Федеральное государственное автономное образовательное учреждение
высшего образования

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(НИУ «БЕЛГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Р.Г. АСАДУЛЛАЕВ

Нечеткая логика и нейронные сети

Учебное пособие
по направлению подготовки 38.03.05 «Бизнес-информатика»

Белгород 2017

УДК 004.032.26+510.6
ББК 22.12+ 32.813.5

Составитель: Кандидат технических наук, доцент кафедры прикладной информатики и информационных технологий Института инженерных технологий и естественных наук
Р.Г. Асадуллаев

Рецензенты: Кандидат технических наук, доцент кафедры технической кибернетики БГТУ им. В.Г. Шухова
А.Г. Бажанов;
Кандидат технических наук, доцент, заведующий кафедрой прикладной информатики и информационных технологий
В.В. Ломакин

Нечеткая логика и нейронные сети: учебное пособие / сост. Р.Г. Асадуллаев. – Белгород, 2017. – 309 с.

Учебное пособие предназначено для проведения лекционных и практических занятий по дисциплине «нечеткая логика и нейронные сети». Пособие содержит теоретические положения теории нечетких множеств и нейронных сетей, а также лабораторный практикум в пакете MatLab.

Оглавление

Тема 1. Нечеткие множества как способ формализации нечеткости

Цель: изучить понятие нечеткого множества с точки зрения способа формализации нечеткости.

Задачи:

1. Определить необходимость использования нечеткой логики в практике управления.
2. Ознакомиться с понятием нечеткого множества.
3. Рассмотреть основные характеристики нечетких множеств.
4. Рассмотреть понятие гетерогенного нечеткого множества.

1.1. Необходимость использования нечеткой логики в практике управления

Понимание какого-либо процесса или явления отождествляется с возможностью его количественного анализа. В настоящее время, однако, правомерность такого анализа, основанного на использовании дифференциальных или конечно-разностных уравнений для описания систем, в которых участвует человек, подвергается сомнению.

Системы, неотъемлемым (или даже основным) фактором которых является именно человек и его суждения, относятся к классу так называемых **слабоструктурированных (сложных) систем**, для которых обычные количественные методы анализа и описания не применимы по своей сути. В основе этого тезиса лежит **принцип несовместимости**, сформулированный Заде, который утверждает, что чем сложнее система, тем менее мы способны дать точные и в то же время имеющие практическое значение суждения. Для систем, сложность которых превосходит некоторый пороговый уровень, точность и практический смысл, т.е. содержательность, становятся почти взаимоисключающими характеристиками.

Можно перейти из дискретного пространства сценариев в непрерывное, заменив дискретное весовое распределение факторов непрерывной плотностью распределения. Имея такие распределения на входе в модель, можно точно или приближенно восстановить распределение выходных параметров модели (например, финансовых показателей). И такой путь, снимая проблему ограниченности сценариев, не снимает другую проблему – обоснованности модельных вероятностных распределений.

Если рассматривать классическое понимание вероятности, то, прежде всего такая вероятность вводится как частота однородных событий, происходящих в неизменных внешних условиях. В реальном менеджменте нет ни однородности, ни неизменности условий. Даже два предприятия, принадлежащие к одной отрасли и работающие на одном и том же рынке,

развиваются по-разному в силу внутренних особенностей. Так, успешный менеджмент одной такой компании приводит ее к успеху, а неуспешный менеджмент другой – к банкротству. На уровне «черных ящиков» обе компании могут выглядеть одинаково, однородно, но при раскрытии информации о компаниях, при детализации вся однородность пропадает.

Не сохраняется однородность и с течением времени. Так, американский фондовый рынок образца 2002 года – это вовсе не то же самое, что рынок образца 1999 года. Кардинально различны все макроэкономические параметры. Ясно, что рынку до кризиса может быть сопоставлена одна сценарно-вероятностная модель, а для послекризисного рынка она будет совсем другой: изменятся как сами сценарии, так и их веса.

Много усилий в науке было потрачено на то, чтобы отойти от классического понимания вероятностей. По мере перехода от классической вероятности к аксиологической (субъективной) возрастала роль эксперта, назначающего вероятностные, веса, увеличивалось влияние субъективных предпочтений эксперта на оценку. Соответственно, чем более субъективной становилась вероятность, тем менее научной она оказывалась.

Появление субъективных вероятностей в экономическом анализе далеко не случайно. Этим было ознаменовано первое стратегическое отступление науки перед лицом неопределенности, которая имеет неустранимый характер. Такая неопределенность является неустранимой, так как не обладает структурой, которую можно было бы один раз и навсегда модельно описать вероятностями и вероятностными процессами. То, что с большим успехом используется в технике, в теории массового обслуживания, в статистике как науке о поведении большого числа однородных (принадлежащих одному модельному классу) объектов, то, например, совершенно не проходит в моделях финансового менеджмента. Исследователь имеет дело с ограниченным набором событий, разнородных по своему происхождению, и он затрудняется в том, какие выводы сделать на основе полученной информации.

Таким образом, сам эксперт, его научная активность, его предпочтения начинают сами выступать как объект научного исследования. Уверенность (неуверенность) эксперта в оценке приобретают количественное выражение, и здесь вероятностям делать уже совершенно нечего. Аналогия может быть такой, что если раньше врач пытался лечить больного, то теперь в лечении нуждается он сам. Объект научного исследования доопределился: если ранее в него входил только объект (корпорация, отрасль, страна), то в современном менеджменте объект научного исследования дополняется лицом, принимающим решения (ЛПР). Таким лицом выступает как менеджер, так и аналитик (конкретного вида менеджмента), готовящий решения для менеджера. Активность обоих этих лиц подлежит детальному исследованию.

Самое главное в такой постановке научной задачи – научиться моделировать субъектную активность. В частности, важно представлять, по

каким критериям ЛПР производит распознавание текущей экономической ситуации, состояния объекта исследования, поля для принятия решений. Информации не хватает, она не очень высокого качества. Соответственно, ЛПР сознательно или подсознательно отходит от точечных числовых оценок, заменяя их качественными характеристиками ситуации, выраженными на естественном языке (например, «высокий/низкий уровень фактора», «большой/малый/незначительный размер денежного потока», «приемлемый/запредельный риск» и т.д.). Пока терминам естественного языка не сопоставлена количественная оценка, они могут интерпретироваться произвольно. Но если такая оценка состоялась как модель, образованная на пересечении мнений и предпочтений целого ряда экспертов, наблюдающих примерно одну и ту же экономическую реальность, тогда она обладает значимостью для моделирования экономического объекта, наряду с данными о самом этом объекте.

В результате точный количественный анализ слабо структурированных систем не имеет практического значения при решении экономических, социальных, политических и других задач, в которых сложность описания рассматриваемой системы достаточно велика. Альтернативный подход состоит в использовании при анализе таких систем не количественных значений, а нечетких множеств, когда переход от принадлежности некоторого объекта к какому-либо классу происходит не скачкообразно, а непрерывно. В самом деле, человек в свой повседневной и профессиональной деятельности мыслит скорее на качественном, понятийном уровне, чем на количественном. При этом понятия являются, как правило, неточными, нечеткими (например, «высокая температура», «сильный шум» и т.д.).

Такая вездесущая нечеткость человеческого мышления позволяет судить о том, что логика рассуждений человека не является двузначной (или даже многозначной). Это - логика с нечеткими истинами, нечеткими отношениями и нечеткими правилами вывода. Предполагается, что именно такая логика играет ключевую роль в мыслительной деятельности человека, наделяет его мышление чрезвычайно важной способностью - оценивать информацию и обобщать ее.

Благодаря нечеткой логике обеспечивается решение задач, с которыми неспособна справиться четкая логика. Например, разрешаются парадоксы «Куча: одно пшеничное зерно не образует кучи. То же верно и для двух зерен, трех и так далее. Следовательно, куча не существует» и «Лысый человек: человек без волос или только с одним волосом – лысый. То же верно и для человека с двумя волосами и так далее. Следовательно, все люди – лысые».

Способность манипулировать нечеткими понятиями является важнейшей особенностью и достижением человеческого интеллекта, которые выгодно отличают его от машинного интеллекта. С этой точки зрения традиционные методы анализа систем не подходят для работы со слабо

структурированными системами, поскольку они не позволяют выявить и использовать то, что играет чрезвычайно важную роль в этих системах, - нечеткость мышления в поведении человека.

Методы нечеткой логики позволяют строить логико-лингвистические модели, отражающие общую смысловую постановку задачи, используя качественные представления соответствующие "человеческим" способам рассуждений и принятия решений.

1.2. Понятие нечеткого множества

Теория нечетких множеств представляет собой обобщение и переосмысление важнейших направлений классической математики. У ее истоков лежат идеи и достижения:

- многозначной логики, которая указала на возможности перехода от двух к произвольному числу значений истинности и поставила проблему оперирования понятиями с изменяющимся содержанием;

- теории вероятностей, которая, породив большое количество различных способов статистической обработки экспериментальных данных, открыла пути определения и интерпретации функции принадлежности;

- дискретной математики, которая предложила инструмент для построения моделей многомерных и многоуровневых систем, удобный при решении практических задач.

Подход к формализации понятия нечеткого множества состоит в обобщении понятия принадлежности. В обычной теории множеств существует несколько способов задания множества. Одним из них является задание с помощью характеристической функции, определяемой следующим образом. Пусть U — так называемое универсальное множество, из элементов которого образованы все остальные множества, рассматриваемые в данном классе задач, например множество всех целых чисел, множество всех гладких функций и т.д. Характеристическая функция множества $A \subseteq U$ — это функция μ_A, значения которой указывают, является ли $x \in U$ элементом множества A:

$$\mu_A(x) = \begin{cases} 1, & if \quad x \in A \\ 0, & if \quad x \notin A \end{cases}$$

Особенностью этой функции является бинарный характер ее значений. С точки зрения характеристической функции, нечеткие множества есть естественное обобщение обычных множеств, когда мы отказываемся от бинарного характера этой функции и предполагаем, что она может принимать любые значения на отрезке [0, 1]. В теории нечетких множеств характеристическая функция называется **функцией принадлежности**, а ее значение $\mu_A(x)$ — **степенью принадлежности** элемента x нечеткому множеству A.

Функция принадлежности (membership function) - функция, которая позволяет вычислить степень принадлежности произвольного элемента универсального множества к нечеткому множеству.

Более строго, **нечетким множеством** A называется совокупность пар

$$A = \{\langle x, \mu_A(x)\rangle \mid x \in U\}$$

где $\mu_A(x)$ — функция принадлежности, т.е. $\mu_A(x) : U \to [0,1]$.

Пусть, например,

$$U = \{a, b, c, d, e\},$$

$$A = \{\langle a, 0\rangle,\ \langle b, 0{,}1\rangle,\ \langle c, 0{,}5\rangle,\ \langle d, 0{,}9\rangle,\ \langle e, 1\rangle\}.$$

Будем говорить, что элемент a не принадлежит множеству A, элемент b принадлежит ему в малой степени, элемент c более или менее принадлежит, элемент d принадлежит в значительной степени, e является элементом множества A.

Примеры записи нечеткого множества. Пусть $E = \{x_1, x_2, x_3, x_4, x_5\}$, $M = [0,1]$. A – нечеткое множество, для которого $\mu_A(x_1) = 0{,}3; \mu_A(x_2) = 0; \mu_A(x_3) = 1; \mu_A(x_4) = 0{,}5; \mu_A(x_5) = 0{,}9$.

Тогда A можно представить в виде $A = \{0{,}3/x_1;\ \ 0/x_2;\ \ 1/x_3;\ \ 0{,}5/x_4;\ \ 0{,}9/x_5\}$,

или $A = \{0{,}3/x_1 + 0/x_2 + 1/x_3 + 0{,}5/x_4 + 0{,}9/x_5\}$,

или $A = \frac{x_1}{0{,}3} | \frac{x_2}{0} | \frac{x_3}{1} | \frac{x_4}{0{,}5} | \frac{x_5}{0{,}9}$.

Здесь знак «+» не является обозначением операции сложения, а имеет смысл объединения.

Пример. Пусть универсум U есть множество действительных чисел. Нечеткое множество A, обозначающее множество чисел, близких к 10 (рисунок 1.1), можно задать следующей функцией принадлежности:

$$\mu_A(x) = (1 + |x - 10|^m)^{-1},$$

где $m \in N$.

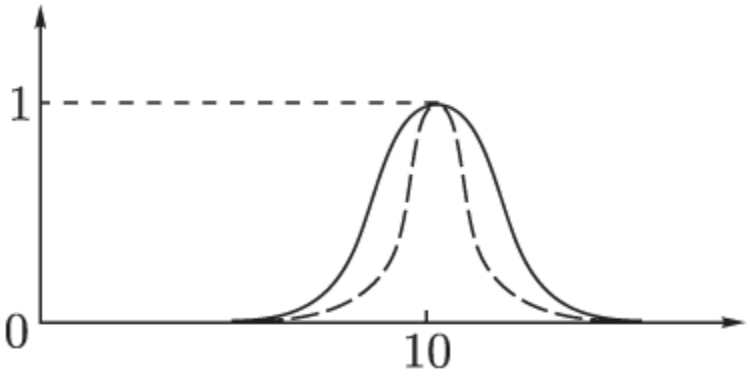


Рисунок 1.1 – Функция принадлежности множества чисел близких к 10

Показатель степени *m* выбирается в зависимости от степени близости к *10*. Например, для описания множества чисел, очень близких к *10*, можно положить *m=4*; для множества чисел, не очень далеких от *10*, *m=1*.

Коротко остановимся на понятии лингвистической переменной. **Лингвистическую переменную** можно определить как переменную, значениями которой являются не числа, а слова или предложения естественного (или формального) языка. Например, лингвистическая переменная "возраст" может принимать следующие значения: "очень молодой", "молодой", "среднего возраста", "старый", "очень старый" и др. Ясно, что переменная "возраст" будет обычной переменной, если ее значения — точные числа; лингвистической она становится, будучи использованной в нечетких рассуждениях человека.

Каждому значению лингвистической переменной соответствует определенное нечеткое множество со своей функцией принадлежности. Так, лингвистическому значению "молодой" может соответствовать функция принадлежности, изображенная на рисунке 1.2.

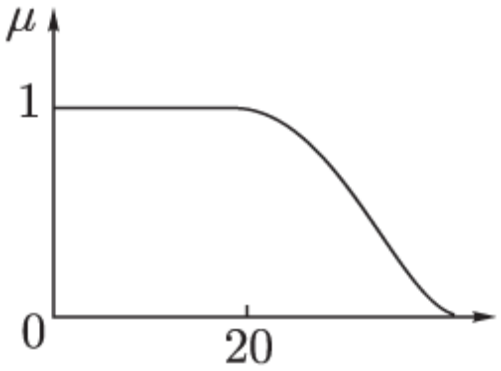


Рисунок 1.2 – Функция принадлежности лингвистического значения «молодой»

1.3. Основные характеристики нечетких множеств

Носитель нечеткого множества A - четкое множество $\tilde{A}$ таких точек в U, для которых величина $\mu_A(x)$ положительна, т.е. $\tilde{A} = \{x \mid \mu_A(x) > 0\}$.

Высота нечеткого множества A - верхняя граница его функции принадлежности: $\sup_U \mu_A(x)$.

Нормальное нечеткое множество A, если $\sup_U \mu_A(x) = 1$. В противном случае оно называется **субнормальным нечетким множеством**.

Нечеткое множество унимодально, если $\mu_A(x) = 1$ только на одном элементе $x \in A$.

Пустое нечеткое множество, если $\forall x \in U \quad (\mu_A(x) = 0)$. Очевидно, что в данном универсуме U существует единственное пустое нечеткое множество. Непустое субнормальное нечеткое множество можно привести к нормальному (нормализовать) по формуле:

$$\mu'_A(x) = \frac{\mu_A(x)}{\sup\limits_U \mu_A(x)}$$

Пример. На рисунке 1.3 показана нормализация нечеткого множества $\tilde{A}'$ с функцией принадлежности $\mu_{A'}(u) = \frac{0.6}{1+(10-u)^2}$.

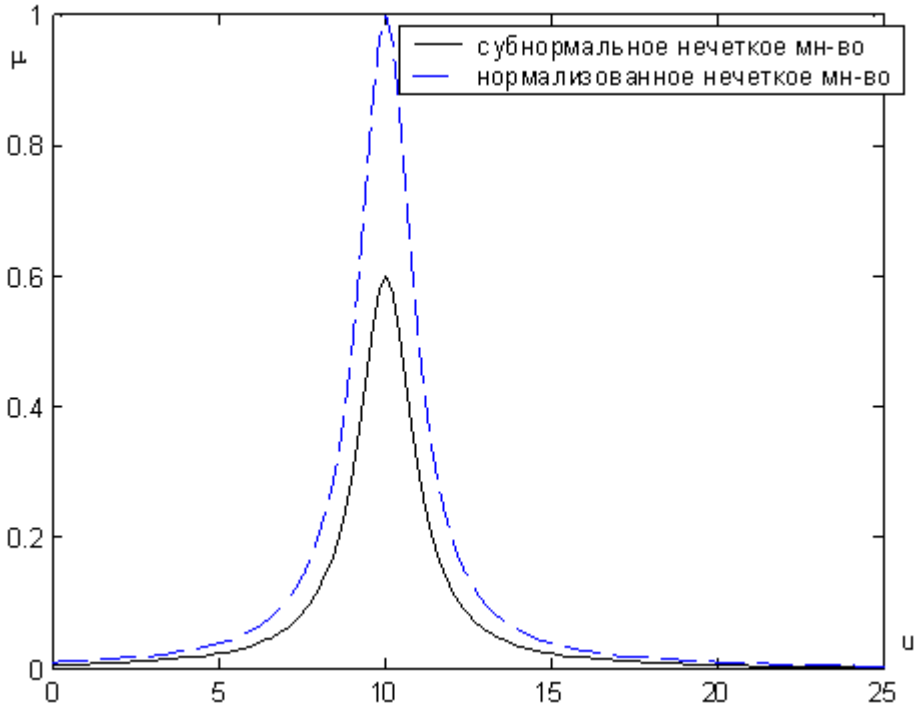


Рисунок 1.3 - Нормализация нечеткого множества

Ядро нечеткого множества - четкое подмножество универсального множества U, элементы которого имеют степени принадлежности равные единице. Ядро субнормального нечеткого множества пустое.

Множество уровня α (α-срезом) нечеткого множества A - четкое подмножество универсального множества U, определяемое по формуле:

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha\}, \quad if \quad \alpha \in [0,1]$$

Множество строгого уровня определяется в виде $A_\alpha = \{x \mid \mu_A(x) > \alpha\}$. В частности, носителем нечеткого множества является множество элементов, для которых $\mu_A(x) > 0$. Понятие множества уровня является расширением понятия интервала. Оно представляет собой объединение не более чем счетного числа интервалов. Соответственно, алгебра интервалов есть частный случай алгебры множеств уровня.

Рисунок 1.4 иллюстрирует определения носителя, ядра, α-сечения и α-уровня нечеткого множества.

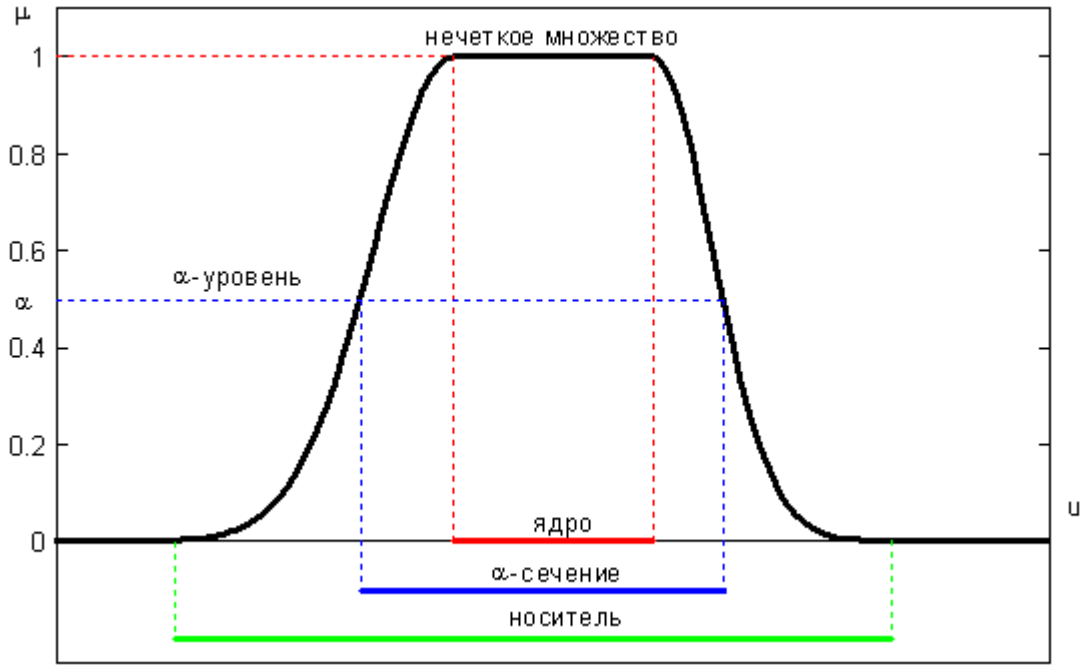


Рисунок 1.4 - Ядро, носитель и α-сечение нечеткого множества

Точка перехода нечеткого множества A — это такой элемент $x \in U$, для которого $\mu_A(x) = 0{,}5$.

Четкое множество $\dot{A}$, ближайшее к нечеткому множеству A, определяется следующим образом:

$$\mu_A(x) = \begin{cases} 0, & if \quad \mu_A(x) < 0{,}5 \\ 1, & if \quad \mu_A(x) > 0{,}5 \\ 0 \; or \; 1, & else \end{cases}$$

Выпуклое нечеткое множество $\tilde{A}$ - если: $\mu_A(\lambda u_1 + (1-\lambda)u_2) \geq \min(\mu_A(u_1), \mu_A(u_2))$, $u_1, u_2 \in U$, $\lambda \in [0,1]$.

На рисунке 1.5 приведены примеры выпуклого и невыпуклого нечетких множеств.

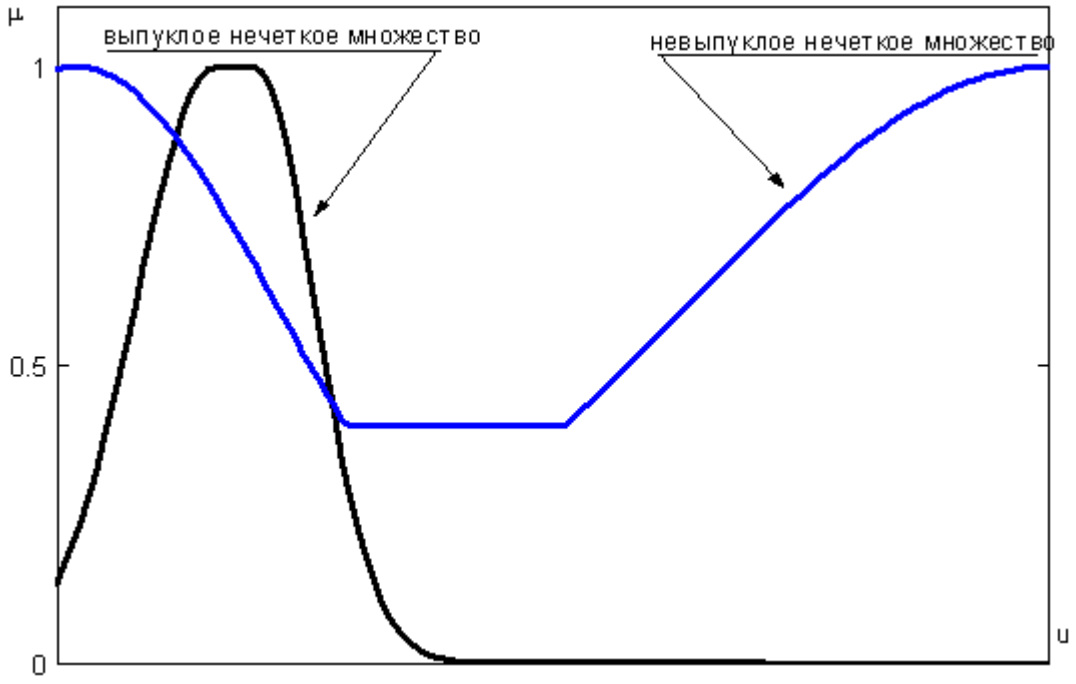


Рисунок 1.5 - К определению выпуклого нечеткого множества

Принцип обобщения как одна из основных идей теории нечетких множеств носит эвристический характер и позволяет расширить область определения исходного отображения f на класс нечетких множеств. Пусть $f: U \to V$ — заданное отображение, и A — нечеткое множество вида

$$A = \{\mu_1 / u_1 + \mu_2 / u_2 + \ldots + \mu_n / u_n\}$$

Тогда принцип обобщения утверждает, что

$$f(A) = f(\mu_1 / u_1 + \mu_2 / u_2 + \ldots + \mu_n / u_n) \equiv \mu_1 f(u_1) + \mu_2 f(u_2) + \ldots + \mu_n f(u_n)$$

Итак, образ множества A при отображении f можно получить, зная образы элементов $u_1, \ldots, u_n$ при этом отображении.

Пример. Пусть $U = 1 + 2 + \ldots + 10$, и пусть f — операция возведения в квадрат. Пусть «малый» — нечеткое подмножество множества U вида

$$\textit{малый} = 1/1 + 1/2 + 0.8/3 + 0.6/4 + 0.4/5.$$

Тогда, учитывая нечеткое подмножество «малый», имеем

$$\textit{малый}^2 = 1/1 + 1/4 + 0.8/9 + 0.6/16 + 0.4/25.$$

Примеры нечетких множеств:

1. Пусть $U = \{0,1,2,3.,..,10\}$, $M = [0,1]$. Нечеткое множество A=«Несколько» можно определить следующим образом: $A = \{0,5/3 + 0,8/4 + 1/5 + 1/6 + 0,8/7 + 0,5/8\}$. Характеристиками данного множества являются: высота $\sup_U \mu_A(x) = 1$, носитель $\tilde{A} = \{3,4,5,6,7,8\}$, точка перехода $\{3,8\}$.

2. Пусть $U=\{0,1,2,3.,..,n,....\}$, тогда нечеткое множество A=«Малый» можно определить следующим образом:

$$A=\{\mu_A(n)=\frac{1}{1+(\frac{n}{10})^2}/n$$

3. Пусть $U=\{1,2,3.,..,100\}$ и соответствует понятию «Возраст», тогда нечеткое множество A=«Молодой» может быть определено с помощью:

$$\mu_A(x)=\begin{cases}1, & if \quad x\in[1,25],\\ \dfrac{1}{1+((x-25)/5)^2}, & if \quad x>25.\end{cases}$$

1.4. Гетерогенные нечеткие множества

В том случае, когда набор нечетких множеств A_i, $i=1,2,\ldots n$ в U соответствует m различным свойствам рассматриваемого объекта, каждый элемент $x\in U$ характеризуется вектором значений принадлежности $(\mu_1(x),\mu_2(x),....\mu_m(x))$, выражающим степень соответствия этим свойствам. Таким образом, строится функция $\mu:U\to[0,1]^m$, где $[0,1]^m$ — полная решетка.

Дальнейшим обобщением понятия нечеткого множества является понятие гетерогенного нечеткого множества. По признаку однородности/неоднородности области значений функции принадлежности все описанные выше виды нечетких множеств являются **гомогенными** в том смысле, что одна и та же структура области значений функции принадлежности берется при оценке всех элементов универсального множества U. Если же допустить, что на различных элементах универсального множества U функция принадлежности может принимать свои значения из различных наиболее подходящих математических структур, то мы приходим к понятию **гетерогенного нечеткого множества.**

Гетерогенные нечеткие множества и связанные с ними составные лингвистические переменные высокого порядка позволяют моделировать ситуации многокритериального принятия решения, когда имеются признаки, как с количественными, так и с порядковыми шкалами.

Вопросы для повторения и закрепления материала

1. Где применяется аппарат нечетких множеств?
2. В чем заключается суть нечеткого множества?
3. С какой целью используется функция принадлежности?
4. Какие значения может принимать функция принадлежности?
5. Что такое гетерогенные нечеткие множества?

6. В каком случае употребляют понятие гетерогенное нечеткое множество?

7. Назовите основные характеристики нечетких множеств?

8. В каком диапазоне изменяется значение функции принадлежности нечеткого множества?

Задания для самостоятельной работы

1. Ознакомиться с трудами основателя нечеткой логики Лотфи Заде.

2. Рассмотреть понятие и свойства сложных систем.

3. Проанализировать виды профессиональной деятельности своего направления подготовки и попробовать определить системы и процессы, в которых можно применить аппарат нечетких множеств.

Тема 2. Операции над нечеткими множествами

Цель: изучить логические операции над нечеткими множествами, а также основные виды функций принадлежности.

Задачи:

1. Изучить логические операции над нечеткими множествами.

2. Рассмотреть свойства логических операций над нечеткими множествами.

3. Рассмотреть нечеткие операторы.

4. Определить основные виды функций принадлежности нечетких множеств.

2.1. Классификация операций над нечеткими множествами

Над нечеткими множествами можно производить различные операции, при этом необходимо определить их так, чтобы в частном случае, когда множество является четким, операции переходили в обычные операции теории множеств, то есть операции над нечеткими множествами должны обобщать соответствующие операции над обычными множествами. При этом обобщение может быть реализовано различными способами, из-за чего какой-либо операции над обычными множествами может соответствовать несколько операций в теории нечетких множеств. Для определения пересечения и объединения нечетких множеств наибольшей популярностью пользуются следующие три группы операций:

1. Максиминные:

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\},$$

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}.$$

2. Алгебраические:

$$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x),$$
$$\mu_{A\cap B}(x) = \mu_A(x)\mu_B(x).$$

3. Ограниченные:

$$\mu_{A\cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\},$$
$$\mu_{A\cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}.$$

Дополнение нечеткого множества во всех трех случаях определяется одинаково: $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$.

В теории нечетких множеств оператор дополнения не является единственным. Помимо общеизвестного $\forall x \quad \overline{\mu}(x) = 1 - \mu(x)$, существует целый набор операторов дополнения нечеткого множества.

Пусть задано некоторое отображение $\lambda : [0,1] \to [0,1]$. Это отображение будет называться оператором отрицания в теории нечетких множеств, если выполняются следующие условия:

1. $\lambda(0) = 1,$
   $\lambda(1) = 0$
2. $\mu_A \le \mu_B \Rightarrow \lambda(\mu_A) \ge \lambda(\mu_B)$

Если кроме этого выполняются условия:

3. λ — строго убывающая функция
4. λ — непрерывная функция

то она называется **строгим отрицанием**.

Функция λ называется **сильным отрицанием** или **инволюцией**, если наряду с условиями (1) и (2) для нее справедливо:

5. $\lambda(\lambda(\mu)) = \mu$

Примеры функции отрицания:

- Классическое отрицание: $\lambda(\mu) = \overline{\mu}(x) = 1 - \mu(x)$.
- Квадратичное отрицание: $\lambda(\mu) = \sqrt{1 - \mu^2}$.
- Отрицание Сугено: $\lambda(\mu) = \dfrac{1-\mu}{1+k\mu}, \quad where \quad -1 < k < \infty$.
- Дополнение порогового типа: $\lambda(\mu) = \begin{cases} 1, & if \quad \mu \le \alpha, \\ 0, & if \quad \mu > \alpha. \end{cases}$

Будем называть любое значение λ, для которого $\lambda(\mu) = \mu$, **равновесной точкой**. Для любого непрерывного отрицания существует единственная равновесная точка.

Пример. Пусть А — нечеткое множество "от 5 до 8" (рисунок 2.1 μ_A) и В — нечеткое множество "около 4" (рисунок 2.1 μ_B), заданные своими функциями принадлежности:

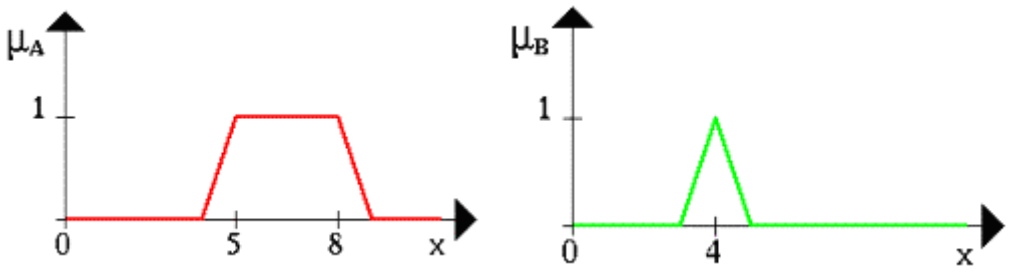


Рисунок 2.1 – Функции принадлежности μ_A «от 5 до 8» и μ_B «около 4»

Тогда, используя максиминные операции, мы получим множества, изображенные на рисунке 2.2.

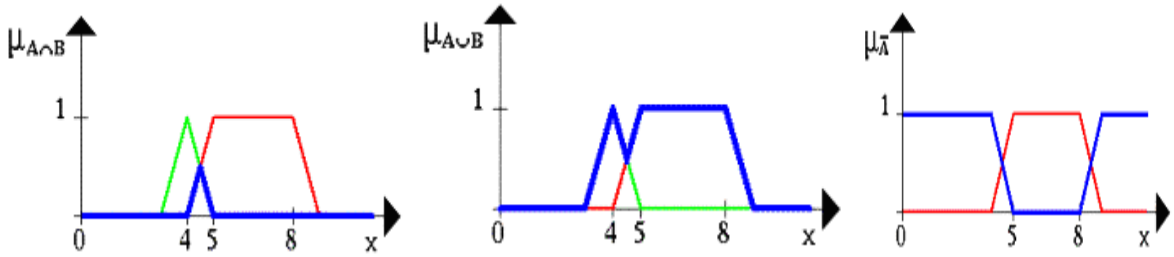


Рисунок 2.2 – Множества после применения максиминных операций

Заметим, что при максиминном и алгебраическом определении операций не будут выполняться законы противоречия и исключения третьего $A \cap \overline{A} \neq 0$, $A \cup \overline{A} \neq U$, а в случае ограниченных операций не будут выполняться свойства идемпотентности $A \cap A \neq A$, $A \cup A \neq A$ и дистрибутивности:

$$A \cup (B \cap C) \neq (A \cap B) \cup (A \cap C),$$

$$A \cap (B \cup C) \neq (A \cup B) \cap (A \cup C)$$

2.2. Логические операции над нечеткими множествами

Включение. Пусть A и B - нечеткие множества на универсальном множестве X. Говорят, что A содержится в B, или B включает A, т.е. $A \subset B$, если $\forall x \in X \; \mu_A(x) \leq \mu_B(x)$. Иногда используют термин «доминирование», т.е. B доминирует A при $A \subset B$ (рисунок 2.3).

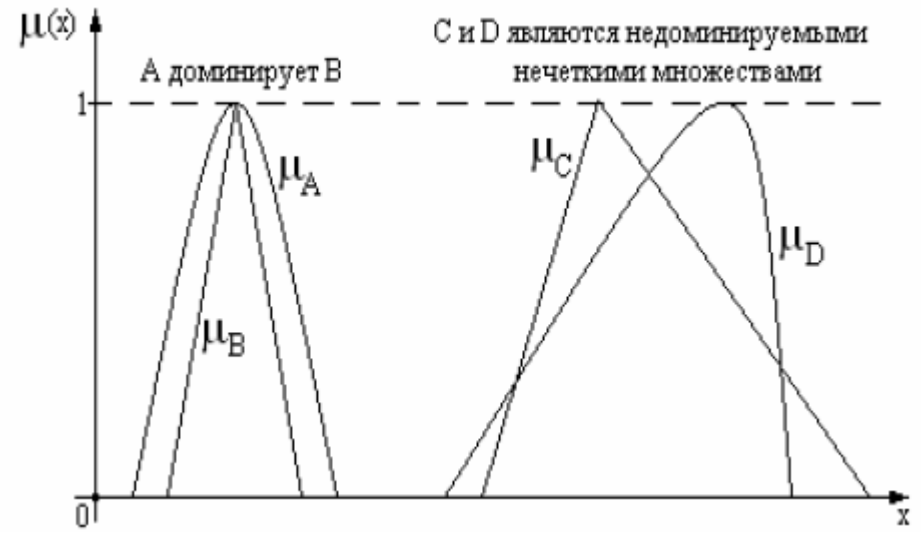


Рисунок 2.3 - Операция включение (доминирование) нечетких множеств

Равенство. Пусть A и B - нечеткие множества на универсальном множестве X . Говорят, что A и B равны, т.е. $A=B$, если $\forall x \in X\ \mu_A(x) = \mu_B(x)$. В противном случае $A \neq B$ (рисунок 2.4).

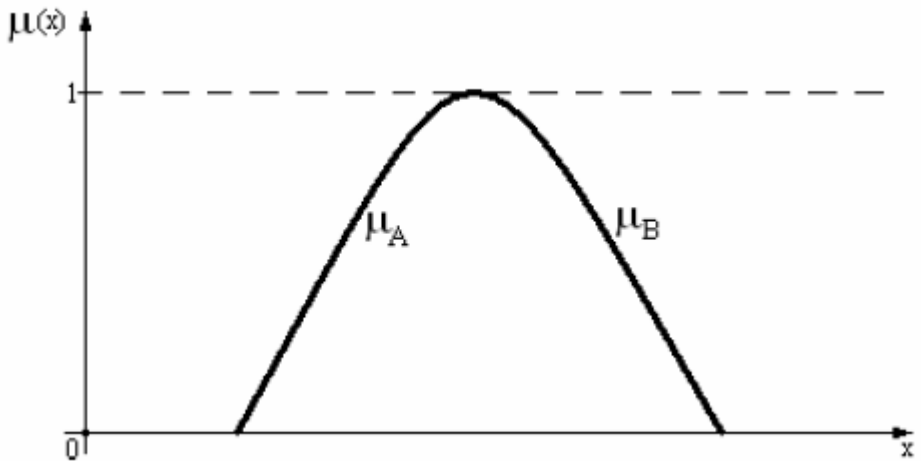


Рисунок 2.4 - Операция равенства нечетких множеств

Дополнение. Пусть A и B – нечеткие множества с множеством принадлежностей характеристических функций $M = [0:1]$, заданные на универсальном множестве X . Говорят, что A и B дополняют друг друга, т.е. $A = \bar{B}$ или $B = \bar{A}$, если $\forall x \in X\ \mu_A(x) = 1 - \mu_B(x)$ (рисунок 2.5). Очевидно следствие $\bar{\bar{A}}$=A так называемое свойство **инволюции.**

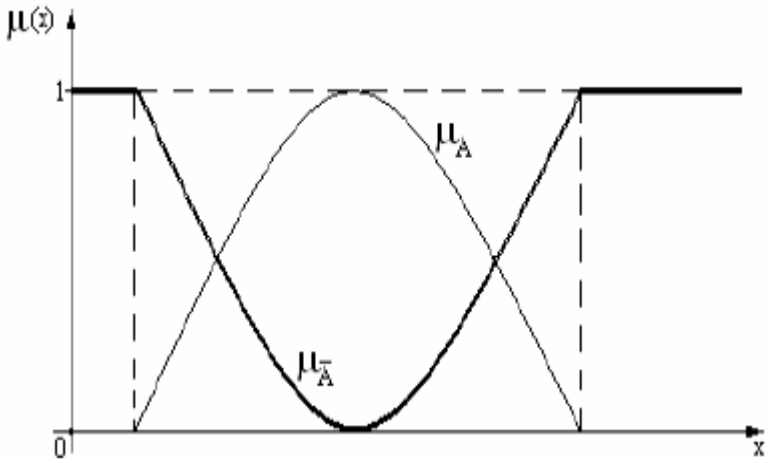


Рисунок 2.5 - Операция дополнение нечетких множеств

Пересечение нечетких множеств (рисунок 2.6) A и B , заданных на универсальном множестве X , - это наибольшее нечеткое множество A ∩ B, содержащееся одновременно и в A , и в B с функцией принадлежности:

$$\forall x \in X \quad \mu_{A\cap B}(x) = \min\{\mu_A(x); \mu_B(x)\}.$$

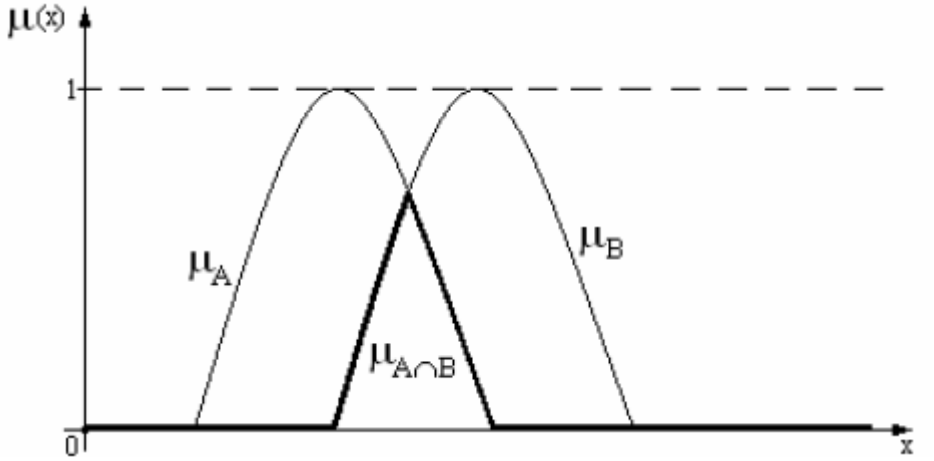


Рисунок 2.6 - Операция пересечение нечетких множеств

Объединение нечетких множеств (рисунок 2.7) A и B , заданных на универсальном множестве X , - это наименьшее нечеткое множество $A \cup B$, включающее как A , так и B с функцией принадлежности, заданной следующим образом:

$$\forall x \in X \quad \mu_{A\cup B}(x) = \max\{\mu_A(x); \mu_B(x)\}.$$

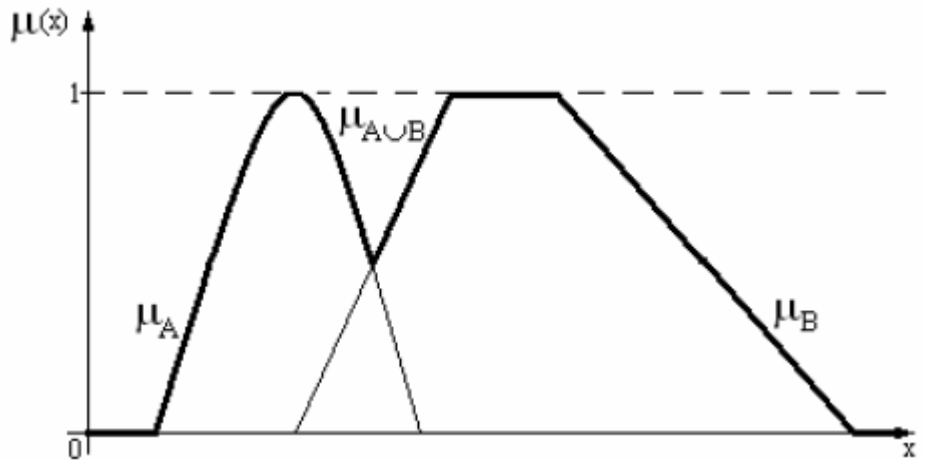


Рисунок 2.7 - Операция объединение нечетких множеств

Разность нечетких множеств *A* и *B* (рисунок 2.8), заданных на универсальном множестве *X*, - это нечеткое множество $A\backslash B=A\cap\overline{B}$ с функцией принадлежности, заданной как:

$$\forall x \in X\ \mu_{A\backslash B}(x)=\mu_{A\cap\overline{B}}(x)=\min\{\mu_A(x);1-\mu_B(x)\}$$

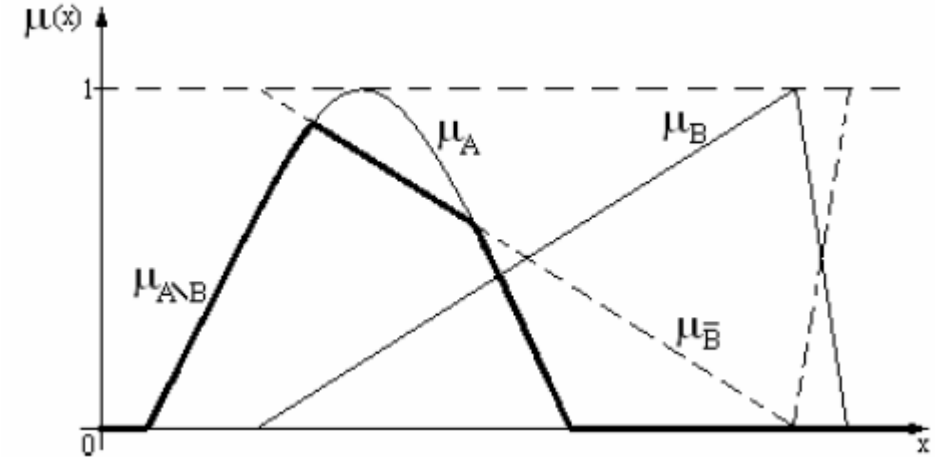


Рисунок 2.8 - Операция разность нечетких множеств

Симметрическая разность нечетких множеств *A* и *B* , заданных на универсальном множестве *X* , - это нечеткое множество *A*-*B* с функцией принадлежности, заданной следующим образом:

$$\forall x \in X\ \mu_{A-B}(x)=\left|\mu_A(x)-\mu_B(x)\right|.$$

Дизъюнктивная сумма нечетких множеств *A* и *B* (рисунок 2.9), заданных на универсальном множестве *X*, - это нечеткое множество $A\oplus B=(A\setminus B)\cup(B\setminus A)=(A\cap\overline{B})\cup(\overline{A}\cap B)$ с функцией принадлежности, заданной следующим образом:

$$\forall x \in X\ \mu_{A\oplus B}(x)=\max\{\min\{\mu_A(x);1-\mu_B(x)\};\min\{1-\mu_A(x);\mu_B(x)\}\}.$$

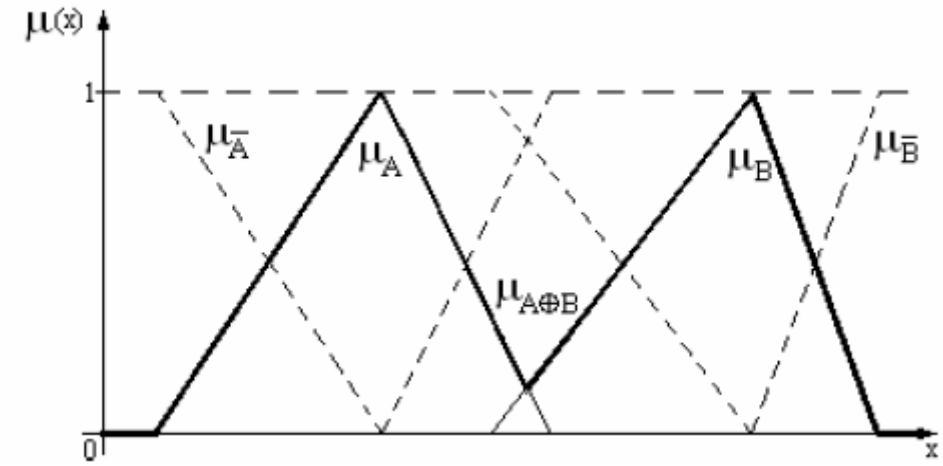


Рисунок 2.9 - Операция дизъюнктивная сумма нечетких множеств

Операция концентрирования нечетких множеств CON(A) = A^2.
Операция размывания нечетких множеств DIL(A) = $A^{0,5}$.

Операции CON(A) и DIL(A) используются при работе с лингвистическими переменными (рисунок 2.10).

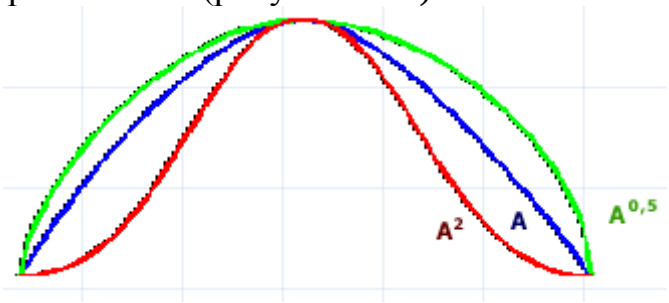


Рисунок 2.10 – Операции концентрирования и размывания

Пример. Пусть даны три нечетких множества:

$A = \{0{,}4/x_1;\ \ 0{,}2/x_2;\ \ 0/x_3;\ \ 1/x_4\};$

$B = \{0{,}7/x_1;\ \ 0{,}9/x_2;\ \ 0{,}1/x_3;\ \ 1/x_4\};$

$C = \{0{,}1/x_1;\ \ 1/x_2;\ \ 0{,}2/x_3;\ \ 0{,}9/x_4\}.$

Здесь:

1. $A \subset B$. C несравнимо ни с A ни с B.
2. $A \neq B \neq C$.

   $\bar{A} = \{0{,}6/x_1;\ \ 0{,}8/x_2;\ \ 1/x_3;\ \ 0/x_4\};$

3) $\bar{B} = \{0{,}3/x_1;\ \ 0{,}1/x_2;\ \ 0{,}9/x_3;\ \ 0/x_4\};$

   $\bar{C} = \{0{,}9/x_1;\ \ 0/x_2;\ \ 0{,}8/x_3;\ \ 0{,}1/x_4\}.$

4. $A \cap B = \{0{,}4/x_1;\ \ 0{,}2/x_2;\ \ 0/x_3;\ \ 1/x_4\}.$
5. $A \cup B = \{0{,}7/x_1;\ \ 0{,}9/x_2;\ \ 0{,}1/x_3;\ \ 1/x_4\}.$

6. $A - B = A \cap \overline{B} = \{0,3/x_1;\ \ 0,1/x_2;\ \ 0/x_3;\ \ 0/x_4\};$
   $B - A = B \cap \overline{A} = \{0,6/x_1;\ \ 0,8/x_2;\ \ 0,1/x_3;\ \ 0/x_4\}.$
7. $A \oplus B = \{0,6/x_1;\ \ 0,8/x_2;\ \ 0,1/x_3;\ \ 0/x_4\}.$

2.3. Свойства операций над нечеткими множествами

Пусть A, B и C - нечеткие множества, заданные на универсальном множестве X. Тогда для операций пересечения и объединения нечетких множеств выполняются следующие свойства:

- $\begin{cases} A \cap B = B \cap A, \\ A \cup B = B \cup A, \end{cases}$ - коммутативность;
- $\begin{cases} (A \cap B) \cap C = A \cap (B \cap C), \\ (A \cup B) \cup C = A \cup (B \cup C), \end{cases}$ - ассоциативность;
- $\begin{cases} A \cap (B \cup C) = (A \cap B) \cup (A \cap C), \\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C), \end{cases}$ - дистрибутивность;
- $\begin{cases} A \cap A = A, \\ A \cup A = A, \end{cases}$ - идемпотентность;
- $A \cup \varnothing = A$, где $\varnothing$ - пустое множество, т.е. $\mu_{\varnothing}(x) = 0$, $\forall x \in X$.
- $A \cap \varnothing = \varnothing$;
- $A \cap X = A$;
- $A \cup X = X$;
- $\begin{cases} \overline{A \cap B} = \overline{A} \cup \overline{B}, \\ \overline{A \cup B} = \overline{A} \cap \overline{B}, \end{cases}$ - теоремы Де Моргана.
- $\begin{cases} A \cap \overline{A} \neq \varnothing, \\ A \cup \overline{A} \neq X, \end{cases}$ - в отличие от аналогичных свойств для обычного (четкого)

множества, заданного на множестве E подмножества D, для которого $\begin{cases} D \cap \overline{D} = \varnothing, \\ D \cup \overline{D} = E, \end{cases}$

- $A \oplus B = (A \setminus B) \cup (B \setminus A)$.

2.4. Нечеткие операторы

Важным вопросом использования нечетких множеств в прикладных задачах является построение соответствующих операторов агрегирования

нечеткой информации и анализ их семантик. В теории нечетких множеств имеется возможность применять различные операции объединения, пересечения и дополнения множеств в зависимости от контекста и ситуации. Однако можно показать, что для любых нечетких множеств операторы F=min и G=max являются единственно возможными операторами пересечения и объединения при выполнении следующих свойств:

1. Коммутативность:

$$F(\mu_A, \mu_B) = F(\mu_B, \mu_A),$$

$$G(\mu_A, \mu_B) = G(\mu_B, \mu_A).$$

2. Ассоциативность:

$$F(\mu_A, F(\mu_B, \mu_C)) = F(F(\mu_A, \mu_B), \mu_c),$$

$$G(\mu_A, G(\mu_B, \mu_C)) = G(G(\mu_A, \mu_B), \mu_c).$$

3. Дистрибутивность:

$$F(\mu_A, G(\mu_B, \mu_C)) = G(F(\mu_A, \mu_B), F(\mu_A, \mu_c)),$$

$$G(\mu_A, F(\mu_B, \mu_C)) = F(G(\mu_A, \mu_B), G(\mu_A, \mu_c)).$$

4. Монотонность:

$$\mu_A \le \mu_C, \quad \mu_B \le \mu_D \Rightarrow F(\mu_A, \mu_B) \le F(\mu_C, \mu_D), \quad G(\mu_A, \mu_B) \le G(\mu_C, \mu_D).$$

$$\mu_A < \mu_B \Rightarrow F(\mu_A, \mu_A) < F(\mu_B, \mu_B), \quad G(\mu_A, \mu_A) \le G(\mu_B, \mu_B).$$

$$F(1,1) = 1, \quad G(0,0) = 0.$$

$$F(\mu_A, \mu_B) \le \min(\mu_A, \mu_B), \quad G(\mu_A, \mu_B) \ge \max(\mu_A, \mu_B).$$

С другой стороны, ясно, что жесткие, поточечно однозначные операторы недостаточно полно отражают смысл многозначных лингвистических преобразований термов лингвистических переменных. Поэтому большой практический интерес представляет построение обобщенных нечетких операторов, т.е. параметризованных операторов пересечения, объединения, дополнения и др. Весьма общий подход к целенаправленному формированию нечетких операторов пересечения и объединения заключается в их определении в классе треугольных норм и конорм.

Нечеткое расширение «И» - t-норма (триангулярная норма или треугольная норма) - двухместная действительная функция $T : [0,1] \times [0,1] \to [0,1]$, удовлетворяющая следующим условиям:

1. Ограниченность:

$$T(\mu_A, 0) = T(0, \mu_A) = 0,$$

$$T(\mu_A, 1) = T(1, \mu_A) = \mu_A.$$

2. Коммутативность:

$$T(\mu_A, \mu_B) = T(\mu_B, \mu_A).$$

3. Ассоциативность:

$$T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C).$$

4. Монотонность:

$$\mu_A \le \mu_C, \quad \mu_B \le \mu_D \Rightarrow T(\mu_A, \mu_B) \le T(\mu_C, \mu_D).$$

Аксиома ограниченности обеспечивает выполнение граничных условий, которые должны выполняться для всех операций пересечения нечетких множеств, включая и обычные множества.

Аксиома монотонности гарантирует неизменность порядка величин значений функций принадлежности от каких бы то ни было других функций принадлежности.

Аксиомы коммутативности и ассоциативности обеспечивают выполнение соответствующих свойств у всех операций пересечения.

График операции минимума представлен на рисунке 2.11.

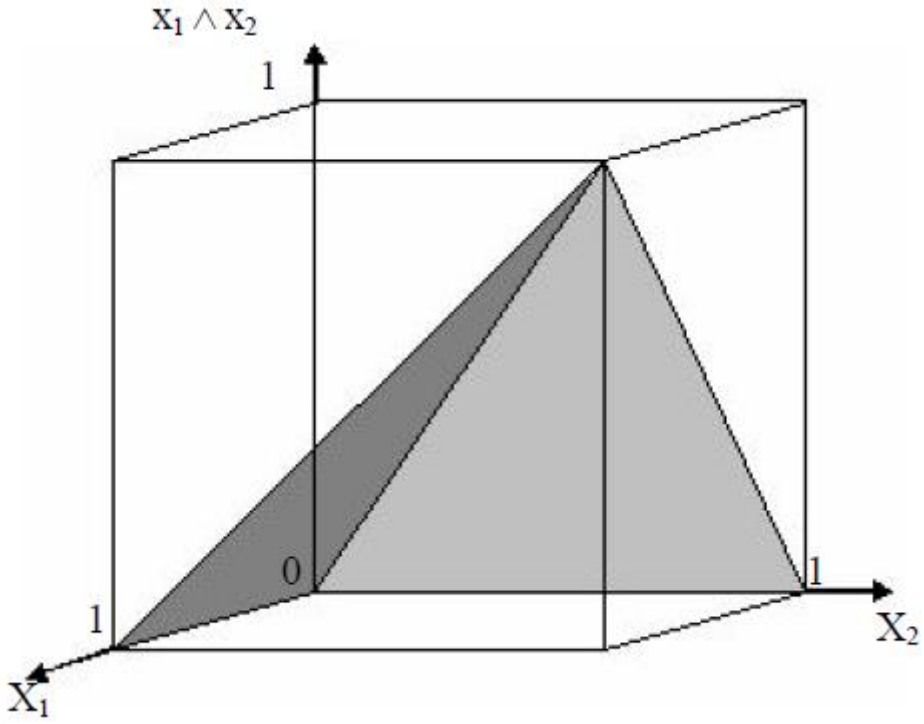


Рисунок 2.11 – Геометрическая интерпретация t-нормы

Рассмотрим геометрический смысл произвольной t-нормы.

Из аксиом $T1^0$ и $T2^0$ следует, что область определения x_1 T x_2 находится на сторонах единичного куба в плоскости (x_1; x_2). Другими словами, из аксиомы $T1^0$ следует, что на стороне $x_2 = 1$ единичного куба образуется линия x_1 T $x_2 = x_1$, на стороне $x_2 = 0$ – линия в плоскости x_1 T $x_2 = 0$. Кроме того, если использовать симметричность аксиомы $T2^0$, то на стороне $x_1 = 1$ получается прямая линия x_1 T $x_2 = x_2$, а на стороне $x_1 = 0$ – линия в плоскости x_1 T $x_2 = x_2$. Таким образом, значения x_1 T x_2 в четырех вершинах единичного куба являются также значениями четкой операции «И». К тому же из аксиомы $T2^0$ очевидно, что график симметричен относительно плоскости, образуемой наклонными $x_1 = x_2$.

На рис.13 отображены граничные условия t – нормы.

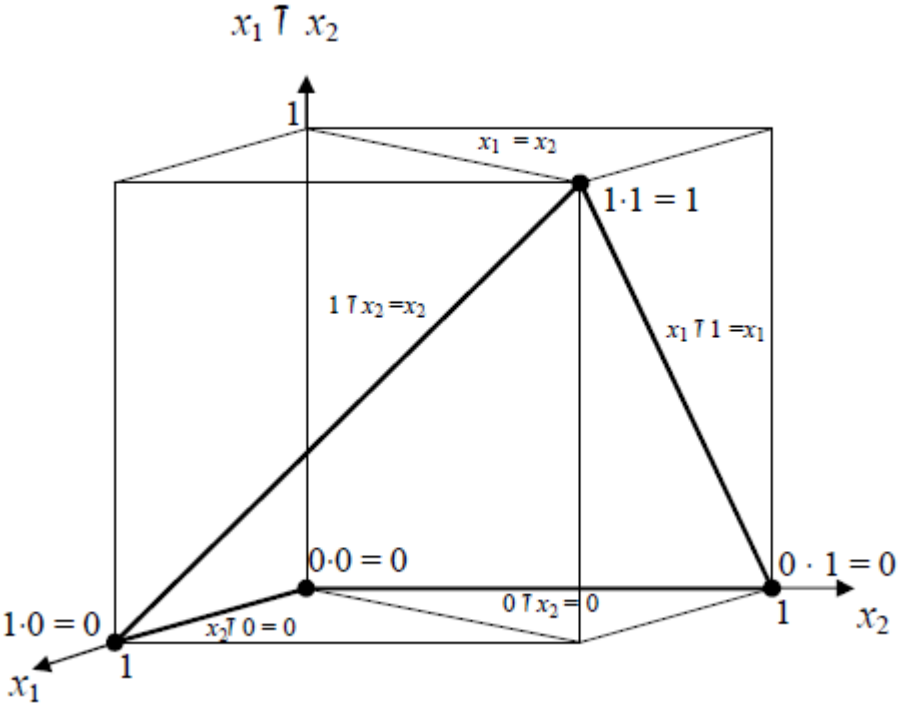


Рисунок 2.12 – Геометрическая интерпретация t-нормы

Нечеткое расширение «ИЛИ» - t-конорма (треугольная конорма или s-норма) - двухместная действительная функция $S:[0,1]\times[0,1]\to[0,1]$, удовлетворяющая следующим условиям:

1. Ограниченность:

$$S(\mu_A,1) = S(1,\mu_A) = 1,$$
$$S(\mu_A,0) = S(0,\mu_A) = \mu_A.$$

2. Коммутативность:

$$S(\mu_A, \mu_B) = S(\mu_B, \mu_A).$$

3. Ассоциативность:

$S(\mu_A, S(\mu_B, \mu_C)) = S(S(\mu_A, \mu_B), \mu_C).$

4. Монотонность:

$$\mu_A \geq \mu_C, \quad \mu_B \geq \mu_D \Rightarrow S(\mu_A, \mu_B) \geq S(\mu_C, \mu_D).$$

Аксиоматика рассмотренных операторов практически одинакова, кроме первой аксиомы.

Эта аксиома обеспечивает выполнение граничных условий, которые должны выполняться для всех операций объединения нечетких множеств, включая и обычные множества.

График операции представлен на рисунке 2.13.

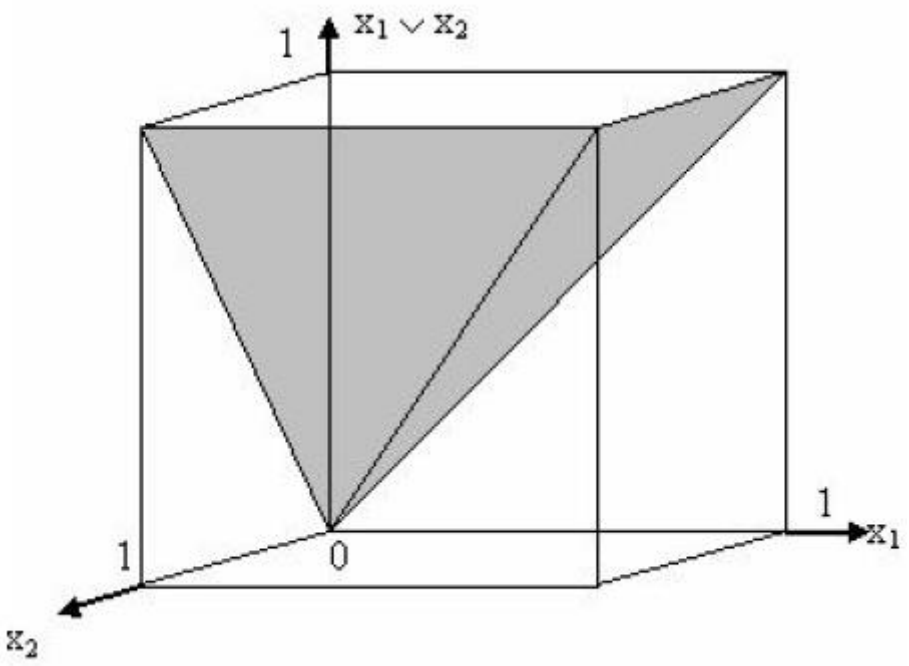


Рисунок 2.13 – Геометрическая интерпретация t-конормы

Граничные условия произвольной s-нормы отображены на рисунке 2.14.

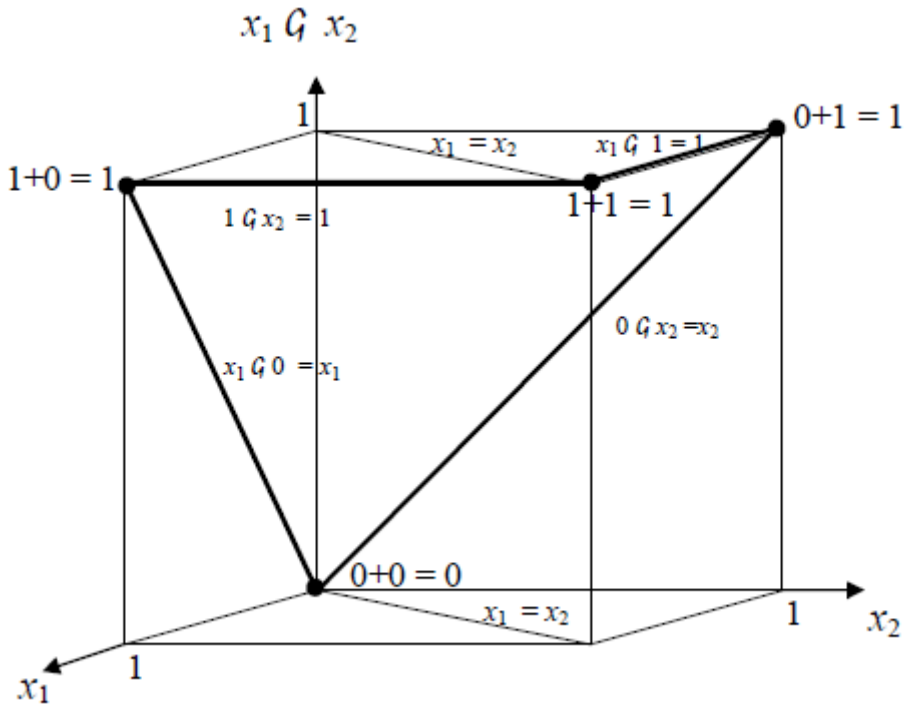


Рисунок 2.14 – Геометрическая интерпретация t-конормы

Вопросы для повторения и закрепления материала

1. Перечислите логические операции над нечеткими множествами?
2. В чем состоит отличие операций над нечеткими множествами от операций над классическим множеством?
3. Назовите свойства логических операций над нечеткими множествами?
4. Что представляет собой понятие нечеткого оператора?
5. В чем заключается суть принципа обобщения?
6. Дайте определение нечеткому оператору.
7. Укажите свойства нечетких операторов.
8. Что представляет собой треугольная норма?
9. Что такое треугольная конорма?
10. Приведите примеры нескольких видов функций принадлежности?
11. Назовите максиминные операции?
12. Перечислите операции над нечеткими множествами?

Задания для самостоятельной работы

1. Постройте функции принадлежности, характеризующие возраст человека.

Тема 3. Нечеткие отношения

Цель: изучить понятие и назначение нечеткого отношения.

Задачи:

1. Ознакомиться с основными определениями нечетких отношений.
2. Рассмотреть операции над нечеткими отношениями.
3. Рассмотреть свойства нечетких отношений.
4. Ознакомиться с принципом декомпозиции нечетких отношений.
5. Рассмотреть понятие транзитивного замыкания нечетких отношений.
6. Рассмотреть понятие проекции нечетких отношений.

3.1. Основные определения

Теория нечетких отношений находит также приложение в задачах, в которых традиционно применяется теория обычных (четких) отношений. Как правило, аппарат теории четких отношений используется при качественном анализе взаимосвязей между объектами исследуемой системы, когда связи носят дихотомический характер и могут быть проинтерпретированы в терминах "связь присутствует", "связь отсутствует", либо когда методы количественного анализа взаимосвязей по каким-либо причинам неприменимы и взаимосвязи искусственно приводятся к дихотомическому виду. Например, когда величина связи между объектами принимает значения из ранговой шкалы, выбор порога на силу связи позволяет преобразовать связь к требуемому виду. Однако, подобный подход, позволяя проводить качественный анализ систем, приводит к потере информации о силе связей между объектами либо требует проведения вычислений при разных порогах на силу связей. Этого недостатка лишены методы анализа данных, основанные на теории нечетких отношений, которые позволяют проводить качественный анализ систем с учетом различия в силе связей между объектами системы.

Подобно нечеткому множеству, **нечеткое отношение** можно задать с помощью его функции принадлежности

$$\mu_R : X_1 \times \ldots \times X_n \to L,$$

где в общем случае будем считать, что L — это полная дистрибутивная решетка. Таким образом, L — это частично упорядоченное множество, в котором любое непустое подмножество имеет наибольшую нижнюю и наименьшую верхнюю грани и операции пересечения и объединения в L удовлетворяют законам дистрибутивности. Все операции над нечеткими отношениями определяются с помощью этих операций из L. Например, если в качестве L взять ограниченное множество вещественных чисел, то операциями пересечения и объединения в L будут, соответственно, операции $\min$ и $\max$, и эти операции будут определять и операции над нечеткими отношениями.

Далее мы ограничимся рассмотрением лишь **бинарных нечетких отношений**, являющихся отображением на отрезок $[0,1]$, т.е. $\mu_R\colon X \times Y \to [0,1]$.

Если множества X и Y конечны, нечеткое отношение R между X и Y можно представить с помощью его **матрицы отношения**, первой строке и первому столбцу которой ставятся в соответствие элементы множеств X и Y, а на пересечении строки x и столбца y помещается элемент $\mu_R(x,y)$

Пример. Задать нечеткое отношение $x \approx y$ ("x приблизительно равно y"). Пусть $x,y \in \{0,1,2,3\}$. Тогда нечеткое отношение удобно задавать матрицей вида (таблица 3.1).

Таблица 3.1 – Матрица отношения R

R	**y_0**	**y_1**	**y_2**	**y_3**
x_0	1	0,5	0,2	0,1
x_1	0,5	1	0,6	0,3
x_2	0,2	0,6	1	0,8
x_3	0,1	0,3	0,8	1

Для непрерывных множеств $X = [0,3]$ и $Y = [0,3]$ нечеткое отношение можно задать следующей функцией принадлежности: $\mu_{\tilde{R}}(x,y) = e^{-0.2(x-y)^2}$. Нечеткие отношения $x \approx y$ на дискретных и непрерывных множествах изображены на рисунке 3.1.

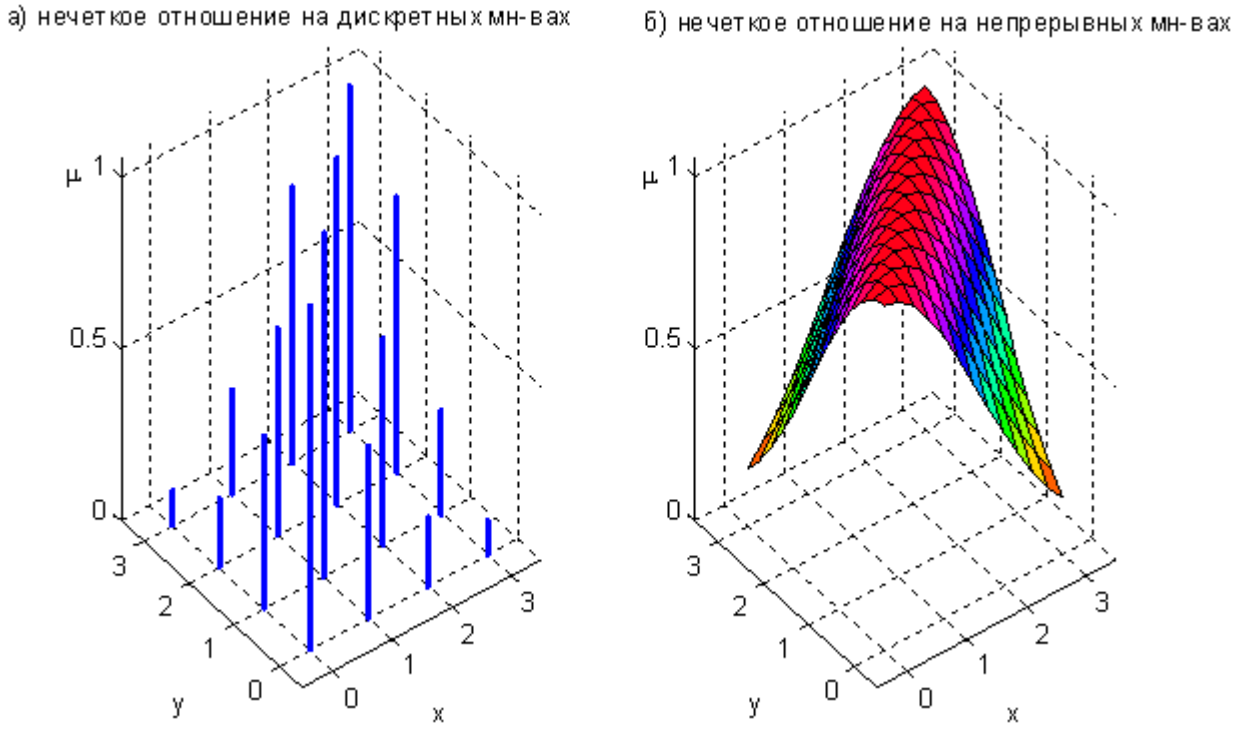


Рисунок 3.1 - Нечеткое отношение "x приблизительно равно y"

В случае, когда множества X и Y совпадают, нечеткое отношение R называют **нечетким отношением на множестве** X.

В случае конечных или счетных универсальных множеств очевидна интерпретация нечеткого отношения в виде взвешенного графа, в котором каждая пара вершин (x, y) из $X \times Y$ соединяется ребром с весом $R(x, y)$.

Пример. Пусть $X = x_1, x_2$ и $Y = y_1, y_2, y_3$, тогда нечеткий граф, изображенный на рисунке 3.2, задает некоторое нечеткое отношение $R \subset X \times Y$.

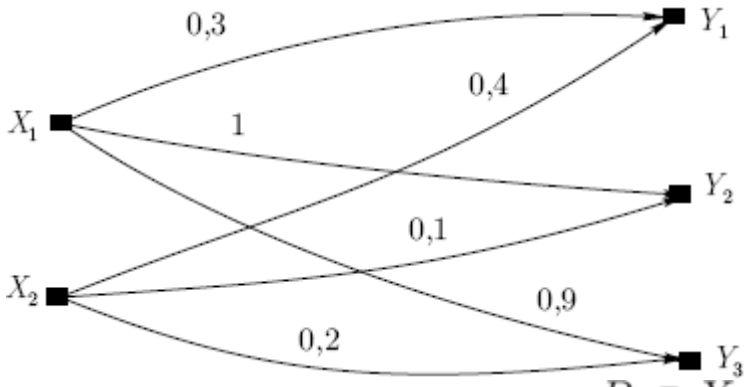


Рисунок 3.2 – Граф нечеткого отношения $R \subset X \times Y$

Пример. Задать нечеткое отношение "x намного меньше, чем y". Пусть $x, y \in \{0, 1, 2, 3\}$. Тогда нечеткое отношение можно задать матрицей вида:

$$\tilde{R} = \begin{matrix} \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \leftarrow y \\ \begin{bmatrix} 0 & 0.2 & 0.6 & 1 \\ 0 & 0 & 0.2 & 0.6 \\ 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ x \downarrow \end{matrix} \begin{matrix} \\ 0 \\ 1 \\ 2 \\ 3 \end{matrix}.$$

Для непрерывных множеств $X = [0,3]$ и $Y = [0,3]$ нечеткое отношение "x намного меньше, чем y" можно определить такой функцией принадлежности:

$$\mu_{\tilde{R}}(x, y) = \begin{cases} 0, & \text{если } x \geq y \\ \dfrac{1}{1 + 5/(x - y)^4}, & \text{если } x < y \end{cases}.$$

Нечеткие отношения "x намного меньше, чем y" на дискретных и непрерывных множествах изображены на рисунке 3.3.

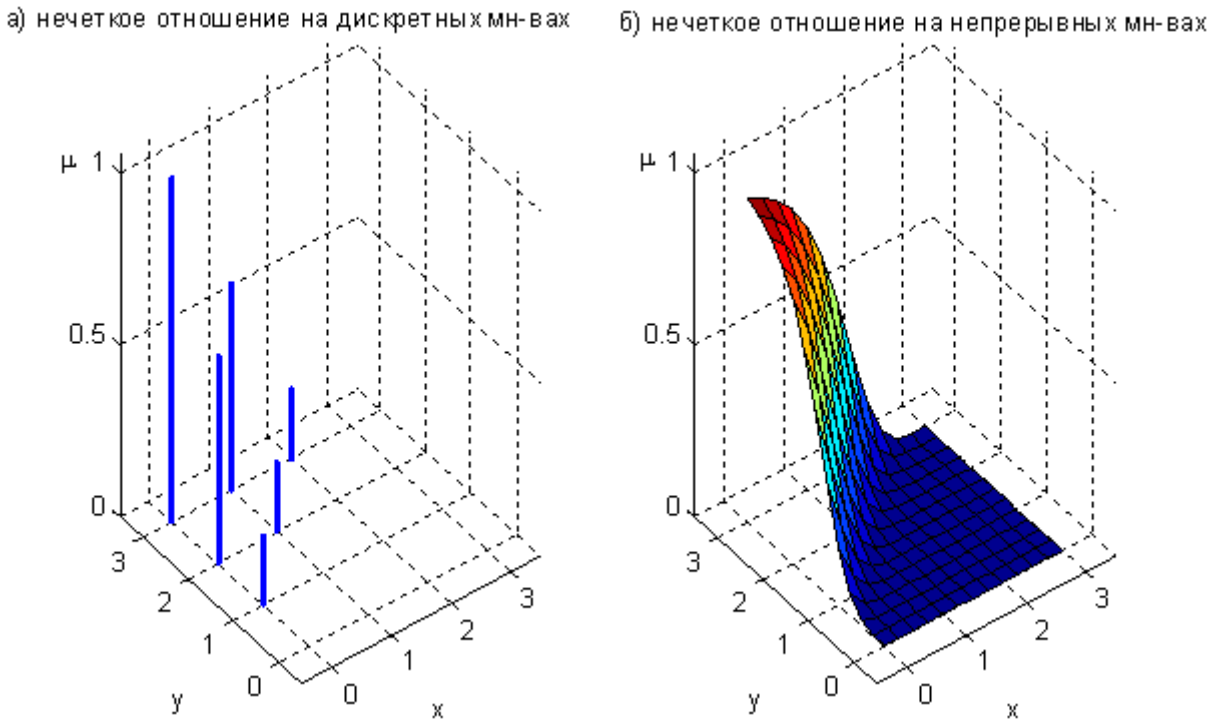


Рисунок 3.3 - Нечеткое отношение "x намного меньше, чем y"

Как видно из примеров, нечеткие отношения являются более гибкими по сравнению с традиционными отношениями. Они позволяют задать не только сам факт выполнения отношения, но и указывать степень его выполнения, что является очень важным для многих практических задач.

Пример. Задать отношение "схожий менталитет" для следующих национальностей {Украинцы(У), Чехи (Ч), Австрийцы (А), Немцы (Н)}.

Использование обычного, не нечеткого отношения позволяет выделить только одну пару наций со схожими менталитетами - немцев и австрийцев. Этим отношением не отражается тот факт, что по менталитету чехи более близки к немцам, чем украинцы. Нечеткое отношение позволяет легко представить такую информацию:

$$\tilde{R} = \begin{array}{c} \begin{matrix} У & Ч & А & Н \end{matrix} \\ \begin{bmatrix} 1 & 0.4 & 0.2 & 0.1 \\ 0.4 & 1 & 0.4 & 0.3 \\ 0.2 & 0.4 & 1 & 0.8 \\ 0.1 & 0.3 & 0.8 & 1 \end{bmatrix} \end{array} \begin{matrix} У \\ Ч \\ А \\ Н \end{matrix}.$$

3.2. Операции над нечеткими отношениями

Объединение и пересечение нечетких отношений определяется следующим образом:

$$\forall x \in X \; \forall y \in Y \qquad R \cup S\,(x,y) = R(x,y) \vee S(x,y),$$

$$\forall x \in X \; \forall y \in Y \qquad R \cap S\,(x,y) = R(x,y) \wedge S(x,y)$$

Пересечение нечетких отношений $\tilde{A}$ и $\tilde{B}$, заданных на $X \times Y$, называется нечеткое отношение $\tilde{C} = \tilde{A} \cap \tilde{B}$ с функцией принадлежности $\mu_{\tilde{C}}(x,y) = t(\mu_{\tilde{A}}(x,y), \mu_{\tilde{B}}(x,y))$, $(x,y) \in X \times Y$, где $t(.)$ - t-норма.

Объединение нечетких отношений $\tilde{A}$ и $\tilde{B}$, заданных на $X \times Y$, называется нечеткое отношение $\tilde{D} = \tilde{A} \cup \tilde{B}$ с функцией принадлежности $\mu_{\tilde{D}}(x,y) = s(\mu_{\tilde{A}}(x,y), \mu_{\tilde{B}}(x,y))$, $(x,y) \in X \times Y$, где $s(.)$ - s-норма (t-конорма).

Пересечение и объединение нечетких отношений x приблизительно равно y и x намного меньше, чем y показаны на рисунке 3.4. В качестве t-нормы и s-нормы использовались операции нахождения минимума и максимума, соответственно.

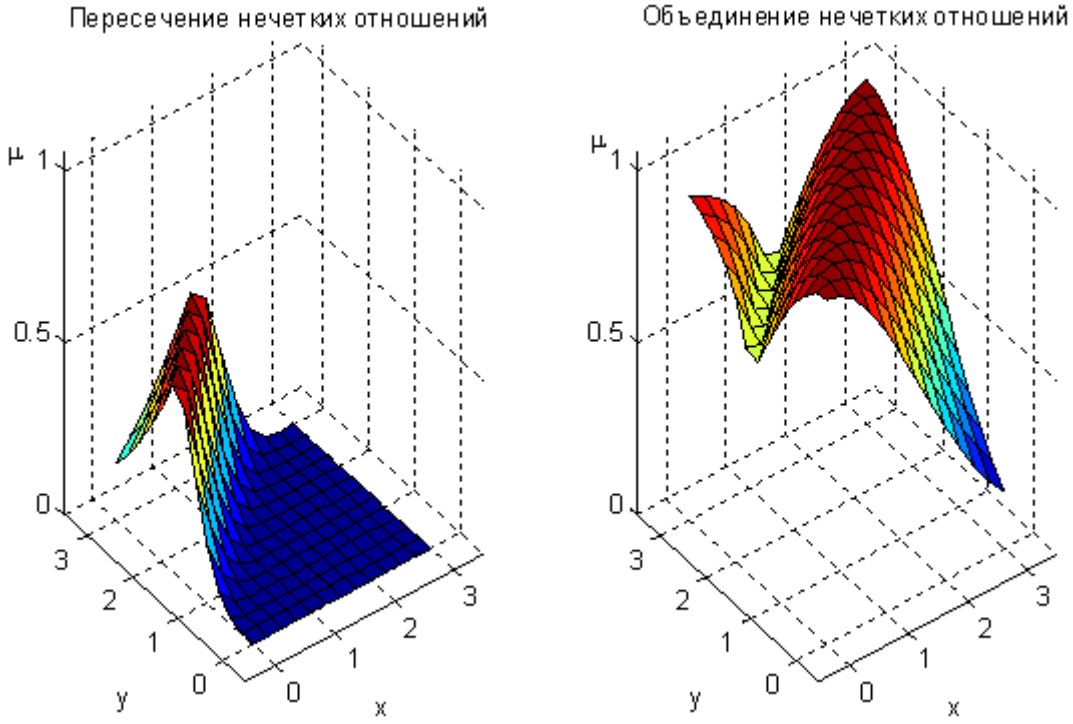


Рисунок 3.4 -Операции над нечеткими отношениями из примеров 5 и 6

Дополнением нечеткого отношения $\tilde{R}$, заданного на $X \times Y$, называется нечеткое отношение $\tilde{R}'$ с функцией принадлежности , $(x,y) \in X \times Y$.

Отношение включения $R \subseteq S$ для нечетких отношений определяется с помощью отношения частичного порядка на L:

$$\forall x \in X \; \forall y \in Y \qquad R \subseteq S \Leftrightarrow R(x,y) \leqslant S(x,y).$$

Множество $\rho(X \times Y)$ всех нечетких отношений между X и Y образует дистрибутивную решетку по отношению к операциям объединения и пересечения и удовлетворяет следующим тождествам:

1. Идемпотентность: $R \cap R = R, \quad R \cup R = R.$
2. Коммутативность: $R \cap S = S \cap R, \quad R \cup S = S \cup R.$

3. Ассоциативность:

$$R \cap (S \cap T) = (R \cap S) \cap T, \quad R \cup (S \cup T) = (R \cup S) \cup T.$$

4. Дистрибутивность:

$$R \cap (S \cup T) = (R \cap S) \cup (R \cap T), \quad R \cup (S \cap T) = (R \cup S) \cap (R \cup T).$$

Выполнение этих тождеств для $\rho(X \times Y)$ следует из выполнения соответствующих тождеств для решетки L. В $\rho(X \times Y)$ выполняется также следующее соотношение:

$$S \subseteq T \quad \Rightarrow \quad R \cup S \subseteq R \cup T, \quad R \cap S \subseteq R \cap T.$$

Из полноты решетки L следует, что она обладает наименьшим **0** и наибольшим **I** элементами. Эти элементы определяют, соответственно, **пустое** и **универсальное** нечеткие отношения:

$$\forall x \, \forall y \; \Theta(x, y) = 0, \qquad \forall x \, \forall y \; U(x, y) = I.$$

Максминной композицией (произведением) нечетких отношений $\tilde{A}$ и $\tilde{B}$, заданных на X и Y, называется нечеткое отношение $\tilde{G} = \tilde{A} \circ \tilde{B}$ на множестве $X \times Y$ с функцией принадлежности $\mu_{\tilde{G}}(x, y) = \sup_{z \in Z} \min\left(\mu_{\tilde{A}}(x, z), \mu_{\tilde{B}}(z, y)\right)$, $(x,y) \in X \times Y$, $(x,z) \in X \times Z$, $(z,y) \in Z \times Y$. В случае конечных множеств X, Y, Z матрица нечеткого отношения $\tilde{G} = \tilde{A} \circ \tilde{B}$ получается как максминное произведение матриц $\tilde{A}$ и $\tilde{B}$. Эта операция выполняется как обычное произведение матриц, в котором операция поэлементного умножения заменена на нахождение минимума, а суммирование - на нахождение максимума. Аналогично определяются операции минимаксной и максимультипликативной композиции. Композиция играет ключевую роль в нечетком логическом выводе.

