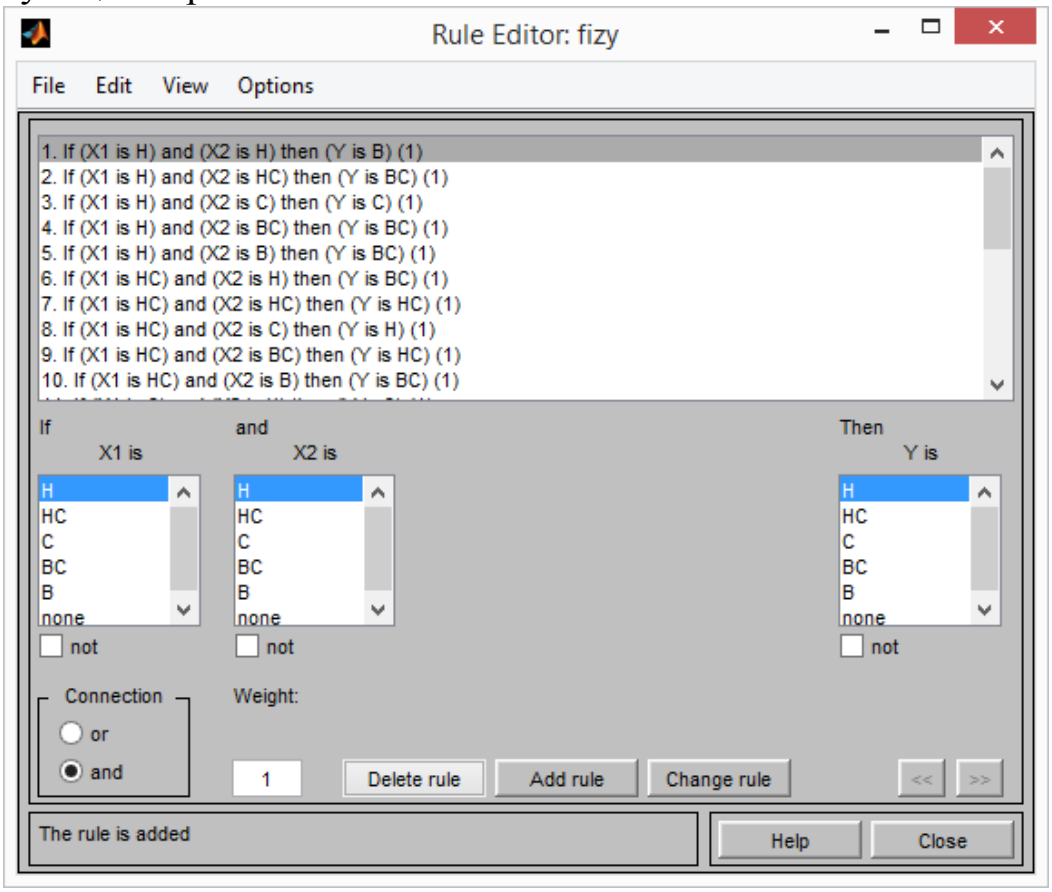


Рисунок 12. База знаний в RuleEditor

– *Шаг 14.* Сохраним созданную систему. Для этого в меню **File** выбираем в подменю **Export** команду **To disk**.

На рис. 13 приведено окно визуализации нечеткого логического вывода. Это окно активизируется командой **View rules...** меню **View.** В поле **Input** указываются значения входных переменных, для которых выполняется логический вывод.

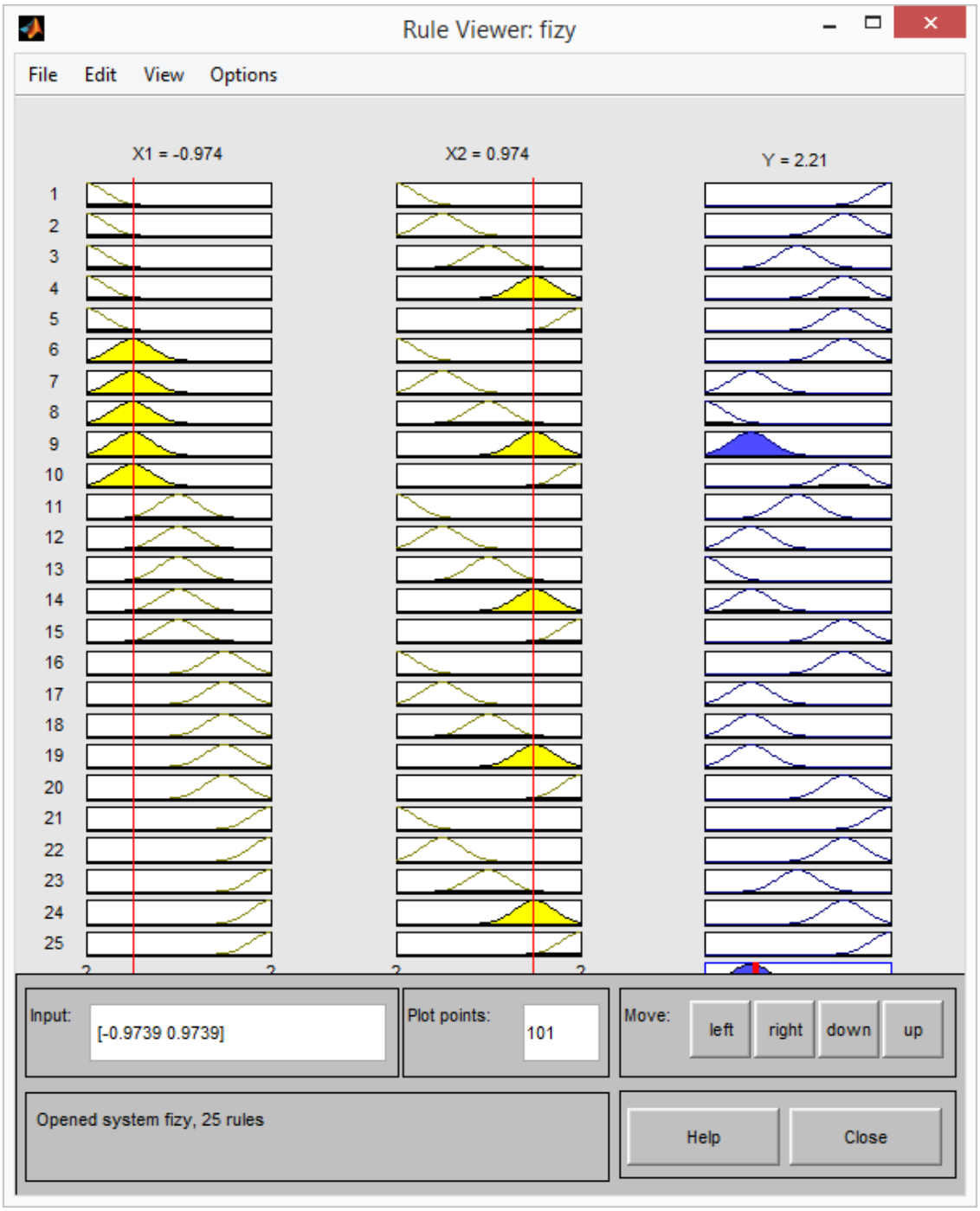


Рисунок 13. Визуализация нечеткого логического вывода в RuleViewer

На рис. 14 приведена поверхность “входы-выход”, соответствующая синтезированной нечеткой системе. Для вывода этого окна необходимо использовать команду **View surface...** меню **View.** Сравнивая поверхности на рис. 7 и на рис. 14 можно сделать вывод, что нечеткие правила достаточно хорошо описывают сложную нелинейную зависимость.

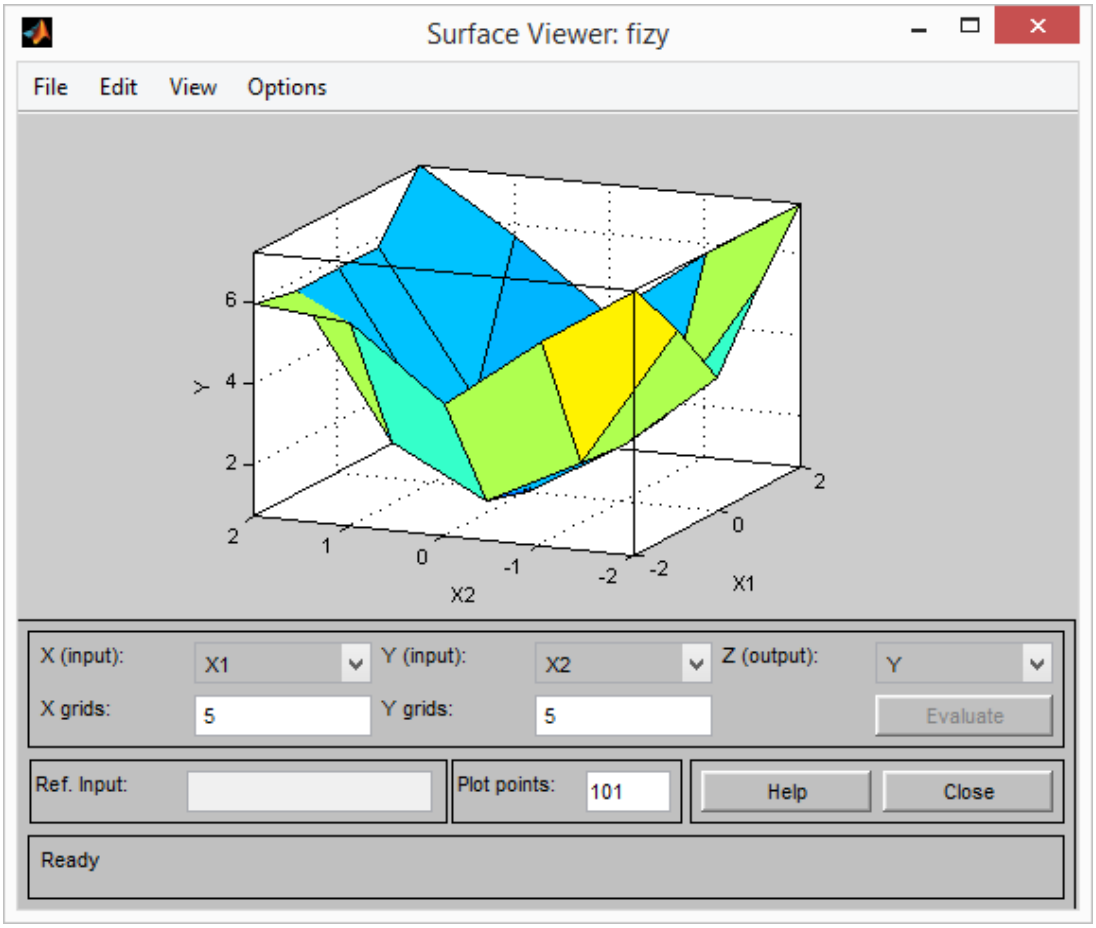


Рисунок 14. Поверхность “входы-выход” в окне SurfaceViwer

1.7.2 Оценка погрешности аппроксимации

Для оценки погрешности аппроксимации необходимо находить значение получаемое спроектированной нечеткой экспертной системой в зависимости от значений входных аргументов, для того используется команда evalfis.