Пример. Заданы нечеткие отношения $\tilde{A} = \begin{bmatrix} 0.1 & 0.2 \\ 0.8 & 1 \end{bmatrix}$ и $\tilde{B} = \begin{bmatrix} 0.6 & 0.4 \\ 0.5 & 0.3 \end{bmatrix}$. Тогда максминная ($\tilde{G}_1$), минимаксная ($\tilde{G}_2$) и максимультипликативная ($\tilde{G}_3$) композиции этих нечетких отношений определяются такими матрицами: $\tilde{G}_1 = \begin{bmatrix} 0.1 & 0.1 \\ 0.5 & 0.3 \end{bmatrix}$; $\tilde{G}_2 = \begin{bmatrix} 0.5 & 0.3 \\ 0.8 & 0.8 \end{bmatrix}$; $\tilde{G}_3 = \begin{bmatrix} 0.1 & 0.06 \\ 0.5 & 0.32 \end{bmatrix}$.

Существуют и другие варианты операции композиции, которые определяются с помощью дополнительных операций, выводимых в L. В зависимости от того, является ли L множеством векторов, множеством лингвистических переменных или множеством чисел, эти дополнительные операции будут иметь соответствующий вид. Например, если L является множеством действительных чисел, то операция $\wedge$ может быть заменена на операцию взятия среднего арифметического, что дает другое определение операции композиции:

$$\forall x \in X \; \forall z \in Z \quad R \circ S(x, y) = \bigvee_{y \in Y} (0,5(R(x, y) + S(y, z))) .$$

В случае $L = [0, 1]$ мы имеем

$$\forall x \in X \; \forall z \in Z \quad \mu_{R \circ S}(x, z) = \bigvee_{e \in Y} (\mu_R(x, y) \wedge \mu_S(x, y)).$$

Замена операции $\wedge$ на операцию умножения дает следующее определение композиции:

$$\forall x \in X \, \forall z \in Z \; \mu_{R \circ S}(x, z) = \bigvee_{e \in Y} (\mu_R(x, y) \cdot \mu_S(x, y)).$$

Нечеткое отношение E такое, что

$$E(x, y) = \begin{cases} I, & \text{если } x = y, \\ 0 & \text{в противном случае.} \end{cases}$$

играет по отношению к операции композиции роль единицы: $E \circ R = R \circ E = R$. В теории четких отношений отношение E называется **отношением равенства**.

Для любого нечеткого отношения R определяется также обратное отношение R^{-1}:

$$\forall x, y \in X \quad R^{-1}(x, y) = R(x, y).$$

3.3. Свойства нечетких отношений

Различные типы нечетких отношений определяются с помощью свойств, аналогичных свойствам обычных отношений, причем для нечетких отношений можно указать различные способы обобщения этих свойств.

1. **Рефлексивность**: если для любого $x \in X$ выполняется равенство $\mu_{\tilde{R}}(x, x) = 1$. В случае конечного множества X все элементы главной диагонали матрицы $\tilde{R}$ равны 1. Примером рефлексивного нечеткого отношения может быть отношение "приблизительно равны".

$$E \subseteq R, \qquad \forall x \in X \quad R(x, x) = I.$$

2. **Слабая рефлексивность**:

$$\forall x, y \in X \quad R(x, y) \leqslant R(x, x).$$

3. **Сильная рефлексивность**:

$$\forall x, y \in X \quad R(x, y) < I.$$

4. **Антирефлексивность**: если для любого $x \in X$ выполняется равенство $\mu_{\tilde{R}}(x, x) = 0$.В случае конечного множества X все элементы главной диагонали матрицы $\tilde{R}$ равны 0. Примером антирефлексивного нечеткого отношения может быть отношение "значительно больше".

$$R \cap E = \varnothing \qquad \forall x \in X \quad R(x, x) = 0.$$

5. **Слабая антирефлексивность**:

$$\forall x, y \in X \quad R(x, x) \leqslant R(x, y).$$

6. **Сильная антирефлексивность**:

$$\forall x, y \in X \quad 0 < R(x, y).$$

7. **Симметричность**: если для любой пары $(x,y) \in X \times Y$ выполняется равенство $\mu_{\tilde{R}}(x, y) = \mu_{\tilde{R}}(y, x)$.Матрица симметричного нечеткого отношения, заданного на конечном множестве, симметричная.

$$R = R^{-1}, \qquad \forall x, y \in X \quad R(x, y) = R(y, x).$$

8. **Антисимметричность**:

$$R \cap R^{-1} \subseteq E, \qquad \forall x, y \in X (x \neq y) \quad R(x, y) \wedge R(y, x) = 0.$$

9. **Асимметричность:** если выражение $\mu_{\tilde{R}}(x, y) > 0 \Rightarrow \mu_{\tilde{R}}(y, x) = 0$ справедливо для любой пары $(x,y) \in X \times Y$. Примером асимметричного нечеткого отношения может служить отношение "намного больше".

$$R \cap R^{-1} = \varnothing, \qquad \forall x, y \in X \quad R(x, y) \wedge R(y, x) = 0.$$

10. **Сильная линейность**:

$$R \cup R^{-1} = U, \qquad \forall x, y \in X \quad R(x, y) \vee R(y, x) = I.$$

11. **Слабая линейность**:

$$\forall x, y \in X \quad R(x, y) \vee R(y, x) > 0.$$

12. **Транзитивность**:

$$R \supseteq R \circ R, \qquad \forall x, y, z \in X \quad R(x, z) \geqslant R(x, y) \wedge R(y, z).$$

3.4. Декомпозиция нечетких отношений

Одно из важнейших свойств нечетких отношений заключается в том, что они могут быть представлены в виде совокупности обычных отношений, причем могут быть упорядочены по включению, представляя собой иерархическую совокупность отношений.

Носителем нечеткого отношения $\tilde{R}$ на множествах X и Y называется подмножество декартова произведения $X \times Y$ вида: $\mathrm{supp}\tilde{R} = \{(x,y) : (x,y) \in X \times Y, \mu_{\tilde{R}}(x, y) > 0\}$.

Носитель нечеткого отношения можно рассматривать как обычное отношение, связывающего все пары $(x,y) \in X \times Y$, для которых степень выполнения нечеткого отношения $\tilde{R}$ не равна нулю. Более полезным является использование α-сечений нечеткого отношения, определения которых аналогично определениям множеств α-уровня

Разложение нечеткого отношения на совокупность обыкновенных отношений основано на понятии α - уровня нечеткого отношения. Здесь для простоты будем полагать, что L линейно упорядочено.

α -уровнем нечеткого отношения R называется обычное отношение R_α, определяемое для всех $\alpha > 0$ следующим образом:

$$R_\alpha = \left\{(x, y) \in X^2 | R(x, y) \geqslant \alpha\right\}.$$

Очевидно, что α -уровни нечетких отношений удовлетворяют соотношению:

$$\alpha \leqslant \beta \quad \Rightarrow \quad R_\alpha \supseteq R_\beta,$$

представляя собой совокупность вложенных друг в друга отношений.

Теорема. Нечеткое отношение R обладает каким-либо свойством из перечисленных (кроме сильной рефлексивности, сильной антирефлексивности, слабой линейности) тогда и только тогда, если этим свойством обладают все его α -уровни.

Эта теорема играет важную роль в теории нечетких отношений. Во-первых, она показывает, что основные типы обычных отношений и их свойства могут быть обобщены и на случай нечетких отношений, и приводит ясный способ такого обобщения. Во-вторых, оказывается, что основные типы нечетких отношений могут быть представлены как совокупность, иерархия обычных отношений того же типа. И если решением практической задачи является получение на множестве X некоторого отношения заданного типа, например эквивалентности или порядка, то построение на X соответствующего нечеткое отношение позволяет получать сразу ансамбль необходимых обычных отношений, а это дает возможность учитывать неоднозначность решений, присущих практическим ситуациям, и предоставляет лицу, принимающему решение, некоторую свободу выбора. В-третьих, теория нечетких множеств, допуская подобную неоднозначность возможных решений, ограничений и целей, дает возможность оперировать сразу всей совокупностью таких объектов как единым целым.

Нечеткое отношение R может быть представлено в следующем виде:

$$R = \bigcup_\alpha \alpha R_\alpha,$$

где отношения αR_α определяются следующим образом:

$$\alpha R_\alpha(x,y) = \begin{cases} \alpha, & \text{если } R_\alpha(x,y) = 1, \\ 0 & \text{в противном случае.} \end{cases}$$

Кроме всех вышеописанных свойств, выполняющихся для всех α -уровней, могут быть определены аналогичные свойства, выполняющиеся только для одного или нескольких α -уровней. Приведем примеры таких α - свойств, предполагая, что элемент α фиксированный:

α -симметричность

$$\forall x, y \in X \quad R(x,y) \geqslant \alpha \quad \Rightarrow \quad R(y,x) \geqslant \alpha;$$

α -транзитивность

$$\forall x, y, z \in X \quad R(x,y) \geqslant \alpha, R(y,z) \geqslant \alpha \quad \Rightarrow \quad R(x,z) \geqslant R(x,y)$$

Аналогично могут быть определены и другие α -свойства. Они могут рассматриваться в задачах, в которых вводится порог на силу отношения R либо ищется такое α, при котором R_α обладает требуемым свойством.

3.5. Транзитивное замыкание нечетких отношений

Большое значение в приложениях теории нечетких отношений играют транзитивные отношения. Они обладают многими удобными свойствами и определяют некоторую правильную структуру множества X. Например, если отношение R в X характеризует сходство между объектами, то транзитивность такого отношения обеспечивает возможность разбиения множества X на непересекающиеся классы сходства. Если же отношению в X придать смысл "предпочтения" или "доминирования", то транзитивность такого отношения обеспечивает возможность естественного упорядочения объектов множества X, существование "наилучших", "недоминируемых" объектов и т.п. Поэтому представляет большой интерес возможность преобразования исходного нетранзитивного отношения в транзитивное. Такое преобразование обеспечивает операция транзитивного замыкания нечеткого отношения.

Транзитивным замыканием отношения R называется отношение $\hat{R}$, определяемое следующим образом:

$$\hat{R} = R^1 \cup R^2 \cup \ldots \cup R^k \cup \ldots,$$

где отношения R^k определяются рекурсивно:

$$R^1 = R, \quad R^k = R^{k-1} \circ R, \quad k = 2, 3, 4, \ldots.$$

Теорема. Транзитивное замыкание $\hat{R}$ любого нечеткого отношения R транзитивно и является наименьшим транзитивным отношением, включающим R, т.е. $R \subseteq \hat{R}$, и для любого транзитивного отношения T, такого, что $R \subseteq T$, следует $\hat{R} \subseteq T$.

Как следствие из данной теоремы получаем, что R транзитивно тогда и только тогда, если $R = \hat{R}$.

Если множество X содержит n элементов, то имеем

$$\hat{R} = R^1 \cup R^2 \cup \ldots \cup R^n.$$

В случае, когда R рефлексивно, имеем

$$R \subseteq R^1 \subseteq \ldots \subseteq R^{n-1} = R^n = R^{n+1} = \ldots$$

Весьма полезным фактором является то, что α -уровень транзитивного замыкания нечеткого отношения R совпадает с транзитивным замыканием соответствующего α -уровня:

$$(\hat{R})_\alpha = (\hat{R}_\alpha), \qquad \text{для всех} \quad \alpha \neq 0.$$

Заметим, что при транзитивном замыкании нечеткого отношения R в общем случае сохраняются лишь некоторые свойства отношения R. Такими свойствами являются рефлексивность, симметричность, линейность и транзитивность.

3.6. Проекции нечетких отношений

Важную роль в теории нечетких множеств играет понятие проекции нечеткого отношения. Дадим определение проекции бинарного нечеткого отношения.

Пусть $\mu_Q(x,y)$ — функция принадлежности нечеткого отношения в $U \times V$. Проекции Q_U и Q_V отношения Q на U и V — есть множества в U и V с функцией принадлежности вида

$$\mu_{Q_U}(x) = \sup_V \mu_Q(x,y),$$

$$\mu_{Q_V}(x) = \sup_U \mu_Q(x,y).$$

Условной проекцией нечеткого отношения Q на U, при произвольном фиксированном $y_0 \in V$, называется множество P_U с функцией принадлежности вида $\mu_{P_U}(x|y_0) = \mu_Q(x,y_0)$.

Аналогично определяется условная проекция на V при заданном $x_0 \in U$:

$$\mu_{P_V}(y|x_0) = \mu_Q(x_0,y).$$

Из данного определения видно, что проекции Q_U и Q_V не влияют на условные проекции P_U и P_V, соответственно. Дадим далее определение, которое учитывает их взаимосвязь.

Условные проекции второго типа определяются следующим образом:

$$\mu_{P_U}(x|y_0) = \frac{\mu_Q(x,y_0)}{\mu_{Q_V}(y_0)}, \quad \mu_{Q_V}(y_0) > 0,$$

$$\mu_{P_U}(y|x_0) = \frac{\mu_Q(x_0,y)}{\mu_{Q_U}(x_0)}, \quad \mu_{Q_U}(x_0) > 0.$$

Если $\mu_{Q_V}(y_0) = 0$ или $\mu_{Q_U}(x_0) = 0$, то полагаем, соответственно, что $\mu_{P_U}(x|y_0) = 0$ или $\mu_{P_U}(y|x_0) = 0$.

Заметим, что условные проекции первого типа содержатся в соответствующих проекциях второго типа.

Пусть U и V — базовые множества, Q — нечеткое отношение в $U \times V$ и Q_U и Q_V — его проекции на U и V, соответственно.

Нечеткие множества Q_U и Q_V называются **независимыми**, если $Q = Q_U \times Q_V$.

Следовательно, они независимы по первому типу, если $\mu_Q(x,y) = \mu_{Q_U}(x) \wedge \mu_{Q_V}(y)$, и независимы по второму типу, если $\mu_Q(x,y) = \mu_{Q_U}(x) \cdot \mu_{Q_V}(y)$.

В противном случае проекции Q_U и Q_V являются зависимыми (соответствующего типа).

Независимость второго типа можно интерпретировать следующим образом. Данные соотношения с учетом произвольности x_0 и u_0перепишем в виде

$$\mu_Q(x,y) = \mu_{P_U}(x|y)\mu_{Q_V}(y),$$
$$\mu_Q(x,y) = \mu_{P_V}(y|x)\mu_{Q_U}(x).$$

Вопросы для повторения и закрепления материала

1. Дайте определение нечеткому отношению?
2. Каким образом задается нечеткое отношение?
3. Приведите пример операций над нечеткими отношениями?
4. Назовите свойства нечетких отношений?
5. Что такое декомпозиций нечетких отношений?
6. Дайте определение транзитивному замыканию нечеткого отношения?
7. Что такое проекция нечеткого отношения?

Задания для самостоятельной работы

1. Разработайте несколько пример нечетких отношений и охарактеризуйте их свойства.

Тема 4. Виды функций принадлежности нечеткого множества

Цель: изучить виды и математическое описание функций принадлежности нечетких множеств.

Задачи:

1. Изучить кусочно-линейные функции принадлежности.
2. Изучить Z-образные и S-образные функции принадлежности.
3. Изучить П-образные функции принадлежности.
4. Ознакомиться со встроенными функциями принадлежности в Fuzzy Logic Toolbox.

4.1. Кусочно-линейные функции принадлежности

В качестве первого типа функций принадлежности рассмотрим функции, которые, как следует из их названия, состоят из отрезков прямых линий, образуя непрерывную или кусочно-непрерывную функцию. Наиболее характерным примером таких функций являются "треугольная" (рисунок 4.1, а) и "трапециевидная" (рисунок 4.1, б) функции принадлежности. В нашем случае каждая из этих функций задана на универсуме X=[0, 10], в качестве которого выбран замкнутый интервал действительных чисел. В общем

случае выбор универсума может быть произвольным, и не ограничен никакими правилами.

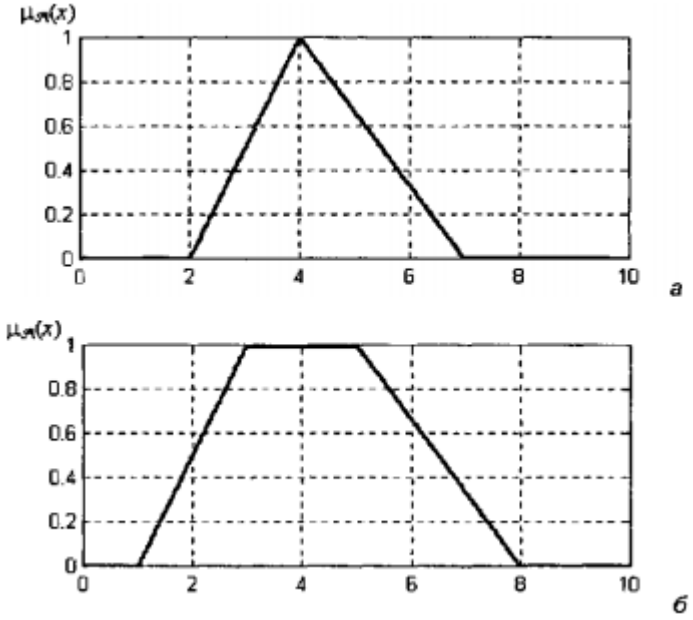


Рисунок 4.1 - Графики функций принадлежности треугольной (а) и трапециевидной (б) формы

Первая из этих функций принадлежности в общем случае может быть задана аналитически следующим выражением:

$$f_{\Delta}(x:a,b,c)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a<x\le b\\ \dfrac{c-x}{c-b}, & b<x\le c\\ 0, & c<x\end{cases}$$

где a, b, c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a\le b\le c$.

Применительно к конкретной функции, изображенной на рисунке 4.1, а, значения параметров равны: a=2, b=4, c=7. Как нетрудно заметить, параметры a и c характеризуют основание треугольника, а параметр b — его вершину. Как можно заметить, эта функция принадлежности порождает нормальное выпуклое унимодальное нечеткое множество с носителем — интервалом (a, c), границами (a, c) \{b}, ядром {b} и модой b.

Трапециевидная функция принадлежности в общем случае может быть задана аналитически следующим выражением:

$$f_T(x:a,b,c,d)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a<x\le b\\ 1, & b<x\le c\\ \dfrac{d-x}{d-c}, & c<x\le d\\ 0, & d<x\end{cases}$$

где a, b, c, d— некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a\le b\le c\le d$.

Применительно к конкретной функции, изображенной на рисунке 4.1, б, значения параметров равны: a=1, b=3, c=5, d=8. Как нетрудно заметить, параметры a и d характеризуют нижнее основание трапеции, а параметры b и c— верхнее основание трапеции. При этом данная функция принадлежности порождает нормальное выпуклое нечеткое множество с носителем — интервалом (a, d), границами (a, b) (c, d) и ядром [b, c].

Эти функции используются для задания таких свойств множеств, которые характеризуют неопределенность типа: "приблизительно равно", "среднее значение", "расположен в интервале", "подобен объекту", "похож на предмет" и др. Они также служат, для представления нечетких чисел и интервалов, которые будут рассмотрены далее.

4.2. Z-образные и S-образные функции принадлежности

Эти функции принадлежности также получили свое название по виду кривых, которые представляют их графики. Первая из функций этой группы называется Z-образной кривой или сплайн-функцией и в общем случае может быть задана аналитически следующим выражением:

$$\boldsymbol{f_{z_1}(x;,a,b)=\begin{cases}1, & x\le a\\ \dfrac{1}{2}+\dfrac{1}{2}\cos\left(\dfrac{x-a}{b-a}\pi\right), & a\le x\le b\\ 0, & x>b\end{cases}},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b. График этой функции для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.2, а, при этом значения параметров соответственно равны a=3, b=6.

Сплайн-функция может быть также задана другим выражением:

Общий функции может быть также задана др.

$$f_{z_2}(x;,a,b)=\begin{cases}1, & x\le a\\ 1-2\left(\dfrac{x-a}{b-a}\right)^2, & a<x\le\dfrac{a+b}{2}\\ 2\left(\dfrac{b-x}{b-a}\right)^2, & \dfrac{a+b}{2}<x<b\\ 0, & b\le x\end{cases},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b. График этой функции для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.2, б, при этом значения параметров соответственно равны a=3, b=6.

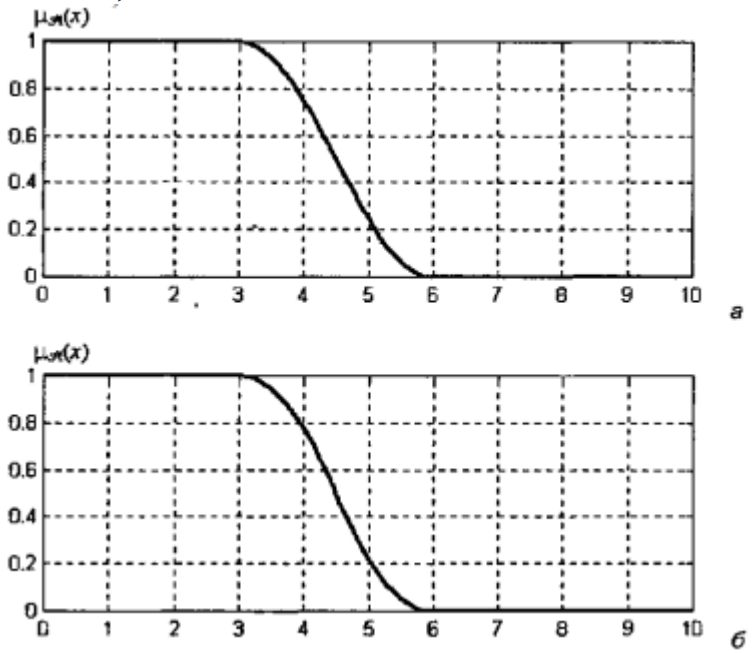


Рисунок 4.2 - Графики Z-образных функций принадлежности и для значений параметров a=3, b=6

Данные функции принадлежности порождают нормальные выпуклые нечеткие множества с ядром (- , a] и носителем (- , b).

Эти функции используются для представления таких свойств нечетких множеств, которые характеризуются неопределенностью типа: "малое количество", "небольшое значение", "незначительная величина", "низкая себестоимость продукции", "низкий уровень цен или доходов", "низкая процентная ставка" и многих других. Общим для всех таких ситуаций является слабая степень проявления того или иного качественного или количественного признака. Особенность нечеткого моделирования при этом заключается в представлении соответствующих нечетких множеств с помощью невозрастающих (монотонно убывающих) функций принадлежности.

Вторая из функций рассматриваемой группы называется S-образной кривой или сплайн-функцией и в общем случае может быть задана аналитически следующим выражением:

$$f_{S1}(x;,a,b)=\begin{Bmatrix}0, & x<a\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right), & a\le x\le b\\ 1, & x>b\end{Bmatrix},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b. График этой функции для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.3, а, при этом значения параметров соответственно равны a=3, b=6.

Сплайн-функция может быть также задана другим выражением:

$$f_{S2}(x;,a,b)=\begin{Bmatrix}0, & x\le a\\ 2\left(\frac{x-a}{b-a}\right)^2, & a<x\le\frac{a+b}{2}\\ 1-2\left(\frac{b-x}{b-a}\right)^2, & \frac{a+b}{2}<x<b\\ 1, & b\le x\end{Bmatrix},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b. График этой функции для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.3, б, при этом значения параметров соответственно равны a=3, b=6.

Данные функции принадлежности порождают нормальные выпуклые нечеткие множества с ядром [b, +) и носителем (a, +).

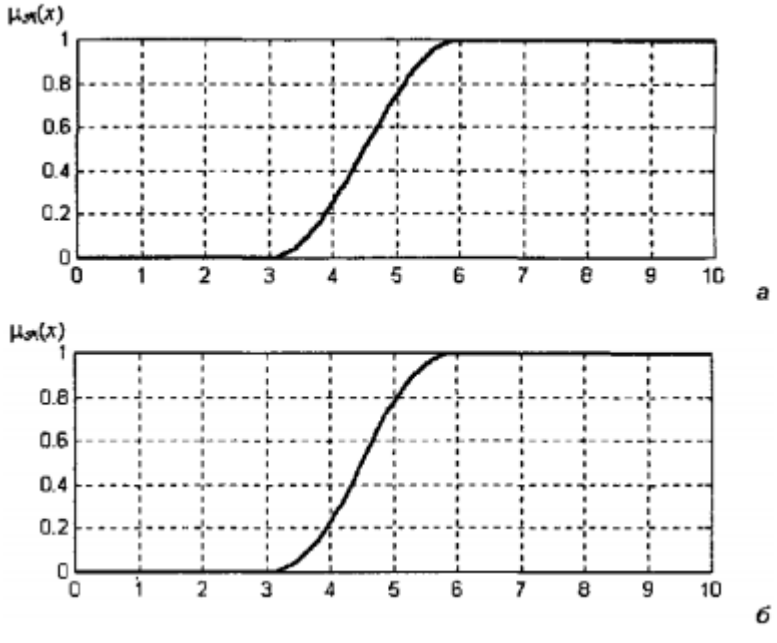


Рисунок 4.3 - Графики S-образных функций принадлежности fS1 и fS2 для значений параметров a=3, b=6

К типу S-образных и одновременно Z-образных функций принадлежности может быть отнесена так называемая сигмоидальная функция (сигмоид), которая в общем случае задается аналитически следующим выражением:

$$f_{S3}(x;a,b)=\frac{1}{1+e^{-a(x-b)}},$$

здесь a, b— некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b, а e— основание натуральных логарифмов, которое инициирует задание соответствующей экспоненциальной функции. При этом в случае a>0 может быть получена S-образная функция принадлежности, а в случае a<0 — Z-образная функция принадлежности.

Графики этой функции для некоторого нечеткого множества и универсума X=[0, 10] изображены на рисунке 4.4. При этом S-образной функции принадлежности соответствуют значения параметров a=3, b=6 (рисунок 4.4, а), а Z-образной функции принадлежности соответствуют значения параметров a=-3, b=6 (рисунок 4.4, б).

Данные функции принадлежности порождают субнормальные выпуклые нечеткие множества с носителем и границей и точкой перехода b.

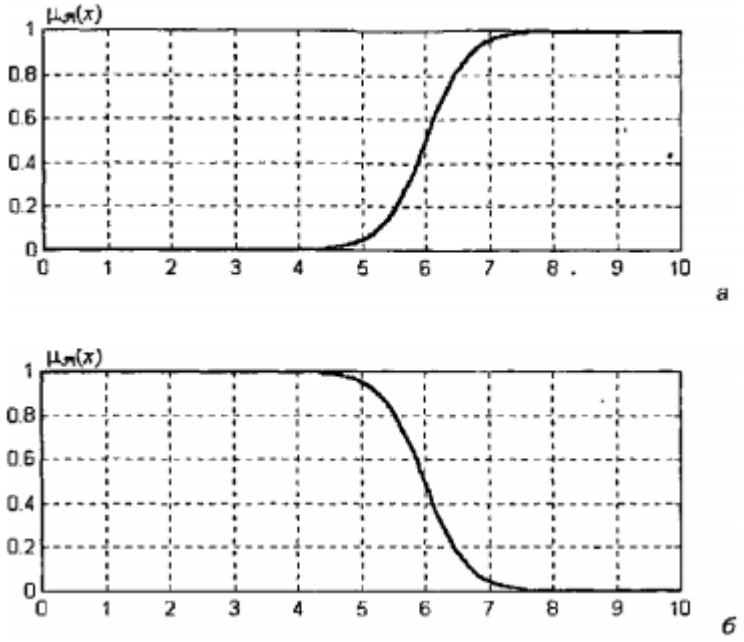


Рисунок 4.4 - Графики сигмоидальной функции принадлежности fS3 для значений параметров a=3, b=6 (а) и a=-3, b=6 (б)

Рассмотренные S-образные функции используются для представления таких нечетких множеств, которые характеризуются неопределенностью

типа: "большое количество", "большое значение", "значительная величина", "высокий уровень доходов и цен", "высокая норма прибыли", "высокое качество услуг", "высокий сервис обслуживания" и многих других. Общим для всех таких ситуаций является высокая степень проявления того или иного качественного или количественного признака. Особенность нечеткого моделирования при этом заключается в представлении соответствующих нечетких множеств с помощью неубывающих (монотонно возрастающих) функций принадлежности.

В качестве частных случаев Z- и S-образных кривых удобно рассматривать так называемую линейную Z-образную функцию (рисунок 4.5, а) и линейную S-образную функцию (рисунок 4.5, б). Первая из этих функций в общем случае может быть задана аналитически следующим выражением:

$$f_{\downarrow}(x;a,b)=\begin{cases}1, & x\le a\\ \dfrac{b-x}{b-a}, & a<x<b\\ 0, & b\le x\end{cases},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b. График этой функции для некоторого нечеткого множества и универсума X=[0,10] изображен на рисунке 4.5, а, при этом значения параметров соответственно равны a=3, b=6.

Вторая из этих функций в общем случае может быть задана аналитически следующим выражением:

$$f_{\uparrow}(x;a,b)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a<x<b\\ 1, & b\ge x\end{cases},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b. График этой функции для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.5, б, при этом значения параметров соответственно также равны a=3, b=6.

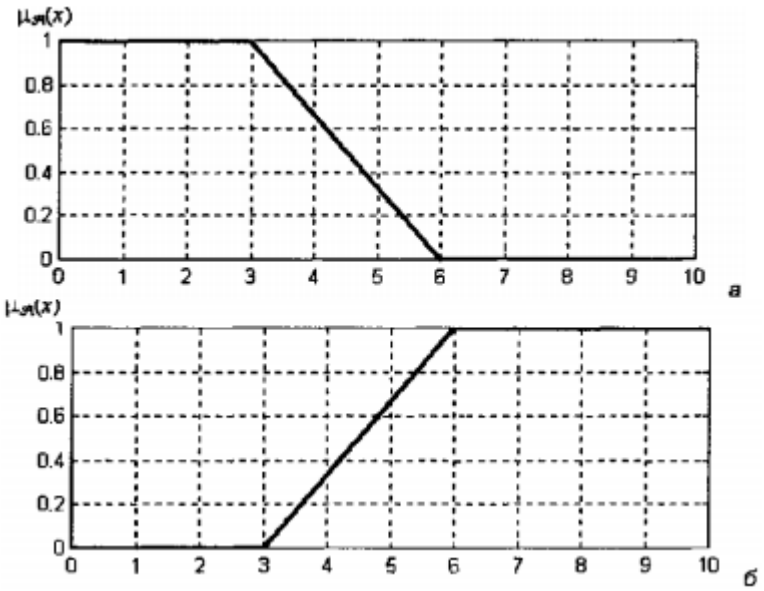


Рисунок 4.5 - Графики линейной Z-образной функции (а) и линейной S-образной функции (б) принадлежности для значений параметров a=3, b=6

Данные функции принадлежности порождают нормальные выпуклые нечеткие множества с границами (a, b).

Следует заметить, что данные линейные Z- и S-образные функции могут быть использованы для построения рассмотренных выше треугольной и трапециевидной функций принадлежности. В частности треугольная функция принадлежности получается как композиция линейной Z-образной и линейной S-образной функций по следующей формуле:

$$f_{\Delta}(x; a, b, c) = \min_{x \in X}\{f_{\uparrow}(x; a, b), f_{\downarrow}(x; b, c)\},$$

где a, b, c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a b c. В выражении используется операция взятия минимума (обозначенная знаком min) из всех значений, указанных в фигурных скобках через запятую. При этом если соответствующие функциональные значения зависят от некоторой независимой переменной (в нашем случае от x), то под знаком минимума явно указывается диапазон или множество значений этой переменной (в нашем случае — универсум).

Трапециевидная функция принадлежности получается как композиция двух линейных Z-образной и S-образной функций по следующей формуле:

$$f_{T}(x; a, b, c, d) = \min_{x \in X}\{f_{\uparrow}(x; a, b), f_{\downarrow}(x; c, d)\},$$

где a, b, c, d — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a b c d.

4.3. П-образные функции принадлежности

К данному типу функций принадлежности можно отнести целый класс кривых, которые по своей форме напоминают колокол, сглаженную трапецию или букву "П".

Первая из подобных функций так и называется — П-образная функция, и в общем случае задается аналитически следующим выражением:

$$f_{\Pi}(x; a, b, c, d) = f_S(x; a, b) \cdot f_Z(x; c, d),$$

где a, b, c, d — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a b c d, а знак " " обозначает обычное арифметическое произведение значений соответствующих функций.

При этом могут быть использованы любые из рассмотренных выше Z- и S-образных функций. В частности, если использовать функции fSl и fZ1 то получим П-функцию fП1, график которой для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.6, а. При этом значения параметров для функции fS1 равны a=1, b=4, а для функции fZ1 — c=5, d=9. Если же использовать функции fS2 и fZ2, то получим П-функцию fП2, график которой для некоторого нечеткого множества и универсума X=[0, 10] изображен на рисунке 4.6, б для тех же значений параметров.

Очевидно, этот тип функций принадлежности порождает нормальные выпуклые нечеткие множества с носителем (a, d) и ядром [b, c].

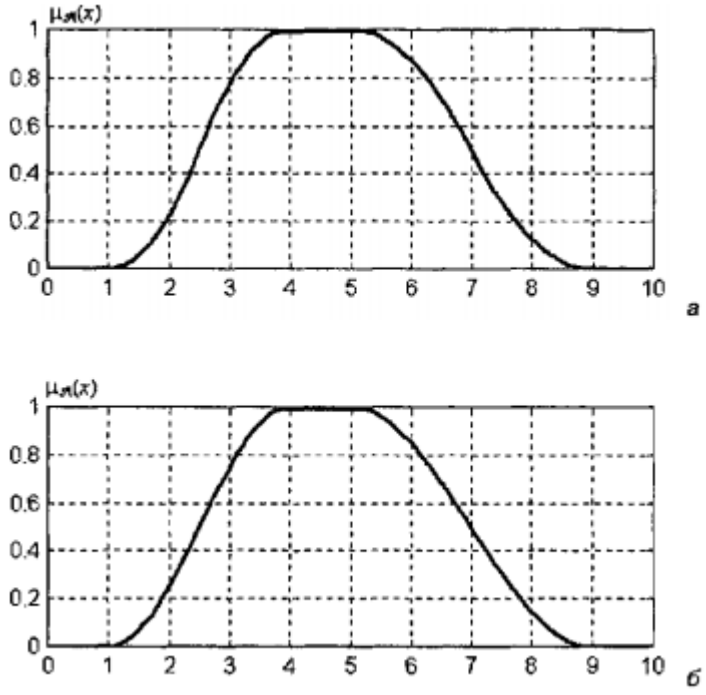


Рисунок 4.6 - Графики П-образных функций принадлежности fП1 (а) и fП2 (б) для значений параметров a=1, b=4, c=5, d=9

Следующая функция этого типа П-образных функций определяется как произведение двух сигмоидальных функций и в общем случае может быть

задана аналитически следующим выражением (рисунок 4.7):

$$f_{П3}(x; a, b, c, d) = f_{S3}(x; a, b) \cdot f_{S3}(x; c, d),$$

$$f_{П3}(x; a, b, c, d) = f_{S3}(x; a, b) \cdot f_{S3}(x; c, d),$$

$$f_{П4}(x;, a, b, c, d) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}},$$

где a, b, c— некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b<c, причем параметр b>0. Здесь функция |x| обозначает модуль действительного числа

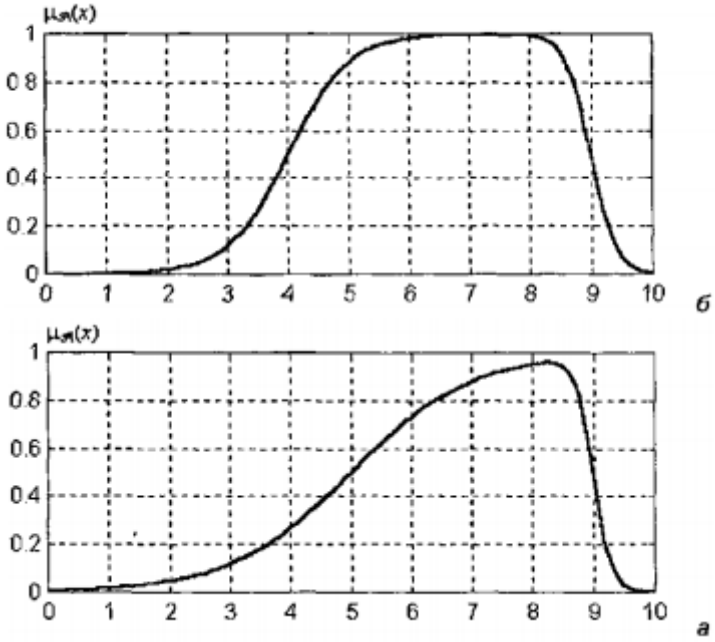


Рисунок 4.7 - Графики П-образных функций принадлежности fП3 для значений параметров a=1, b=5, c=-7, d=9 (а) и для значений параметров a=2, b=4, c=-5, c/=9 (б)

Наконец, последней из рассматриваемых функций данного типа является хорошо известная в теории вероятностей функция плотности нормального распределения в предположении, что, и которая в нашем случае задается аналитически следующим выражением (рисунок 4.8):

$$f_{П5}(x; \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}}.$$

Здесь и c — числовые параметры, при этом квадрат первого из них в теории вероятностей называется дисперсией распределения, а второй параметр c — математическим ожиданием.

Очевидно, эти последние типы функций принадлежности порождают нормальные выпуклые нечеткие множества.

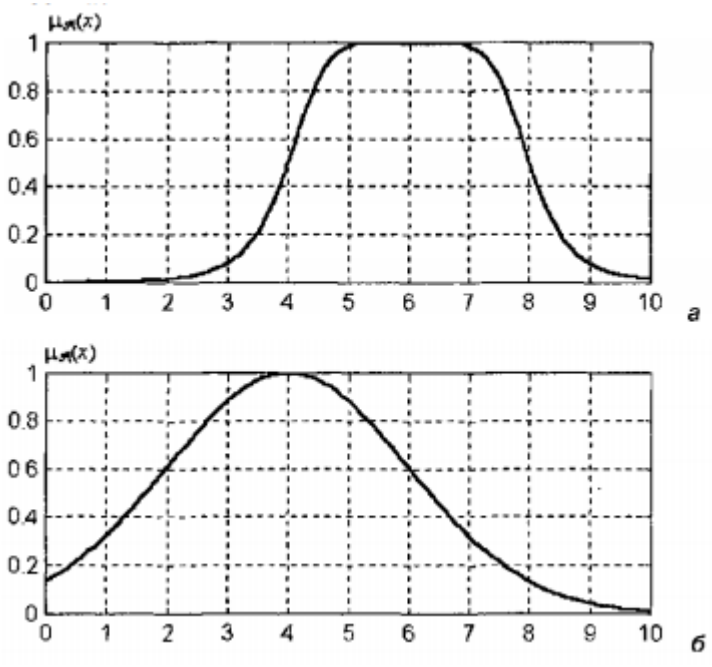


Рисунок 4.8 -. Графики П-образных функций принадлежности fП4 для значений параметров a=2, b=3, c-6 (а) и fП5 для значений параметров c=2, c=4 (б)

4.4. Встроенные функции принадлежности в Fuzzy Logic Toolbox

Fuzzy Logic Toolbox включает 11 встроенных функций принадлежностей, которые используют следующие основные функции:

- кусочно-линейную;
- гауссовское распределение;
- сигмоидную кривую;
- квадратическую и кубические кривые.

Для удобства имена всех встроенных функций принадлежности оканчиваются на **mf**. Вызов функции принадлежности осуществляется следующим образом:

namemf(x, params),

где namemf – наименование функции принадлежности;

x – вектор, для координат которого необходимо рассчитать значения функции принадлежности;

params – вектор параметров функции принадлежности.

Простейшие функции принадлежности треугольная (trimf) и трапециевидная (trapmf) формируется с использованием кусочно-линейной аппроксимации. Трапециевидная функция принадлежности является обобщение треугольной, она позволяет задавать ядро нечеткого множества в виде интервала.

Две функции принадлежности – симметричная гауссовская (gaussmf) и двухстороняя гауссовская (gaussmf) формируется с использованием

гауссовского распределения. Функция gaussmf позволяет задавать ассиметричные функция принадлежности. Обобщенная колоколообразная функция принадлежности (gbellmf) по своей форме похожа на гауссовские. Эти функции принадлежности часто используются в нечетких системах, так как на всей области определения они является гладкими и принимают ненулевые значения.

Функции принадлежности sigmf, dsigmf, psigmf основаны на использовании сигмоидной кривой. Эти функции позволяют формировать функции принадлежности, значения которых начиная с некоторого значения аргумента и до + (-) ∞ равны 1. Такие функции удобны для задания лингвистических термов типа “высокий” или “низкий”.

Полиномиальная аппроксимация применяется при формировании функций zmf, pimf и smf, графические изображения которых похожи на функции sigmf, dsigmf, psigmf, соответственно.

Основная информация о встроенных функциях принадлежности сведена в таблице 4.1.

Таблица 4.1 - Функции принадлежности

Наименование функции	**Описание**	**Аналитическая формула**	**Порядок параметров**
dsigmf	функция принадлежности в виде разности между двумя сигмоидными функциями	$\mu(x) = \frac{1}{1+e^{-a_1(x-c_1)}} - \frac{1}{1+e^{-a_2(x-c_2)}}$	[a1 c1 a2 c2]
gauss2mf	двухсторонняя гауссовская функция принадлежности	если c1<c2, то $\mu(x) = \begin{cases} \exp\left((x-c_1)^2/(-2a_1^2)\right), & x < c_1 \\ 1, & c_1 \le x \le c_2 \\ \exp\left((x-c_2)^2/(-2a_2^2)\right), & x > c_2 \end{cases}$; если c1>c2, то $\mu(x) = \begin{cases} \exp\left((x-c_1)^2/(-2a_1^2)\right), \\ \exp\left((x-c_1)^2/(-2a_1^2)\right) * \exp\left((x-c_2)^2/(-2a_2^2)\right) \\ \exp\left((x-c_2)^2/(-2a_2^2)\right), \end{cases}$.	[a1 c1 a2 c2]

gaussmf	симметричная гауссовская функция принадлежности	$\mu(x) = e^{-\frac{(x-b)^2}{2c^2}}$	[c b]
gbellmf	обобщенная колокообразная функция принадлежности	$\mu(x) = \frac{1}{1 + \left\|\frac{x-c}{a}\right\|^{2b}}$	[a b c]
pimf	пи-подобная функция принадлежности	произведение smf и zmf функций	[a b c d] [a d] – носитель нечеткого множества; [b c] – ядро нечеткого множества;
psigmf	произведение двух сигмоидных функций принадлежности	$\mu(x) = \frac{1}{1 + e^{-a_1(x-c_1)}} \cdot \frac{1}{1 + e^{-a_2(x-c_2)}}$	[a1 c1 a2 c2]
sigmf	сигмоидная функция принадлежности	$\mu(x) = \frac{1}{1 + e^{-a(x-c)}}$	[a c]
smf	s-подобная функция принадлежности	$\mu(x) = \begin{cases} 0, & x \le a \\ \text{нелинейная аппроксимация}, & a < x < b \\ 1, & x \ge b \end{cases}$	[a, b]

trapmf	трапециевидная функция принадлежности	$\mu(x) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases}$	[a, b, c, d]
trimf	треугольная функция принадлежности	$\mu(x) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$	[a, b, c]
zmf	z-подобная функция принадлежности	$\mu(x) = \begin{cases} 1, & x \le a \\ \text{нелинейная аппроксимация}, & a < x < b \\ 0, & x \ge b \end{cases}$	[a, b]

На рисунке 4.9 приведены графические изображения функций принадлежности, полученные с помощью демонстрационной сценария mfdemo. Как видно из рисунка, встроенные функции принадлежности позволяют задавать разнообразные нечеткие множества.

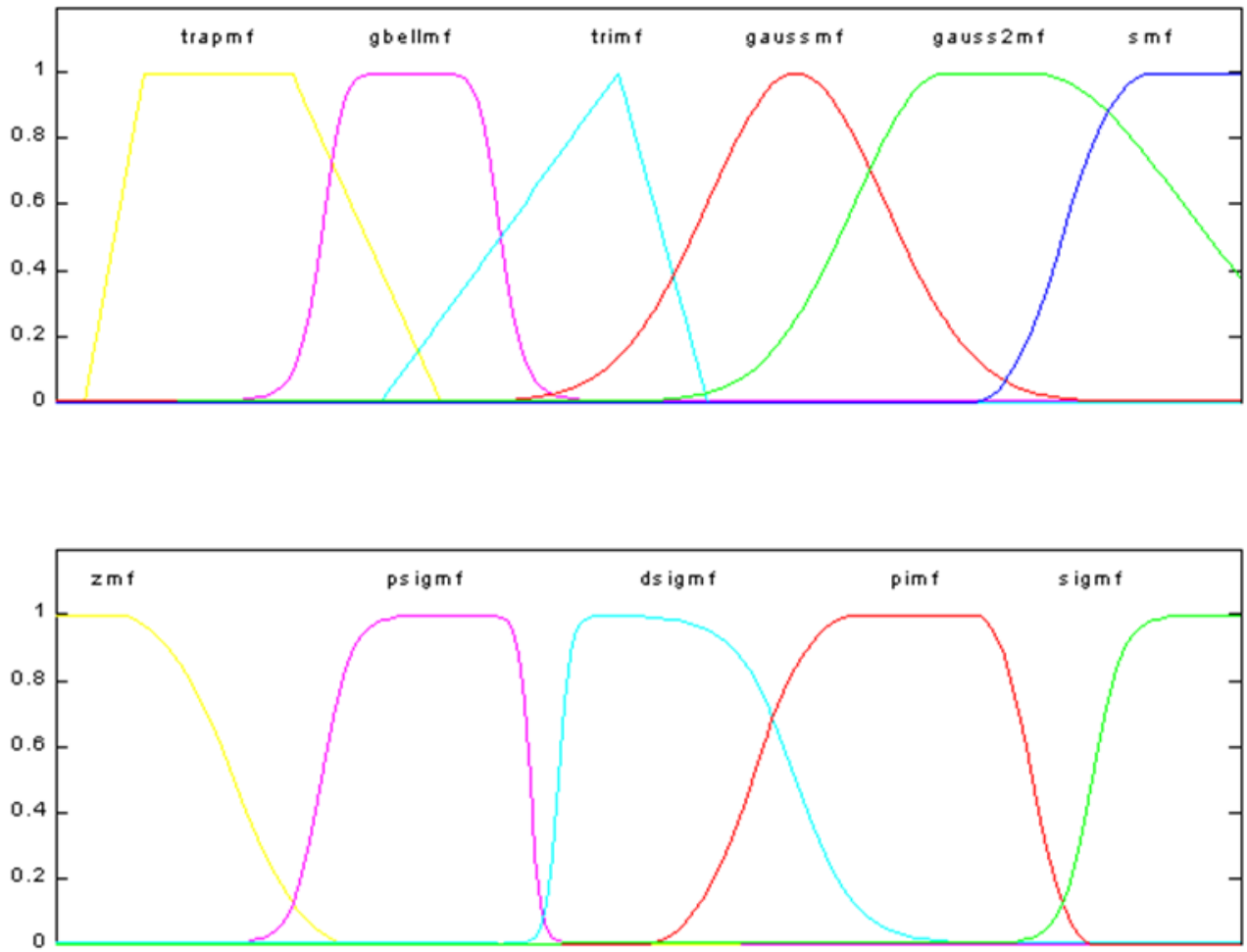


Рисунок 4.9 - Встроенные функции принадлежности

В Fuzzy Logic Toolbox предусмотрена возможность для пользователя создания собственной функции принадлежности. Для этого необходимо создать m-функцию, содержащую два входных аргумента – вектор, для координат которого необходимо рассчитать значения функции принадлежности и вектор параметров функции принадлежности. Выходным аргументом функции должен быть вектор степеней принадлежности. Ниже приведена m-функция, реализующая колоколообразную функцию принадлежности $\mu(x) = \frac{1}{1 + (\frac{x - b}{a})^2}$:

```
function mu=bellmf(x, params)
%bellmf – bell membership function;
%x – input vector;
%params(1) – concentration coefficient (>0);
%params(2) – coordinate of maximuma.
a=params(1);
b=params(2);
mu=1./(1+ ((x-b)/a).^2);
```

Вопросы для повторения и закрепления материала

1. Приведите пример кусочно-линейных функций принадлежности?
2. В чем заключается особенность Z-образных и S-образных функций принадлежности?
3. В чем заключается особенность П-образных функций принадлежности?
4. Назовите встроенные функции принадлежности в Fuzzy Logic Toolbox?
5. Охарактеризуйте типы функций принадлежности с областью их применения?

Задания для самостоятельной работы

1. Ознакомьтесь с функциями принадлежности встроенными в Fuzzy Logic Toolbox.

Тема 5. Принципы и методы построения функции принадлежности

Цель: ознакомиться с принципами и методами построения функции принадлежности.

Задачи:

1. Рассмотреть типы шкал.
2. Изучить прямые методы построения функции принадлежности для одного эксперта.
3. Изучить косвенные методы построения функции принадлежности для одного эксперта.
4. Изучить прямые методы построения функции принадлежности для группы экспертов.
5. Изучить косвенные методы построения функции принадлежности для группы экспертов.
6. Изучить методы построения функции принадлежности терм-множеств.

5.1. Типы шкал

Шкалы наименований или классификации используются для описания принадлежности объектов к определенным классам. Всем объектам одного и того же класса присваивается одно и то же число, объектам разных классов — разные.

Шкала порядка применяется для измерения упорядочения объектов по единичному или совокупности признаков. Числа в шкале порядка отражают только порядок следования объектов и не дают возможности сказать, на сколько или во сколько один объект предпочтительнее другого.

Допустимыми преобразованиями для данного типа шкалы являются все монотонные преобразования, т.е. такие, которые не нарушают порядок следования значений измеряемых величин. Такие шкалы появляются, например, в результате сравнения тел по твердости. Записи "1; 2; 3" и "5, 3; 12,5; 109,2" содержат одинаковую информацию о том, что первое тело самое твердое, второе менее твердое, а третье — самое мягкое. И никакой информации о том, во сколько раз одно тверже другого, на сколько единиц оно тверже и т.д., в этих записях нет, и полагаться на конкретные значения чисел, на их отношения или разности нельзя.

Разновидностью шкалы порядка является шкала рангов, где используются только числа, идущие подряд от 1 вверх по возрастанию. Если среди **m** измеряемых объектов одинаковых нет, то ранговое место каждого объекта в протоколе будет указано одним из целых чисел от 1 до **m**. При одинаковом значении измеряемого свойства у **k** объектов, занимающих порядковые места с **t**-го по **(t+k)**-е, их ранги будут обозначены одинаковым числом, равным их "среднему" рангу **x**, где $x = (1:k)S(i+t-1)$, **i=1-k**.

Такая разновидность шкалы порядка называется "нормированной шкалой рангов".

К типу шкал порядка относится и широко используемая шкала баллов. При этом используются целые числа в ограниченном диапазоне их значений: от 1 до 5 в системе образования, от 0 до 6 или до 10 в спорте и т.д. В любом из этих случаев протокол содержит информацию только о трех эмпирических отношениях: "<", ">" и "=".

Шкала интервалов применяется для отображения величины различия между свойствами объектов (измерение температуры по Фаренгейту и Цельсию). Шкала может иметь произвольные масштаб и точки отсчета.

Здесь между протоколами **y** и **x** допустимы линейные преобразования: **y=ax+b**, где **a** — любое положительное число, а **b** может быть как положительным, так и отрицательным. Это значит, что в разных протоколах может использоваться разный масштаб единиц (a) и разные начала отсчета (b). Примером шкал этого типа могут быть шкалы для измерения температуры. Если в протоколе указаны градусы, но не говорится, в какой шкале (Цельсия, Кельвина и т.д.), то во избежание недоразумений при описании закономерностей можно использовать только отношения интервалов, так как при любых значениях **a** и **b** сохраняется равенство $(y_1-y_2):(y_3-y_4)=[(ax_1+b)-(ax_2+b)]:[(ax_3+b)-(ax_4+b)]$.

Если записи в протоколе сопровождаются информацией о том, какие именно градусы имеются в виду (например, "18^0C"), то мы имеем дело с протоколом в абсолютной шкале.

Шкала отношений используется, например, для измерения массы, длины, веса. В этой шкале числа отражают отношения свойств объектов, т.е. во сколько раз свойство одного объекта превосходит свойство другого.

Между разными протоколами, фиксирующими один и тот же эмпирический факт на разных языках, при этом типе шкалы должно выполняться соотношение: **y=ax,** где **a** — любое положительное число. Один и тот же эмпирический смысл имеют протоколы "16 кг", "16000 г", "0,016 т" и т.д. От любой записи можно перейти к любой другой, подобрав соответствующий множитель "**a**". Этот тип шкалы удобен для измерения весов, длин и т.д. Если нам неизвестно, в каких именно единицах записаны веса тел в разных протоколах, то мы можем полагаться только на отношение весов двух тел: например, тело с весом 10 единиц в два раза тяжелее тела с весом 5 единиц вне зависимости от того, что было взято за единицу — тонна или грамм. Инвариантность отношений отражена в названии шкалы данного типа. Если же в протоколе указана единица веса, то такой протокол отражает свойства тел в абсолютной шкале.

Шкала разностей используется для измерения свойств объектов при необходимости указания, на сколько один объект превосходит другой по одному или нескольким признакам. Является частным случаем шкалы интервалов при выборе единицы масштаба.

Абсолютная шкала — частный случай шкалы интервалов. В ней обозначается нулевая точка отсчета и единичный масштаб. Применяется для измерения количества объектов.

Допустимое преобразование для шкал данного типа представляет собой тождество, т.е. если на одном языке в протоколе записано "**y** ", а на другом языке "**x**", то между ними должно выполняться простое соотношение: **y=x.** Этот тип шкалы удобен для записи количества элементов в некотором конечном множестве. Если, пересчитав количество яблок, один эксперт запишет в протоколе "6", а другой — "VI", то нам достаточно знать, что "6" и "VI" означают одно и то же, т.е., что между этими записями существует тождественное отношение: 6=VI.

5.2. Прямые методы для одного эксперта

Прямые методы для одного эксперта состоят в непосредственном задании функции, позволяющей вычислять значения.

Например, пусть переменная "ВОЗРАСТ" принимает значения из интервала U=[0,100]. Слово "МОЛОДОЙ" можно интерпретировать как имя нечеткого подмножества U, которое характеризуется функцией совместимости. Таким образом, степень, с которой численное значение возраста, скажем u=28, совместимо с понятием "МОЛОДОЙ", есть 0,7, в то время как совместимость u=30 и u=35 с тем же понятием есть 0,5 и 0,2 соответственно.

Метод семантических дифференциалов предложенный Осгудом. Практически в любой области можно получить множество шкал оценок, используя следующую процедуру:

1. определить список свойств, по которым оценивается понятие (объект);
2. найти в этом списке полярные свойства и сформировать полярную шкалу;
3. для каждой пары полюсов оценить, в какой степени введенное понятие обладает положительным свойством.

Совокупность оценок по шкалам была названа **профилем понятия**. Следовательно, вектор с координатами, изменяющимися от 0 до 1, также называется профилем. Профиль есть нечеткое подмножество положительного списка свойств или шкал.

Пример. В задаче распознавания лиц можно выделить следующие шкалы:

$\mathbf{X_1}$	Высота лба	Низкий-широкий
$\mathbf{X_2}$	Профиль носа	Горбатый-курносый
$\mathbf{X_3}$	Длина носа	Короткий-длинный
$\mathbf{X_4}$	Разрез глаз	Узкие-широкие
$\mathbf{X_5}$	Цвет глаз	Темные-светлые
$\mathbf{X_6}$	Форма подбородка	Остроконечный-квадратный
$\mathbf{X_7}$	Толщина губ	Тонкие-толстые
$\mathbf{X_8}$	Цвет лица	Смуглое-светлое
$\mathbf{X_9}$	Очертание лица	Овальное-квадратное

Светлое квадратное лицо, у которого чрезвычайно широкий лоб, курносый длинный нос, широкие светлые глаза, остроконечный подбородок, может быть определено как нечеткое множество $\{\langle x_1, 1\rangle,\ \langle x_2, 1\rangle, \ldots, \langle x_9, 1\rangle\}$.

Способ вычисления частичной принадлежности друг другу строгих множеств. Пусть покрытием **K** обычного множества **U** является любая совокупность обычных подмножеств $\{A_1, \ldots, A_k\}$ множества **U** таких, что $A_i \neq \varnothing, \quad A_1 \cup \ldots \cup A_k = U$. В крайнем случае, когда для любых $i, j \quad (i \neq j)$, $A_i \cap A_j = \varnothing$, имеет место разбиение **U**. Предположим, что имеется $B \subseteq U$, тогда **B** может рассматриваться как нечеткое подмножество **K** с функцией принадлежности

$$\mu_B(A_i) = \frac{|A_i \cap B|}{|A_i \cup B|},$$

где $|A|$ — мощность множества **A**.

Пример. Пусть $U = \{1, 2, \ldots, 9\}$, $K = \{\{1, 2, 3, 5\}, \{3, 6, 9\}, \{2, 4, 8\}, \{1, 3, 7\}, \{2, 3, 8\}\} = \{A_1, A_2, A_3, A_4, A_5\}$, $B = \{2, 3, 5, 8, 9\}$. Тогда, рассматривая **B** как нечеткое подмножество **K**, можно написать

$$B = \left\{ \langle A_1, 1/3 \rangle, \ \langle A_2, 1/3 \rangle, \ \langle A_3, 1/3 \rangle, \ \langle A_4, 1/7 \rangle, \ \langle A_5, 3/5 \rangle \right\}.$$

Любое решение задачи многоцелевой оптимизации можно рассматривать как нечеткое подмножество значений целевой функции следующим образом. Пусть $f_1, \ldots, f_k$ — целевые функции, где $f_i : R^n \to R$, и пусть требуется решить задачу $f_i \to \max$ для всех **i**. Пусть $f_i^* < \infty$ — максимальное значение функции f_i и $C = \{f_1, \ldots, f_k\}$ — множество целевых функций, тогда любое значение **x** в области определения f_i можно рассматривать как нечеткое множество на **C** с вектором значений принадлежности

$$\mu_x = \langle \mu_1, \ldots, \mu_k \rangle, \quad \text{где } \mu_i = \frac{f_i^* - f_i(x)}{f_i^*}.$$

5.3. Косвенные методы для одного эксперта

В обыденной жизни мы часто сталкиваемся со случаями, когда не существует элементарных измеримых свойств и признаков, которые определяют интересующие нас понятия, **например,** красоту, интеллектуальность. Бывает трудно проранжировать степень проявления свойства у рассматриваемых элементов. Так как степени принадлежности рассматриваются на данном реальном множестве, а не в абсолютном смысле, то интенсивность принадлежности можно определять, исходя из попарных сравнений рассматриваемых элементов.

Среди косвенных методов определения функции принадлежности наибольшее распространение получил **метод парных сравнений Саати**. Сложность использования этого метода заключается в необходимости нахождения собственного вектора матрицы парных сравнений, которая задается с помощью специально предложенной шкалы. Причем эти сложности увеличиваются с ростом размерности универсального множества, на которой задается лингвистический терм.

Метод парных сравнений, не требующий нахождения собственного вектора матрицы, т.е. освобождает исследователя от трудоемких процедур решения характеристических уравнений.

Пусть **A** — некоторое свойство, которое рассматривается как лингвистический терм. Нечеткое множество, с помощью которого формализуется терм **A**, представляет собой совокупность пар:

$$A = \{< u_1, \mu_A(u_1) >, < u_2, \mu_A(u_2) > , \ldots , < u_n, \mu_A(u_n) >\},$$

где $U = \{u_1, u_2, \ldots, u_n\}$ — универсальное множество, на котором задается нечеткое множество **А**. Задача состоит в том, чтобы определить значения $\mu_A(u_i)$ для всех $i = 1, \ldots, n$. Совокупность этих значений и будет составлять неизвестную функцию принадлежности.

Метод, который предлагается для решения поставленной проблемы, базируется на идее распределения степеней принадлежности элементов универсального множества согласно с их рангами.

В нашем случае под рангом элемента $u_i \in U$ будем понимать число $r_A(u_i)$, которое характеризует значимость этого элемента в формировании свойства, описываемого нечетким термом. Допускаем, что выполняется правило: **чем больший ранг элемента, тем больше степень принадлежности.**

Для последующих построений введем такие обозначения: $r_A(u_i) = r_i$, $\mu_A(u_i) = \mu_i$. Тогда правило распределения степеней принадлежности можно задать в виде системы соотношений:

$$\begin{cases} \frac{\mu_1}{r_1} = \frac{\mu_2}{r_2} = \ldots = \frac{\mu_n}{r_n}, \\ \mu_1 + \mu_2 + \ldots + \mu_n = 1. \end{cases}$$

Используя данные соотношения, легко определить степени принадлежности всех элементов универсального множества через степень принадлежности опорного элемента.

Если опорным является элемент $u_i \in U$ с принадлежностью μ_i, то

$$\mu_j = \frac{r_j}{r_i}\mu_i, \quad \text{для всех } j \neq i.$$

Учитывая условие нормирования, находим:

$$\mu_1 = \left(1 + \frac{r_2}{r_1} + \frac{r_3}{r_1} + \ldots + \frac{r_n}{r_1}\right)^{-1},$$

$$\mu_2 = \left(\frac{r_1}{r_2} + 1 + \frac{r_3}{r_2} + \ldots + \frac{r_n}{r_2}\right)^{-1},$$

$$\ldots\ldots\ldots$$

$$\mu_n = \left(\frac{r_1}{r_n} + \frac{r_2}{r_n} + \frac{r_3}{r_n} + \ldots + 1\right)^{-1}.$$

Полученные формулы дают возможность вычислять степени принадлежности элементов $u_i \in U$ к нечеткому терму A двумя независимыми путями:

1. по абсолютным оценкам уровней r_i, которые определяются согласно методикам, предложенным в теории структурного анализа систем;

2. по относительным оценкам рангов $\frac{r_i}{r_j} = s_{ij}$, которые образуют матрицу $S = (s_{ij})$.

Эта матрица обладает следующими свойствами:

а) она диагональная, т.е. $s_{ii} = 1, \quad i = 1, \ldots, n;$

б) ее элементы, которые симметричны относительно главной диагонали, связаны зависимостью $s_{ij} = \frac{1}{s_{ji}};$

в) она транзитивна, т.е. $s_{ik} \cdot s_{kj} = s_{ij}$.

Наличие этих свойств приводит к тому, что при известных элементах одной строки матрицы **S** легко определить элементы всех других строк. Если известна **r**-я строка, т.е. элементы s_{rj}, $j = 1, \ldots, n$, то произвольный элемент s_{ij} находится так:

$$s_{ij} = \frac{s_{kj}}{s_{ki}}.$$

Поскольку матрица **S** может быть интерпретирована как матрица парных сравнений рангов, то для экспертных оценок элементов этой матрицы можно использовать 9 балльную шкалу Саати. В нашем случае шкала формируется так:

Числовая оценка (s_{ij})	**Качественная оценка (сравнение** r_i **и** r_j **)**
1	отсутствие преимущества r_i над r_j
3	слабое преимущество r_i над r_j
5	существенное преимущество r_i над r_j
7	явное преимущество r_i над r_j
9	абсолютное преимущество r_i над r_j
2, 4, 6, 8	промежуточные сравнительные оценки

Таким образом, с помощью полученных формул экспертные знания о рангах элементов или их парные сравнения преобразуются в функцию принадлежности нечеткого терма.

Шкала предлагает общий метод варьирования прототипов получения численного значения функции принадлежности. Пусть имеется прототип (или идеальный объект) **P**, описание которого можно деформировать изменением параметров $p_1, p_2, \ldots, p_n$. Если дан некоторый объект **A**, то, варьируя параметры, можно добиться наибольшего соответствия прототипа и объекта. Вводится мера сходства между объектом **A** и прототипом **P**: $\rho(A, p_1, p_2, \ldots, p_n)$.

Для более точного измерения сходства объекта с разными прототипами вводится штрафная функция **d**. Далее строится функция:

$$\mathrm{sim}(A) = \min_{p_1 \cdots p_n} \left\{ \rho(A, p_1, \ldots, p_n) + d(p_1, \ldots, p_n) \right\}.$$

Так как прототип полностью соответствует самому себе, то $\mathrm{sim}(P) = 0$. Численные значения функции принадлежности вычисляются по формуле

$$\mu_P(A) = 1 - \frac{\mathrm{sim}(A)}{\max\limits_A \mathrm{sim}(A)}.$$

5.4. Прямые методы для группы экспертов

При интерпретации степени принадлежности как вероятности было предложено получать функции принадлежности для нескольких классов понятий S_j расчетным путем, используя равенство $\mu_{S_j}(u_i) = p(S_j|u_i)$, где условная вероятность определяется по формуле Байеса:

$$p(S_j|u_i) = \frac{p_{u_i}(S_j)\; p(u_i|S_j)}{\sum\limits_{j=1}^{m} p_{u_i}(S_j)\; p(u_i|S_j)},$$

причем

$$p_{u_i}(S_j) = \frac{(y_j)_{u=u_i}}{n}, \qquad j = 1, \ldots, m, \quad i = 1, \ldots, n,$$

y_j — число случаев при значении параметра u_i, когда верной оказалась j -я гипотеза.

Я.Я. Осис предложил следующую методику оценки функции принадлежности. Первоначально определяется то максимальное количество классов, которое может быть описано данным набором параметров. Для каждого элемента **u** значение функции принадлежности класса S_1 дополняет до единицы значения функции принадлежности класса S_2 (в случае двух классов). Таким образом, система должна состоять из классов, представляющих противоположные события. Сумма значений функции принадлежности произвольного элемента u к системе таких классов будет равна единице. Если число классов и их состав четко не определены, то необходимо вводить условный класс, включающий те классы, которые не выявлены. Далее эксперты оценивают в процентах при данном состоянии u степень проявления каждого класса из названного перечня.

Однако в некоторых случаях мнение эксперта очень трудно выразить в процентах, поэтому более приемлемым способом оценки функции принадлежности будет метод опроса, который состоит в следующем. Оцениваемое состояние предъявляется большому числу экспертов, и каждый имеет один голос. Он должен однозначно отдать предпочтение одному из

классов заранее известного перечня. Значение функции принадлежности вычисляется по формуле $\mu_S(u) = n_S/n$, где n — число экспертов, участвовавших в эксперименте, и n_S — число экспертов, проголосовавших за класс S.

Пример. Пусть в результате переписи населения в некоторой области с численностью жителей **p** получено множество значений возраста $U = [0, 100]$. Пусть $y(u)$ — число людей, имеющих возраст **u** и утверждающих, что являются молодыми. Пусть $n(u)$ — действительное число людей, имеющих возраст **u**; тогда $p = \int_0^{100} dn(u)$. Можно считать, что понятие "МОЛОДОЙ" описывается нечетким множеством на **U** с функцией принадлежности $\mu(u) = y(u)/n(y)$. Очевидно, что для малых значений возраста $y(u) = n(u)$, следовательно, $\mu(u) = 1$. Однако, не все $n(35)$ считают себя молодыми, следовательно, $y(35) < n(35)$. Для $u > 80$ число $y(u)$ должно быть очень маленьким.

5.5. Косвенные методы для группы экспертов

Опишем кратко **косвенный метод**, предложенный З.А. Киквидзе. Пусть U — универсальное множество, S — понятие, общее название элементов. Задача определения нечеткого подмножества U, описывающего понятие S, решается путем опроса экспертов. Каждый эксперт A_i, $i = \overline{1,m}$ выделяет из U множество элементов Q_i, по его мнению, соответствующих понятию S. Ранжируя все элементы множества $Q = \bigcup_{i=1}^{m} Q_i$ по предпочтению в смысле соответствия понятию S, каждый эксперт упорядочивает Q, используя отношение порядка «>» или $\approx$. Отношение $\approx$ указывает на одинаковую степень предпочтения между любыми элементами $q_\alpha, q_\beta \in Q$. Предполагается, что эксперты могут поставить коэффициенты степени предпочтения γ перед элементами в упорядоченной последовательности, усиливая или ослабляя отношение предпочтения. Вводится расстояние между элементами указанной последовательности $q^i{}_\alpha, q^i{}_\beta \in Q$:

$$\rho(q^i_\alpha, q^i_\beta) = \frac{1}{\gamma}.$$

Здесь α, β — порядковые номера элементов в упорядочении. Расстояние вычисляется через первый в упорядочении элемент:

$$\rho(q^i_\alpha, q^i_\beta) = \rho(q^i_1, q^i_\beta) - \rho(q^i_1, q^i_\alpha) = \rho^i_\beta - \rho^i_\alpha.$$

Эта разность показывает, насколько предпочтительнее q_α^i по сравнению с q_β^i. При решении задачи взвешивания предпочтительности элементов множества Q предполагается, что разность между весами $\varphi(q_\alpha^i) - \varphi(q_\beta^i)$ пропорциональна разности $\rho_\beta^i - \rho_\alpha^i$: $\varphi(q_{\beta+\nu}^i) - \varphi(q_\beta^i) = c(\rho_{\beta+\nu}^i - \rho_\beta^i)$. Когда $\nu = 1$, формула превращается в рекуррентную формулу, и задача сводится к определению веса первого элемента. При использовании рекуррентных формул вес последнего элемента должен отличаться от нуля. Например, в качестве $\varphi(q_1^i)$ можно выбрать $\max_\alpha \rho_\alpha^i + \rho_0$. На основании всех $\varphi(q_\alpha^i) \quad (i = 1, \ldots, m)$ для q_α определяется значение $\varphi(q_\alpha) = \frac{1}{m}\sum_{i=1}^{m} \varphi(q_\alpha^i)$; это и есть степень принадлежности элемента $u \in U$ некоторому нечеткому множеству с общим названием S.

Зиммерман предлагает метод, сочетающий преимущества косвенных методов в их простоте и стойкости к искажениям ответов экспертов и преимущества прямых методов, позволяющих получить непосредственно значения степени принадлежности. Выборку объектов необходимо проводить так, чтобы достаточно равномерно представить степень принадлежности от 0 до 1 по отношению к рассматриваемому нечеткому множеству. Эта выборка должна удовлетворять условию безоговорочного экстремума, т.е. должна содержать, по крайней мере, два объекта, значения функции принадлежности на которых имеют определенность 0 и 1 (все эксперты приписывают эти числа экстремумам). Далее, когда множество подходящих объектов отобрано, эксперты опрашиваются о степенях принадлежности в процентной шкале. Оценка позиции по шкале каждого объекта определяется посредством медианы из распределений значений принадлежности. В качестве процедуры шкалирования используется метод, основанный на законе Терстона об измерении категорий. Процедура, требующая отсортировки *n* объектов в *(k+1)* категории на некотором континууме свойств *N* экспертами, дает распределение частоты для каждого объекта по категориям. Средние значения границ категорий, полученные методом наименьших квадратов, позволяют определить значения оценок объектов на шкале.

5.6. Методы построения терм-множеств

Считается, что для практических задач достаточно наличия нечеткого языка с фиксированным конечным словарем — ограничение не слишком сильное с точки зрения практического использования. Лингвистическая

переменная **L**, используемая при формализации задач принятия решения, на практике, как правило, имеет базовое **терм-множество** $T = \{T_i\}$, состоящее из 2—10 термов. Каждый терм описывается нечетким подмножеством множества значений **U** некоторой базовой переменной **u** и рассматривается как лингвистическое значение **L**. Предполагается, что объединение всех этих элементов терм-множества покрывает полностью **U**. Это гарантирует, что любой элемент $u \in U$ описывается некоторым $T_i \in T$.

Существует способ построения частотных оценок **S** = {"редко", "часто", "иногда",...}, который основан на предположении о том, что слово s_i употребляется человеком не для обозначения зарегистрированной частоты появления факта, а для обозначения относительного числа событий в прошлой деятельности человека, когда рассматривалась такая же частота. Каждому s_i ставится в соответствие нечеткое подмножество интервала [0,1]. Функции принадлежности μ_{S_i} получаются на основании психологического эксперимента следующим образом: группе испытуемых предъявляется набор стимулов (оценок частоты) и шкала из k категорий, упорядоченных по степени интенсивности частоты от наименьшей (1) до наибольшей (k); испытуемым предлагается разбить стимулы на k классов согласно интенсивности частоты, независимо оценивая каждый стимул и помещая в любую категорию любое число стимулов. Каждому числу u_j из [0,1], $u_j = (j-1)/(k-1)$, ставятся в соответствие степени употребления группой испытуемых слова s_i для обозначения категории. Значения функции принадлежности определяются в результате нормирования: $\mu_{S_i}(u) : [0,1] \to [0,1]$.

Предложенная методика оправдана следующим: выбор обозначения категории не отражается сколь-нибудь значительно на проведении испытания. Во-первых, число категорий (деление шкалы) не влияет кардинально на результаты эксперимента, в котором производится шкалирование субъективных ощущений. Во-вторых, шкала из k категорий является шкалой равно кажущихся интервалов, поскольку предполагается, что ее деления отстоят на психологическом континууме на равных интервалах.

Естественным шагом при построении функций принадлежности элементов терм-множества лингвистической переменной является построение одновременно всех функций принадлежности этого терм-множества, сгруппированных в так называемое отношение моделирования R. Процесс построения состоит в заполнении таблицы, где, например, для лингвистической переменной "РАССТОЯНИЕ" столбцы индексированы расстояниями в метрах, а строки — элементами терм-множества "ОЧЕНЬ БЛИЗКО", "БЛИЗКО",..., "ДАЛЕКО", "ОЧЕНЬ ДАЛЕКО". На пересечении

соответствующей строки и столбца стоит степень сходства для испытуемого данных понятий в определенной семантической ситуации, например, насколько сходны понятия "БЛИЗКО" и "5 метров" в ситуации перебегания улицы перед быстро идущим транспортом. Расстояние берется от пешехода до машины и в данном случае является синонимом опасности. Вообще говоря, каждую клеточку таблицы можно заполнять отдельно, а потом, переставляя строки и столбцы, постараться сделать строки и столбцы унимодальными. Если это удается, то исходное терм-множество может быть использовано для построения нечеткой шкалы измерений, точками отсчета которой являются сами элементы терм-множества. Перевод в эту шкалу будет осуществляться с помощью минимаксного умножения строки, задающей исходную лингвистическую переменную в шкале метров, на отношение моделирования. Отношение сходства между элементами терм-множества $R \circ R^T$, полученное с помощью умножения матрицы R на транспонированную, задает набор функций принадлежности элементов лингвистической шкалы в самой шкале, а отношение $R^T \circ R$ задает набор функций принадлежности расстояний в метрах в метрической шкале.

Вопросы для повторения и закрепления материала

1. Почему особенно важно определиться с типом шкалы?
2. Где используются шкалы порядка?
3. Где применяются шкалы интервалов?
4. Что такое абсолютная шкала?
5. Назовите несколько методов измерений?
6. Что такое ранжирование?
7. Назовите методы групповой экспертизы?
8. Каким образом классифицируются методы построения функций принадлежности?
9. В чем заключается различие прямых и косвенных методов построения функций принадлежности?
10. В чем заключается суть построения функции принадлежности при прямом методе для одного эксперта?
11. В чем заключается суть построения функции принадлежности при косвенном методе для одного эксперта?
12. Что такое матрица парных сравнений?
13. В чем заключается суть построения функции принадлежности при прямом методе для группы экспертов?
14. В чем заключается суть построения функции принадлежности при косвенном методе для группы экспертов?
15. Каким образом определяются терм-множества?

Задания для самостоятельной работы

1. Подготовьте реферат, содержащий детальное описание одного из методов построения функции принадлежности с последующей формализацией его алгоритма в виде блок-схемы.

Тема 6. Нечеткая логика

Цель: изучить принципы организации работы с нечеткой логикой.

Задачи:

1. Рассмотреть место нечеткой логики в управлении сложными системами.
2. Изучить понятия нечеткой переменной и лингвистической переменной.
3. Рассмотреть лингвистические переменные истинности.
4. Ознакомиться с логическими связками в нечеткой лингвистической логике.
5. Рассмотреть значения истинности «Неизвестно» и «Не определено».

6.1. Нечеткая логика в управлении сложными системами

С точки зрения сложности построения адекватной математической модели все объекты могут быть условно разделены на два класса: «простые» и «сложные».

Простые объекты допускают построение вполне адекватной и сравнительно несложной математической модели, которая соответствует реальным процессам в объекте и пригодна для реализации на вычислительной технике. Сложные объекты, образующие гораздо более обширный класс по сравнению с простыми, обладают рядом отличительных особенностей.

1) Количество факторов, оказывающих влияние на протекающие в объекте процессы, настолько велико, а взаимосвязи между его отдельными элементами являются в такой степени сложными, что создание адекватной математической модели весьма затруднительно, а в ряде случаев вообще невозможно. Если же такую модель все-таки удается построить, она оказывается, как правило, очень громоздкой и неприемлемой для практического применения в силу того, что время реакции системы управления на входное воздействие получается недопустимо большим. С другой стороны, игнорирование отдельных фактов и взаимосвязей в структуре объекта с целью получения более простой математической модели может привести к неоправданной идеализации объекта и потере адекватности.

2) Отсутствует достаточно точная и достоверная информация о характере функционирования объекта и протекающих в нем процессах, либо

количество такой информации оказывается недостаточным для построения точной и адекватной модели при помощи традиционного математического аппарата.

3) Значительная часть информации, являющейся необходимой для математического описания объекта, существует лишь в форме представлений специалистов, имеющих опыт работы с рассматриваемой объектом.

4) В ряде случаев условия, влияющие на выбор стратегии управления, могут быть выражены лишь в качественном виде. Зачастую сама цель управления не может быть представлена в виде неких количественных соотношений, например, в виде какой-либо целевой функции.

Для успешного решения задачи управления сложным объектом при помощи методов нечеткой логики следует строить не модель объекта, а модель управления объектом.

6.2. Понятие нечеткой и лингвистической переменной

Нечеткая переменная определяется кортежем параметров $\langle \alpha, X, A \rangle$ где α - название нечеткой переменной, X - область определения нечеткой переменной, $A = \{\mu_A(x)/x\}$- заданное на X нечеткое множество, описывающее возможные значения нечеткой переменной.

Пример. Нечеткая переменная, описывающая скорость движения объекта: < a = «приблизительно нулевая скорость движения объекта», $X = (-x_{max}; x_{max}), A = \{e^{-x^2}/x\}, x \in X$ >.

Если областью определения нечеткой переменной является множество действительных чисел X=R, то такая нечеткая переменная называется **действительным нечетким числом**.

Лингвистическая переменная отличается от числовой переменной тем, что ее значениями являются не числа, а слова или предложения в естественном или формальном языке. 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 частности, нечеткое множество, которое представляет собой ограничение, связанное со значениями лингвистической переменной, можно рассматривать как совокупную характеристику различных подклассов элементов универсального множества. В этом смысле роль нечетких множеств аналогична той роли, которую играют слова и предложения в естественном языке.

Например, прилагательное "КРАСИВЫЙ" отражает комплекс характеристик внешности индивидуума. Это прилагательное можно также рассматривать как название нечеткого множества, которое является ограничением, обусловленным нечеткой переменной "КРАСИВЫЙ". С этой

точки зрения термины "ОЧЕНЬ КРАСИВЫЙ", "НЕКРАСИВЫЙ", "ЧЕРЕЗВЫЧАЙНО КРАСИВЫЙ", "ВПОЛНЕ КРАСИВЫЙ" и т.п. — названия нечетких множеств, образованных путем действия модификаторов "ОЧЕНЬ, НЕ, ЧЕРЕЗВЫЧАЙНО, ВПОЛНЕ" и т.п. на нечеткое множество "КРАСИВЫЙ". В сущности, эти нечеткие множества вместе с нечетким множеством "КРАСИВЫЙ" играют роль значений лингвистической переменной "ВНЕШНОСТЬ".

Важный аспект понятия лингвистической переменной состоит в том, что эта переменная более высокого порядка, чем нечеткая переменная, в том смысле, что значениями лингвистической переменной являются нечеткие переменные.

Например, значениями лингвистической переменной "ВОЗРАСТ" могут быть: "МОЛОДОЙ, НЕМОЛОДОЙ, СТАРЫЙ, ОЧЕНЬ СТАРЫЙ, НЕ МОЛОДОЙ И НЕ СТАРЫЙ" и т.п. Каждое из этих значений является названием нечеткой переменной. Если $\tilde{x}$ — название нечеткой переменной, то ограничение, обусловленное этим названием, можно интерпретировать как смысл нечеткой переменной $\tilde{x}$.

Другой важный аспект понятия лингвистической переменной состоит в том, что лингвистической переменной присущи два правила:

1. Синтаксическое, которое может быть задано в форме грамматики, порождающей название значений переменной;
2. Семантическое, которое определяет алгоритмическую процедуру для вычисления смысла каждого значения.

Лингвистическая переменная (linguistic variable) - переменная, значениями которой могут быть слова или словосочетания некоторого естественного или искусственного языка.

Терм–множество (term set) - множество всех возможных значений лингвистической переменной.

Терм (term) - любой элемент терм–множества. В теории нечетких множеств терм формализуется нечетким множеством с помощью функции принадлежности.

Лингвистическая переменная характеризуется набором свойств $(X, T(X), U, G, M)$, в котором:

- **X** — название переменной;
- **T(X)** обозначает терм-множество переменной **X**, т.е. множество названий лингвистических значений переменной **X**, причем каждое из таких значений является нечеткой переменной $\tilde{x}$ со значениями из универсального множества **U** с базовой переменной **u**;
- **G** — синтаксическое правило, порождающее названия $\tilde{x}$ значений переменной **X**;

- M — семантическое правило, которое ставит в соответствие каждой нечеткой переменной $\tilde{x}$ ее смысл $M(\tilde{x})$, т.е. нечеткое подмножество $M(\tilde{x})$ универсального множества **U**.

Конкретное название $\tilde{x}$, порожденное синтаксическим правилом **G**, называется термом. Терм, который состоит из одного слова или из нескольких слов, всегда фигурирующих вместе друг с другом, называется **атомарным термом**. Терм, который состоит из более чем одного атомарного терма, называется **составным термом**.

Пример. Рассмотрим лингвистическую переменную с именем **X** = "ТЕМПЕРАТУРА В КОМНАТЕ". Тогда оставшуюся четверку $\langle T, U, G, M \rangle$, можно определить так:

1. универсальное множество U=[5,35];
2. терм-множество T={"ХОЛОДНО", "КОМФОРТНО", "ЖАРКО"} с такими функциями принадлежностями:

$$\mu''_{\text{холодно}''}(u) = \frac{1}{1 + \left(\frac{u-10}{7}\right)^{12}},$$

$$\mu''_{\text{комфортно}'}(u) = \frac{1}{1 + \left(\frac{u-20}{3}\right)^{6}},$$

$$\mu''_{\text{жарко}''}(u) = \frac{1}{1 + \left(\frac{u-30}{6}\right)^{10}};$$

3. синтаксическое правило **G**, порождающее новые термы с использованием квантификаторов "**и**", "**или**", "**не**", "**очень**", "**более-менее**" и других;
4. **M** будет являться процедурой, ставящей каждому новому терму в соответствие нечеткое множество из **X** по правилам: если термы **A** и **B** имели функции принадлежности $\mu_A(u)$ и $\mu_B(u)$ соответственно, то новые термы будут иметь следующие функции принадлежности, заданные в таблице 6.1.

Таблица 6.1 – Функции принадлежности термов

Квантификатор	**Функция принадлежности (** $u \in U$ **)**
не t	$1 - \mu_t(u)$
очень t	$(\mu_t(u))^2$
более-менее t	$\sqrt{\mu_t(u)}$
A и B	$\max(\mu_A(x), \mu_B(x))$
A или B	$\min(\mu_A(x), \mu_B(x))$

Графики функций принадлежности термов "холодно", "не очень холодно" и т.п. к лингвистической переменной "температура в комнате" показаны на рисунке 6.1:

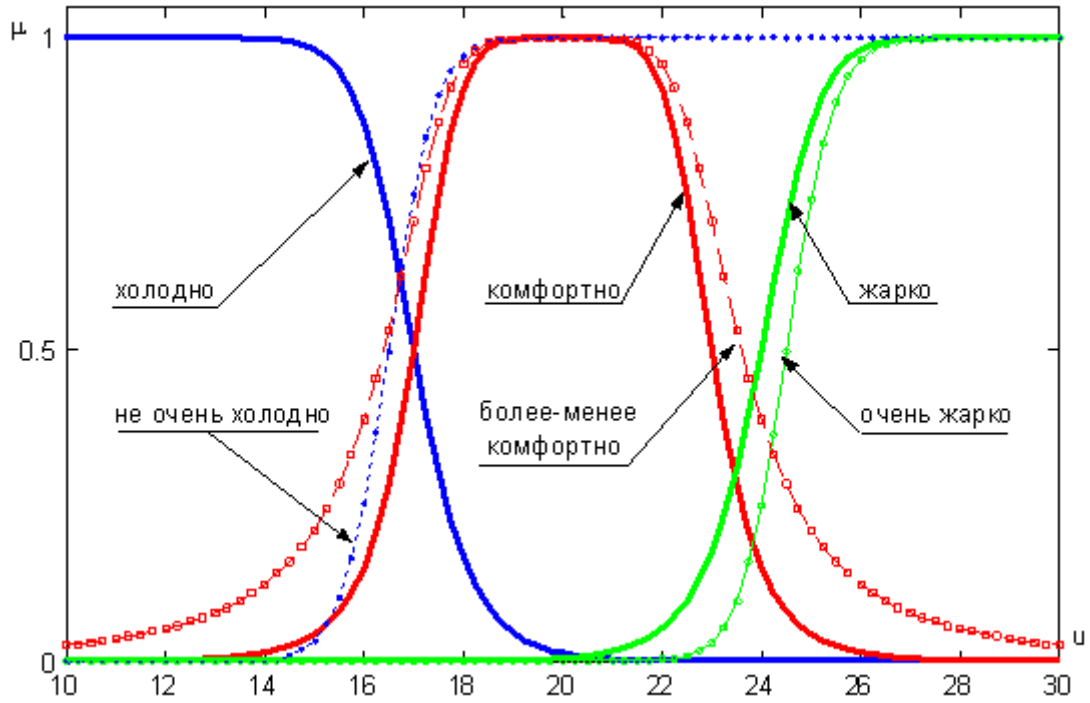


Рисунок 6.1 – Функции принадлежности термов лингвистической переменной «температура в комнате»

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 в **T(X)** 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.

Будем говорить, что лингвистическая переменная **X структурирована**, если ее терм-множество **T(X)** и функцию **M**, которая ставит в соответствие каждому элементу терм-множества его смысл, можно задать алгоритмически.

Пример. В качестве очень простой иллюстрации той роли, которую играют синтаксическое и семантическое правила в случае структурированной лингвистической переменной, рассмотрим переменную **РОСТ**, терм-множество которой можно записать в виде:

T(РОСТ)={ВЫСОКИЙ, ОЧЕНЬ ВЫСОКИЙ, ОЧЕНЬ-ОЧЕНЬ ВЫСОКИЙ,...}.

М(ВЫСОКИЙ)= $\begin{cases} \left(1+\left(\frac{u-60}{5}\right)^{-2}\right)^{-1}, & \text{если } u \geqslant 60, \\ 0, & \text{в противном случае.} \end{cases}$

М(ОЧЕНЬ ВЫСОКИЙ)=(М(ВЫСОКИЙ))^2, и т.д.

Лингвистическую переменную будем называть **булевой**, если ее термы являются булевыми комбинациями переменных вида X_p и hX, где **h** — лингвистическая неопределенность, X_p — атомарный терм.

Пример. Пусть "ВОЗРАСТ" — булева лингвистическая переменная с терм-множеством вида:

Т(ВОЗРАСТ)={МОЛОДОЙ, НЕМОЛОДОЙ, СТАРЫЙ, НЕСТАРЫЙ, ОЧЕНЬ МОЛОДОЙ, НЕ МОЛОДОЙ И НЕ СТАРЫЙ, МОЛОДОЙ ИЛИ НЕ ОЧЕНЬ СТАРЫЙ, ...}.

В этом примере имеется два атомарных терма — МОЛОДОЙ и СТАРЫЙ и одна неопределенность — ОЧЕНЬ.

Если отождествлять союз «И» с операцией пересечения нечетких множеств, «ИЛИ» — с операцией объединения нечетких множеств, отрицание «НЕ» — с операцией взятия дополнения и модификатор «ОЧЕНЬ» — с операцией концентрирования, то данная переменная будет полностью структурирована.

6.3. Лингвистические переменные истинности

В каждодневных разговорах мы часто характеризуем степень истинности утверждения посредством таких выражений, как "очень верно", "совершенно верно", "более или менее верно", "ложно", "абсолютно ложно" и т.д. Сходство между этими выражениями и значениями лингвистической переменной наводит на мысль о том, что в ситуациях, когда истинность или ложность утверждения определены недостаточно четко, может оказаться целесообразным трактовать ИСТИННОСТЬ как лингвистическую переменную, для которой ИСТИНО и ЛОЖНО — лишь два атомарных терма в терм-множестве этой переменной. Такую переменную будем называть **лингвистической переменной истинности**, а ее значения — **лингвистическими значениями истинности**.

Трактовка истинности как лингвистической переменной приводит к нечеткой лингвистической логике, которая совершенно отлична от обычной двузначной или даже многозначной логики. Такая нечеткая логика является основой того, что можно было бы назвать приближенными рассуждениями, т.е. видом рассуждений, в которых значения истинности и правила их вывода являются нечеткими, а не точными. Приближенные рассуждения во многом сродни тем, которыми пользуются люди в некорректно определенных или не поддающихся количественному описанию ситуациях. В самом деле, вполне

возможно, что многие, если не большинство человеческих рассуждений по своей природе приближенны, а не точны.

В дальнейшем будем пользоваться термином "**нечеткое высказывание**" для обозначения утверждения вида "**u** есть **A** ", где **u** — название предмета, а **A** — название нечеткого подмножества универсального множества **U**, **например**, "Джон — молодой", "X — малый", "яблоко — красное" и т.п. Если интерпретировать **A** как нечеткий предикат, то утверждение "**u** есть **A** " можно перефразировать как "**u** имеет свойство **A**".

Будем полагать, что высказыванию типа "**u** есть **A**" соответствуют два нечетких подмножества:

1. **M(A)** — смысл **A**, т.е. нечеткое подмножество с названием **A** универсального множества **U**;
2. Значение истинности утверждения "**u** есть **A**", которое будем обозначать $v(A)$ и определять как возможно нечеткое подмножество универсального множества значений истинности **V**. Будем предполагать, что **V=[0,1]**.

Значение истинности, являющееся числом в [0,1], **например** $v(A) = 0,8$, будем называть **числовым** значением истинности. Числовые значения истинности играют роль значений базовой переменной для лингвистической переменной ИСТИННОСТЬ. Лингвистические значения переменной ИСТИННОСТЬ будем называть **лингвистическими значениями истинности**. Более точно будем предполагать, что ИСТИННОСТЬ — название булевой лингвистической переменной, для которой атомарным является терм ИСТИННЫЙ, а терм ЛОЖНЫЙ определяется не как отрицание терма ИСТИННЫЙ, а как его зеркальное отображение относительно точки 0,5.

Предполагается, что смысл первичного терма ИСТИННЫЙ является нечетким подмножеством интервала **V=[0,1]** с функцией принадлежности типа (по Заде)

$$\mu_{\text{истинный}}(u) = \begin{cases} 0, & \text{если} \quad 0 \leqslant u \leqslant a; \\ 2\left(\frac{u-a}{1-a}\right)^2, & \text{если} \quad a \leqslant u \leqslant \frac{1+a}{2}; \\ 1 - 2\left(\frac{u-a}{1-a}\right)^2, & \text{если} \quad \frac{1+a}{2} \leqslant u \leqslant 1, \end{cases}$$

показанной на рисунке 6.2. Они построены при значении параметра a=0,4.

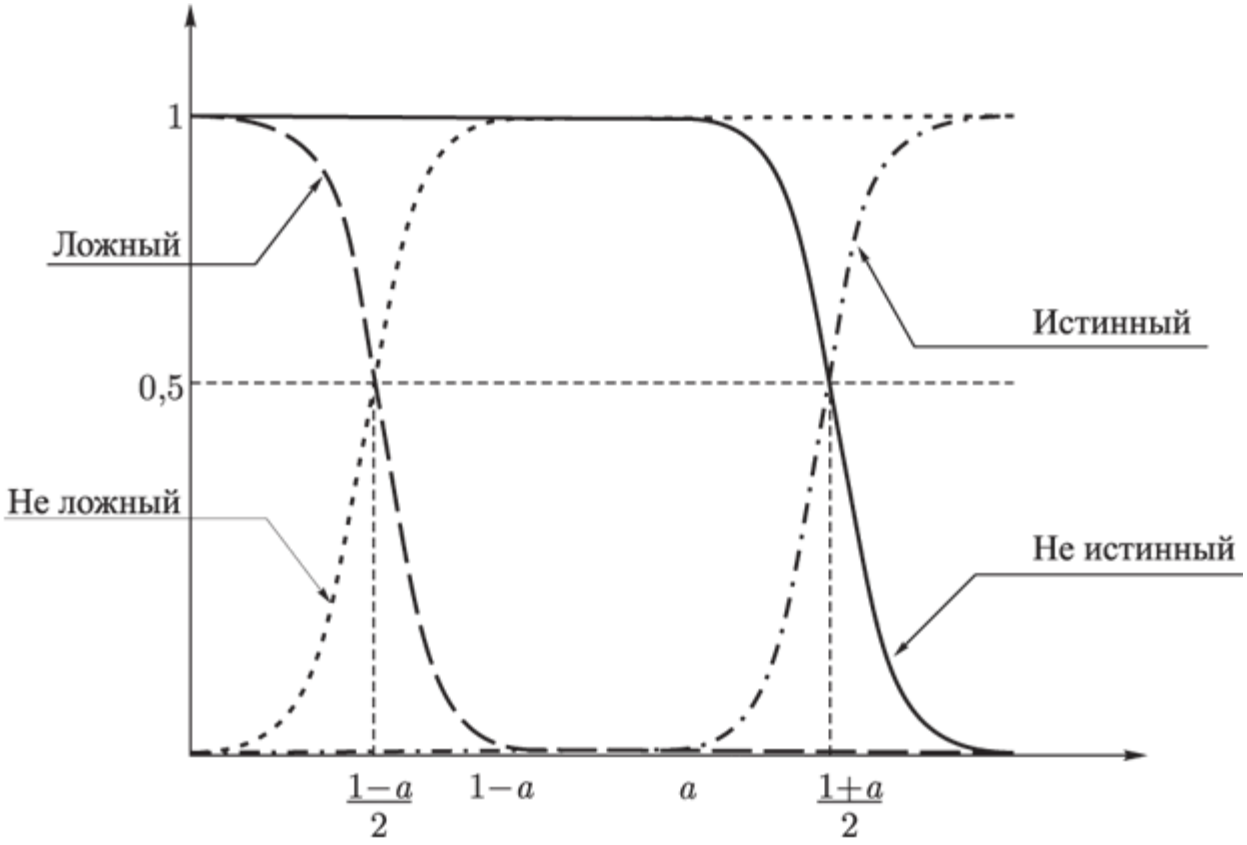


Рисунок 6.2 – Функции принадлежности лингвистической переменной "истинность" по Заде

Здесь точка $u = \frac{1+a}{2}$ является точкой перехода. Соответственно, для терма ЛОЖНЫЙ имеем $\mu_{\text{ЛОЖНЫЙ}}(u) = \mu_{\text{ИСТИННЫЙ}}(1-u).$

Для задания нечеткой истинности Балдвин предложил такие функции принадлежности нечетких "истинно" и "ложно":

$\mu_{\text{"истинно"}}(u) = u;$

$\mu_{\text{"ложно"}}(u) = 1 - u;$

где $u \in [0,1]$.

Квантификаторы "более-менее" и "очень" часто применяют к нечеткими множествами "истинно" и "ложно", получая таким образом термы "очень ложно", "более-менее ложно", "более-менее истинно", "очень истинно", "очень, очень истинно", "очень, очень ложно" и т.п. Функции принадлежности новых термов получают, выполняя операции концентрации и растяжения нечетких множеств "истинно" и "ложно". Операция концентрации соответствует возведению функции принадлежности в квадрат, а операция растяжения - возведению в степень ½. Следовательно, функции принадлежности термов "очень, очень ложно", "очень ложно", "более-менее ложно", "более-менее истинно", "истинно", "очень истинно" и "очень, очень истинно" задаются так:

$\mu_{\text{"очень ложно"}}(u) = (\mu_{\text{"ложно"}}(u))^2;$

$$\mu_{\text{"очень, очень ложно"}}(u) = \left(\mu_{\text{"очень ложно"}}(u)\right)^2$$

$$\mu_{\text{"более-менее ложно"}}(u) = \left(\mu_{\text{"ложно"}}(u)\right)^{1/2};$$

$$\mu_{\text{"более-менее истинно"}}(u) = \left(\mu_{\text{"истинно"}}(u)\right)^{1/2};$$

$$\mu_{\text{"очень истинно"}}(u) = \left(\mu_{\text{"истинно"}}(u)\right)^2;$$

$$\mu_{\text{"очень, очень истинно"}}(u) = \left(\mu_{\text{"очень истинно"}}(u)\right)^2.$$

Графики функций принадлежности этих термов показаны на рисунке 6.3.

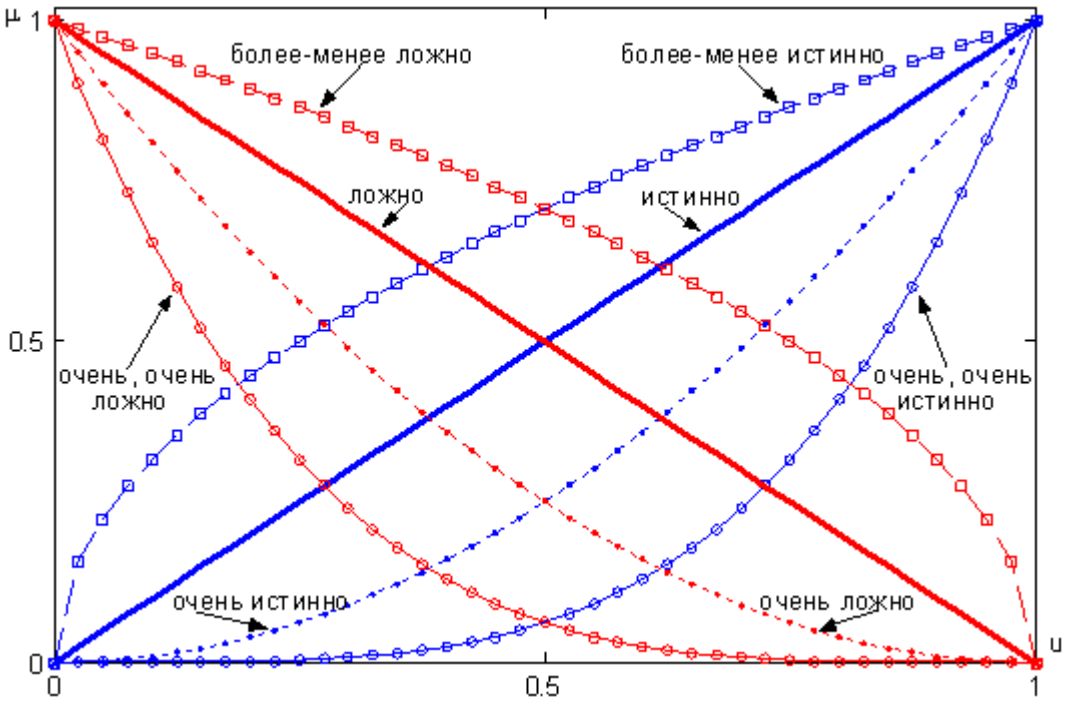


Рисунок 6.3 - Лингвистическая переменная "истинность" по Балдвину

6.4. Логические связки в нечеткой лингвистической логике

Чтобы заложить основу для нечеткой лингвистической логики, необходимо расширить содержание таких логических операций, как отрицание, дизъюнкция, конъюнкция и импликация, применительно к высказываниям, которые имеют не числовые, а лингвистические значения истинности.

При рассмотрении этой проблемы полезно иметь в виду, что если **A** — нечеткое подмножество универсального множества **U** и $u \in U$, то два следующих утверждения эквивалентны:

1. Степень принадлежности элемента u нечеткому множеству A есть $\mu_A(u)$.
2. Значение истинности нечеткого предиката " u есть A " также равно $\mu_A(u)$.

Таким образом, вопрос "Что является значением истинности высказывания " u есть A " И " u есть B ", если заданы лингвистические значения истинности высказываний " u есть A " и " u есть B ?" аналогичен вопросу "Какова степень принадлежности элемента u множеству $A \cap B$, если заданы степени принадлежности элемента u множествам A и B ?".

В частности, если $v(A)$ — точка в $V = [0, 1]$, представляющая значение истинности высказывания " u есть A " (или просто A), где u — элемент универсального множества U, то значение истинности высказывания " u есть не A " (или $\bar{A}$) определяется выражением $v(\neg A) = 1 - v(A)$.

Предположим теперь, что $v(A)$ — не точка в $[0, 1]$, нечеткое подмножество интервала $[0, 1]$, представленное в виде $v(A) = f(x), \qquad f : [0, 1] \to [0, 1]$.

Тогда получим $v(\neg A) = f(1 - x)$.

В частности, если значение истинности A есть ИСТИННО, т.е. $v(A) =$ ИСТИННО, то значение истинности ЛОЖНО является значением истинности для высказывания $\neg A$.

Замечание. Следует отметить, что если ИСТИННЫЙ $= f(x)$, то функция $1 - f(x)$ будет интерпретироваться термом НЕ ИСТИННЫЙ, а функция $f(1 - x)$ — термом ЛОЖНЫЙ, что в принципе не одно и то же (рисунок 6.2).

То же самое относится к лингвистическим неопределенностям. Например, если ИСТИННЫЙ $= f(x)$, то значение терма ОЧЕНЬ ИСТИННЫЙ равно $f^2(x)$ (рисунок 6.3).

С другой стороны, если значение истинности высказывания A есть $f(x)$, то функция $f(x^2)$ будет выражать значение истинности высказывания "очень A ".

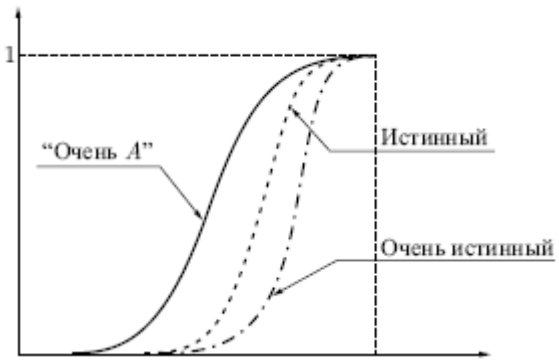


Рисунок 6.3 – Значения термов «очень истинный» и «истинный»

Перейдем к бинарным связкам. Пусть $v(A)$ и $v(B)$ — лингвистические значения истинности высказываний A и B соответственно. В случае, когда $v(A)$ и $v(B)$ — точечные оценки, имеем:

$$v(A) \wedge v(B) = v(A \text{ И } B), v(A) \vee v(B) = v(A \text{ ИЛИ } B),$$

где операции $\wedge$ и $\vee$ сводятся к операциям нечеткой логики (см. "Нечеткая логика").

Если $v(A)$ и $v(B)$ — лингвистические значения истинности, заданные функциями $v(A) = f(x), \quad v(B) = g(x), \qquad f, g : [0,1] \to [0,1],$ то, согласно принципу обобщения, конъюнкция и дизъюнкция будут вычисляться по следующим формулам:

$$v(A) \wedge v(B) \quad \Leftrightarrow \quad \sup_{z=x \wedge y} \left(\mu_A(x) \wedge \mu_B(y)\right),$$

$$v(A) \vee v(B) \quad \Leftrightarrow \quad \sup_{z=x \vee y} \left(\mu_A(x) \wedge \mu_B(y)\right).$$

Замечание Важно четко понимать разницу между связкой И (ИЛИ) в терме, например, ИСТИННЫЙ И (ИЛИ) НЕ ИСТИННЫЙ и символом $\wedge$ ($\vee$) в высказывании ИСТИННЫЙ $\wedge$ ($\vee$) НЕ ИСТИННЫЙ. В первом случае, нас интересует смысл терма ИСТИННЫЙ И (ИЛИ) НЕ ИСТИННЫЙ, и связка И (ИЛИ) определяется отношением M (ИСТИННЫЙ И (ИЛИ) НЕ ИСТИННЫЙ) = M (ИСТИННЫЙ) $\cap$ ($\cup$) M (НЕ ИСТИННЫЙ),

где $M(A)$ — смысл терма A. Напротив, в случае терма ИСТИННЫЙ $\wedge$ ($\vee$) НЕ ИСТИННЫЙ нас в основном интересует значение истинности высказывания ИСТИННЫЙ $[\wedge\ (\vee)]$ НЕ ИСТИННЫЙ, которое получается из равенства v (А И (ИЛИ) В) = $v(A) \wedge\ (\vee) v(B)$.

6.5. Значения истинности «Неизвестно» и «Не определено»

Среди возможных значений истинности лингвистической переменной ИСТИННОСТЬ два значения привлекают особое внимание, а именно пустое множество $\varnothing$ и единичный интервал $\mathfrak{F} = [0,1]$, которые соответствуют наименьшему и наибольшему элементам (по отношению включения) решетки нечетких подмножеств интервала $[0,1]$. Важность именно этих значений истинности обусловлена тем, что их можно интерпретировать как значения истинности НЕ ОПРЕДЕЛЕНО и НЕИЗВЕСТНО соответственно.

Важно четко понимать разницу между 0 и $\oslash$. Когда мы говорим, что степень принадлежности точки u множеству A есть $\oslash$, мы имеем в виду, что функция принадлежности $\mu_A : U \to [0,1]$ не определена в точке u. Предположим, например, что U — множество действительных чисел, а μ_A — функция, определенная на множестве целых чисел, причем $\mu_A(u) = 1$, если u четное, и $\mu_A(u) = 0$, если u нечетное. Тогда степень принадлежности числа $u = 1,5$ множеству A есть $\oslash$, а не 0.

С другой стороны, если бы μ_A была определена на множестве действительных чисел и $\mu_A(u) = 1$ тогда и только тогда, если u — четное число, то степень принадлежности числа 1,5 множеству A была бы равна 0.

Понятие значения истинности НЕИЗВЕСТНО в сочетании с принципом обобщения помогает уяснить некоторые понятия и соотношения обычных двухзначных и трехзначных логик. Эти логики можно рассматривать как вырожденные случаи нечеткой логики, в которой значением истинности НЕИЗВЕСТНО является весь единичный интервал, а не множество $\{0,1\}$.

Вопросы для повторения и закрепления материала

1. В чем заключается суть нечеткой логики?
2. Какие факторы определяют выбор нечеткой логики для моделирования систем?
3. Каким образом определяется нечеткая переменная?
4. Каким образом задается лингвистическая переменная?
5. Охарактеризуйте лингвистические переменные истинности?
6. Какую роль играют логические связки в нечеткой логике?
7. Что представляет собой синтаксическое правило?
8. Что представляет собой семантическое правило?

Задания для самостоятельной работы

1. Опишите процесс изменения курса валюты при помощи нечеткой логики с описанием всех элементов лингвистической переменной.

Тема 7. Теория приближенных рассуждений

Цель: ознакомиться с основными положениями теории приближенных рассуждений.

Задачи:

1. Рассмотреть композиционное правило.
2. Рассмотреть правило modus ponens как частный случай композиционного правила вывода.

3. Ознакомиться с понятием и принципами работы нечетких экспертных систем.

7.1. Композиционное правило вывода

Под **приближенными рассуждениями** понимается процесс, при котором из нечетких посылок получают некоторые следствия, возможно, тоже нечеткие. Приближенные рассуждения лежат в основе способности человека понимать естественный язык, разбирать почерк, играть в игры, требующие умственных усилий, в общем, принимать решения в сложной и не полностью определенной среде. Эта способность рассуждений в качественных, неточных терминах отличает интеллект человека от интеллекта вычислительной машины.

Основным правилом вывода в традиционной логике является правило modus ponens, согласно которому мы судим об истинности высказывания B по истинности высказываний A и $A \to B$. **Например**, если A — высказывание "Джон в больнице", B — высказывание "Джон болен", то если истинны высказывания "Джон в больнице" и "Если Джон в больнице, то он болен", то истинно и высказывание "Джон болен".

Во многих привычных рассуждениях, однако, правило modus ponens используется не в точной, а в приближенной форме. Так, обычно мы знаем, что A истинно и что $A^* \to B$, где A^* есть, в некотором смысле, приближение A. Тогда из $A^* \to B$ мы можем сделать вывод о том, что B приближенно истинно.

Однако, в отличие от традиционной логики, нашим главным инструментом будет не правило modus ponens, а так называемое композиционное правило вывода, частным случаем которого является правило modus ponens.

Композиционное правило вывода — это всего лишь обобщение следующей знакомой процедуры. Предположим, что имеется кривая $y = f(x)$ (рисунок 7.1(А)) и задано значение $x = a$. Тогда из того, что $y = f(x)$ и $x = a$, мы можем заключить, что $y = b = f(a)$.

Обобщим теперь этот процесс, предположив, что a — интервал, а $f(x)$ — функция, значения которой суть интервалы, как на рисунке 7.1(Б). В этом случае, чтобы найти интервал $y = b$, соответствующий интервалу a, мы сначала построим цилиндрическое множество $\bar{a}$ с основанием a и найдем его пересечение I с кривой, значения которой суть интервалы. Затем спроецируем это пересечение на ось OY и получим желаемое значение y в виде интервала b.

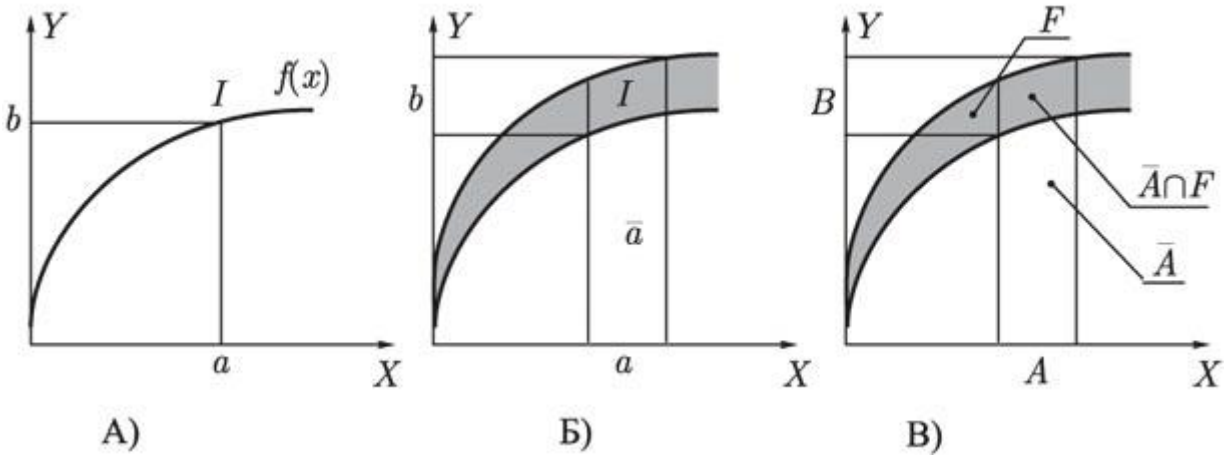


Рисунок 7.1 – Композиционное правило

Чтобы продвинуться еще на один шаг по пути обобщения, предположим, что A — нечеткое подмножество оси OX, а F — нечеткое отношение в $OX \times OY$ (рисунок 7.1(В)). Вновь образуя цилиндрическое нечеткое множество $\bar{A}$ с основанием A и его пересечение с нечетким отношением F, мы получим нечеткое множество $\bar{A} \cap F$, которое является аналогом точки пересечения I на рисунке 7.1(А). Таким образом, из того, что $y = f(x)$ и $x = A$ — нечеткое подмножество оси OX, мы получаем значение y в виде нечеткого подмножества B оси OY.

Правило. Пусть U и V — два универсальных множества с базовыми переменными u и v, соответственно. Пусть A и F — нечеткие подмножества множеств U и $U \times V$. **Тогда композиционное правило вывода утверждает,** что из нечетких множеств A и F следует нечеткое множество $B = A \circ F$. Согласно определению композиции нечетких множеств, получим

$$\mu_B(v) = \bigvee_{u \in U} (\mu_A(u) \wedge \mu_F(u, v)) .$$

Пример. Пусть $U = V = \{1, 2, 3, 4\}$,

А = МАЛЫЙ $= \{\langle 1|1\rangle , \langle 0,6|2\rangle , \langle 0,2|3\rangle , \langle 0|4\rangle\}$,

F=ПРИМЕРНО РАВНЫ =

	1	2	3	4
1	1	0,5	0	0
2	0,5	1	0,5	0
3	0	0,5	1	0,5
4	0	0	0,5	1

Тогда получим ($\mu_{\tilde{G}}(x, y) = \sup_{z \in Z} \min\left(\mu_{\tilde{A}}(x, z), \mu_{\tilde{B}}(z, y)\right)$)

$$B = [1 \quad 0,6 \quad 0,2 \quad 0] \circ \begin{bmatrix} 1 & 0,5 & 0 & 0 \\ 0,5 & 1 & 0,5 & 0 \\ 0 & 0,5 & 1 & 0,5 \\ 0 & 0 & 0,5 & 1 \end{bmatrix} = [1 \quad 0,6 \quad 0,5 \quad 0,2],$$

что можно проинтерпретировать следующим образом:

B = БОЛЕЕ ИЛИ МЕНЕЕ МАЛЫЙ, если терм БОЛЕЕ ИЛИ МЕНЕЕ определяется как оператор увеличения нечеткости.

Словами этот приближенный вывод можно записать в виде

$$\begin{array}{lll} u \quad - \text{ МАЛЫЙ} & & \text{предпосылка} \\ \underline{u, v \quad - \text{ ПРИМЕРНО РАВНЫ}} & & \text{предпосылка} \\ v \quad - \text{ БОЛЕЕ ИЛИ МЕНЕЕ МАЛЫЙ} & & \text{приближенный вывод} \end{array}$$

7.2. Правило modus ponens как частный случай композиционного правила вывода

Правило modus ponens можно рассматривать как частный случай композиционного правила вывода. Чтобы установить эту связь, мы сперва обобщим понятие материальной импликации с пропозициональными переменными на нечеткие множества.

Пусть A и B — нечеткие высказывания и μ_A, μ_B — соответствующие им функции принадлежности. Тогда импликации $A \to B$ будет соответствовать некоторая функция принадлежности $\mu_{A \to B}$. По аналогии с традиционной логикой, можно предположить, что

$$A \to B \equiv \neg A \vee B.$$

Тогда

$$\mu_{A \to B}(x, y) = \max\{1 - \mu_A(x), \mu_B(y)\}.$$

Однако, это не единственное обобщение оператора импликации. В таблице 7.1 показаны различные интерпретации этого понятия.

Таблица 7.1 – Интерпретации операции импликация

Larsen	$\mu_{A \to B}(x, y) = \mu_A(x)\mu_B(y)$
Lukasiewicz	$\mu_{A \to B}(x, y) = \min\{1, 1 - \mu_A(x) + \mu_B(y)\}$
Mamdani	$\mu_{A \to B}(x, y) = \min\{\mu_A(x), \mu_B(y)\}$
Standard Strict	$\mu_{A \to B}(x, y) = \begin{cases} 1, & \text{если } \mu_A(x) \leqslant \mu_B(y); \\ 0, & \text{в противном случае.} \end{cases}$
Godel	$\mu_{A \to B}(x, y) = \begin{cases} 1, & \text{если } \mu_A(x) \leqslant \mu_B(y); \\ \mu_B(y), & \text{в противном случае.} \end{cases}$

Gaines	$\mu_{A\to B}(x,y)=\begin{cases}1, & \text{если } \mu_A(x)\leqslant\mu_B(y);\\ \frac{\mu_B(y)}{\mu_A(x)}, & \text{в противном случае.}\end{cases}$
Kleene-Dienes	$\mu_{A\to B}(x,y)=\max\{1-\mu_A(x),\ \mu_B(y)\}$
Kleene-Dienes-Lu	$\mu_{A\to B}(x,y)=1-\mu_A(x)+\mu_A(x)\mu_B(y)$

Определим теперь **обобщенное правило** modus ponens (generalized modus ponens).

Предпосылка	$A\to B$
Событие	A^*
Вывод	$A*\circ(A\to B)$

Приведенная формулировка имеет два отличия от традиционной формулировки правила modus ponens: во-первых, здесь допускается, что A, A^*, B — нечеткие множества, и, во-вторых, A^* необязательно идентично A.

7.3. Нечеткие экспертные системы

Логико-лингвистические методы описания систем основаны на том, что поведение исследуемой системы описывается в естественном (или близком к естественному) языке в терминах лингвистических переменных. Входные и выходные параметры системы рассматриваются как лингвистические переменные, а качественное описание процесса задается совокупностью высказываний следующего вида:

L_1 : если A_{11} и/или A_2 и/или ... и/или A_{1m}, то B_{11} и/или ... и/или B_{1n},

L_2 : если A_{21} и/или A_{22} и/или ... и/или A_{2m}, то B_{21} и/или ... и/или B_{2n},

.....

L_k : если A_{k1} и/или A_{k2} и/или ... и/или A_{km}, то B_{k1} и/или ... и/или B_{kn},

где A_{ij}, $i=1,2\ldots,k \quad j=1,2,\ldots,m$ — нечеткие высказывания, определенные на значениях входных лингвистических переменных, а B_{ij}, $i=1,2,\ldots,k \quad j=1,2,\ldots,n$ — нечеткие высказывания, определенные на значениях выходных лингвистических переменных. Эта совокупность правил носит название **нечеткой базы знаний**.

Подобные вычисления составляют основу нечетких экспертных систем. Каждая нечеткая экспертная система использует нечеткие утверждения и правила.

Затем с помощью операторов вычисления дизъюнкции и конъюнкции описание системы можно привести к виду

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 $A_1, A_2, \ldots, A_k$ — нечеткие множества, заданные на декартовом произведении X универсальных множеств входных лингвистических переменных, а $B_1, B_2, \ldots, B_k$ — нечеткие множества, заданные на декартовом произведении Y универсальных множеств выходных лингвистических переменных.

В основе построения логико-лингвистических систем лежит рассмотренное выше композиционное правило вывода Заде.

Преимущество данной модели - в ее универсальности. Нам неважно, что именно на входе — конкретные числовые значения или некоторая неопределенность, описываемая нечетким множеством. Но за данную универсальность приходится расплачиваться сложностью системы — нам приходится работать в пространстве размерности $m \times n$. Поэтому этой общей моделью на практике пользуются довольно редко. Обычно же используют ее упрощенный вариант, называемый нечетким выводом. Он основывается на предположении, что все входные лингвистические переменные имеют известные нам числовые значения (как и бывает довольно часто на практике). Также обычно не используют более одной выходной лингвистической переменной.

Нечеткой базой знаний называется совокупность нечетких правил "Если - то", определяющих взаимосвязь между входами и выходами исследуемого объекта.

Пример. Следующая нечеткая база знаний описывает зависимость между возрастом водителя (x) и возможностью дорожно-транспортного происшествия (y):

Если x = Молодой, то y = Высокая;

Если x = Средний, то y = Низкая;

Если x = Очень старый, то y = Высокая.

Пусть функции принадлежностей термов имеют вид, показанный на рисунке 7.2.

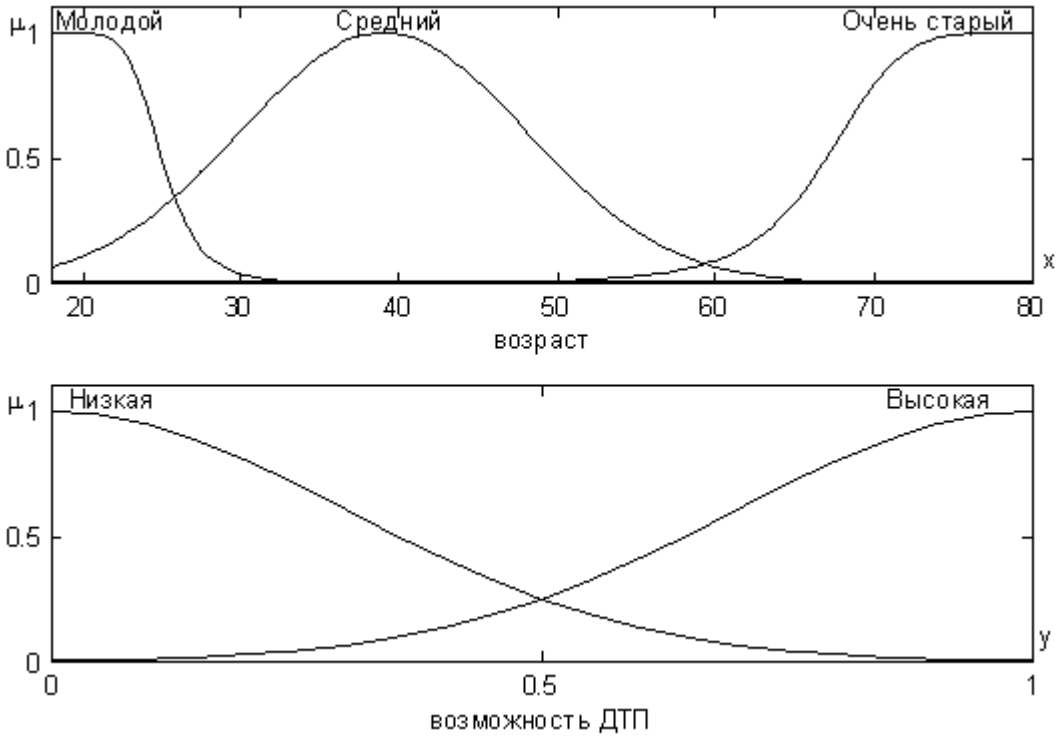


Рисунок 7.2 - Функции принадлежности термов

Тогда нечеткие отношения, соответствующие правилам базы знаний, будут такими, как на рисунке 7.3.

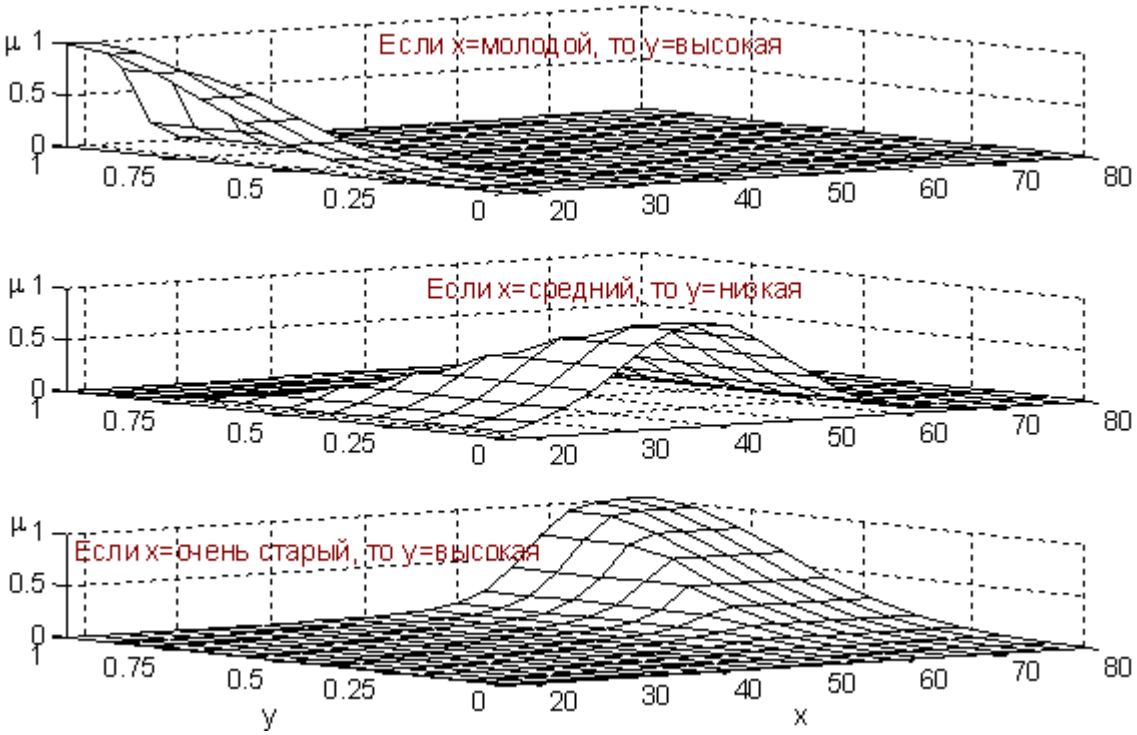


Рисунок 7.3 - Нечеткие отношения, соответствующие правилам базы знаний

Для задания многомерных зависимостей "входы-выходы" используют нечеткие логические операции И и ИЛИ. Удобно правила формулировать так, чтобы внутри каждого правил переменные объединялись логической операцией И, а правила в базе знаний связывались операцией ИЛИ. В этом случае нечеткую базу знаний, связывающую входы с выходом, можно представить в следующем виде:

ЕСЛИ $(x_1 = a_{1,j1})$ И $(x_2 = a_{2,j1})$ И...И $(x_n = a_{n,j1})$

ИЛИ $(x_1 = a_{1,j2})$ И $(x_2 = a_{2,j2})$ И...И $(x_n = a_{n,j2})$

...

ИЛИ $(x_1 = a_{1,jk_j})$ И $(x_2 = a_{2,jk_j})$ И...И $(x_n = a_{n,jk_j})$,

ТО $y = d_j$, $j = \overline{1,m}$,

где $a_{i,jp}$ - нечеткий терм, которым оценивается переменная x_i в строчке с номером jp ($p = \overline{1,k_j}$);

k_j - количество строчек-конъюнкций, в которых выход y оценивается значений d_j;

m - количество различных значений, используемых для оценки выходной переменной y.

Приведенную выше базу знаний удобно представлять таблицей, которую иногда называют **матрицей знаний** (таблица 7.2).

Таблица 7.2 - Нечеткая база знаний

x_1	x_2	...	x_n	y
$a_{1,1,1}$	$a_{2,1,1}$	...	$a_{n,1,1}$	d_1
$a_{1,1,2}$	$a_{2,1,2}$	...	$a_{n,1,2}$	
...	...	...	...	
$a_{1,1,k_1}$	$a_{2,1,k_1}$	...	$a_{n,1,k_1}$	
$a_{1,2,1}$	$a_{2,2,1}$	...	$a_{n,2,1}$	d_2
$a_{1,2,2}$	$a_{2,2,2}$	...	$a_{n,2,2}$	
...	...	...	...	
$a_{1,2,k_2}$	$a_{2,2,k_2}$	...	$a_{n,2,k_2}$	
...				
$a_{1,m,1}$	$a_{2,m,1}$	...	$a_{n,m,1}$	d_m
$a_{1,m,2}$	$a_{2,m,2}$	...	$a_{n,m,2}$	

…	…	…	…	
a_{1,m,k_m}	a_{2,m,k_m}	…	a_{n,m,k_m}	

Для учета различной степени уверенности эксперта в адекватности правил используют весовые коэффициенты. Нечеткую базу знаний из таблицы 7.2 с весовыми коэффициентами правил можно записать следующим образом:

$$\bigcup_{p=1}^{k_j}\left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \text{ с весом } w_{jp}\right) \to y = d_j, \quad j = \overline{1,m},$$

где $\cup$ - нечеткая логическая операция ИЛИ;

$\cap$ - нечеткая логическая операция И;

$w_{jp} \in [0,1]$ - ; весовой коэффициент правила с номером jp.

Нечеткий логический вывод (fuzzy logic inference) - аппроксимация зависимости $Y = f(X_1, X_2, \ldots, X_n)$ каждой выходной лингвистической переменной от входных лингвистических переменных и получение заключения в виде нечеткого множества, соответствующего текущим значениям входов, с использованием нечеткой базы знаний и нечетких операций. Основу нечеткого логического вывода составляет **композиционное правило Заде**.

В общем случае нечеткий вывод решения происходит за три (или четыре) шага:

1) **этап фаззификации**. С помощью функций принадлежности всех термов входных лингвистических переменных и на основании задаваемых четких значений из универсумов входных лингвистических переменных определяются степени уверенности в том, что выходная лингвистическая переменная принимает конкретное значение. Эта степень уверенности есть ордината точки пересечения графика функции принадлежности терма и прямой $x =$ четкое значение ЛП.

2) **этап непосредственного нечеткого вывода**. На основании набора правил — нечеткой базы знаний — вычисляется значение истинности для предпосылки каждого правила на основании конкретных нечетких операций, соответствующих конъюнкции или дизъюнкции термов в левой части правил. В большинстве случаев это либо максимум, либо минимум из степеней уверенности термов, вычисленных на этапе фаззификации, который применяется к заключению каждого правила. Используя один из способов построения нечеткой импликации, мы получим нечеткую переменную, соответствующую вычисленному значению степени уверенности в левой части правила и нечеткому множеству в правой части правила.

Обычно в качестве вывода используется минимизация или правила продукции. При минимизирующем логическом выводе выходная функция

принадлежности ограничена сверху в соответствии с вычисленной степенью истинности посылки правила (нечеткое логическое И). В логическом выводе с использованием продукции выходная функция принадлежности масштабируется с помощью вычисленной степени истинности предпосылки правила.

3) **этап композиции (агрегации, аккумуляции)**. Все нечеткие множества, назначенные для каждого терма каждой выходной лингвистической переменной, объединяются вместе, и формируется единственное нечеткое множество — значение для каждой выводимой лингвистической переменной. Обычно используются функции **MAX** или **SUM**.

4) **этап дефаззификации (необязательный)**. Используется тогда, когда полезно преобразовать нечеткий набор значений выводимых лингвистических переменных к точным. Имеется достаточно большое количество методов перехода к точным значениям (по крайней мере, 30). Два примера общих методов — "методы полной интерпретации" и "по максимуму". В методе полной интерпретации точное значение выводимой переменной вычисляется как значение "центра тяжести" функции принадлежности для нечеткого значения. В методе максимума в качестве точного значения выводимой переменной принимается максимальное значение функции принадлежности.

В теории нечетких множеств процедура дефаззификации аналогична нахождению характеристик положения (математического ожидания, моды, медианы) случайных величин в теории вероятности. Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа, соответствующего максимуму функции принадлежности. Однако пригодность этого способа распространяется лишь на одноэкстремальные функции принадлежности. Для многоэкстремальных функций принадлежности часто используются следующие методы дефаззификации:

1) **COG** (Center Of Gravity) — "центр тяжести". Физическим аналогом этой формулы является нахождение центра тяжести плоской фигуры, ограниченной осями координат и графиком функции принадлежности нечеткого множества.

2) **MOM** (Mean Of Maximums) — "центр максимумов". При использовании метода центра максимумов требуется найти среднее арифметическое элементов универсального множества, имеющих максимальные степени принадлежностей.

3) **First Maximum** — "первый максимум" — максимум функции принадлежности с наименьшей абсциссой.

В теории нечетких множеств процедура дефаззификации аналогична нахождения характеристик положения (математического ожидания, моды, медианы) случайных величин в теории вероятности. Простейшим способом выполнения процедуры дефаззификации является выбор четкого числа,

соответствующего максимуму функции принадлежности. Однако пригодность этого способа ограничивается лишь одноэкстремальными функциями принадлежности. Для многоэкстремальных функций принадлежности в Fuzzy Logic Toolbox запрограммированы такие методы дефаззификации:

- Centroid - центр тяжести;
- Bisector - медиана;
- LOM (Largest Of Maximums) - наибольший из максимумов;
- SOM (Smallest Of Maximums) - наименьший из максимумов;
- Mom (Mean Of Maximums) - центр максимумов.

Дефаззификация нечеткого множества $\tilde{A} = \int_{[\underline{u},\bar{u}]} \mu_A(u)/u$ **по методу центра тяжести** осуществляется по формуле

$$a = \frac{\int_{\underline{u}}^{\bar{u}} u \cdot \mu_A(u)\, du}{\int_{\underline{u}}^{\bar{u}} \mu_A(u)\, du}.$$

Физическим аналогом этой формулы является нахождение центра тяжести плоской фигуры, ограниченной осями координат и графиком функции принадлежности нечеткого множества. В случае дискретного универсального множества дефаззификация нечеткого множества $\tilde{A} = \sum_{i=1}^{k} \mu_A(u_i)/u_i$ по методу центра тяжести осуществляется по формуле

$$a = \frac{\sum_{i=1}^{k} u_i \cdot \mu_A(u_i)}{\sum_{i=1}^{k} \mu_A(u_i)}.$$

Дефаззификация нечеткого множества $\tilde{A} = \int_{[\underline{u},\bar{u}]} \mu_A(u)/u$ **по методу медианы** состоит в нахождении такого числа a, что

$$\int_{\underline{u}}^{a} \mu_A(u)\, du = \int_{a}^{\bar{u}} \mu_A(u)\, du.$$

Геометрической интерпретацией метода медианы является нахождения такой точки на оси абсцисс, что перпендикуляр, восстановленный в этой точке, делит площадь под кривой функции принадлежности на две равные части. В случае дискретного универсального множества дефаззификация нечеткого множества $\tilde{A} = \sum_{i=1}^{k} \mu_A(u_i)/u_i$ по методу медианы осуществляется по формуле

$$a = \min_{\forall j \sum_{i=1}^{j} \mu_A(u_i) \geq \frac{1}{2} \sum_{i=1}^{k} \mu_A(u_i)} (u_j)$$

.

Дефаззификация нечеткого множества $\tilde{A} = \int_{[\underline{u},\bar{u}]} \mu_A(u)/u$ **по методу центра максимумов** осуществляется по формуле:

$$a = \frac{\int_G u du}{\int_G du},$$

где G – множество всех элементов из интервала $[\underline{u},\bar{u}]$, имеющих максимальную степень принадлежности нечеткому множеству $\tilde{A}$.

В методе центра максимумов находится среднее арифметическое элементов универсального множества, имеющих максимальные степени принадлежностей. Если множество таких элементов конечно, то формула упрощается к следующему виду:

$$a = \frac{\sum_{u_j \in G} u_j}{|G|},$$

где $|G|$ - мощность множества G.

В дискретном случае дефаззификация по методам наибольшего из максимумов и наименьшего из максимумов осуществляется по формулам $a = \max(G)$ и $a = \min(G)$, соответственно. Из последних трех формулы видно, что если функция принадлежности имеет только один максимум, то его координата и является четким аналогом нечеткого множества.

Функциональная схема процесса нечеткого вывода в упрощенном виде представлена на рисунке 7.4. На этой схеме выполнение первого этапа нечеткого вывода — фаззификации — осуществляет фаззификатор. За процедуру непосредственно нечеткого вывода ответственна машина нечеткого логического вывода, которая производит второй этап процесса вывода на основании задаваемой нечеткой базы знаний (набора правил), а также этап композиции. Дефаззификатор выполняет последний этап нечеткого вывода — дефаззификацию.

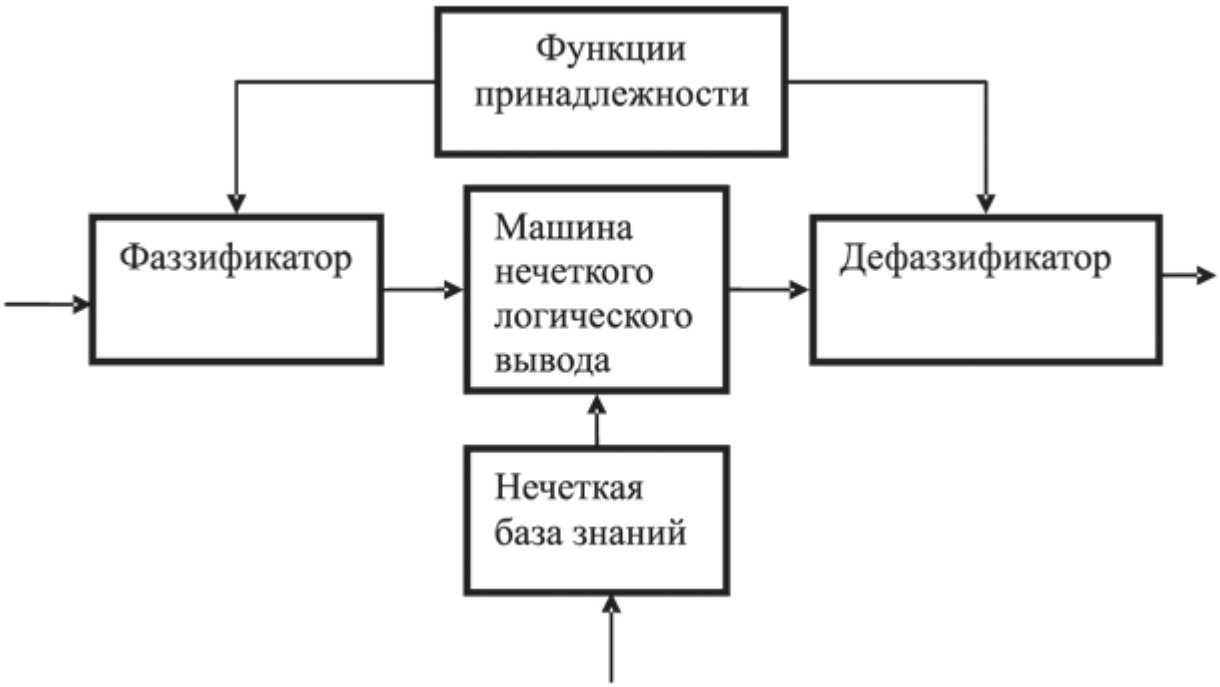


Рисунок 7.4 - Функциональная схема процесса нечеткого вывода

Вопросы для повторения и закрепления материала

1. В чем заключается суть теории приближенных рассуждений?
2. Что такое композиционное правило?
3. Что представляет собой правило modus ponens?
3. Дайте понятие нечеткой экспертной системе?
4. Охарактеризуйте принципы работы нечетких экспертных систем?
5. Перескажите пример реализации нечеткого логического вывода.
6. Назовите этапы осуществления нечеткого логического вывода?
7. Что делается на этапе фаззификации?

Задания для самостоятельной работы

1. Опишите нечеткую экспертную систему управления уровнем жидкости в баке.

2. Проведите анализ литературы на предмет областей использования нечетких экспертных систем.

Тема 8. Алгоритмы реализации нечеткого логического вывода

Цель: ознакомиться с алгоритмами осуществления нечеткого логического вывода Мамдани и Сугено.

Задачи:

1. Рассмотреть алгоритм Мамдани.

2. Рассмотреть Алгоритм Сугено.

3. Рассмотреть пример реализации нечеткого логического вывода.

8.1. Нечеткий логический вывод Мамдани

Нечеткий логический вывод по алгоритму Мамдани выполняется по нечеткой базе знаний:

$$\bigcup_{p=1}^{k_j}\left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \ \text{ с весом } w_{jp}\right) \to y = d_j, \quad j = \overline{1,m},$$

в которой значения входных и выходной переменной заданы нечеткими множествами. Введем следующие обозначения:

$\mu_{jp}(x_i)$ - функция принадлежности входа x_i нечеткому терму $a_{i,jp}$, т.е. $a_{i,jp} = \int_{\underline{x}_i}^{\bar{x}_i} \mu_{jp}(x_i)/x_i$, $x_i \in \left[\underline{x}_i, \bar{x}_i\right]$.

$\mu_{d_j}(y)$ - функция принадлежности выхода y нечеткому терму d_j, т.е. $d_j = \int_{\underline{y}}^{\bar{y}} \mu_{d_j}(y)/y$, $y \in \left[\underline{y}, \bar{y}\right]$.

Степени принадлежности входного вектора $X^* = \left(x_1^*, x_2^*, \ldots x_n^*\right)$ нечетким термам d_j из базы знаний рассчитывается следующим образом:

$$\mu_{d_j}\left(X^*\right) = \bigvee_{p=\overline{1,k_j}} w_{jp} \cdot \bigwedge_{i=\overline{1,n}} \left[\mu_{jp}\left(x_i^*\right)\right], \ j = \overline{1,m},$$

где $\vee(\wedge)$ - операция из s-нормы (t-нормы), т.е. из множества реализаций логической операций ИЛИ (И). Наиболее часто используются следующие реализации: для операции ИЛИ - нахождение максимума и для операции И - нахождение минимума.

В результате получаем такое нечеткое множество $\tilde{y}$, соответствующее входному вектору X^*:

$$\tilde{y} = \frac{\mu_{d_1}(X^*)}{d_1} + \frac{\mu_{d_2}(X^*)}{d_2} + \ldots + \frac{\mu_{d_m}(X^*)}{d_m}.$$

Особенностью этого нечеткого множества является то, что универсальным множеством для него является терм-множество выходной переменной y. Такие нечеткие множества называются нечеткими множествами второго порядка.

Для перехода от нечеткого множества, заданного на универсальном множестве нечетких термов $\{d_1, d_2, \ldots, d_m\}$ к нечеткому множеству на интервале $\left[\underline{y}, \bar{y}\right]$ необходимо:

1. "срезать" функции принадлежности $\mu_{d_j}(y)$ на уровне $\mu_{d_j}(X^*)$;

2. объединить (агрегировать) полученные нечеткие множества. Математически это записывается следующим образом:

$$\tilde{y} = \underset{j=1,m}{agg}\left(\int_{\underline{y}}^{\overline{y}} \min\left(\mu_{d_j}(X^*),\ \mu_{d_j}(y)\right) / y\right),$$

где agg - агрегирование нечетких множеств, которое наиболее часто реализуется операцией нахождения максимума.

Четкое значение выхода y, соответствующее входному вектору X^* определяется в результате деффаззификации нечеткого множества $\tilde{y}$. Наиболее часто применяется дефаззификация по методу центра тяжести:

$$y = \frac{\int_{\underline{y}}^{\overline{y}} y \cdot \mu_{\tilde{y}}(y)dy}{\int_{\underline{y}}^{\overline{y}} \mu_{\tilde{y}}(y)dy},$$

где $\int$ - здесь символ интеграла.

Пример. Пусть дана нечеткая база знаний, описывающая зависимость между возрастом водителя (x) и возможностью дорожно-транспортного происшествия (y):

Если x = Молодой, то y = Высокая;

Если x = Средний, то y = Низкая;

Если x = Очень старый, то y = Высокая.

Пусть функции принадлежностей термов имеют вид, показанный на рисунке 8.1.

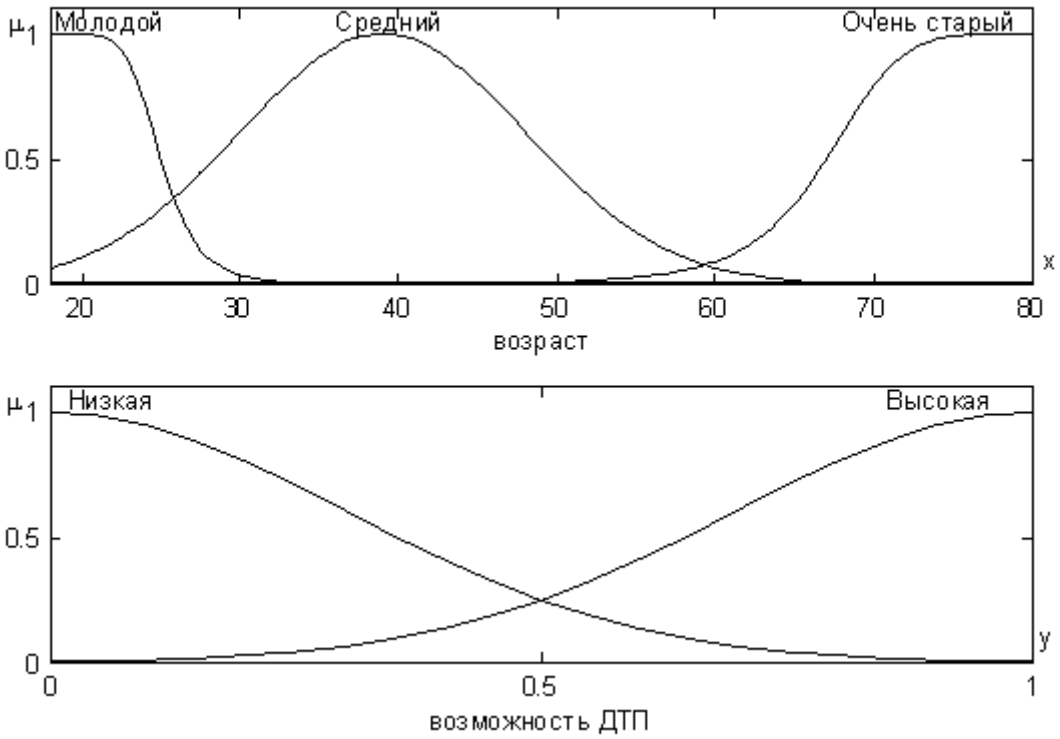


Рисунок 8.1 - Функции принадлежности термов

Тогда нечеткие отношения, соответствующие правилам базы знаний, будут такими, как на рисунке 8.2.

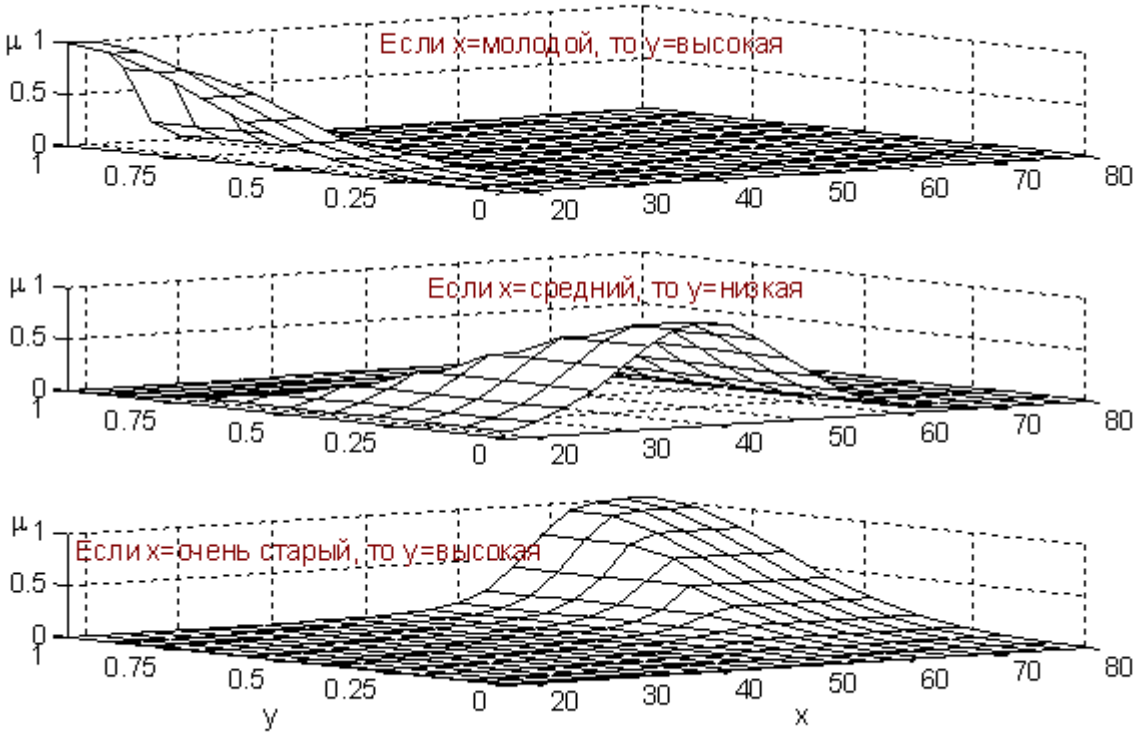


Рисунок 8.2 - Нечеткие отношения, соответствующие правилам базы знаний

По нечеткой базе знаний выполнить нечеткий логический вывод при значениях входной переменной $x = 28$ и $x =$"старый":

Выполнение нечеткого логического вывода при значениях входной переменной $x = 28$ и $x =$"старый" показано на рисунках 8.3 и 8.4.

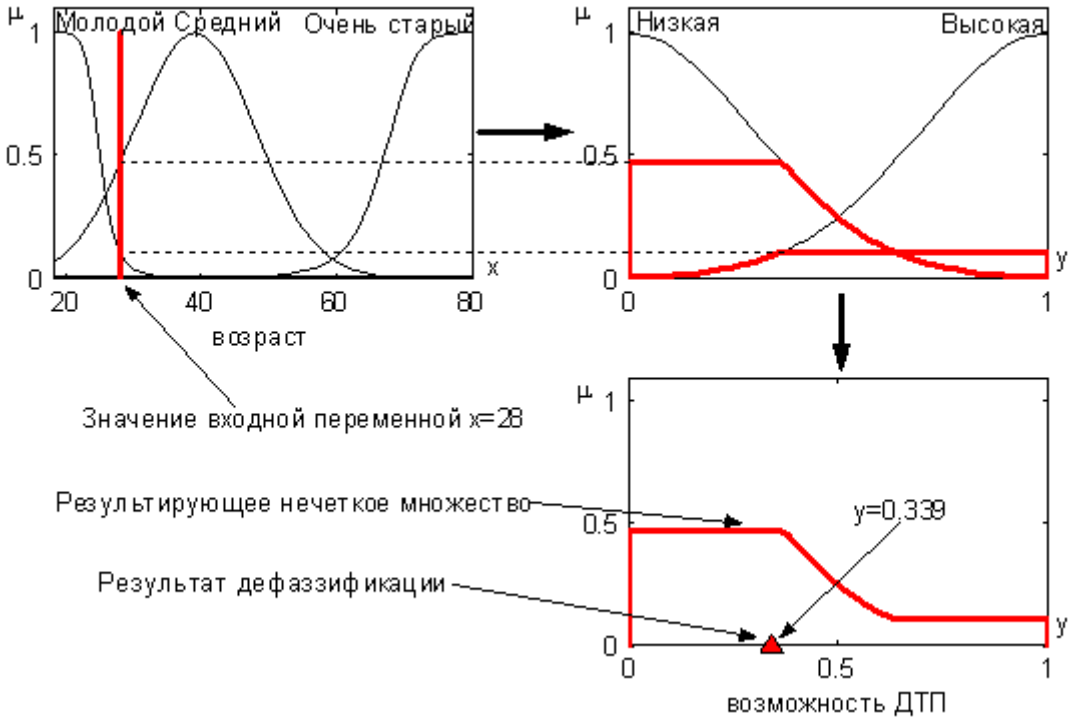


Рисунок 8.3 - Нечеткий логический вывод Мамдани при четком значении входной переменной

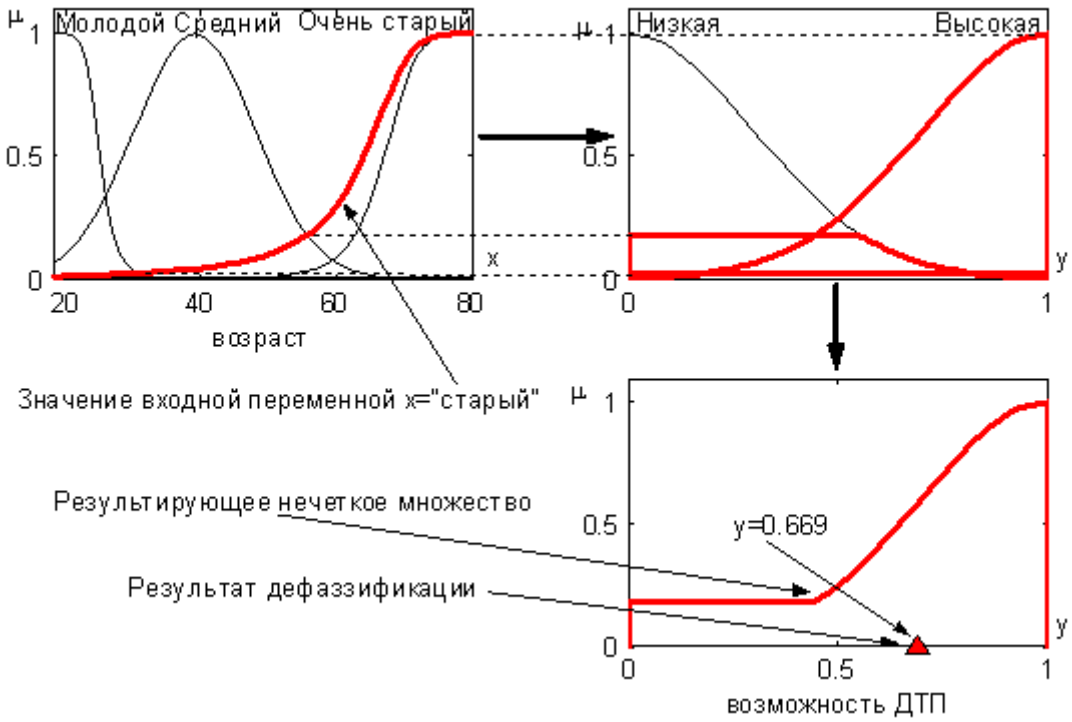


Рисунок 8.4 - Нечеткий логический вывод Мамдани при нечетком значении входной переменной

Операция агрегирования осуществлялась нахождением максимума. Дефаззификация проводилась по методу центра тяжести. На рисунке 8.5 показана зависимость "вход-выход", соответствующая нечеткой базе знаний. Участки графика, соответствующие первому, второму и третьему правилу базы знаний обозначены на рисунке #1, #2 и #3.

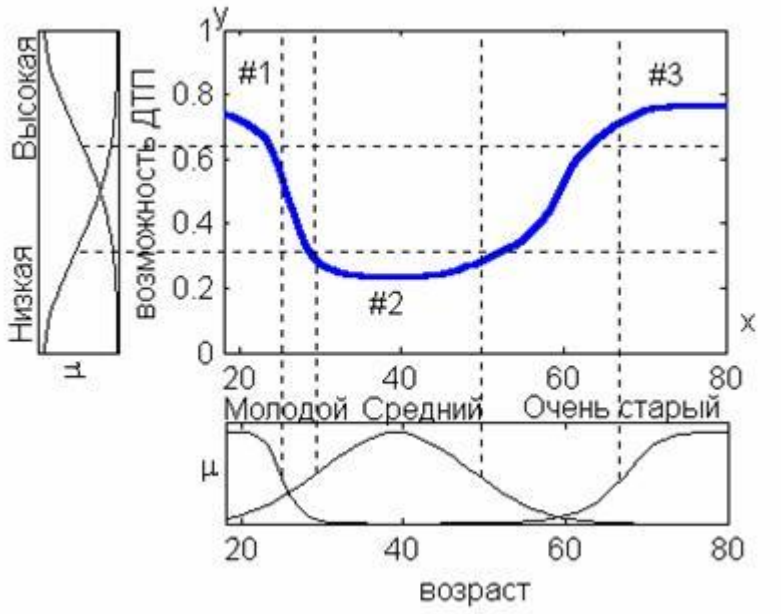


Рисунок 8.5 - Зависимость "вход-выход" для нечеткой базы знаний

8.2. Нечеткий логический вывод Сугено

Нечеткий логический вывод по алгоритму Сугено (иногда говорят алгоритм Такаги-Сугено) выполняется по нечеткой базе знаний:

$$\bigcup_{p=1}^{k_j}\left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \ \text{ с весом } w_{jp}\right) \to y = b_{j,0} + b_{j,1}\cdot x_1 + b_{j,2}\cdot x_2 + \ldots + b_{j,n}\cdot x_n, \quad j = \overline{1,m},$$

где $b_{j,i}$ - некоторые числа.

База знаний Сугено аналогична базе знаний Мамдани за исключением заключений правил d_j, которые задаются не нечеткими термами, а линейной функцией от входов: $d_j = b_{j,0} + \sum_{i=\overline{1,n}} b_{j,i}\cdot x_i$. Правила в базе знаний Сугено являются своего рода переключателями с одного линейного закона "входы - выход" на другой, тоже линейный. Границы подобластей размытые, следовательно, одновременно могут выполняться несколько линейных законов, но с различными степенями. Степени принадлежности входного вектора $X^* = \left(x_1^*, x_2^*, \ldots x_n^*\right)$ к значениям $d_j = b_{j,0} + \sum_{i=\overline{1,n}} b_{j,i}\cdot x_i$ рассчитывается следующим образом:

$$\mu_{d_j}\left(X^*\right) = \bigvee_{p=\overline{1,k_j}} w_{jp}\cdot \bigwedge_{i=\overline{1,n}}\left[\mu_{jp}\left(x_i^*\right)\right],\ j=\overline{1,m},$$

где $\vee(\wedge)$ - операция из s-нормы (t-нормы), т.е. из множества реализаций логической операций ИЛИ (И). В нечетком логическом выводе Сугено наиболее часто используются следующие реализации треугольных норм: вероятностное ИЛИ как s-норма и произведение как t-норма.

В результате получаем такое нечеткое множество $\tilde{y}$, соответствующее входному вектору X^*:

$$\tilde{y} = \frac{\mu_{d_1}(X^*)}{d_1} + \frac{\mu_{d_2}(X^*)}{d_2} + \ldots + \frac{\mu_{d_m}(X^*)}{d_m}.$$

Обратим внимание, что в отличие от результата вывода Мамдани, приведенное выше нечеткое множество является обычным нечетким множеством первого порядка. Оно задано на множестве четких чисел. Результирующее значение выхода y определяется как суперпозиция линейных зависимостей, выполняемых в данной точке X^* $n-$мерного факторного пространства. Для этого дефаззифицируют нечеткое множество $\tilde{y}$, находя взвешенное среднее $y = \frac{\sum_{j=1,m}\mu_{d_j}(X^*)\cdot d_j}{\sum_{j=1,m}\mu_{d_j}(X^*)}$ или взвешенную сумму $y = \sum_{j=1,m}\mu_{d_j}(X^*)\cdot d_j$.

Пример. Известна нечеткая база знаний:

Если x=низкий, то $y = 3x$;

Если x=высокий, то $y = 3 - x$.

Функции принадлежности термов заданы следующими выражениям:

$\mu_{\text{"низкий"}}(x) = \exp\left(-\frac{x^2}{0.18}\right)$ и $\mu_{\text{"высокий"}}(x) = \exp\left(-\frac{(x-1)^2}{0.18}\right)$, $x \in [0,1]$.

Необходимо выполнить нечеткий логический вывод при значении входной переменной $x = 0.4$.

Выполнение нечеткого логического вывода показано на рисунке 8.6.

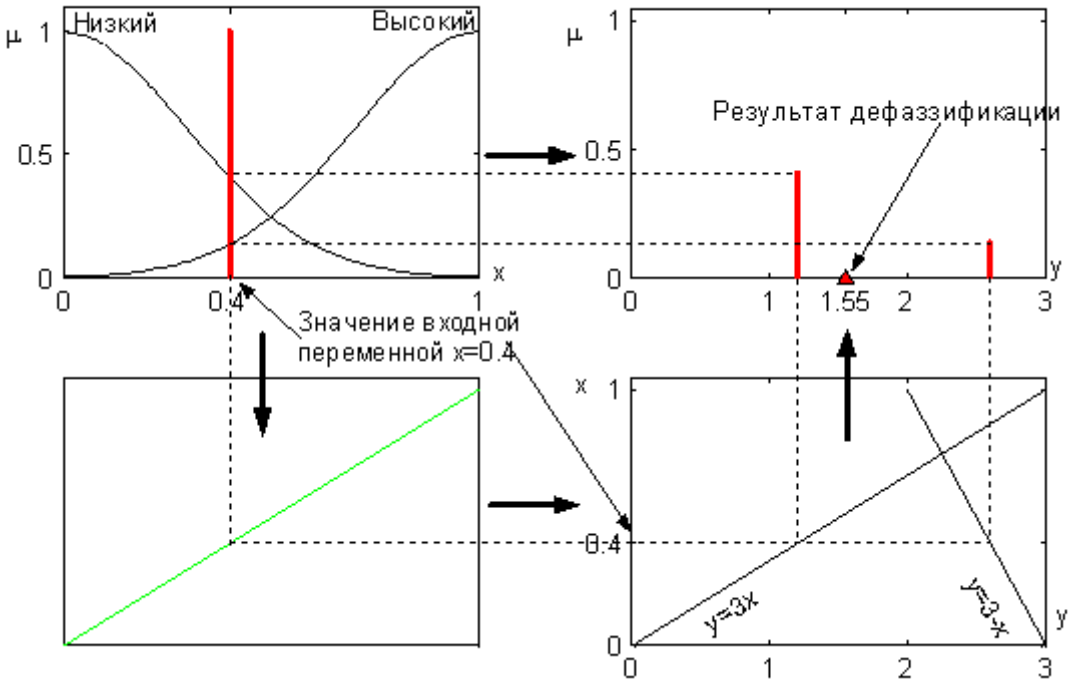


Рисунок 8.6 - Выполнение нечеткого логического вывода Сугено

Дефаззификация проводилась по методу центра тяжести (взвешенного среднего). На рисунке 8.7 показана зависимость "вход-выход" для приведенной выше нечеткой базы знаний. Участки графика, соответствующие первому и второму правилу базы знаний обозначены на рисунке #1 и #2.

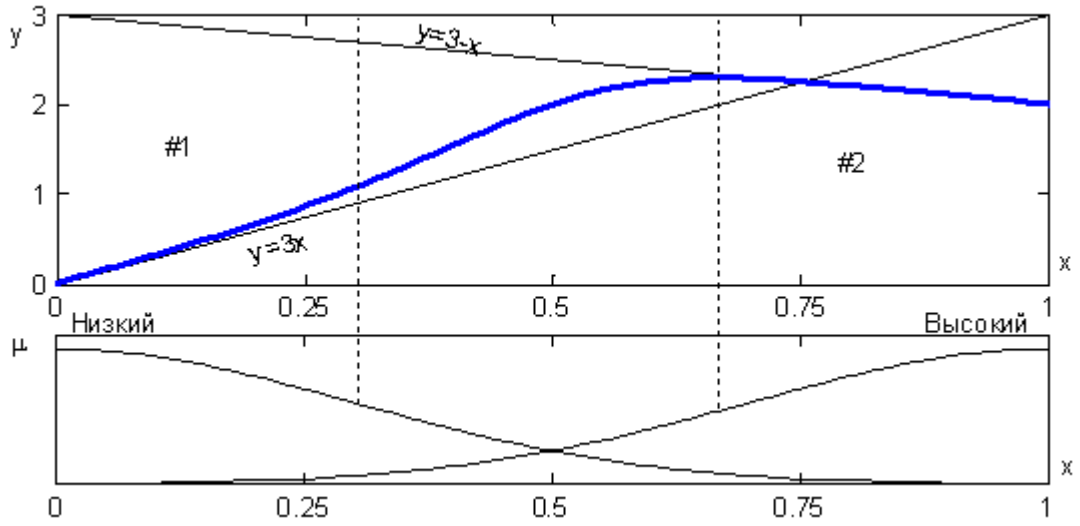


Рисунок 8.7 - Зависимость "вход-выход" для нечеткой базы знаний

8.3. Пример осуществления нечеткого логического вывода

Пусть у нас есть некоторая система, например, реактор, описываемая тремя параметрами: температура, давление и расход рабочего вещества. Все показатели измеримы, и множество возможных значений известно. Также из опыта работы с системой известны некоторые правила, связывающие значения этих параметров. Предположим, что сломался датчик, измеряющий значение одного из параметров системы, но знать его показания необходимо хотя бы приблизительно. Тогда встает задача об отыскании этого неизвестного значения (пусть это будет давление) при известных показателях двух других параметров (температуры и расхода) и связи этих величин в виде следующих правил:

- если Температура низкая и Расход малый, то Давление низкое;

- если Температура средняя, то Давление среднее;

- если Температура высокая или Расход большой, то Давление высокое.

В нашем случае Температура, Давление и Расход — лингвистические переменные. Опишем каждую из них.

Температура. Универсум (множество возможных значений) — отрезок [0,150]. Начальное множество термов {Высокая, Средняя, Низкая} (рисунок 8.8).

Рисунок 8.8 – Функции принадлежности термов лингвистической переменной «Температура»

Давление. Универсум — отрезок [0,100]. Начальное множество термов {Высокое, Среднее, Низкое} (рисунок 8.9).

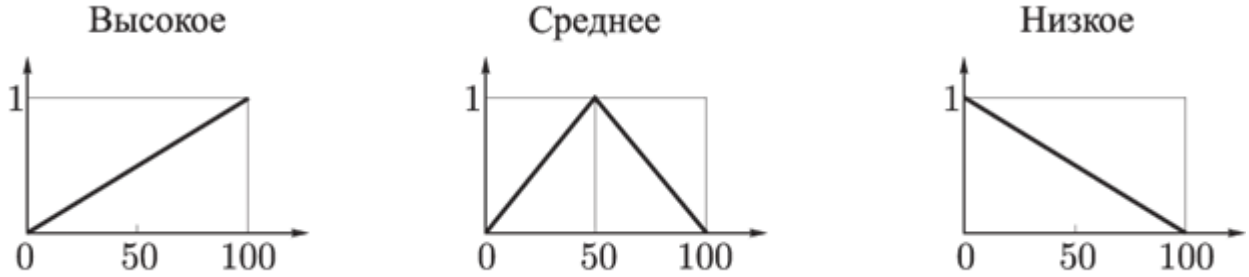


Рисунок 8.9 - Функции принадлежности термов лингвистической переменной «Давление»

Расход. Универсум — отрезок [0,8]. Начальное множество термов {Большой, Средний, Малый} (рисунок 8.10).

Рисунок 8.10 - Функции принадлежности термов лингвистической переменной «Расход»

Пусть известны значения Температура 85 и Расход 3,5 . Произведем расчет значения давления.

Последовательно рассмотрим этапы нечеткого вывода:

Сначала по заданным значениям входных параметров найдем степени уверенности простейших утверждений вида "Лингв. переменная A есть Терм Лингв. переменной A ". Этот этап называется фаззификацией, т.е. переходом от заданных четких значений к степеням уверенности. Получаем следующие степени уверенности:

- Температура Высокая — 0,7;
- Температура Средняя — 1;
- Температура Низкая — 0,3;
- Расход Большой — 0;
- Расход Средний — 0,75;
- Расход Малый — 0,25.

Затем вычислим степени уверенности посылок правил:

- Температура низкая и Расход малый: min(Темп. Низкая, Расход Малый)= min(0.3, 0.25)=0.25;
- Температура Средняя: 1;
- Температура Высокая или Расход Большой: max(Темп. Высокая, Расход Большой)= max(0.7,0)=0,7.

Следует отметить также тот факт, что с помощью преобразований нечетких множеств любое правило, содержащее в левой части как конъюнкции, так и дизъюнкции, можно привести к системе правил, в левой части каждого будут либо только конъюнкции, либо только дизъюнкции. Таким образом, не уменьшая общности, можно рассматривать правила, содержащие в левой части либо только конъюнкции, либо только дизъюнкции.

Каждое из правил представляет из себя нечеткую импликацию. Степень уверенности посылки мы вычислили, а степень уверенности заключения задается функцией принадлежности соответствующего терма. Поэтому, используя один из способов построения нечеткой импликации, мы получим новую нечеткую переменную, соответствующую степени уверенности в значении выходных данных при применении к заданным входным соответствующего правила. Используя определение нечеткой импликации как минимума левой и правой частей (определение Mamdani), имеем (рисунок 8.11).

Рисунок 8.11 - Правила

Теперь необходимо объединить результаты применения всех правил.

Этот этап называется аккумуляцией. Один из основных способов аккумуляции — построение максимума полученных функций принадлежности (рисунок 8.12).

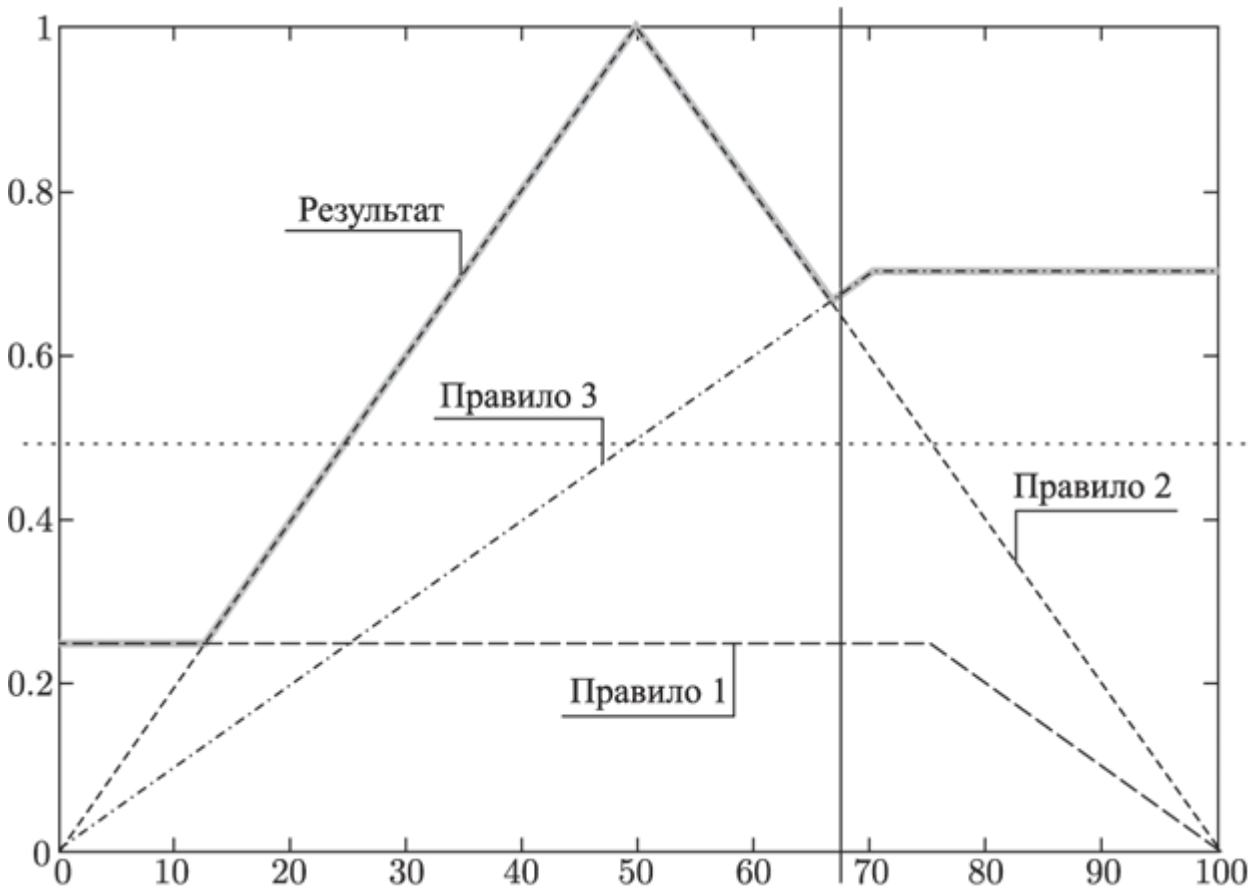


Рисунок 8.12 - Максимум полученных функций принадлежности

Полученную функцию принадлежности уже можно считать результатом. Это новый терм выходной переменной Давление. Его функция принадлежности говорит о степени уверенности в значении давления при заданных значениях входных параметров и использовании правил, определяющих соотношение входных и выходных переменных. Но обычно все-таки необходимо какое-то конкретное числовое значение. Для его получения используется этап дефаззификации, т.е. получения конкретного значения из универсума по заданной на нем функции принадлежности.

Существует множество методов дефаззификации, но в нашем случае достаточно метода первого максимума. Применяя его к полученной функции принадлежности, получаем, что значение давления — 50. Таким образом, если мы знаем, что температура равна 85, а расход рабочего вещества — 3,5, то можем сделать вывод, что давление в реакторе равно примерно 50.