Далее приведен фрагмент кода для вычисления погрешности аппроксимации:

```
%Вычисление погрешности
a=readfis('fuz');
s1=0;
s2=0;
  for (i=1:5)
    for (j=1:5)
        y0=(i-3)^2+(j-3)^2;
        y1=evalfis([(i-3),(j-3)], a);
        s1=s1+((y0-y1)^2);
        s2=s2+(y0^2);
    end
  end
 s1=s1^(1/2);
 d=s1/(s2^(1/2));
```

```
ans=d
```

В рассмотренном примере относительная погрешность составила δ=0.1830≈18%

2 Нейронные сети

Принято считать, что человеческий мозг - это естественная нейронная сеть, а модель мозга - это искусственная нейронная сеть или просто нейронная сеть.

Искусственные нейронные сети (ИНС) – вид математических моделей, которые строятся по принципу организации и функционирования их биологических аналогов – сетей нервных клеток (нейронов) мозга.

В основе их построения лежит идея о том, что нейроны можно моделировать довольно простыми автоматами (называемыми искусственными нейронами), а вся сложность мозга, гибкость его функционирования и другие важнейшие качества определяются связями между нейронами.

2.1 Модель нейрона

Человеческий мозг – естественная нейронная сеть, состоит и клеток называемых нейронами. Сама по себе клетка состоит из ядра и внешней оболочки. Каждый нейрон имеет уровень активации, лежащий в диапазоне между максимумом и минимумом, следовательно, в отличие от булевой логики, существует более чем два уровня активации (рис. 15).

Для увеличения или уменьшения активности данного нейрона другими нейронами существуют так называемые *синапсы*. Они переносят величину активности от нейрона-отправителя к нейрону-получателю. Если синапс является возбуждающим, то величина активности нейрона-отправителя увеличивает активность нейрона-получателя. Если синапс является тормозящим, то величина активности нейрона-отправителя уменьшает активность нейрона-получателя. Синапсы различаются не только по признаку торможения или возбуждения нейрона-получателя, но также и по суммарному воздействию (синаптическая мощность). Выходной сигнал каждого нейрона передается по так называемому *аксону*, который заканчивается более чем 10000 синапсами, влияющими на другие нейроны.

Рассмотренная модель нейрона лежит в основе большинства сегодняшних применений нейронной сети. Данная модель является лишь очень грубым приближением действительности.

Рисунок 15. Упрощенная схема человеческого нейрона.

2.1.1 Математическая модель нейрона

Множество математических моделей нейрона может быть построено на базе простой концепции строения нейрона. На рис. 16 показана наиболее общая схема. Так называемая суммирующая функция объединяет все входные сигналы x_i, которые поступают от нейронов-отправителей. Значением такого объединения является взвешенная сумма, где веса w_i представляют собой синаптические мощности. Возбуждающие синапсы имеют положительные веса, а тормозящие синапсы - отрицательные веса. Для выражения нижнего уровня активации нейрона к взвешенной сумме прибавляется компенсация (смещение) Θ.

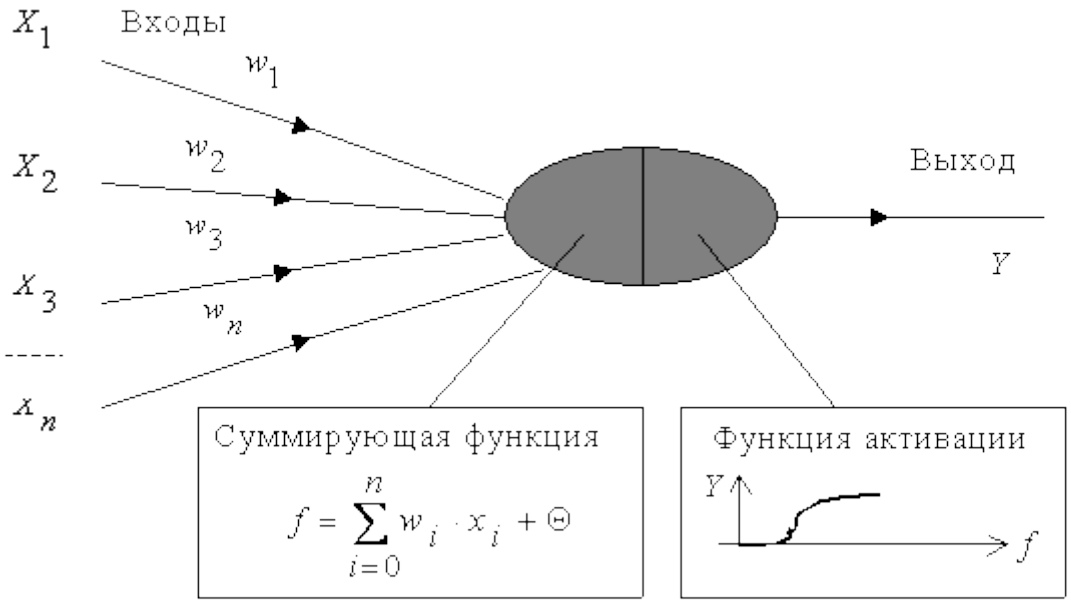


Рисунок 16. Простая математическая модель нейрона.

Так называемая функция активации рассчитывает выходной сигнал нейрона *Y* по уровню активности *f*. Функция активации обычно является сигмоидной, как показано в правой нижней рамке на рис. 16. Другими возможными видами функций активации являются линейная и радиально-симметричная функции, показанные на рис. 17.

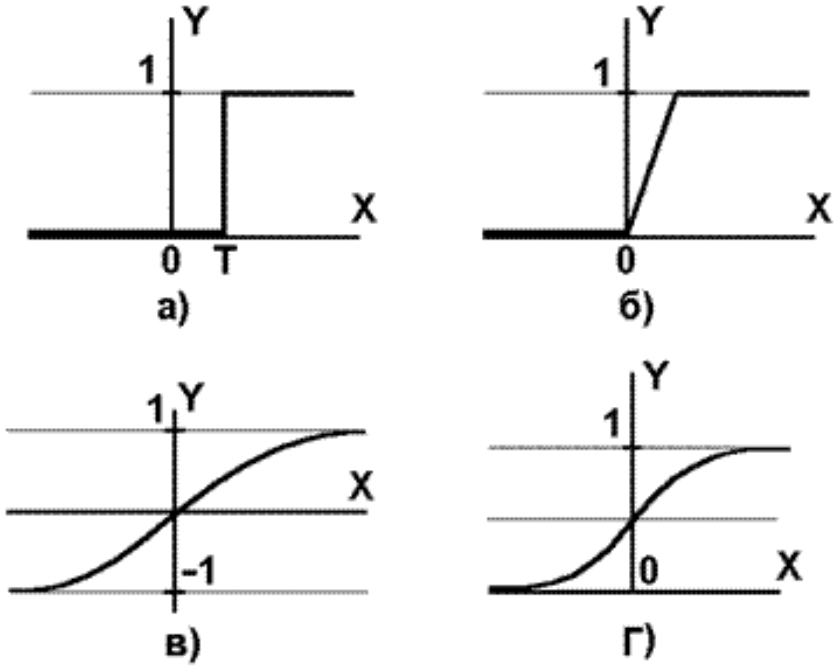


Рисунок 17. Функции активации нейрона:
а) функция единичного скачка; б) линейный порог (гистерезис);
в) сигмоид – гиперболический тангенс; г) сигмоид

2.2 Архитектуры (структуры) нейронных сетей.

Выделяют несколько стандартных архитектур, из которых путем вырезания лишнего или добавления строят большинство используемых сетей (рис. 18).

Можно выделить следующие базовые архитектуры:

- полносвязные сети;
- слабосвязанные сети;
- многослойные сети.