Вопросы для повторения и закрепления материала

1. Назовите этапы осуществления нечеткого логического вывода?
2. Назовите основные этапы алгоритма Мамдани?
3. Назовите основные этапы алгоритма Сугено?
4. В чем заключается различие алгоритмов Мамдани и Сугено?
5. Назовите достоинства и недостатки алгоритмов Мамдани и Сугено?

6. Что делается на этапе фаззификации?

Задания для самостоятельной работы

1. Проведите анализ литературы на предмет областей использования нечетких экспертных систем.

Тема 9. Основы искусственных нейронных сетей

Цель: ознакомиться с основами искусственных нейронных сетей.

Задачи:

1. Ознакомиться с предпосылками создания искусственных нейронных сетей.
2. Рассмотреть структуру биологического нейрона.
3. Изучить структуру и свойства искусственных нейронных сетей.
4. Ознакомиться с классификацией нейронных сетей
5. Рассмотреть общие принципы обучения нейронных сетей.

9.1. Введение в искусственные нейронные сети

Искусственные нейронные сети (ИНС или НС) строятся по принципам организации и функционирования их биологических аналогов. Они способны решать широкий круг задач распознавания образов, идентификации, прогнозирования, оптимизации, управления сложными объектами. Дальнейшее повышение производительности компьютеров все в большой мере связывают с ИНС, в частности, с нейрокомпьютерами (НК), основу которых составляет искусственная нейронная сеть.

Термин «нейронные сети» сформировался к середине 50-х годов XX века. Основные результаты в этой области связаны с именами У. Маккалоха, Д. Хебба, Ф. Розенблатта, М. Минского, Дж. Хопфилда. Приведем краткую историческую справку:

- 1943 г. У. Маккалох (W. McCulloch) и У. Питтс (W. Pitts) предложили модель нейрона и сформулировали основные положения теории функционирования головного мозга.
- 1949 г. Д. Хебб (D. Hebb) высказал идеи о характере соединений нейронов мозга и их взаимодействии (клеточные ансамбли, синаптическая пластичность). Впервые предложил правила обучения нейронной сети.
- 1957 г. Ф. Розенблатт (F. Rosenblatt) разработал принципы организации и функционирования персептронов, предложил вариант технической реализации первого в мире нейрокомпьютера Mark.
- 1959 г. Д. Хьюбел (D, Hubel) и Т. Визель (T. Wiesel) показали распределенный и параллельный характер хранения и обработки информации в биологических нейронных сетях.

- 1960-1968 гг. Активные исследования в области искусственных нейронных сетей, например, АДАЛИНА и МАДАЛИНА В. Уидроу (W. Widrow) (1960-1962 гг.), ассоциативные матрицы К. Штайнбуха (К. Steinbuch) (1961 г.).

- 1969 г. Публикация книги М. Минского (M. Minsky) и С. Пей-перта (S. Papert) «Персептроны», в которой доказывается принципиальная ограниченность возможностей персептронов. Угасание интереса к искусственным нейронным сетям.

- 1970-1976 гг. Активные разработки в области персептронов в СССР (основные заказчики - военные ведомства).

- Конец 1970-х гг. Возобновление интереса к искусственным нейронным сетям как следствие накопления новых знаний о деятельности мозга, а также значительного прогресса в области микроэлектроники и компьютерной техники.

- 1982-1985 гг. Дж. Хопфилд (J. Hopfield) предложил семейство оптимизирующих нейронных сетей, моделирующих ассоциативную память.

- 1985 г. Появление первых коммерческих нейрокомпьютеров, например, Mark III фирмы TRW (США).

- 1987 г. Начало широкомасштабного финансирования разработок в области ИНС и НК в США, Японии и Западной Европе (японская программа «Human Frontiers» и европейская программа «Basic Research in Adaptive Intelligence and Neurocomputing»).

- 1989 г. Разработки и исследования в области ИНС и НК ведутся практически всеми крупными электротехническими фирмами. Нейрокомпьютеры становятся одним из самых динамичных секторов рынка (за два года объем продаж вырос в пять раз). Агентством DARPA (Defence Advanced Research Projects Agency) министерства обороны США начато финансирование программы по созданию сверхбыстродействующих образцов НК для разнообразных применений.

- 1990 г. Активизация советских исследовательских организаций в области ИНС и НК (Институт кибернетики им. Глушкова в Киеве, Институт многопроцессорных вычислительных систем в Таганроге, Институт нейрокибернетики в Ростове-на-Дону). Общее число фирм, специализирующихся в области ИНС и НК, достигает трехсот.

- 1991 г. Годовой объем продаж на рынке ИНС и НК приблизился к 140 млн. долларам. Создаются центры нейрокомпьютеров в Москве, Киеве, Минске, Новосибирске, С.-Петербурге.

- 1992 г. Работы в области ИНС находятся стадии интенсивного развития. Ежегодно проводится десятки международных конференций и форумов по нейронным сетям, число специализированных периодических научных изданий по указанной тематике достигло двух десятков наименований.

- 1996 г. Число международных конференций по ИНС и НК достигло ста.

- 1997 г. Годовой объем продаж на рынке ИНС и НК превысил 2 млрд. долларов, а ежегодный прирост составил 50%.

- 2000 г. Переход на субмикронные и нанотехнологии, а также успехи молекулярной и биомолекулярной технологии приводят к принципиально новым архитектурным и технологическим решениям по созданию нейрокомпьютеров.

Проблемы, решаемые искусственными нейронными сетями:

- **Классификация образов**. Задача состоит в указании принадлежности входного образа, представленного вектором признаков, 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 также как классификация образов без учителя, отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает 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 2), (X 2, Y 2),..., (X N, Y N)), которая генерируется неизвестной функцией, искаженной шумом. Задача аппроксимации состоит в нахождении оценки этой функции.

- **Предсказание/прогноз**. Пусть заданы N дискретных отсчетов { y (t 1), y (t 2),..., y(t n)} в последовательные моменты времени t 1, t 2,..., t n. Задача состоит в предсказании значения y(t n +1) в момент t n +1. Прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике.

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

- **Память, адресуемая по содержанию.** В модели вычислений фон Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Память, адресуемая по содержанию, или ассоциативная память, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании перспективных информационно-вычислительных систем.

- **Управление**. Рассмотрим динамическую систему, заданную совокупностью {u(t), y(t)}, где u(t) является входным управляющим воздействием, а y(t) - выходом системы в момент времени f. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия u(f), при котором система следует по желаемой траектории, диктуемой эталонной моделью.

Каким образом нейронная сеть решает все эти, часто неформализуемые или трудно формализуемые задачи? Как известно, для решения таких задач традиционно применяются два основных подхода. Первый, основанный на правилах (rule-based), характерен для экспертных систем. Он базируется на описании предметной области в виде набора правил (аксиом) «если..., то...» и правил вывода. Искомое знание представляется в этом случае теоремой, истинность которой доказывается посредством построения цепочки вывода. При этом подходе, однако, необходимо заранее знать весь набор закономерностей, описывающих предметную область. При использовании другого подхода, основанного на примерах (case-based), надо лишь иметь достаточное количество примеров для настройки адаптивной системы с заданной степенью достоверности. Нейронные сети представляют собой классический пример такого подхода.

9.2. Биологический прототип нейрона

Развитие ИНС основывается на биологии. То есть, рассматривая сетевые конфигурации и алгоритмы, исследователи применяют термины, заимствованные из принципов организации мозговой деятельности. Но на этом аналогия заканчивается. Наши знания о работе мозга столь ограничены, что мало бы нашлось точно доказанных закономерностей для тех, кто пожелал бы руководствоваться ими. Поэтому разработчикам сетей приходится выходить за пределы современных биологических знаний в поисках структур, способных выполнять полезные функции. Во многих случаях это приводит к необходимости отказа от биологического правдоподобия, мозг становится просто метафорой, и создаются сети, невозможные в живой материи или требующие неправдоподобно больших допущений об анатомии и функционировании мозга.

Несмотря на то, что связь с биологией слаба и зачастую несущественна, ИНС продолжают сравнивать с мозгом. Их функционирование часто имеет внешнее сходство с человеческим познанием, поэтому трудно избежать этой аналогии.

Мозг представляет собой нейронную сеть содержащую узлы — нейроны (рисунок 9.1) и их соединения — синапсические связи. **Нейрон** (нервная клетка) - особая биологическая клетка, которая обрабатывает информацию. Он состоит из **тела** (cell body), или **сомы** (soma), и отростков нервных волокон двух типов - **дендритов** (dendrites), по которым

принимаются импульсы, и единственного **аксона** (axon), по которому нейрон может передавать импульс.

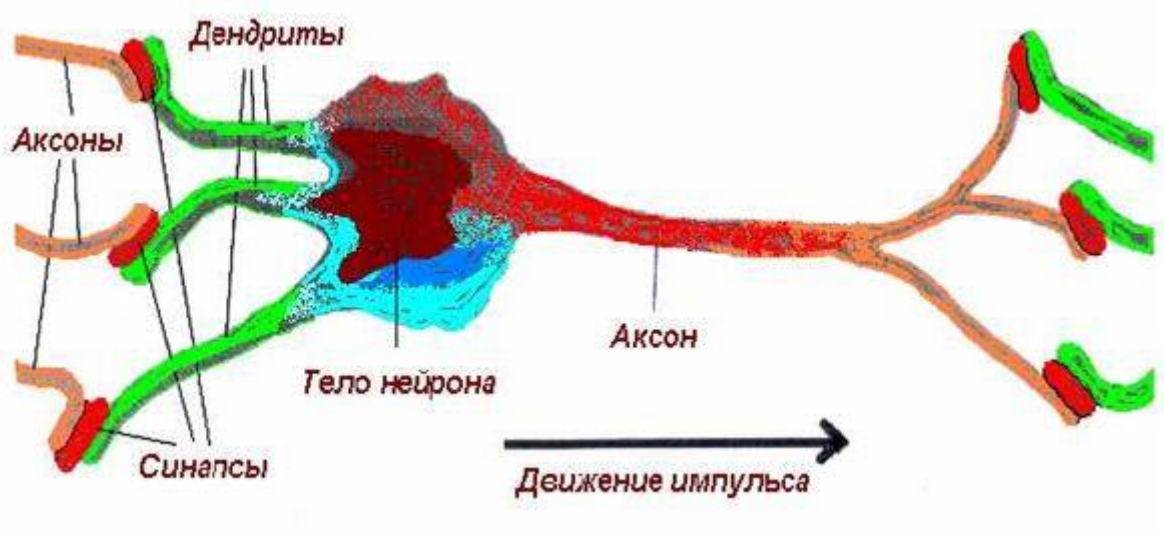


Рисунок 9.1 – Биологический нейрон

Тело нейрона является средством выполнения пороговой функции активации над сигналами, пришедшими по входам нейрона – дендритам (нейрон мозга содержит до 10 тыс. дендритов.). Кора головного мозга человека содержит около 10^{11} нейронов и представляет собой протяженную поверхность толщиной от 2 до 3 мм с площадью около 2200 см^2. Каждый нейрон связан с 10^3-10^4 другими нейронами. В целом мозг человека содержит приблизительно от 10^{14} до 10^{15} взаимосвязей. Для того чтобы перейти в возбужденное состояние, в теле нейрона выполняются около 240 химических реакций. Величина возбуждения нейрона с помощью ветвящегося аксона передается на дендриты других нейронов.

Важными управляющими элементами связей нейронов являются синапсы. Синапс аналогичен переменному сопротивлению, определяющему вес связи (вес дендрита). Этот вес является коэффициентом, с которым дендрит принимает возбуждение нейрона, связанного с данным. Поэтому связи между нейронами называются синапсическими.

Изменение весов синапсических связей позволяет регулировать направления и пути распространения возбуждений в нейронной сети, т.е. в множестве взаимосвязанных нейронов. Так в этой сети образуются связи вида "если ... то".

Поэтому настройка весов синапсических связей является основной задачей обучения нейронной сети, когда возбуждение некоторого множества нейронов обязательно должно приводить к возбуждению определенного нейрона. Это важнейший элемент выполнения мозгом функций распознавания, управления и принятия решений. Математическая, абстрактная, модель нейрона во взаимодействии с другими нейронами сети представлена на рисунке 9.2.

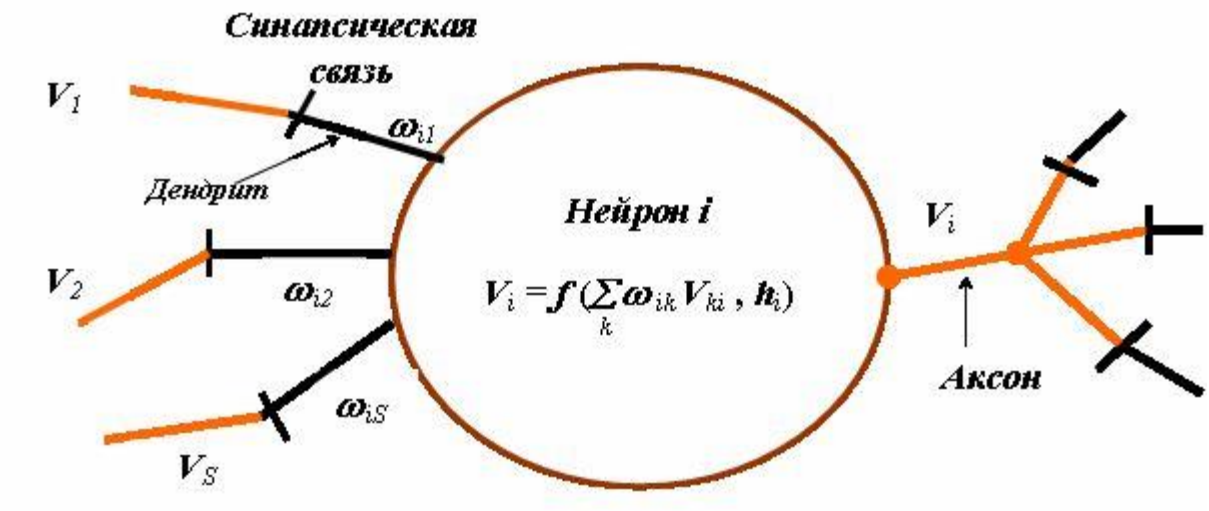


Рисунок 9.2 – Математическая модель нейрона

Здесь V_k — импульсы возбуждения, выработанные другими нейронами и поступившие на дендриты нейрона i, ω_{ik} — веса дендритов, h_i — пороги. В свою очередь, выработанный импульс V_i также направляется на дендриты нейронов, с которыми связан нейрон i с помощью ветвящегося аксона. Значение импульса возбуждения V_i находится, как говорилось ранее, в результате счета функции активации, возможный вид которой приведен на рисунке.

Значения весов синапсических связей ω_i и значения порогов h_i могут регулироваться. Такое регулирование, во многих вариантах реализованное в разных моделях, и определяет возможность обучения и самообучения сети. Оно задает направление распространения возбуждений через сеть, простейшим образом формируя связи "посылка — следствие".

9.3. Структура и свойства искусственного нейрона

Искусственный нейрон имитирует в первом приближении свойства биологического нейрона. На вход искусственного нейрона поступает некоторое множество сигналов, каждый из которых является выходом другого нейрона. Каждый вход умножается на соответствующий вес, аналогичный синаптической силе, и все произведения суммируются, определяя уровень активации нейрона.

На рисунке 9.3 показана структура искусственного нейрона. Он состоит из элементов трех типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапсы осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи, (вес синапса). Сумматор выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента - выхода

сумматора. Эта функция называется функцией активации или передаточной функцией нейрона.

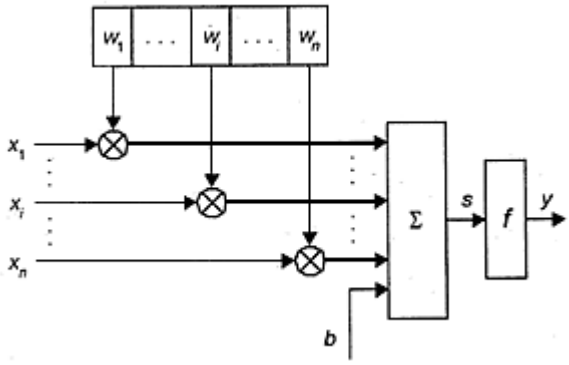


Рисунок 9.3 - Структура искусственного нейрона

Нейрон в целом реализует скалярную функцию векторного аргумента. Математическая модель нейрона:

$$S = \sum_{i=1}^{n} w_i x_i + b$$

$$y = f(s)$$

где w_i, - вес (weight) синапса, i = 1...n; b - значение смещения (bias); s - результат суммирования (sum); x - компонент входного вектора (входной сигнал), x_i = 1... n; y - выходной сигнал нейрона; n - число входов нейрона; f - нелинейное преобразование (функция активации).

В общем случае входной сигнал, весовые коэффициенты и смещение могут принимать действительные значения, а во многих практических задачах - лишь некоторые фиксированные значения. Выход (y) определяется видом функции активации и может быть как действительным, так и целым.

Синаптические связи с положительными весами называют возбуждающими, с отрицательными весами - тормозящими.

На рисунке 9.4 представлена модель, реализующая структуру, представленную на рисунке 9.3. Множество входных сигналов, обозначенных $x_1, x_2, \ldots, x_n$, поступает на искусственный нейрон. Эти входные сигналы, в совокупности обозначаемые вектором X, соответствуют сигналам, приходящим в синапсы биологического нейрона. Каждый сигнал умножается на соответствующий вес $w_1, w_2, \ldots, w_n$, и поступает на суммирующий блок, обозначенный Σ. Каждый вес соответствует "силе" одной биологической синаптической связи. (Множество весов в совокупности обозначается вектором W.) Суммирующий блок, соответствующий телу биологического элемента, складывает взвешенные входы алгебраически, создавая выход, который мы будем называть NET. В векторных обозначениях это может быть компактно записано следующим образом: $NET = XW$.

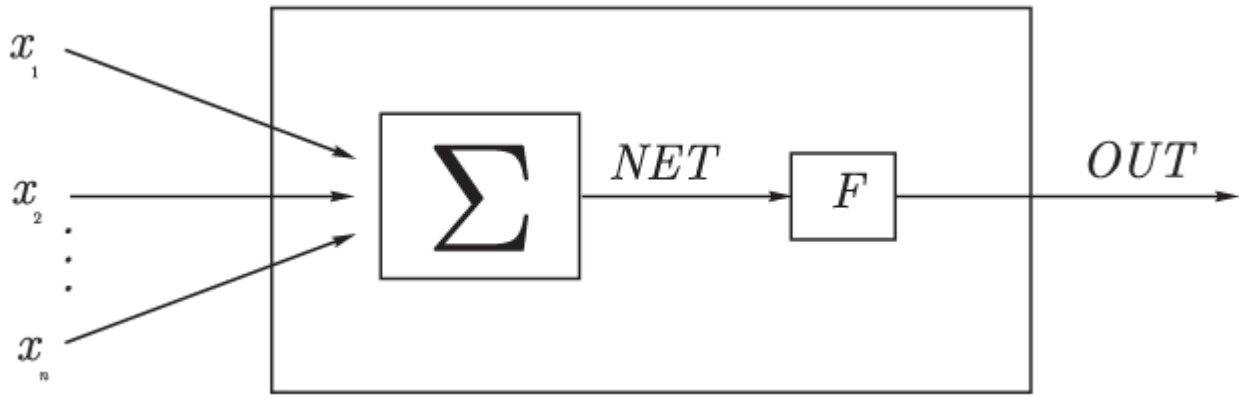


Рисунок 9.4 – Модель искусственного нейрона

Сигнал NET далее, как правило, преобразуется активационной функцией F и дает выходной нейронный сигнал OUT. Активационная функция может быть обычной линейной функцией

$$OUT = F(NET),$$

где F — константа, пороговой функцией

$$OUT = \begin{cases} 1, & \text{если } NET > T; \\ 0, & \text{если } NET \leqslant T \end{cases}$$

где T — некоторая постоянная пороговая величина, или же функция, более точно моделирующая нелинейную передаточную характеристику биологического нейрона и предоставляющей нейронной сети большие возможности.

На рисунке 9.4 блок, обозначенный F, принимает сигнал NET и выдает сигнал OUT. Если блок F сужает диапазон изменения величины NET так, что при любых значениях NET значения OUT принадлежат некоторому конечному интервалу, то F называется "сжимающей" функцией. В качестве "сжимающей" функции часто используется логистическая или "сигмоидальная" (S-образная) функция, показанная на рисунке 9.5. Эта функция математически выражается как $F(x) = 1/(1 + e^{-x})$. Таким образом,

$$OUT = \frac{1}{1 + e^{-NET}}.$$

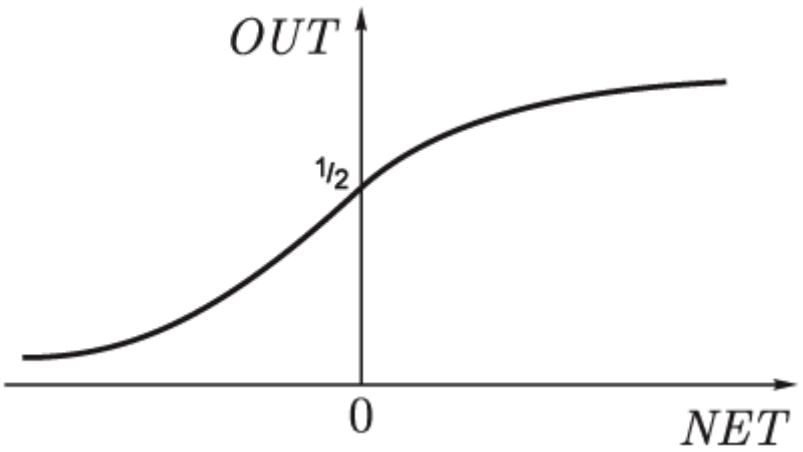


Рисунок 9.5 – Сжимающая S-образная (сигмоидальная) функция

По аналогии с электронными системами активационную функцию можно считать нелинейной усилительной характеристикой искусственного нейрона. Коэффициент усиления вычисляется как отношение приращения величины OUT к вызвавшему его небольшому приращению величины NET. Он выражается наклоном кривой при определенном уровне возбуждения и изменяется от малых значений при больших отрицательных возбуждениях (кривая почти горизонтальна) до максимального значения при нулевом возбуждении и снова уменьшается, когда возбуждение становится большим положительным. С. Гроссберг (1973) обнаружил, что подобная нелинейная характеристика решает поставленную им дилемму шумового насыщения. Каким образом одна и та же сеть может обрабатывать как слабые, так и сильные сигналы? Слабые сигналы нуждаются в большом сетевом усилении, чтобы дать пригодный к использованию выходной сигнал. Однако усилительные каскады с большими коэффициентами усиления могут привести к насыщению выхода шумами усилителей (случайными флуктуациями), которые присутствуют в любой физически реализованной сети. Сильные входные сигналы, в свою очередь, также будут приводить к насыщению усилительных каскадов, исключая возможность полезного использования выхода. Центральная область логистической функции, имеющая большой коэффициент усиления, решает проблему обработки слабых сигналов, в то время как области с падающим усилением на положительном и отрицательном концах подходят для больших возбуждений. Таким образом, нейрон функционирует с большим усилением в широком диапазоне уровня входного сигнала

$$OUT = \frac{1}{1 + e^{-NET}} = F(NET).$$

Другой широко используемой активационной функцией является гиперболический тангенс. По форме она сходна с логистической функцией и часто используется биологами в качестве математической модели активации

нервной клетки. В качестве активационной функции искусственной нейронной сети она записывается следующим образом:

$$OUT = \mathrm{th}(x).$$

Подобно логистической функции гиперболический тангенс является S-образной функцией, но он симметричен относительно начала координат, и в точке $NET = 0$ значение выходного сигнала OUT равно нулю (рисунок 9.6). В отличие от логистической функции, гиперболический тангенс принимает значения различных знаков, и это его свойство применяется для целого ряда сетей.

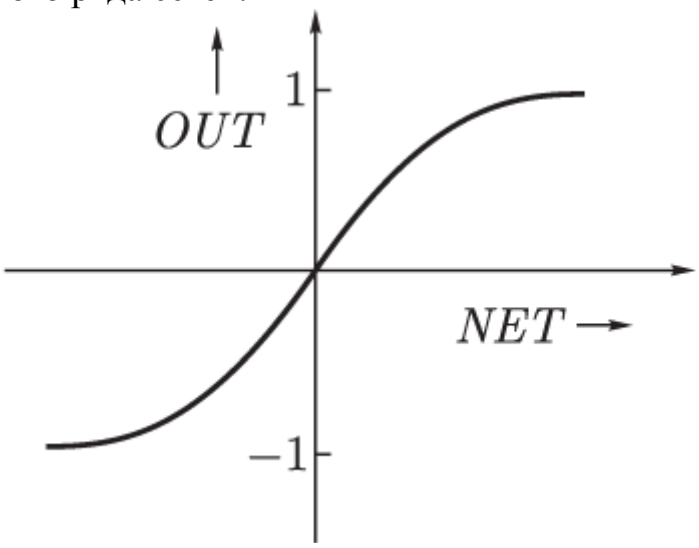


Рисунок 9.6 - Сжимающая S-образная (гиперболический тангенс) функция

Примеры активационных функций представлены в таблице 9.1. и на рисунке 9.7.

Таблица 9.1 – Функции активации нейронов

Функции активации нейронов

Название	Формула	Область значений
Линейная	$f(s)=k\,s$	$(-\infty, \infty)$
Полулинейная	$f(s)=\begin{cases} k\,s, & s>0, \\ 0, & s\le 0 \end{cases}$	$(0, \infty)$
Логистическая (сигмоидальная)	$f(s)=\dfrac{1}{1+e^{-as}}$	(0, 1)
Гиперболический тангенс (сигмоидальная)	$f(s)=\dfrac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$	(–1, 1)
Экспоненциальная	$f(s)=e^{-as}$	$(0, \infty)$
Синусоидальная	$f(s)=\sin(s)$	(–1, 1)
Сигмоидальная (рациональная)	$f(s)=\dfrac{s}{a+\lvert s\rvert}$	(–1, 1)
Шаговая (линейная с насыщением)	$f(s)=\begin{cases} -1, & s\le -1, \\ s, & -1<s<1, \\ 1, & s\ge 1 \end{cases}$	(–1, 1)
Пороговая	$f(s)=\begin{cases} 0, & s<0, \\ 1, & s\ge 0 \end{cases}$	(0, 1)
Модульная	$f(s)=\lvert s\rvert$	$(0, \infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases} 1, & s>0, \\ -1, & s\le 0 \end{cases}$	(–1, 1)
Квадратичная	$f(s)=s^2$	$(0, \infty)$

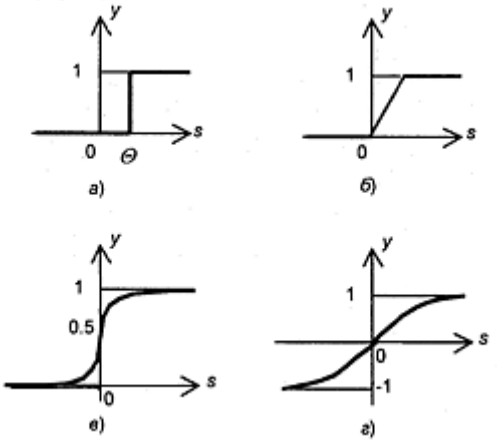


Рисунок 9.7 - Примеры активационных функций а - функция единичного скачка; б - линейный порог (гистерезис); в - сигмоид (логистическая функция); г - сигмоид (гиперболический тангенс)

Рассмотренная простая модель искусственного нейрона игнорирует многие свойства своего биологического двойника. Например, она не принимает во внимание задержки во времени, которые воздействуют на динамику системы. Входные сигналы сразу же порождают выходной сигнал. И, что более важно, она не учитывает воздействий функции частотной модуляции или синхронизирующей функции биологического нейрона, которые ряд исследователей считают решающими в нервной деятельности естественного мозга.

Несмотря на эти ограничения, сети, построенные из таких нейронов, обнаруживают свойства, сильно напоминающие биологическую систему.

9.4. Классификация нейронных сетей и их свойства

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

- **входные нейроны,** на которые подается вектор, кодирующий входное воздействие или образ внешней среды. В них обычно не осуществляется вычислительных процедур, а информация передается с входа на выход путем изменения их активации.

- **выходные нейроны,** выходные значения которых представляют выходы нейронной сети.

- **промежуточные нейроны,** составляющие основу нейронных сетей.

В большинстве нейронных моделей тип нейрона связан с его расположением в сети. Если нейрон имеет только выходные связи, то это входной нейрон, если наоборот - выходной нейрон. В процессе функционирования сети осуществляется преобразование входного вектора в выходной, некоторая переработка информации. Конкретный вид выполняемого сетью преобразования данных обусловливается не только характеристиками нейроподобных элементов, но и особенностями ее архитектуры, а именно топологией межнейронных связей, выбором определенных подмножеств нейроподобных элементов для ввода и вывода информации, способами обучения сети, наличием или отсутствием конкуренции между нейронами, направлением и способами управления и синхронизации передачи информации между нейронами.

С точки зрения топологии можно выделить три основных типа нейронных сетей:

- полносвязные (рисунок 9.8, а);
- многослойные или слоистые (рисунок. 9.8, б);
- слабосвязные (с локальными связями) (рисунок 9.8, в).

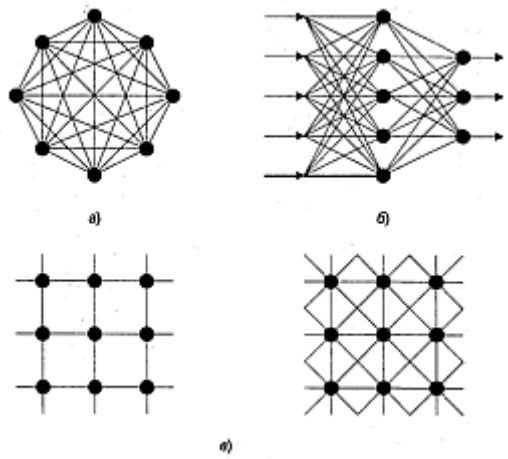


Рисунок 9.8 - Архитектуры нейронных сетей: а - полносвязная сеть, б - многослойная сеть с последовательными связями, в - слабосвязные сети

Полносвязные нейронные сети - каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

Многослойные нейронные сети - нейроны объединяются в 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 Q слоев, пронумерованных слева направо. 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 скрытых слоев. Связи от выходов нейронов некоторого слоя q к входам нейронов следующего слоя (q +1) называются последовательными.

В свою очередь, среди многослойных нейронных сетей выделяют следующие типы:

1) **Монотонные.** Это частный случай слоистых сетей с дополнительными условиями на связи и нейроны. Каждый слой кроме последнего (выходного) разбит на два блока: возбуждающий и тормозящий. Связи между блоками тоже разделяются на тормозящие и возбуждающие. Если от нейронов блока А к нейронам блока В ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока является монотонной неубывающей функцией любого выходного сигнала блока А. Если же эти связи только тормозящие, то любой выходной сигнал блока В является невозрастающей функцией любого выходного сигнала блока А. Для нейронов монотонных сетей необходима монотонная зависимость выходного сигнала нейрона от параметров входных сигналов.

2) **Сети без обратных связей.** 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 q-го слоя подастся на вход всех нейронов (q+1)-го слоя; однако возможен вариант соединения q-го слоя с произвольным (q+p)-м слоем.

3) **Сети с обратными связями.** В сетях с обратными связями информация с последующих слоев передается на предыдущие. Среди них, в свою очередь, выделяют следующие:

- слоисто-циклические, отличающиеся тем, что слои замкнуты в кольцо: последний слой передает свои выходные сигналы первому; все слои

равноправны и могут как получать входные сигналы, так и выдавать выходные;

- слоисто-полносвязанные состоят из слоев, каждый из которых представляет собой полносвязную сеть, а сигналы передаются как от слоя к слою, так и внутри слоя; в каждом слое цикл работы распадается на три части: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача к последующему слою;

- полносвязанно-слоистые, по своей структуре аналогичные слоисто-полносвязанным, но функционирующим по-другому: в них не разделяются фазы обмена внутри слоя и передачи следующему, на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих.

В качестве примера сетей с обратными связями на рисунке 9.9 представлены частично-рекуррентные сети Элмана и Жордана.

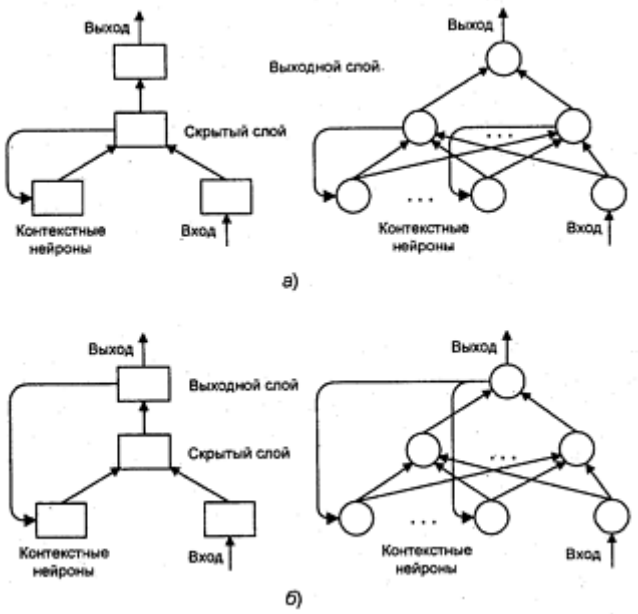


Рисунок 9.9 - Частично-рекуррентные сети: а - Элмана, б – Жордана

Известные нейронные сети можно разделить по типам структур нейронов на:

- гомогенная (однородная) сеть состоят из нейронов одного типа с единой функцией активации;

- гетерогенная сеть состоит из нейронов с различными функциями активации.

Существуют бинарные и аналоговые сети. Первые из них оперируют только двоичными сигналами, и выход каждого нейрона может принимать значение либо логического ноля (заторможенное состояние) либо логической единицы (возбужденное состояние).

Еще одна классификация делит нейронные сети на синхронные и асинхронные. В первом случае в каждый момент времени лишь один нейрон

меняет свое состояние, во втором - состояние меняется сразу у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в нейронных сетях задается итерационным выполнением однотипных действий над нейронами.

Выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Для решения отдельных типов задач уже существуют оптимальные конфигурации. Если же задача не может быть сведена ни к одному из известных типов, приходится решать сложную проблему синтеза новой конфигурации. При этом необходимо руководствоваться следующими основными правилами:

- возможности сети возрастают с увеличением числа нейронов сети, плотности связей между ними и числом слоев;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;
- сложность алгоритмов функционирования сети, введение нескольких типов синапсов способствует усилению мощности нейронной сети.

9.5. Обучение искусственных нейронных сетей

Возможности обучения искусственных нейронных сетей ограничены, и нужно решить много сложных задач, чтобы определить, находимся ли мы на правильном пути.

Цель обучения. Сеть обучается, чтобы для некоторого множества входов давать желаемое (или, по крайней мере, сообразное с ним) множество выходов. Каждое такое входное (или выходное) множество рассматривается как вектор. Обучение осуществляется путем последовательного предъявления входных векторов с одновременной подстройкой весов в соответствии с определенной процедурой. В процессе обучения веса сети постепенно становятся такими, чтобы каждый входной вектор вырабатывал выходной вектор (рисунок 9.10).

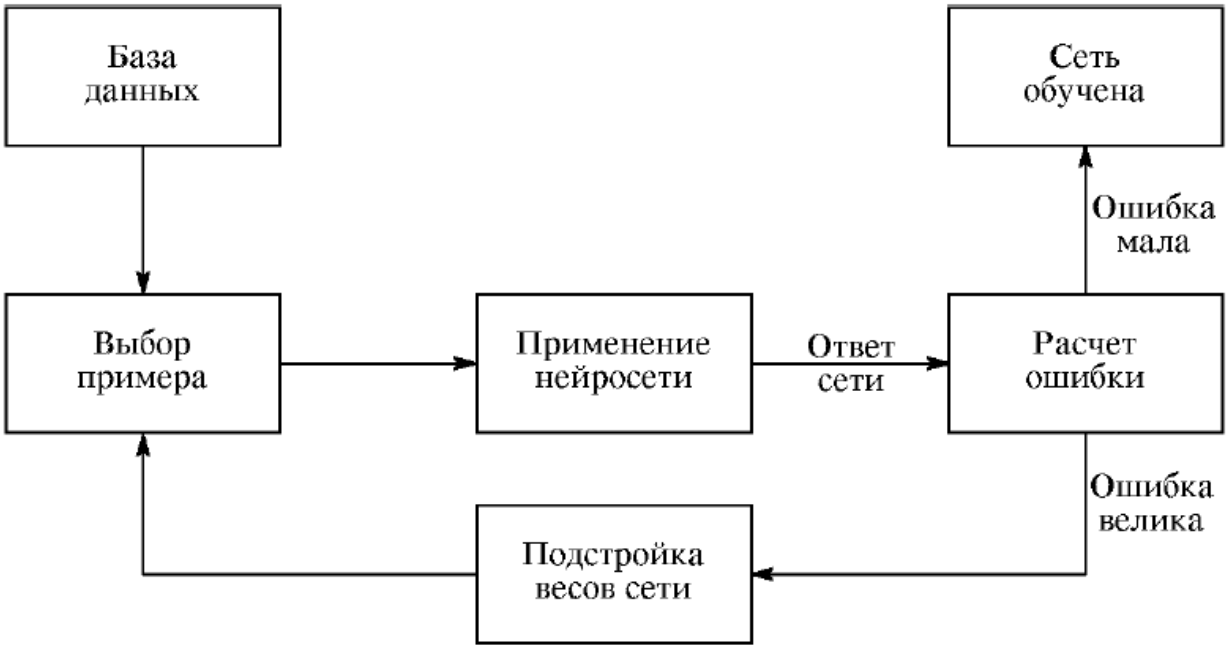


Рисунок 9.10 – Процесс обучения нейронной сети

Обучение с учителем. Различают алгоритмы обучения с учителем и без учителя. Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Обычно сеть обучается на некотором числе таких обучающих пар. Предъявляется выходной вектор, вычисляется выход сети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в сеть, и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, ошибки вычисляются и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

Обучение без учителя. Несмотря на многочисленные прикладные достижения, обучение с учителем критиковалось за свою биологическую неправдоподобность. Трудно вообразить обучающий механизм в мозге, который бы сравнивал желаемые и действительные значения выходов, выполняя коррекцию с помощью обратной связи. Обучение без учителя является намного более правдоподобной моделью обучения для биологической системы. Развитая Кохоненом и многими другими, она не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами. Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса сети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет статистические свойства обучающего множества и группирует сходные

векторы в классы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, но до обучения невозможно предсказать, какой выход будет производиться данным классом входных векторов. Следовательно, выходы подобной сети должны трансформироваться в некоторую понятную форму, обусловленную процессом обучения. Это не является серьезной проблемой. Обычно не сложно идентифицировать связь между входом и выходом, установленную сетью.

Алгоритмы обучения. Большинство современных алгоритмов обучения выросло из концепций Д.О. Хэбба. Он предложил модель обучения без учителя, в которой синаптическая сила (вес) возрастает, если активированы оба нейрона, источник и приемник. Таким образом, часто используемые пути в сети усиливаются и феномены привычки и обучения через повторение получают объяснение.

В искусственной нейронной сети, использующей обучение по Хэббу, наращивание весов определяется произведением уровней возбуждения передающего и принимающего нейронов. Это можно записать как

$$w_{ij}(n+1) = w(n) + \alpha OUT_i OUT_j,$$

где $w_{ij}(n)$ — значение веса от нейрона i к нейрону j до подстройки, $w_{ij}(n+1)$ — значение веса от нейрона i к нейрону j после подстройки, α — коэффициент скорости обучения, OUT_i — выход нейрона i и вход нейрона j , OUT_j — выход нейрона j.

Сети, использующие обучение по Хэббу, конструктивно развивались, однако за последние 20 лет появились и разрабатывались более эффективные алгоритмы обучения. В частности, были развиты алгоритмы обучения с учителем, приводящие к сетям с более широким диапазоном характеристик обучающих входных образов и большими скоростями обучения, чем использующие простое обучение по Хэббу.

Вопросы для повторения и закрепления материала

1. Где используются нейронные сети?
2. Опишите биологический нейрон?
3. Опишите основные этапы разработок в области нейронных сетей?
4. Каким образом строится искусственная модель нейрона?
5. Охарактеризуйте структуру и свойства искусственного нейрона?
6. Что такое активационная функция?
7. Какого типа бывают функции активации нейронов?
8. Проклассифицируйте нейронные сети?
9. Опишите процесс обучения нейронной сети?

10. В чем заключается отличие процесса обучения нейронной сети с учителем и без его участия?

Задания для самостоятельной работы

1. Провести анализ литературы на предмет наиболее важных сфер человеческой деятельности, в которых был использован аппарат нейросетей.

Тема 10. Персептроны. Представимость и разделимость

Цель: рассмотреть принципы реализации нейронов на примере персептрона.

Задачи:

1. Ознакомиться с этапами зарождения нейронных сетей.
2. Освоить понятие персептронной представимости и проблему функции исключающего ИЛИ.
3. Изучить понятие линейной разделимости.
4. Освоить способы преодоления линейной разделимости.
5. Рассмотреть эффективность запоминания.

10.1. Персептроны и зарождение искусственных нейронных сетей

В качестве предмета исследования искусственные нейронные сети впервые заявили о себе в 1940-е годы. Стремясь воспроизвести функции человеческого мозга, исследователи создали простые аппаратные (а позже программные) модели биологического нейрона и системы его соединений. Когда нейрофизиологи достигли более глубокого понимания нервной системы человека, эти ранние попытки стали восприниматься как весьма грубые аппроксимации. Тем не менее, на этом пути были достигнуты впечатляющие результаты, стимулировавшие дальнейшие исследования, которые привели к созданию более изощренных сетей.

Первое систематическое изучение искусственных нейронных сетей было предпринято Маккалокком и Питтсом в 1943 г. Позднее они исследовали сетевые парадигмы для распознавания изображений, подвергаемых сдвигам и поворотам. Простая нейронная модель, показанная на рисунке 10.1, использовалась в большей части их работ.

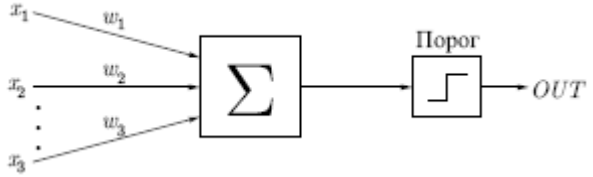


Рисунок 10.1 – Модель персептрона

Элемент Σ умножает каждый вход x на вес w и суммирует взвешенные входы. Если полученная сумма больше заданного порогового значения, выход равен единице, в противном случае — нулю. Эти системы (и множество им подобных) получили название **персептронов.** Они состоят из одного слоя искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов (рисунок 10.2), хотя, в принципе, описываются и более сложные системы. В 60-е годы персептроны вызвали большой интерес и оптимизм. Одной из первых искусственных сетей, способных к перцепции (восприятию) и формированию реакции на воспринятый раздражитель, явился PERCEPTRON Розенблатта (F.Rosenblatt, 1957). Персептрон рассматривался его автором не как конкретное техническое (вычислительное) устройство, а как модель работы мозга. Розенблатт называл такую нейронную сеть трехслойной, однако, по современной терминологии, представленная сеть обычно называется **однослойной**, так как имеет только один слой нейропроцессорных элементов.

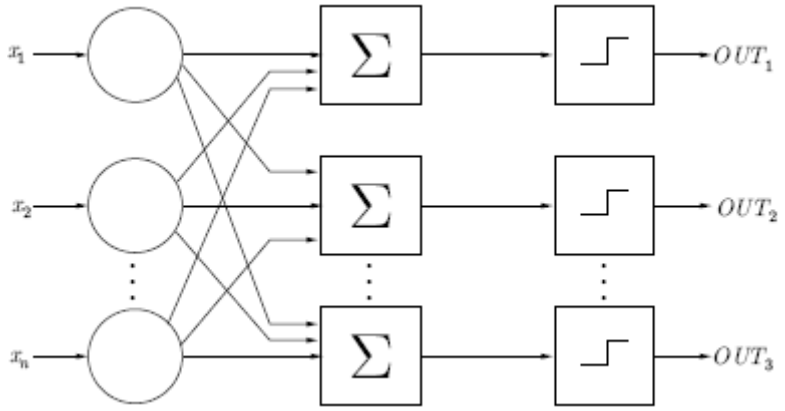


Рисунок 10.2 – Персептронная сеть

В Корнеллской авиационной лаборатории была разработана электротехническая модель персептрона MARK-1, которая содержала 8 выходных элементов. На этом персептроне была проведена серия экспериментов по распознаванию букв алфавита и геометрических образов.

Ф. Розенблатт доказал теорему об обучении персептронов. Д. Уидроу дал ряд убедительных демонстраций систем персептронного типа, и исследователи во всем мире стремились изучить возможности этих систем. Первоначальная эйфория сменилась разочарованием, когда оказалось, что персептроны не способны обучаться решению ряда простых задач. Минский строго проанализировал эту проблему и показал, что имеются жесткие ограничения того, что могут выполнять однослойные персептроны, и, следовательно, того, чему они могут обучаться. Так как в то время методы обучения многослойных сетей не были известны, исследователи занялись более многообещающими проектами, и исследования в области нейронных

сетей пришли в упадок. Недавнее открытие методов обучения многослойных сетей привело к возрождению интереса и возобновлению исследований.

Работа М.Л. Минского, возможно, и охладила пыл энтузиастов персептрона, но обеспечила время для необходимой консолидации и развития лежащей в основе теории. Важно отметить, что анализ Минского не был опровергнут. Он остается актуальным исследованием и должен непременно учитываться как часть базовых знаний, чтобы ошибки 60-х годов не повторились. Несмотря на свои ограничения, персептроны широко изучались. Теория персептронов является основой для многих других типов искусственных нейронных сетей, персептроны иллюстрируют важные принципы. В силу этих причин они являются логической исходной точкой для изучения искусственных нейронных сетей.

10.2. Персептронная представляемость и проблема функции «Исключающего ИЛИ»

Доказательство теоремы обучения персептрона показало, что персептрон способен научиться всему, что он способен представлять. Важно при этом уметь различать представляемость и обучаемость. Понятие **представляемости** относится к способности персептрона (или другой сети) моделировать определенную функцию. **Обучаемость** же требует наличия систематической процедуры настройки весов сети для реализации этой функции.

Для иллюстрации проблемы представляемости допустим, что у нас есть множество карт, помеченных цифрами от 0 до 9. Допустим также, что мы обладаем гипотетической машиной, способной отличать карты с нечетным номером от карт с четным номером и зажигающей индикатор на своей панели при предъявлении карты с нечетным номером. Представима ли такая машина персептроном? То есть, возможно ли сконструировать персептрон и настроить его веса (неважно, каким образом) так, чтобы он обладал такой же разделяющей способностью? Если это достижимо, то говорят, что персептрон способен представлять желаемую машину. Мы увидим, что возможности представления однослойными персептронами весьма ограниченны. Имеется много простых машин, которые не могут быть представлены персептроном, независимо от того, как настраиваются его веса.

Проблема функции Исключающего ИЛИ.

Один из самых пессимистических результатов М.Л. Минского гласит, что однослойный персептрон не может воспроизвести такую простую функцию, как ИСКЛЮЧАЮЩЕЕ ИЛИ. Это функция от двух аргументов, каждый из которых может быть нулем или единицей. Она принимает значение единицы, когда один из аргументов равен единице (но не оба). Проблему можно проиллюстрировать с помощью однослойной однонейронной системы с двумя входами, показанной на рисунке 10.3.

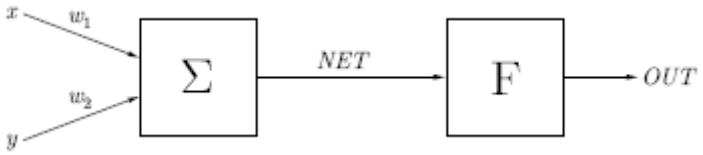


Рисунок 10.3 – Однослойная система

Обозначим один вход через **x**, а другой через **y**, тогда все их возможные комбинации будут состоять из четырех точек на плоскости XOY, как показано на рисунке 10.4.

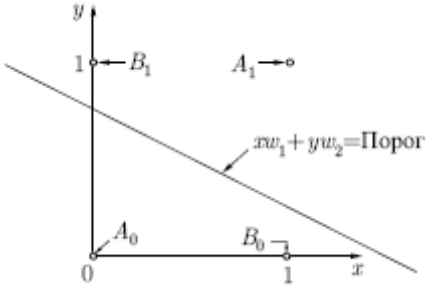


Рисунок 10.4 – Точки функции «Исключающее ИЛИ»

Например, точка x=0 и y=0 обозначена на рисунке как точка A_0. Таблица 10.1 показывает требуемую связь между входами и выходом, где входные комбинации, которые должны давать нулевой выход, помечены A_0 и A_1, единичный выход - B_0 и B_1.

Таблица 10.1 - Связь между входами и выходом

Точки	**Значения x**	**Значения y**	**Требуемый выход**
A_0	0	0	0
B_0	1	0	1
B_1	0	1	1
A_1	1	1	0

В сети на рисунке 10.3 функция F является обычным порогом, так что OUT принимает значение 0, когда NET меньше 0,5, и 1 в случае, когда NET больше или равно 0,5. Нейрон выполняет следующее вычисление:

$$NET = x\omega_1 + y\omega_2 \qquad (10.1)$$

Никакая комбинация значений двух весов не может дать соотношения между входом и выходом, задаваемого таблицей 10.1. Чтобы понять это ограничение, зафиксируем NET на величине порога 0,5. Сеть в этом случае описывается уравнением (10.2). Это уравнение линейно по **x** и **y**, т.е. все значения по **x** и **y**, удовлетворяющие этому уравнению, будут лежать на некоторой прямой в плоскости x-y.

$$x\omega_1 + y\omega_2 = 0{,}5 \qquad (10.2)$$

Любые входные значения для **x** и **y** на этой линии будут давать пороговое значение 0,5 для NET. Входные значения с одной стороны прямой

обеспечат значения NET больше порога, следовательно, OUT=1. Входные значения по другую сторону прямой обеспечат значения NET меньше порога, делая OUT равным 0. Изменения значений w_1, w_2 и порога будут менять наклон и положение прямой. Для того чтобы сеть реализовала функцию ИСКЛЮЧАЮЩЕЕ ИЛИ, заданную таблицей 10.1, нужно расположить прямую так, чтобы точки A_0, A_1 были с одной стороны прямой, а точки B_0, B_1 — с другой. Попытавшись нарисовать такую прямую (рисунок 10.4), убеждаемся, что это невозможно. Это означает, что какие бы значения ни приписывались весам и порогу, сеть неспособна воспроизвести соотношение между входом и выходом, требуемое для представления функции ИСКЛЮЧАЮЩЕЕ ИЛИ. Взглянув на задачу с другой точки зрения, рассмотрим NET как поверхность над плоскостью XOY. Каждая точка этой поверхности находится над соответствующей точкой плоскости XOY на расстоянии, равном значению NET в этой точке. Можно показать, что наклон этой NET - поверхности одинаков для всей поверхности XOY. Все точки, в которых значение NET равно величине порога, проектируются на линию уровня плоскости NET (рисунок 10.5).

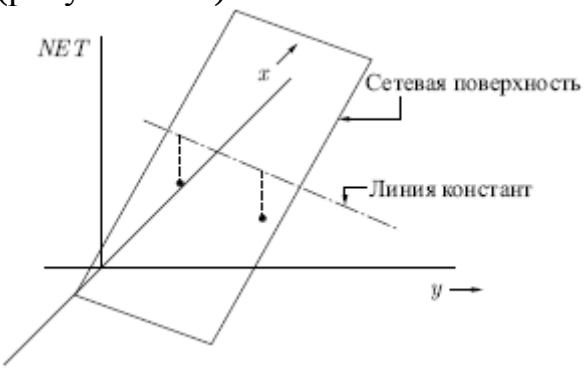


Рисунок 10.5 - Линия уровня плоскости NET

Ясно, что все точки по одну сторону пороговой прямой проецируются в значения NET большие порога, а точки по другую сторону дадут меньшие значения NET. Таким образом, пороговая прямая разбивает плоскость **x-y** на две области. Во всех точках по одну сторону пороговой прямой значение OUT равно единице, по другую сторону — нулю.

10.3. Линейная разделимость

Как мы убедились, невозможно нарисовать прямую линию, разделяющую плоскость x-y так, чтобы реализовывалась функция ИСКЛЮЧАЮЩЕЕ ИЛИ. К сожалению, этот пример не единственный. Имеется обширный класс функций, не реализуемых однослойной сетью. Об этих функциях говорят, что они являются линейно неразделимыми: они-то и накладывают определенные ограничения на возможности однослойных сетей.

Линейная разделимость ограничивает однослойные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. Для нашего случая с двумя входами разделитель является прямой линией. В случае трех входов разделение осуществляется плоскостью, рассекающей трехмерное пространство. Для четырех или более входов визуализация невозможна, и необходимо мысленно представить n-мерное пространство, рассекаемое "гиперплоскостью" — геометрическим объектом, который делит пространство четырех или большего числа измерений.

Так как линейная разделимость ограничивает возможности персептронного представления, то важно знать, является ли данная функция разделимой. К сожалению, не существует простого способа определить это, если число переменных велико.

Нейрон с **n** двоичными входами может иметь 2^n различных входных образов, состоящих из нулей и единиц. Так как каждый входной образ может соответствовать двум различным бинарным выходам (единица и ноль), то всего имеется 2^{2^n} функций от **n** переменных.

Как следует из таблицы 10.2, вероятность того, что случайно выбранная функция окажется линейно разделимой, весьма мала даже для умеренного числа переменных. По этой причине однослойные персептроны на практике ограничены простыми задачами.

Таблица 10.2 - Вероятность того, что случайно выбранная функция окажется линейно разделимой

n	2^{2^n}	Число линейно разделимых функций
1	4	4
2	16	14
3	256	104
4	65536	1882
5	$4{,}3 \times 10^9$	94572
6	$1{,}8 \times 10^{19}$	15028134

10.4. Преодоление ограничения линейной разделимости

К концу 1960-х годов проблема линейной разделимости была хорошо понята. К тому же, было известно, что это серьезное ограничение представляемости однослойными сетями можно преодолеть, добавив дополнительные слои. Например, двухслойные сети можно получить каскадным соединением двух однослойных сетей. Они способны выполнять более общие классификации, отделяя те точки, которые содержатся в выпуклых ограниченных или неограниченных областях. **Выпуклая область,** если для любых двух ее точек соединяющий их отрезок целиком лежит в области. **Ограниченная область,** если ее можно заключить в некоторый

круг. Неограниченную область невозможно заключить внутрь круга (например, область между двумя параллельными линиями). Примеры выпуклых ограниченных и неограниченных областей представлены на рисунке 10.6.

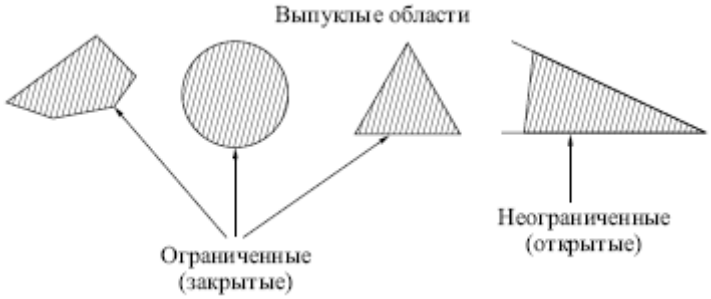


Рисунок 10.6 – Выпуклые ограниченные и неограниченные области

Чтобы уточнить требование выпуклости, рассмотрим простую двуслойную сеть с двумя входами, которые подведены к двум нейронам первого слоя, соединенными с единственным нейроном в слое 2 (рисунок 10.7а). Пусть порог выходного нейрона равен 0,75, а оба его веса равны 0,5. В этом случае для того, чтобы порог был превышен и на выходе появилась единица, требуется, чтобы оба нейрона первого уровня на выходе имели единицу. Таким образом, выходной нейрон реализует логическую функцию «И». На рисунке 10.7а каждый нейрон слоя 1 разбивает плоскость XOY на две полуплоскости, один обеспечивает единичный выход для входов ниже верхней линии, другой — для входов выше нижней линии. На рисунке 10.7б показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V-образной области. Аналогично, во втором слое может быть использовано три нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник любой желаемой формы. Все такие многогранники выпуклы, так как они образованы с помощью операции «И» над областями, задаваемыми линиями: следовательно, только выпуклые области и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

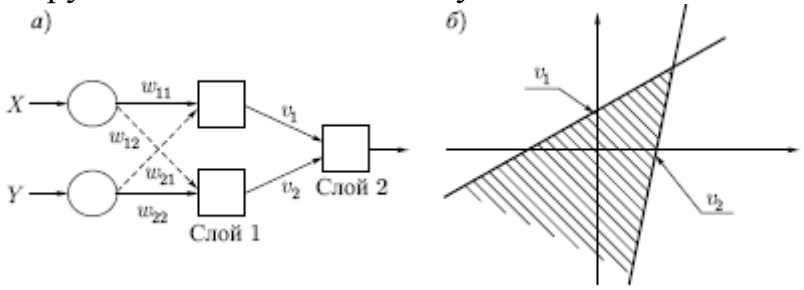


Рисунок 10.7 – Двуслойная нейронная сеть (а) и результат двойного разбиения (б)

Нейрон второго слоя не ограничен функцией «И». Он может реализовывать многие другие функции при подходящем выборе весов и порога. Например, можно сделать так, чтобы единичный выход любого из нейронов первого слоя приводил к появлению единицы на выходе нейрона второго слоя, реализовав тем самым логическое «ИЛИ». Например, имеется 16 двоичных функций от двух переменных. Если выбирать подходящим образом веса и порог, то можно воспроизвести 14 из них (все, кроме ИСКЛЮЧАЮЩЕЕ ИЛИ и ИСКЛЮЧАЮЩЕЕ НЕТ).

Входы не обязательно должны быть двоичными. Вектор непрерывных входов может представлять собой произвольную точку на плоскости XOY. В этом случае мы имеем дело со способностью сети разбивать плоскость на непрерывные области, а не с разделением дискретных множеств точек. Для всех этих функций, однако, линейная разделимость показывает, что выход нейрона второго слоя равен единице только в части плоскости XOY, ограниченной многоугольной областью. Поэтому для разделения плоскостей P и Q необходимо, чтобы все P лежали внутри выпуклой многоугольной области, не содержащей точек Q (или наоборот).

Трехслойная сеть, впрочем, есть более общий случай. Ее классифицирующие возможности ограничены лишь числом искусственных нейронов и весов. Ограничения на выпуклость отсутствуют. Теперь нейрон третьего слоя принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть невыпуклой. На рисунке 10.8б иллюстрируется ситуация, когда два треугольника A и B, скомбинированные с помощью функций "A и не B", задают невыпуклую область. При добавлении нейронов и весов число сторон многоугольников может неограниченно возрастать. Это позволяет аппроксимировать область любой формы с любой точностью. Вдобавок, не все выходные области второго слоя должны пересекаться. Возможно, следовательно, объединять различные области, выпуклые и невыпуклые, выдавая на выходе единицу всякий раз, когда входной вектор принадлежит одной из них.

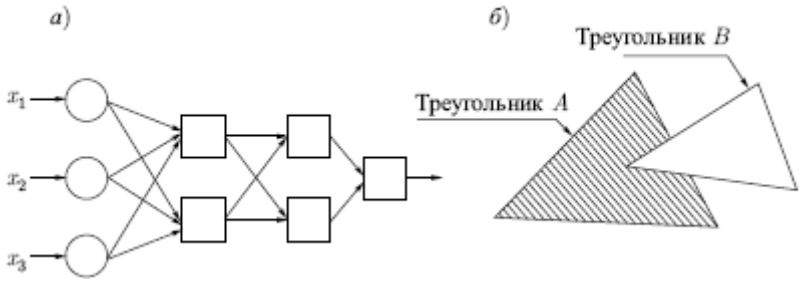


Рисунок 10.8 - Трехслойная нейронная сеть (а) и результат тройного разбиения (б)

Несмотря на то, что возможности многослойных сетей были известны давно, в течение многих лет не было теоретически обоснованного алгоритма для настройки их весов.

10.5. Эффективность запоминания

Серьезные вопросы существуют относительно эффективности запоминания информации в персептроне (или любых других нейронных сетях) по сравнению с обычной компьютерной памятью и методами поиска информации в ней. **Например,** в компьютерной памяти можно хранить все входные образы вместе с классифицирующими битами. Компьютер должен найти требуемый образ и дать его классификацию. Многочисленные и хорошо известные методы могли бы применяться для ускорения поиска. Если точное соответствие не найдено, то для ответа может быть использовано правило ближайшего соседа.

Число битов, необходимое для хранения этой же информации в весах персептрона, может быть значительно меньшим по сравнению с методом обычной компьютерной памяти, если образы допускают экономичную запись. Однако М.Л. Минский построил патологические примеры, в которых число битов, требуемых для представления весов, растет в зависимости от размерности задачи быстрее, чем экспоненциально. В этих случаях требования к памяти быстро становятся невыполнимыми. Если, как он предположил, эта ситуация не является исключением, то персептроны часто могут быть ограничены только малыми задачами. Насколько общими являются такие неподатливые множества образов? Вопрос остается открытым и относится ко всем нейронным сетям. Поиски ответа чрезвычайно важны для дальнейших исследований в этой области.

Вопросы для повторения и закрепления материала

1. В чем особенность персептрона?
2. Опишите модель персептрона?
3. Что такое персептронная представляемость?
4. В чем суть проблемы исключающего ИЛИ?
5. Что такое линейная разделимость?
6. Что на выходе персептрона?
7. Каким образом преодолевается линейная разделимость?
8. Какие функции булевой алгебры можно смоделировать на основании персептрона?

Задания для самостоятельной работы

1. Разработайте персептронные сети, моделирующие работу логических функций булевой алгебры.

Тема 11. Персептроны. Обучение персептрона

Цель: рассмотреть способы обучения персептрона.

Задачи:

1. Ознакомиться с термином обучение персептрона.
2. Рассмотреть алгоритм обучения однослойного персептрона.
3. Рассмотреть понятие целочисленности весов персептрона.
4. Ознакомиться с двуслойностью персептрона.
5. Ознакомиться с трудностями алгоритма обучения персептрона.

11.1. Понятие термина «обучение» персептрона

Способность искусственных нейронных сетей к обучению является их наиболее важным свойством. Подобно биологическим системам, которые они моделируют, эти нейронные сети сами совершенствуют себя в результате попыток создать лучшую модель поведения.

Используя критерий линейной разделимости, можно решить, способна ли однослойная нейронная сеть реализовывать требуемую функцию. Даже в том случае, когда ответ положительный, это принесет мало пользы, если у нас нет способа найти нужные значения для весов и порогов. Чтобы сеть представляла практическую ценность, нужен систематический метод (алгоритм) для вычисления этих значений. Ф. Розенблатт создал такой метод в своем алгоритме обучения персептрона и доказал: персептрон может быть обучен всему, что он может реализовывать.

Обучение может быть с учителем или без него. Для обучения с учителем нужен "внешний" учитель, который оценивал бы поведение системы и управлял ее последующими модификациями. При обучении без учителя, сеть путем самоорганизации делает требуемые изменения. Обучение персептрона является обучением с учителем.

Алгоритм обучения персептрона может быть реализован на цифровом компьютере или другом электронном устройстве, и сеть становится в определенном смысле самоподстраивающейся. По этой причине процедуру подстройки весов обычно называют "обучением" и говорят, что сеть "обучается". Доказательство Розенблатта стало основной вехой и дало мощный импульс исследованиям в этой области. Сегодня в той или иной форме элементы алгоритма обучения персептрона встречаются во многих сетевых парадигмах.

11.2. Алгоритм обучения однослойного персептрона

Персептрон должен решать задачу классификации по бинарным входным сигналам. Набор входных сигналов будем обозначать n-мерным вектором **x**. Все элементы вектора являются булевыми переменными (переменными, принимающими значения "Истина" или "Ложь"). Однако иногда полезно оперировать числовыми значениями. Будем считать, что

значению "ложь" соответствует числовое значение 0, а значению "Истина" соответствует 1.

Персептроном будем называть устройство, вычисляющее следующую систему функций:

$$\psi = [\sum_{i=1}^{m} \omega_i x_i > \theta] \qquad (11.1)$$

где w_I — веса персептрона, θ — порог, x_I — значения входных сигналов, скобки [] означают переход от булевых (логических) значений к числовым значениям по правилам, изложенным выше.

Обучение персептрона состоит в подстройке весовых коэффициентов. Пусть имеется набор пар векторов (x^α, y^α), $\alpha = 1, \ldots, p$, называемый **обучающей выборкой**. Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора x^α на выходах всякий раз получается соответствующий вектор y^α.

Предложенный Ф.Розенблаттом метод обучения состоит в итерационной подстройке матрицы весов, последовательно уменьшающей ошибку в выходных векторах. Алгоритм включает несколько шагов:

1. Начальные значения весов всех нейронов $W(t=0)$ полагаются случайными.
2. Сети предъявляется входной образ x^α, в результате формируется выходной образ $\widetilde{y^\alpha} \neq y^\alpha$.
3. Вычисляется вектор ошибки $\delta^\alpha = (y^\alpha - \widetilde{y^\alpha})$, делаемой сетью на выходе. Дальнейшая идея состоит в том, что изменение вектора весовых коэффициентов в области малых ошибок должно быть пропорционально ошибке на выходе и равно нулю, если ошибка равна нулю.
4. Вектор весов модифицируется по следующей формуле: $W(t+\Delta T) = W(t) + \eta x^\alpha \cdot (\delta^\alpha)^T$. Здесь $0 < \eta < 1$ — темп обучения.
5. Шаги 2—4 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется **эпохой.** Обучение завершается по истечении нескольких эпох: а) когда итерации сойдутся, т.е. вектор весов перестает изменяться, или б) когда полная, просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

Объясним данный алгоритм более подробно. Подаем на вход персептрона такой вектор **x**, для которого уже известен правильный ответ. Если выходной сигнал персептрона совпадает с правильным ответом, то никаких действий предпринимать не надо. В случае ошибки, необходимо обучить персептрон правильно решать данный пример. Ошибки могут быть двух типов. Рассмотрим каждый из них:

1. Первый тип ошибки: на выходе персептрона — 0, а правильный ответ — 1. Для того чтобы персептрон выдавал правильный ответ, необходимо, чтобы сумма в правой части (11.1) стала больше. Поскольку переменные принимают значения 0 или 1, увеличение суммы может быть достигнуто за счет увеличения весов w_i. Однако нет смысла увеличивать веса при переменных x_i, которые равны нулю. Таким образом, следует увеличить веса w_i при тех переменных x_i, которые равны 1.

Первое правило. Если на выходе персептрона получен 0, а правильный ответ равен 1, то необходимо увеличить веса связей между одновременно активными нейронами. При этом выходной персептрон считается активным.

2. Второй тип ошибки: на выходе персептрона — 1, а правильный ответ равен нулю. Для обучения правильному решению данного примера следует уменьшить сумму в правой части (11.1). Следовательно, необходимо уменьшить веса связей w_i при тех переменных, которые равны 1 (поскольку нет смысла уменьшать веса связей при равных нулю переменных x_i). Необходимо также провести эту процедуру для всех активных нейронов предыдущих слоев. В результате получаем второе правило.

Второе правило. Если на выходе персептрона получена единица, а правильный ответ равен нулю, то необходимо уменьшить веса связей между одновременно активными нейронами.

Таким образом, процедура обучения сводится к последовательному перебору всех примеров обучающего множества с применением правил обучения для ошибочно решенных примеров. Если после очередного цикла предъявления всех примеров окажется, что все они решены правильно, то процедура обучения завершается.

Нерассмотренными остались два вопроса:

1. О сходимости процедуры обучения.
2. На сколько нужно увеличивать (уменьшать) веса связей при применении правил обучения.

Ответ на первый вопрос дают следующие теоремы.

Теорема о сходимости персептрона. Если существует вектор параметров w, при котором персептрон правильно решает все примеры обучающей выборки, то при обучении персептрона по вышеописанному алгоритму решение будет найдено за конечное число шагов.

Теорема о зацикливании персептрона. Если не существует вектора параметров w, при котором персептрон правильно решает все примеры обучающей выборки, то при обучении персептрона по данному правилу через конечное число шагов вектор весов начнет повторяться.

Таким образом, данные теоремы утверждают, что, запустив процедуру обучения персептрона, через конечное время мы либо получим обучившийся персептрон, либо ответ, что данный персептрон поставленной задаче обучиться не может.

11.3. Целочисленность весов персептронов

Для ответа на вопрос о количественных характеристиках вектора w рассмотрим следующую теорему.

Теорема. Любой персептрон можно заменить другим персептроном того же вида с целыми весами связей.

Доказательство. Обозначим множество примеров одного класса (правильный ответ равен 0) через X_0, а другого (правильный ответ равен 1) — через X_1. Вычислим максимальное и минимальное значения суммы в правой части (11.1):

$$s_0 = \max_{x \in X_0} \sum_i w_i x_i, \quad s_1 = \min_{x \in X_1} \sum_i w_i x_i.$$

Определим допуск S как минимум из S_0 и S_1. Положим $\delta = s/(m+1)$, где m — число слагаемых в (11.1). Поскольку персептрон (11.1) решает поставленную задачу классификации и множество примеров в обучающей выборке конечно, то $\delta > 0.$ Из теории чисел известна теорема о том, что любое действительное число можно сколь угодно точно приблизить рациональными числами. Заменим веса w_i на рациональные числа так, чтобы выполнялись следующие неравенства: $|w_i - w_i'| < \delta.$

Из этих неравенств следует, что при использовании весов w_i' персептрон будет работать с теми же результатами, что и первоначальный персептрон. Действительно, если правильным ответом примера является 0, имеем $\sum_i w_i x_i \leqslant -s.$

Подставив новые веса, получим:

$$\sum_i w_i' x_i = \sum_i (w_i' - w_i) x_i + \sum_i w_i x_i \leqslant \sum_i |w_i' - w_i| x_i - s \leqslant \leqslant \sum_i |w_i' - w_i| - s < (m+1)\delta - s = 0.$$

Откуда следует необходимое неравенство

$$\sum_i \omega_i' x_i < 0 \qquad (11.2)$$

Аналогично, в случае правильного ответа равного 1, имеем $\sum_i w_i x_i < s$, откуда, подставив новые веса и порог, получим:

$$\sum_i w_i' x_i = \sum_i (w_i' - w_i) x_i + \sum_i w_i x_i \geqslant s - \sum_i |w_i' - w_i| x_i \geqslant \geqslant s - \sum_i |w_i' - w_i| > s - (m+1)\delta = 0.$$

Отсюда следует выполнение неравенства

$$\sum_i \omega_i' x_i > 0 \qquad (11.3)$$

Неравенства (11.2) и (11.3) доказывают возможность замены всех весов и порога любого персептрона рациональными числами. Очевидно также, что

при умножении всех весов и порога на одно и то же ненулевое число персептрон не изменится. Поскольку любое рациональное число можно представить в виде отношения целого числа к натуральному числу, получим

$$\psi = \left[\sum_{i=1}^{m} w_i x_i > 0\right] = \left[\sum_{i=1}^{m} w_i' x_i > 0\right] = \left[\sum_{i=1}^{m} \frac{w_i''}{r_i} x_i > 0\right], \tag{11.4}$$

где w_i'' — целые числа. Обозначим через r произведение всех знаменателей: $r = \prod_{i=0}^{m} r_i$. Умножим все веса и порог на r. Получим веса целочисленные $w''' = rw''$. Из (11.2), (11.3) и (11.4) получаем

$$\psi = \left[\sum_{i=1}^{m} w_i x_i > 0\right] = \left[\sum_{i=1}^{m} w_i' x_i > 0\right] = \left[\sum_{i=1}^{m} \frac{w_i''}{r_i} x_i > 0\right] = \left[\sum_{i=1}^{m} w_i''' x_i > 0\right],$$

что и завершает доказательство теоремы.

Поскольку из доказанной теоремы следует, что веса персептрона являются целыми числами, то вопрос о выборе шага при применении правил обучения решается просто: веса и порог следует увеличивать (уменьшать) на единицу.

11.4. Двуслойность персептрона

Алгоритм обучения персептрона возможно использовать и для многослойных персептронов. Однако теоремы о сходимости и зацикливании персептрона, приведенные выше, верны только при обучении однослойного персептрона— или многослойного персептрона при условии, что обучаются только веса персептрона, стоящего в последнем слое сети. В случае произвольного многослойного персептрона они не работают. Следующий пример демонстрирует основную проблему, возникающую при обучении многослойных персептронов.

Пусть веса всех слоев персептрона в ходе обучения сформировались так, что все примеры обучающего множества, кроме первого, решаются правильно. При этом правильным ответом первого примера является 1. Все входные сигналы персептрона последнего слоя равны нулю. В этом случае первое правило не дает результата, поскольку все нейроны предпоследнего слоя не активны. Существует множество способов решать эту проблему. Однако все эти методы не являются регулярными и не гарантируют сходимость многослойного персептрона к решению, даже при условии, что такое решение существует.

В действительности, проблема настройки (обучения) многослойного персептрона решается следующей теоремой.

Теорема о двуслойности персептрона. Любой многослойный персептрон может быть представлен в виде двуслойного персептрона с необучаемыми весами первого слоя.

Для доказательства этой теоремы потребуется одна теорема из математической логики.

Теорема о дизъюнктивной нормальной форме. Любая булева функция булевых аргументов может быть представлена в виде дизъюнкции конъюнкций элементарных высказываний и отрицаний элементарных высказываний:

$$f = \vee(\& \, x_i \, \& \, \neg x_j).$$

Напомним некоторые свойства дизъюнктивной нормальной формы.

Свойство 1. В каждый конъюнктивный член (слагаемое) входят все элементарные высказывания либо в виде самого высказывания, либо в виде его отрицания.

Свойство 2. При любых значениях элементарных высказываний в дизъюнктивной нормальной форме может быть истинным не более одного конъюнктивного члена (слагаемого).

Доказательство теоремы о двуслойности персептрона. Из теоремы о дизъюнктивной нормальной форме следует, что любой многослойный персептрон может быть представлен в следующем виде:

$$\psi = |\vee(\& \, x_i \, \& \neg x_j)|. \qquad (11.5)$$

В силу второго свойства дизъюнктивной нормальной формы, равенство (11.5) можно переписать в виде

$$\psi = [\vee(\& \, x_i \, \& \, \neg x_j)] = \left[\sum[(\& \, x_i \, \& \, \neg x_j)] > 0\right]. \qquad (11.6)$$

Переведем в арифметическую форму все слагаемые в выражении (11.6). Конъюнкцию заменяем на умножение, а отрицание на разность: $\neg x_j = 1 - x_j$. Произведя эту замену и приведя подобные члены, получим:

$$\psi = \left[\sum_l \alpha_l \prod_{i \in I_l} x_i > 0\right], \qquad (11.7)$$

где I_l — множество индексов сомножителей в l-м слагаемом, α_l — число, указывающее, сколько раз такое слагаемое встретилось в выражении (11.6) после замены и раскрытия скобок (число подобных слагаемых).

Заменим i-е слагаемое в формуле (11.7) персептроном следующего вида:

$$\varphi_i = \prod_{l \in I_l} x_l = \left[\sum_{l \in I_l} x_l > |I_l| - 1\right]. \qquad (11.8)$$

Подставив выражение (11.8) в формулу (11.7), получим равенство (11.1), то есть произвольный многослойный персептрон представлен в виде (11.1) с целочисленными коэффициентами. В качестве персептронов первого слоя используются персептроны вида (11.8) с необучаемыми весами. Теорема доказана.

11.5. Трудности с алгоритмом обучения персептрона

Иногда бывает сложно определить, выполнено ли условие разделимости для конкретного обучающего множества. Кроме того, во многих встречающихся на практике ситуациях входы часто меняются во времени и могут быть разделимы в один момент времени и неразделимы - в другой. В доказательстве алгоритма обучения персептрона ничего не говорится также о том, сколько шагов требуется для обучения сети. Мало утешительного знать, что обучение закончится за конечное число шагов, если необходимое для этого время сравнимо с геологической эпохой. Кроме того, не доказано, что персептронный алгоритм обучения более быстр по сравнению с простым перебором всех возможных значений весов, и в некоторых случаях этот примитивный подход может оказаться лучше.

На эти вопросы никогда не находилось удовлетворительного ответа, они относятся к природе обучающего материала. Ответы для современных сетей, как правило, не более удовлетворительны, чем для персептрона. Эти проблемы являются важной областью современных исследований.

Следует отметить следующие основные свойства персептронов:

1. Любой персептрон может содержать один или два слоя. В случае двухслойного персептрона веса первого слоя не обучаются.
2. Веса любого персептрона можно заменить на целочисленные.
3. При обучении после конечного числа итераций возможны два исхода: персептрон обучится или вектор весов персептрона будет повторяться (персептрон зациклится).

Знание этих свойств позволяет избежать "усовершенствований" типа модификации скорости обучения и других, столь же "эффективных" модернизаций.

Вопросы для повторения и закрепления материала

1. В чем суть понятия обучение персептрона?
2. Опишите основные этапы обучения однослойного персептрона?
3. Каким образом формируется обучающая выборка?
4. В чем заключается смысл теоремы о сходимости?
5. В чем заключается смысл теоремы о зацикливании?
6. Что подразумевается под целочисленностью весов персептрона?
7. Каким образом выглядит двуслойный персептрон?
8. В чем заключается смысл теоремы о двуслойности?
9. В чем заключается смысл теоремы о дизъюнктивной нормальной форме?

10. Какие трудности связаны с алгоритмом обучения персептрона?

Задания для самостоятельной работы

1. Проведите анализ методов обучения нейронных сетей с выявлением их узких мест.

Тема 12. Процедура обратного распространения

Цель: изучить принципы работы процедуры обратного распространения.

Задачи:

1. Ознакомиться с основными понятиями процедуры обратного распространения.
2. Изучить обучающий алгоритм обратного распространения.
3. Ознакомиться с дальнейшими алгоритмическими разработками.

12.1. Введение в процедуру обратного распространения

Среди различных структур 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. Такие НС называются полносвязными. 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. По этому принципу строится, например, алгоритм обучения однослойного персептрона. 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.

Один из вариантов решения этой проблемы — разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо. Второй вариант — динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный "метод тыка", несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, более приемлемый вариант — 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 **процедуры обратного распространения**. Разработка алгоритма обратного распространения сыграла важную роль в возрождении интереса к искусственным нейронным сетям. Обратное распространение — это систематический метод для обучения многослойных искусственных нейронных сетей. Он имеет солидное математическое обоснование. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала богатые возможности этой методики.

12.2. Обучающий алгоритм обратного распространения

На рисунке 12.1 показан нейрон, используемый в качестве основного строительного блока в сетях обратного распространения. Подается множество входов, идущих либо извне, либо от предшествующего слоя. Каждый из них умножается на вес, и произведения суммируются:

$$NET = o_1 w_1 + o_2 w_2 + \ldots + o_n w_n.$$

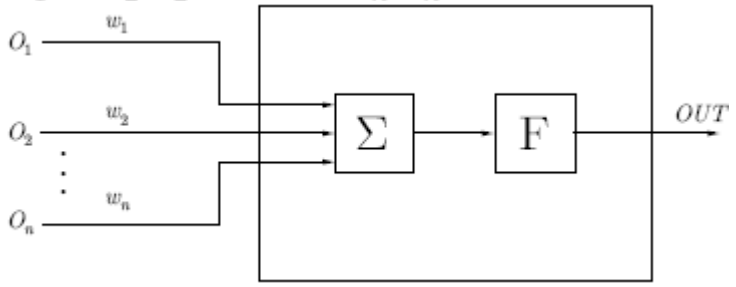


Рисунок 12.1 – Структура нейрона

Эта сумма, обозначаемая NET, должна быть вычислена для каждого нейрона сети. После того, как величина NET вычислена, она модифицируется с помощью активационной функции, и получается сигнал OUT. Для алгоритмов обратного распространения обычно используется функция

$$OUT = \frac{1}{1 + e^{-NET}}. \qquad (12.1)$$

Как показывает уравнение (12.1), эта функция, называемая **сигмоидом**, весьма удобна, так как имеет простую производную, что используется при реализации алгоритма обратного распространения:

$$\frac{\partial OUT}{\partial NET} = OUT(1 - OUT). \qquad (12.2)$$

Сигмоид, который иногда называется также **логистической** или **сжимающей функцией**, сужает диапазон изменения NET так, что значение OUT лежит между нулем и единицей. Как указывалось выше, многослойные нейронные сети обладают большей представляющей мощностью, чем

однослойные, лишь в случае присутствия нелинейности. Сжимающая функция обеспечивает требуемую нелинейность.

В действительности имеется множество функций, которые могли бы быть использованы. Для алгоритма обратного распространения требуется только, чтобы функция была всюду дифференцируема. Сигмоид удовлетворяет этому требованию. Его дополнительное преимущество состоит в автоматическом контроле усиления. Для слабых сигналов (величина NET близка к нулю) кривая вход-выход имеет сильный наклон, дающий большое усиление. Когда величина сигнала становится больше, усиление падает. Таким образом, большие сигналы воспринимаются сетью без насыщения, а слабые сигналы проходят по сети без чрезмерного ослабления.

Рассмотрим иерархическую сетевую структуру, в которой связанные между собой нейроны объединены в несколько слоев (рисунок 12.2). На возможность построения таких архитектур указал еще Ф. Розенблатт, однако им не была решена проблема обучения. Межнейронные синаптические связи сети устроены таким образом, что каждый нейрон на данном уровне иерархии принимает и обрабатывает сигналы от каждого нейрона более низкого уровня. Таким образом, в данной сети имеется выделенное направление распространения нейроимпульсов — от входного слоя через один (или несколько) скрытых слоев к выходному слою нейронов. Нейросеть такой топологии мы будем называть **обобщенным многослойным персептроном** или, если это не будет вызывать недоразумений, просто персептроном.

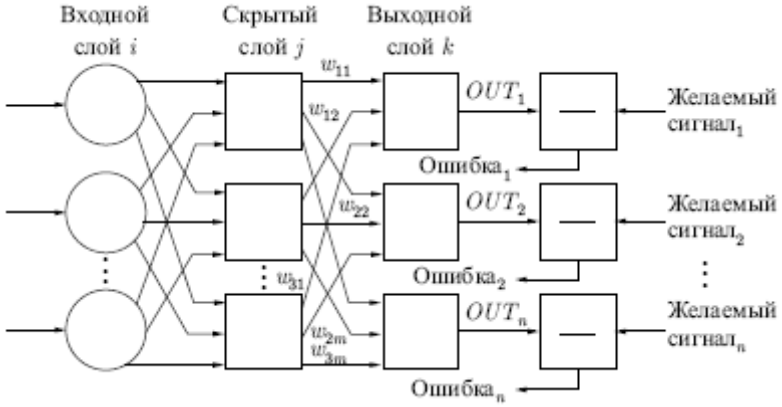


Рисунок 12.2 – Структура многослойной персептронной сети

Персептрон представляет собой сеть, состоящую из нескольких последовательно соединенных слоев нейронов. На низшем уровне иерархии находится **входной** слой сенсорных элементов, задачей которого является только прием и распространение по сети входной информации. Далее имеются один или, реже, несколько **скрытых** слоев. Каждый нейрон на скрытом слое имеет несколько входов, соединенных с выходами нейронов предыдущего слоя или непосредственно со входными сенсорами

$x_1, \ldots, x_n$, и один выход. Выходы нейронов последнего, **выходного**, слоя описывают результат классификации Y=Y(X). Особенности работы персептрона состоят в следующем. Каждый нейрон суммирует поступающие к нему сигналы от нейронов предыдущего уровня иерархии с весами, определяемыми состояниями синапсов, и формирует ответный сигнал (переходит в возбужденное состояние), если полученная сумма выше порогового значения. Персептрон переводит входной образ, определяющий степени возбуждения нейронов самого нижнего уровня иерархии, в выходной образ, определяемый нейронами самого верхнего уровня. Число последних обычно сравнительно невелико. Состояние возбуждения нейрона на верхнем уровне говорит о принадлежности входного образа к той или иной категории.

Традиционно рассматривается аналоговая логика, при которой допустимые состояния синаптических связей определяются произвольными действительными числами, а степени активности нейронов - действительными числами между 0 и 1. Иногда исследуются также модели с дискретной арифметикой, в которой синапс характеризуется двумя булевыми переменными: активностью (0 или 1) и полярностью (-1 или +1). Состояния нейронов могут при этом описываться одной булевой переменной. Данный дискретный подход делает конфигурационное пространство состояний нейронной сети конечным (не говоря уже о преимуществах при аппаратной реализации).

Мы рассмотрим классический вариант многослойной сети с аналоговыми синапсами и сигмоидальной передаточной функцией нейронов, определяемой формулой (12.1).

В литературе нет единого мнения относительно того, что именно считать числом слоев в таких сетях. Одни авторы используют число слоев нейронов (включая не суммирующий входной слой), другие — число слоев весов. Так как последнее определение - функционально описательное, то оно будет использовано и нами. Согласно этому определению, сеть на рисунке 12.2 рассматривается как двухслойная. Нейрон объединен с множеством весов, присоединенных к его входу. Таким образом, веса первого слоя оканчиваются на нейронах первого слоя. Вход распределительного слоя считается нулевым слоем.

Процедура обратного распространения применима к сетям с любым числом слоев. Однако для того, чтобы продемонстрировать алгоритм, достаточно двух слоев. Сейчас будут рассматриваться лишь сети прямого действия, хотя обратное распространение применимо и к сетям с обратными связями.

Обзор обучения. Целью обучения сети является такая подстройка ее весов, чтобы приложение некоторого множества входов приводило к требуемому множеству выходов. Для краткости эти множества входов и выходов будут называться **векторами**. При обучении предполагается, что

для каждого входного вектора существует парный ему целевой вектор, задающий требуемый выход. Вместе они называются **обучающей парой**. Как правило, сеть обучается на многих парах. Например, входная часть обучающей пары может состоять из набора нулей и единиц, представляющего двоичный образ некоторой буквы алфавита. На рисунке 12.3 показано множество входов для буквы "А", нанесенной на сетке. Если через квадрат проходит линия, то соответствующий нейронный вход равен единице, в противном случае он равен нулю. Выход может быть числом, представляющим букву "А", или другим набором из нулей и единиц, который может быть использован для получения выходного образа. При необходимости распознавать с помощью сети все буквы латинского алфавита, потребовалось бы 26 обучающих пар. Такая группа обучающих пар называется **обучающим множеством.**

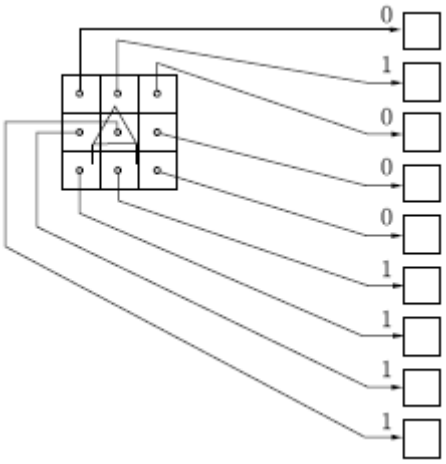


Рисунок 12.3 – Множество входов для буквы «А»

Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других некорректных случаев. **Например,** если всем весам придать одинаковые начальные значения, а для требуемого функционирования нужны неравные значения, то сеть не сможет обучиться.

Обучение сети обратного распространения требует выполнения следующих операций:

1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Подкорректировать веса сети так, чтобы минимизировать ошибку.

5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Операции, выполняемые шагами 1 и 2, сходны с теми, которые выполняются при функционировании уже обученной сети, — подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На рисунке 11.2 сначала вычисляются выходы нейронов слоя j, затем они используются в качестве входов слоя **k**, после чего вычисляются выходы нейронов слоя **k**, которые и образуют выходной вектор сети.

На шаге 3 каждый из выходов сети, которые на рисунке 12.2 обозначены OUT, вычитается из соответствующей компоненты целевого вектора, чтобы получить значение ошибки. Эта ошибка используется на шаге 4 для коррекции весов сети, причем знак и величина изменений весов определяются алгоритмом обучения (см. ниже).

После достаточного числа повторений этих четырех шагов разность между действительными и целевыми выходами должна уменьшиться до приемлемой величины: при этом говорят, что сеть обучилась. Теперь сеть используется для распознавания, и веса не изменяются.

На шаги 1 и 2 можно смотреть как на "проход вперед", так как сигнал распространяется по сети от входа к выходу. Шаги 3, 4 составляют "обратный проход", здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов. Эти два прохода теперь будут детализированы и записаны как математические выражения.

Проход вперед. Шаги 1 и 2 могут быть выражены в векторной форме следующим образом: подается входной вектор X и на выходе получается вектор Y. Векторная пара вход — цель X и H берется из обучающего множества. Вычисления проводятся над вектором X, чтобы получить выходной вектор Y.

Как мы видели, вычисления в многослойных сетях выполняются слой за слоем, начиная с ближайшего к входу. Величина NET каждого нейрона первого слоя вычисляется как взвешенная сумма входов нейрона. Затем активационная функция F "сжимает" NET и дает величину OUT для каждого нейрона в этом слое. Когда множество выходов слоя получено, оно является входным множеством для следующего слоя. Процесс повторяется слой за слоем, пока не будет получено заключительное множество выходов сети.

Этот процесс может быть выражен в сжатой форме с помощью векторной нотации. Веса между нейронами будем рассматривать как матрицу W. **Например**, вес от нейрона 8 в слое 2 к нейрону 5 слоя 3 обозначается $w_{8,5}$. Тогда NET -вектор слоя N может быть выражен не как сумма произведений, а как произведение X и W. В векторном обозначении N=XW. Покомпонентным применением функции F к NET - вектору N получаем

выходной вектор O. Таким образом, для данного слоя вычислительный процесс описывается следующим выражением:

$$O=F(XW) \qquad (12.3)$$

Выходной вектор одного слоя является входным вектором для следующего, поэтому вычисление выходов последнего слоя требует применения уравнения (12.3) к каждому слою от входа сети к ее выходу.

Обратный проход. **Подстройка весов выходного слоя**. Так как для каждого нейрона выходного слоя задано целевое значение, то подстройка весов легко осуществляется с использованием дельта-правила. Внутренние слои называют "скрытыми слоями", для их выходов не имеется целевых значений для сравнения, поэтому обучение усложняется.

Рассмотрим процесс обучения для одного веса от нейрона **p** в скрытом слое **j** к нейрону **q** в выходном слое **k**. Выход нейрона слоя **k**, вычитаемый из целевого значения (Target), дает сигнал ошибки. Он умножается на производную сжимающей функции [OUT(1-OUT], вычисленную для этого нейрона слоя **k**, давая, таким образом, величину δ.

$$\delta = OUT(1 - OUT)(Target - OUT). \qquad (11.4)$$

Затем δ умножается на величину OUT нейрона j, из которого выходит рассматриваемый вес. Это произведение, в свою очередь, умножается на коэффициент скорости обучения η (обычно от 0,01 до 1,0), и результат прибавляется к весу. Такая же процедура выполняется для каждого веса от нейрона скрытого слоя к нейрону в выходном слое.

Следующие уравнения иллюстрируют это вычисление:

$$\Delta w_{pq,k} = \eta \delta_{q,k} OUT_{pj}, \qquad (12.5)$$

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k}, \qquad (12.6)$$

где $w_{pq,k}(n)$ — величина веса от нейрона **p** в скрытом слое **k** к нейрону **q** в выходном слое на шаге **n** (до коррекции); отметим, что индекс **k** относится к слою, в котором заканчивается данный вес (т. е. к слою, с которым он объединен); $w_{pq,k}(n+1)$ — величина веса на шаге n+1 (после коррекции); $\delta_{q,k}$ — величина δ для нейрона **q**, в выходном слое **k**; $OUT_{p,j}$ — величина OUT для нейрона **p** в скрытом слое **j**.

Подстройка весов скрытого слоя. Рассмотрим один нейрон в скрытом слое, предшествующем выходному слою. При проходе вперед этот нейрон передает свой выходной сигнал нейронам в выходном слое через соединяющие их веса. Во время обучения эти веса функционируют в обратном порядке, пропуская величину δ от выходного слоя назад к скрытому слою. Каждый из этих весов умножается на величину δ нейрона, к которому он присоединен в выходном слое. Величина δ, необходимая для

нейрона скрытого слоя, получается суммированием всех таких произведений и умножением на производную сжимающей функции (рисунок 12.4):

$$\delta_{q,k} = OUT_{p,j}(1 - OUT_{p,j}) \left[\sum_{q} \delta_{q,k} w_{pq,k} \right] . \quad (12.7)$$

Когда значение δ получено, веса, питающие первый скрытый уровень, могут быть подкорректированы с помощью уравнений (12.5) и (12.6), где индексы модифицируются в соответствии со слоем.

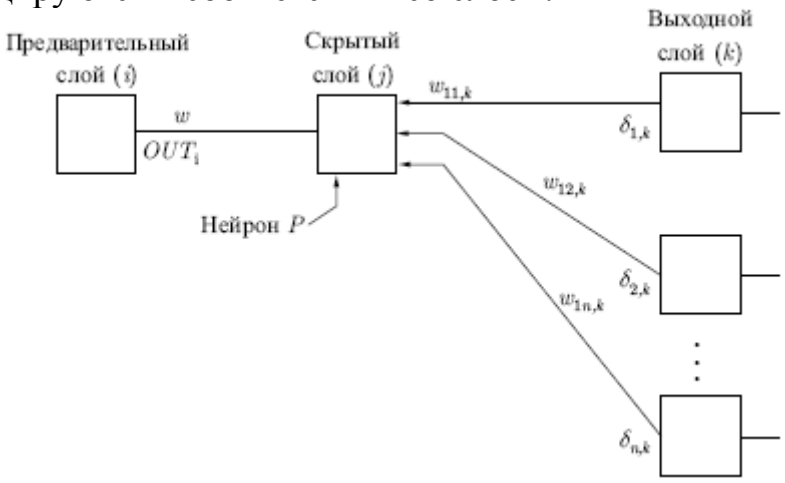


Рисунок 12.4 – Слои нейронной сети

Для каждого нейрона в данном скрытом слое должно быть вычислено δ и подстроены все веса, ассоциированные с этим слоем. Этот процесс повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы.

С помощью векторных обозначений операция обратного распространения ошибки может быть записана значительно компактнее. Обозначим множество величин δ выходного слоя через D_k и множество весов выходного слоя как массив W_k. Чтобы получить D_j, δ -вектор выходного слоя, достаточно следующих двух операций:

1. Умножить о-вектор выходного слоя D_k на транспонированную матрицу весов W'_k, соединяющую скрытый уровень с выходным уровнем.
2. Умножить каждую компоненту полученного произведения на производную сжимающей функции соответствующего нейрона в скрытом слое.

Добавление нейронного смещения. Во многих случаях желательно наделять каждый нейрон обучаемым смещением. Это позволяет сдвигать начало отсчета логистической функции, давая эффект, аналогичный подстройке порога персептронного нейрона, и приводит к ускорению процесса обучения. Такая возможность может быть легко введена в обучающий алгоритм с помощью добавляемого к каждому нейрону веса,

который присоединен к+1. Этот вес обучается так же, как и все остальные веса, за исключением того, что подаваемый на него сигнал всегда равен +1, а не выходу нейрона предыдущего слоя.

Импульс. Существует метод ускорения обучения для алгоритма обратного распространения, увеличивающий также устойчивость процесса. Этот метод, названный импульсом, заключается в добавлении к коррекции веса члена, пропорционального величине предыдущего изменения веса. Как только происходит коррекция, она "запоминается" и служит для модификации всех последующих коррекций. Уравнения коррекции модифицируются следующим образом:

$$\Delta w_{pq,k}(n+1) = \eta \delta_{q,k} OUT_{p,j} + \alpha \Delta w_{pq,k}(n),$$

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k}(n+1),$$

где α — коэффициент импульса, который обычно устанавливается около 0,9.

Используя метод импульса, сеть стремится идти по дну "узких оврагов" поверхности ошибки (если таковые имеются), а не двигаться "от склона к склону". Этот метод, по-видимому, хорошо работает на некоторых задачах, но дает слабый или даже отрицательный эффект на других.

Существует сходный метод, основанный на экспоненциальном сглаживании, который может иметь преимущество в ряде приложений.

$$\Delta w_{pq,k}(n+1) = (1-\alpha)\delta_{q,k} OUT_{p,j} + \alpha \Delta w_{pq,k}(n).$$

Затем вычисляется изменение веса

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \eta \Delta w_{pq,k}(n+1),$$

где α — коэффициент сглаживания, варьируемый в диапазоне от 0,0 до 1,0. Если α равен 1,0, то новая коррекция игнорируется и повторяется предыдущая. В области между 0 и 1 коррекция веса сглаживается величиной, пропорциональной α. По-прежнему, η является коэффициентом скорости обучения, служащим для управления средней величиной изменения веса.

Вопросы для повторения и закрепления материала

1. Для чего предназначена процедура обратного распространения?
2. Опишите алгоритм процедуры обратного распространения?
3. Что такое многослойный персептрон?
4. Смысл прохода вперед?
5. Смысл обратного прохода?
6. Каким образом осуществляется подстройка весов скрытого слоя?
7. Зачем используется нейронное смещение?

Задания для самостоятельной работы

1. Разработайте персептронную сеть процесса из будущей профессиональной деятельности.

2. Произведите обучение разработанной сети?

Тема 13. Анализ процедуры обратного распространения

Цель: провести анализ процедуры обратного распространения.

Задачи:

1. Рассмотреть понятия переобучение и обобщение.
2. Рассмотреть принципы отбора данных.
3. Изучить этапы обучения многослойного персептрона.
4. Ознакомиться с предостережениями, вызванными процедурой обратного распространения.

13.1. Переобучение и обобщение

Одна из наиболее серьезных трудностей алгоритма обратного распространения заключается в том, что таким образом мы минимизируем не ту ошибку, которую на самом деле нужно минимизировать, — ошибку, которую можно ожидать от сети, когда ей будут подаваться совершенно новые наблюдения. Иначе говоря, мы хотели бы, чтобы нейронная сеть обладала способностью **обобщать** результат на новые наблюдения. В действительности, сеть обучается минимизировать ошибку на обучающем множестве, и в отсутствие идеального и бесконечно большого обучающего множества это совсем не то же самое, что минимизировать "настоящую" ошибку на поверхности ошибок в заранее неизвестной модели явления.

Сильнее всего это различие проявляется в проблеме переобучения, или **слишком близкой подгонки.** Это явление проще будет продемонстрировать не для нейронной сети, а на примере аппроксимации посредством полиномов, — при этом суть явления абсолютно та же.

Полином (или многочлен) — это выражение, содержащее только константы и целые степени независимой переменной. Графики полиномов могут иметь различную форму, причем, чем выше степень многочлена (и, тем самым, чем больше членов в него входит), тем более сложной может быть эта форма. Если у нас есть некоторые данные, мы можем попробовать подогнать к ним полиномиальную кривую (модель) и получить, таким образом, объяснение для имеющейся зависимости. Наши данные могут быть зашумлены, поэтому нельзя считать, что самая лучшая модель задается кривой, которая в точности проходит через все имеющиеся точки. Полином низкого порядка может быть недостаточно гибким средством для аппроксимации данных, в то время как полином высокого порядка может оказаться чересчур гибким и будет точно следовать данным, принимая при

этом форму замысловатую и не имеющую никакого отношения к реальной зависимости.

У нейронной сети проблема точно такая же. Сети с большим числом весов моделируют более сложные функции и, следовательно, склонны к переобучению. Сеть же с небольшим числом весов может оказаться недостаточно гибкой для того, чтобы смоделировать имеющуюся зависимость. Например, сеть без промежуточных слоев моделирует обычную линейную функцию.

Как же выбрать "правильную" степень сложности для сети? Почти всегда более сложная сеть дает меньшую ошибку, но это может свидетельствовать не о хорошем качестве модели, а о переобучении. Выход состоит в том, чтобы использовать механизм контрольной **кросс-проверки**. Мы резервируем часть обучающих наблюдений и не используем их в обучении по алгоритму обратного распространения. Вместо этого, по мере работы алгоритма, они используются для независимого контроля результата. В самом начале работы ошибка сети на обучающем и контрольном множестве будет одинаковой (если они существенно отличаются, то, вероятно, разбиение всех наблюдений на два множества было неоднородно). По мере того как сеть обучается, ошибка обучения, естественно, убывает, и, пока обучение уменьшает действительную функцию ошибок, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, значит, сеть начала слишком близко аппроксимировать данные и обучение следует остановить. Это явление чересчур точной аппроксимации в процессе обучения и называется **переобучением**. Если такое случилось, то обычно советуют уменьшить число скрытых элементов и/или слоев, ибо сеть является слишком мощной для данной задачи. Если же сеть, наоборот, была взята недостаточно богатой для того, чтобы моделировать имеющуюся зависимость, то переобучения, скорее всего, не произойдет и обе ошибки — обучения и проверки — не примут достаточно малое значение.

Описанные проблемы с локальными минимумами и выбором размера сети приводят к тому, что при практической работе с нейронными сетями, как правило, приходится экспериментировать с большим числом различных сетей, порой обучая каждую из них несколько раз (чтобы не быть введенным в заблуждение локальными минимумами) и сравнивая полученные результаты. Главным показателем качества результата является здесь контрольная ошибка. В соответствии с общенаучным принципом, согласно которому при прочих равных следует предпочесть более простую модель, имеет смысл из двух сетей с приблизительно равными ошибками контроля выбрать ту, которая меньше.

Необходимость многократных экспериментов ведет к тому, что контрольное множество начинает играть ключевую роль в выборе модели, то есть становится частью процесса обучения. Тем самым ослабляется его роль

как независимого критерия качества модели — при большом числе экспериментов есть риск выбрать "удачную" сеть, дающую хороший результат на контрольном множестве. Для того чтобы придать окончательной модели должную надежность, часто (по крайней мере, когда объем обучающих данных это позволяет) поступают так: резервируют еще одно, тестовое множество наблюдений. Итоговая модель тестируется на данных из этого множества, чтобы убедиться, что результаты, достигнутые на обучающем и контрольном множествах, реальны, а не являются артефактами процесса обучения. Разумеется, для того чтобы соответствовать своей роли, тестовое множество должно быть использовано только один раз: если его использовать повторно для корректировки процесса обучения, то оно фактически превратится в контрольное множество.

13.2. Отбор данных

На всех предыдущих этапах мы постоянно опирались на одно предположение, а именно: обучающее, контрольное и тестовое множества должны быть репрезентативными (представительными) с точки зрения существа задачи (более того, эти множества должны быть репрезентативны каждое в отдельности). Известное изречение программистов "garbage in, garbage out" ("мусор на входе — мусор на выходе") нигде не справедливо в такой степени, как при нейросетевом моделировании. Если обучающие данные не репрезентативны, то модель, как минимум, будет не очень хорошей, а в худшем случае — бесполезной. Имеет смысл перечислить ряд причин, которые ухудшают качество обучающего множества.

Будущее не похоже на прошлое. Обычно в качестве обучающих берутся исторические данные. Если обстоятельства изменились, то закономерности, имевшие место в прошлом, могут больше не действовать.

Следует учесть все возможности. Нейронная сеть может обучаться только на тех данных, которыми она располагает. Предположим, что лица с годовым доходом более $100000 имеют высокий кредитный риск, а обучающее множество не содержало лиц с доходом более $40000 в год. Тогда едва ли можно ожидать от сети правильного решения в совершенно новой для нее ситуации.

Сеть обучается тому, чему проще всего обучиться. Классическим (возможно, вымышленным) примером является система машинного зрения, предназначенная для автоматического распознавания танков. Сеть обучалась на ста картинках, содержащих изображения танков, и на ста других картинках, где танков не было. Был достигнут стопроцентно "правильный" результат. Но когда на вход сети были поданы новые данные, она безнадежно провалилась. В чем же была причина? Выяснилось, что фотографии с танками были сделаны в пасмурную, дождливую погоду, а фотографии без танков — в солнечный день. Сеть научилась улавливать (очевидную)

разницу в общей освещенности. Чтобы сеть могла результативно работать, ее следовало обучать на данных, где присутствовали бы все погодные условия и типы освещения, при которых сеть будут реально использовать, — и это не говоря еще о рельефе местности, угле и дистанции съемки и т.д.

Несбалансированный набор данных. Коль скоро сеть минимизирует общую погрешность, важное значение приобретают пропорции, в которых представлены данные различных типов. Сеть, обученная на 900 хороших и 100 плохих примерах, будет искажать результат в пользу хороших наблюдений, поскольку это позволит алгоритму уменьшить общую погрешность (которая определяется в основном хорошими случаями). Если в реальной популяции хорошие и плохие объекты представлены в другой пропорции, то результаты, выдаваемые сетью, могут оказаться неверными. Хорошим примером служит задача выявления заболеваний. Пусть, например, при обычных обследованиях в среднем 90% человек оказываются здоровыми. Сеть обучается на имеющихся данных, в которых пропорция здоровые/больные равна 90/10. Затем она применяется для диагностики пациентов с определенным жалобами, среди которых это соотношение уже 50/50. В этом случае сеть будет ставить диагноз чересчур осторожно и не распознает заболевание у некоторых больных. Если же, наоборот, сеть обучить на данных "с жалобами", а затем протестировать на "обычных" данных, то она будет выдавать повышенное число неправильных диагнозов о наличии заболевания. В таких ситуациях обучающие данные нужно скорректировать так, чтобы были учтены различия в распределении (например, можно повторять редкие наблюдения или удалить часто встречающиеся), или же видоизменить решения, выдаваемые сетью, посредством **матрицы потерь**. Как правило, лучше всего постараться равномерно представить наблюдения различных типов и соответственно этому интерпретировать результаты, которые выдает сеть.

13.3. Как обучается многослойный персептрон

Мы сможем лучше понять, как устроен и как обучается многослойный персептрон, если выясним, какие функции он способен моделировать. Вспомним, что уровнем активации элемента называется взвешенная сумма его входов с добавленным к ней пороговым значением. Таким образом, уровень активации представляет собой простую линейную функцию входов. Эта активация затем преобразуется с помощью сигмоидной (имеющей S-образную форму) кривой.

Комбинация линейной функции нескольких переменных и скалярной сигмовидной функции приводит к характерному профилю "сигмовидного склона", который выдает элемент первого промежуточного слоя. На рисунке 13.1 соответствующая поверхность изображена в виде функции двух

входных переменных. Элемент с большим числом входов выдает многомерный аналог такой поверхности.

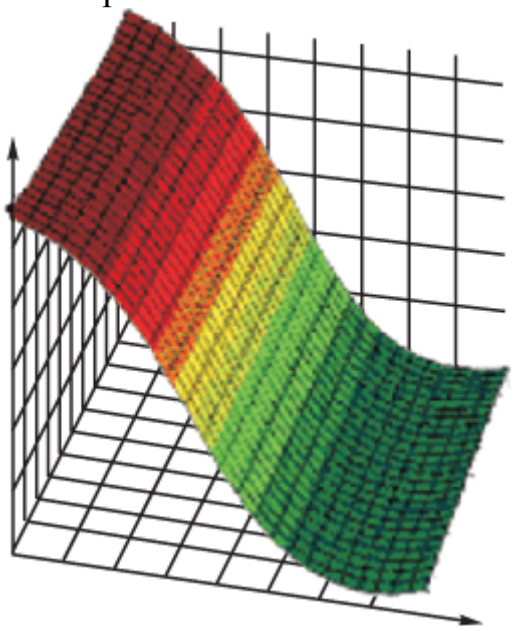


Рисунок 13.1 – Поверхность функции двух входных переменных

При изменении весов и порогов меняется и поверхность отклика; может меняться как ориентация всей поверхности, так и крутизна склона — большим значениям весов соответствует более крутой склон. Так, например, если увеличить все веса в два раза, то ориентация не изменится, а наклон будет более крутым. В многослойной сети подобные функции отклика комбинируются друг с другом с помощью последовательного взятия их линейных комбинаций и применения нелинейных функций активации. На рисунке 13.2 изображена типичная поверхность отклика для сети с одним промежуточным слоем, состоящим из двух элементов, и одним выходным элементом, для классической задачи "исключающего или". Две разных сигмоидных поверхности объединены в одну поверхность, имеющую форму буквы "U".

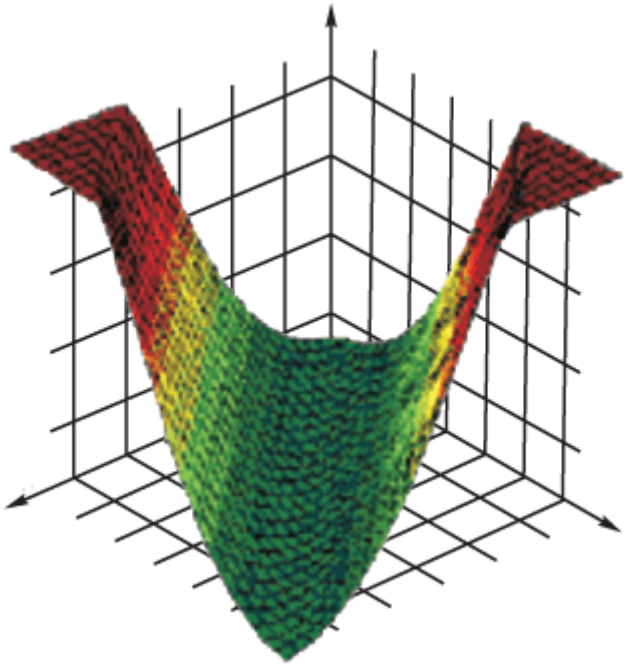


Рисунок 13.2 - Поверхность отклика для сети с одним промежуточным слоем

Перед началом обучения сети весам и порогам случайным образом присваиваются небольшие по величине начальные значения. Тем самым, отклики отдельных элементов сети имеют малый наклон и ориентированы хаотично — фактически они не связаны друг с другом. По мере того, как происходит обучение, поверхности отклика элементов сети вращаются и сдвигаются в нужное положение, а значения весов увеличиваются, поскольку они должны моделировать отдельные участки целевой поверхности отклика.

В задачах классификации выходной элемент должен выдавать сильный сигнал в случае, если данное наблюдение принадлежит к интересующему нас классу, и слабый — в противоположном случае. Иначе говоря, этот элемент должен стремиться смоделировать функцию, равную единице в области пространства объектов, где располагаются объекты из нужного класса, и равную нулю вне этой области. Такая конструкция известна как **дискриминантная функция** в задачах распознавания. "Идеальная" дискриминантная функция должна иметь плоскую структуру: точки соответствующей поверхности будут располагаться либо на нулевом уровне, либо на высоте "единица".

Если сеть не содержит скрытых элементов, то на выходе она может моделировать только одинарный "сигмовидный склон": точки, находящиеся по одну его сторону, располагаются низко, по другую — высоко. При этом всегда будет существовать область между ними (на склоне), где высота принимает промежуточные значения, но по мере увеличения весов эта область будет сужаться.

Такой сигмовидный склон фактически работает как линейная дискриминантная функция. Точки, лежащие по одну сторону склона, классифицируются как принадлежащие нужному классу, а лежащие по другую сторону — как не принадлежащие. Следовательно, сеть без скрытых слоев может служить классификатором только в линейно-отделимых задачах: когда можно провести линию (или, в случае более высоких размерностей, гиперплоскость), разделяющую точки в пространстве признаков.

Сеть, содержащая один промежуточный слой, строит несколько сигмоидных склонов, — по одному для каждого скрытого элемента, — и затем выходной элемент комбинирует из них "возвышенность". Эта возвышенность получается выпуклой, т.е. не содержащей впадин. При этом в некоторых направлениях она может уходить на бесконечность (как длинный полуостров). Подобная сеть может моделировать большинство реальных задач классификации.

Сеть с двумя промежуточными слоями строит комбинацию из нескольких таких возвышенностей. Их будет столько, сколько элементов во втором слое, и у каждой из них будет столько сторон, сколько элементов было в первом скрытом слое. После несложного размышления делаем вывод, что, используя достаточное число таких возвышенностей, можно воспроизвести поверхность любой формы — в том числе с впадинами и вогнутостями.

Как следствие наших рассмотрений мы получаем, что, теоретически, для моделирования любой задачи достаточно многослойного персептрона с двумя промежуточными слоями (в точной формулировке этот результат известен как теорема Колмогорова). При этом может оказаться, что для решения некоторой конкретной задачи будет более простой и удобной сеть с еще большим числом слоев. Однако для решения большинства практических задач достаточно всего одного промежуточного слоя, два слоя применяются как резерв в особых случаях, а сети с тремя слоями практически не применяются.

В задачах классификации очень важно понять, как следует интерпретировать те точки, которые попали на склон или лежат близко от него. Стандартный подход заключается в том, чтобы для пороговых значений установить некоторые доверительные пределы (принятия или отвержения), которые должны быть достигнуты, чтобы данный элемент считался "принявшим решение". Например, если установлены пороги принятия/отвержения 0,95/0.05, то при уровне выходного сигнала выше 0,95 элемент считается активным, при уровне ниже 0,05 — неактивным, а в промежутке — "неопределенным". Имеется и более тонкий (и, вероятно, более полезный) способ интерпретировать уровни выходного сигнала: считать их вероятностями. В этом случае сеть выдает несколько большую информацию, чем просто "да/нет": она сообщает нам, насколько (в

некотором формальном смысле) мы можем доверять ее решению. При этом, однако, вероятностная интерпретация обоснована только в том случае, если выполняются определенные предположения о распределении исходных данных (конкретно, что данные являются выборкой из некоторого распределения, принадлежащего к семейству экспоненциальных распределений). Здесь, как и ранее, может быть принято решение по классификации, но, кроме того, вероятностная интерпретация позволяет ввести концепцию "решения с минимальными затратами".

13.4. Предостережения

Несмотря на многочисленные успешные применения обратного распространения, оно не является панацеей. Больше всего неприятностей доставляет неопределенно долгий процесс обучения. В сложных задачах для обучения сети могут потребоваться дни или даже недели, она может и вообще не обучиться. Длительное время обучения может быть результатом неоптимального выбора длины шага. Неудачи в обучении обычно возникают по двум причинам: паралича сети и попадания в локальный минимум.

Паралич сети. В процессе обучения сети значения весов могут в результате коррекции стать очень большими величинами. Это может привести к тому, что все или большинство нейронов будут функционировать при очень больших значениях OUT, в области, где производная сжимающей функции очень мала. Так как посылаемая обратно в процессе обучения ошибка пропорциональна этой производной, то процесс обучения может практически замереть. Теоретически эта проблема изучена плохо. Обычно пытаются уменьшать размера шага η, но это увеличивает время обучения. Различные эвристики использовались для предохранения от паралича или для восстановления после него, но пока что они могут рассматриваться лишь как экспериментальные.

Локальные минимумы. В прошлой лекции было описано, как с помощью алгоритма обратного распространения осуществляется градиентный спуск по поверхности ошибок. Короче говоря, происходит следующее: в данной точке поверхности находится направление скорейшего спуска, затем делается прыжок вниз на расстояние, пропорциональное коэффициенту скорости обучения и крутизне склона, при этом учитывается инерция, то есть стремление сохранить прежнее направление движения. Можно сказать, что метод ведет себя как слепой кенгуру — каждый раз прыгает в направлении, которое кажется ему наилучшим. На самом деле, шаг спуска вычисляется отдельно для всех обучающих наблюдений, взятых в случайном порядке, но в результате получается достаточно хорошая аппроксимация спуска по совокупной поверхности ошибок. Существуют и другие алгоритмы обучения, однако все они используют ту или иную стратегию скорейшего продвижения к точке минимума.

Обратное распространение использует разновидность градиентного спуска, т. е. осуществляет спуск вниз по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму. Поверхность ошибки сложной сети сильно изрезана и состоит из холмов, долин, складок и оврагов в пространстве высокой размерности. Сеть может попасть в локальный минимум (неглубокую долину), когда рядом имеется гораздо более глубокий минимум. В точке локального минимума все направления ведут вверх и сеть неспособна из него выбраться. Статистические методы обучения могут помочь избежать этой ловушки, но они медленны. П.Д. Вассерман предложил метод, объединяющий статистические методы машины Коши с градиентным спуском обратного распространения и приводящий к системе, которая находит глобальный минимум, сохраняя высокую скорость обратного распространения.

Размер шага. Внимательный разбор доказательства сходимости показывает, что коррекции весов предполагаются бесконечно малыми. Ясно, что это неосуществимо на практике, так как ведет к бесконечному времени обучения. Размер шага должен браться конечным, и при определении его приходится полагаться только на опыт. Если размер шага очень мал, то сходимость слишком медленная, если же очень велик, то может возникнуть паралич или постоянная неустойчивость. П.Д. Вассерман описал адаптивный алгоритм выбора шага, автоматически корректирующий размер шага в процессе обучения.

Временная неустойчивость. Если сеть учится распознавать буквы, то нет смысла учить "Б", если при этом забывается "А". Процесс обучения должен быть таким, чтобы сеть обучалась на всем обучающем множестве без пропусков того, что уже выучено. В доказательстве сходимости это условие выполнено, но требуется также, чтобы сети предъявлялись все векторы обучающего множества, прежде чем выполняется коррекция весов. Необходимые изменения весов должны вычисляться на всем множестве, что требует дополнительной памяти; после ряда таких обучающих циклов веса сойдутся к минимальной ошибке. Этот метод может оказаться бесполезным, если сеть находится в постоянно меняющейся внешней среде, так что второй раз один и тот же вектор может уже не повториться. В этом случае процесс обучения может никогда не сойтись, бесцельно блуждая или сильно осциллируя. В этом смысле обратное распространение не похоже на биологические системы.

13.3. Дальнейшие алгоритмические разработки

Многими исследователями были предложены методы улучшения и обобщения описанного выше основного алгоритма обратного распространения. Некоторые из этих подходов могут оказаться

действительно фундаментальными, другие же со временем исчезнут. Перечислим некоторые из наиболее многообещающих разработок.

Метод ускорения сходимости алгоритма обратного распространения. Названный обратным распространением второго порядка, он использует вторые производные для более точной оценки требуемой коррекции весов. Показано, что этот алгоритм оптимален в том смысле, что невозможно улучшить оценку, даже используя производные более высокого порядка. Метод требует дополнительных вычислений по сравнению с обратным распространением первого порядка, и необходимы дальнейшие эксперименты для доказательства оправданности этих затрат.

Метод улучшения характеристик обучения сетей обратного распространения. Указывается, что общепринятый от 0 до 1 динамический диапазон входов и выходов скрытых нейронов не оптимален. Так как величина коррекции веса $\Delta w_{pq,k}$ пропорциональна выходному уровню нейрона, порождающего $OUT_{p,j}$, то нулевой уровень ведет к тому, что вес не меняется. При двоичных входных векторах половина входов в среднем будет равна нулю, и веса, с которыми они связаны, не будут обучаться! Решение состоит в приведении входов к значениям $\pm 1/2$ и добавлении смещения к сжимающей функции, чтобы она также принимала значения $\pm 1/2.$ Новая сжимающая функция выглядит следующим образом:

$$OUT = -1/2 + \frac{1}{1 + e^{-NET}}.$$

С помощью таких простых средств время сходимости сокращается в среднем от 30 до 50%. Это один из примеров практической модификации, существенно улучшающей характеристику алгоритма.

Методика обратного распространения применима и к сетям с обратными связями, т. е. к таким сетям, у которых выходы подаются через обратную связь на входы. Как показано, обучение в подобных системах может быть очень быстрым и критерии устойчивости легко удовлетворяются.

Обратное распространение было применено в широкой сфере прикладных исследований. Некоторые из них описываются здесь, чтобы продемонстрировать богатые возможности этого метода.

Фирма NEC в Японии объявила недавно, что обратное распространение было ею использовано для визуального распознавания букв, причем точность превысила 99%. Это улучшение было достигнуто с помощью комбинации обычных алгоритмов с сетью обратного распространения, обеспечивающей дополнительную проверку.

Достигнут впечатляющий успех с Net-Talk системой, которая превращает печатный английский текст в высококачественную речь. Магнитофонная запись процесса обучения сильно напоминает звуки голоса ребенка на разных этапах обучения речи.

Обратное распространение также использовалось в машинном распознавании рукописных английских слов. Буквы, нормализованные по размеру, наносились на сетку, и брались проекции линий, пересекающих квадраты сетки. Эти проекции служили затем входами для сети обратного распространения. Сообщалось о точности 99,7% при использовании словарного фильтра.

Обратное распространение успешно применяется при сжатии изображений, когда образы представляются одним битом на пиксель, что явилось восьмикратным улучшением по сравнению с входными данными.

Вопросы для повторения и закрепления материала

1. Что такое переобучение сети?
2. Что подразумевается под понятием обобщение?
3. Каким образом отбираются данные для обучения?
4. Что такое несбалансированный набор данных?
5. Как обучается многослойный персептрон?
6. Что такое паралич сети?
7. Охарактеризуйте проблемы, связанные с локальными минимумами?
8. Почему важно определиться с шагом?
9. Что такое временная неустойчивость?
10. Опишите разработки в области обучения персептрона?

Задания для самостоятельной работы

1. Проведите анализ литературы на предмет детального обзора методов обучения нейронных сетей.

Тема 14. Сети встречного распространения

Цель: рассмотреть сети встречного распространения.

Задачи:

1. Ознакомиться с общими понятиями сетей встречного распространения.
2. Рассмотреть структуру сети встречного распространения.
3. Ознакомиться с режимом нормального функционирования сети встречного распространения.
4. Изучить этапы обучения слоя Конохена.
5. Рассмотреть процесс обучения слоя Гроссберга.

14.1. Введение в сети встречного распространения

По своим возможностям сети встречного распространения превосходят возможности однослойных сетей. Время же их обучения, по сравнению с обратным распространением, может уменьшаться в сто раз. **Встречное распространение** не настолько общее, как обратное распространение, но оно может давать решение в тех приложениях, где долгая обучающая процедура невозможна. Будет показано, что, помимо преодоления ограничений других сетей, **встречное распространение** обладает собственными интересными и полезными свойствами.

Во **встречном распространении** объединены два хорошо известных алгоритма: самоорганизующаяся карта Кохонена и звезда Гроссберга. При этом появляются свойства, которых нет ни у одного из них в отдельности.

Методы, которые, подобно **встречному распространению**, объединяют различные сетевые парадигмы как строительные блоки, могут привести к сетям, более близким по архитектуре к мозгу, чем любые другие однородные структуры. Похоже, что в естественном мозге именно каскадные соединения модулей различной специализации позволяют выполнять требуемые вычисления.

Сеть встречного распространения функционирует подобно столу справок, способному к обобщению. В процессе обучения входные векторы ассоциируются с соответствующими выходными векторами; они могут быть двоичными, состоящими из нулей и единиц, или непрерывными. Когда сеть обучена, приложение входного вектора приводит к требуемому выходному вектору. Обобщающая способность сети позволяет получать правильный выход даже при приложении входного вектора, который является неполным или слегка неверным. Таким образом, возможно использовать данную сеть для распознавания образов, восстановления образов и усиления сигналов.

14.2. Структура сети

На рисунке 14.1 показана упрощенная версия прямого действия сети встречного распространения. Здесь иллюстрируются функциональные свойства этой парадигмы.

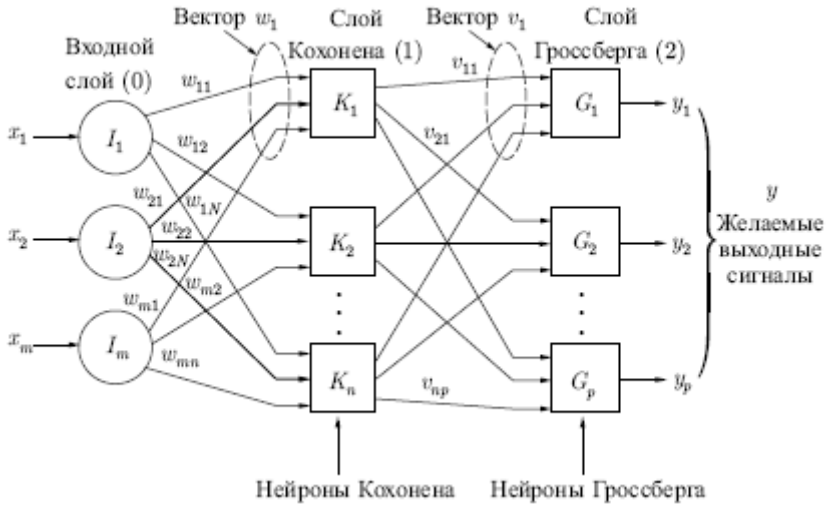


Рисунок 14.1 – Упрощенное прямое действие сети встречного распространения

Нейроны слоя 0 (показанные кружками) служат лишь точками разветвления и не выполняют вычислений. Каждый нейрон слоя 0 соединен с каждым нейроном слоя 1 (называемого **слоем Кохонена**) отдельным весом w_{mn}. Эти веса в целом рассматриваются как матрица весов W Аналогично, каждый нейрон в слое Кохонена (слое 1) соединен с каждым нейроном в слое Гроссберга (слое 2) весом v_{np}. Эти веса образуют матрицу весов V. Все это весьма напоминает другие сети, различие, однако, в операциях, выполняемых нейронами Кохонена и Гроссберга.

Как и многие другие сети, встречное распространение функционирует в двух режимах: в нормальном режиме, при котором принимается входной вектор X и выдается выходной вектор Y, и в режиме обучения, при котором подается входной вектор и веса корректируются, чтобы дать требуемый выходной вектор.

14.3. Нормальное функционирование

Слои Кохонена. В своей простейшей форме слой Кохонена функционирует в духе "победитель забирает все", т.е. для данного входного вектора один и только один нейрон Кохонена выдает на выходе логическую единицу, а все остальные выдают ноль. Нейроны Кохонена можно воспринимать как набор электрических лампочек, и для любого входного вектора "загорается" одна из них.

Ассоциированное с нейронами Кохонена множество весов связывает каждый нейрон с каждым входом. Например, на рисунке 14.1 нейрон Кохонена K_1 имеет веса $w_{11}, w_{21}, \ldots, w_{m1}$, составляющие весовой вектор W_1. Они соединяются через входной слой с входными сигналами $x_1, x_2, \ldots, x_m$, составляющими входной вектор X. Подобно нейронам

большинства сетей, выход NET каждого нейрона Кохонена является просто суммой взвешенных входов. Это может быть выражено следующим образом:

$$NET_j = \sum_i x_i w_{ij}$$

где NET_j — это выход NET нейрона Кохонена j, или, в векторной записи,

N=XW,

где N — вектор выходов NET слоя Кохонена.

Нейрон Кохонена с максимальным значением NET является "победителем". Его выход равен единице, у остальных он равен нулю.

Слой Гроссберга. Слой Гроссберга функционирует в сходной манере. Его выход NET является взвешенной суммой выходов k_1, k_2, . . ., k_n слоя Кохонена, образующих вектор K. Вектор соединяющих весов, обозначенный через V, состоит из весов $v_{11}, v_{21}, \ldots, v_{\text{np}\cdot}$ Тогда выход NET каждого нейрона Гроссберга есть

$$NET_j = \sum_i k_i v_{ij},$$

где NET_j — выход j-го нейрона Гроссберга, или, в векторной форме, Y=KV,

где Y — выходной вектор слоя Гроссберга, K — выходной вектор слоя Кохонена, V — матрица весов слоя Гроссберга.

Если слой Кохонена функционирует таким образом, что лишь у одного нейрона величина NET равна единице, а у остальных равна нулю, то всего один элемент вектора K отличен от нуля и вычисления очень просты. Фактически каждый нейрон слоя Гроссберга лишь выдает величину веса, который связывает этот нейрон с единственным ненулевым нейроном Кохонена.

14.4. Обучение слоя Кохонена

Слой Кохонена классифицирует входные векторы в группы схожих. Это достигается с помощью такой подстройки весов слоя Кохонена, что близкие входные векторы активируют один и тот же нейрон данного слоя. Затем задачей слоя Гроссберга является получение требуемых выходов.

Обучение Кохонена является самообучением, протекающим без учителя. Поэтому трудно (и не нужно) предсказывать, какой именно нейрон Кохонена будет активироваться для заданного входного вектора. Необходимо лишь гарантированно добиться, чтобы в результате обучения разделялись несхожие входные векторы.

14.4.1. Предварительная обработка входных векторов

Весьма желательно (хотя и не обязательно) нормализовать входные векторы перед тем, как предъявлять их сети. Операция выполняется с помощью деления каждой компоненты входного вектора на длину вектора. Эта длина находится извлечением квадратного корня из суммы квадратов компонент вектора. В алгебраической записи

$$s_i' = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}}. \qquad (14.1)$$

Таким образом, входной вектор превращается в единичный вектор с тем же самым направлением, т.е. в вектор единичной длины в n-мерном пространстве.

Уравнение (14.1) обобщает хорошо известный случай двух измерений, когда длина вектора равна гипотенузе прямоугольного треугольника, образованного его x и y компонентами, как это следует из известной теоремы Пифагора. На рисунке 14.2 такой двумерный вектор V представлен в координатах x-y, причем координата x равна четырем, а координата y — трем. Квадратный корень из суммы квадратов этих компонент равен пяти. Деление каждой компоненты V на пять дает вектор V с компонентами 4/5 и 3/5, где $\mathbf{V}'$ указывает в том же направлении, что и V, но имеет единичную длину.

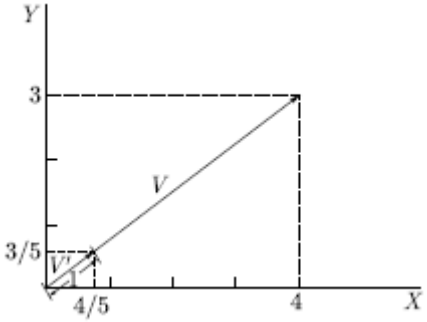


Рисунок 14.2 - Двумерный вектор NET

На рисунке 14.3 показано несколько единичных векторов. Они оканчиваются в точках единичной окружности (окружности единичного радиуса), а это происходит, когда у сети лишь два входа. В случае трех входов векторы представлялись бы стрелками, оканчивающимися на поверхности единичной сферы. Такие представления могут быть перенесены на сети, имеющие произвольное число входов, где каждый входной вектор является стрелкой, оканчивающейся на поверхности единичной гиперсферы (полезной абстракцией, хотя и не допускающей непосредственной визуализации).

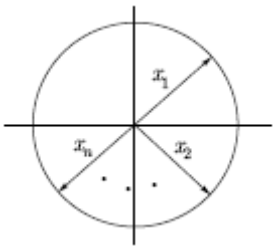


Рисунок 14.3 - Несколько единичных векторов

При обучении слоя Кохонена на вход подается входной вектор и вычисляются его скалярные произведения с векторами весов, связанными со всеми нейронами Кохонена. Нейрон с максимальным значением скалярного произведения объявляется "победителем", и его веса подстраиваются. Так как скалярное произведение, используемое для вычисления величин NET, является мерой сходства между входным вектором и вектором весов, то процесс обучения состоит в выборе нейрона Кохонена с весовым вектором, наиболее близким к входному вектору, и дальнейшем приближении весового вектора к входному. Снова отметим, что процесс является самообучением, выполняемым без учителя. Сеть самоорганизуется таким образом, что данный нейрон Кохонена имеет максимальный выход для данного входного вектора. Уравнение, описывающее процесс обучения, имеет следующий вид:

$$w_{\text{н}} = w_c + \alpha(x - w_c),$$

где $w_{\text{н}}$ — новое значение веса, соединяющего входную компоненту **x** с выигравшим нейроном; w_c — предыдущее значение этого веса; α — коэффициент скорости обучения, который может варьироваться в процессе обучения.

Каждый вес, связанный с выигравшим нейроном Кохонена, изменяется пропорционально разности между его величиной и величиной входа, к которому он присоединен. Направление изменения минимизирует разность между весом и его входом.

На рисунке 14.4 этот процесс показан геометрически в двумерном виде. Сначала ищем вектор $X - W_c$, для этого проводится отрезок из конца W в конец X. Затем этот вектор укорачиваем умножением его на скалярную величину α, меньшую единицы, в результате чего получаем вектор изменения δ. Окончательно новый весовой вектор $W_{\text{н}}$ является отрезком, направленным из начала координат в конец вектора δ. Отсюда можно видеть, что эффект обучения состоит во вращении весового вектора в направлении входного вектора без существенного изменения его длины.

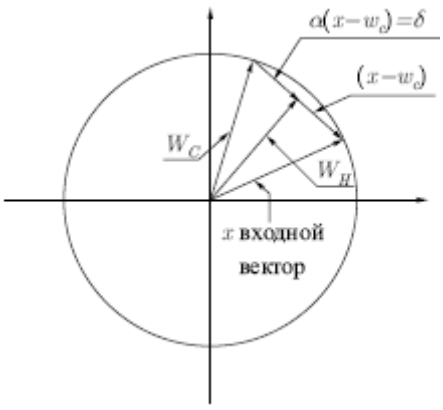


Рисунок 14.4 – Процесс обучения в двухмерном виде

Переменная α является коэффициентом скорости обучения, который вначале обычно равен $\sim 0,7$ и может постепенно уменьшаться в процессе обучения. Это позволяет делать большие начальные шаги для быстрого грубого обучения и меньшие шаги при подходе к окончательной величине.

Если бы с каждым нейроном Кохонена ассоциировался один входной вектор, то слой Кохонена мог бы быть обучен с помощью одного вычисления на вес. Веса нейрона-победителя приравнивались бы к компонентам обучающего вектора ($\alpha = 1$). Как правило, обучающее множество включает много сходных между собой входных векторов, и сеть должна быть обучена активировать один и тот же нейрон Кохонена для каждого из них. В этом случае веса этого нейрона должны вычисляться усреднением входных векторов, которые его активируют. Постепенное уменьшение величины α уменьшает воздействие каждого обучающего шага, и окончательное значение будет средней величиной от входных векторов, на которых происходит обучение. Таким образом, веса, ассоциированные с нейроном, примут значение вблизи "центра" входных векторов, для которых данный нейрон является "победителем".

14.4.2. Выбор начальных значений весовых векторов

Всем весам сети перед началом обучения следует придать начальные значения. Общепринятой практикой при работе с нейронными сетями является присваивание весам небольших случайных значений. При обучении слоя Кохонена случайно выбранные весовые векторы следует нормализовать. Окончательные значения весовых векторов после обучения совпадают с нормализованными входными векторами. Поэтому нормализация перед началом обучения приближает весовые векторы к их окончательным значениям, сокращая, таким образом, продолжительность обучающего процесса.

Рандомизация весов слоя Кохонена может породить серьезные проблемы при обучении, так как в результате весовые векторы распределяются равномерно по поверхности гиперсферы. Из-за того, что

входные векторы, как правило, распределены неравномерно и имеют тенденцию группироваться на относительно малой части поверхности гиперсферы, большинство весовых векторов будут так удалены от любого входного вектора, что они никогда не смогут дать наилучшее соответствие. Эти нейроны Кохонена будут всегда иметь нулевой выход и окажутся бесполезными. Более того, оставшихся весов, дающих наилучшие соответствия, может оказаться слишком мало, чтобы разделить входные векторы на классы, которые расположены близко друг к другу на поверхности гиперсферы.

Допустим, что имеется несколько множеств входных векторов, все эти множества сходные, но необходимо разделить их на различные классы. Сеть должна быть обучена активировать отдельный нейрон Кохонена для каждого класса. Если начальная плотность весовых векторов в окрестности обучающих векторов слишком мала, то, возможно, не удастся разделить сходные классы из-за того, что весовых векторов в интересующей нас окрестности не хватит, чтобы приписать по одному из них каждому классу входных векторов.

Наоборот, если несколько входных векторов получены незначительными изменениями из одного и того же образца и должны быть объединены в один класс, то они должны включать один и тот же нейрон Кохонена. Если же плотность весовых векторов очень высока вблизи группы слегка различных входных векторов, то каждый входной вектор может активировать отдельный нейрон Кохонена. Это не является катастрофой, так как слой Гроссберга может отобразить различные нейроны Кохонена в один и тот же выход, но это расточительная трата нейронов Кохонена.

Наиболее желательное решение будет таким: распределить весовые векторы в соответствии с плотностью входных векторов, подлежащих разделению, и для этого поместить больше весовых векторов в окрестности большого числа входных векторов. Конечно, на практике это невыполнимо, но существует несколько методов приближенного достижения тех же целей.

Одно из решений, известное под названием **метода выпуклой комбинации** (convex combination method), состоит в том, что все веса приравниваются к одной и той же величине

$$w_i = \frac{1}{\sqrt{n}},$$

где n — число входов и, следовательно, число компонент каждого весового вектора. Благодаря этому все весовые векторы совпадают и имеют единичную длину. Каждой же компоненте входа $\mathbf{X}$ придается значение

$$x_i = ax_i + \frac{1-a}{\sqrt{n}},$$

где n — число входов. В начале α очень мало, вследствие чего все входные векторы имеют длину, близкую к $1/\sqrt{n}$, и почти совпадают с

векторами весов. В процессе обучения сети α постепенно возрастает, приближаясь к единице. Это позволяет разделять входные векторы и окончательно приписывать им их истинные значения. Весовые векторы отслеживают один или небольшую группу входных векторов и в конце обучения дают требуемую картину выходов.

14.4.3. Примеры обучения

Рассмотрим примеры обучения сети Кохонена обычным методом и методом выпуклой комбинации. В первом методе будем выбирать равномерно распределенные случайные векторы весов (ядер классов). На рисунке 14.5 представлен пример обучения. Точками обозначены векторы x^p обучающего множества, кружками — векторы весовых коэффициентов.

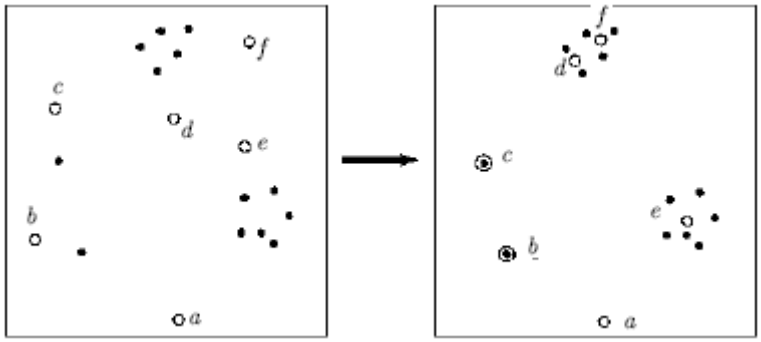


Рисунок 14.5 – Пример обучения сети Конохена обычным методом

Вектор весов нейрона a не обучается, т.к. ни для одного из векторов обучающего множества этот нейрон не получает максимального выхода. Кроме того, в области из шести обучающих векторов (справа внизу) оказывается всего один вектор весов нейрона **e**, что не соответствует высокой плотности обучающих векторов в этой области. Эти недостатки присущи обычному методу обучения сети Кохонена.

Разберем работу метода выпуклой комбинации. Последовательное изменение картины векторов и весов показано на рисунке 14.6.

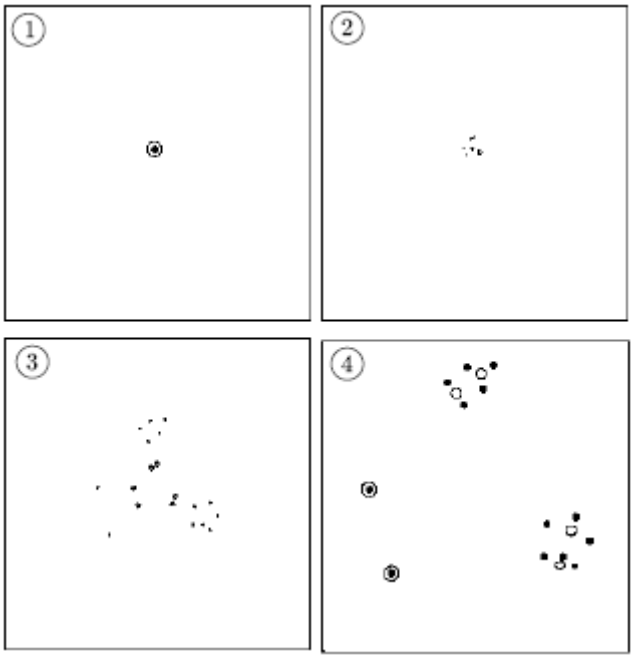


Рисунок 14.6 – Пример обучения сети Конохена методом выпуклой комбинации

На первой схеме все векторы весов и обучающего множества имеют одно и то же значение. По мере обучения обучающие векторы расходятся к своим истинным значениям, а векторы весов следуют за ними. В итоге в сети не остается необученных нейронов и плотность векторов весов соответствует плотности векторов обучающего множества. Однако метод выпуклой комбинации хорошо работает, но замедляет процесс обучения, так как весовые векторы подстраиваются к изменяющейся цели. Другой подход состоит в добавлении шума к входным векторам. Тем самым они подвергаются случайным изменениям, схватывая в конце концов весовой вектор. Этот метод также работоспособен, но еще более медленен, чем метод выпуклой комбинации.

Третий метод начинает работу со случайных весов, но на начальной стадии обучающего процесса подстраивает все веса, а не только связанные с выигравшим нейроном Кохонена. Тем самым весовые векторы перемещаются ближе к области входных векторов. В процессе обучения коррекция весов начинает производиться лишь для ближайших к победителю нейронов Кохонена. Этот радиус коррекции постепенно уменьшается, так что в конце корректируются только веса, связанные с выигравшим нейроном Кохонена.

Еще один метод наделяет каждый нейрон Кохонена "чувством справедливости". Если он становится победителем чаще своей "законной доли" (примерно 1/k, где **k** — число нейронов Кохонена), он временно увеличивает свой порог, что уменьшает его шансы на выигрыш, давая тем самым возможность обучаться и другим нейронам.

Во многих приложениях точность результата существенно зависит от распределения весов. К сожалению, эффективность различных решений исчерпывающим образом не оценена и остается проблемой, ожидающей своего решения.

14.4.4. Модификации алгоритма обучения

Чувство справедливости: чтобы не допустить отсутствие обучения по любому из нейронов, вводится "чувство справедливости". Если нейрон чаще других выигрывает "состязание", т.е. получает максимальный выход чаще, чем в 1 из M случаев, то его значение выхода искусственно уменьшается, чтобы дать возможность выиграть другим нейронам. Это включает все нейроны сети в процесс обучения.

Коррекция весов пропорционально выходу: в этой модификации корректируются веса не только выигравшего нейрона, но и всех остальных, пропорционально их нормированному выходу. Нормировка выполняется по максимальному значению выхода слоя или по его среднему значению. Этот метод также исключает "мертвые" нейроны и улучшает распределение плотности весов.

14.4.5. Режим интерполяции

До сих пор мы обсуждали алгоритм обучения, в котором для каждого входного вектора активировался только один нейрон Кохонена. Это называется **методом аккредитации.** Его точность ограничена, так как выход полностью является функцией лишь одного нейрона Кохонена.

В **методе интерполяции** целая группа нейронов Кохонена, имеющих максимальные выходы, может передавать свои выходные сигналы в слой Гроссберга. Число нейронов в такой группе должно выбираться в зависимости от задачи, и убедительных данных относительно оптимального размера группы не имеется. Как только группа определена, ее множество выходов NET рассматривается как вектор, длина которого нормализуется на единицу делением каждого значения NET на корень квадратный из суммы квадратов значений NET в группе. Все нейроны вне группы имеют нулевые выходы.

Метод интерполяции способен устанавливать более сложные соответствия и может давать более точные результаты. По-прежнему, однако, нет убедительных данных, позволяющих сравнить достоинства и недостатки режимов интерполяции и аккредитации.

14.4.6. Статистические свойства обученной сети

Метод обучения Кохонена обладает полезной и интересной способностью извлекать статистические свойства из множества входных данных. Как показано Кохоненом, для полностью обученной сети вероятность того, что случайно выбранный входной вектор (в соответствии с

функцией плотности вероятности входного множества) будет ближайшим к любому заданному весовому вектору, равна 1/k, где k — число нейронов Кохонена. Это является оптимальным распределением весов на гиперсфере. (Предполагается, что используются все весовые векторы, а это возможно лишь в том случае, если используется один из вышеупомянутых методов распределения весов.)

14.5. Обучение слоя Гроссберга

Слой Гроссберга обучается относительно просто. Входной вектор, являющийся выходом слоя Кохонена, подается на слой нейронов Гроссберга, и выходы слоя Гроссберга вычисляются как при нормальном функционировании. Далее, каждый вес корректируется только в том случае, если он соединен с нейроном Кохонена, имеющим ненулевой выход. Величина коррекции веса пропорциональна разности между весом и требуемым выходом нейрона Гроссберга, с которым этот вес соединен. В символьной записи

$$v_{ijн} = v_{ijc} + \beta(y_j - v_{ijc})k_i,$$

где k_i — выход i -го нейрона Кохонена (только для одного нейрона Кохонена он отличен от нуля); y_j — j -я компонента вектора желаемых выходов.

Первоначально β берется равным приблизительно 0,1 и затем постепенно уменьшается в процессе обучения.

Отсюда видно, что веса слоя Гроссберга будут сходиться к средним величинам от желаемых выходов, тогда как веса слоя Кохонена обучаются на средних значениях входов. Обучение слоя Гроссберга — это обучение с учителем, алгоритм располагает желаемым выходом, по которому он обучается. Обучающийся без учителя, самоорганизующийся слой Кохонена дает выходы в недетерминированных позициях. Они отображаются в желаемые выходы слоем Гроссберга.

14.6. Сеть встречного распространения полностью

На рисунке 14.7 показана сеть встречного распространения целиком. В режиме нормального функционирования предъявляются входные векторы X и Y, и обученная сеть дает на выходе векторы X' и Y', являющиеся аппроксимациями соответственно для X и Y. Векторы X и Y предполагаются здесь нормализованными единичными векторами, следовательно, порождаемые на выходе векторы также будут иметь тенденцию быть нормализованными.

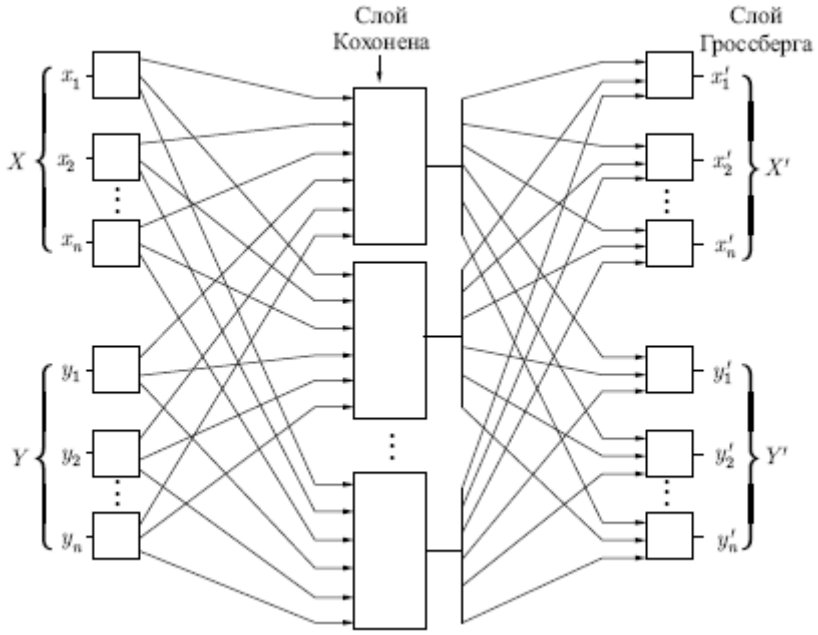


Рисунок 14.7 – Структура сети встречного распространения

В процессе обучения векторы X и Y подаются одновременно и как входные векторы сети, и как желаемые выходные сигналы. Вектор X используется для обучения выходов X', а вектор Y — для обучения выходов Y' слоя Гроссберга. Сеть встречного распространения целиком обучается с использованием того же самого метода, который описывался для сети прямого действия. Нейроны Кохонена принимают входные сигналы как от векторов X, так и от векторов Y. Но эта ситуация неотличима от той, когда имеется один большой вектор, составленный из векторов X и Y, и тем самым не влияет на алгоритм обучения.

В качестве результирующего получается единичное отображение, при котором предъявление пары входных векторов порождает их копии на выходе. Этот вывод не представляется особенно интересным, если не заметить, что предъявление только вектора X (с вектором Y, равным нулю) порождает как выходы X', так и выходы Y'. Если F — функция, отображающая X в Y', то сеть аппроксимирует ее. Также, если F обратима, то предъявление только вектора Y (приравнивая X нулю) порождает X'. Уникальная способность сети встречного распространения — порождать функцию и обратную к ней — делает эту сеть полезной в ряде приложений.

Рисунок 14.7, в отличие от первоначальной конфигурации, не демонстрирует противоток в сети, по которому она получила свое название. Такая форма выбрана потому, что она также иллюстрирует сеть без обратных связей и позволяет обобщить понятия, развитые в предыдущих лекциях.

Приложение: сжатие данных.

В дополнение к обычным функциям отображения векторов, **встречное распространение** оказывается полезным и в некоторых менее очевидных

прикладных областях. Одним из наиболее интересных примеров является сжатие данных.

Сеть встречного распространения может быть использована для сжатия данных перед их передачей, уменьшая тем самым число битов, которые должны быть переданы. Допустим, что требуется передать некоторое изображение. Оно может быть разбито на подизображения S, как показано на рисунке 14.8. Каждое подизображение разбито на пиксели (мельчайшие элементы изображения). Тогда каждое подизображение является вектором, элементами которого являются пиксели, из которых состоит подизображение. Допустим для простоты, что каждый пиксель - это единица (свет) или нуль (чернота). Если в подизображении имеется **n** пикселей, то для его передачи потребуется n бит. Если допустимы некоторые искажения, то для передачи типичного изображения требуется существенно меньшее число битов, что позволяет передавать изображение быстрее. Это возможно из-за статистического распределения векторов подизображений. Некоторые из них встречаются часто, тогда как другие встречаются так редко, что могут быть грубо аппроксимированы. Метод, называемый **векторным квантованием,** находит более короткие последовательности битов, наилучшим образом представляющие эти подизображения.

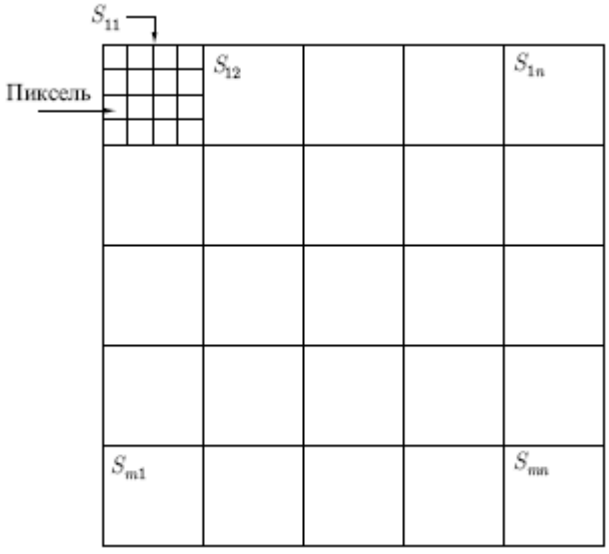


Рисунок 14.8 – Разбивка изображения на подизображения

Сеть встречного распространения может быть использована для выполнения векторного квантования. Множество векторов подизображений используется в качестве входа для обучения слоя Кохонена по методу аккредитации, когда выход единственного нейрона равен 1. Веса слоя Гроссберга обучаются выдавать бинарный код номера того нейрона Кохонена, выход которого равен 1. Например, если выходной сигнал нейрона 7 равен 1 (а все остальные равны 0), то слой Гроссберга будет обучаться выдавать 00...000111 (двоичный код числа 7). Это и будет являться более короткой битовой последовательностью передаваемых символов.

На приемном конце идентичным образом обученная сеть встречного распространения принимает двоичный код и реализует обратную функцию, аппроксимирующую первоначальное подизображение.

Этот метод применялся на практике как к речи, так и к изображениям, с коэффициентом сжатия данных от 10:1 до 100:1. Качество было приемлемым, хотя некоторые искажения данных на приемном конце признаются неизбежными.

Вопросы для повторения и закрепления материала

1. Что такое сеть встречного распространения?
2. Опишите структуру сети встречного распространения?
3. Как работает слой Конохена?
4. Как работает слой Гроссберга?
5. Опишите процесс нормального функционирования слоя Конохена?
6. Опишите процесс нормального функционирования слоя Гроссберга?
7. Опишите процесс обучения слоя Конохена?
8. Опишите процесс обучения слоя Гроссберга?

Задания для самостоятельной работы

1. Проведите обзор предметных областей, в которых использовались сети встречного распространения?

Тема 15. Нейронные сети Хопфилда и Хэмминга

Цель: рассмотреть нейронные сети Хопфилда и Хэмминга.

Задачи:

1. Ознакомиться с конфигурациями сетей с обратным распространением
2. Освоить бинарные системы.
3. Изучить принцип устойчивости нейронной сети.
4. Рассмотреть понятие ассоциативной памяти и задачу распознавания образов.

15.1. Конфигурации сетей с обратными связями

Рассмотренный нами ранее персептрон относится к классу сетей с направленным потоком распространения информации и не содержит обратных связей. На этапе функционирования каждый нейрон выполняет свою функцию — передачу возбуждения другим нейронам — ровно один раз. Динамика состояний нейронов является не итерационной.

Несколько более сложной является динамика в сети Кохонена. Конкурентное соревнование нейронов достигается путем итераций, в процессе которых информация многократно передается между нейронами.

В общем случае может быть рассмотрена нейронная сеть, содержащая произвольные обратные связи, по которым переданное возбуждение возвращается к данному нейрону, и он повторно выполняет свою функцию. Наблюдения за биологическими локальными нейросетями указывают на наличие множественных обратных связей. Нейродинамика в таких системах становится итерационной. Это свойство существенно расширяет множество типов нейросетевых архитектур, но одновременно приводит к появлению новых проблем.

Не итерационная динамика состояний нейронов является, очевидно, всегда устойчивой. Обратные связи могут приводить к возникновению **неустойчивостей**, подобных тем, которые возникают в усилительных радиотехнических системах при положительной обратной связи. В нейронных сетях неустойчивость проявляется в блуждающей смене состояний нейронов, не приводящей к возникновению стационарных состояний. В общем случае, ответ на вопрос об устойчивости динамики произвольной системы с обратными связями крайне сложен и до настоящего времени является открытым.

Остановимся на важном частном случае нейросетевой архитектуры, для которой свойства устойчивости подробно исследованы. На рисунке 15.1 показана сеть с обратными связями, состоящая из двух слоев. Способ представления несколько отличается от использованного в работе Хопфилда и других сходных, но эквивалентен им с функциональной точки зрения. Нулевой слой, как и на предыдущих рисунках, не выполняет вычислительной функции, а лишь распределяет выходы сети обратно на входы. Каждый нейрон первого слоя вычисляет взвешенную сумму своих входов, давая сигнал NET, который затем с помощью нелинейной функции F преобразуется в сигнал OUT. Эти операции сходны с нейронами других сетей.

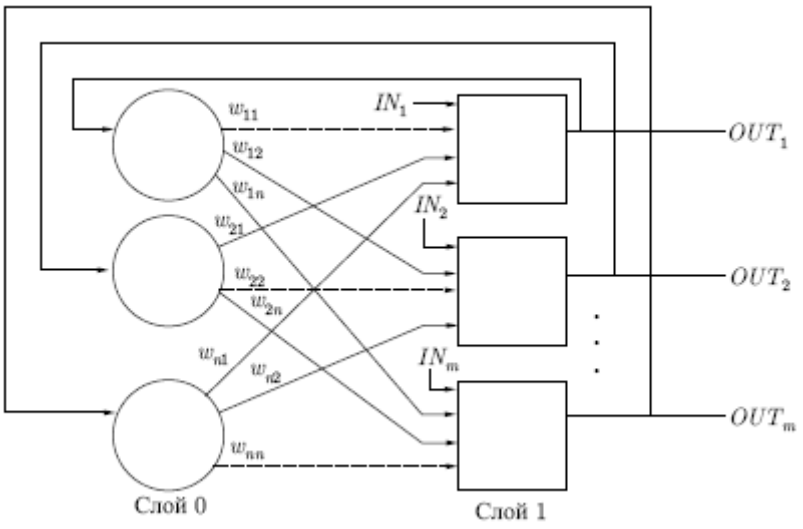


Рисунок 15.1 – Нейронная сеть с обратными связями

15.2. Бинарные системы

В первой работе Д. Хопфилда функция F была просто пороговой функцией. Выход такого нейрона равен единице, если взвешенная сумма выходов с других нейронов больше порога T_j, в противном случае она равна нулю. Порог вычисляется следующим образом:

$$NET_j = \sum_{i \neq j} w_{ij} OUT_i + IN_j,$$

$$OUT_j = \begin{cases} 1, & \text{если } NET_j > T_j, \\ 0, & \text{если } NET_j < T_j, \\ \text{не меняется}, & \text{если } NET_j = T_j. \end{cases}$$

Состояние сети — это просто множество текущих значений сигналов OUT от всех нейронов. В первоначальной сети Хопфилда состояние каждого нейрона менялось в дискретные случайные моменты времени, в последующем - состояния нейронов могли меняться одновременно. Так как выходом бинарного нейрона может быть только ноль или единица (промежуточных уровней нет), то текущее состояние сети является двоичным числом, каждый бит которого является сигналом OUT некоторого нейрона.

Задачи, решаемые данной сетью, как правило, формулируются следующим образом. Известен некоторый набор двоичных сигналов (изображений, оцифровок звука, прочих данных, описывающих некие объекты или характеристики процессов), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий образец (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образцов. В общем случае, любой сигнал может быть описан вектором $X = \{x_i : i = 0 \ldots n-1\}$, n — число нейронов в сети и размерность входных и выходных векторов. Каждый элемент x_i равен либо 1, либо 0. Обозначим вектор, описывающий k-й образец, через X^k, а его компоненты, соответственно, — x_i^k, $k = 0, \ldots, m-1$, m — число компонентов. Когда сеть распознает (или "вспомнит") какой-либо образец на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть $Y = X^k$, где Y - вектор выходных значений сети: $Y = \{y_i : i = 0, \ldots, n-1\}$. В противном случае, выходной вектор не совпадет ни с одним образцовым.

Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха) или же "вольную импровизацию" сети (в случае неудачи).

На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, & \text{если } i \neq j, \\ 0, & \text{если } i = j. \end{cases}$$

Здесь i и j — индексы, соответственно, предсинаптического и постсинаптического нейронов; x_i^k, x_j^k — i-й и j-й элементы вектора k-го образца.

Алгоритм функционирования сети следующий (**p** — номер итерации):

1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов: $y_i(0) = x_i, \quad i = 0, \ldots, n-1$, поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от y_i означает нулевую итерацию в цикле работы сети.
2. Рассчитывается новое состояние нейронов: $s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), \quad j = 0, \ldots, n-1$ и новые значения аксонов $y_j(p+1) = f\lfloor s_j(p+1) \rfloor$.

   где **f** — активационная функция в виде скачка.
3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да — переход к пункту 2, иначе (если выходы стабилизировались) — конец процедуры. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Как говорилось выше, иногда сеть не может провести распознавание и выдает на выходе несуществующий образ. Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых образов m не должно превышать величины, примерно равной 0,15n. Кроме того, если два образа **А** и **Б** имеют значительное сходство, они, возможно, будут вызывать у сети перекрестные ассоциации, то есть предъявление на входы сети вектора **А** приведет к появлению на ее выходах вектора **Б** и наоборот.

Когда нет необходимости, чтобы сеть выдавала образец в явном виде и достаточно, скажем, получать номер образца, ассоциативную память

успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, более экономным использованием памяти и меньшим объемом вычислений, что становится очевидным из ее структуры (рисунок 15.2).

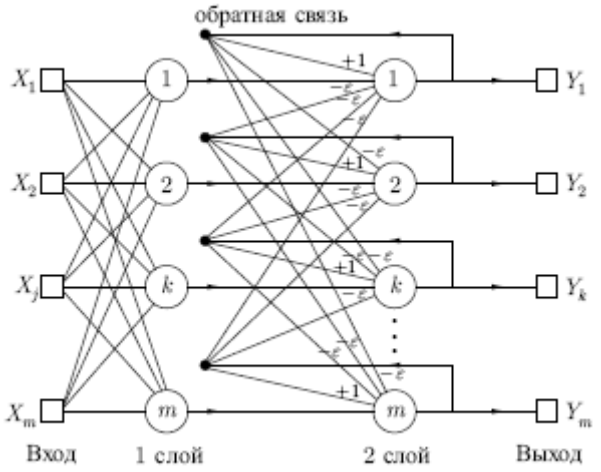


Рисунок 15.2 – Нейронная сеть Хэмминга

Сеть состоит из двух слоев. Первый и второй слои имеют по m нейронов, где m — число образцов. Нейроны первого слоя имеют по **n** синапсов, соединенных с входами сети (которые образуют фиктивный нулевой слой). Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями. Единственный синапс с положительной обратной связью для каждого нейрона соединен с его же аксоном.

Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов. Расстоянием Хэмминга называется число отличающихся битов в двух бинарных векторах. Сеть должна выбрать образец с минимальным расстоянием Хэмминга до неизвестного входного сигнала, в результате чего будет активизирован только один выход сети, соответствующий именно этому образцу.

На стадии инициализации весовым коэффициентам первого слоя и порогу активационной функции присваиваются следующие значения:

$$w_{ik} = \frac{x_i^k}{2}, \quad i = 0, \ldots, n-1, \; k = 0, \ldots, m-1,$$

$$T_k = \frac{n}{2}, \quad k = 0, \ldots, m-1.$$

Здесь x_i^k - i -й элемент k -го образца.

Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине $0 < \varepsilon < 1/m$. Синапс нейрона, связанный с его же аксоном, имеет вес +1.

Алгоритм функционирования сети Хэмминга следующий:

1. На входы сети подается неизвестный вектор

$$X = \{x_i | i = 0, \ldots, n\},$$

исходя из которого рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j, \quad j = 0, \ldots, m-1.$$

После этого полученными значениями инициализируются значения аксонов второго слоя:

$$y_j^{(2)} = y_j^{(1)}, \quad j = 0, \ldots, m-1.$$

2. Вычисляются новые состояния нейронов второго слоя:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), \quad k \neq j, \quad j = 0, \ldots, m-1$$

и значения их аксонов:

$$y_j^{(2)}(p+1) = f\lfloor s_j^{(2)}(p+1) \rfloor, \quad j = 0, \ldots, m-1.$$

Активационная функция f имеет вид порога, причем величина F должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

3. Проверить, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да — перейти к шагу 2. Иначе — конец процедуры.

Из оценки алгоритма видно, что роль первого слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть вообще исключен из сети.

15.3. Устойчивость

Как и в других сетях, веса между слоями в этой сети могут рассматриваться в виде матрицы W. Сеть с обратными связями является устойчивой, если ее матрица симметрична и имеет нули на главной диагонали, т. е. если $w_{ij} = w_{ji}$ и $w_{ii} = 0$ для всех i.

Устойчивость такой сети может быть доказана с помощью математического метода. Допустим, что найдена функция, которая всегда убывает при изменении состояния сети. В конце концов, эта функция должна достичь минимума и прекратить изменение, гарантируя тем самым устойчивость сети. Такая функция, называемая функцией Ляпунова, для рассматриваемых сетей с обратными связями может быть введена следующим образом:

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} OUT_i OUT_j - \sum_j I_j OUT_j + \sum_j T_j OUT_j,$$

где E — искусственная энергия сети; w_{ij} — вес от выхода нейрона i к входу нейрона j; OUT_i — выход нейрона j; I_j — внешний вход нейрона j; T_j — порог нейрона j.

Изменение энергии E, вызванное изменением состояния j -нейрона, есть

$$\delta E = \left[\sum_{i \neq j} (w_{ij} OUT_i) + I_j - T_j\right] \delta OUT_j = -[NET_j - T_j]\delta OUT_j,$$

где δOUT_j — изменение выхода j-го нейрона.

Допустим, что величина NET нейрона j больше порога. Тогда выражение в скобках будет положительным, а из данных уравнений следует, что выход нейрона j должен измениться в положительную сторону (или остаться без изменения). Это значит, что δOUT_j может быть только положительным или нулем и δE должно быть отрицательным. Следовательно, энергия сети должна либо уменьшиться, либо остаться без изменения.

Далее, допустим, что величина NET меньше порога. Тогда величина δOUT_i может быть только отрицательной или нулем. Следовательно, опять энергия должна уменьшиться или остаться без изменения.

И окончательно, если величина NET равна порогу, δ_i равна нулю и энергия остается без изменения.

Мы показали, что любое изменение состояния нейрона либо уменьшит энергию, либо оставит ее без изменения. Благодаря такому непрерывному стремлению к уменьшению энергия, в конце концов, должна достигнуть минимума и прекратить изменение. По определению такая сеть является устойчивой.

Симметрия сети является достаточным, но не необходимым условием для устойчивости системы. Имеется много устойчивых систем (например, все сети прямого действия), которые ему не удовлетворяют. Можно продемонстрировать примеры, в которых незначительное отклонение от симметрии будет приводить к непрерывным осцилляциям. Однако приближенной симметрии обычно достаточно для устойчивости систем.

15.4. Ассоциативность памяти и задача распознавания образов

Динамический процесс последовательной смены состояний нейронной сети Хопфилда завершается в некотором стационарном состоянии, являющимся локальным минимумом энергетической функции E(S). Невозрастание энергии в процессе динамики приводит к выбору такого локального минимума S, в бассейн притяжения которого попадает начальное состояние (исходный, предъявляемый сети образ) S_0. В этом случае также говорят, что состояние S_0 находится в чаше минимума S.

При последовательной динамике в качестве стационарного состояния будет выбран такой образ S, который потребует минимального числа изменений состояний отдельных нейронов. Поскольку для двух двоичных векторов минимальное число изменений компонент, переводящее один вектор в другой, является расстоянием Хемминга $\rho_H(S, S_0)$, то можно заключить, что динамика сети заканчивается в ближайшем по Хеммингу локальном минимуме энергии.

Пусть состояние S соответствует некоторому идеальному образу памяти. Тогда эволюцию от состояния S_0 к состоянию S можно сравнить с процедурой постепенного восстановления идеального образа S по его искаженной (зашумленной или неполной) копии S_0. Память с такими свойствами процесса считывания информации является **ассоциативной.** При поиске искаженные части целого восстанавливаются по имеющимся неискаженным частям на основе ассоциативных связей между ними.

Ассоциативный характер памяти сети Хопфилда качественно отличает ее от обычной, адресной, компьютерной памяти. В последней извлечение необходимой информации происходит по **адресу** ее начальной точки (ячейки памяти). Потеря адреса (или даже одного бита адреса) приводит к потере доступа ко всему информационному фрагменту. При использовании же ассоциативной памяти доступ к информации производится непосредственно по ее **содержанию**, т.е. по частично известным искаженным фрагментам. Потеря части информации или ее зашумление не приводит к катастрофическому ограничению доступа, если оставшейся информации достаточно для извлечения идеального образа.

Поиск идеального образа по имеющейся неполной или зашумленной его версии называется задачей **распознавания образов**. Рассмотрим особенности решения этой задачи нейронной сетью Хопфилда будут продемонстрированы на примерах, которые получены с использованием модели сети на персональной ЭВМ.

В рассматриваемой модели сеть содержала 100 нейронов, упорядоченных в матрицу 10x10. Сеть обучалась по правилу Хебба на трех идеальных образах — шрифтовых начертаниях латинских букв M, A и G (рисунок 15.3). После обучения нейросети в качестве начальных состояний

нейронов предъявлялись различные искаженные версии образов, которые в дальнейшем эволюционировали с последовательной динамикой к стационарным состояниям.

Рисунок 15.3 – Образцы букв M, A, G

Для каждой пары изображений на рисунке 15.4, левый образ является начальным состоянием, а правый — результатом работы сети, достигнутым стационарным состоянием.

Образ на рисунке 15.4(А) был выбран для тестирования адекватности поведения на идеальной задаче, когда предъявленное изображение точно соответствует информации в памяти. В этом случае за один шаг было достигнуто стационарное состояние. Образ на рисунке 15.4(Б) характерен для задач распознавания текста независимо от типа шрифта. Начальное и конечное изображения безусловно похожи, но попробуйте это объяснить машине!

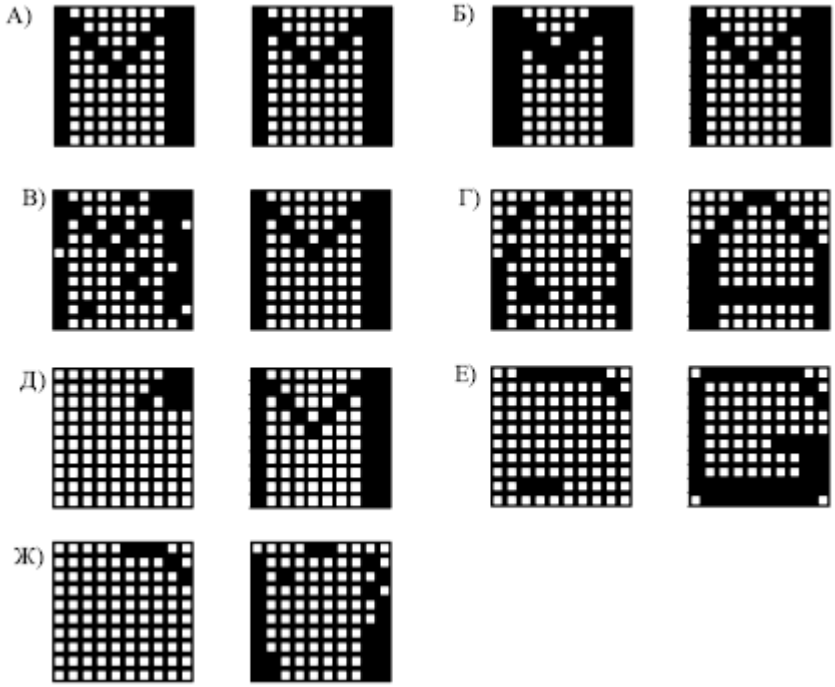


Рисунок 15.4 – Пары изображений букв M, A, G

Задания на рисунках 15.4(В, Г) характерны для практических приложений. Нейросетевая система способна распознавать практически полностью зашумленные образы. Задачи, соответствующие рисункам 15.4(Д, Е), демонстрируют замечательное свойство сети Хопфилда: она способна ассоциативно узнавать образ по его небольшому фрагменту. Важнейшей особенностью работы сети является генерация ложных образов. Пример

ассоциации к ложному образу показан на рисунке 15.4(Ж). Ложный образ является устойчивым локальным экстремумом энергии, но не соответствует никакому идеальному образу. Он является в некотором смысле собирательным образом, наследующим черты идеальных собратьев. Ситуация с ложным образом эквивалентна нашему "Где-то я уже это видел".

В данной простейшей задаче ложный образ является "неверным" решением и поэтому вреден. Однако можно надеяться, что такая склонность сети к обобщениям может быть как-то использована. Характерно, что при увеличении объема полезной информации (сравните рисунки 15.4 (Е) и (Ж)) исходное состояние попадает в область притяжения требуемого стационарного состояния, и образ распознается.

Вопросы для повторения и закрепления материала

1. Как выглядит нейронная сеть с обратными связями?
2. Что такое бинарные системы?
3. Что такое устойчивость сети?
4. Ассоциативная память это?
5. Каким образом реализуется ассоциативная память с использованием нейронных сетей?
6. Могут ли использоваться нейронные сети в задачах распознавания образов?

Задания для самостоятельной работы

1. Проведите обзор строения нейронных сетей Хопфилда и Хэмминга.

Лабораторная работа №1. Основы программирования в системе MatLab

Цель работы: освоить принципы работы интегрированной среды MatLab.

Теоретическая часть

Командное окно системы MatLAB

После запуска системы MatLAB на экране появиться окно, показанное на рисунке 1.1. Представленное окно содержит три области: Command Window, предназначенное для ввода команд и вывода результатов; Command History, содержащую историю всех выполненные в Command Window команд и Workspace, где отображаются все созданные в рабочей области переменные.