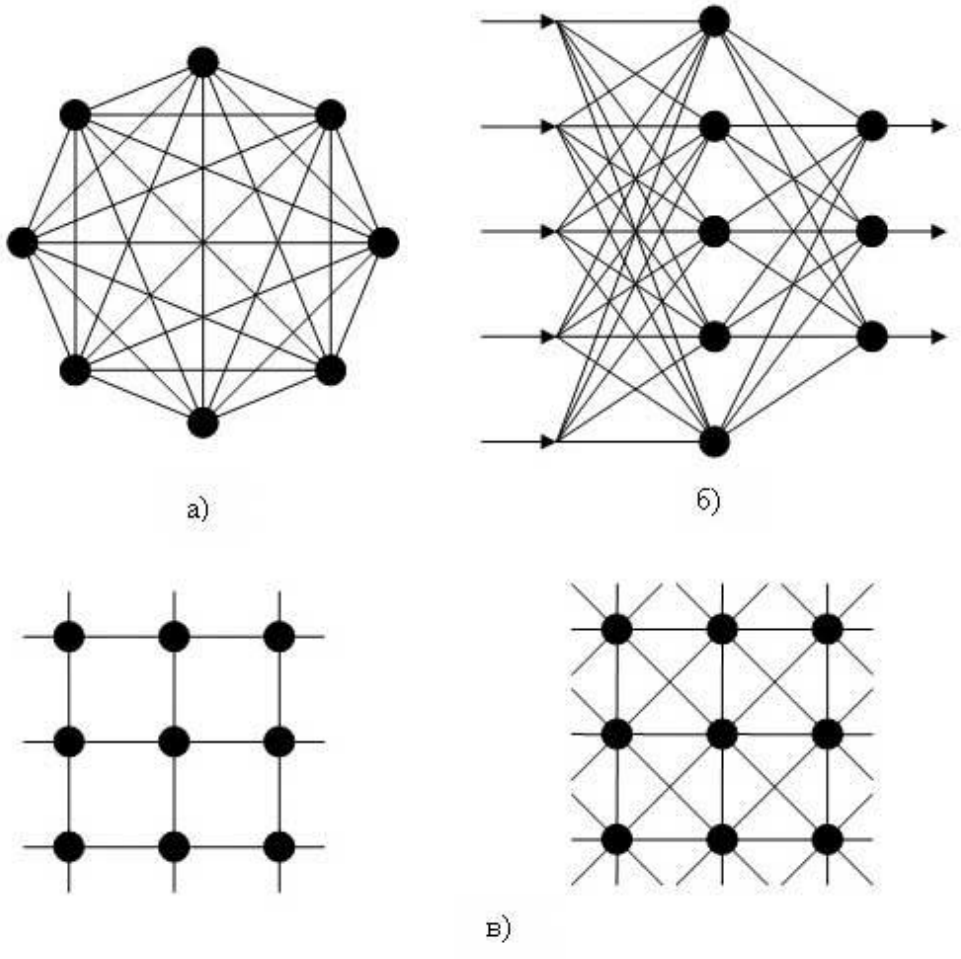


Рисунок 18. Виды нейронных сетей:
а) *полносвязные;* б) *многослойные;* в) *слабосвязные*

В *полносвязных нейронных сетях* каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигнала-

ми сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

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

В *многослойных нейронных сетях* (их часто называют *персептронами*) нейроны объединяются слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях. В общем случае сеть состоит из нескольких слоев, пронумерованных слева на право.

Внешние входные сигналы подаются на входы нейронов входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько так называемых *скрытых слоев*. В свою очередь, среди многослойных сетей выделяют:

- Сети прямого распространения (feedforward networks)
- Сети с обратными связями (recurrent networks)

Сети прямого распространения (feedforward networks) – сети без обратных связей (рис. 19). В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя.

Если не оговорено противное, то каждый выходной сигнал n-го слоя передастся на вход всех нейронов (n+1)-го слоя; однако возможен вариант соединения n-го слоя с произвольным $(n+p)$-м слоем.

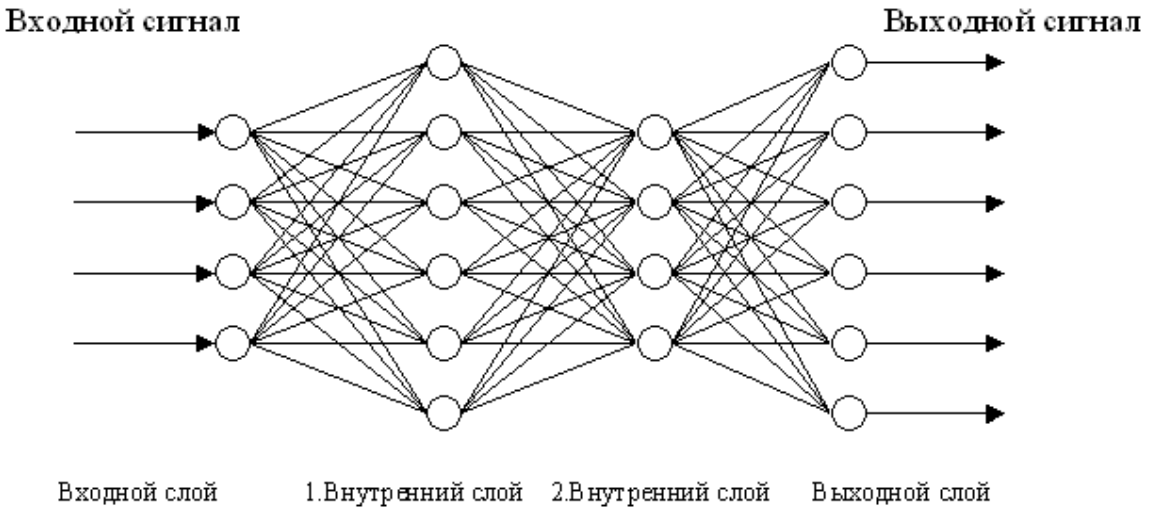


Рисунок 19. Структура слоистой нейронной сети

В сетях с обратными связями информация передается с последующих слоев на предыдущие. Следует иметь в виду, что после введения обратных связей сеть уже не просто осуществляет отображение множества входных векторов на множество выходных, она превращается в динамическую систему и возникает вопрос об ее устойчивости.

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные, на сегодняшний день, конфигурации.

Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами:

- возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;
- сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тромозящих и др.) также способствует усилению мощи НС.

Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки.

Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

2.3 Обучение нейронных сетей.

2.3.1 Функционирование НС

Процесс функционирования НС, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой НС, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными). Этот этап называется обучением НС, и от того, насколько качественно он будет выполнен, зависит способность сети решать поставленные перед ней проблемы во время эксплуатации.

После завершения обучения нейронная сеть готова к использованию. Это - рабочая фаза.

На этапе обучения кроме параметра качества подбора весов важную роль играет время обучения. Как правило, эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса

2.3.2 Правило Хебба

Как же обучается нейронная сеть? Как правило, это демонстрируется на примере известных собак Павлова. Когда он показывал собакам еду, у них выделялась слюна. В собачьих клетках устанавливались звоночки. Когда звонил звоночек, у собак не выделялась слюна, т. е. они не видели связи между звонком и едой. Тогда Павлов стал обучать собак иначе, каждый раз используя звоночек при предъявлении пищи. После этого, даже при отсутствии еды, наличие звоночка вызывало у собак слюну.

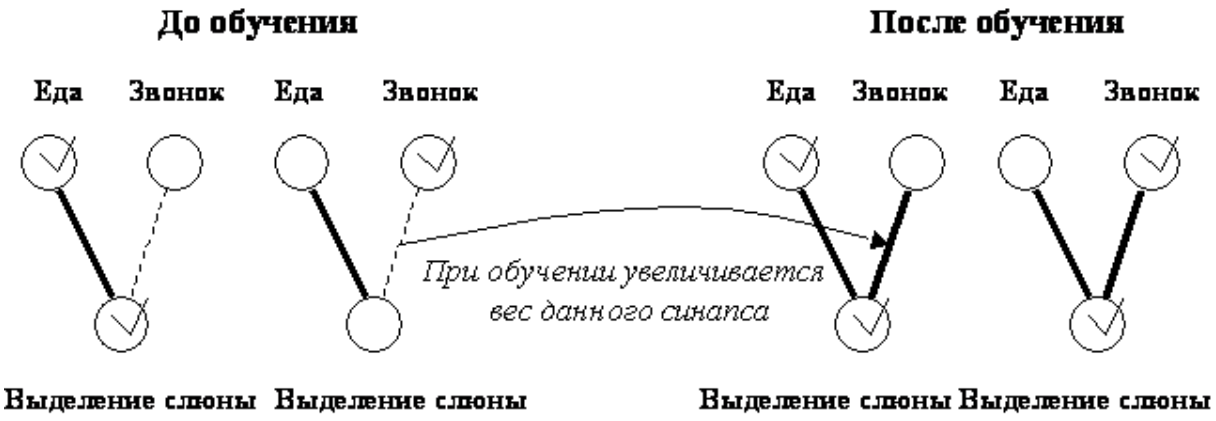


Рисунок 20. Принцип эксперимента Павлова над собаками

На рис. 20 показано, как простая модель нейрона может быть представлена на примере собаки Павлова. Имеется два входных нейрона: один из них соответствует тому, что собака видит пищу, другой - наличию звонка. Оба входных нейрона имеют связи с выходным нейроном. Эти связи соответствуют синапсам, а толщина линий - весам синапсов. Перед обучением собака реагирует лишь на еду, но не на звонок. Следовательно, линия между левым входным и выходным нейронами является жирной, в то время как линия между правым входным и выходным нейронами является очень тонкой. Совершенно очевидно, что звонок при предъявлении пищи вырабатывает ассоциацию между ним и едой. Следовательно, правая линия также становится толще, поскольку увеличивается вес синапса.

На основании этих наблюдений Хебб в 1949 году предложил следующее обучающее правило:

1. Увеличивать вес активного входа нейрона, если выход этого нейрона должен быть активным.
2. Уменьшить вес активного входа нейрона, если выход этого нейрона не должен быть активным.

Это правило, названное правилом Хебба, предшествует всем обучающим правилам.

2.3.3 Алгоритмы обучения нейронных сетей

Существует великое множество различных алгоритмов обучения, которые однако делятся на два больших класса: детерминистские

и стохастические. В первом из них подстройка весов представляет собой жесткую последовательность действий, во втором – она производится на основе действий, подчиняющихся некоторому случайному процессу.

Алгоритмы обучения бывают 3-х видов:

- Обучение с учителем.
- Обучение с поощрением.
- Обучение без учителя.

При обучении с учителем сети предъявляется набор обучающих примеров. Каждый обучающий пример представляют собой пару: вектор входных значений и желаемый выход сети. В ходе обучения весовые коэффициенты подбираются таким образом, чтобы по этим входам давать выходы максимально близкие к правильным.

При обучении с поощрением сети не указывается точное значение желаемого выхода, однако, ей выставляется оценка хорошо она поработала или плохо.