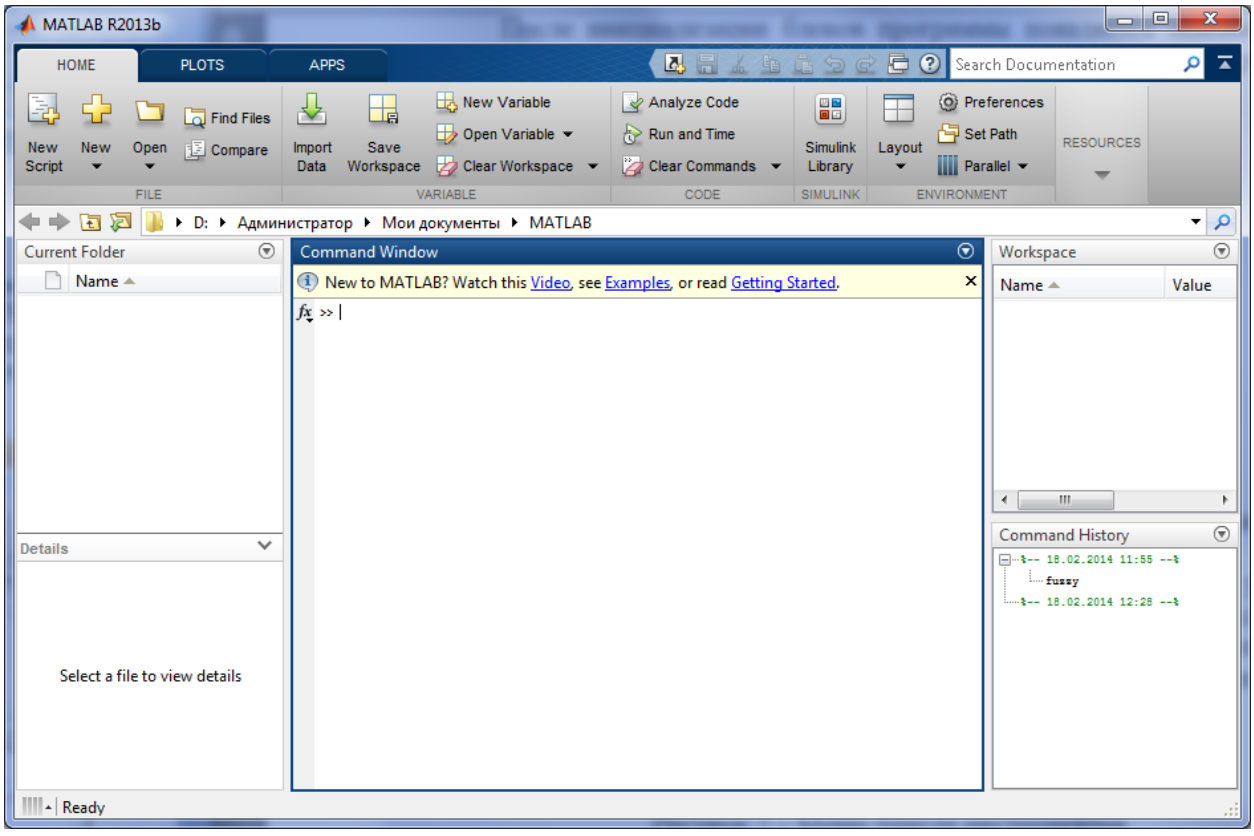


Рисунок - 1.1. Командное окно системы MatLAB

Знак «>>» показывает готовность системы к выполнению введенных команд. Набрав простейшие математические выражения в естественной форме записи, сразу же вычисляется результат. Это выражение может быть записано в двух видах: <*Выражение*> или <*Имя переменной*> = <*Выражение*>. Во втором случае результат не только вычисляется, но и присваивается указанной переменой. MatLAB не требует от пользователя

специальных команд для объявления переменных, они создаются автоматически при первом указании пользователем их имени. В первом случае на самом деле результат выражения присваивается специальной служебной переменной имеющей имя «ans», так же можно использовать эту переменную в расчетах. Если нет необходимости, что бы MatLAB выводил результаты промежуточных выражений на экран, то необходимо поставить в конце выражения символ «;».

Типы данных

Система MatLAB работает со следующими базовыми типами данных:

- Число – вещественное числовое значение.
- Массив – это данные (объекты) одной природы, сгруппированные по одному и тому же характерному признаку.
- Матрица – массив, представленный в виде прямоугольной таблицы, каждый элемент имеет номер (индекс), определяющий однозначно его положение в матрице, в индексировании идет сначала номер строки, а потом номер столбца, где расположен элемент.
- Многомерный массив – пространственная матрица, имеющая три и более размерностей, каждый элемент также имеет индекс, однозначно определяющий его положение. Грубо говоря, многомерный массив – это матрица матриц.
- Вектор – одномерная матрица. Без особых указаний со стороны пользователя это матрица-столбец.
- Структура – это набор разнотипных полей. Поле может содержать как массив так и число так и строку. Одно поле содержит данные только одного типа.
- Строка – набор (массив) символов символьных таблиц компьютера.

Фактически MatLAB содержит один тип данных – массив или матрица. Массив это группа ячеек памяти, имеющие одно имя. Массивы бывают одномерные – строка или столбец, прямоугольные, квадратные (число строк рано числу столбцов). Когда Вы указываете переменную и присваиваете ей одно число, фактически MatLAB создает матрицу из одной строки и одного столбца (размерность массива отображается в окне Workspace в поле Size). Любая переменная в MatLAB – это матрица.

На рисунке 1.2 приведены примеры столбца – а, строки – б, прямоугольной матрицы – в, квадратной матрицы – г, матрицы единичной размерности – е (простой переменной).

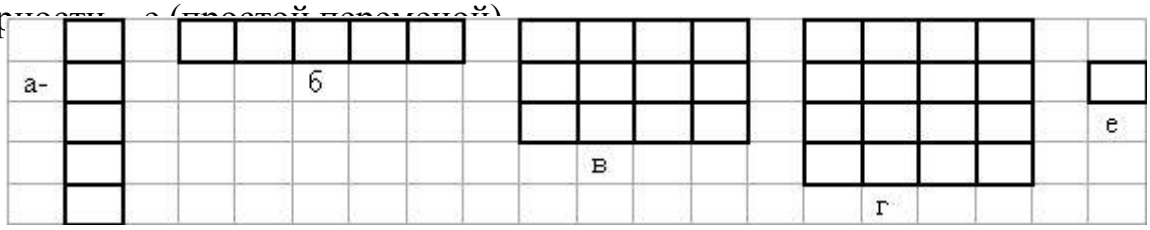


Рисунок - 1.2. Виды матриц

Помимо одномерных и двумерных матриц MatLAB поддерживает ряд других типов данных. К ним относятся многомерные массивы, строки, структуры, массивы ячеек, а также объекты.

Числа, переменные и функции

Числа в MATLAB могут быть положительными и отрицательными, целыми и дробными, действительными и комплексными. Они могут представляться с фиксированной и плавающей точкой, с мантиссой и порядком. Особенности представления чисел в MATLAB:

• мнимая единица кодируется с помощью двух символов: i или j;

• целая часть числа от дробной отделяется точкой;

• отделение порядка числа от мантиссы осуществляется символом e.

Форматы чисел:

• format chort – короткое представление (5 знаков числа);

• format chort e – короткое представление в экспоненциальной форме (5 знаков мантиссы, 3 знака порядка);

• format lonq – длинное представление числа (15 знаков);

• format lonq e – длинное представление в экспоненциальной форме (15 знаков мантиссы, 3 знака порядка).

Переменные – это символы, используемые для обозначения некоторых хранимых данных. Переменная имеет имя, называемое *идентификатором*. Имя переменной начинается с буквы и может состоять из букв и цифр и некоторых (допустимых) символов.

Константы – это численное значение уникального имени, имеющего математический смысл. Наиболее часто в MATLAB используются следующие константы:

• pi – число π;

• inf – машинная бесконечность;

• ans – имя переменной, хранящей результат вычисления;

• NaN – нечисловой характер данных, возникает, например, как результат операции 1/0.

Элементарные функции:

• abs(x) – абсолютное значение *x*;

• exp(x) – экспоненциальная функция *e^x*;

• log(x), log10(x), log2(x) – логарифмы чисел с основанием *e*, 10, 2;

• sqrt(x) – корень квадратный из *x*;

• sin(x), cos(x), tan(x), cot(x), sec(x), csc(x) – тригонометрические функции sin *x*, cos *x*, tg *x*, ctg *x*, sec *x*, cosec *x*

• asin(x), acos(x), atan(x), acot(x), asec(x), acsc(x) – обратные тригонометрические функции arcsin *x*, arcos *x*, arctg *x*, arcctg *x*, arcsec *x*, arccosec *x*;

• sinh(x), cosh(x), tanh(x), coth(x), sech(x), csch(x) – гиперболические функции sh *x*, ch *x*, th *x*, cth *x*, sch *x*, csch *x*;

• asinh(x), acosh(x), atanh(x), acoth(x), asech(x), acsch(x) – обратные гиперболические функции arsh *x*, arch *x*, arth *x*, arcth *x*, arsch *x*, arcsch *x*.

Для формирования логических условий полезны также следующие функции:

• all(x) — возвращает 1, если все элементы x отличны от нуля;

• any(x) — возвращает 1, если хотя бы один элемент x отличен от нуля;

• isequal(x, y) — возвращает 1, если значения x и y совпадают. В отличие от конструкции all(x==y), использование данной функции не приведет к ошибке, если размеры x и y не совпадают;

• isempty(x) — возвращает 1, если x является пустой матрицей (то есть имеет размер 0x0).

Со списком элементарных функций можно ознакомиться, набрав команду:

```
>> help elfun
```

Со списком операторов можно ознакомиться, набрав команду:

```
>> help ops
```

Графика системы MatLab

В режиме прямых вычислений доступны почти все возможности системы, в том числе построение графиков. При этом графики строятся в отдельных масштабируемых и перемещаемых окнах.

Двумерная графика. Для построения графика необходимо выполнение следующих действий:

• задать интервал изменения аргумента;

• использовать команду построения графиков plot.

Пример:

```
>> x=0:0.1:10;
>> plot(sin(x))
```

Построение в одном окне графиков нескольких функций. Функции могут быть обозначены переменными, не имеющими явного указания аргумента: y1=sin(x); y2=cos(x); y3=x*sin(x). Это обусловлено тем, что переменные y1, y2, y3 являются векторами, как и переменная x. Используем одну из форм команды plot(a1,f1,a2,f2,a3,f3,…), где a1, a2, a3 –векторы аргументов функций (в нашем случае они все - x), а f1, f2, f3 – векторы значений функций, графики которых строятся в одном окне.

Для построения в одном окне графиков нескольких функций необходимо:

• задать эти функции;

• использовать одну из команд plot, например plot(a1,f1,a2,f2,a3,f3,…);

Пример (рисунок 1.3):

```
>> x=-5:0.5:5;
>> y1=sin(x); y2=cos(x); y3=sin(x)./x;
```

```
>> plot(x,y1,x,y2,x,y3)
ИЛИ
>> x=-5:0.5:5;
>> y1=sin(x);
>> y2=cos(x);
>> hold on
>> plot(x,y1);
>> plot(x,y2)
```

Обратите внимание на задание третьей функции. Если x представляет собой массив (вектор), то нельзя использовать оператор матричного деления (/) (или умножения (*)). При вычислении функции, в нашем случае - sin(x)/x, надо использовать оператор почленного деления (./). При таком задании функции построение графика возможно, но возникает предупреждение о делении на ноль, в момент, когда x=0:

Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warning.)

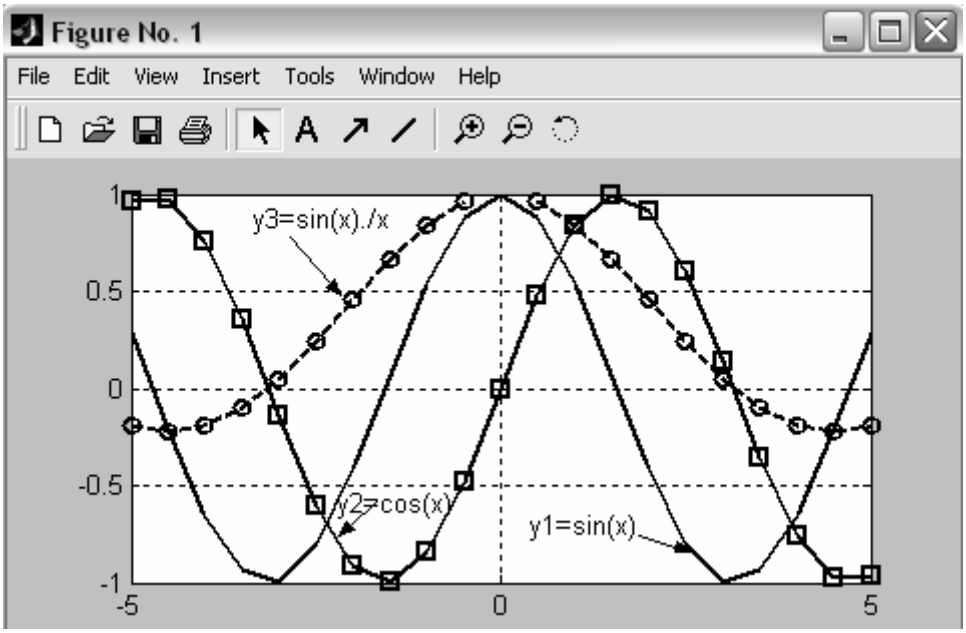


Рисунок 1.3 – График 3-х функций y1=sin(x); y2=cos(x); y3=sin(x)./x

Команда hold on включает режим сохранения текущего графика и свойств объекта axes, так что последующие команды приведут к добавлению новых графиков в графическом окне.

Команда fplot. Команда fplot('f(x)',[xmin xmax]) позволяет строить графики функций, которые имеют устранимые неопределенности. Команда fplot позволяет строить графики функции f(x), заданной в символьном виде, в интервале изменения аргумента x от xmin до xmax без фиксированного шага изменения x.

Пример (рисунок 1.4):

```
>> clear; fplot('sin(x)/x',[-15,15]); grid on
```

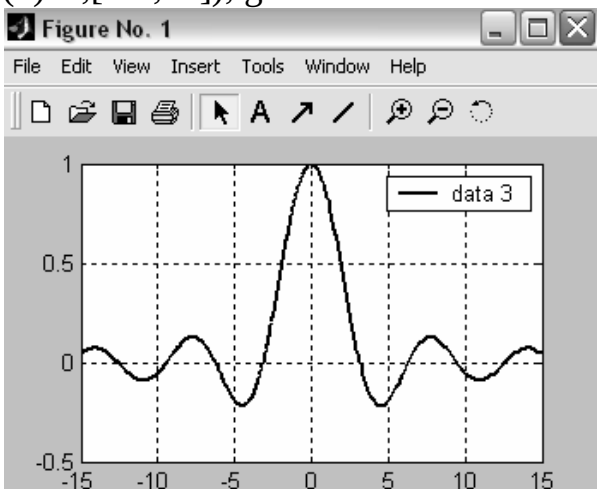


Рисунок 1.4 – График выполнения команды fplot('sin(x)/x',[-15,15]).

Здесь команда clear очищает графическое окно, а команда grid on включает отображение сетки.

Столбцовые диаграммы (команда bar(V)). В прикладных расчетах часто встречаются графики, именуемые столбцовыми диаграммами, отражающие содержание некоторого вектора V. При этом каждый элемент вектора представляется столбцом, высота которого пропорциональна значению элемента. Столбцы нумеруются и масштабируются по отношению к максимальному значению наиболее высокого столбца. Особенно часто столбцовые диаграммы используются при представлении данных финансово-экономических расчетов.

Пример (рисунок 1.5):

```
>> V=[1,2,3,4,5,4,3,2,1];
>> bar(V)
```

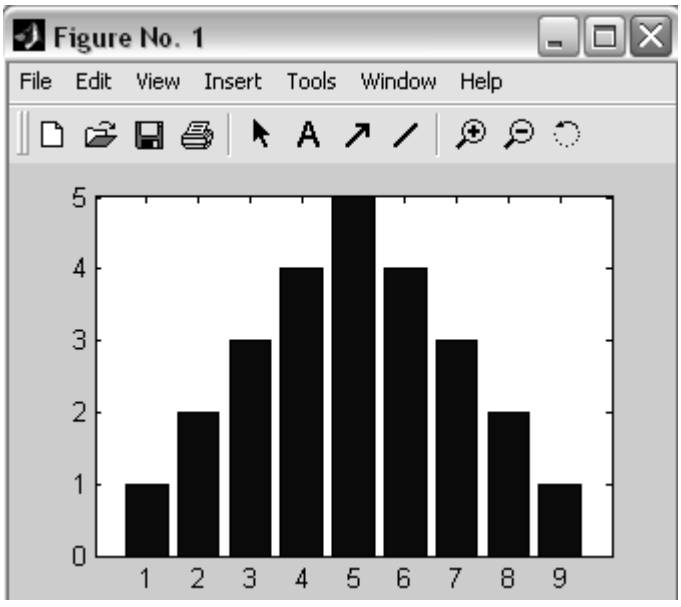


Рисунок 1.5 – Результат выполнения команды bar(V)

График в виде ступенчатой линии (команда stairs(x,f(x))) (рисунок 1.6).

```
>> x=-5:0.1:5;
>> stairs(x,x.^2)
```

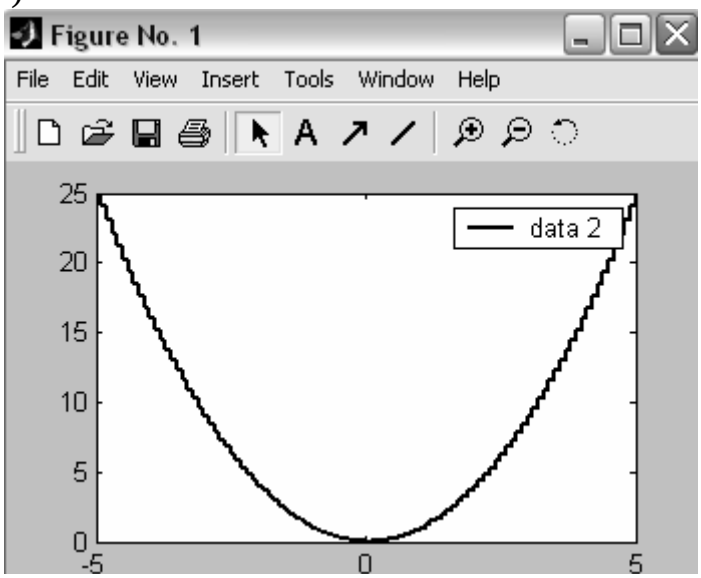


Рисунок 1.6 – Результат выполнения команды stairs(x,x.^2)

График в виде «стебельков» (команда stem(x,y)) (рисунок 1.7).

```
>> x=-5:0.1:5;
>> y=x.^3;
>> stem(x,y)
```

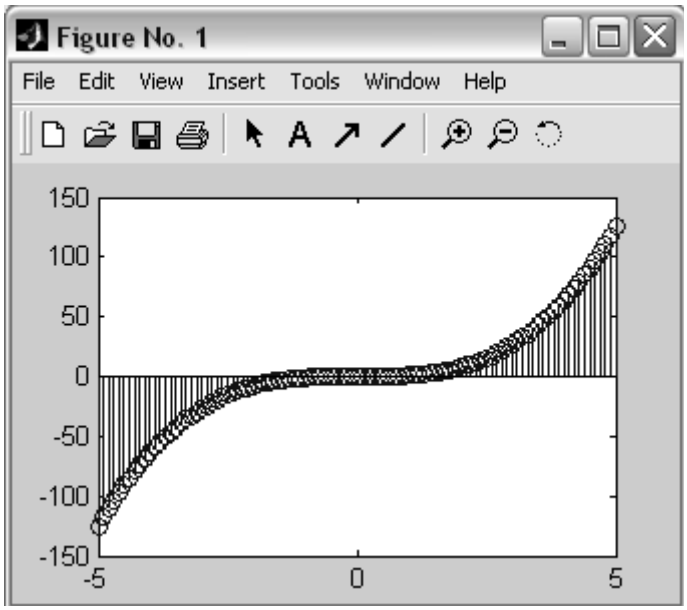


Рисунок 1.7 – Результат выполнения команды stem(x,y)

Трехмерная графика. В MATLAB имеется функция plot3(x,y,z), которая является трехмерным аналогом функции plot, т.е. она позволяет строить линию в трехмерном пространстве по координатам точек. Для построения

графика функции двух переменных в виде поверхностей используют функции:

• mesh(x,y,z) – построение поверхности в виде сетки с закрашенными ребрами и незакрашенными четырехугольными ячейками;

• meshc(x,y,z) – комбинация функций mesh и contour. Линии уровня выводятся на нижней координатной плоскости;

• meshz(x,y,z) – то же что и mesh, но с краев построенной сетчатой поверхности вниз спадает «занавес»;

surf(x,y,z) – построение сетчатой поверхности. Координаты углов каждой ячейки задаются значениями четырех соседних элементов массивов x,y,z с индексами (i,j), (i, j+1), (i+1, j), (i+1, j+1). Значение массива z рассчитываются по формуле функциональной зависимости с использованием поэлементных операций над массивами x,y;

• surfc(x,y,z) - комбинация функций surf и contour. Линии уровня выводятся на нижней координатной плоскости;

• waterfall(x,y,z) – то же что и mesh, но ребра, разделяющие ячейки, проводятся только вдоль оси OX. В результате объемное тело выглядит «нарезанным на ломтики»;

• stem3(x,y,z) – вывод трехмерного графика в виде «стебельков», начинающихся при z=0 в точках, задаваемых массивами x,y. Высота «стебельков» определяется массивом z;

• contourf(x,y,z) – то же что contour, но пространство между линиями равного уровня окрашено в разные цвета в зависимости от значений z;

• contour3(x,y,z) - то же что contour, но линии равного уровня рисуются не в одной плоскости, а в зависимости от значений z;

• pcolor(x,y,z) – строит двумерный график, представляющий собой сетку. Тот же результат можно получить, если для поверхности, построенной с помощью функции surf, установить точку обзора точно сверху

Команда surf:

• сначала надо задать (сетку) диапазон изменения параметров x и y. Удобнее всего массивы формировать с помощью специальной функции [x,y]=meshgrid(x,y). Формируемые массивы имеют length(y) строк и length(x) столбцов;

• затем задать функцию, зависящую от двух переменных;

• применить функцию surf.

```
x=-5:0.2:5;
[x,y]=meshgrid(x);
z=sinc(sqrt(x.^2+y.^2));
surf(x,y,z)
colormap gray (вывод графика серого
цвета)
```

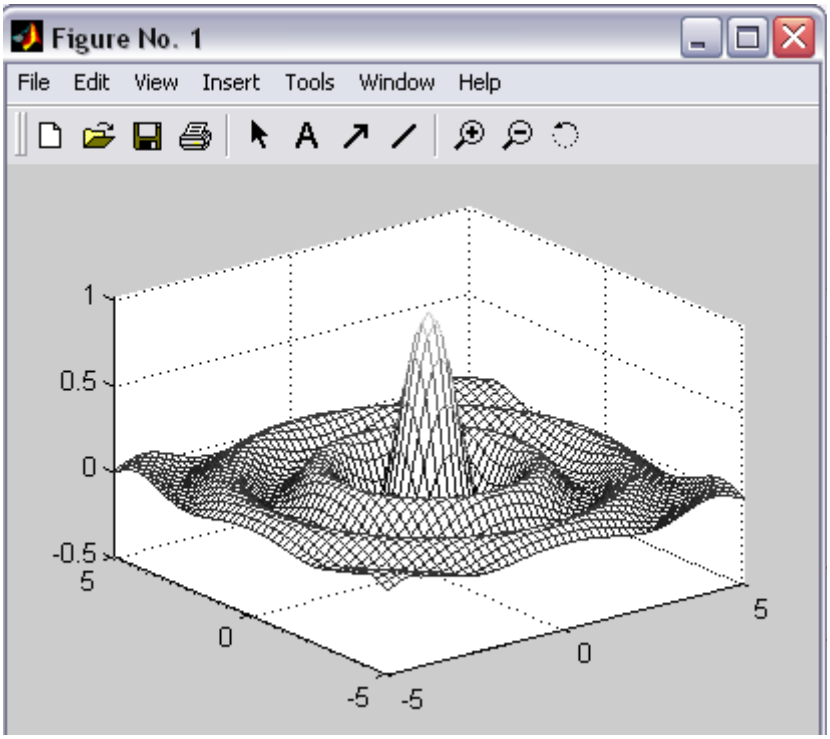


Рисунок 1.8 – Результат выполнения команды surf(x,y,z)

Команда mesh(x,y,z) (рисунок 1.9).

```
>>[x,y]=meshgrid(-5:0.1:5);
>> z=x.*sin(x+y);
>> mesh(x,y,z)
```

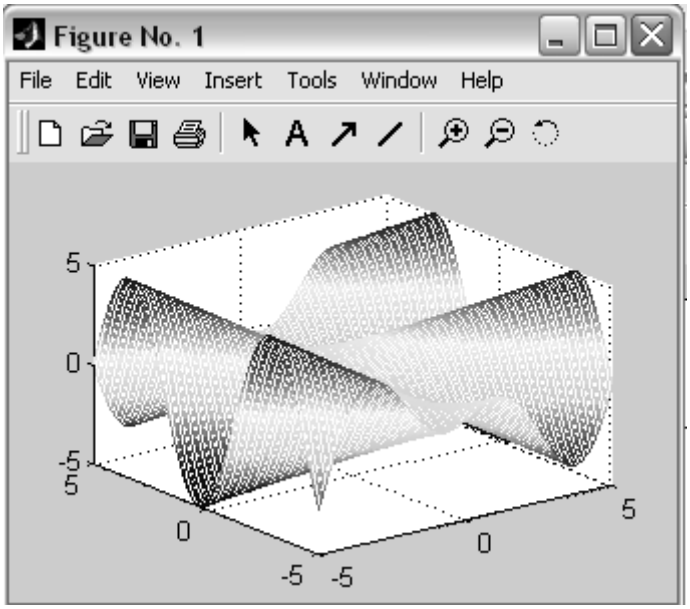


Рисунок 1.9 – Результат выполнения команды mesh(x,y,z)

Графики разного типа в одном окне. При необходимости можно расположить несколько координатных осей с различными графиками без наложения их друг на друга. Для этого используются команды:

• subplot – создает новые объекты класса axes (подокна);

• subplot(m,n,p) - или subplot(mnp) – разбивает графическое окно на m*n подокон, где m – число подокон по горизонтали, n – по вертикали, а p – номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам);

• subplot(H), где H – дескриптор для объекта axes, дает альтернативный способ задания подокна для текущего графика;

• subplot('position', [left bottom width height]) – создает подокно с заданными нормализованными координатами (в пределах от 0.0 до 1.0);

• subplot(111) и clf reset – удаляют все подокна и возвращают графическое окно в обычное состояние.

Пример. Построим четыре графика различного типа в одном окне с помощью функции subplot (рисунок 1.10):

```
>> x=-5:0.1:5;
>>[x,y]=meshgrid(x);
>> z=x.^2+y.^2;
>> subplot(2,2,1),plot(x,sin(x))
>> subplot(2,2,2),plot(sin(5*x),cos(2*x+0.2))
>> subplot(2,2,3), plot3(x,y,z)
>> subplot(2,2,4),surf(peaks) {где peaks – встроенный объект}
```

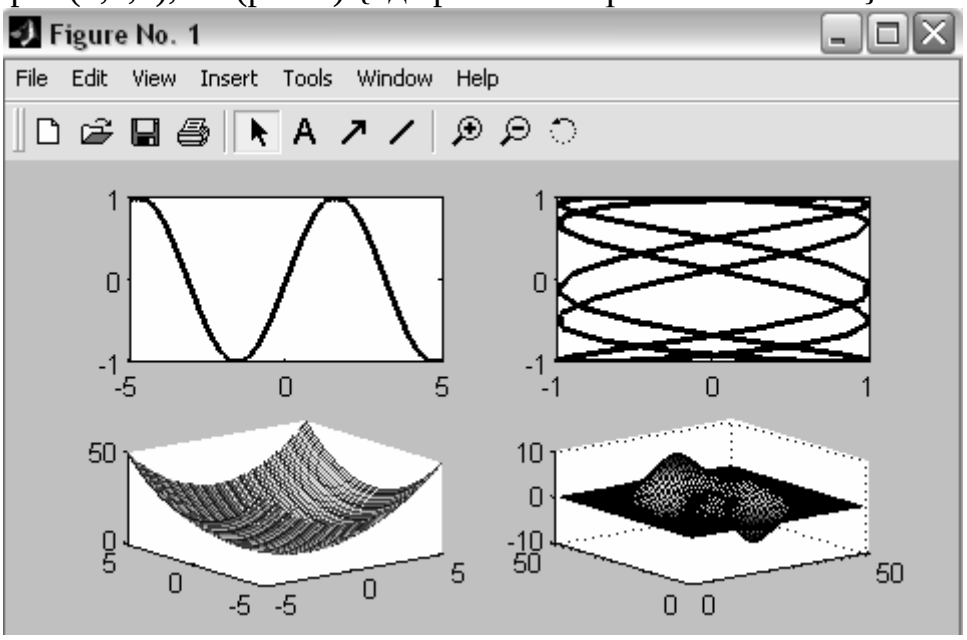


Рисунок 1.10 – Четыре графика в одном окне

Программирование в системе MatLab

Программа в среде MatLAB может быть введена двумя способами: непосредственно в Command Window, либо заранее в редакторе. Программирование в системе MatLAB очень близко к обычному программированию. Программа создается в любом текстовом редакторе. Файл должен иметь расширение *.m. Текст М-файла функции должен начинаться с заголовка *function*, имеющего следующий вид:

function [y1, y2, ...] = fname(x1, x2, ...)

Здесь *fname* – имя функции, *x1*, *x2* и т.д. – входные параметры, *y1*, *y2* и т.д. – выходные параметры. Входные и/или выходные параметры могут отсутствовать. На самом деле имя функции определяется не строкой *fname*, а именем, под которым сохранен М-файл, оно должно совпадать с именем функции.

В качестве примера создадим функцию myfunc, которая будет строить трехмерных график и принимать три входных параметра: точку начала построения графика, точку окончания построения, шаг. Для этого, открыв окно редактора командой меню *File -> New -> M-file*, вводим текст:

```
function myfunc(x1, x2, step)
[X,Y]=meshgrid([x1:step:x2]);
Z=X.*exp(-X.^2-Y.^2);
mesh(X, Y, Z);
```

Введя текст, необходимо сохранить файл под именем *myfunc*.

Другим способом создания нового М-файла служит команда *edit fname.m*. С помощью нее можно как редактировать уже имеющиеся М-файлы, так и создавать новые.

Для того чтобы функция была доступна из системы MatLab, система должна быть способна найти соответствующий М-файл. Поиск файлов осуществляется следующим образом: сначала просматривается текущий рабочий каталог (его имя показано в панели инструментов главного окна), а затем каталоги входящие в путь поиска (MatLab search path). Для вызова М файла необходимо набрать его имя в командной строке MatLAB, и если необходимо его аргументы. Важным элементом облегчающим программирование являются комментарии. Строка комментария начинается в MatLAB символом ‘%’.

Ввод с клавиатуры

```
x=input(‘строка подсказки’)
x=input(‘строка подсказки’, ‘s’)
```

Функция *input* выводит на экран строку подсказки и ждет ввода переменной. Функция *x=input(‘строка подсказки’, ‘s’)* возвращает введенную пользователем строку. При вводе переменных допустимо пользоваться стандартными функциями.

Визуализация вычислений

Система MATLAB имеет богатые возможности графического представления информации. Она позволяет строить двумерные и трехмерные графики функций, заданных в аналитическом виде, в виде векторов и матриц, дает возможность построения множества функций на одном графике: позволяет представлять графики разными цветами, типами точек и линий и в различных системах координат.

Основными функциями двухмерной графики являются:

plot(x, y)

plot(x, y, s)

plot(x1, y1, s1, x2, y2, s2, …, xn, yn, sn)

где:

♦ x – аргумент функции, задаваемой в виде вектора;

♦ y – функция, представленная в аналитическом виде или в виде вектора или матрицы;

♦ s – вектор стилей графика; константа, определяющая цвет линий графика, тип точек и тип линий;

♦ x1, x2, …, xn – аргументы *n* функций, изображаемых на одном графике;

♦ y1, y2, …, yn – функции, изображаемые на одном графике.

В таблице 1.1 приведены стили графиков системы MATLAB.

Таблица 1.1 - Стили графиков

Тип точки		**Цвет линии**		**Тип линии**	
,	Точка	Y	Желтый	-	Сплошная
O	Окружность	M	Фиолетовый	:	Двойной пунктир
X	Крест	C	Голубой	-.	Штрих-пунктир
+	Плюс	R	Красный	--	Штриховая
*	Восьмиконечная снежинка	G	Зеленый		
S	Квадрат	B	Синий		
D		W	Белый		
V, ^, <, >	Треугольник вверх, вниз, влево, вправо	K	Черный		
P	Пятиконечная звезда				
H	Шестиконечная звезда				

В дополнение к управлением выводом координатной сетки, можно настроить режим затирания предыдущего графика при выводе нового. По умолчанию каждая следующая команда plot затирает вывод предыдущей. При помощи команды hold on затирание отключается, команда hold off вновь включает режим затирания.

Основы построения циклических структур в системе MatLab

Генерация случайных чисел

Оператор RAND

Синтаксис:

X = rand(n)

X = rand(m, n)

X = rand(size(A))

X = rand

Функция X = rand(n) формирует массив размера n x n, элементами которого являются случайные величины, распределенные по равномерному закону в интервале [0, 1].

Функция X = rand(m, n) формирует массив размера m x n, элементами которого являются случайные величины, распределенные по равномерному закону в интервале [0, 1].

Функция X = rand(size(A)) формирует массив соразмерный с матрицей A, элементами которого являются случайные величины, распределенные по равномерному закону в интервале [0, 1].

Функция rand без аргументов формирует одно случайное число, подчиняющееся равномерному закону распределения в интервале [0, 1], которое изменяется при каждом последующем вызове. На рисунке 1.11 представлен пример работы функции rand.

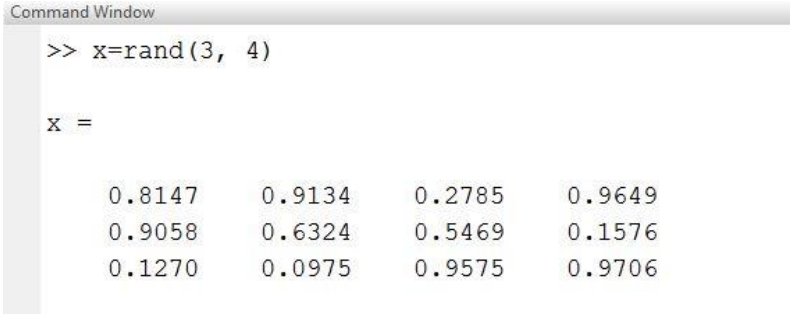


Рисунок 1.11 – Пример работы оператора RAND

Оператор RANDI

Синтаксис:

X = randi(m)

X = randi(a, m)

X = randi([range], m,n)

Функция X = randi(m) формирует случайное целое число в интервале [0, m].

Функция X = randi(a, m) формирует массив размера m x m, элементами которого являются случайные целые числа в интервале [0, a].

Функция X = randi([range], m, n) формирует массив размера m x n, элементами которого являются независимые случайные целые числа из диапазона [range]. На рисунке 1.12 представлен пример работы оператора RANDI.

```
>> x=randi(10, 4)

x =

    10     4     1     9
     7     7     3     7
     8     2     1     4
     8     8     1    10
```

Рисунок 1.12 – Пример работы оператора RANDI

В MatLab также можно сгенерировать множество рациональных чисел в заданном интервале.

Для генерации случайных рациональных чисел необходимо:

1. Из наибольшего значения, которое может быть задано, вычесть наименьшее значение;
2. Умножить полученное число на оператор генерации RANDI;
3. Прибавить к произведению наименьшее значение.

Сгенерируем матрицу 4x4, элементами которой будут являться случайные рациональные числа в интервале [-5, 15]:

1. 15-(-5)=20;
2. 20*rand(4);
3. 20*rand(4)-5 (рисунок 1.13).

```
>> x=20*rand(4)-5

x =

   -4.3111   10.9040    7.9263    8.5941
    3.7749   -1.2625    9.1873    8.1020
    2.6312    4.7953   10.0937   -1.7478
   10.3103    3.9117    0.5205   -2.6200
```

Рисунок 1.13 – Пример генерации матрицы рациональных чисел

Проверка условия

Оператор проверки условия позволяет организовать разветвление исполнения программы. Внешний вид оператора представлен на рисунке 1.14.

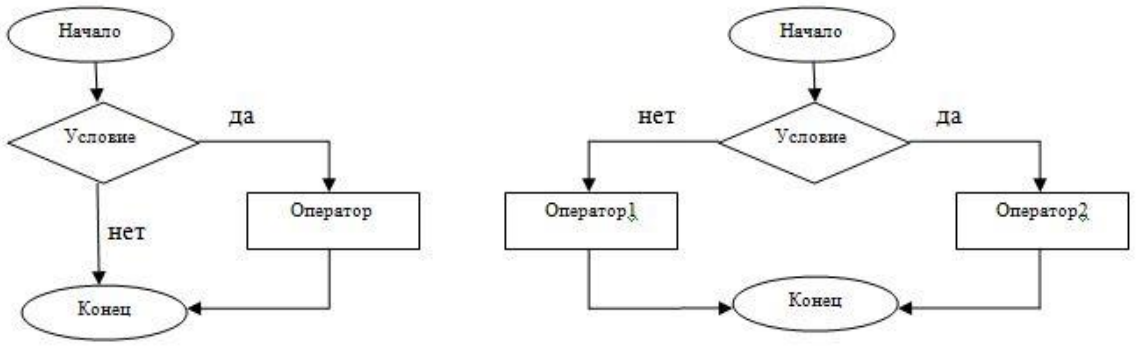


Рисунок 1.14 - Блок-схема условного оператора: редуцированная и полная формы

Формат записи оператора:

```
if <expression 1>
% Executes when the expression 1 is true
<statement(s)>
elseif <expression 2>
% Executes when the boolean expression 2 is true
<statement(s)>
Elseif <expression 3>
% Executes when the boolean expression 3 is true
<statement(s)>
else
%  executes when the none of the above condition is
true
<statement(s)>
end
```

Если условие верно, то выполняются команды MATLAB, размещенные между if и end, а если условие не верно, то происходит переход к командам, расположенных после end.

В зависимости от выполнения того или иного условия работает соответствующая ветвь программы, если все условия неверны, то выполняются команды, размещенные после else.

Пример Оператор switch.

Синтаксис

switch переменная

case значение1

команды1

case значение2
команды2
……
case значениеn
командыn
otherwise
команды
end

Каждая ветвь определяется оператором case, переход в нее выполняется тогда, когда переменная оператора switch принимает значение, указанное после case, или одно из значение из списка case. После выполнения какой-либо из ветвей происходит выход из switch, при этом значения, заданные в других case, уже не проверяются. Если подходящих значений для переменной не нашлось, то выполняется ветвь программы, соответствующая otherwise.

Прерывания цикла. Исключительные ситуации.

Оператор break

Синтаксис

break

Оператор break используется при организации циклических вычислений: for…end, while…end. При выполнении условия

if условие
break
end

оператор break заканчивает цикл (for или while) и происходит выполнение операторов, которые расположены в строках, следующих за end. В случае вложенных циклов break осуществляет выход из внутреннего цикла.

Обработка исключительных ситуаций, оператор try…catch

Синтаксис

try
операторы, выполнение которых может привести к ошибке
catch
операторы, которые следует выполнить при возникновении ошибки в блоке между try и catch
end

Описание

Конструкция try…catch позволяет обойти исключительные ситуации (ошибки, приводящие к окончанию работы программы, например, обращение к несуществующему файлу) и предпринять некоторые действия в случае их возникновения.

Сервисные функции

disp – осуществляет вывод текста или значения переменной в командное окно.

input – осуществляет запрос на ввод с клавиатуры. Используется при создании приложений с интерфейсом из командной строки.

eval – выполняет содержимое строки или строковой переменной, как команды MATLAB

clear – удаляет переменные рабочей среды.

clc – производит очистку командного окна.

Более подробную информацию об этих и других функциях можно узнать, выполнив в командной строке

help имя_функции.

Циклы

MatLAB предоставляет пользователю два способа организации цикла. Первый из них цикл с известным количеством повторений. Блок-схема этого цикла представлена на рисунке 1.15.

Рисунок 1.15– Блок-схема цикла for

Формат оператора:

for count=start:step:final

команды MATLAB
end

Описание
count – переменная цикла,
start – начальное значение переменной цикла,
final – конечное значение переменной цикла,
step – шаг, на который увеличивается count при каждом следующем заходе в цикл, цикл заканчивается, как только значение count становится больше final.

Поле *step* в конструкции оператора не является обязательным.

Например:

```
for i=1:100
  x(i)=sin(2*PI*i/100);
end
```

Тело цикла обязательно заканчивается служебным словом *end*.

При работе с циклом *for* допустимо использование оператора прерывания цикла *break*. При выполнении оператора break работа цикла завершается и управление передается на следующий после конца цикла оператор. Блок-схема программы, иллюстрирующий использование оператора *break*, приведена на рисунке 1.16.

Текст программы соответствующий рисунку 1.16.

```
for i=1:100
  операторы
  if a(i) == 0
          break
  end
  операторы
end
```

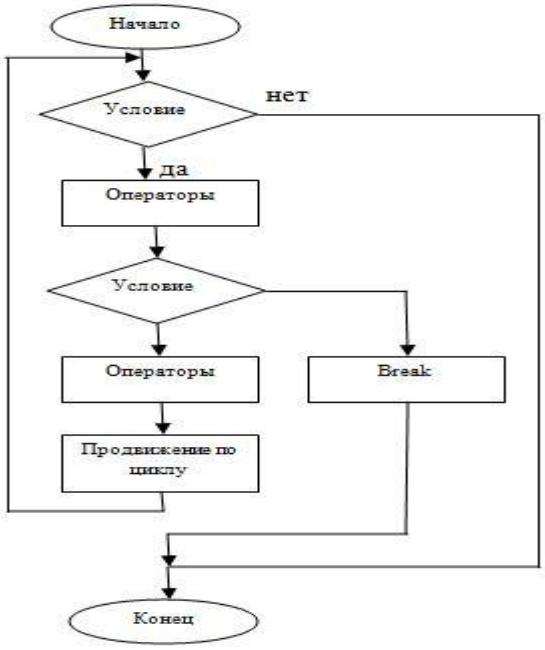


Рисунок 1.16 - Иллюстрация использования оператора break

Пример. Пусть требуется вывести семейство кривых для x€ [0, 2π], которое задано функцией y(x, a) = e-ax sin x, для значений параметра a от -0.1 до 0.1. Ниже приведен листинг файл-программы для вывода семейства кривых.

Листинг программы

```
Figure
x = [0:pi/30:2*pi];
for a = -0.1:0.02:0.1
y = exp (-a*x).*sin(x);
hold on
plot (x, y)
end
```

В результате выполнения программы появится графическое окно, которое содержит требуемое семейство кривых (рисунок 1.17).

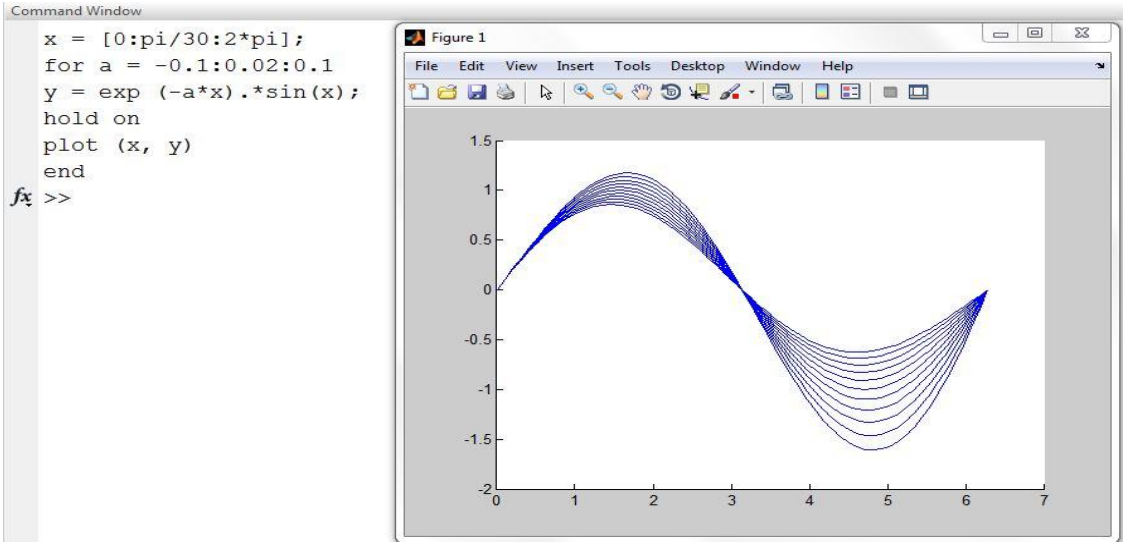


Рисунок 1.17 – Иллюстрация выполнения программы для вывода семейства кривых

Кроме цикла *for* в программировании на MatLAB используется цикл *while*. В отличие от цикла *for* в структуре цикла *while* не предусмотрены элементы для изменения переменной, по которой организован цикл. Эти элементы должен предусмотреть сам пользователь в операторах цикла.

Формат оператора цикла с неизвестным количеством повторений:

while условие

операторы

end

Тело цикла обязательно заканчивается служебным словом *end*.

Цикл работает, пока выполняется (истинно) условие цикла. Для задания условия выполнения цикла допустимы следующие операции отношения (таблица 1.2)

Таблица 1.2 - Операции отношения.

Обозначение	Операция отношения
==	Равенство
<	Меньше
<=	Меньше или равно
>=	Больше или равно

>	Больше
~=	Не равно

Задание более сложных условий производится с применением логических операторов. Логические операторы приведены в таблице 2.

Таблица 1.3 - Логические операторы

Оператор	Условие	Эквивалентная запись
Логическое «и»	x<3 и k=4	(x<3) & (k==4)
Логическое «или»	x=1 или x=2	(x==1) | (x==2)
Отрицание «не»	a≠1.9	~(a==1.9)

Пример. Программа определения точности вычислений:

```
a=1 ;
while a+1 ~= 1
  a=a/2;
end
a
```

Последняя строчка программы выведет на экран значение переменной

Основы работы с матрицами в системе MatLab

1. Создание матриц. Простейшей операций с матрицей является ее создание. Для создания строки необходимо указать его имя, знак равенства и в квадратных скобках через запятую или через пробел перечислить значения элементов:

```
>> A = [1 2 3 4 5];
```

В случае если необходимо создать столбец чисел, то в качестве разделителя выступает символ точка с запятой:

```
>> B = [1 ; 3 ; 5 ; 7];
```

Для создания квадратной или прямоугольной матрицы понадобится чередовать оба этих способа.

```
>> C = [1 2 3 ; 4 5 6 ; 7 8 9];
```

2. Создание матриц специального вида. Для генерации векторов пользователю предоставляется следующая команда:

<*Имя вектора*>=<*Начальное значение*>:<*Шаг*>:<*Конечное значение*>

```
>> X = 6:0.2:26;
```

В результате получиться вектор X следующего вида:

6	6.2	6.4	6.6	...	25.6	25.8	26

В математике часто встречаются матрицы специального вида. Ниже приведен ряд из них:

- Единичная матрица, рисунок 1.18(а). В единичной матрице все элементы равны нулю, кроме элементов стоящих на главной диагонали (матрица является квадратной). Для создания единичной матрицы Вам необходимо подать команду <*Имя матрицы*>=*eye*(<*Размер*>);

>> a = eye(4);

- Матрица со всеми единицами, рисунок 1.18(б). Эта матрица содержит единицы во всех ячейках. Для создания матрицы необходимо указать <*Имя матрицы*>=*ones*(<*Кол-во строк*>, <*Кол-во столбцов*>);

>> b = ones(3, 4);

- Нулевая матрица, рисунок 1.18(в). Эта матрица содержит во всех своих ячейках одни нули. Для создания необходимо выполнить следующую команду:

<*Имя матрицы*>=*zeros*(<*Кол-во строк*>, <*Кол-во столбцов*>);

>> c = zeros(4, 2);

- Случайная матрица, рисунок 1.18(г). Все значения этой матрицы получаются с генератора случайных чисел. Для создания такой матрицы необходимо дать следующую команду: <*Имя матрицы*>=*rand*(<*Кол-во строк*>, <*Кол-во столбцов*>);

>> d = randi(6, 2, 2);

1	0	0	0		1	1	1	1		0	0		5	6					
0	1	0	0		1	1	1	1		0	0		4	1					
0	0	1	0		1	1	1	1		0	0		0	3					
0	0	0	1							0	0								
	а					б					в			г					

Рисунок 1.8 - Специальные матрицы

3. Доступ к ячейкам матрицы. Для доступа к ячейкам матрицы необходимо указать имя матрицы, номер строки и номер столбца. Нумерация строк и столбцов ведется с ЕДИНИЦЫ! Номера пишутся в круглых скобках. Общий формат записи: <*Имя массива*>(<*Номер строки*>, <*Номер столбца*>);

```
>> d(3,2)
ans = 2
>> d(3,1) = 9;
>> d
ans =
  5     6
  4     1
  0     3
```

4. Скалярные операции. В математике для всех матриц определена операция умножения (деления) матрицы на скаляр (число) – «.*» («./»). Все

значения матрицы в этом случае умножаются (делятся) на это число. Все скалярные (поэлементные) операции в MatLAB обозначаются при помощи точки (т.е. для сложения «.+», для вычитания «.–» и т.д.). Следует помнить, что отсутствие точки перед знаком действия приводит к выполнению матричной операции, которая может потребовать выполнения специальных требований к операндам (например, совпадения размеров).

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)
A(3,1)	A(3,2)	A(3,3)

.* r =

A(1,1).*r	A(1,2).*r	A(1,3).*r
A(2,1).*r	A(2,2).*r	A(2,3).*r
A(3,1).*r	A(3,2).*r	A(3,3).*r

5. Сложение, вычитание скаляра из матрицы. Кроме операции умножения матрицы на скаляр, для матрицы и скаляра определены операции сложение «+» и вычитания «–». Действия так же выполняются с каждой ячейкой матрицы отдельно.

6. Сложение, вычитание матриц. Эта операция допустима только с матрицами одинакового размера. При выполнении операции действие выполняется с соответствующими друг другу ячейками.

Пример:

a(1,1)	a(1,2)	a(1,3)
a(2,1)	a(2,2)	a(2,3)
a(3,1)	a(3,2)	a(3,3)

+

b(1,1)	b(1,2)	b(1,3)
b(2,1)	b(2,2)	b(2,3)
b(3,1)	b(3,2)	b(3,3)

=

a(1,1)+ b(1,1)	a(1,2)+ b(1,2)	a(1,3)+ b(1,3)
a(2,1)+ b(2,1)	a(2,2)+ b(2,2)	a(2,3)+ b(2,3)
a(3,1)+ b(3,1)	a(3,2)+ b(3,2)	a(3,3)+ b(3,3)

Сумма. Найдем сумму двух матриц.

```
format short
>> M=[1 2; 3 4];
>> N=[3 4; 1 2];
>> C=M+N
C = 4 6
    4 6
```

Аналогично ищется разность и частное.

Если надо провести операции сразу над всем массивом, то перед знаком операции ставится точка. Например, оператор (*) означает умножение для векторов и матриц, а оператор (.*) – поэлементное умножение всех элементов массива.

Применим к векторам **v** и **u** операцию деления (/):

```
>> v=[2 6 10];
>> u=[1 3 5];
>> v/u
```

ans = 2 – скаляр, выражающий отношение векторов.

Применим к векторам **v** и **u** операцию почленного деления (./):

```
>> v./u
```

ans = 2 2 2 – вектор

7. Произведение матриц. При выполнении операции перемножения матриц выполняется последовательное умножение строки на столбец. При этом количество столбцов в первой матрице должно равняться количеству строк во второй матрице. Матрица результата будет иметь столько же строк сколько и в первой матрице, и количество столбцов равное количеству столбцов во второй матрице.

a(1,1)	a(1,2)	a(1,3)		b(1,1)	b(1,2)	b(1,3)		a(1,1)+ b(1,1)	a(1,2)+ b(1,2)	a(1,3)+ b(1,3)
a(2,1)	a(2,2)	a(2,3)	+	b(2,1)	b(2,2)	b(2,3)	=	a(2,1)+ b(2,1)	a(2,2)+ b(2,2)	a(2,3)+ b(2,3)
a(3,1)	a(3,2)	a(3,3)		b(3,1)	b(3,2)	b(3,3)		a(3,1)+ b(3,1)	a(3,2)+ b(3,2)	a(3,3)+ b(3,3)

Умножим матрицу на 2, т.е. получим матрицу, каждый элемент которой в два раза больше элемента исходной матрицы. Для умножения матрицы на скаляр допустимы оба оператора.

```
format short
M=[1 2 3; 4 5 6];
>>D=M*2
>>D = 2  4  6
        8 10 12
```

При умножении матрицы на матрицу или вектор следует различать умножение справа и слева:

```
>> format short
>> M=[1 2; 3 4];
>> N=[4 5; 6 7];
>> C=M*N
C = 16 19
     36 43
>> D=N*M
D = 19 28
     27 40
```

8. Удаление отдельных столбцов или строк. Для удаления отдельных столбцов или строк матрицы используются пустые квадратные скобки [].

Рассмотрим матрицу M=[1 4 7; 2 5 8; 3 6 9].

Удалим второй столбец, используя оператор двоеточие (:)

```
>> M(:,2)=[]
M = 1 7
     2 8
     3 9
```

Удалим вторую строку:

```
>> M(2,:)=[]
M = 1 7
     3 9
```

9. Преобразование вектора в диагональную матрицу

```
>> v=1:3
v = 1 2 3
>> diag(v)
ans = 1 0 0
      0 2 0
      0 0 3
```

10. Сведение матрицы к треугольному виду

```
>> A=[1 2 3; 6 5 4; 7 8 9];
>> triu(A)
ans = 1 2 3
      0 5 4
      0 0 9
```

11. Конкатенация. Операция конкатенации – объединение малых матриц в большую

```
A=[1 2; 3 4];
B=[2 4; 5 6];
C=[A A+B;A-B B]
C = 1  2  3 6
    3  4  8 10
   -1 -2 2 4
```

12. Вычисление суммы столбцов матрицы

```
>> sum(C)
ans = 1 2 18 26
```

13.Вычисление суммы строк матрицы

```
>> sum(C.')
ans = 12 25 3 7
```

14. Вычисление суммы диагоналей матрицы

```
>> sum(diag(C))
ans = 13
```

15. Размерность матрицы (функция size)

```
>> A=[1 2; 5 6]
>> size(A)
ans = 2 2
```

16. Обратная матрица (функция inv)

```
>> A=[1 2; 3 4];
>> B=inv(A)
B = -2.0000 1.0000
     1.5000 -0.5000
```

17. Определитель (функция det)

```
>> det(A)
ans = -2
```

18.Транспонирование

```
>> A'
ans = 1 3
      2 4
```

Предусмотрено также выполнение следующих операций с матрицами и векторами:

- Сложение, вычитание (**+, -**);
- Умножение (*****);
- Обращение (**inv**);
- Деление **(/)**;
- Возведение в степень (^);
- Трансформирование (‘).
- Создание нижней треугольной матрицы А: **tril(A).**
- Создание верхней треугольной матрицы А: **triu(A).**
- Вращение матрицы А относительно вертикальной оси: **fliplr(A).**
- Вращение матрицы А относительно горизонтальной оси: **flipud(A).**
- Поворот матрицы А на кратное 90° значение: **rot90(A,k)**, где k=±1, ±2,... – множитель, на который умножается угол 900.
- Формирование единичной матрицы заданного размера n: **eye(n)**.
- Формирование единичной матрицы по размеру данной квадратной матрицы А: **eye(size(A)).**
- Матрица единиц данного размера n X m: **ones(n,m)**. Для создания квадратной матрицы: **ones(n)**.
- Матрица единиц по размеру заданной матицы А: **ones(size(A))**.
- Матрица нулей данного размера n X m: **zeros(n,m)**. Для создания квадратно матрицы: **zeros(n).**
- Матрица нулей по размеру заданной матрицы А: **zeros(size(A)).**
- Извлечение диагонали заданной матрицы А: **diag(A).**
- Вычисление следа матрицы А: **trace(A).**
- Магический квадрат размера n (n>2): **magic(n)**.
- Создание диагональной матрицы по заданной матрице А: **diag(diag (A))**.
- Собственные числа действительной или комплексной матрицы А: **eig(A)**.
- Получение помощи для заданной встроенной функции: **help-пробел-функция.**
- Операции с массивами (перед знаком арифметического действия ставится точка), например: [1 2 3; 4 5 6].^2 - возведение каждого элемента матрицы в квадрат.
- Формирование коэффициентов характеристического полинома заданной числовой матрицы А: **poly(A)**.
- Формирование характеристического полинома заданной числовой матрицы А: **poly(sum(A)).** По умолчанию независимой

переменной полинома является x; независимая переменная полинома может назначаться (например s): poly(sum(A),sum('s')).

- Формирование коэффициентов характеристического полинома матрицы A по ее заданным собственным числам: **poly(eig(F)).**
- Формирование характеристического полинома по заданным корням, являющимися элементами вектора P: **poly(P).**
- Формирование полинома с коэффициентами, являющимися элементами заданного вектора P: **poly2sum(P).** Степень полинома на единицу меньше размерности заданного вектора P.
- Размерность матрицы A: **size(A).**
- Суммирование элементов столбцов матрицы A: **sum(A).** Результат – строка, состоящая из сумм элементов каждого столбца матрицы A.
- Формирование произведения элементов столбцов матрицы A: **prod(A).**
- Формирование матрицы с элементами из возможных перестановок элементов заданного числового вектора P: **perms(P).**
- Суммирование элементов вектора P: **sum(P).** Результат – число.
- Длина вектора P: **length(P).**

Общая постановка задачи

Вычислить значение выражения, соответствующего Вашему варианту. Построить график функции y=f(x), соответствующей Вашему варианту. Сгенерировать матрицу чисел, соответствующую условиям Вашего варианта. Написать файл-программу, которая бы выполняла задание, соответствующее Вашему варианту. Осуществить операции над матрицами согласно Вашего варианта.

Список индивидуальных данных

Задание 1. Вычислить значение выражения, соответствующего Вашему варианту:

Номер варианта	**Задание**
1	$1-\frac{2(\sin 58)^2}{8\cos 116}$
2	$\frac{9\sin 132}{\sin 228}$
3	$\frac{258\sin 179\cos 179}{\sin 358}$
4	$\frac{15(\sin^2 69-\cos^2 69)}{\cos 138}$

5	$36\sqrt{3}tg\frac{\Pi}{3}\sin\frac{\Pi}{6}$
6	$46\sqrt{6}\cos\frac{\Pi}{6}\cos\frac{7\Pi}{4}$
7	$\frac{54}{\sin(-\frac{34\Pi}{3})\cos(\frac{35\Pi}{6})}$
8	$16\sqrt{2}\cos 585$
9	$24\sqrt{3}tg(-1020)$
10	$-12\sqrt{2}\sin 225$
11	$10\sqrt{6}\cos(-\frac{\Pi}{4})\sin(-\frac{\Pi}{3})$
12	$\frac{12}{\sin^2 37+\sin^2 127}$
13	$\frac{2\cos(2\Pi-30)-3\sin(-\frac{\Pi}{2}+30)}{2\cos(30-3\Pi)}$
14	$5\cos(2\Pi+30)+4\sin(-\frac{3\Pi}{2}+30)$
15	$\cos\frac{\Pi}{3}-\frac{1}{2}$

Задание 2. Построить график функции, соответствующей Вашему варианту:

Номер варианта	Функция	Интервал построения	Шаг аргумента	Вид графика
1	$y=2x^3-3x^2+x+5$	[-10;10]	0.5	График красного цвета; точки графика «*»; линия – сплошная.
2	$y=\frac{x^3}{x^2+1}$	[-4;4]	0.5	График желтого цвета; точки графика «+»; линия – штрих-пунктирная.
3	$y=\frac{x^2+x}{x^2-3x+2}$	[-4;4]	0.1	График зеленого цвета; точки графика «O»; линия – штриховая.
4	$y=(x^2-2x)e^x$	[-4;4]	0.1	График синего цвета; точки графика «*»; линия – сплошная.
5	$y=\sin^3 x+\cos^3 x$	[-5;5]	0.1	График красного цвета; точки графика «+»; линия – штрих-пунктирная.
6	$y=x^3-3x^2-9x+11$	[-5;5]	0.1	График желтого цвета; точки графика «O»; линия – штриховая.
7	$y=\sqrt[3]{(x^2-1)}$	[-3;3]	0.1	График зеленого цвета; точки графика «*»; линия – сплошная.
8	$y=\frac{e^x}{x}$	[-4;4]	0.1	График синего цвета; точки графика «+»; линия – штрих-пунктирная.

9	$y = -\frac{1}{4}(x^3 - 3x^2 + 4)$	[-3;4]	0.1	График красного цвета; точки графика «О»; линия – штриховая.
10	$y = \ln\frac{x+1}{x+2}$	[-7;5]	0.1	График желтого цвета; точки графика «*»; линия – сплошная.
11	$y = \frac{x}{\sqrt{x^2 + x}}$	[-5;6]	0.1	График зеленого цвета; точки графика «+»; линия – штрих-пунктирная.
12	$y = \frac{x^3 - 1}{4x^2}$	[-4;4]	0.1	График синего цвета; точки графика «О»; линия – штриховая.
13	$y = \frac{x^3}{x^2 - 1}$	[-5;5]	0.1	График красного цвета; точки графика «*»; линия – сплошная.
14	$y = \frac{x^3}{2(x+5)^2}$	[-20;10]	0.5	График желтого цвета; точки графика «+»; линия – штрих-пунктирная.
15	$y = \frac{x^3}{x^2 - 4}$	[-8;8]	0.1	График зеленого цвета; точки графика «О»; линия – штриховая.

Задание 3. Сгенерировать матрицу чисел, соответствующую условиям Вашего варианта:

№ варианта	**Интервал генерации**	**Размер матрицы**	**Тип чисел**
1	[63, 82]	4x1	Рациональные
2	[-75, 83]	2x1	Целые
3	[-81, 27]	1x5	Целые
4	[-45, 9]	4x2	Рациональные
5	[92, 93]	5x1	Рациональные
6	[-69, 95]	3x2	Целые
7	[-3, 92]	4x4	Целые
8	[-72, 60]	1x3	Рациональные
9	[-16, 84]	3x4	Целые
10	[59, 92]	4x4	Рациональные
11	[-93, 31]	2x4	Целые
12	[70, 87]	4x1	Рациональные
13	[36, 52]	1x3	Рациональные
14	[-22, 49]	5x2	Целые
15	[-66, 31]	3x2	Целые

Задание 4. Написать файл-программу, которая бы выполняла задание, соответствующее Вашему варианту:

1. Дано действительное число x. Вычислить x -x3/3! + x5/5! - x7/7! + x9/9! - x11/11! + x13/13! (x=2.5)

2. Даны натуральное n, действительное x. Вычислить sin x + sin2 x + ... + sin (nx) (x=2.5 , n=5)
3. Дано действительное число x. Вычислить (x-2)(x-4)(x-8)...(x-64)(x-1)(x-3)(x-7)...(x-63) (x=2.5)
4. Даны натуральное n, действительное x. Вычислить sin x + sin(sin x) + ... + sin sin... sin x (x=2.5 , n=5)
5. Пусть $v_1=v_2=0$; $v_3=1.5$; $v_i=v_{i-1}*(^{(i+1)}/_{(i^2+1)})-v_{i-2}\cdot v_{i-3}$ i=4,5,..
   Дано натуральное n (n>=4). Получить v_n.(n=10)
6. Пусть $a_0=a_1=1$, $a_i=a_{i-2}+a_{i-1}/2*i$ -1 , i=2,3,... Найти произведение $a_0*a_1*...*a_{14}$.
7. Вычислить сумму 1/1!+1/2!+...1/10!.
8. Вычислить сумму 1/13+1/23+1/33+...1/503.
9. Даны натуральное число n, действительное число x. Вычислить $_xn^3/3n$. (n=10; x=2.5)
10. Даны натуральное число n, действительное число x. Вычислить $_xn^2/2n$. (n=10; x=2.5)
11. Пусть $x_1=x_2=x_3=1$; $x_i=x_{i-1}+x_{i-3}$; i=4,5,... Найти сумму $x_i/2*i$ i=1..100
12. Даны два целых числа A и B (A < B). Найти сумму всех целых чисел от A до B включительно. (A=38, B=51)
13. Даны натуральные числа от 20 до 50. Напечатать те из них, которые делятся на 3, но не делятся на 5.
14. Найти произведение двузначных нечетных чисел, кратных 13.
15. Даны два целых числа A и B (A < B). Найти сумму квадратов всех целых чисел от A до B включительно. (A=1, B=40).

Задание 5. Осуществить операции над матрицами согласно Вашего варианта:

1. Размер матрицы — 128x127
Диапазон генерации случайных чисел — от +10 до +200
Размер первого вектора — 1x127, первое число вектора — +10, шаг — -2
Заменить 35-ю строку матрицы на первый вектор
Размер второго вектора — 128x1
Вставить второй вектор в 10-й столбец
Разбить матрицу на две равные матрицы и перемножить их
Вывести часть полученной матрицы размером 16x16

2. Размер матрицы — 256x32
Диапазон генерации случайных чисел — от -1 до +1
Размер первого вектора — 256x1, первое число вектора — -7, шаг — +1

Заменить 10-й столбец матрицы на первый вектор
Размер второго вектора — 1x32
Вставить второй вектор в 64-ю строку
Разбить матрицу на две равные матрицы и перемножить их поэлементно
Вывести часть полученной матрицы размером 2x8

3. Размер матрицы — 64x127
Диапазон генерации случайных чисел — от -30 до 0
Размер первого вектора — 1x127, первое число вектора — 0, шаг — +2
Заменить 16-ю строку матрицы на первый вектор
Размер второго вектора — 127x1
Вставить второй вектор в 16-й столбец
Разбить матрицу на две равные матрицы и сложить их
Вывести побочную диагональ полученной матрицы

4. Размер матрицы — 128x127
Диапазон генерации случайных чисел — от -10 до 0
Размер первого вектора — 1x127, первое число вектора— +15, шаг — +5
Заменить 15-ю строку матрицы на первый вектор
Размер второго вектора — 128x1
Вставить второй вектор в 11-й столбец
Разбить матрицу на две равные матрицы и вычесть вторую из первой
Вывести часть полученной матрицы размером 16x16

5. Размер матрицы — 127x32
Диапазон генерации случайных чисел — от -17 до +19
Размер первого вектора — 127x1, первое число вектора — -5, шаг — -5
Заменить 2-й столбец матрицы на первый вектор
Размер второго вектора — 1x32
Вставить второй вектор в 4-ю строку
Разбить матрицу на две равные матрицы и перемножить их поэлементно
Вывести часть полученной матрицы размером 8x8

6. Размер матрицы — 128x128
Диапазон генерации случайных чисел — от -1 до +2
Размер первого вектора — 1x128, первое число вектора — 11, шаг — +2
Заменить 16-ю строку матрицы на первый вектор
Размер второго вектора — 128x1
Вставить второй вектор в 5-й столбец
Разбить матрицу на две равные матрицы и перемножить их поэлементно
Вывести часть полученной матрицы размером 16x16

7. Размер матрицы — 16x255

Диапазон генерации случайных чисел — от +5 до +10
Размер первого вектора — 16x1, первое число вектора — -1, шаг — +10
Заменить 100-й столбец матрицы на первый вектор
Размер второго вектора — 1x255
Вставить второй вектор в 4-ю строку
Разбить матрицу на две равные матрицы и перемножить их
Вывести часть полученной матрицы размером 4x16

8. Размер матрицы — 128x127
Диапазон генерации случайных чисел — от +17 до +18
Размер первого вектора — 1x127, первое число вектора — -3, шаг — -2
Заменить 60-ю строку матрицы на первый вектор
Размер второго вектора — 128x1
Вставить второй вектор в 50-й столбец
Разбить матрицу на две равные матрицы и вычесть вторую из первой
Вывести главную диагональ полученной матрицы

9. Размер матрицы — 128x256
Диапазон генерации случайных чисел — от -8 до +20
Размер первого вектора — 1x128, первое число вектора — -9, шаг — +12
Заменить 2-ю строку матрицы на первый вектор
Размер второго вектора — 256x1
Вставить второй вектор в 98-й столбец
Разбить матрицу на две равные матрицы и перемножить их
Вывести главную диагональ полученной матрицы

10. Размер матрицы — 128x127
Диапазон генерации случайных чисел — от +1 до +2
Размер первого вектора — 1x127, первое число вектора — +1, шаг — -4
Заменить 16-ю строку матрицы на первый вектор
Размер второго вектора — 128x1
Вставить второй вектор в 10-й столбец
Разбить матрицу на две равные матрицы и перемножить их
Вывести часть полученной матрицы размером 6x16

11. Размер матрицы — 256x32
Диапазон генерации случайных чисел — от -10 до -1
Размер первого вектора — 256x1, первое число вектора — -3, шаг — +3
Заменить 25-й столбец матрицы на первый вектор
Размер второго вектора — 1x32
Вставить второй вектор в 25-ю строку
Разбить матрицу на две равные матрицы и перемножить их поэлементно
Вывести часть полученной матрицы размером 4x4

12. Размер матрицы — 64x127
Диапазон генерации случайных чисел — от -3 до +10
Размер первого вектора — 1x127, первое число вектора — +4, шаг — +1
Заменить 18-ю строку матрицы на первый вектор
Размер второго вектора — 127x1
Вставить второй вектор в 26-й столбец
Разбить матрицу на две равные матрицы и сложить их
Вывести побочную диагональ полученной матрицы

13. Размер матрицы — 128x127
Диапазон генерации случайных чисел — от -16 до 0
Размер первого вектора — 1x127, первое число вектора — +15, шаг — -3
Заменить 55-ю строку матрицы на первый вектор
Размер второго вектора — 128x1
Вставить второй вектор в 61-й столбец
Разбить матрицу на две равные матрицы и вычесть вторую из первой
Вывести часть полученной матрицы размером 8x8

14. Размер матрицы — 127x32
Диапазон генерации случайных чисел — от -7 до +9
Размер первого вектора — 127x1, первое число вектора — -19, шаг — +2
Заменить 12-й столбец матрицы на первый вектор
Размер второго вектора — 1x32
Вставить второй вектор в 24-ю строку
Разбить матрицу на две равные матрицы и перемножить их поэлементно
Вывести часть полученной матрицы размером 16x8

15. Размер матрицы — 16x255
Диапазон генерации случайных чисел — от +5 до +50
Размер первого вектора — 16x1, первое число вектора — -10, шаг — +4
Заменить 3-й столбец матрицы на первый вектор
Размер второго вектора — 1x255
Вставить второй вектор в 8-ю строку
Разбить матрицу на две равные матрицы и перемножить их
Вывести часть полученной матрицы размером 4x8

Пример выполнения работы

Рассмотрим пример построения графика функции $y=\sin x \exp(-x)$.

```
x=-5:0.5:5; % задание промежутка [-5;5] с шагом 0,1
y=sin(x).*exp(-x); % задание функции y
```

```
    plot(x,y,['R','*','-.']) % выведение графика
красного цвета (R), точки графика в виде снежинок (*),
линии штрихпунктирные (-.)
    grid on % задание сетки
```

График функции приведен на рисунке 1.19.

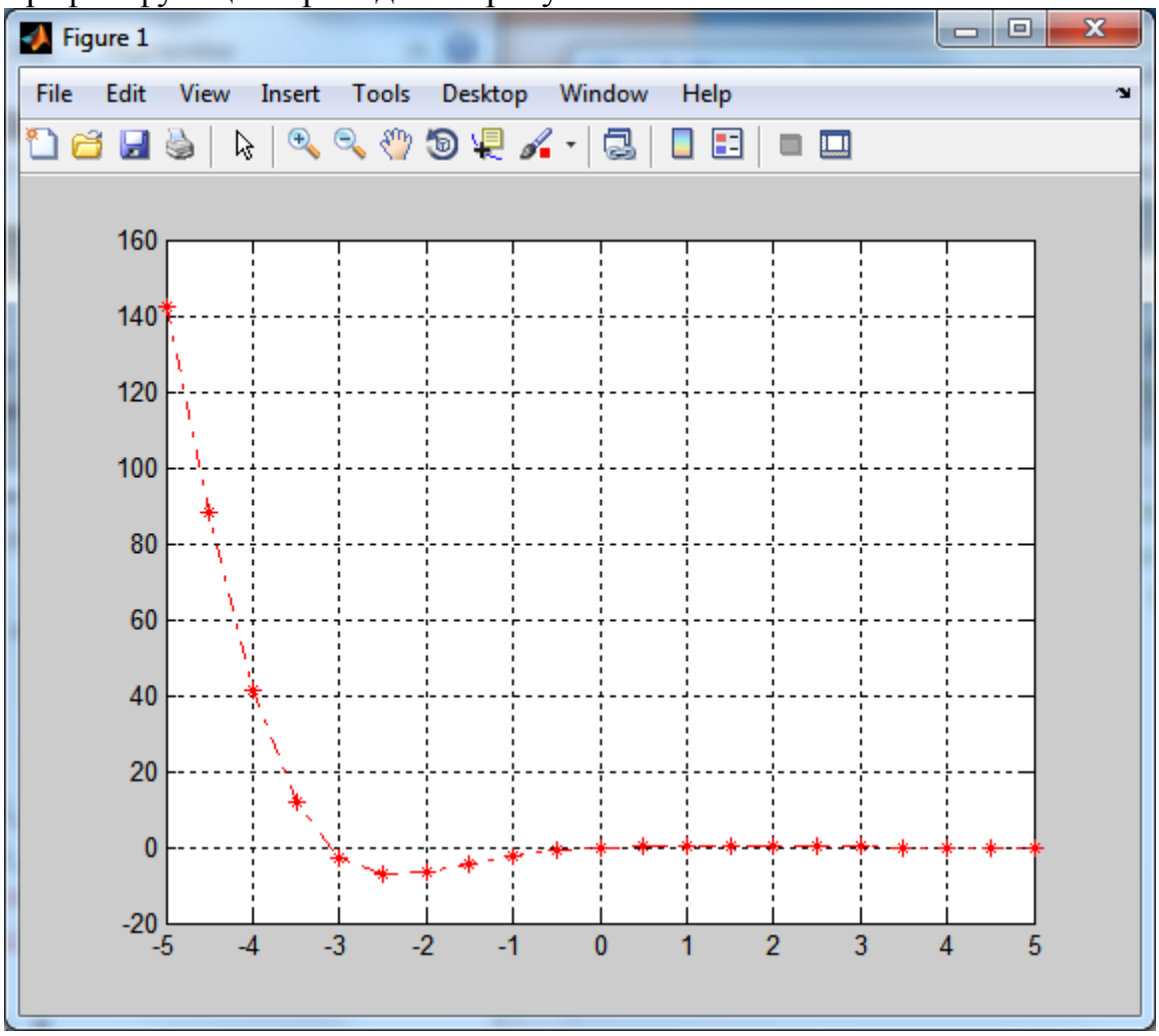


Рисунок 1.19 – График функции $y=sinx\ e^{-x}$

Контрольные вопросы к защите

1. Какая функция строит график?
2. Какие типы графиков может строить система MatLab?
3. Для чего предназначен subplot?
4. Какие можно выполнять действия над матрицами?
5. Какая команда создает единичную матрицу?
6. Какая команда создает нулевую матрицу?
7. Каким образом складываются две матрицы?
8. Как удалить выбранную строку (столбец) матрицы?
9. Как преобразовать вектор в диагональную матрицу?
10. Операция транспонирования матриц?
11. Какая операция считает определитель?

12. Выполнение какой операции в результате приводит к обратной матрице?

13. Каким образом вычисляется сумма строк столбцов?

Лабораторная работа №2. Использование нечетких операций при построении функции принадлежности

Цель работы: Изучить операции, выполняемые над нечеткими множествами.

Теоретическая часть

Нечеткие множества и нечеткая логика

С точки зрения характеристической функции, нечеткие множества есть естественное обобщение обычных множеств, когда мы отказываемся от бинарного характера этой функции и предполагаем, что она может принимать любые значения на отрезке [0, 1]. В теории нечетких множеств характеристическая функция называется функцией принадлежности, а ее значение $\mu_A(x)$ — степенью принадлежности элемента x нечеткому множеству A.

Более строго, нечетким множеством A называется совокупность пар

$$A = \{\langle x, \mu_A(x)\rangle \mid x \in U\}$$

где $\mu_A(x)$ — функция принадлежности, т.е. $\mu_A(x) : U \to [0,1]$.

Операции над нечеткими множествами

Над нечеткими множествами можно производить различные операции, при этом необходимо определить их так, чтобы в частном случае, когда множество является четким, операции переходили в обычные операции теории множеств, то есть операции над нечеткими множествами должны обобщать соответствующие операции над обычными множествами. При этом обобщение может быть реализовано различными способами, из-за чего какой-либо операции над обычными множествами может соответствовать несколько операций в теории нечетких множеств. Для определения пересечения и объединения нечетких множеств наибольшей популярностью пользуются следующие три группы операций:

1. Максиминные:

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\},$$
$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}.$$

2. Алгебраические:

$$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x),$$
$$\mu_{A\cap B}(x) = \mu_A(x)\mu_B(x).$$

3. Ограниченные:

$$\mu_{A\cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\},$$

$$\mu_{A\cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}.$$

Дополнение нечеткого множества во всех трех случаях определяется одинаково: $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$.

Более детально представление операций над нечеткими множествами представлено в темах 1 и 2 настоящего учебного пособия.

Кусочно-линейные функции принадлежности

Наиболее характерным примером таких функций являются "треугольная" (рисунок 2.1, а) и "трапециевидная" (рисунок 2.1, б) функции принадлежности.

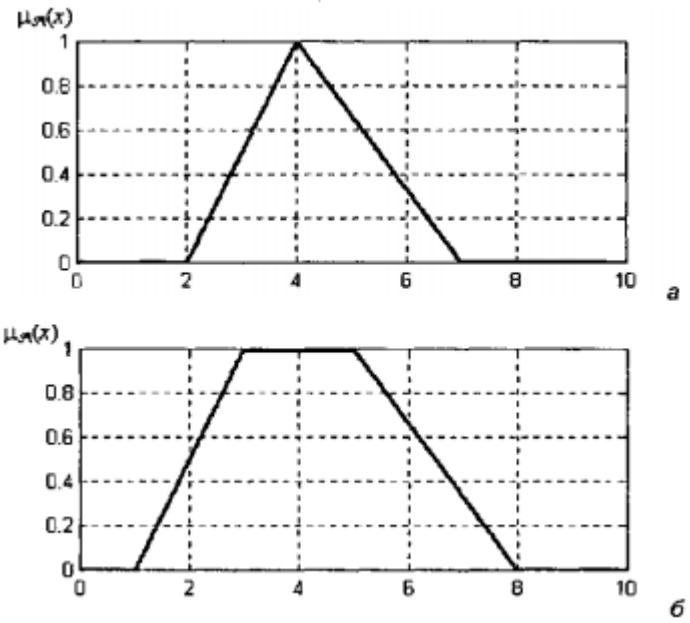


Рисунок 2.1 - Графики функций принадлежности треугольной (а) и трапециевидной (б) формы

Первая из этих функций принадлежности в общем случае может быть задана аналитически следующим выражением:

$$f_{\Delta}(x;, a, b, c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le c \le x \\ 0, & c \le x \end{cases},$$

где a, b, c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a \le b \le c.$ Как нетрудно заметить, параметры a и c характеризуют основание треугольника, а параметр b — его вершину.

Трапециевидная функция принадлежности в общем случае может быть задана аналитически следующим выражением:

$$f_T(x;,a,b,c,d) = \begin{Bmatrix} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{Bmatrix},$$

где a, b, c, d— некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a \le b \le c \le d$.

В качестве частных случаев Z- и S-образных кривых удобно рассматривать так называемую линейную Z-образную функцию (рисунок 2.2, а) и линейную S-образную функцию (рисунок 2.2, б). Первая из этих функций в общем случае может быть задана аналитически следующим выражением:

$$f_{\downarrow}(x;a,b) = \begin{Bmatrix} 1, & x \le a \\ \frac{b-x}{b-a}, & a < x < b \\ 0, & b \le x \end{Bmatrix},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b.

Вторая из этих функций в общем случае может быть задана аналитически следующим выражением:

$$f_{\uparrow}(x;a,b) = \begin{Bmatrix} 0, & x \le a \\ \frac{x-a}{b-a}, & a < x < b \\ 1, & b \ge x \end{Bmatrix},$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: a<b.

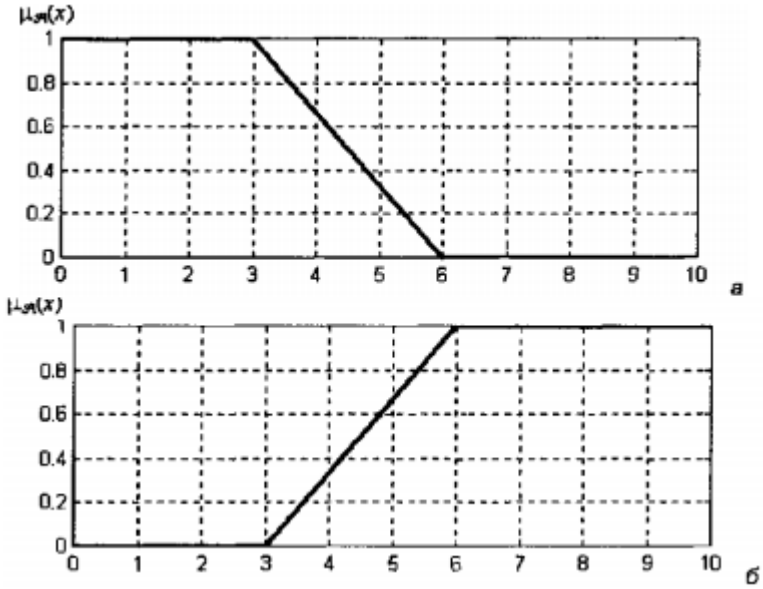


Рисунок 2.2 - Графики линейной Z-образной функции (а) и линейной S-образной функции (б) принадлежности для значений параметров a=3, b=6

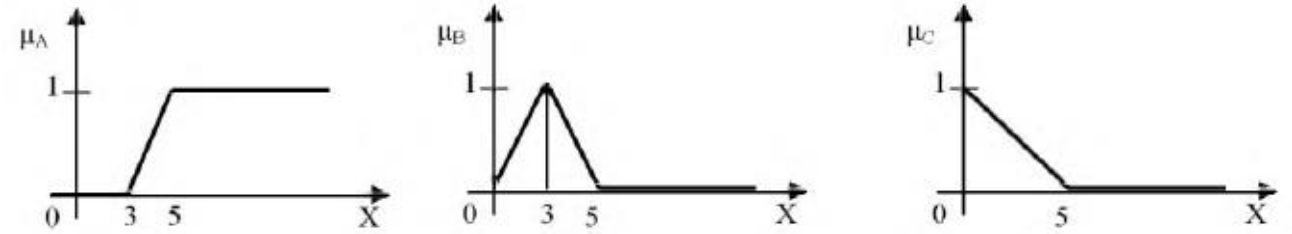
Общая постановка задачи

Дано 3 нечетких множества в форме функций принадлежности. Необходимо построить функцию принадлежности заданного нечеткого множества D согласно Вашего варианта и определить степень принадлежности одного элемента множеству D, используя один из подходов. При этом необходимо рассчитать и построить графики:

1) $A \cap B$; $A \cap C$; $C \cap B$; $A \cap C \cap B$;
2) $A \cup B$; $A \cup C$; $C \cup B$; $A \cup C \cup B$;
3) $A \setminus B$; $A \setminus C$; $C \setminus B$; $C \setminus A$; $B \setminus A$; $B \setminus C$;
4) $\overline{A}$; $\overline{C}$; $\overline{B}$; $\overline{D}$;
5) $\sup_U \mu_A(x)$; $\sup_U \mu_B(x)$; $\sup_U \mu_C(x)$; $\sup_U \mu_D(x)$.