При обучении без учителя сети предъявляются некоторые входные векторы и в ходе их обработки в ней происходят некоторые процессы самоорганизации, приводящие к тому, что сеть становится способной решать какую-то задачу.

2.3.4 Алгоритм обратного распространения ошибки

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС.

Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, минимизирующем ошибку на выходе сети.

В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС.

Наиболее приемлемый вариант обучения в этом случае – распространение сигналов ошибки от выходов НС к ее входам, в направлении,

обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название ***обратного распространения.***

Идея метода заключается в минимизации функции ошибки, зависящей от весовых коэффициентов синаптических связей, методом градиентного спуска:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2$$

где: $y_{j,p}^{(N)}$ – реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа;

d_{jp} – идеальное (желаемое) выходное состояние этого нейрона.

Данный подход имеет недостаток: применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов застабилизируются, необходимо их заново проинициализировать случайными значениями, чтобы начать градиентный спуск из новой точки.

Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

2.4 Применение нейронных сетей к аппроксимации функций

Для решения задачи аппроксимации функции можно использовать многослойную нейронную сеть. При проектировании сети необходимо определить количество слоев и количество элементов (нейронов) в каждом слое. Выбор архитектуры нейронной сети зависит от сложности задачи и основывается на опыте разработчика. При моделировании нейронной сети выделяются следующие этапы:

1. создание сети;
2. обучение сети;
3. тестирование сети;
4. использование сети для решения поставленной задачи.

Многослойные нейронные сети с нейронами с функцией активации сигмоидального типа с математической точки зрения выполняют аппроксимацию функции нескольких переменных $X \in R^N$ во множество выходных переменных $Y \in R^M$. Поскольку сигмоидальная функция, играющая роль функции активации нейронов, имеет ненулевое значение на всем диапазоне входных данных, то в преобразовании сетью входных данных в выходные участвуют многие (если не все) ее нейроны. Вследствие этого аппроксимация сигмоидальными (и, естественно, линейными) нейронами называется глобальной аппроксимацией.

Для аппроксимации функций сеть имеет двухслойную структуру, нулевой слой составляют нейроны соответствующие аргументам функции, первый (скрытый) слой содержит произвольной количество нейронов (чем больше тем лучше), и выходной — один или несколько нейронов соответствующие значениям функции. На рис. 21 представлена структурная схема сети с одним выходным нейроном, предназначенная для аппроксимации функции n аргументов $y=f(x_1,x_2,\ldots,x_n)$.

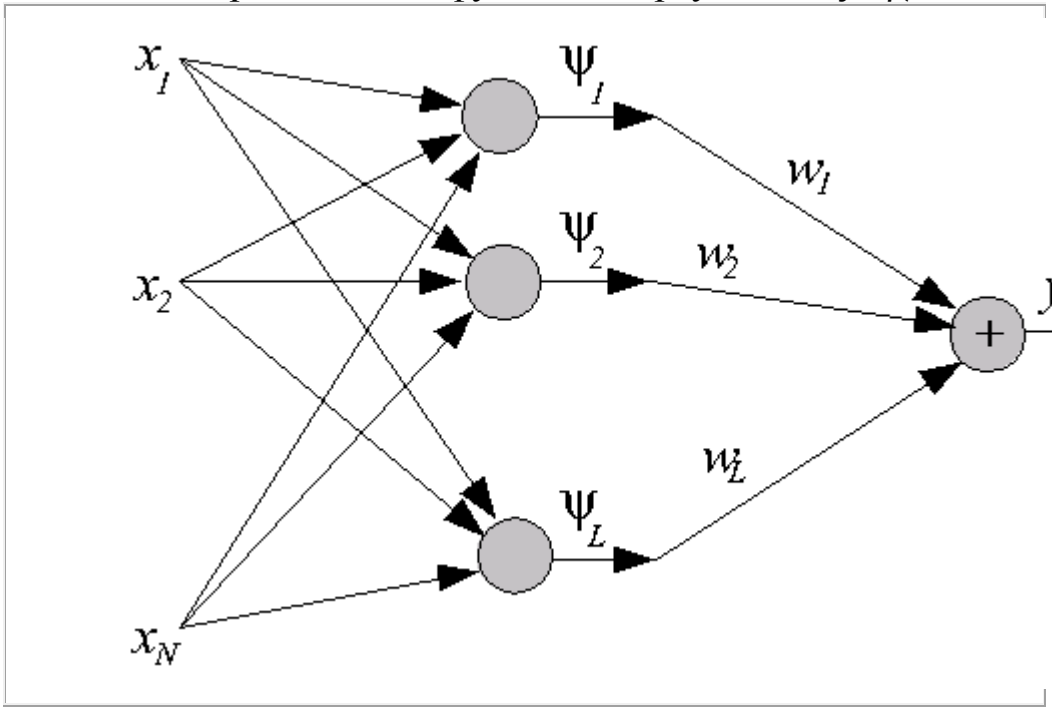


Рисунок 21. Структурная схема радиальной сети с одним выходным нейроном

2.5 *Нейронные сети в пакете MatLab. Графический интерфейс пользователя*

Нейронные сети (NN - Neural Networks) широко используются для решения разнообразных задач. Среди развивающихся областей применения NN - обработка аналоговых и цифровых сигналов, синтез и идентификация электронных цепей и систем. Основы теории и технологии применения NN широко представлены в пакете MATLAB. В этой связи особо следует отметить версию пакета - MATLAB 6.0, где впервые представлен GUI (Graphical User Interface - графический интерфейс пользователя) для NN - NNTool.

Графический интерфейс пользователя NNTool позволяет выбирать структуры NN из обширного перечня и предоставляет множество алгоритмов обучения для каждого типа сети.

2.5.1 Управляющие элементы NNTool

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

>> nntool

после этого появится главное окно NNTool, именуемое "Окном управле-ния сетями и данными" (Network/Data Manager) (рис. 22).

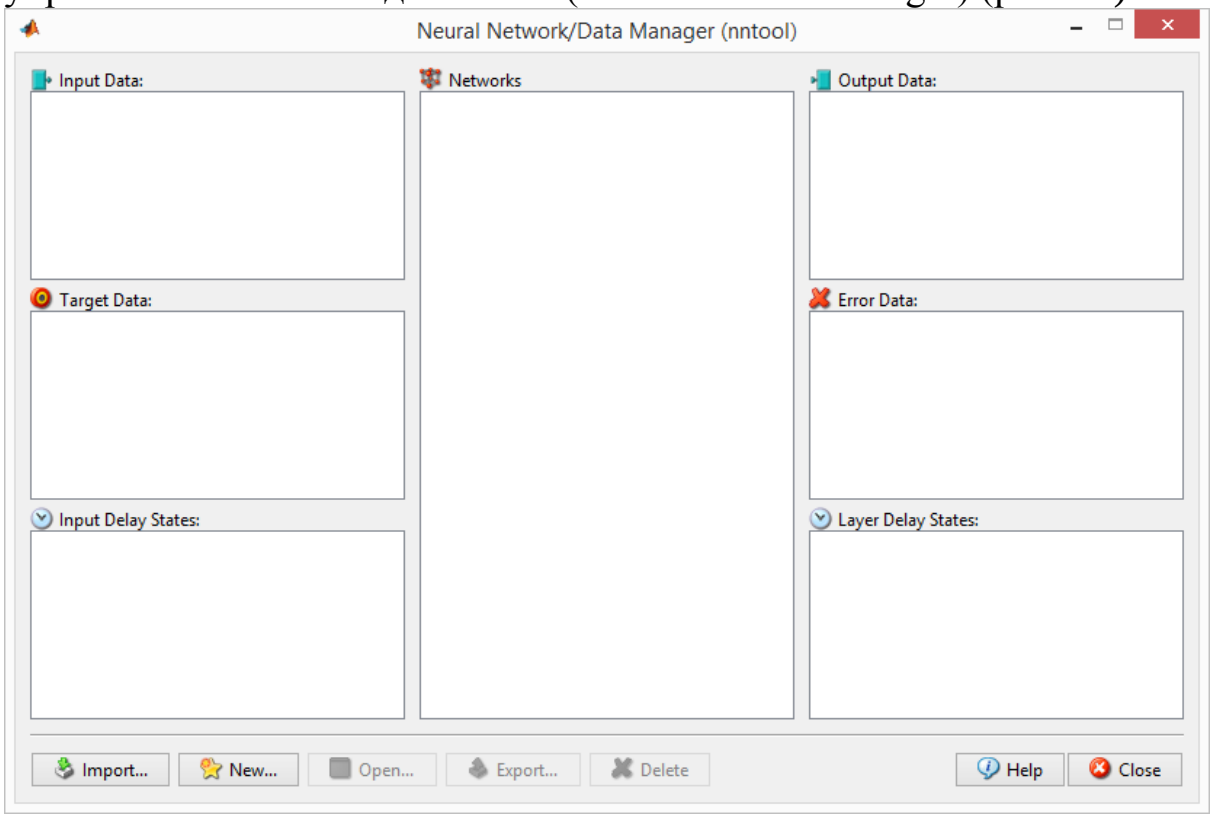


Рисунок 22. Главное окно NNTool

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;
- 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 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 задачи аппроксимации.

Одним из самых замечательных свойств нейронных сетей является способность аппроксимировать и, более того, быть универсальными аппроксиматорами. Сказанное означает, что с помощью нейронных цепей можно аппроксимировать сколь угодно точно непрерывные функции многих переменных. Рассмотрим пример.

2.5.2 Создание и обучение сети в графическом интерфейсе NNTool

Рассмотрим основные этапы создания нейронной сети, моделирующей зависимость $y=f(x_1,x_2)=x_1^2+x_2^2$, в области $-2\le x_1\le 2$, $-2\le x_2\le 2$. Создание и обучение нейронной сети вывода будем проводить на основе значений функции и графического изображения указанной зависимости.