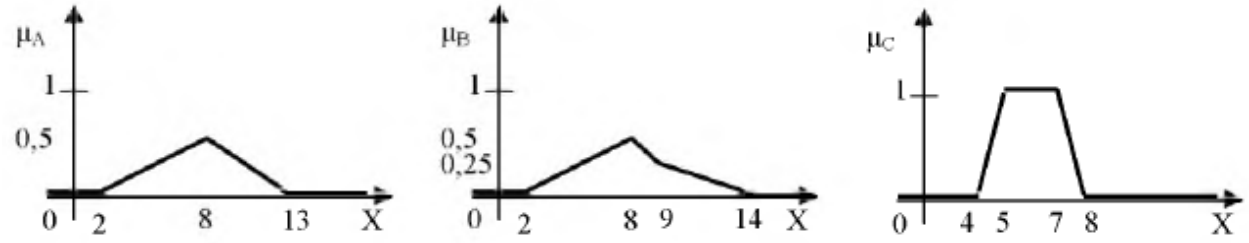
Список индивидуальных данных

1. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

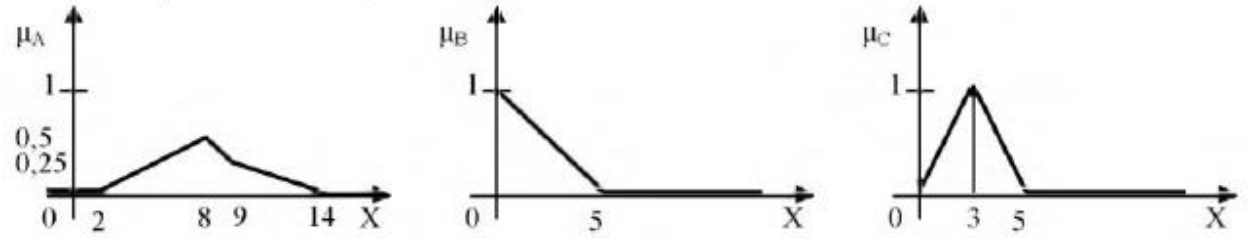


2. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого

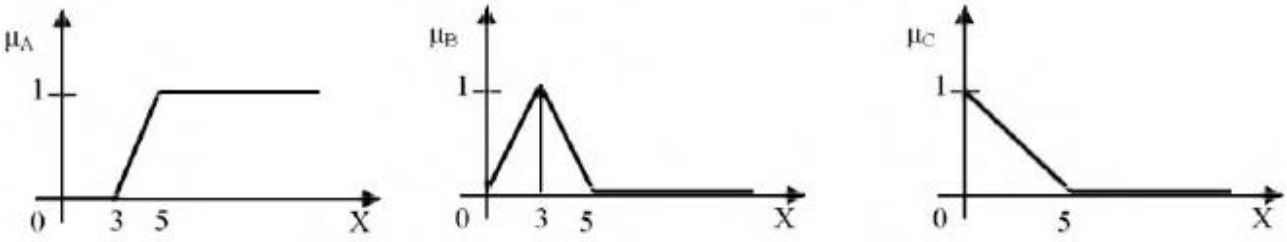
множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



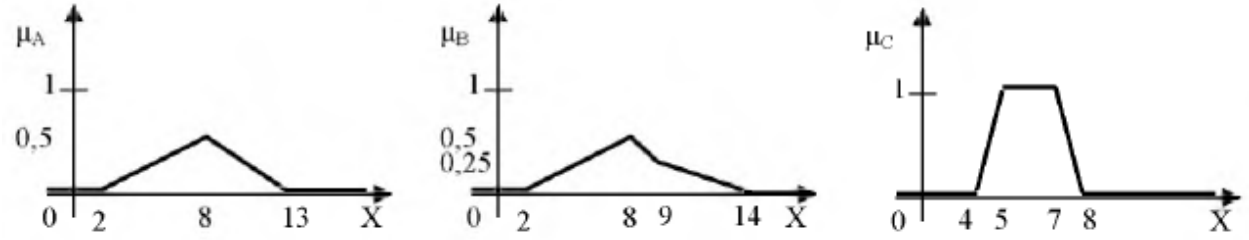
3. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



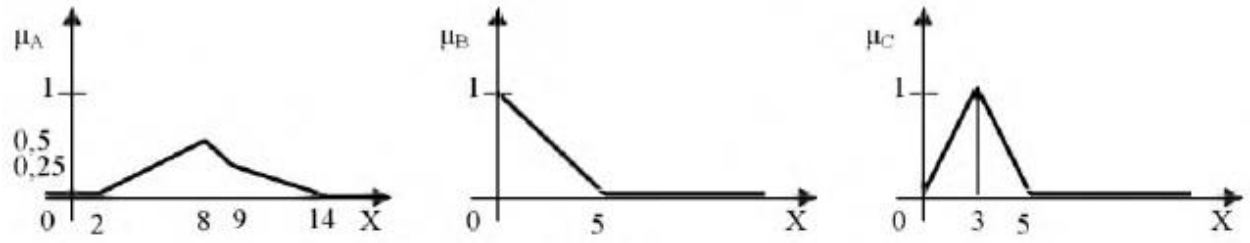
4. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



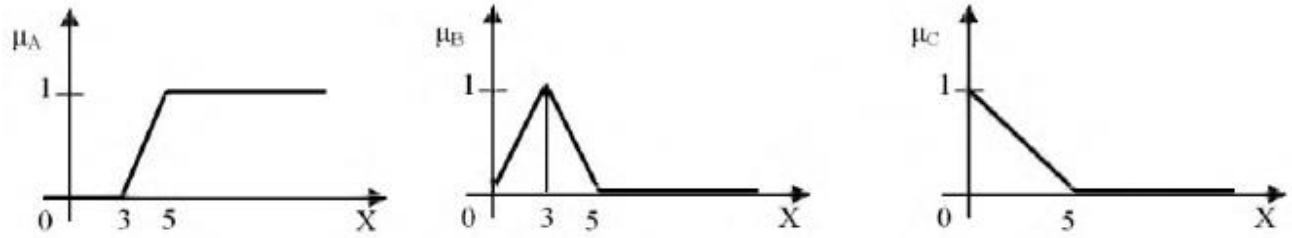
5. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



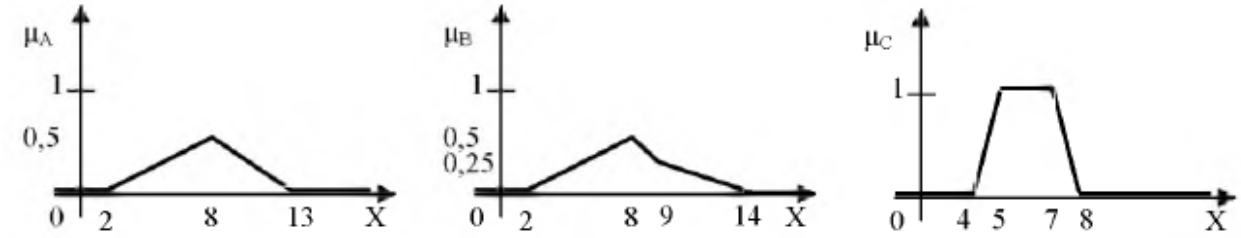
6. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



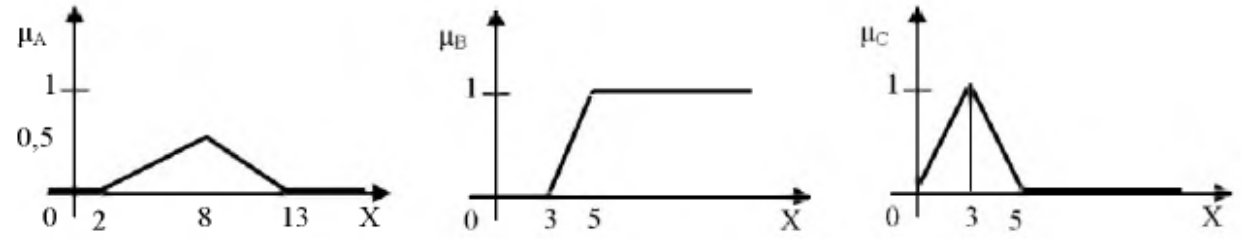
7. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



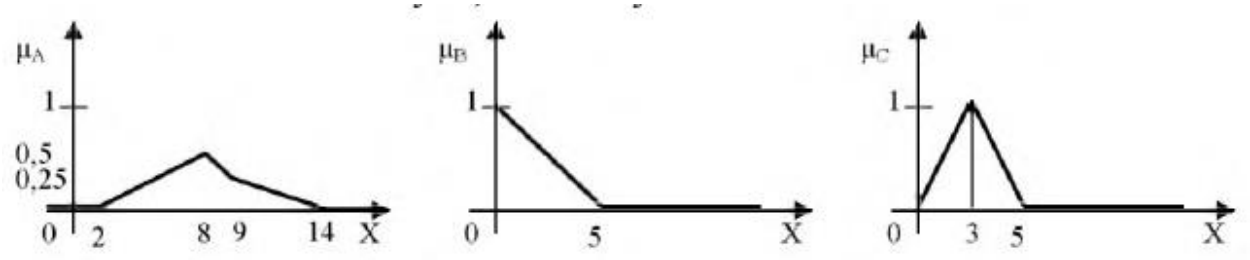
8. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



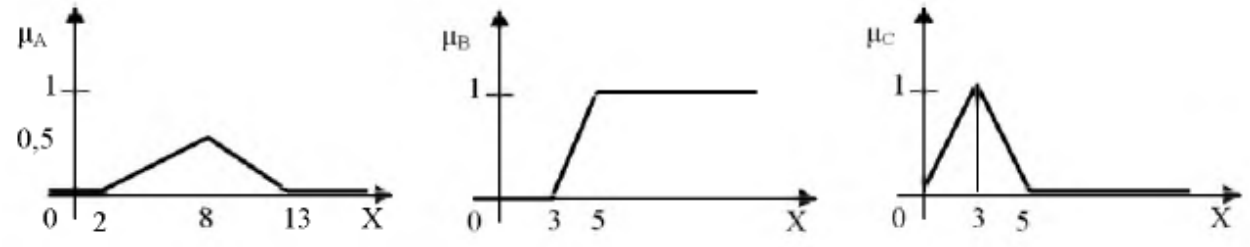
9. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



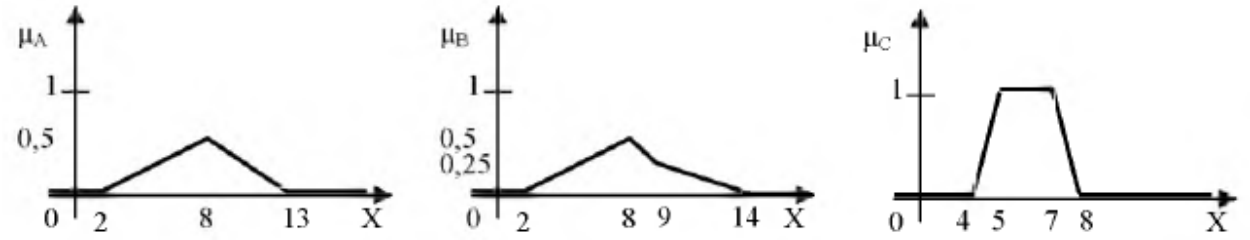
10. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



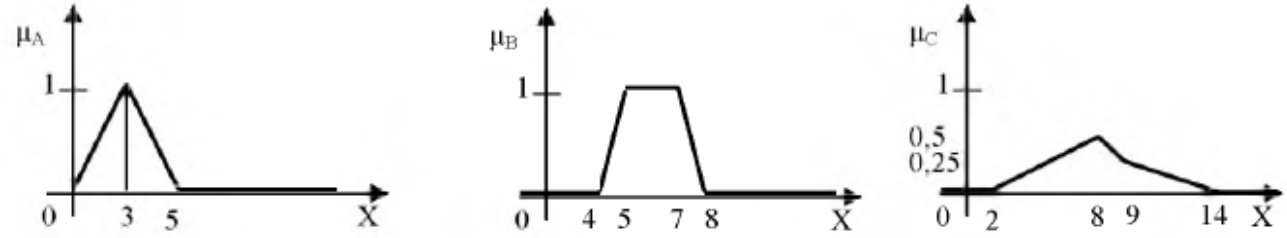
11. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



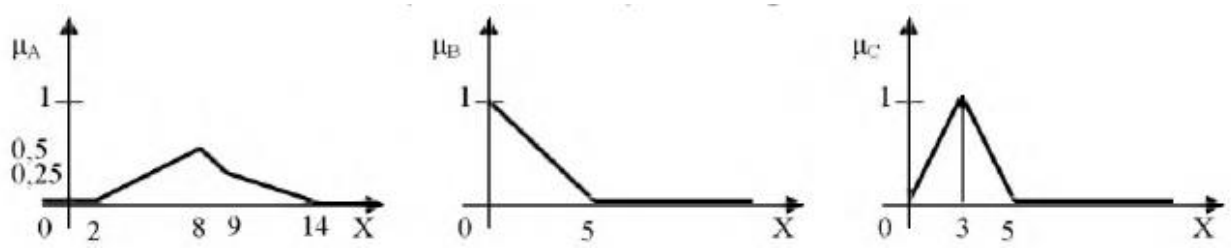
12. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



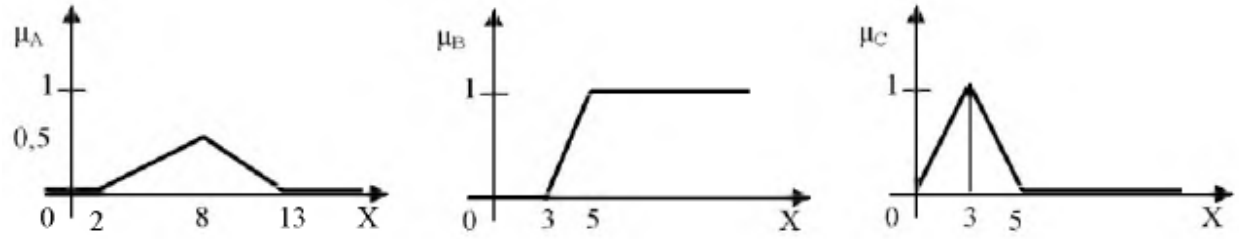
13. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = (\overline{A} \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



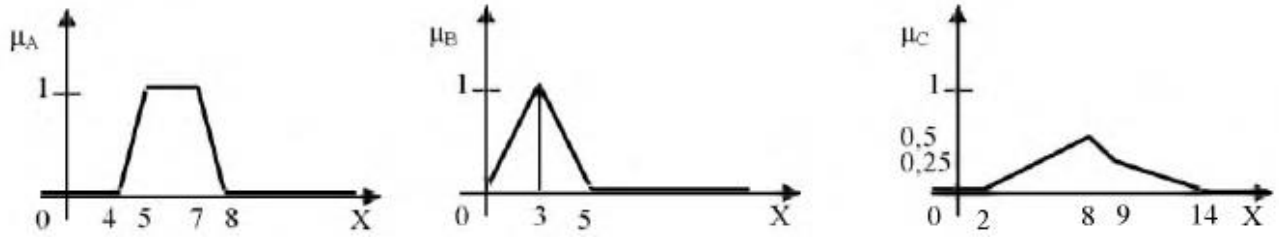
14. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



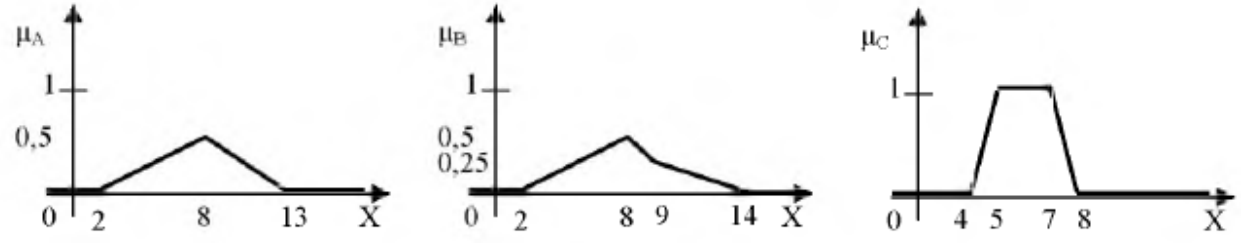
15. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



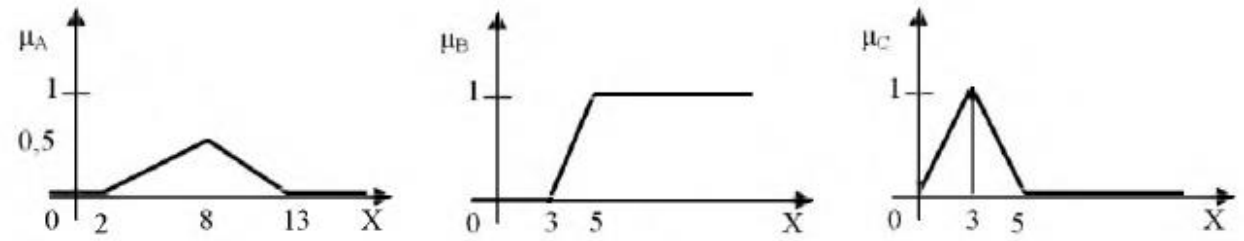
16. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



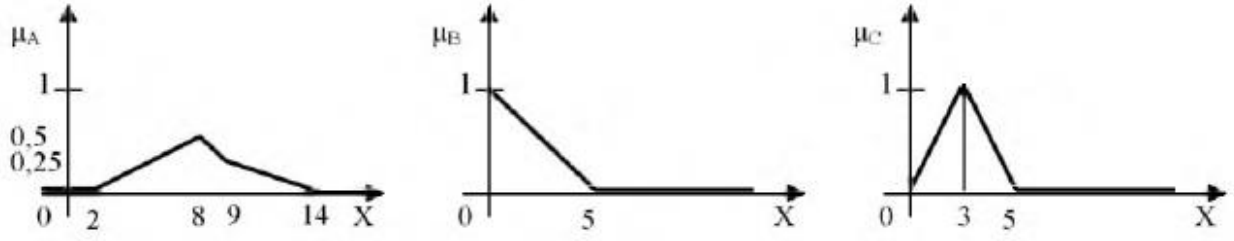
17. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



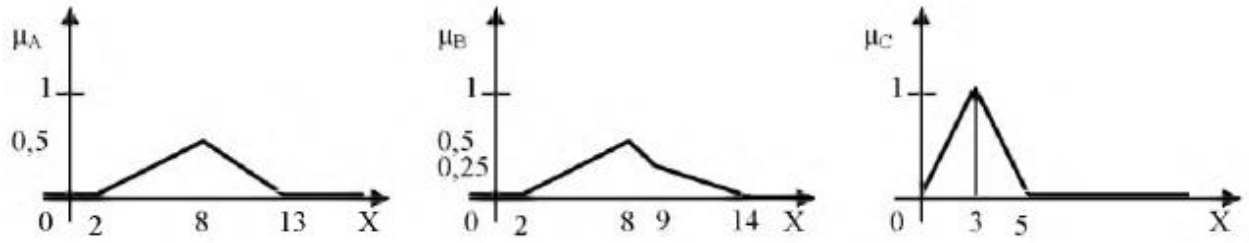
18. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



19. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup \overline{B} \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

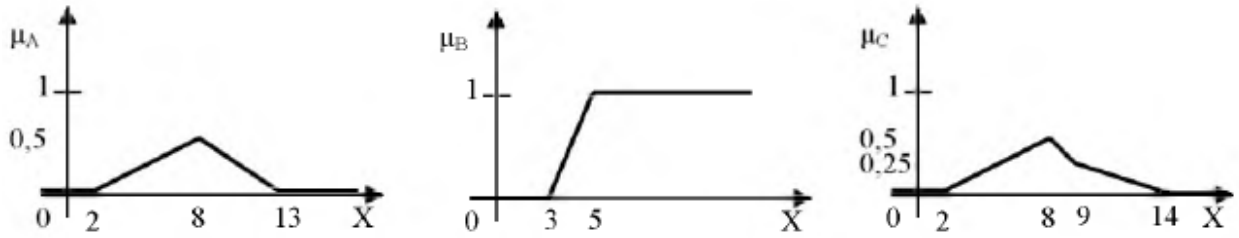


20. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup \overline{B} \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



Пример выполнения работы

Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (A \cup C \cup B)$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



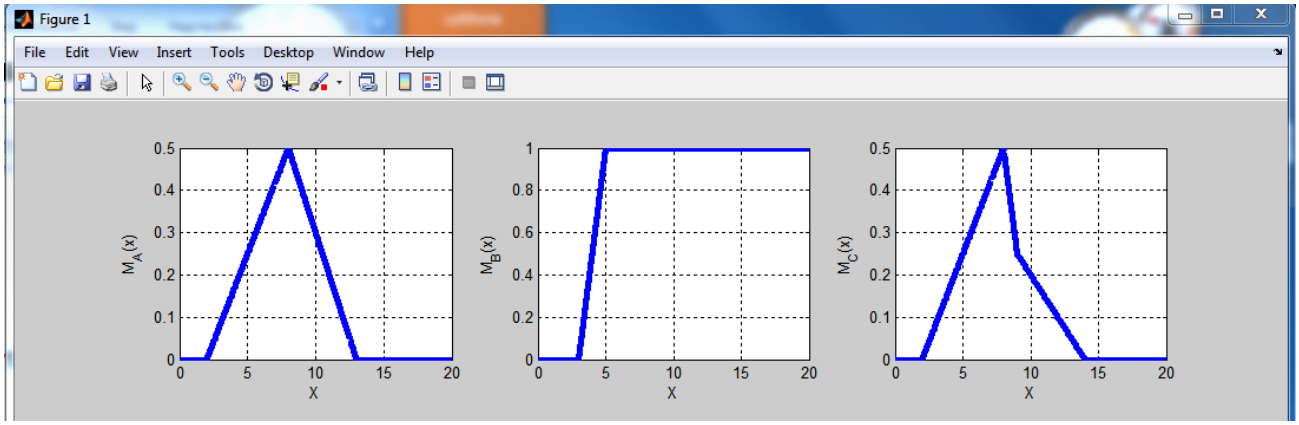

```
t = 0:0.01:20; % определяем число значений по оси абсцисс
a=2; b=8; c=13; %Параметры функции принадлежности M_A
a1=3; b1=5; %Параметры функции принадлежности M_B
a2=2; b2=8; c2=9; d2=14; %Параметры функции принадлежности M_C
% Рассчитываем значения оси ординат функций принадлежности M_A,
M_B, M_C
```

```
j=1;
for i = 0:0.01:20
  if i<=a M_A=0;
  elseif (i>a)&(i<=8) M_A(j)=0.5*(i-a)/(b-a);
  elseif (i>b)&(i<=13) M_A(j)=0.5*(c-i)/(c-b);
  elseif (i>c) M_A(j)=0; end;

  if i<=a1 M_B=0;
  elseif (i>a1)&(i<=b1) M_B(j)=(i-a1)/(b1-a1);
  elseif (i>b1) M_B(j)=1; end;

  if i<=a2 M_C=0;
  elseif (i>a2)&(i<=b2) M_C(j)=0.5*(i-a2)/(b2-a2);
  elseif (i>b2)&(i<=c2) M_C(j)=-0.25*i+2.5;
  elseif (i>c2)&(i<=d2) M_C(j)=0.25*(d2-i)/(d2-c2);
  elseif (i>d2) M_C(j)=0; end

  j=j+1;
end
% Строим график функции принадлежности M_A, M_B, M_C
subplot(2, 3, 1)
plot(t, M_A, 'LineWidth', 4); grid on; xlabel('X');
ylabel('M_A(x)');
subplot(2, 3, 2)
plot(t, M_B, 'LineWidth', 4); grid on; xlabel('X');
ylabel('M_B(x)');
subplot(2, 3, 3)
plot(t, M_C, 'LineWidth', 4); grid on; xlabel('X');
ylabel('M_C(x)');
```



Описание процесса решения. Для построения функции принадлежности нового множества необходимо:

1) Определить последовательность выполнения операций в формуле.

2) Построить на отдельных графиках промежуточные множества, согласно определенной последовательности действий. Свести промежуточные множества на одном графике и определить итоговую функцию принадлежности.

3) Используя определенный в задаче метод, определить аналитически степень принадлежности элемента, входящего в ядро итогового множества.

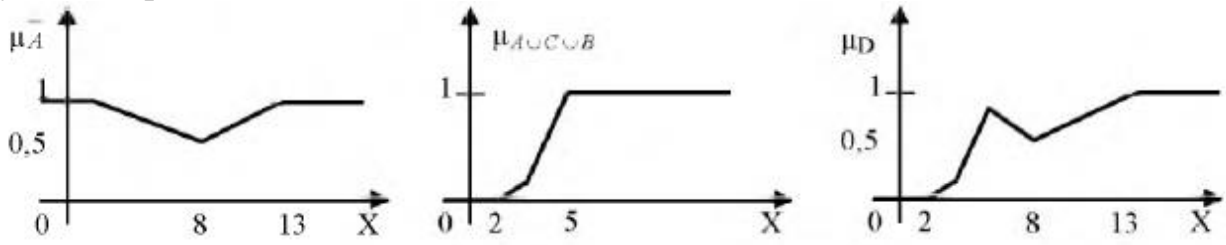
4) Проверить аналитические вычисления по построенному графику функции принадлежности.

Решение.

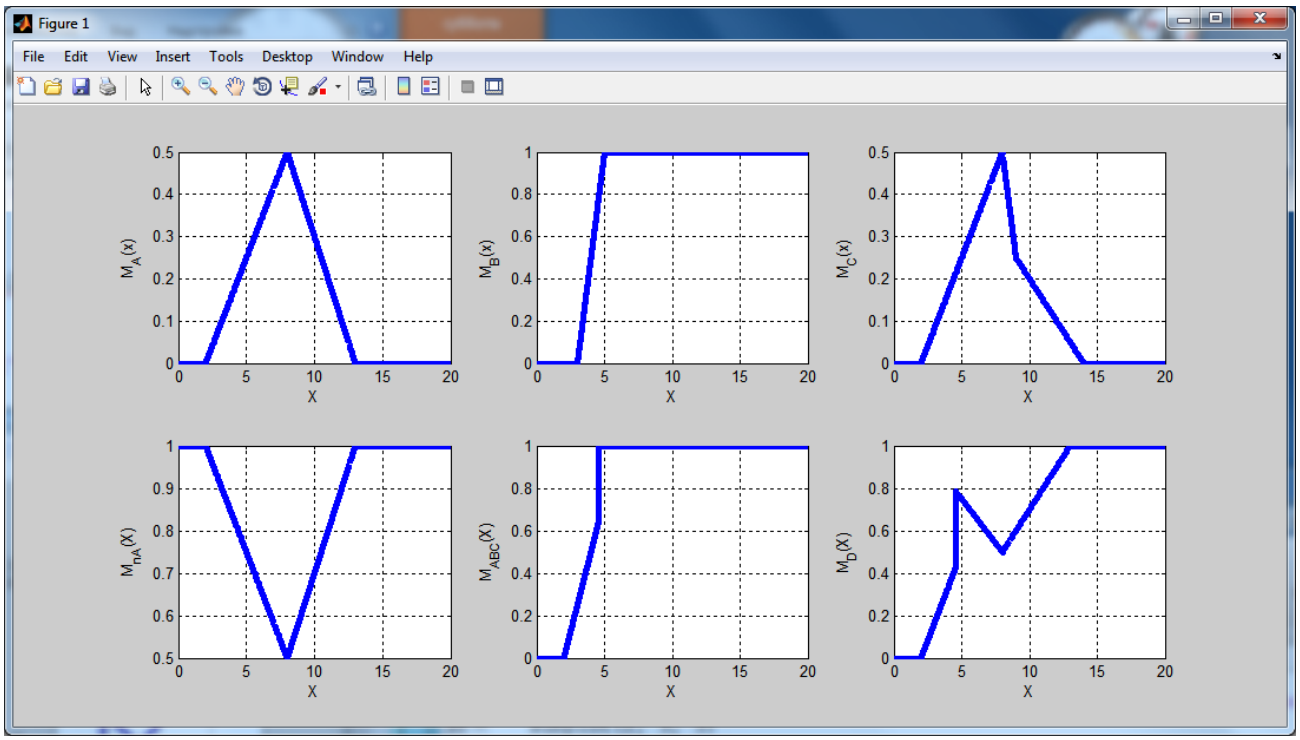
1) Множество $D = \overline{A} \cap (A \cup C \cup B)$, значит, последовательность операций будет следующей: $\overline{A}$, $A \cup C \cup B$, $\overline{A} \cap (A \cup C \cup B)$.

```
% Рассчитываем значения последовательности операций
j=1;
for i = 0:0.01:20
    M_n_A(j)=1-M_A(j); % Рассчитываем значение отрицания A
(M_n_A)
    if (M_A(j)+M_B(j))<1 M_A_B(j)=M_A(j)+M_C(j); else
M_A_B(j)=1; end;% Рассчитываем значение объединения A и B
(M_A_B)
    if (M_A_B(j)+M_C(j))<1 M_A_B_C(j)=M_A_B(j)+M_C(j); else
M_A_B_C(j)=1; end; %Рассчитываем значение объединения (M_A_B_C)
    if (M_A_B_C(j)+M_n_A(j)-1)>0 M_D(j)=M_A_B_C(j)+M_n_A(j)-1;
else M_D(j)=0; end; % Рассчитываем значение M_D
    j=j+1;
end
```

2) Построим согласно этой последовательности операций графики функций принадлежности:




```
% Строим графики функций рассчитанных функций принадлежности
subplot(2, 3, 4)
plot(t, M_n_A, 'LineWidth', 4); grid on; xlabel('X');
ylabel('M_n_A(X)');
subplot(2, 3, 5)
plot(t, M_A_B_C, 'LineWidth', 4); grid on; xlabel('X');
ylabel('M_A_B_C(X)');
subplot(2, 3, 6)
plot(t, M_D, 'LineWidth', 4); grid on; xlabel('X');
ylabel('M_D(X)');
```



3) Носитель множества D состоит из элементов из интервала [2, 15]. Ядро множества D состоит из элементов [13, 14, 15]. Выберем элемент 8.

$\mu_A(8) = 0{,}5;$

$\mu_B(8) = 1;$

$\mu_C(8) = 0{,}5;$

$\mu_{\bar{A}}(8) = 1 - \mu_A(8) = 1 - 0{,}5 = 0{,}5;$

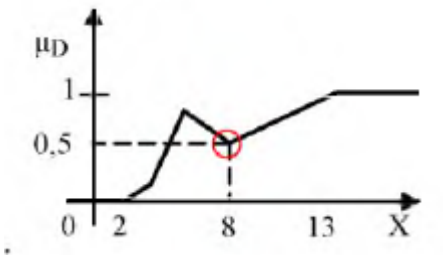
$\mu_{\bar{C}}(8) = 1 - \mu_C(8) = 1 - 0{,}5 = 0{,}5;$

$\mu_{A\cup C}(8) = \min\{1, \mu_C(8) + \mu_A(8)\} = \min\{1{,}1+1\} = 1;$

$\mu_{A\cup C\cup B}(8) = \min\{1, \mu_{A\cup C}(8) + \mu_B(8)\} = \min\{1{,}1+1\} = 1;$

$\mu_{\bar{A}\cap(A\cup C\cup B)}(8) = \max\{0, \mu_{\bar{A}}(8) + \mu_{A\cup C\cup B}(8) - 1\} = \max\{0{,}0.5+1-1\} = 0.5.$

4) $\mu_D(8) = 0{,}5.$



Контрольные вопросы к защите

1. Каким образом задается нечеткое множество?
2. Назовите операции над нечеткими множествами?
3. Что такое функция принадлежности?

4. Каким образом задается функция принадлежности?
5. Может ли функция принадлежности изменяться в интервале [-1, +1]?

Лабораторная работа №3. Исследование способов формирования нечетких множеств и операций над ними в Fuzzy Logic Toolbox

Цель работы: изучить методы построения нечетких множеств с использованием различных типов функций принадлежности. Ознакомиться с наиболее распространенными логическими операциями над нечеткими множествами.

Теоретическая часть

Встроенные функции принадлежности пакета Fuzzy Logic Toolbox

Fuzzy Logic Toolbox включает 11 встроенных функций принадлежностей. Для удобства имена всех встроенных функций принадлежности оканчиваются на **mf**. Вызов функции принадлежности осуществляется следующим образом: **namemf(x, params)**,

где **namemf** – наименование функции принадлежности;

x – вектор, для координат которого необходимо рассчитать значения функции принадлежности;

params – вектор параметров функции принадлежности.

Встроенные функции принадлежности используют следующие основные функции:

1. Кусочно-линейные функции принадлежности:

1.1. **trimf** - треугольная функция принадлежности. Порядок параметров: [a, b, c].

$$\mu(x) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$$

1.2. **trapmf** - трапециевидная функция принадлежности. Порядок параметров: [a, b, c, d].

$$\mu(x) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases}$$

1.3. **smf** - s-подобная функция принадлежности. Порядок параметров: [a b].

$$\mu(x) = \begin{cases} 0, & x \le a \\ \text{нелинейная аппроксимация}, & a < x < b \\ 1, & x \ge b \end{cases}$$

1.4. **zmf** - z-подобная функция принадлежности. Порядок параметров: [a b].

$$\mu(x) = \begin{cases} 1, & x \le a \\ \text{нелинейная аппроксимация}, & a < x < b \\ 0, & x \ge b \end{cases}$$

2. Функции принадлежности, построенные на основаннии гауссовского распределения:

2.1. **gaussmf** - симметричная гауссовская функция принадлежности. Порядок параметров: [c b].

$$\mu(x) = e^{-\frac{(x-b)^2}{2c^2}}$$

2.2. **gauss2mf** - двухсторонняя гауссовская функция принадлежности. Порядок параметров: [a1 c1 a2 c2].

если c1<c2, то $\mu(x) = \begin{cases} \exp\left((x-c_1)^2/(-2a_1^2)\right), & x < c_1 \\ 1, & c_1 \le x \le c_2 \\ \exp\left((x-c_2)^2/(-2a_2^2)\right), & x > c_2 \end{cases}$

если c1>c2, то $\mu(x) = \begin{cases} \exp\left((x-c_1)^2/(-2a_1^2)\right), & x < c_2 \\ \exp\left((x-c_1)^2/(-2a_1^2)\right) * \exp\left((x-c_2)^2/(-2a_2^2)\right) & c_2 \le x \le c_1 \\ \exp\left((x-c_2)^2/(-2a_2^2)\right), & x > c_1 \end{cases}$.

3. Функции принадлежности, построенные на основании сигмоидной кривой:

3.1. **sigmf** - сигмоидная функция принадлежности. Порядок параметров: [a c].

$$\mu(x) = \frac{1}{1 + e^{-a(x-c)}}$$

3.2. **psigmf** - произведение двух сигмоидных функций принадлежности. Порядок параметров: [a1 c1 a2 c2].

$$\mu(x) = \frac{1}{1 + e^{-a_1(x-c_1)}} \cdot \frac{1}{1 + e^{-a_2(x-c_2)}}$$

3.3. **dsigmf** - функция принадлежности в виде разности между двумя сигмоидными функциями. Порядок параметров: [a1 c1 a2 c2].

$$\mu(x) = \frac{1}{1 + e^{-a_1(x-c_1)}} - \frac{1}{1 + e^{-a_2(x-c_2)}}$$

4. Функции принадлежности, построенные на основании квадратической и кубической кривых:

4.1. **gbellmf** - обобщенная колокообразная функция принадлежности. Порядок параметров: [a b c].

$$\mu(x) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}}$$

4.2. **pimf** - пи-подобная функция принадлежности. Порядок параметров: [a b c d], [a d] – носитель нечеткого множества; [b c] – ядро нечеткого множества. Является произведением **smf** и **zmf** функций.

Простейшие функции принадлежности треугольная (**trimf**) и трапециевидная (**trapmf**) формируется с использованием кусочно-линейной аппроксимации. Трапециевидная функция принадлежности является обобщение треугольной, она позволяет задавать ядро нечеткого множества в виде интервала. В случае трапециевидной функции принадлежности возможна следующая удобная интерпретация: ядро нечеткого множества – оптимистическая оценка; носитель нечеткого множества – пессимистическая оценка.

Две функции принадлежности – симметричная гауссовская (**gaussmf**) и двухстороняя гауссовская (**gaussmf**) формируется с использованием гауссовского распределения. Функция **gaussmf** позволяет задавать ассиметричные функция принадлежности. Обобщенная колоколообразная функция принадлежности (**gbellmf**) по своей форме похожа на гауссовские. Эти функции принадлежности часто используются в нечетких системах, так как на всей области определения они является гладкими и принимают ненулевые значения.

Функции принадлежности **sigmf, dsigmf, psigmf** основаны на использовании сигмоидной кривой. Эти функции позволяют формировать функции принадлежности, значения которых начиная с некоторого значения аргумента и до + (-) ∞ равны 1. Такие функции удобны для задания лингвистических термов типа “высокий” или “низкий”.

Полиномиальная аппроксимация применяется при формировании функций **zmf, pimf** и **smf**, графические изображения которых похожи на функции **sigmf, dsigmf, psigmf**, соответственно.

На рисунке 3.1 приведены графические изображения функций принадлежности, полученные с помощью демонстрационной сценария **mfdemo**. Как видно из рисунка, встроенные функции принадлежности позволяют задавать разнообразные нечеткие множества.

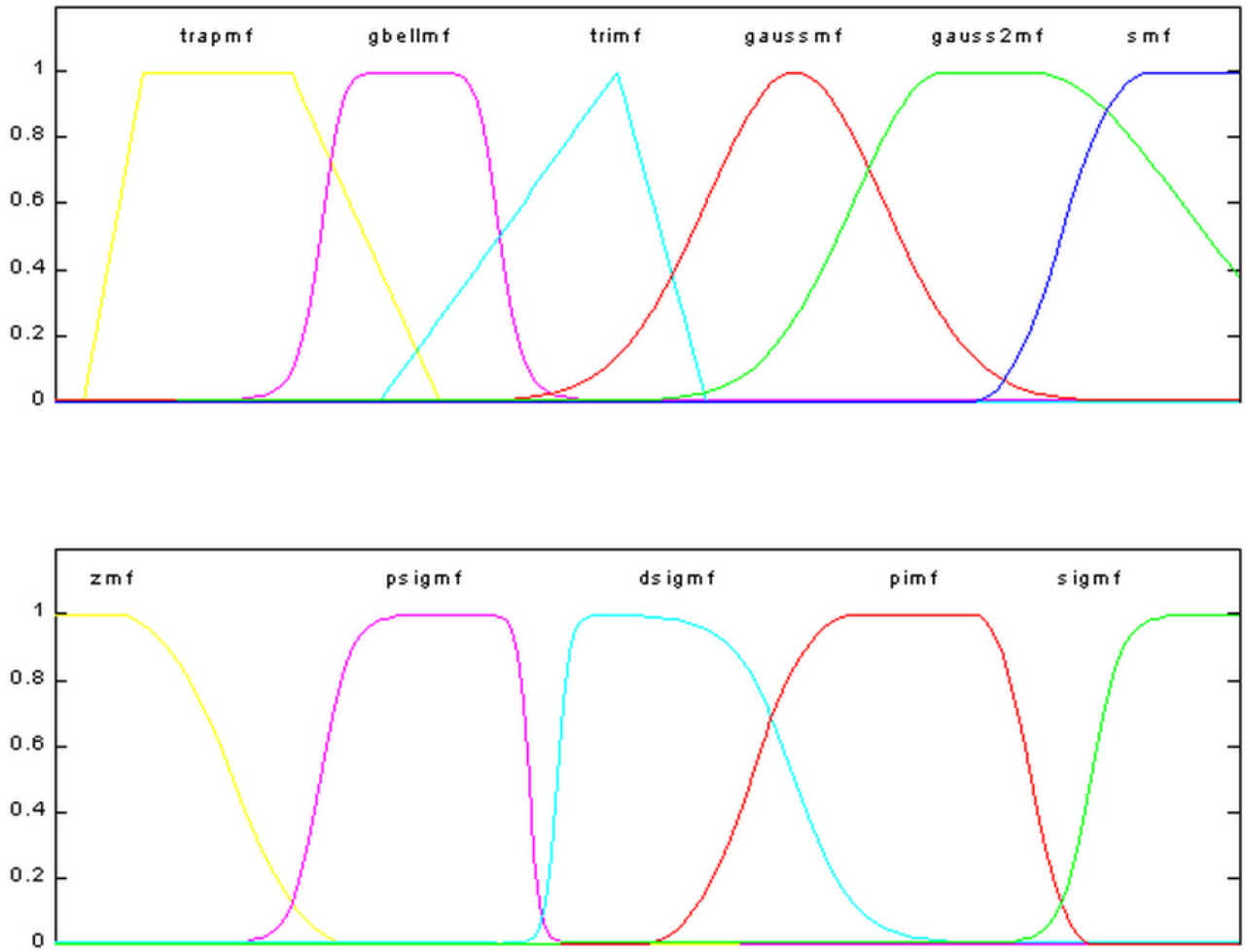


Рисунок 3.1 - Встроенные функции принадлежности

В Fuzzy Logic Toolbox предусмотрена возможность для пользователя создания собственной функции принадлежности. Для этого необходимо создать m-функцию, содержащую два входных аргумента – вектор, для координат которого необходимо рассчитать значения функции принадлежности и вектор параметров функции принадлежности. Выходным аргументом функции должен быть вектор степеней принадлежности. Ниже приведена m-функция, реализующая колоколообразную функцию принадлежности $\mu(x) = \frac{1}{1 + (\frac{x - b}{a})^2}$ (рисунок 3.2):

```
x=0:0.1:5;
a=1;
b=2;
mu=1./(1+ ((x-b)/a).^2);
plot(x, mu);
```

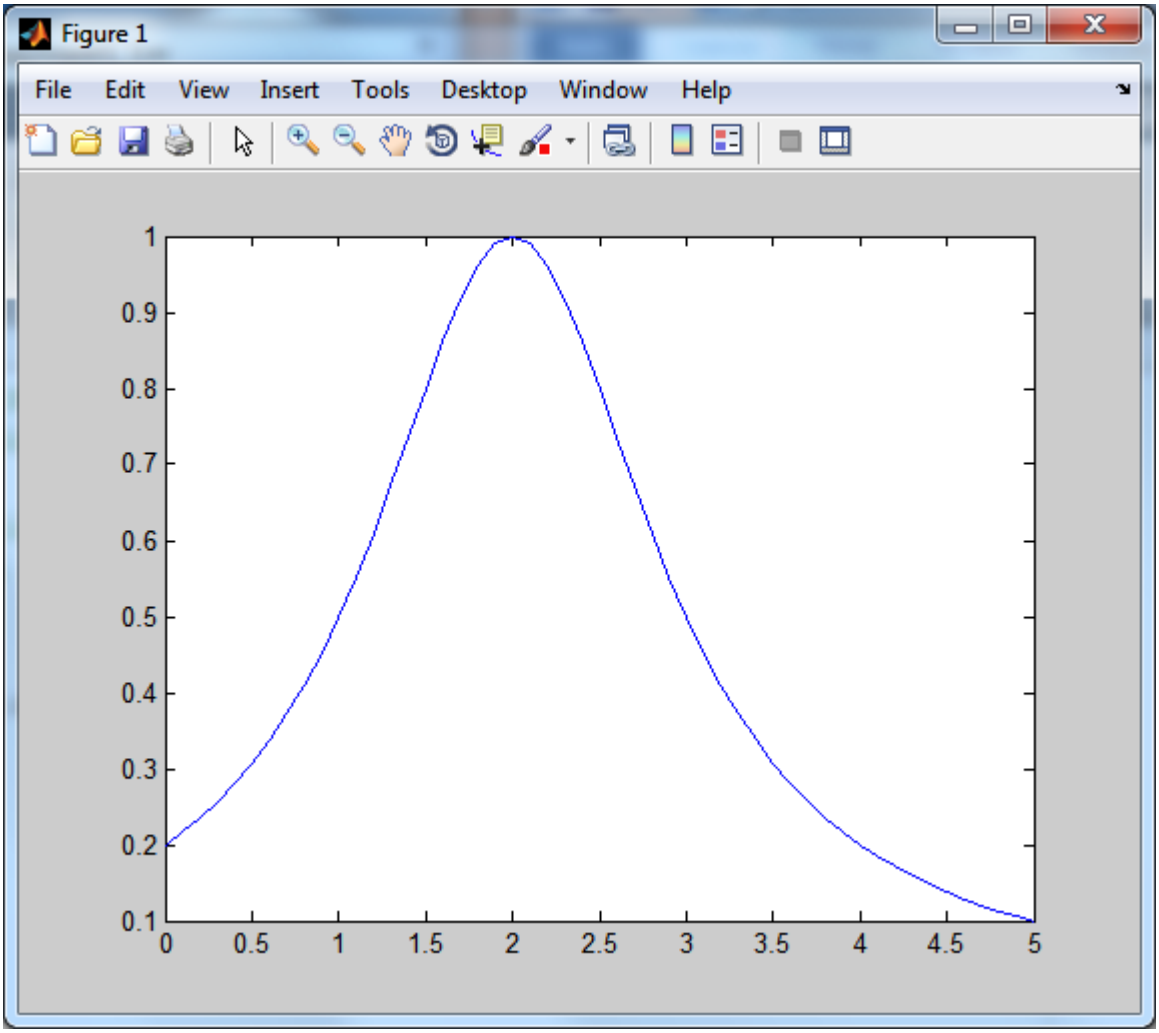


Рисунок 3.2 – Колоколообразная функция принадлежности

Операции с нечеткими множествами

Выделяют три основные логические операции с нечеткими множествами: конъюнкцию, дизъюнкцию и логическое отрицание. В среде Matlab существует возможность определять конъюнктивные и дизъюнктивные операторы с точки зрения минимаксной и вероятностной интерпретаций.

Рассмотрим минимаксную интерпретацию логических операторов, в которой конъюнктивный оператор представляет нахождение минимума (min), а дизъюнктивный — максимума (max):

y = min ([y1; y2]) .

y = max ([y1; y2]) .

Параметры y1 и y2 представляют собой исходные ФП. Функция min работает со списком ФП. В Matlab список оформляется квадратными скобками, а элементы списка разделяются точкой с запятой.

Минимаксная интерпретация является наиболее распространенной при построении нечетких систем. Тем не менее на практике довольно часто

используется альтернативная вероятностная интерпретация конъюнктивных и дизъюнктивных операторов. Matlab содержит соответствующие функции.

В рамках данной интерпретации конъюнктивный оператор представляет собой оператор вычисления алгебраического произведения – **prod**, а дизъюнктивный оператор – оператор вычисления алгебраической суммы – **probor**:

y = prod([y2; y2])

y = probor ([y2; y2])

Параметры y1 и y2 представляют собой исходные ФП.

Дополнение нечеткого множества есть не что иное, как математическое представление вербального выражения «НЕ А», где А — нечеткое множество, описывающее некоторое размытое суждение. Описание функции дополнения:

y = 1 - y*, где y* – исходная ФП.

Общая постановка задачи

Дано 3 нечетких множества в форме функций принадлежности. Необходимо построить функцию принадлежности заданного нечеткого множества D согласно Вашего варианта и определить степень принадлежности одного элемента множеству D, используя один из подходов. При этом необходимо рассчитать и построить графики:

1) $A \cap B;\quad A \cap C;\quad C \cap B;\quad A \cap C \cap B;$

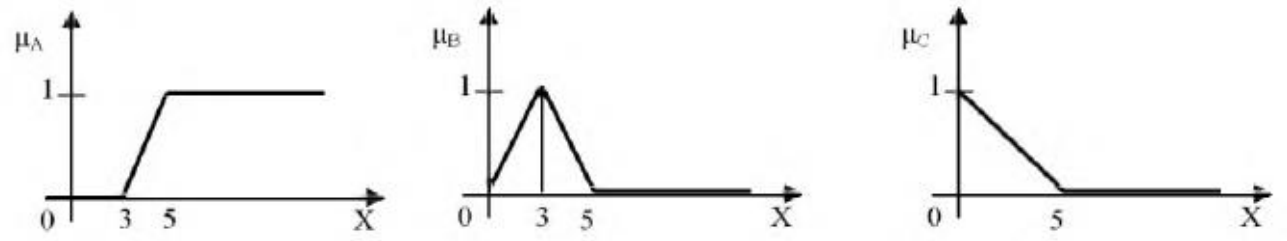
2) $A \cup B;\quad A \cup C;\quad C \cup B;\quad A \cup C \cup B;$

3) $A \setminus B;\quad A \setminus C;\quad C \setminus B;\quad C \setminus A;\quad B \setminus A;\quad B \setminus C;$

4) $\overline{A};\quad \overline{C};\quad \overline{B};\quad \overline{D};$

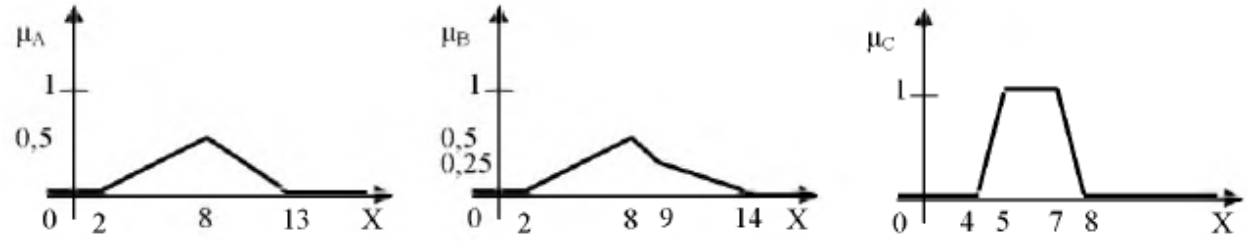
Список индивидуальных данных

1. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=gaussmf; C=zmf.

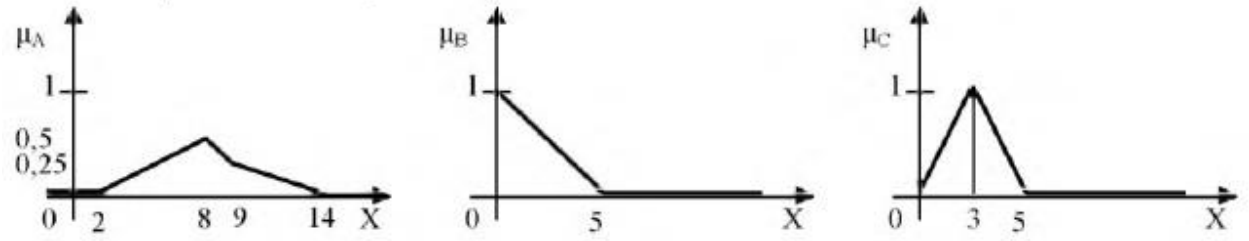


2. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного

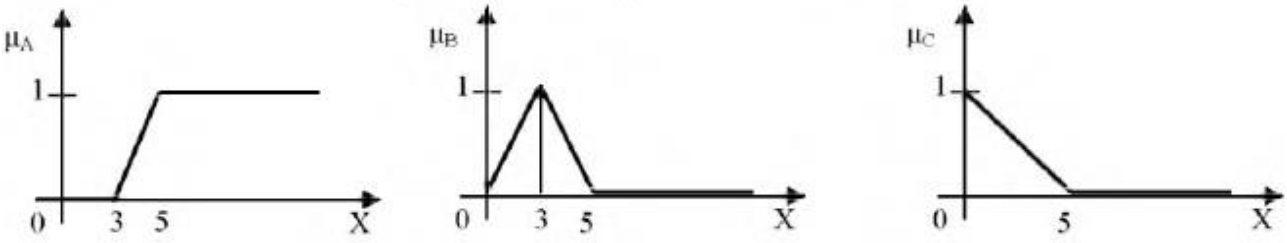
элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B= «создать две функции и посредством логических операций получить исходное множество»; C=gbellmf.



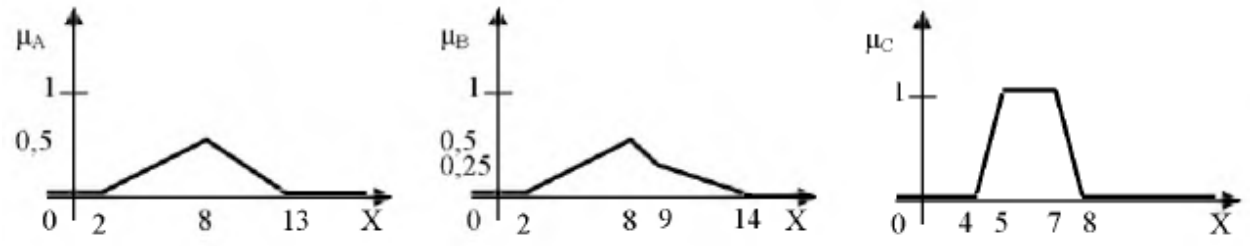
3. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=zmf; C=gaussmf.



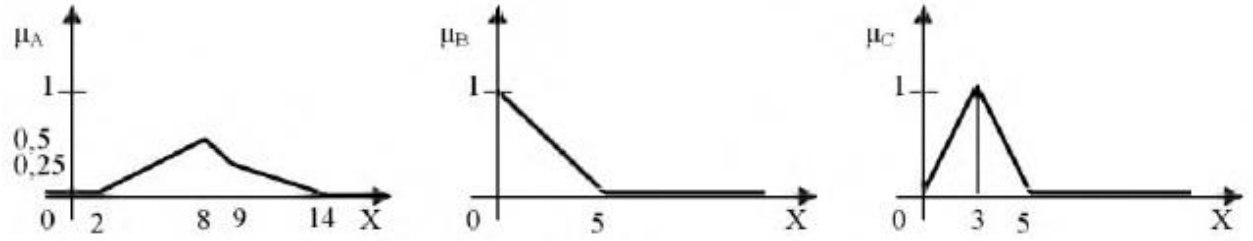
4. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=smf; B=«создать две функции и посредством логических операций получить исходное множество»; C=zmf.



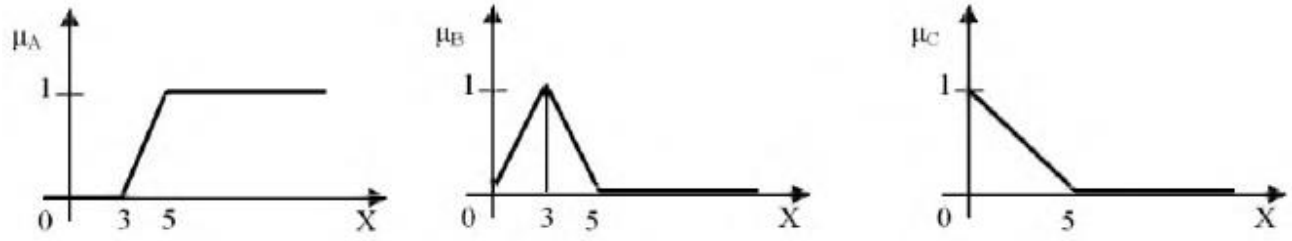
5. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=«создать две функции и посредством логических операций получить исходное множество»; C=psigmf.



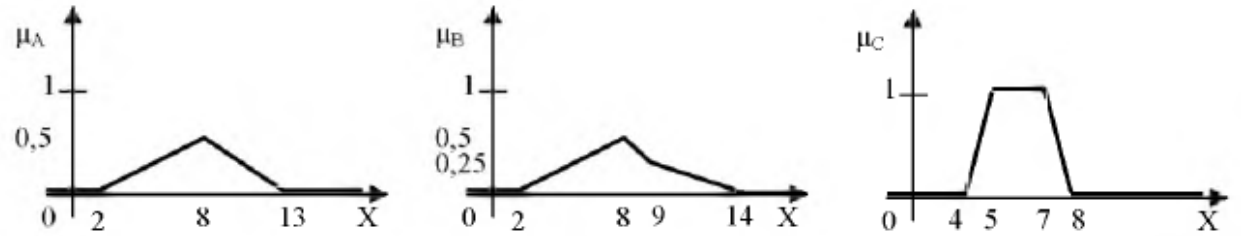
6. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=zmf; C=trimf.



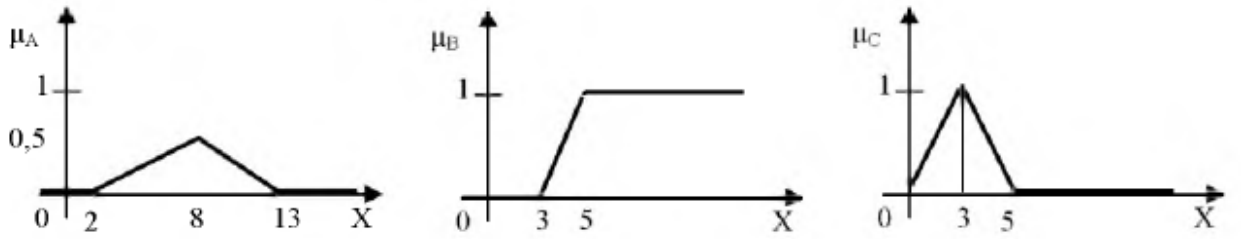
7. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=trimf; C=zmf.



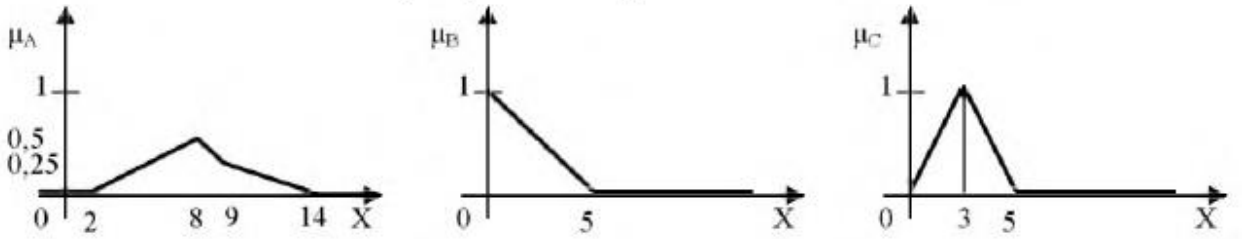
8. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=«создать две функции и посредством логических операций получить исходное множество»; C=trapmf.



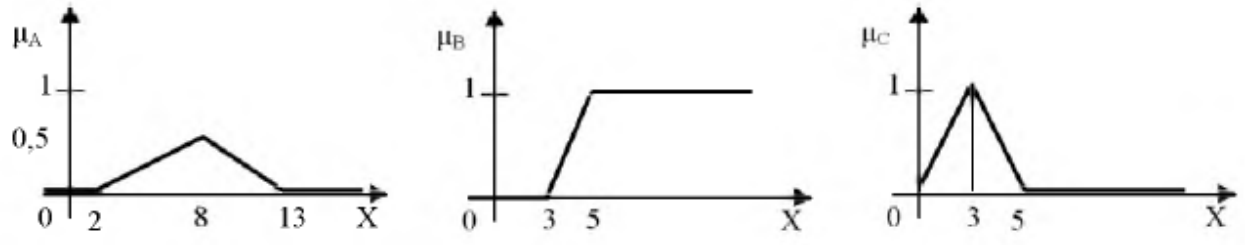
9. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \overline{B} \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=smf; C=«создать две функции и посредством логических операций получить исходное множество».



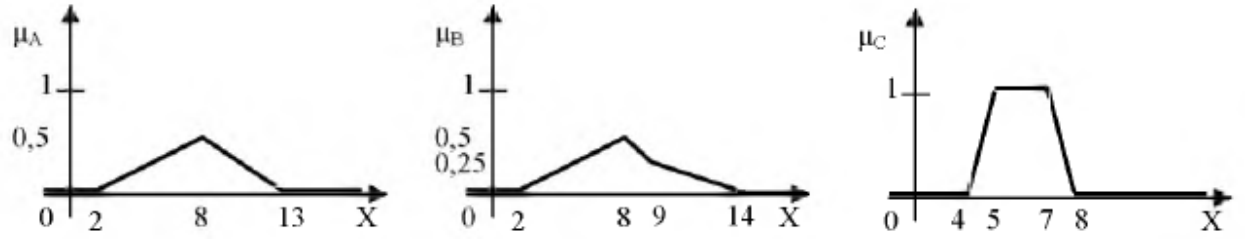
10. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=zmf; C=trimf.



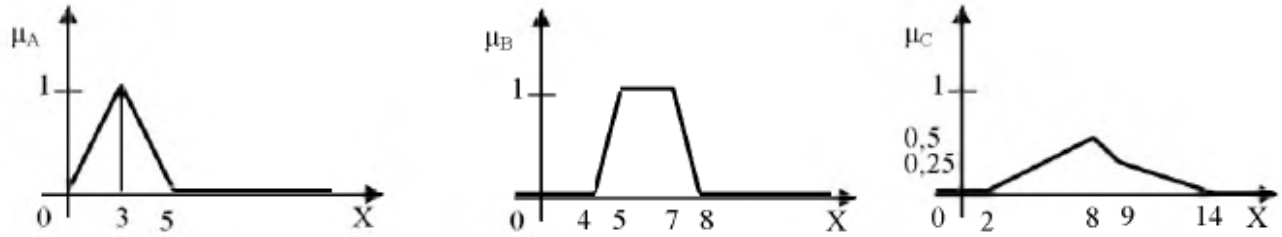
11. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=smf; C=«создать две функции и посредством логических операций получить исходное множество».



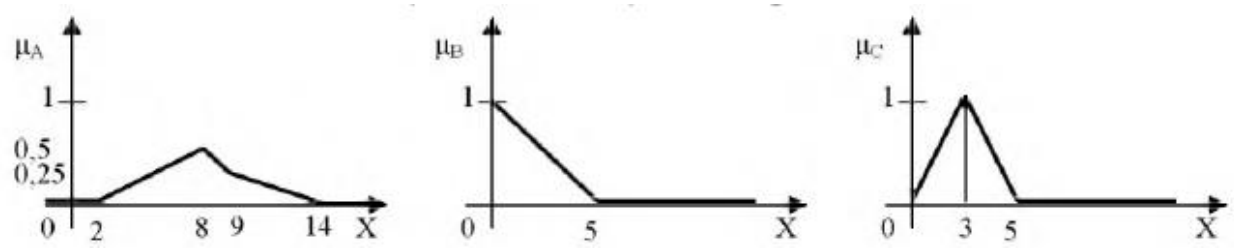
12. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=«создать две функции и посредством логических операций получить исходное множество»; C=dsigmf.



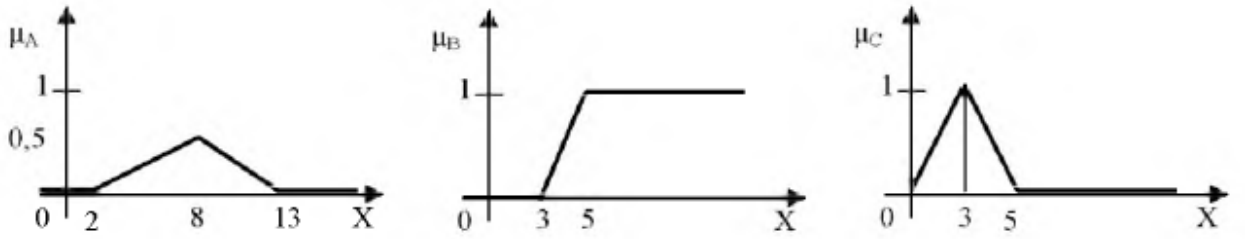
13. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = (\overline{A} \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=trapmf; C=«создать две функции и посредством логических операций получить исходное множество».



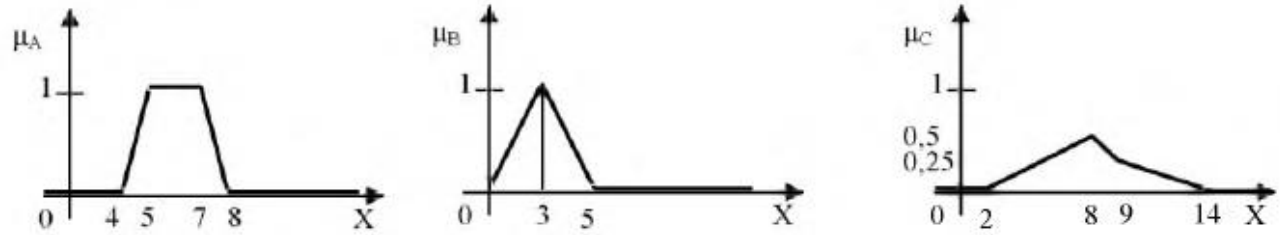
14. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=zmf; C=«создать две функции и посредством логических операций получить исходное множество».



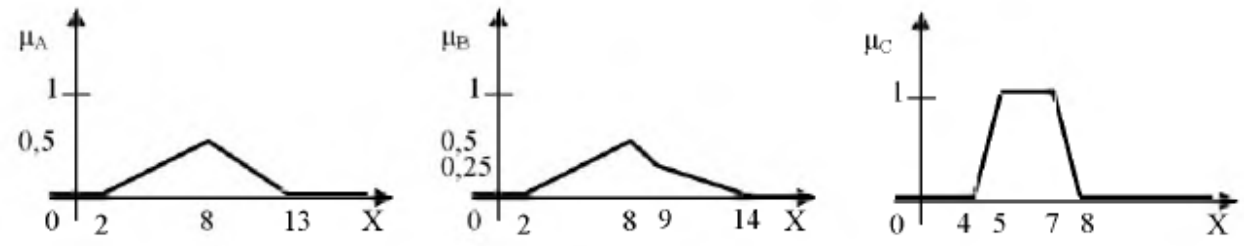
15. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=sigmf; C=trimf.



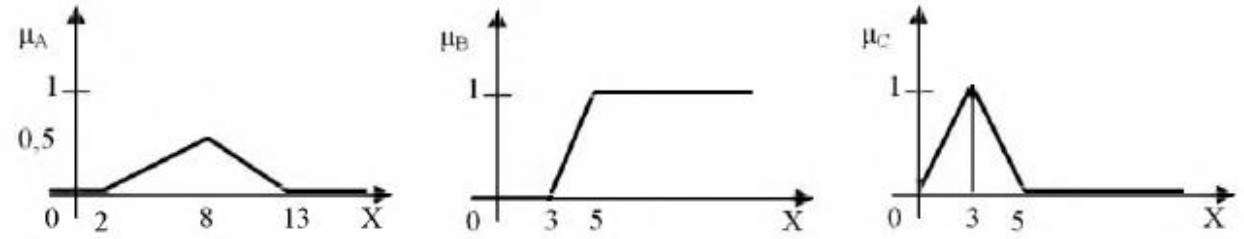
16. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trapmf; B=trimf; C=«создать две функции и посредством логических операций получить исходное множество».



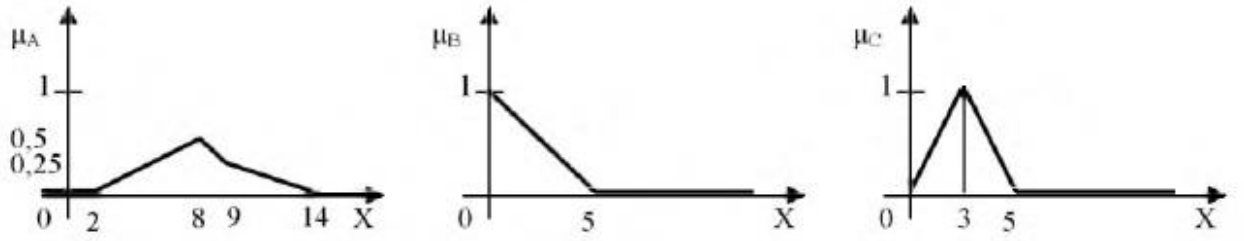
17. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=«создать две функции и посредством логических операций получить исходное множество»; C= gauss2mf.



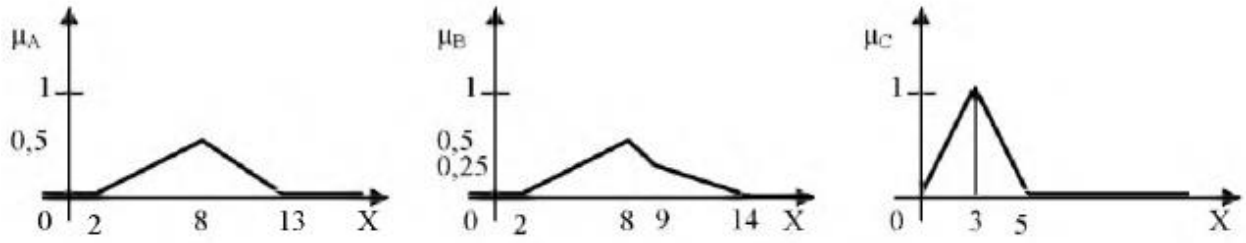
18. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=sigmf; C=trimf.



19. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup \overline{B} \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=«создать две функции и посредством логических операций получить исходное множество»; B=zmf; C=trimf.



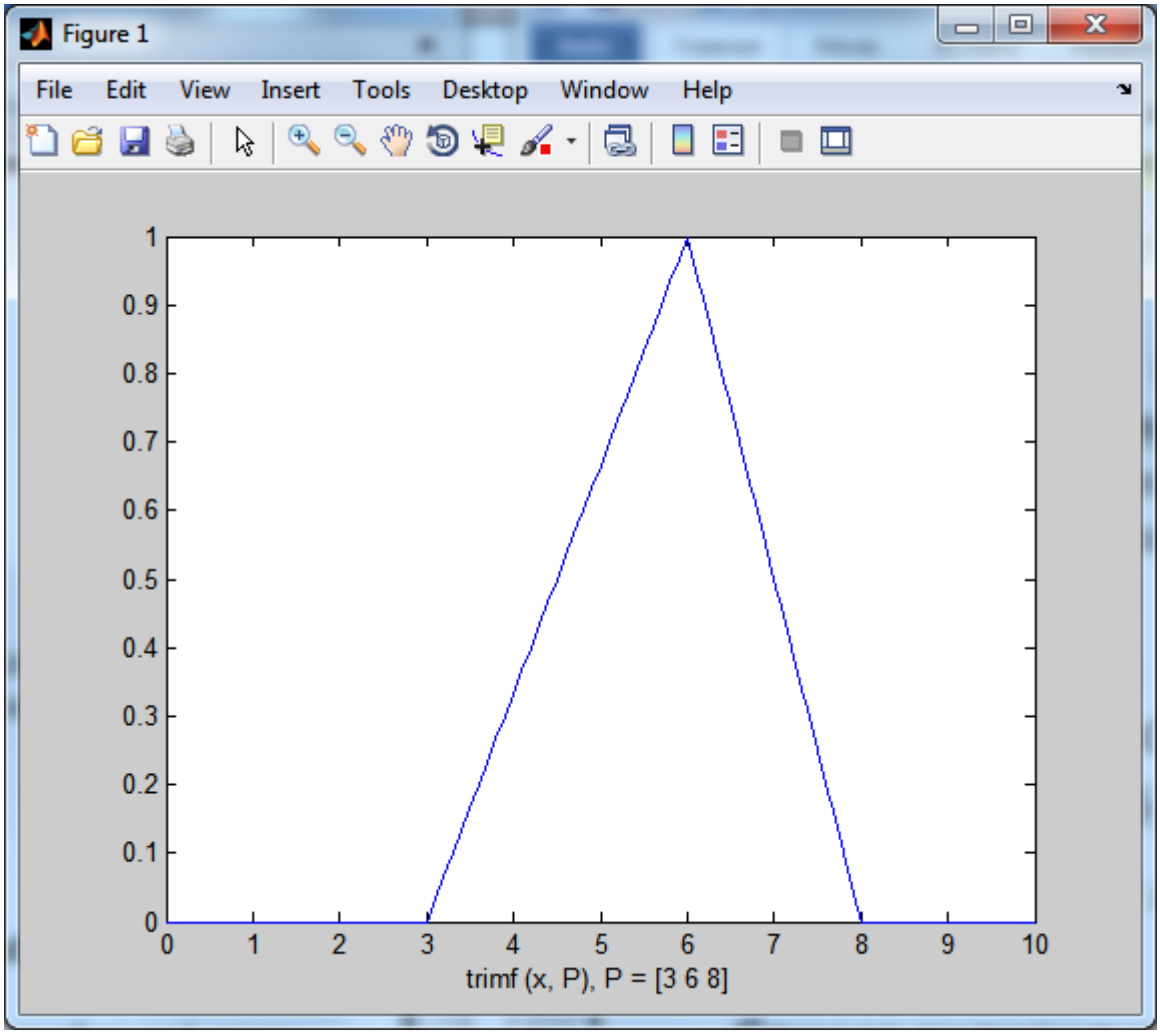
20. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cup \overline{B} \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений. При этом функции принадлежности нечетких множеств A, B, C задать при помощи встроенных функций fuzzy logic: A=trimf; B=«создать две функции и посредством логических операций получить исходное множество»; C=trimf.



Пример выполнения работы

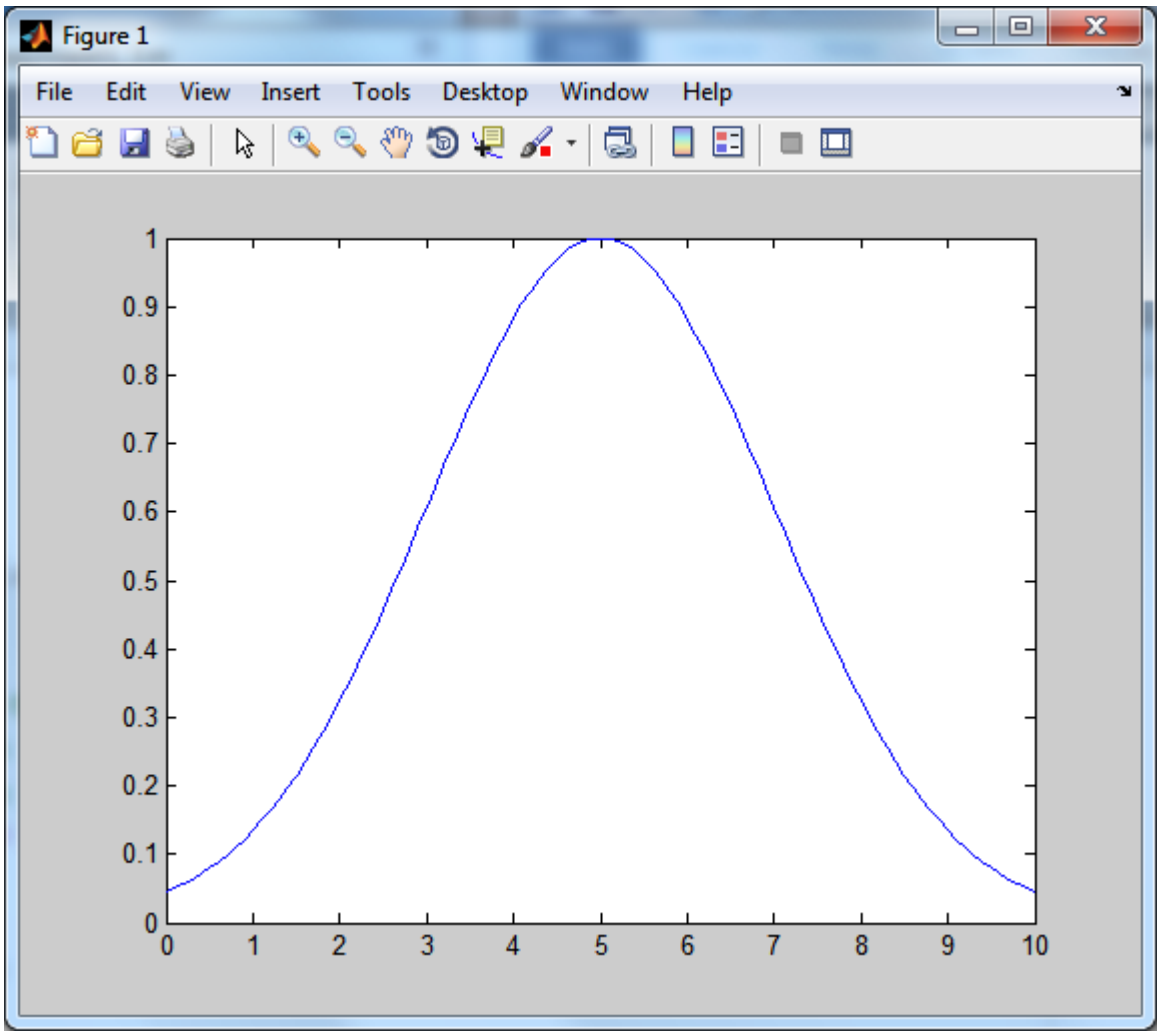
Программа использования ФП trimf:

```
x=0:0.1:10; % Задается базовое множество
y =trimf (x,[3 6 8]); % Определяется треугольная ФП
plot (x,y); % Выводится график функции
xlabel('trimf (x, P), P = [3 6 8]'); % Подписывается график
под осью абсцисс
```



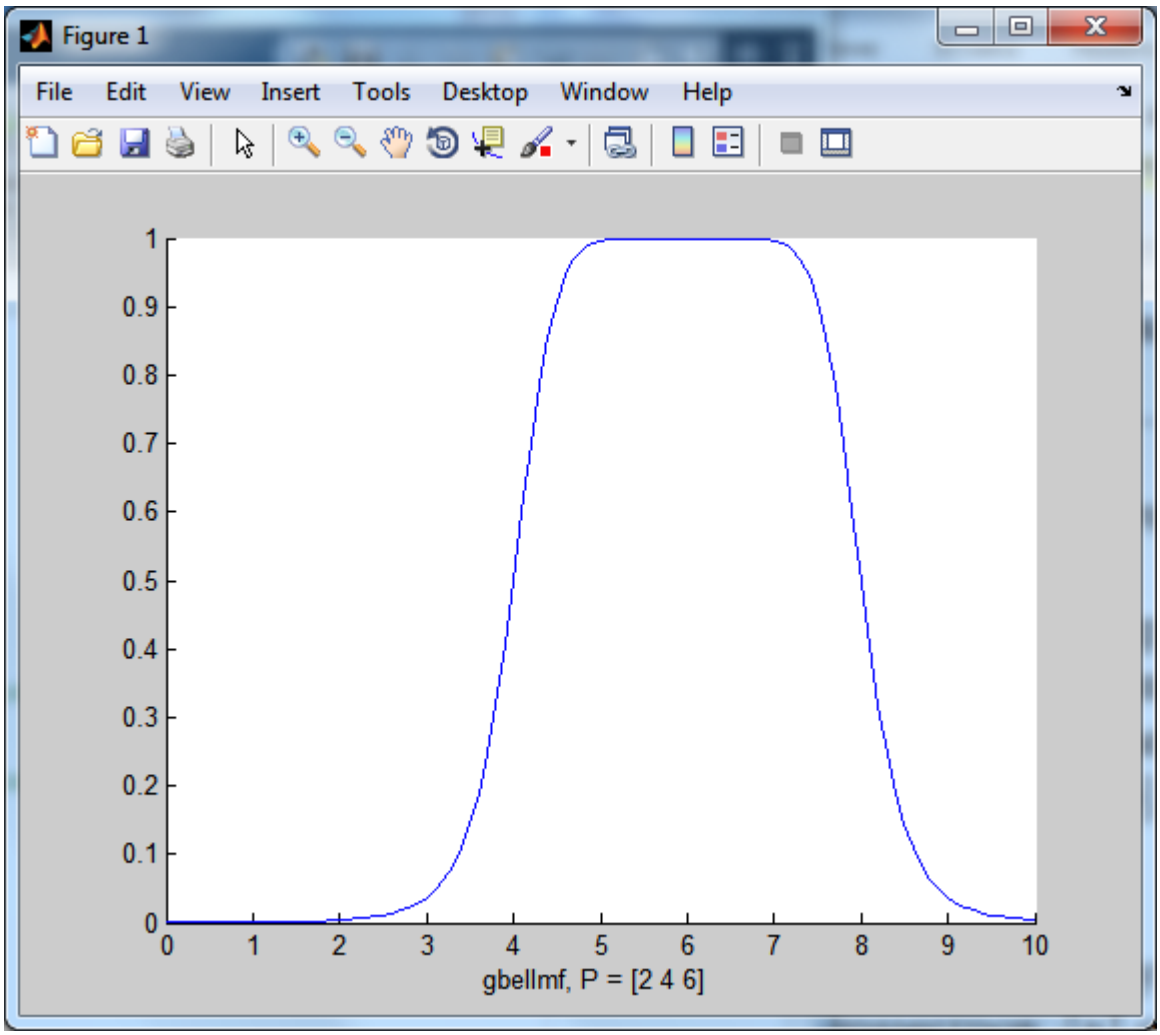
Программа использования ФП gaussmf:

```
x = 0 : 0.1 :10;
y = gaussmf (x, [2 5]);
plot (x, y);
```



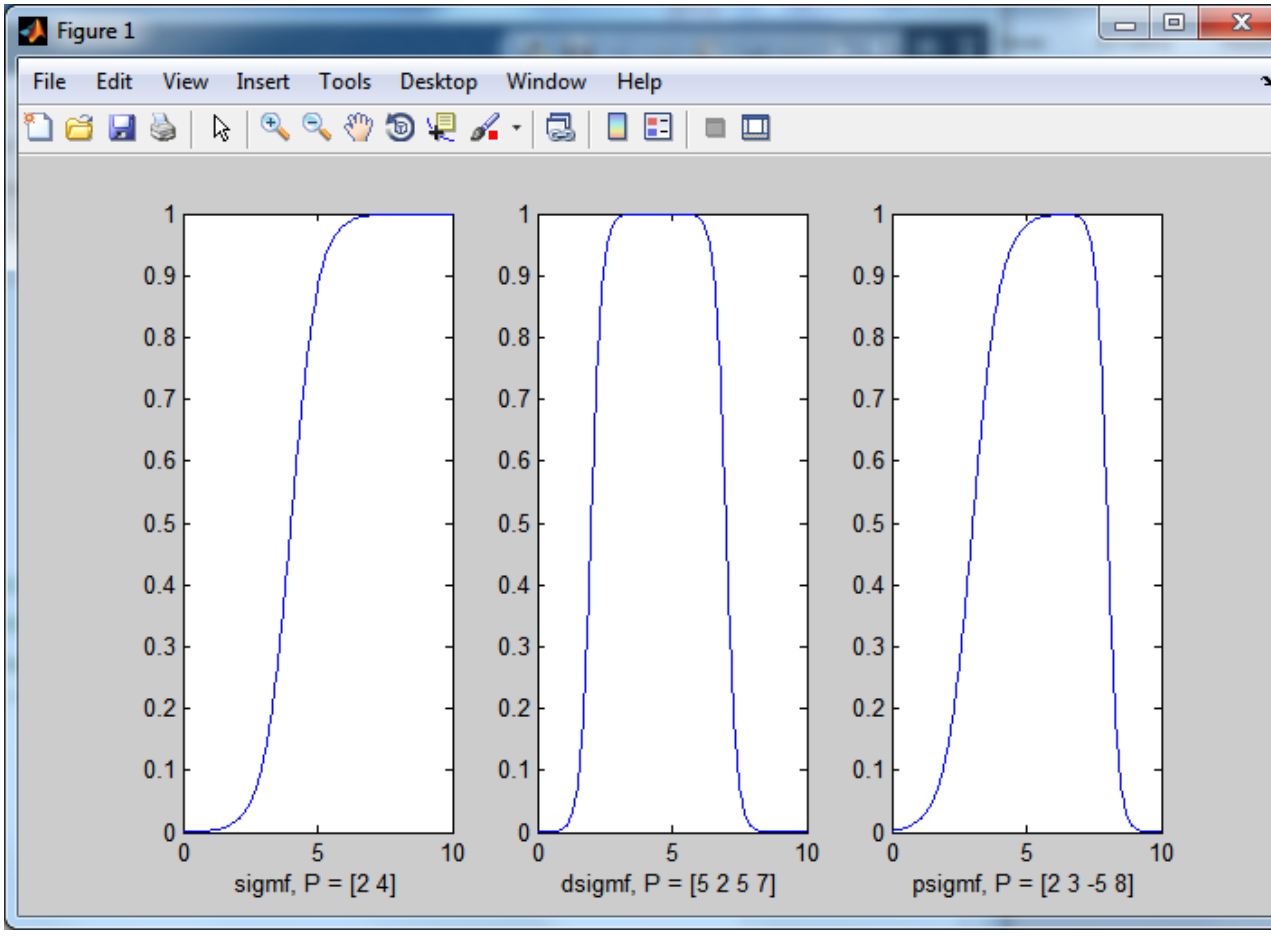
Программа использования ФП gbellmf:

```
x = 0:0.1:10;
y = gbellmf (x, [2 4 6]);
plot (x, y);
xlabel('gbellmf, P = [2 4 6] ')
```



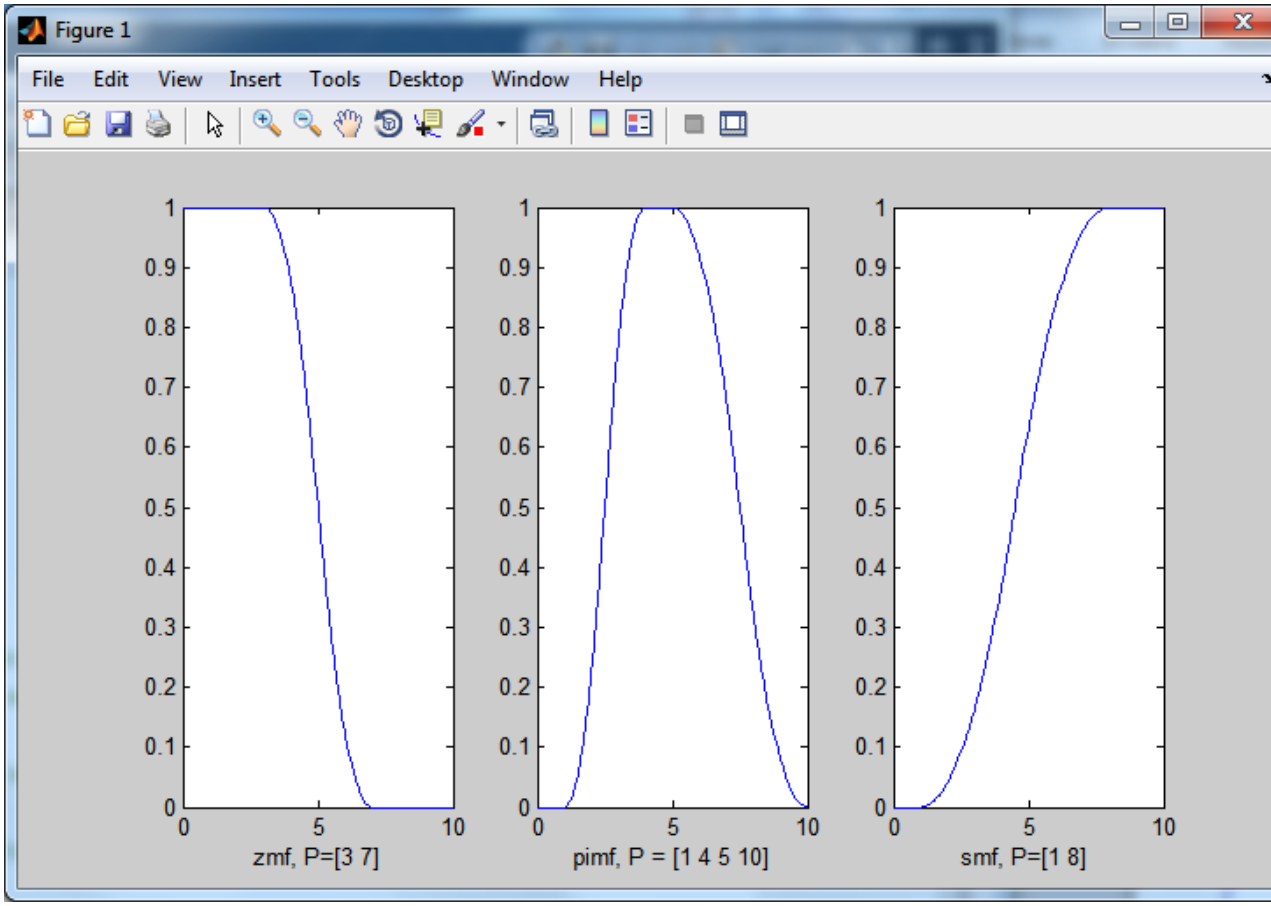
Программа использования сигмоидных функций:

```
x = 0:0.1:10; % определяется базовое множество
subplot (1,3,1) ; % формируется матрица графиков (3 x 1) — первый элемент — текущий
y=sigmf (x, [2 4]) ;
plot (x, y); % выводится график в первый элемент матрицы
xlabel ('sigmf, P = [2 4]');
subplot (1,3,2); % выбирается второй текущий элемент
y = dsigmf (x, [5 2 5 7]) ;
plot (x, y) ; % выводится график во второй элемент матрицы
xlabel ('dsigmf, P = [5 2 5 7]');
subplot (1,3,3); % выбирается третий текущий элемент
y = psigmf (x, [2 3 -5 8]);
plot (x, y); % выводится график в третий элемент матрицы
xlabel ('psigmf, P = [2 3 -5 8]');
```



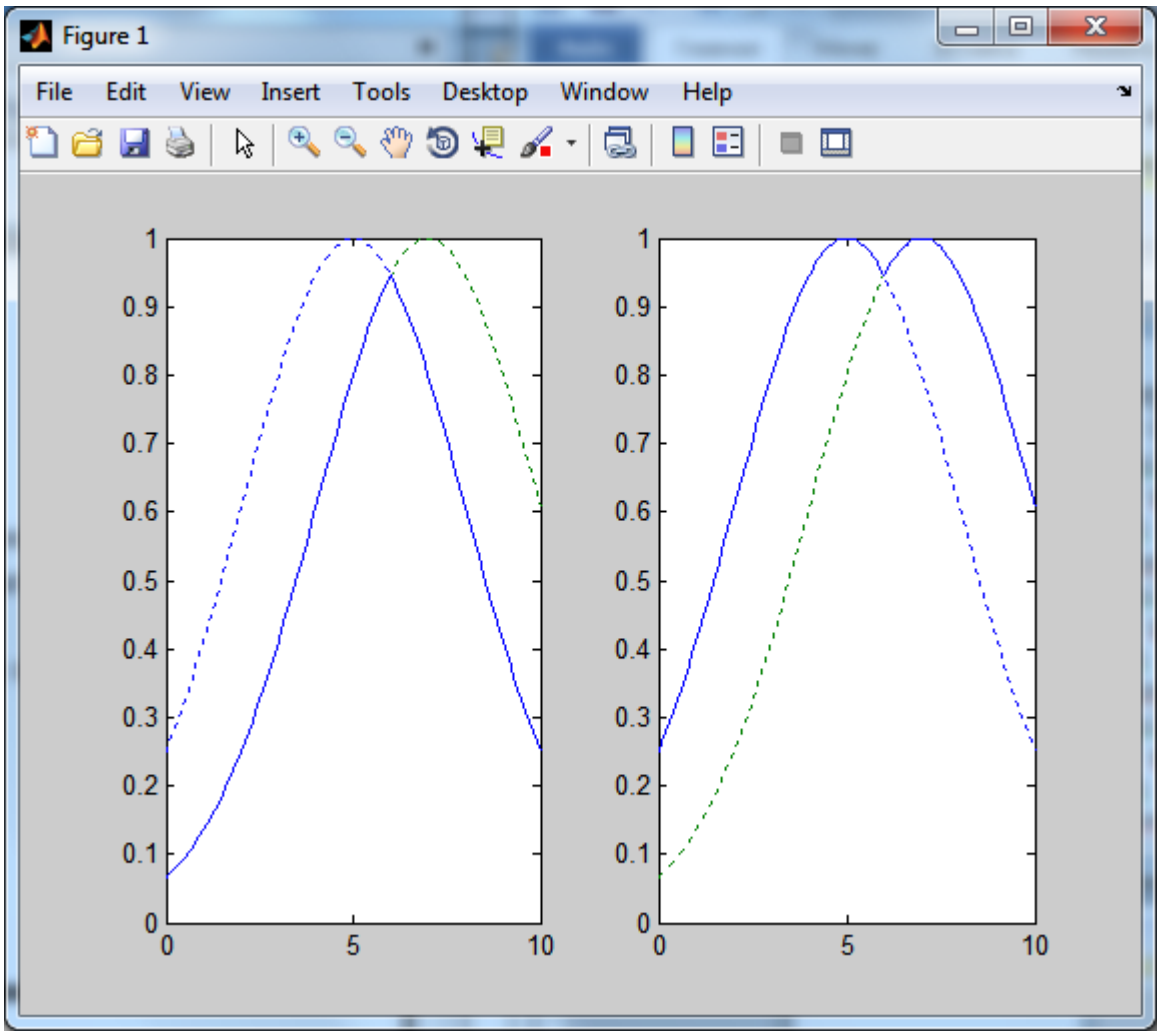
Программа использования полиномиальных кривых:

```
x = 0: 0.1:10;
subplot (1,3,1);
y = zmf (x, [3 7] ) ;
plot (x, y);
xlabel ('zmf, P=[3 7]');
subplot (1, 3, 2);
y = pimf (x,[1 4 5 10]) ;
plot (x, y);
xlabel ('pimf, P = [1 4 5 10]');
subplot (1, 3, 3);
y = smf (x, [1 8]) ;
plot (x, y);
xlabel ('smf, P=[1 8] ')
```



Программа использования максиминных операций min и max:

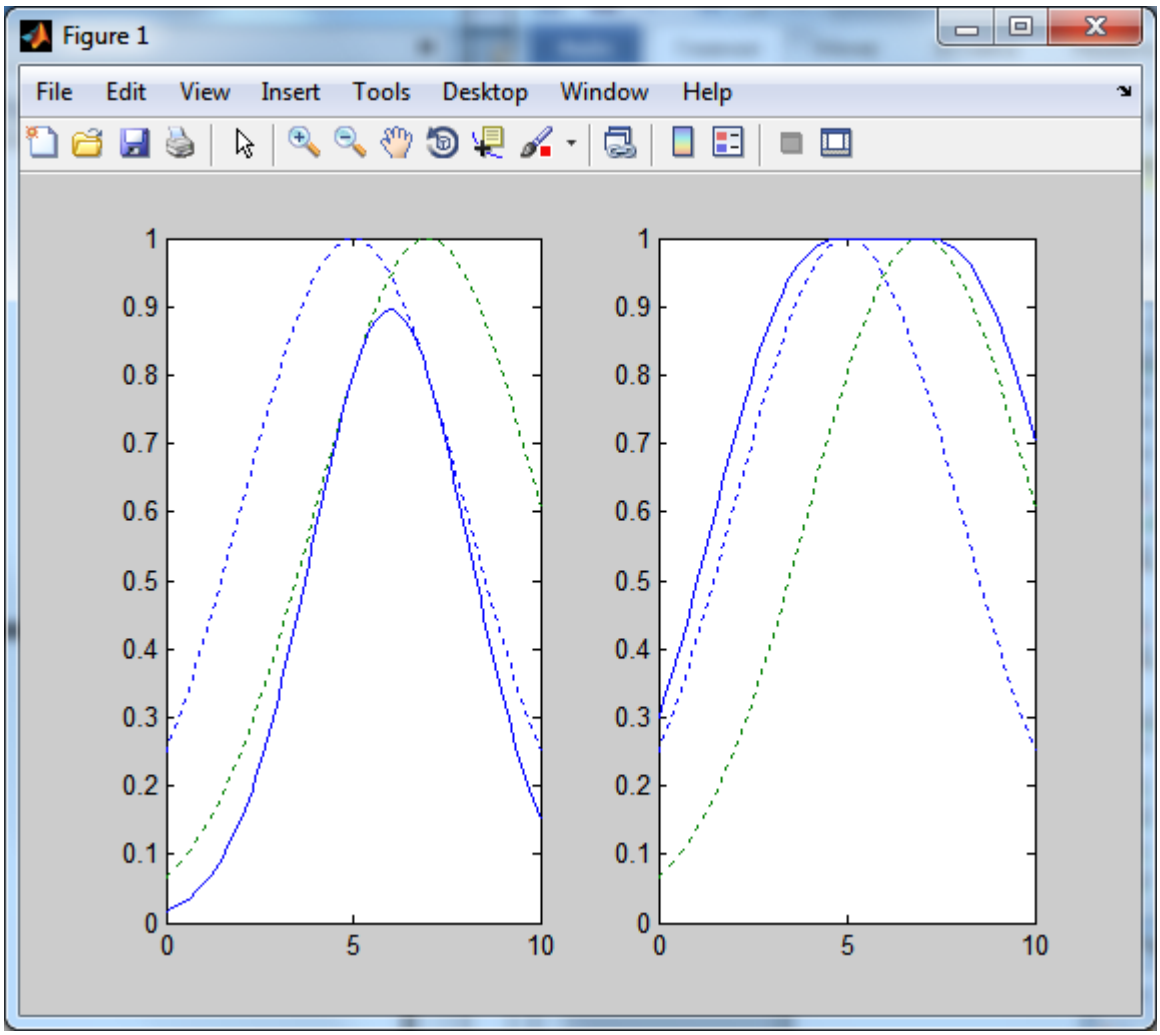
```
x = 0:0.1:10;
subplot(1, 2, 1);
y1 = gaussmf (x, [3 5] ) ;
y2 = gaussmf (x, [3,7]);
y3 = min ( [y1; y2]);
plot (x, [y1 ; y2] , ': ') ; % построение исходных ФП
пунктирной линией
hold on;% включение механизма добавления кривой в текущей
график
plot (x, y3);
hold off; % выключение механизма добавления кривой в текущей
график
subplot(1, 2, 2);
y4 = max ( [y1; y2]);
plot (x, [y1 ; y2] , ': ');
hold on;
plot (x, y4);
hold off;
```



Пунктирной линией на графиках изображены исходные ФП, а сплошной линией – результат действия логических операторов.