Для получения обучающей выборки и построения трехмерного изображения функции $y=f(x_1,x_2)=x_1^2+x_2^2$, в области $-2\le x_1\le 2$, $-2\le x_2\le 2$ составим следующую программу:

```
%Построение графика функции y=x1^2+x2^2
%в области x1є[-2,2] и x2є[-2,2].
n=5;
x1=1:1:5;
x2=1:1:5;
y=zeros(n,n);
s=0;
Input=zeros(2,25);
Target=zeros(1,25);
for j=1:n
    for i=1:n
     y(j,i)=(x1(j)-3)^2+(x2(i)-3)^2;
     s=s+1;
     Input(1,s)=x1(j)-3;
     Input(2,s)=x2(i)-3;
     Target(1,s)=(x1(j)-3)^2+(x2(i)-3)^2;
    end
end
Input
Target
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target');
```

Выполение данной программы позволяет получить векторы входов, целевых значений и график функции

```
Input =
-2  -2 -2  -2  -2  -1  -1  -1  -1  -1   0   0 0 0 0 0   1   1 1 1 1   2   2 2 2 2
-2  -1  0   1   2  -2  -1   0   1   2 -2  -1  0 1 2  -2  -1 0 1 2  -2  -1 0 1 2

Target =
8  5  4  5  8  5  2  1  2  5  4  1  0  1  4  5  2  1  2  5  8  5  4  5  8
```

Эти данные необходимы для создания и обучения сети.

В командном окне Matlab введите команду nntool, откроется графический интерфейс для работы с нейронными сетями:

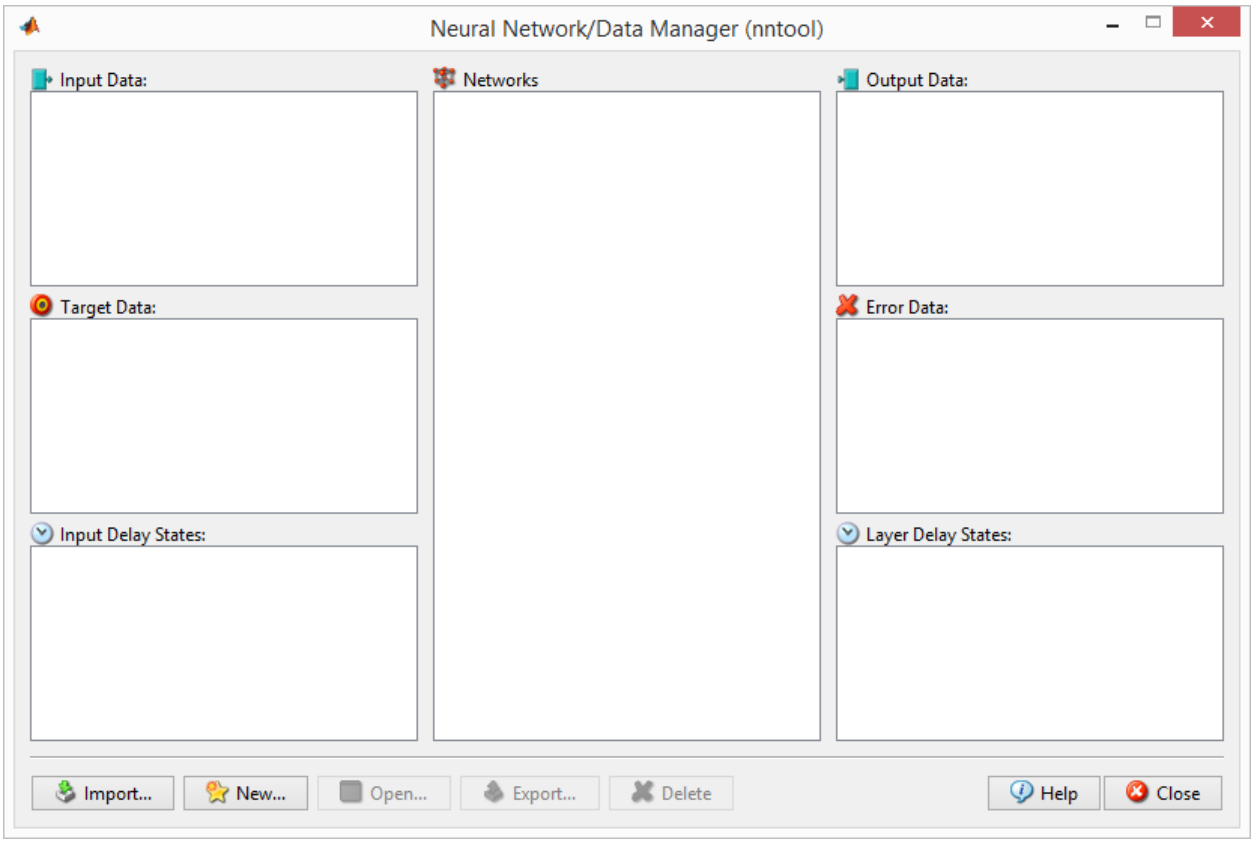


Рисунок 23. Главное окно NNTool

Создадим новую сеть, нажав кнопку “New”. Выберем персептрон (Feed-Forward Back Propagation) с 25 сигмоидными (TANSIG) нейронами скрытого слоя и одним линейным (PURELIN) нейроном выходного слоя.

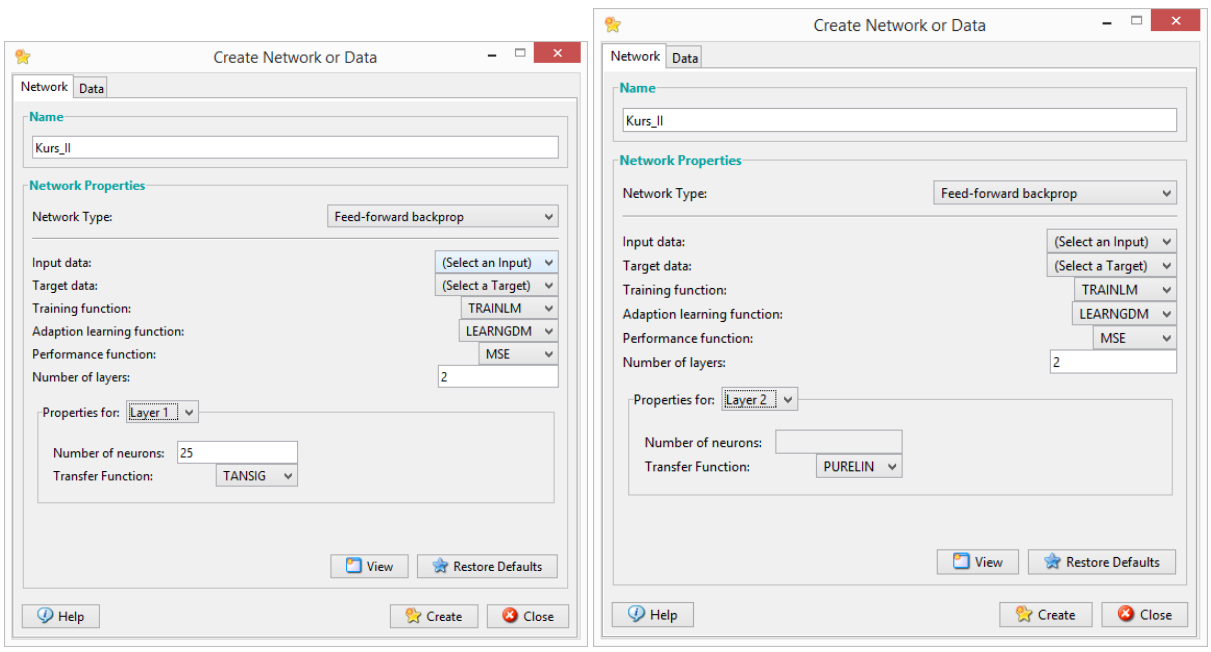


Рисунок 24. Создание сети

Также зададим входные и целевые значения, полученные ранее, для этого нужно перейти на вкладку «Data» (входные и целевые значения должный быть векторами, для этого по синтаксису Matlab нужно заключить значения в квадратные скобки и разделить два входных вектора точкой с запятой):

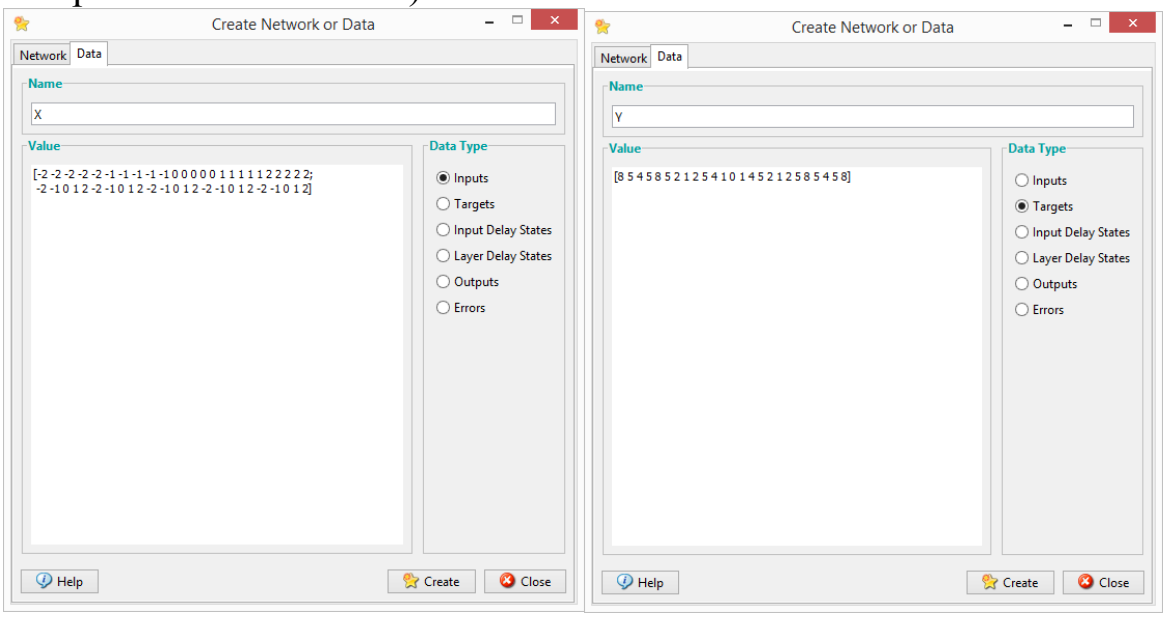


Рис. 18. Входы сети и целевые значения

После ввода исходных данных на вкладке «Network» указываем X и Y:

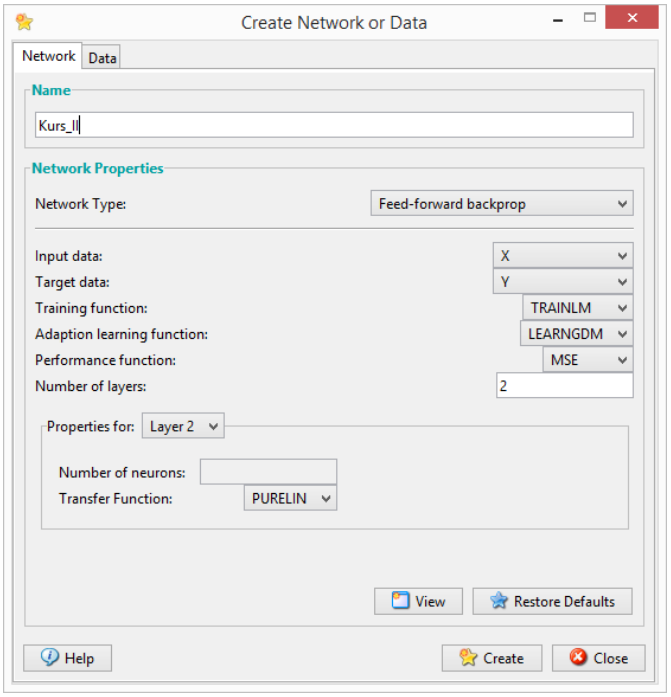


Рисунок 25. Создание сети

Енажав «Create» завершается процесс создания сети. Структура полученной сети:

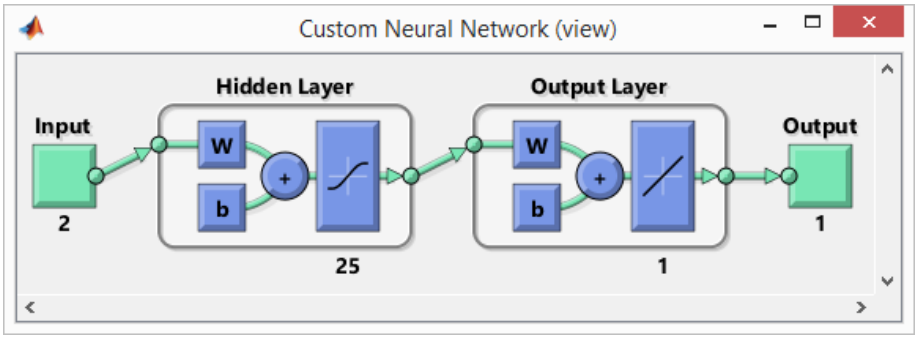


Рисунок 26. Вид созданной сети

Следующий этап это обучение сознанной сети. Обучение будем производить, используя алгоритм обратного распространения ошибки, с минимизации функции ошибки по методу Левенберга-Маркардта (Levenberg-Marquardt), который реализует функция TRAINLM. Функция ошибки - MSE. Для этого необходимо выполнить двойной клик по созданной сети:

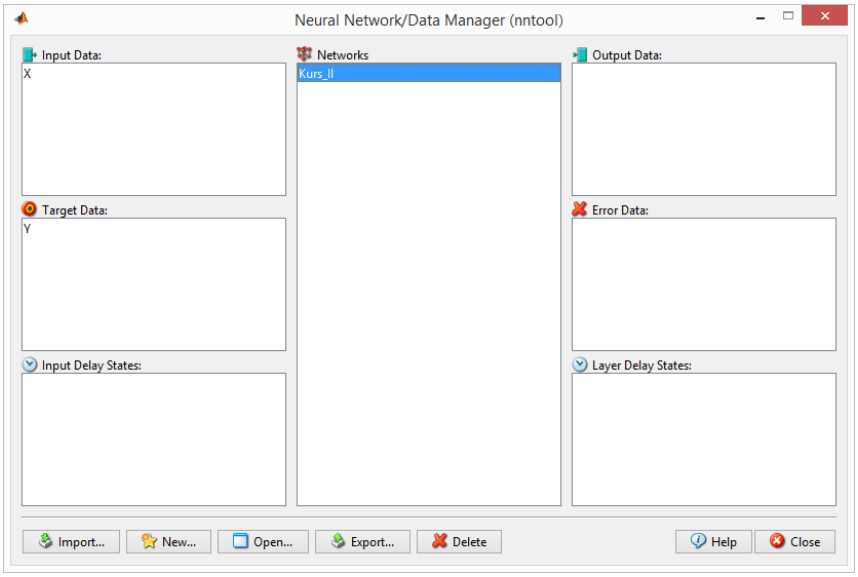


Рисунок 27. Главное окно NNTool, с созданной сетью

Откроется окно свойств сети:

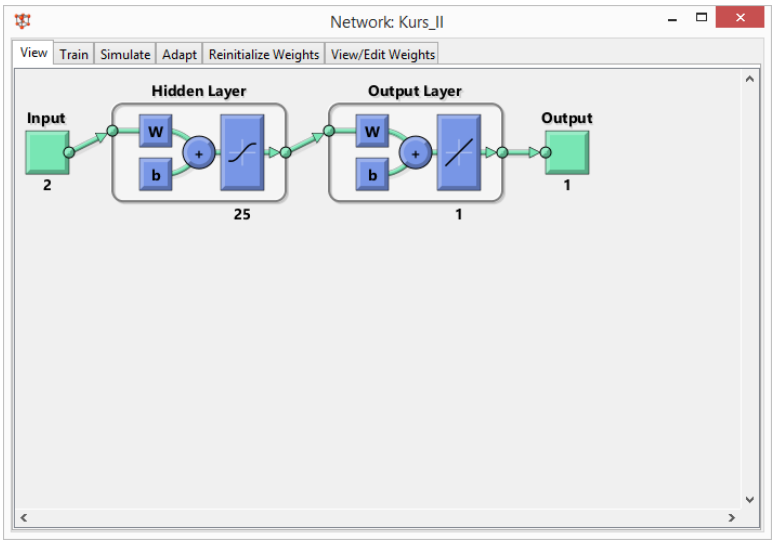


Рисунок 28. Структура созданной сети

Нужно перейти на вкладку обучения «Train» и указать в качестве обучающей выборки значения векторов входов и целей:

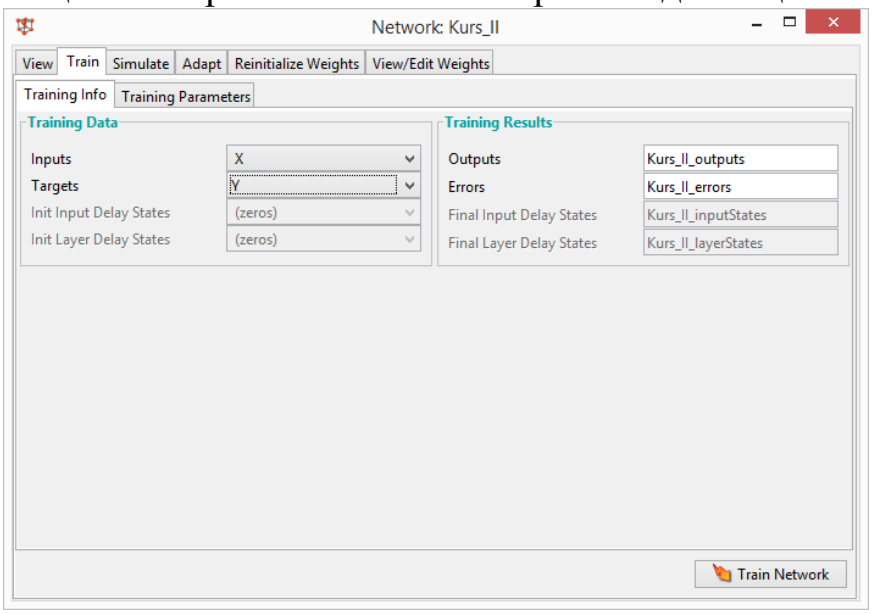


Рисунок 29. Обучение сети

При необходимости можно изменить параметры обучения сети на вкладке «Training Parameters»:

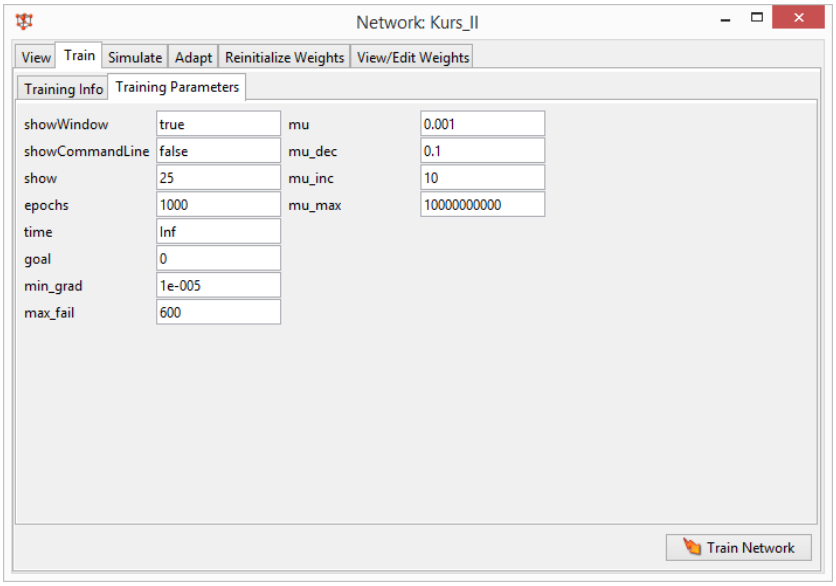


Рисунок 30. Параметры обучения сети

Если в результате обучения сети не удается получить желаемых результов, то в этом случае необходимо заново задать весовые коэффициенты на вкладке «Reinitialize Weights» нажав «Initialize Weights», так же можно изменить диапазон возможных значений весовых коэффициентов указав из в окне «Input Ranges» и нажав «Set Input Ranges»:

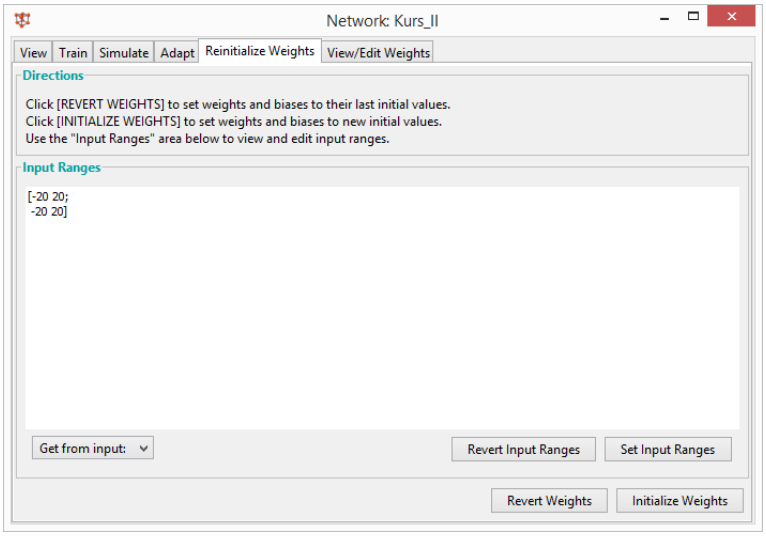


Рисунок 31. Инициализация весовых коэффициентов

После того как настроены все параметры обучения можно перейти, непосредственно, к обучению, для этого необходимо нажать «Train Network» на вкладке «Train». Результаты обучения представлены ниже:

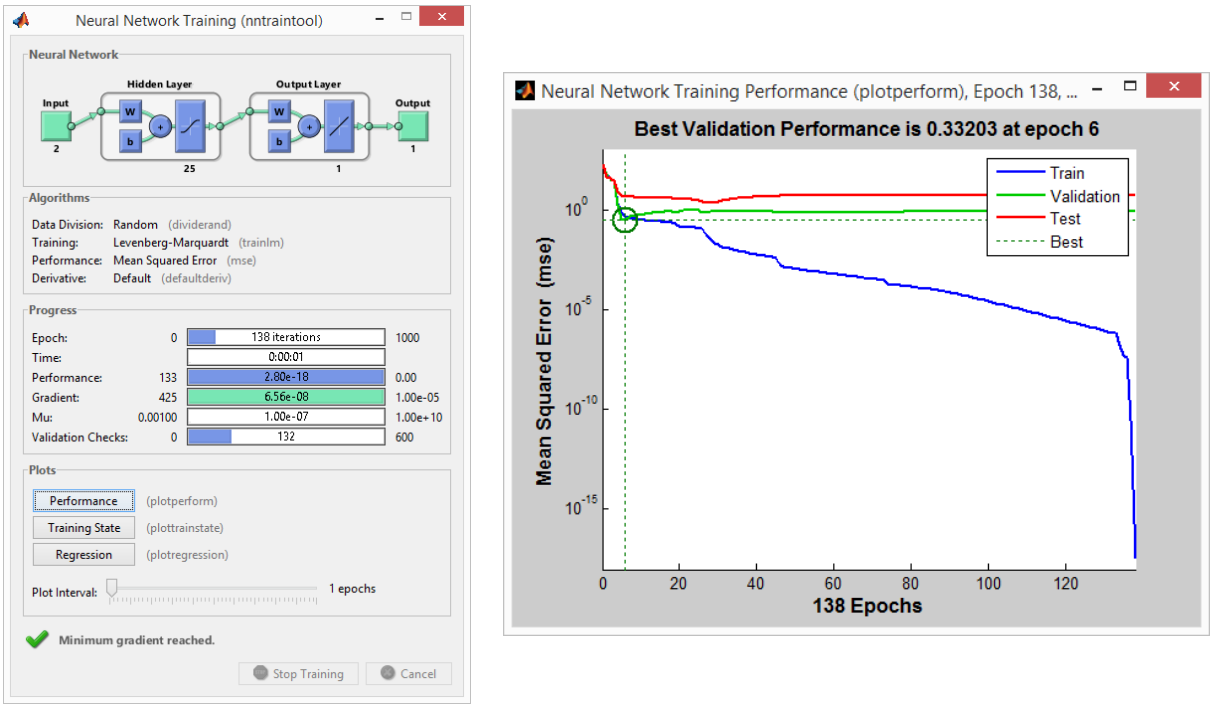


Рисунок 32. Результаты обучение сети

После обучения сети необходимо проверить насколько хорошо сеть приближает функцию, для этого нужно перейти на вкладку ««Simulate», указать вектор входных значений, и нажать Network»:

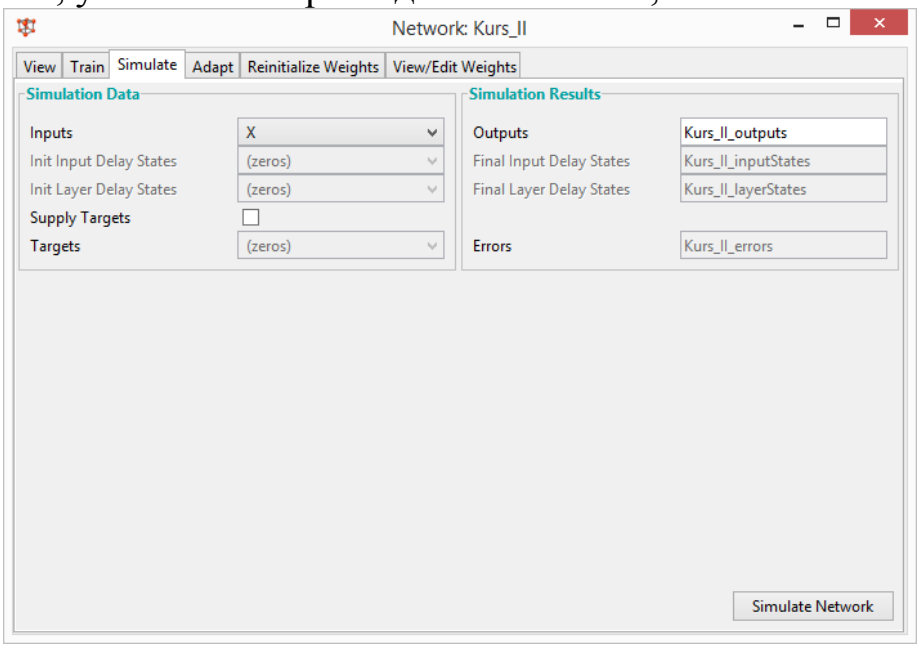


Рисунок 33. Проверка работы сети

Результаты работы сети появятся в главном окне NNTool:

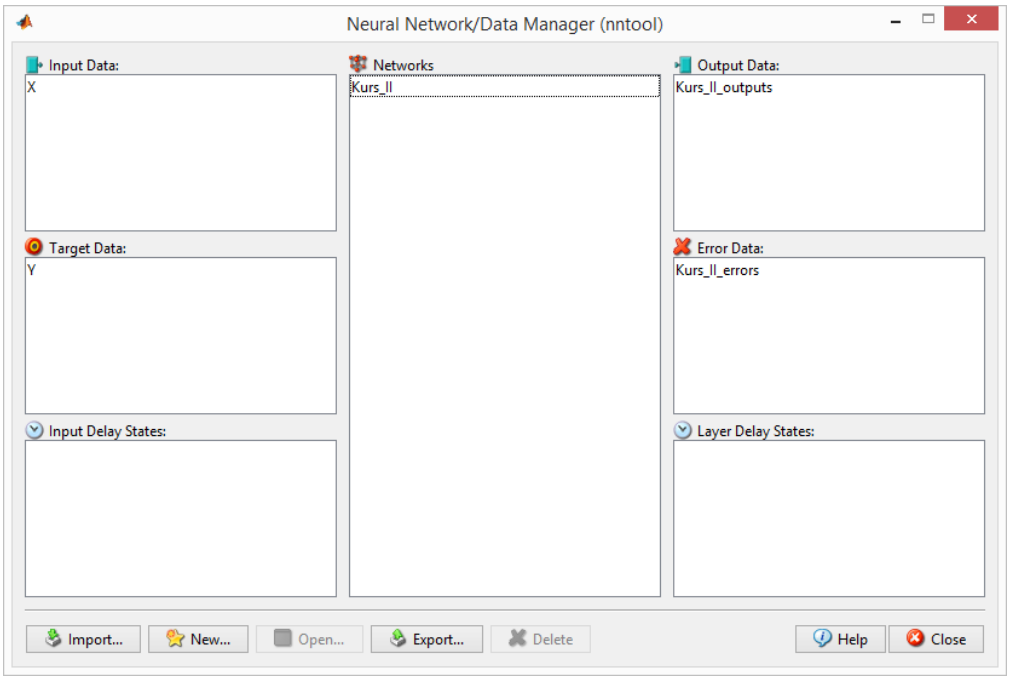


Рисунок 34. Результаты обучения сети

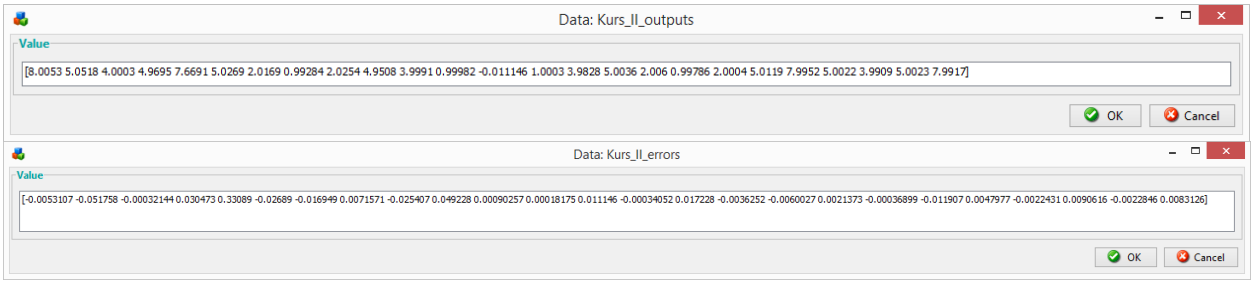


Рисунок 35. Результаты обучения сети: выходы и ошибки

Для сохранения результатов необходимо нажать «Export» в главном окне NNTool:

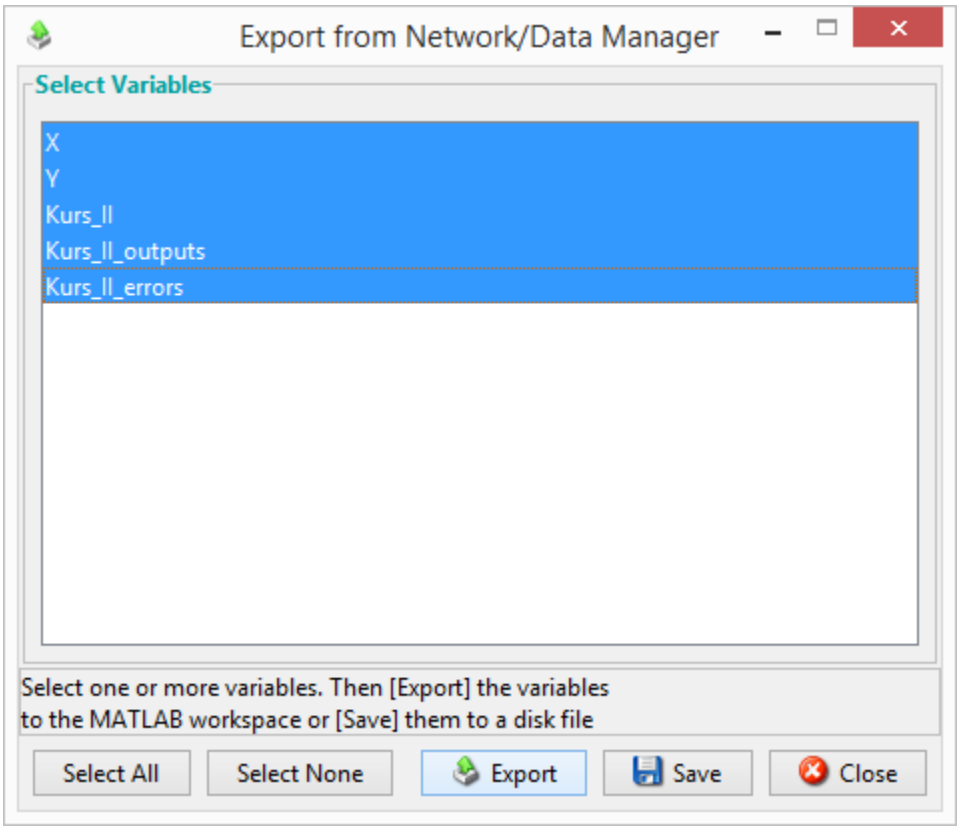


Рисунок 36. Сохранение результатов

2.5.3 Оценка погрешности аппроксимации

Для оценки погрешности аппроксимации необходимо найти корень из суммы квадратов разностей между желаемыми и получаемыми значениями.

Далее приведен фрагмент кода для вычисления погрешности аппроксимации:

```
%Значения векторов желаемых и полученных выходов сети
Target=[8 5 4 5 8 5 2 1 2 5 4 1 0 1 4 5 2 1 2 5 8 5 4 5 8];
Output=[8.0053 5.0518 4.0003 4.9695 7.6691 5.0269 2.0169
0.99284 2.0254 4.9508 3.9991 0.99982 -0.011146 1.0003 3.9828
5.0036 2.006 0.99786 2.0004 5.0119 7.9952 5.0022 3.9909 5.0023
7.9917];

%Вычисление погрешности
s1=0;
s2=0;
  for (i=1:25)
        s1=s1+((Target(i)-Output(i))^2);
        s2=s2+(Target(i)^2);
    end
 s1=s1^(1/2);
 d=s1/(s2^(1/2));
ans=d
```

В рассмотренном примере относительная погрешность составила $\delta = 0.0148 \approx 1.5\%$.

3 Объем и структура оформления курсовой работы

Каждая курсовая работа должна состоять из программ разработанных в среде Matlab и расчетно-пояснительной записки и графической части. Расчетно-пояснительная записка должна иметь 25 – 35 листов формата А4 машинного текста.

Нумерация листов пояснительной записки должна быть сквозной: первым листом является титульный лист. Номер листа проставляют арабскими цифрами. На листе 1 (титульный лист) номер листа не ставят.

Если в пояснительной записке содержатся рисунки и таблицы, которые располагаются на отдельных листах, их необходимо включать в общую нумерацию. Если рисунок или таблица расположены на листе формата больше А4, что не рекомендуется, их следует учитывать как один лист. Номер листа в этих случаях можно не проставлять. Список литературы и приложения необходимо включать в сквозную нумерацию.

Расчетно-пояснительная записка должна включать в себя:

- титульный лист;
- содержание(оглавление);
- введение
- вариант задания на курсовую работу;
- основную часть, включая теоретическую и практическую часть;
- список литературы;
- приложения.

Основная часть не должна содержать программный код.

Задание на курсовую работу

Необходимо реализовать различные подходы к аппроксимации функций с применением аппарата нечетких множеств (нечеткого логического вывода) и искусственных нейронных сетей в качестве универсального аппроксиматора:

- выполнить аппроксимацию функции:

$$y=x_1^2+x_2^2+kx_1x_2$$

на области $D=\{-2\le x_1\le 2;\ -2\le x_2\le 2\}$, где k номер варианта, в соответствии с порядковым номером в журнале старосты.

- оценить погрешность вычислений по норме Гаусса.

Вопросы к защите курсовой работы

1. Определение нечеткого множества.
2. Определение лингвистической переменной.
3. Логико-лингвистическое описание систем (нечеткие модели).
4. Этапы нечеткого логического вывода (схема).
5. Алгоритмы нечеткого логического вывода.
6. Применение нечеткой логики к аппроксимации функций.
7. Создание нечетких моделей с применением Matlab Fuzzy Logic ToolBox.
8. Понятие нейронной сети.
9. Модель искусственного нейрона.
10. Архитектуры (структуры) нейронных сетей.
11. Обучение нейронных сетей, правило Хебба.
12. Алгоритм обратного распространения ошибки.
13. Применение нейронных сетей к аппроксимации функций.
14. Создание и обучение нейронных сетей с помощью Matlab Neural Network ToolBox .

Список литературы

1. Заде Л. Понятие лингвистической переменной и его применение для принятия приближенных решений. М.: Мир, 1976. -165с.
2. Круглов В.В., Дли М.И., Голунов Р.Ю., Нечёткая логика и искусственные нейронные сети. М.: ФИЗМАТЛИТ – 2001.- 221 с.
3. Ротштейн А. П. Интеллектуальные технологии идентификации: нечёткая логика, генетические алгоритмы, нейронные сети / А. П. Ротштейн. – Винница : УНИВЕРСУМ-Винница, 1999. – 320 с.
4. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы/ Рутковская Д., Пилиньский М., Рутковский Л.– Электрон. текстовые данные.– М.: Горячая линия - Телеком, 2013.– 384 с.
5. Интеллектуальные системы: учебное пособие/ А.М. Семенов [и др.].– Электрон. текстовые данные.– Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2013.– 236 с.
6. Интеллектуальные системы: методические указания к лабораторным работам для студентов бакалавриата, обучающихся по направлению подготовки 01.03.04 «Прикладная математика»/ – Электрон. текстовые данные.– М.: Московский государственный строительный университет, Ай Пи Эр Медиа, ЭБС АСВ, 2015.– 57 с.
7. Сысоев Д.В. Введение в теорию искусственного интеллекта: учебное пособие/ Сысоев Д.В., Курипта О.В., Проскурин Д.К.– Электрон. текстовые данные.– Воронеж: Воронежский государственный архитектурно-строительный университет, ЭБС АСВ, 2014.– 171 с.
8. Борисов В.В. Нечеткие модели и сети: монография/ Борисов В.В., Круглов В.В., Федулов А.С.– Электрон. текстовые данные.– М.: Горячая линия - Телеком, 2012.– 284 с.
9. Галушкин А.И. Нейронные сети. Основы теории [Электронный ресурс]: монография/ Галушкин А.И.– Электрон. текстовые данные.– М.: Горячая линия - Телеком, 2012.– 496 с.

10. Барский А.Б. Введение в нейронные сети / Барский А.Б.– Электрон. текстовые данные.– М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.– 358 с.
11. Яхъяева Г.Э. Нечеткие множества и нейронные сети [Электронный ресурс]: учебное пособие/ Яхъяева Г.Э.– Электрон. текстовые данные.– М.: БИНОМ. Лаборатория знаний, Интернет-Университет Информационных Технологий (ИНТУИТ), 2008.– 316 с.

Методические указания к курсовой работе по дисциплине
«Нечеткая логика и нейронные сети»

Составитель
Абрахин Сергей Иванович

Ответственный за выпуск – зав. кафедрой ФиПМ профессор
С.М. Аракелян

Редактор
Корректор
Компьютерная верстка

№ . Подписано в печать
Формат . Гарнитура Таймс.
Редакционно-издательский комплекс
Владимирского государственного университета
600000, Владимир, ул. Горького, 87