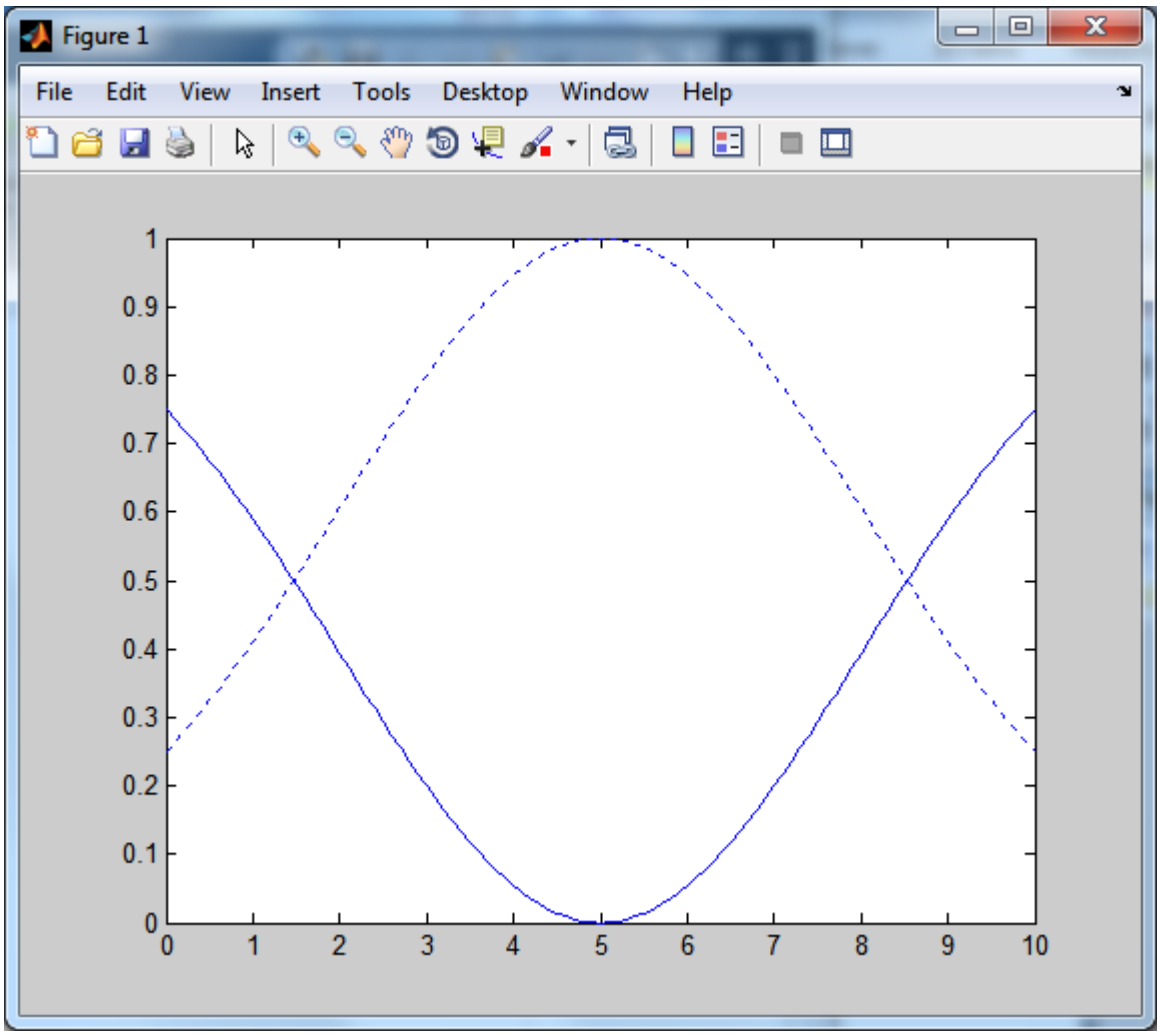
Программа использования вероятностных операторов конъюнкции и дизъюнкции:

```
x = 0:0.1:10;
subplot (1, 2, 1);
y1 = gaussmf (x, [3 5]);
y2 = gaussmf (x, [3 7]);
y3 = prod ([y1; y2]);
plot (x, [y1; y2] , ': ')
hold on;
plot (x, y3) ;
hold off;
subplot(1, 2, 2);
y4 = probor ( [y1; y2]);
plot (x, [y1; y2] , ':') ;
hold on;
plot (x, y4) ;
hold off
```



Программа использования операции дополнения:

```
x = 0 : 0.1 : 10;
y1 = gaussmf (x, [3 5]);
y = 1 - y1;
plot (x, y1, ': ');
hold on;
plot(x, y);
hold off;
```



Контрольные вопросы к защите

1. Назовите встроенные функции принадлежности пакета fuzzy logic?
2. Что такое функция принадлежности?
3. Какие конъюнктивные и дизъюнктивные операторы вы знаете?
4. Назовите операции с нечеткими множествами, встроенные в пакет fuzzy logic?

Лабораторная работа №4. Проектирование системы типа Мамдани средствами пакета Fuzzy Logic Toolbox на примере построения нечеткой аппроксимирующей системы

Цель работы: Проектирование систем нечеткого логического вывода типа Мамдани.

Теоретическая часть

FIS-редактор

FIS-редактор предназначен для создания, сохранения, загрузки и вывода на печать систем нечеткого логического вывода, а также для редактирования следующих свойств:

- тип системы;
- наименование системы;
- количество входных и выходных переменных;
- наименование входных и выходных переменных;
- параметры нечеткого логического вывода.

Загрузка FIS-редактора происходит с помощью команды **fuzzy.** В результате появляется интерактивное графическое окно, приведенное на рисунке 4.1.

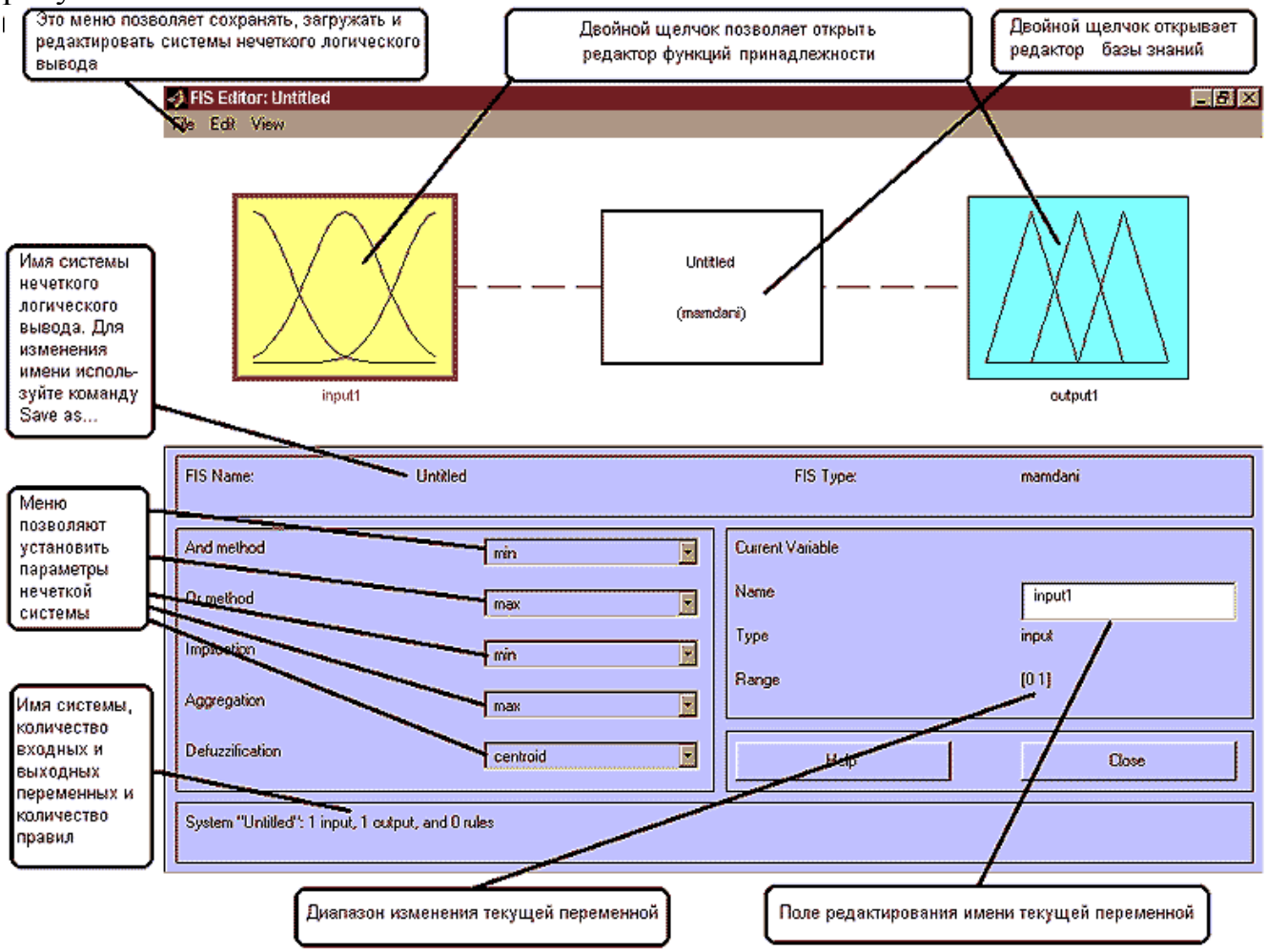


Рисунок 4.1 - Интерактивное графическое окно FIS-редактора

На этом же рисунке также указаны функциональные назначения основных полей графического окна. В нижней части графического окна FIS-редактора расположены кнопки **Help** и **Close**, которые позволяют вызвать окно справки и закрыть редактор, соответственно.

FIS-редактор содержит 8 меню:

1. Меню File. Это общее меню для всех GUI-модулей используемых с системами нечеткого логического вывода:

- С помощью команды **New FIS**… пользователь имеет возможность создать новую систему нечеткого логического вывода. При выборе этой команды появятся две альтернативы: **Mamdani** и **Sugeno**, которые определяют тип создаваемой системы. Создать систему типа **Mamdani** можно также нажатием **Ctrl+N.**

- С помощью команды **Import** пользователь имеет возможность загрузить ранее созданную систему нечеткого логического вывода. При выборе этой команды появятся две альтернативы **From Workspace**… и **From disk**, которые позволяют загрузить систему нечеткого логического вывода из рабочей области MatLab и с диска, соответственно. При выборе команды **From Workspace**… появится диалоговое окно, в котором необходимо указать идентификатор системы нечеткого логического вывода, находящейся в рабочей области MatLab. Файлы систем нечеткого логического вывода имеют расширение **.fis**. Загрузить систему нечеткого логического вывода с диска можно также нажатием **Ctrl+N** или командой **fuzzy FIS_name**, где **FIS_name** – имя файла системы нечеткого логического вывода.

- При выборе команды **Export** появятся две альтернативы **To Workspace… и To disk**, которые позволяют скопировать систему нечеткого логического вывода в рабочую область MatLab и на диск, соответственно.

2. Меню Edit:

- Команда **Undo** отменяет ранее совершенное действие. Команда **Add Variable…** позволяет добавить в систему нечеткого логического вывода еще одну переменную. При выборе этой команды появятся две альтернативы **Input** и **Output**, которые позволяют добавить входную и выходную переменную, соответственно.

- Команда **Remove Selected Variable** удаляет текущую переменную из системы. Признаком текущей переменной является красная окантовка ее прямоугольника. Назначение текущей переменной происходит с помощью однократного щелчка левой кнопки мыши по ее прямоугольнику.

- Команда **Membership Function…** открывает редактор функций принадлежностей.

- Команда **Rules…** открывает редактор базы знаний.

3. Меню View. Это общее меню для всех GUI-модулей, используемых с системами нечеткого логического вывода:

- Команда **Rules** открывает окно визуализации нечеткого логического вывода.

- Команда **Surface** открывает окно вывода поверхности «входы-выход», соответствующей системе нечеткого логического вывода.

4. Меню And Method. Это меню позволяет установить следующие реализации логической операции **И**:

- min – минимум;

- **prod** – умножение;

- команда **Custom…** возможность установить собственную реализацию **операции И**. Для этого необходимо в появившемся графическом окне напечатать имя функции, реализующей эту операцию.

5. Меню Or Method. Это меню позволяет установить следующие реализации логической операции **ИЛИ**:

- **max** – умножение;

- **probor** - вероятностное ИЛИ;

- команда **Custom…** возможность установить собственную реализацию **операции ИЛИ**.

6. Меню Implication. Это меню позволяет установить следующие реализации импликации:

- **min** – минимум;

- **prod** – умножение;

- команда **Custom…**

7. Меню Aggregation. Это меню позволяет установить следующие реализации операции объединения функций принадлежности выходной переменной:

- **max** – максимум;

- **sum** – сумма;

- **probor** - вероятностное ИЛИ;

- команда **Custom…**

8. Меню Defuzzification. Это меню позволяет выбрать метод дефаззификации. Для систем типа Мамдани запрограммированы следующие методы:

- **centroid** – центр тяжести;

- **bisector** –медиана;

- **lom** – наибольший из максимумов;

- **som** – наименьший из максимумов;

- **mom** – среднее из максимумов.

Для систем типа Сугэно запрограммированы следующие методы:

- **wtaver** – взвешенное среднее;

- **wtsum** – взвешенная сумма.

Пользователь также имеет возможность установить собственный метод деффазификации (**Custom…**).

Редактор функций принадлежности

Редактор функций принадлежности (**Membership Function Editor**) редактор предназначен для задания следующей информации о терм-множествах входных и выходных переменных:

- количество термов;

- наименования термов;

- тип и параметры функций принадлежности, которые необходимы для представления лингвистических термов в виде нечетких множеств.

Редактор функций принадлежности может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой **Membership Functions…** меню **Edit** или нажатием клавиш **Ctrl+2.** В FIS-редакторе открыть редактор функций принадлежности можно также двойным щелчком левой кнопкой мыши по полю входной или выходной переменных. Общий вид редактора функций принадлежности с указанием функционального назначения основных полей графического окна приведен на рисунке 4.2. В нижней части графического окна расположены кнопки **Help** и **Close**, которые позволяют вызвать окно справки и закрыть редактор, соответственно.

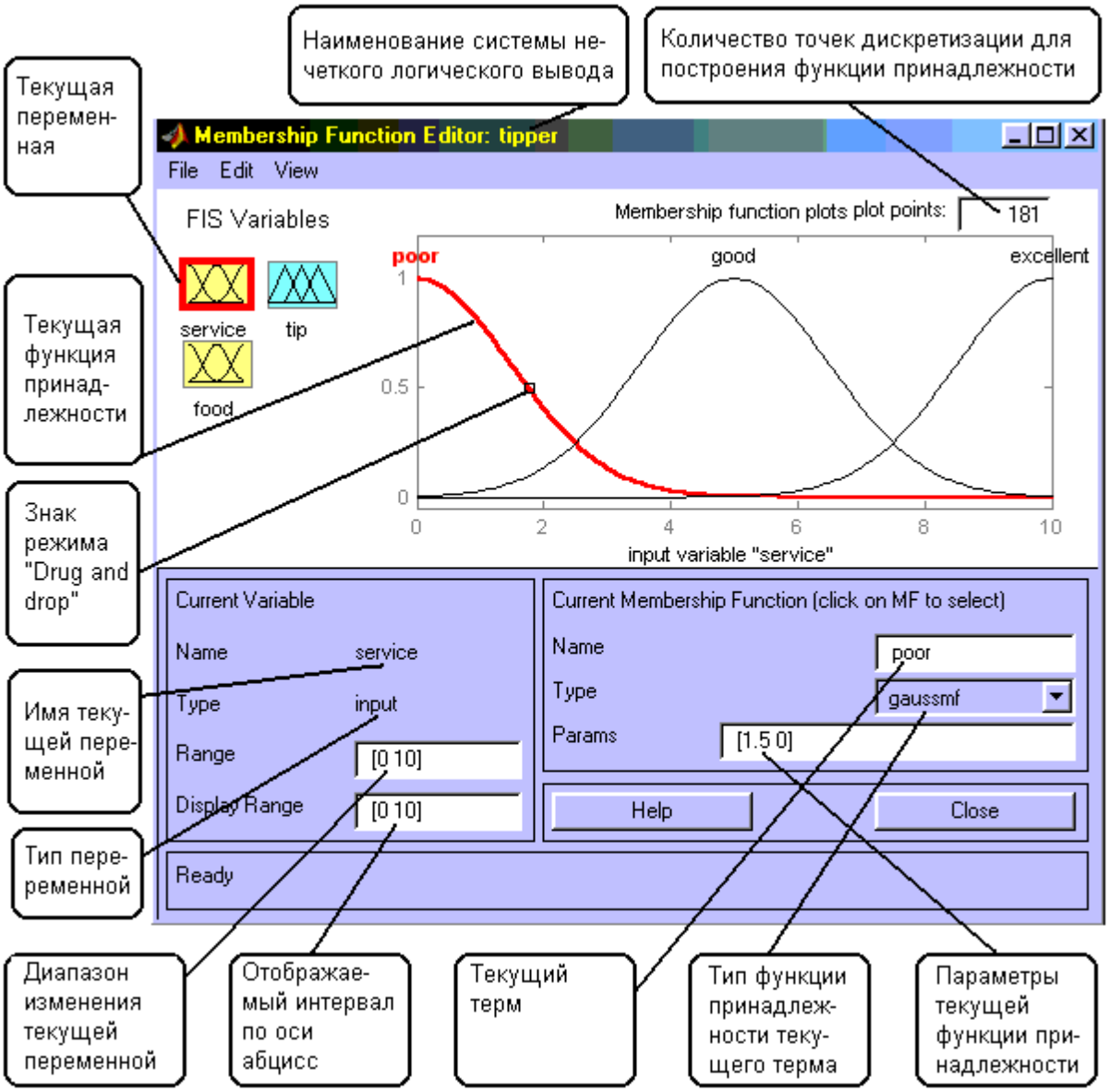


Рисунок 4.2 - Редактор функций принадлежности

Редактор функций принадлежности содержит 4 меню:

1. **Меню File** и **2. View** одинаковые для всех GUI-модулей используемых с системами нечеткого логического вывода.

3. Меню Edit:

- Команда **Undo** отменяет ранее совершенное действие.

- Команда **Add MFs…** позволяет добавить термы в терм-множество, используемое для лингвистической оценки текущей переменной. При выборе этой команды появится диалоговое окно (рисунок 4.3), в котором необходимо выбрать тип функции принадлежности и количество термов. Значения параметров функций принадлежности будут установлены автоматически таким образом, чтобы равномерно покрыть область определения переменной, заданной в окне **Range**. При изменении области определения в окне **Range** параметры функций принадлежности будут промасштабированы.

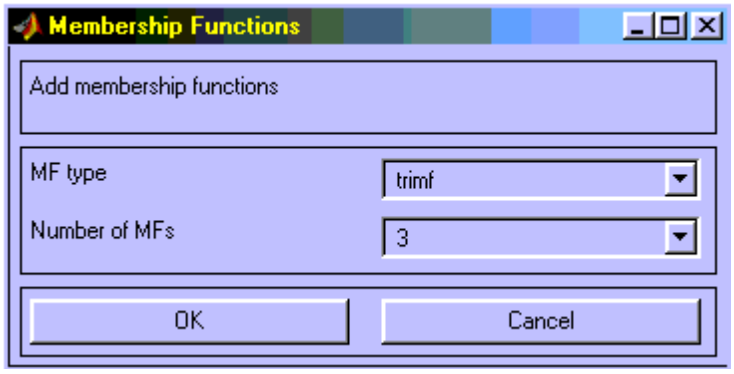


Рисунок 4.3 – Диалоговое окно выбора количества термов и типа функций принадлежности

- Команда **Add Custom MF…** позволяет добавить один лингвистический терм, функция принадлежности которого отличается от встроенных. После выбора этой команды появится графическое окно (рисунок 4.4), в котором необходимо напечатать лингвистически терм (поле **MF name**),, имя функции принадлежности (поле **M-File function name**) и параметры функции принадлежности (поле **Parameter list**).

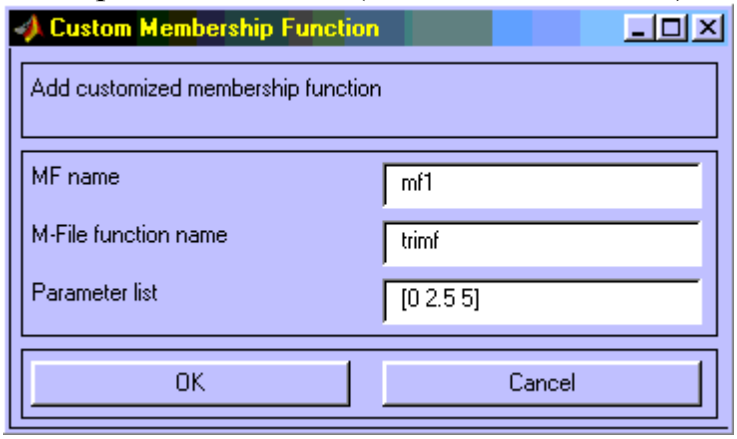


Рисунок 4.4 – Диалоговое окно задания лингвистического терма с невстроенной функцией принадлежности

- Команда **Remove Selected MF** удаляет текущий терм из терм-множества текущей переменной. Признаком текущей переменной является красная окантовка ее прямоугольника. Признаком текущего терма является красный цвет его функции принадлежности.

- Команда **Remove All MFs** удаляет все термы из терм-множества текущей переменной.

- Команда **FIS Properties…** открывает FIS-редактор.

- Команда **Rules…** открывает редактор базы знаний.

4. Меню Type. Это меню позволяет установить тип функций принадлежности термов, используемых для лингвистической оценки текущей переменной. На рисунке 4.5 приведено меню Type, в котором указаны возможные типы функций принадлежности.

Рисунок 4.5 - Меню Type

Редактор функций принадлежности содержит четыре окна ввода информации: **Range, Display Range, Name** и **Params**. Эти четыре окно предназначены для задания диапазона изменения текущей переменной, диапазона вывода функций принадлежности, наименования текущего лингвистического терма и параметров его функции принадлежности, соответственно. *Параметры функции принадлежности можно подбирать и в графическом режиме, путем изменения формы функции принадлежности с помощью технологии "Drug and drop". Для этого необходимо позиционировать курсор мыши на знаке режима "Drug and drop" (см. рис. 7.5), нажать на левую кнопку мыши и не отпуская ее изменять форму функции принадлежности. Параметры функции принадлежности будут пересчитываться автоматически.*

Редактор базы знаний

Редактор базы знаний (**Rule Editor**) предназначен для формирования и модификации нечетких правил. Редактор базы знаний может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой **Rules…** меню **Edit** или нажатием клавиш Ctrl+3. В FIS-редакторе открыть редактор базы знаний можно также двойным щелчком левой кнопкой мыши по прямоугольнику с названием системы нечеткого логического вывода, расположенного в центре графического окна.

Общий вид редактора базы знаний с указанием функционального назначения основных полей графического окна приведен на рисунке 4.6. В нижней части графического окна расположены кнопки **Help** и **Close**.

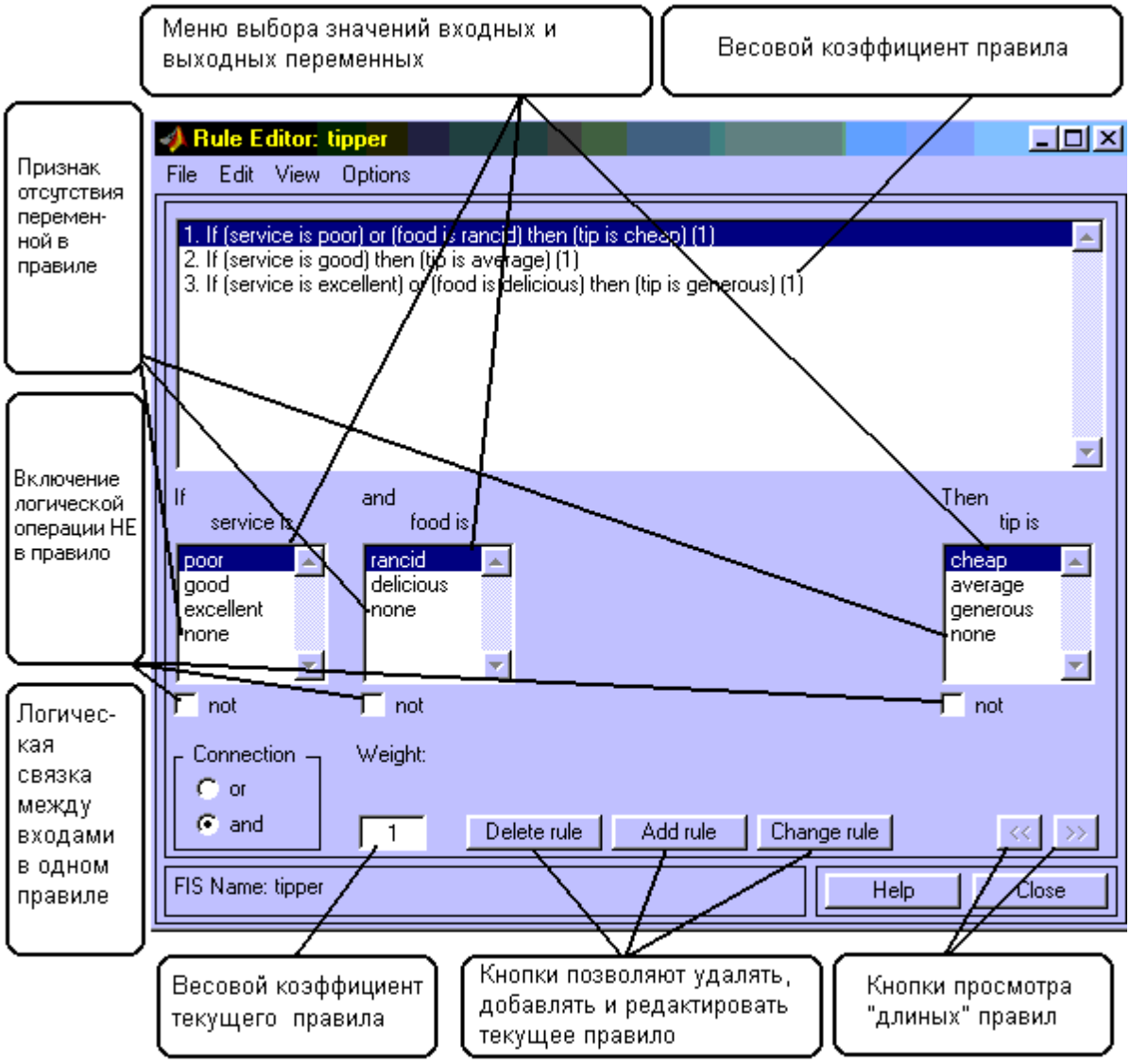


Рисунок 4.6 - Редактор базы знаний

Редактор функций принадлежности содержит четыре системных меню **File, Edit, View, Options**, меню выбора термов входных и выходных переменных, поля установки логических операций **И, ИЛИ, НЕ** и весов правил, а также кнопки редактирования и просмотра правил.

Для ввода нового правила в базу знаний необходимо с помощью мыши выбрать соответствующую комбинацию лингвистических термов входных и выходных переменных, установить тип логической связки (**И** или **ИЛИ**) между переменными внутри правила, установить наличие или отсутствие логической операции **НЕ** для каждой лингвистической

переменной, ввести значение весового коэффициента правила и нажать кнопку **Add Rule**. По умолчанию установлены следующие параметры:

- логическая связка переменных внутри правила – **И;**
- логическая операция **НЕ** – отсутствует;
- значение весового коэффициента правила – **1.**

Возможны случаи, когда истинность правила не изменяется при произвольном значении некоторой входной переменной, т.е. это переменная не влияет на результат нечеткого логического вывода в данной области факторного пространства. Тогда в качестве лингвистического значения этой переменной необходимо установить **none.**

Для удаления правила из базы знаний необходимо сделать однократный щелчок левой кнопкой мыши по этому правилу и нажать кнопку **Delete Rule**.

Для модификации правила необходимо сделать однократный щелчок левой кнопкой мыши по этому правилу, затем установить необходимые параметры правила и нажать кнопку **Edit Rule**.

Меню **File** и **View** одинаковые для всех GUI-модулей используемых с системами нечеткого логического вывода.

Меню Edit:

- Команда **Undo** отменяет ранее совершенное действие.
- Команда **FIS Properties…** открывает FIS-редактор.
- Команда **Membership Function…** открывает редактор функций принадлежностей.

Меню Options. Это меню позволяет установить язык и формат правил базы знаний. При выборе команды **Language** появится список языков English (Английский), Deutsch (Немецкий), Francais (Французский), из которого необходимо выбрать один. При выборе команды **Format** появится список возможных форматов правил базы знаний: **Verbose** - лингвистический; **Symbolic** – логический; **Indexed** – индексированный.

Различные форматы баз знаний демо-системы нечеткого логического вывода **Tipper** приведены на рисунке 4.6 (формат правил Verbose), рисунке 4.7 (формат правил Symbolic) и рисунок 4.8 (формат правил Indexed).

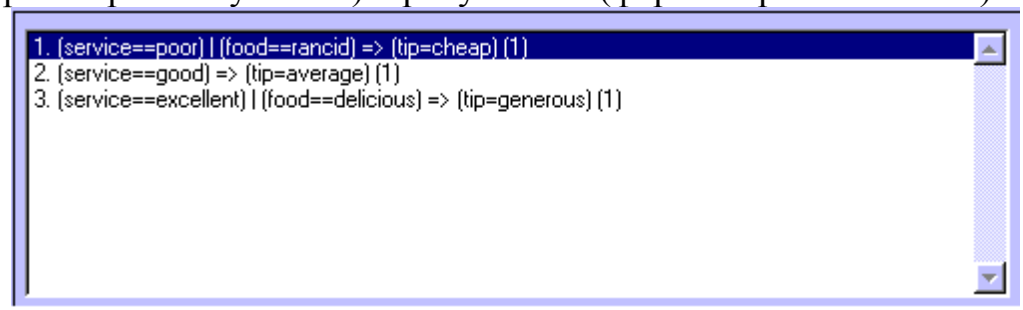


Рисунок 4.7 - База знаний в формате Symbolic

Рисунок 4.8 - База знаний в формате Indexed

Визуализация нечеткого логического вывода

Визуализация нечеткого логического вывода осуществляется с помощью GUI-модуля **Rule Viewer**. Этот модуль позволяет проиллюстрировать ход логического вывода по каждому правилу, получение результирующего нечеткого множества и выполнение процедуры дефаззификации. **Rule Viewer** может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой **View rules**… меню **View**. Вид **Rule Viewer** для системы логического вывода **tipper** с указанием функционального назначения основных полей графического окна приведен на рисунке 4.9.

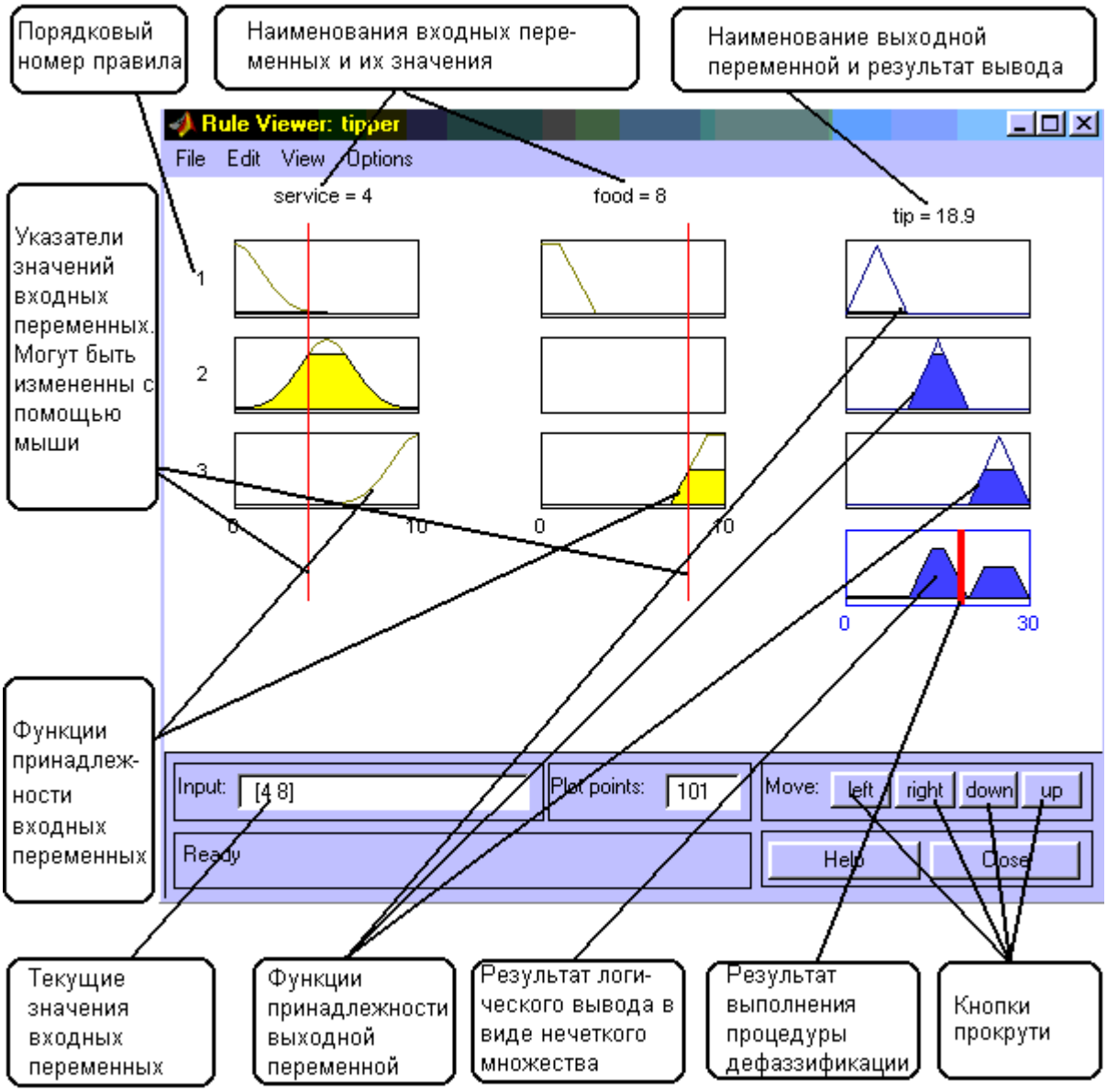


Рисунок 4.9 - Визуализация логического вывода для системы tipper с помощью Rule Viewer

Rule Viewer содержит четыре меню - **File, Edit, View, Options**, два поля ввода информации – **Input** и **Plot points** и кнопки прокрутки изображения влево-вправо (**left-right**), вверх-вниз (**up-down**). В нижней части графического окна расположены также кнопки **Help** и **Close.**

Каждое правило базы знаний представляется в виде последовательности горизонтально расположенных прямоугольников. При этом первые два прямоугольника (рисунок 4.9) отображают функции принадлежностей термов посылки правила (**ЕСЛИ**-часть правила), а последний третий прямоугольник соответствует функции принадлежности терма-следствия выходной переменной (**ТО**-часть правила). Пустой прямоугольник в визуализации второго правила означает, что в этом правиле посылка по переменной **food** отсутствует (**food is none**). Желтая заливка

графиков функций принадлежностей входных переменных указывает насколько значения входов, соответствуют термам данного правила. Для вывода правила в формате **Rule Editor** необходимо сделать однократный щелчок левой кнопки мыши по номеру соответствующего правила. В этом случае указанное правило будет выведено в нижней части графического окна.

Голубая заливка графика функции принадлежности выходной переменной представляет собой результат логического вывода в виде нечеткого множества по данному правилу. Результирующее нечеткое множество, соответствующее логическому выводу по всем правилам показано в нижнем прямоугольнике последнего столбца графического окна. В этом же прямоугольнике красная вертикальная линия соответствует четкому значению логического вывода, полученного в результате дефаззификации.

Ввод значений входных переменных может осуществляться двумя способами:

- путем ввода численных значений в поле **Input;**

- с помощью мыши, путем перемещения линий-указателей красного цвета. Для этого необходимо позиционировать курсор мыши на красной вертикальной линии, нажать на левую кнопку мыши и не отпуская ее переместить указатель на нужную позицию. Новое численное значения соответствующей входной переменной будет пересчитано автоматически и выведено в окно **Input.**

В поле **Plot points** задается количество точек дискретизации для построения графиков функций принадлежности. Значение по умолчанию – 101.

Меню Edit:

- Команда **FIS Properties…** открывает FIS-редактор.

- Команда **Membership Functions…** открывает редактор функций принадлежностей.

- Команда **Rules…** открывает редактор базы знаний.

Меню Options содержит только одну команду **Format**, которая позволяет установить один из следующих форматов вывода выбранного правило в нижней части графического окна:

- **Verbose** - лингвистический;

- **Symbolic** – логический;

- **Indexed** – индексированный.

Визуализация поверхности «входы-выход»

Визуализация поверхности «входы-выход» осуществляется с помощью GUI-модуля **Surface Viewer**. Этот модуль позволяет вывести графическое изображение зависимости значения любой выходной переменной от произвольных двух (или одной) входных переменных. **Surface**

Viewer может быть вызван из любого GUI-модуля, используемого с системами нечеткого логического вывода, командой **View surface …** меню **View**. Общий вид модуля **Surface Viewerc** указанием функционального назначения основных полей графического окна приведен на рисунке 4.10.

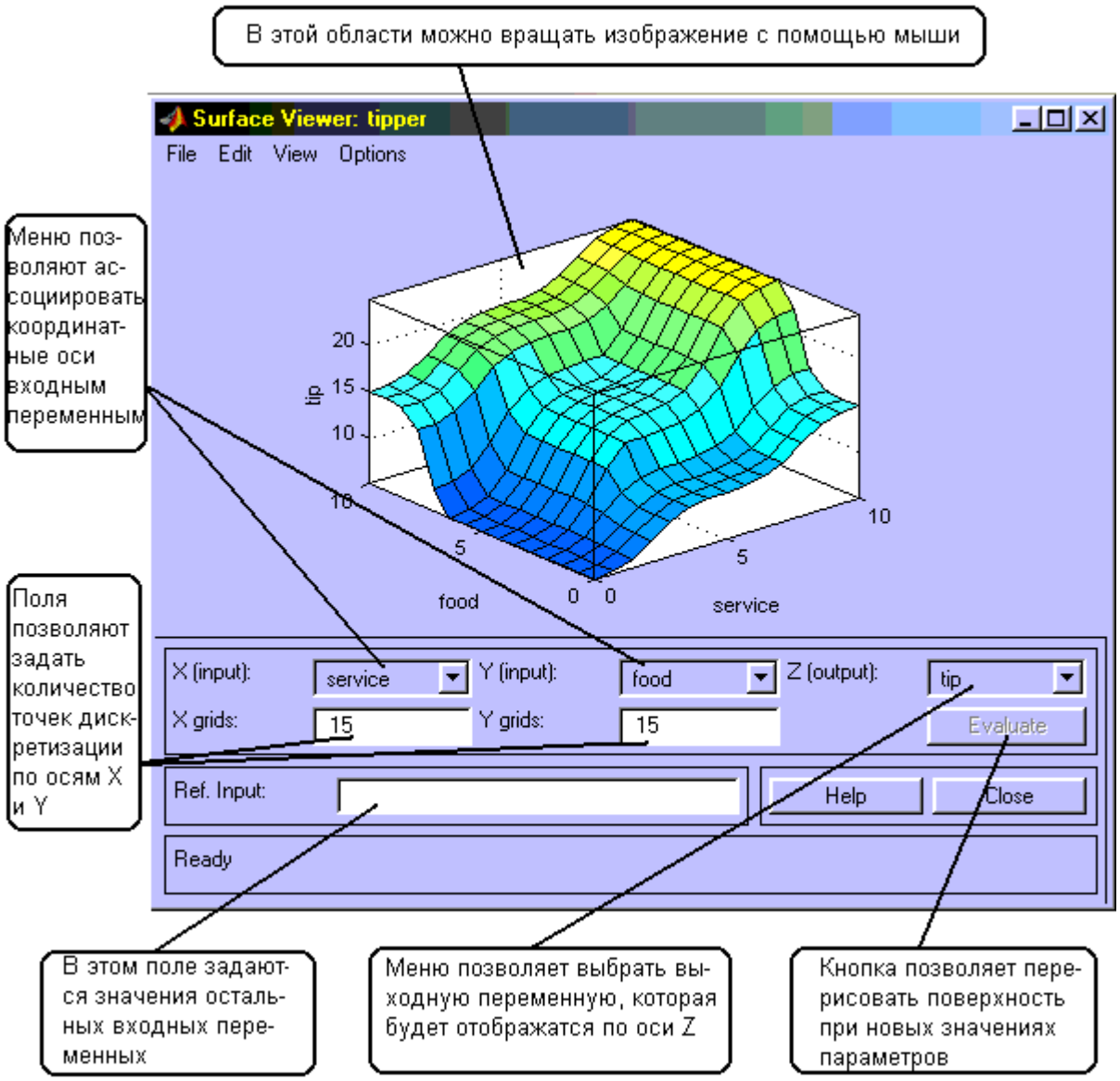


Рисунок 4.10 - Визуализация поверхности «входы-выход» для системы tipper с помощью Surface Viewer

Surface Viewer содержит верхних системных меню - **File, Edit, View, Options**, три меню выбора координатных осей -**X (input), Y (input), Z (output)**, три поля ввода информации – **X girds, Y girds, Ref**. **Input** и кнопку **Evaluate** для построения поверхности при новых параметрах.

Surface Viewer позволяет вращать поверхность “входы-выход” с помощью мыши. Для этого необходимо позиционировать курсор мыши на поверхности “входы-выход”, нажать на левую кнопку мыши и не отпуская ее повернуть графическое изображение на требуемый угол.

Поля **X girds** и **Y girds** предназначены для задания количества точек дискретизации по осям **X** и **Y**, для построения поверхности “входы-выход”. По умолчанию количество дискрет по каждой оси равно 15. Для изменения этого значения необходимо установить маркер на поле **X girds (Y girds)** и ввести новое значение.

Поле **Ref. Input** предназначено для задания значений входных переменных, кроме тех, которые ассоциированы с координатными осями. По умолчанию это значения середины интервалов изменения переменных. Для изменения этого значения необходимо установить маркер на поле **Ref. Input** и ввести новые значение.

Меню **X (input), Y (input), Z (output)** позволяют поставить в соответствие осям координат входные и выходные переменные. При этом входные переменные могут отображаться только по осям **X** и **Y**, а выходные переменные только по оси **Z**. В **Surface Viewer** предусмотрена возможность построения однофакторных зависимостей “вход-выход”. Для этого в меню второй координатной оси (**X (input**) или **Y (input)**) необходимо выбрать **none**.

Меню Options:

- Команда **Plot** позволяет управлять форматом вывода поверхности «входы-выход». При выборе этой команды появляется меню (рисунок 4.11) в котором необходимо выбрать формат вывода поверхности.

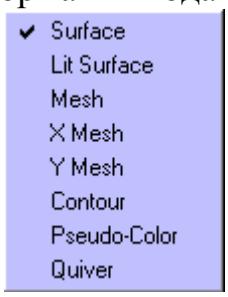


Рисунок 4.11 - Меню Plot

На рисунке 4.12 приведены поверхности «входы-выход» для системы tipper для всех поддерживаемых форматов.

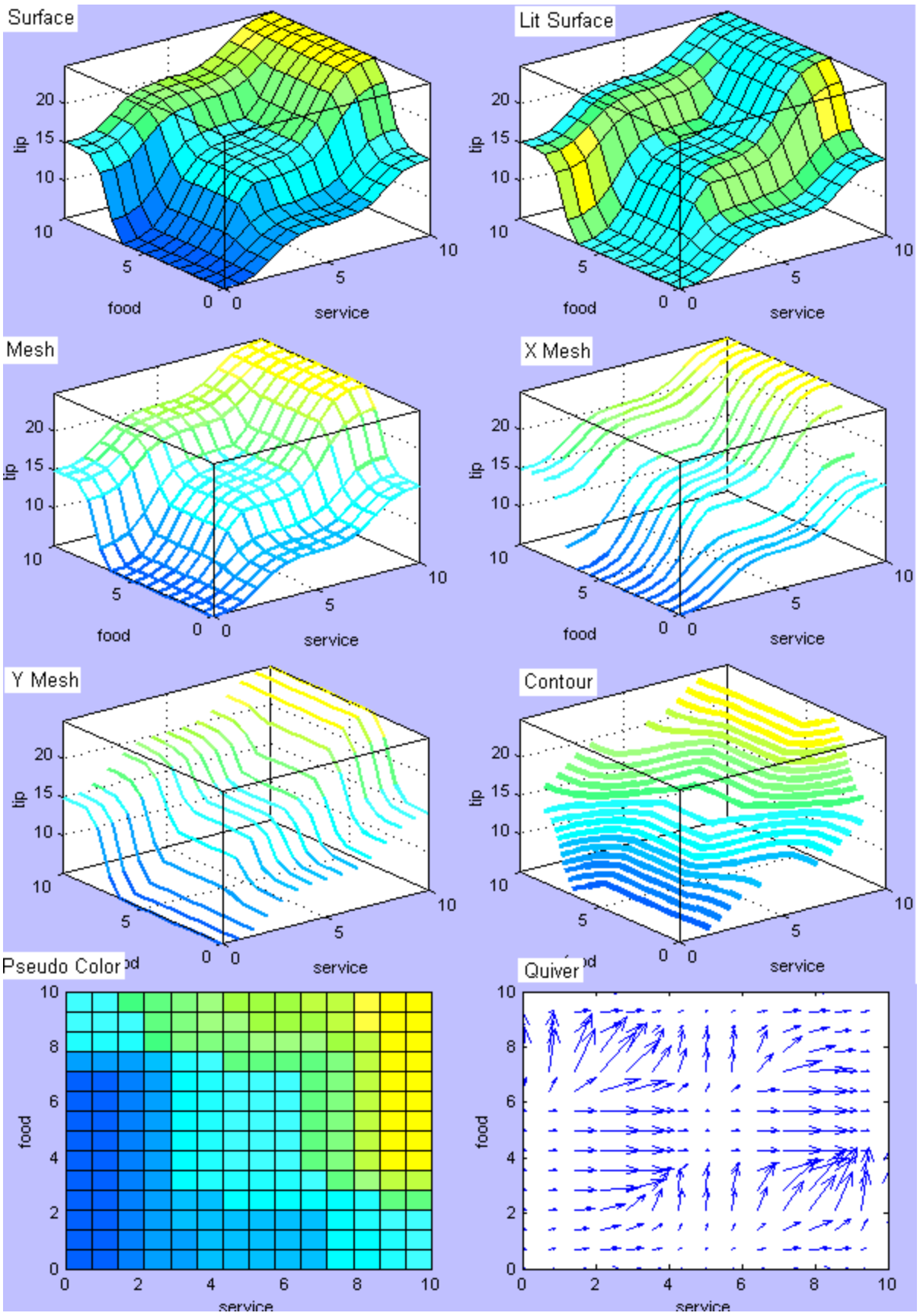


Рисунок 4.12 - Примеры форматов поверхности «входы-выход»

- Команда **Color Map** позволяет управлять палитрой цветов при выводе поверхности «входы-выход». При выборе этой команды появляется меню, в котором необходимо выбрать одну из палитр:

- **default** – использовать палитру, установленную по умолчанию;

- **blue** – холодная сине-голубая палитра;

- **hot** – теплая палитра, состоящая из черного, красного, желтого и белого цветов;

- **HSV** – палитра насыщенных цветов: красный, желтый, зеленый, циан, голубой, мажента, красный.

- Команда **Always evaluate** позволяет установить/отменить режим автоматического, т.е. без нажатия кнопки **Evaluate**, перерисовывания поверхности «входы-выход» при любом изменении параметров.

Общая постановка задачи

Спроектировать систему типа Мамдани средствами пакета Fuzzy Logic Toolbox на примере построения нечеткой аппроксимирующей системы. Выбранную нелинейную функцию описать базой правил для лингвистических переменных, описывающих z(x,y), определенных на множестве из пяти, семи, девяти и одиннадцати термов.

Примечание. Если z выходит за заданные границы, то z принимает значение равное значению минимальной или максимальной границы, соответственно. pi = 3,14159265358979323846264338327 95

Список индивидуальных данных

№ вар.	Функция	Область изменения		
		x	y	z
1	$z = x^2 + \sin(y - pi/2)$	-2<x<2	- pi <y< pi	-1<z<4
2	$z = \cos(x) + \sin(y - pi/2)$	- pi <x< pi	- pi <y< pi	-2<z<2
3	$z = x * \sin(y)$	-pi<x<pi	-pi<y<pi	-3<z<3
4	$z = x^2 - y^2$	-4<x<4	-4<y<4	-4<z<4
5	$z = x^2 * y^2$	-3<x<3	-3<y<3	0<z<4
6	$z = x * y$	-3<x<3	-3<y<3	-4<z<4
7	$z = (x - y) * y + 1$	-3<x<3	-3<y<3	-4<z<3
8	$z = y * \sin(x + y)$	-pi/2<x<pi/2	-pi/2<y<pi/2	-0.5<z<1.5
9	$z = \sin(2*x /pi) * \sin(2*y/pi)$	-5<x<5	5<y<5	1<z<1
10	$z = y * \sin(x)$	-pi/2<x<pi/2	-pi/2<y<pi/2	1<z<1.5
11	$z = y * \cos(x)$	-pi/2<x<pi/2	-pi/2<y<pi/2	-1<z<1.5
12	$z = x * \cos(x) + y * \sin(y)$	- 2 <x< 2	- pi/2 <y< pi/2	-0.6<z<2

Пример выполнения работы

Проектирование систем типа Мамдани.

Рассмотрим основные этапы проектирования систем типа Мамдани на примере создания системы нечеткого логического вывода, моделирующей зависимость $y = x_1^2 \cdot \sin(x_2 - 1)$, $x_1 \in [-7,3], x_2 \in [-4.4,1.7]$. Проектирование системы

нечеткого логического вывода будем проводить на основе графического изображения указанной зависимости.

Для построения трехмерного изображения функции $y = x_1^2 \cdot \sin(x_2 - 1),$ в области $x_1 \in [-7,3], x_2 \in [-4.4,1.7],$ составим следующую программу:

```
%Построение графика функции y=x1^2*sin(x2-1)
%в области x1є[-7,3] и x2є[-4.4,1.7].
n=15;
x1=-7:10/(n-1):3;
x2=-4.4:6.1/(n-1):1.7;
y=zeros(n,n);
for j=1:n
y(j,:)=x1.^2*sin(x2(j)-1);
end
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target');
```

В результате выполнения программы получим графическое изображение, приведенное на рисунке 4.13. Проектирование системы нечеткого логического вывода, соответствующей приведенному графику, состоит в выполнении следующей последовательности шагов.

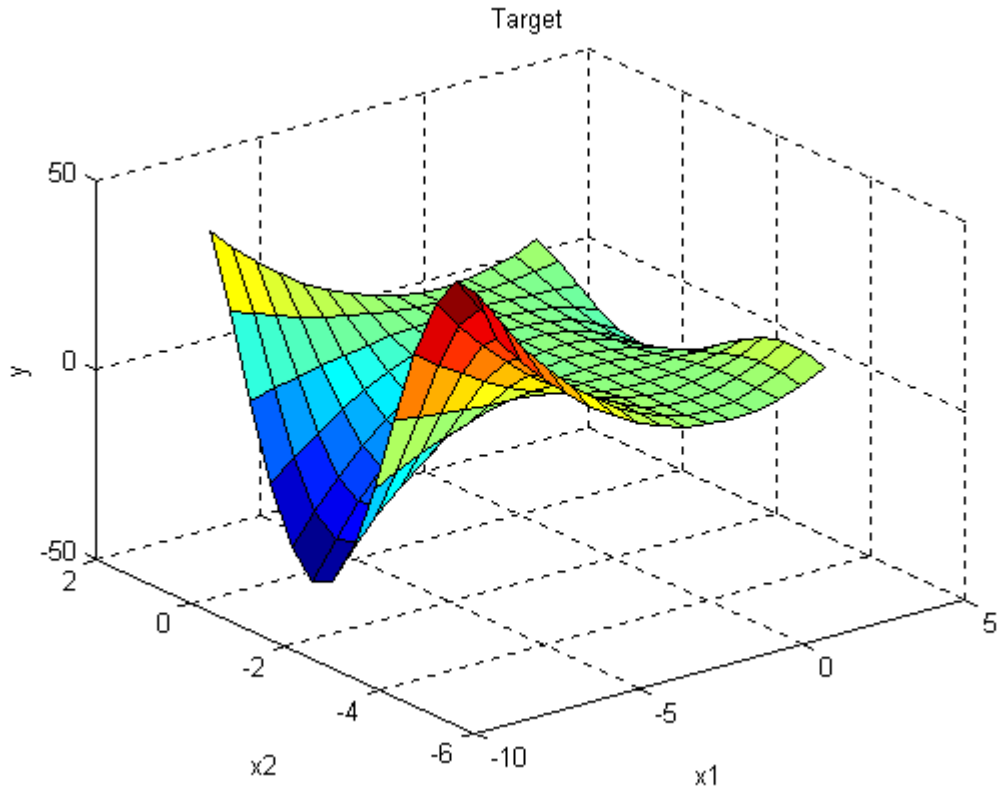


Рисунок 4.13 - Эталонная поверхность

Шаг 1. Для загрузки основного fis-редактора напечатаем слова **fuzzy** в командной строке.

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 **-7 3** в поле **Range** (рисунок 4.14) и нажмем <**Enter**>.

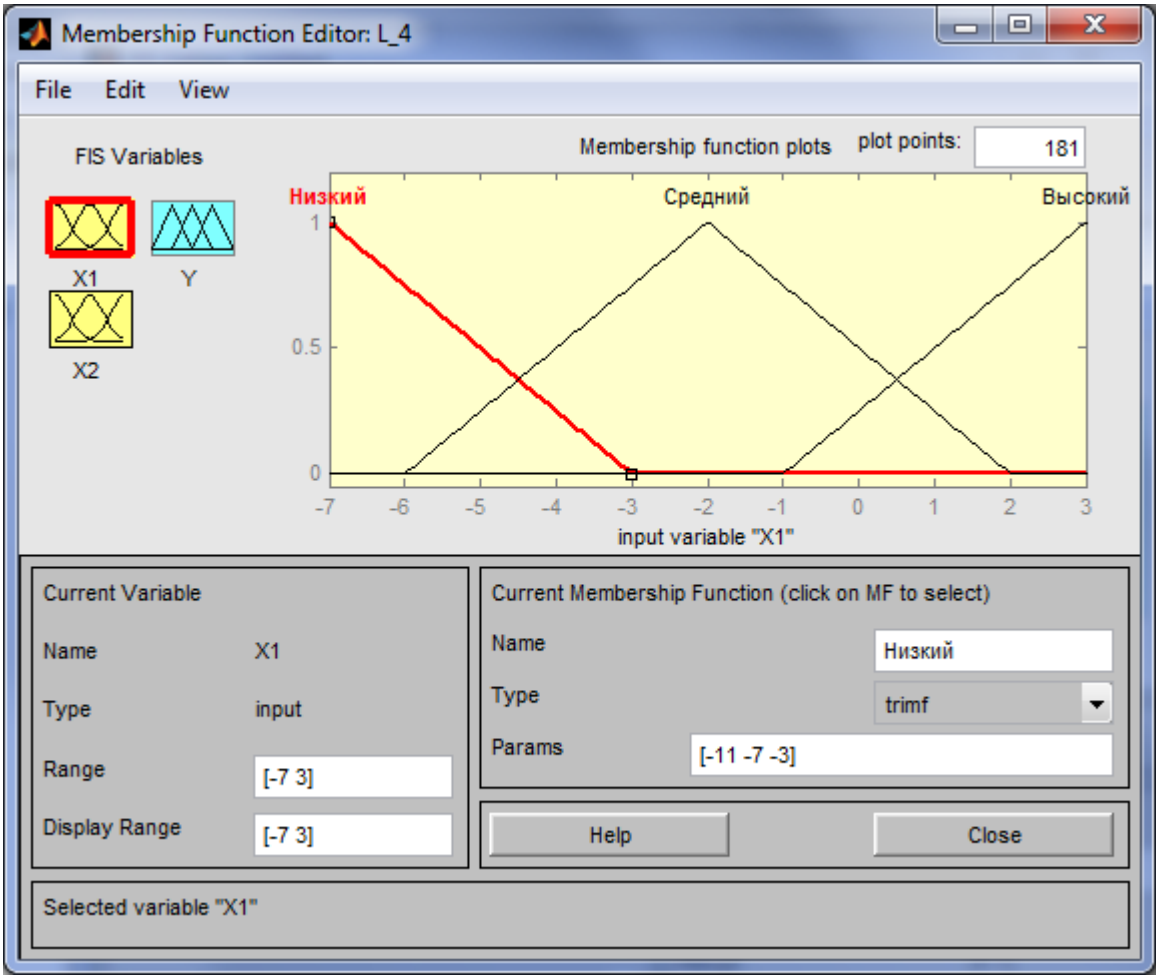


Рисунок 4.14 - Функции принадлежности переменной x1

Шаг 9. Зададим функции принадлежности переменной **x1**. Для лингвистической оценки этой переменной будем использовать 3 терма с треугольными функциями принадлежности. Для этого в меню **Edit** выберем команду **Add MFs...** В результате появиться диалоговое окно выбора типа и количества функций принадлежностей. По умолчанию это 3 терма с треугольными функциями принадлежности. Поэтому просто нажимаем <**Enter**>.

Шаг 10. Зададим наименования термов переменной **x1**. Для этого делаем один щелчок левой кнопкой мыши по графику первой функции принадлежности (рисунок 4.14). Затем вводим наименование терма, например, **Низкий**, в поле **Name** и нажмем <**Enter**>. Затем делаем один щелчок левой кнопкой мыши по графику второй функции принадлежности и вводим наименование терма, например, **Средний**, в поле **Name** и нажмем <**Enter**>. Еще раз делаем один щелчок левой кнопкой мыши по графику третьей функции принадлежности и вводим наименование терма, например, **Высокий**, в поле **Name** и нажмем <**Enter**>. В результате получим графическое окно, изображенное на рисунке 4.14.

Шаг 11. Зададим функции принадлежности переменной **x2**. Для лингвистической оценки этой переменной будем использовать 5 термов с гауссовскими функциями принадлежности. Для этого активизируем переменную **x2** с помощью щелчка левой кнопки мыши на блоке **x2**. Зададим диапазон изменения переменной **x2**. Для этого напечатаем **-4.4 1.7** в поле **Range** (рисунок 4.15) и нажмем<**Enter**>. Затем в меню **Edit** выберем команду **Add MFs....** В появившимся диалоговом окне выбираем тип функции принадлежности **gaussmf** в поле **MF type** и 5 термов в поле **Number of MFs**. После этого нажимаем <**Enter**>.

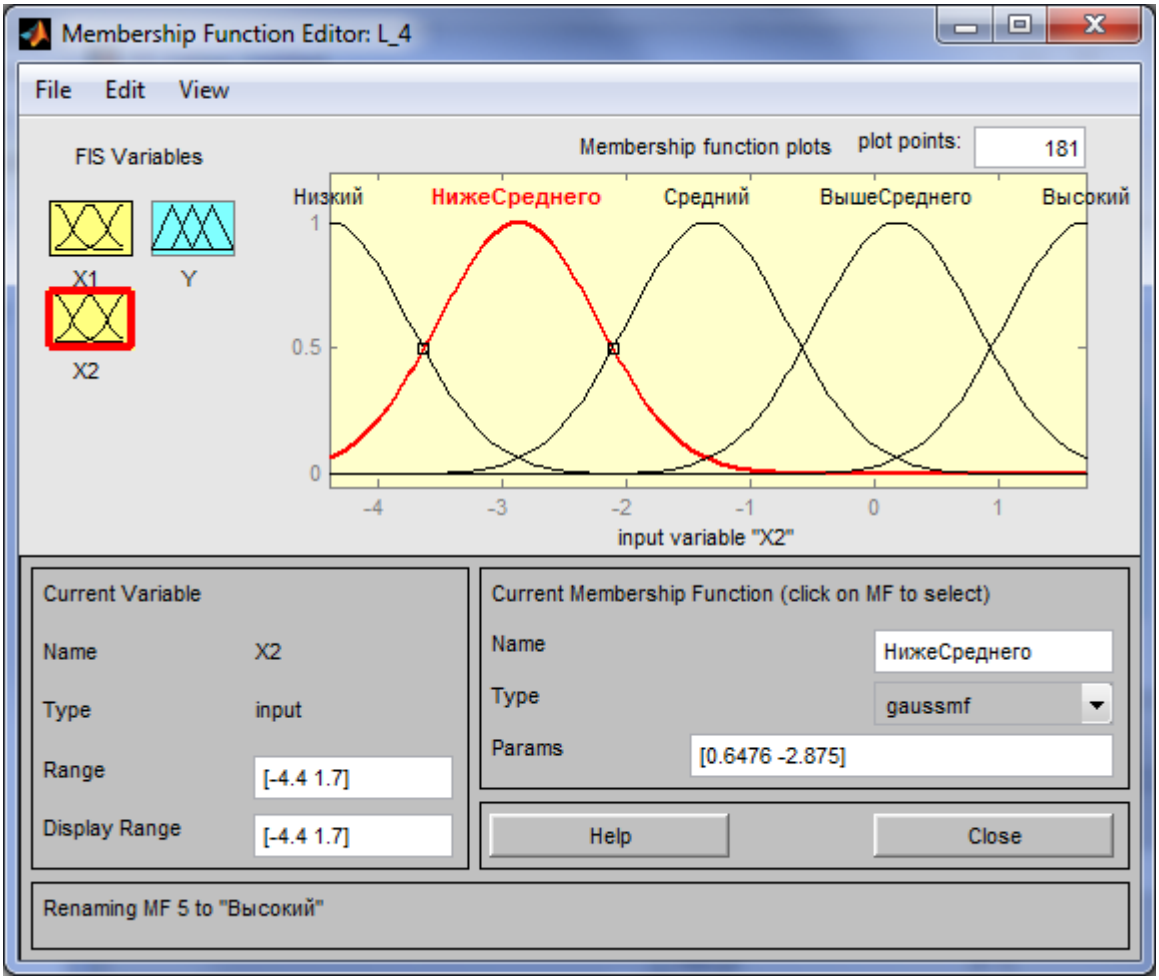


Рисунок 4.15 - Функции принадлежности переменной x2

Шаг 12. По аналогии с шагом 10 зададим следующие наименования термов переменной **x2**: **Низкий, Ниже среднего, Средний, Выше среднего, Высокий**. В результате получим графическое окно, изображенное на рисунке 4.15.

Шаг 13. Зададим функции принадлежности переменной **y**. Для лингвистической оценки этой переменной будем использовать 5 термов с треугольными функциями принадлежности. Для этого активизируем переменную **y** с помощью щелчка левой кнопки мыши на блоке **y**. Зададим диапазон изменения переменной **y**. Для этого напечатаем **-50 50** в поле **Range**

(рисунок 4.16) и нажмем<**Enter**>.Затем в меню **Edit** выберем команду **Add MFs....** В появившимся диалоговом окне выбираем 5 термов в поле Number **of MFs**. После этого нажимаем <**Enter**>.

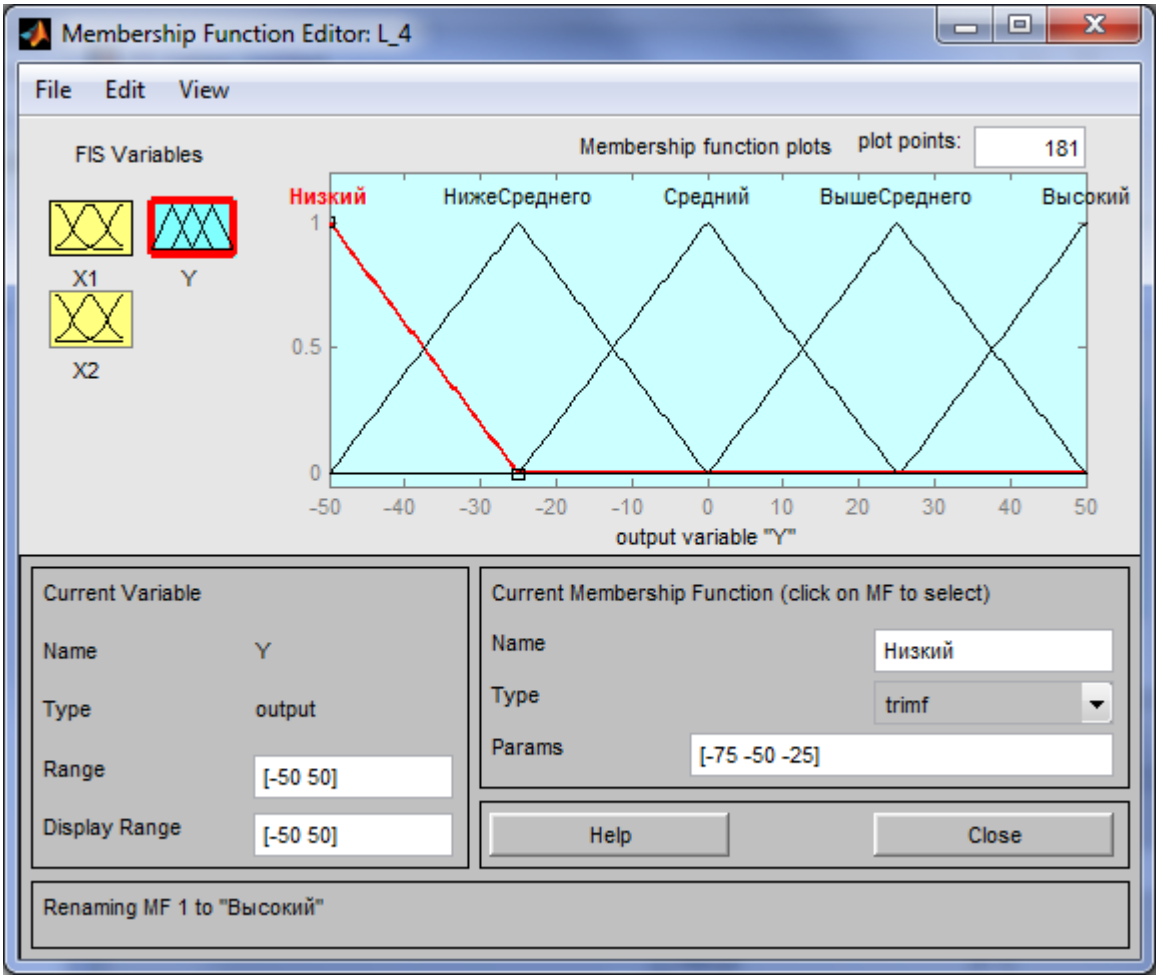


Рисунок 4.16 - Функции принадлежности переменной y

Шаг 14. По аналогии с шагом 10 зададим следующие наименования термов переменной **y**: **Низкий, Ниже среднего, Средний, Выше среднего, Высокий**. В результате получим графическое окно, изображенное на рисунке 4.16.

Шаг 15. Перейдем в редактор базы знаний **RuleEditor**. Для этого выберем в меню **Edit** выберем команду **Edit rules....**

Шаг 16. На основе визуального наблюдения за графиком, изображенным на рисунке 4.13 сформулируем следующие девять правил:

Если x1=Средний, то y=Средний;
Если x1=Низкий и x2=Низкий, то y=Высокий;
Если x1=Низкий и x2=Высокий, то y=Высокий;
Если x1=Высокий и x2=Высокий, то y=Выше Среднего;
Если x1=Высокий и x2=Низкий, то y=Выше Среднего;
Если x1=Высокий и x2=Средний, то y=Средний;
Если x1=Низкий и x2=Средний, то y=Низкий;

Если x1=Высокий и x2=Выше Среднего, то y=Средний;
Если x1=Высокий и x2=Ниже Среднего, то y=Средний.

Для ввода правила необходимо выбрать в меню соответствующую комбинацию термов и нажать кнопку **Add rule**. На рисунке 4.17 изображено окно редактора базы знаний после ввода всех девяти правил. Число, приведенное в скобках в конце каждого правила представляет собой весовым коэффициент соответствующего правила.

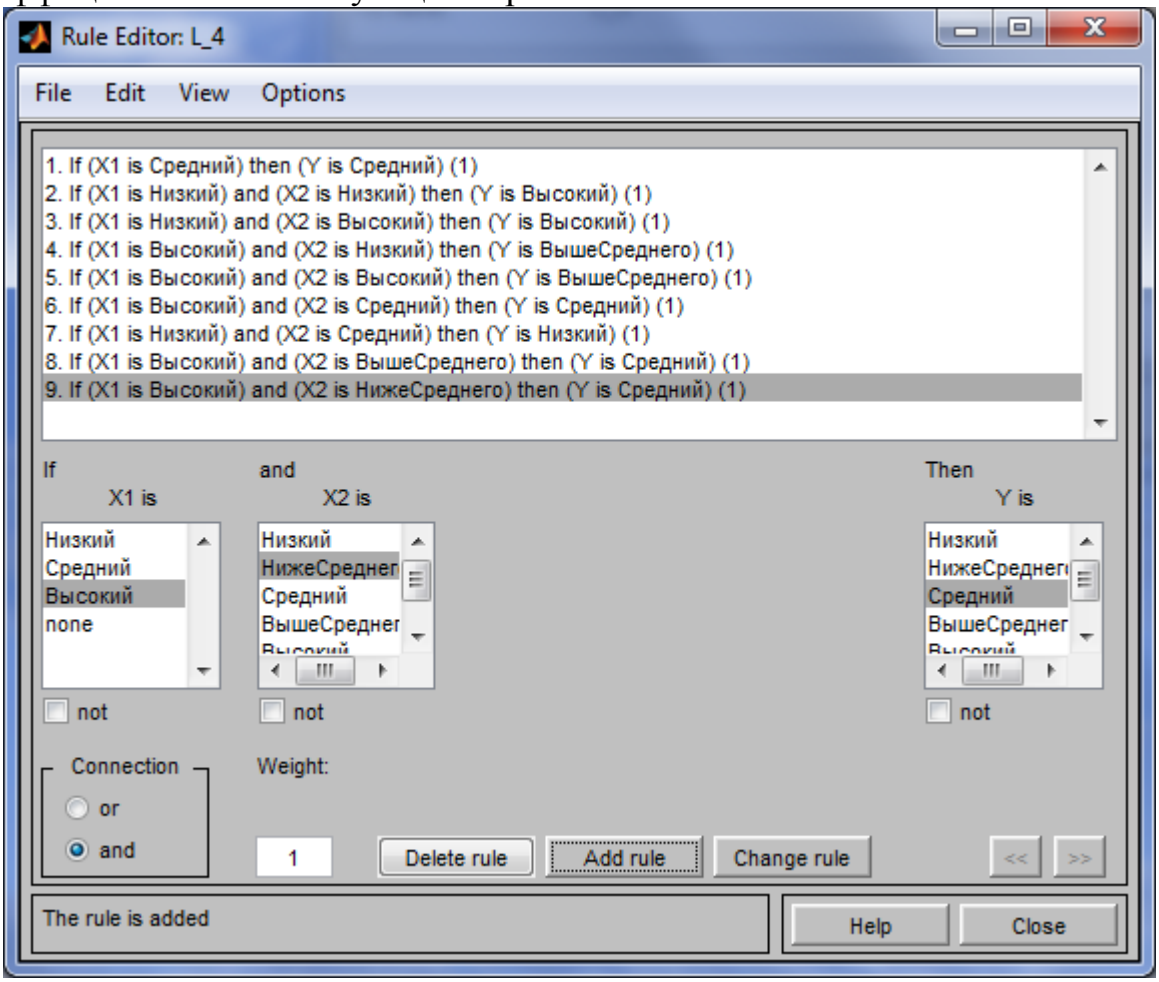


Рисунок 4.17 - База знаний в RuleEditor

Шаг 17. Сохраним созданную систему. Для этого в меню **File** выбираем в подменю **Export** команду **To disk**.

На рисунке 4.18 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.

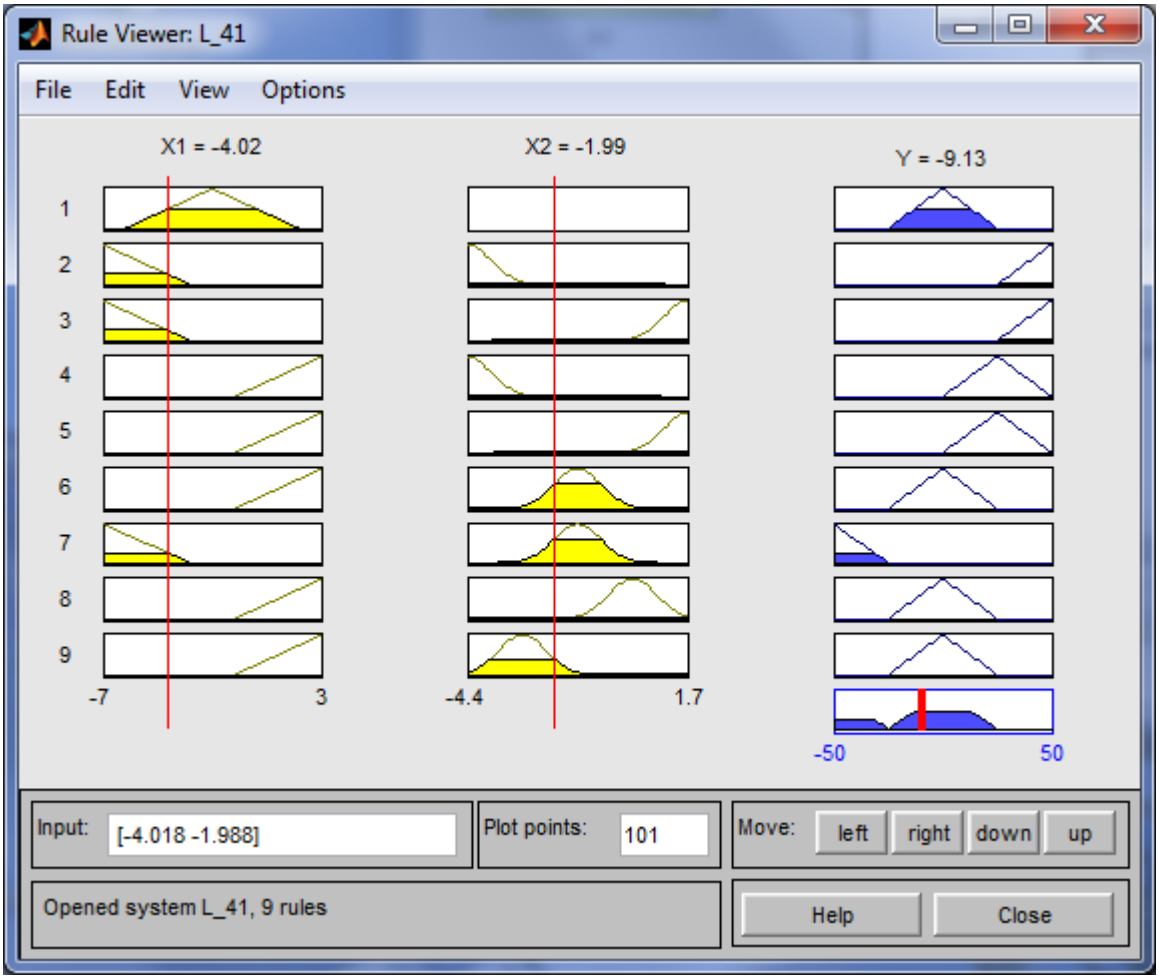


Рисунок 4.18 - Визуализация нечеткого логического вывода в RuleViewer

На рисунке 4.19 приведена поверхность «входы-выход», соответствующая синтезированной нечеткой системе. Для вывода этого окна необходимо использовать команду **View surface...** меню **View**. Сравнивая поверхности на рисунке 4.13 и на рисунке 4.19 можно сделать вывод, что нечеткие правила достаточно хорошо описывают сложную нелинейную зависимость.

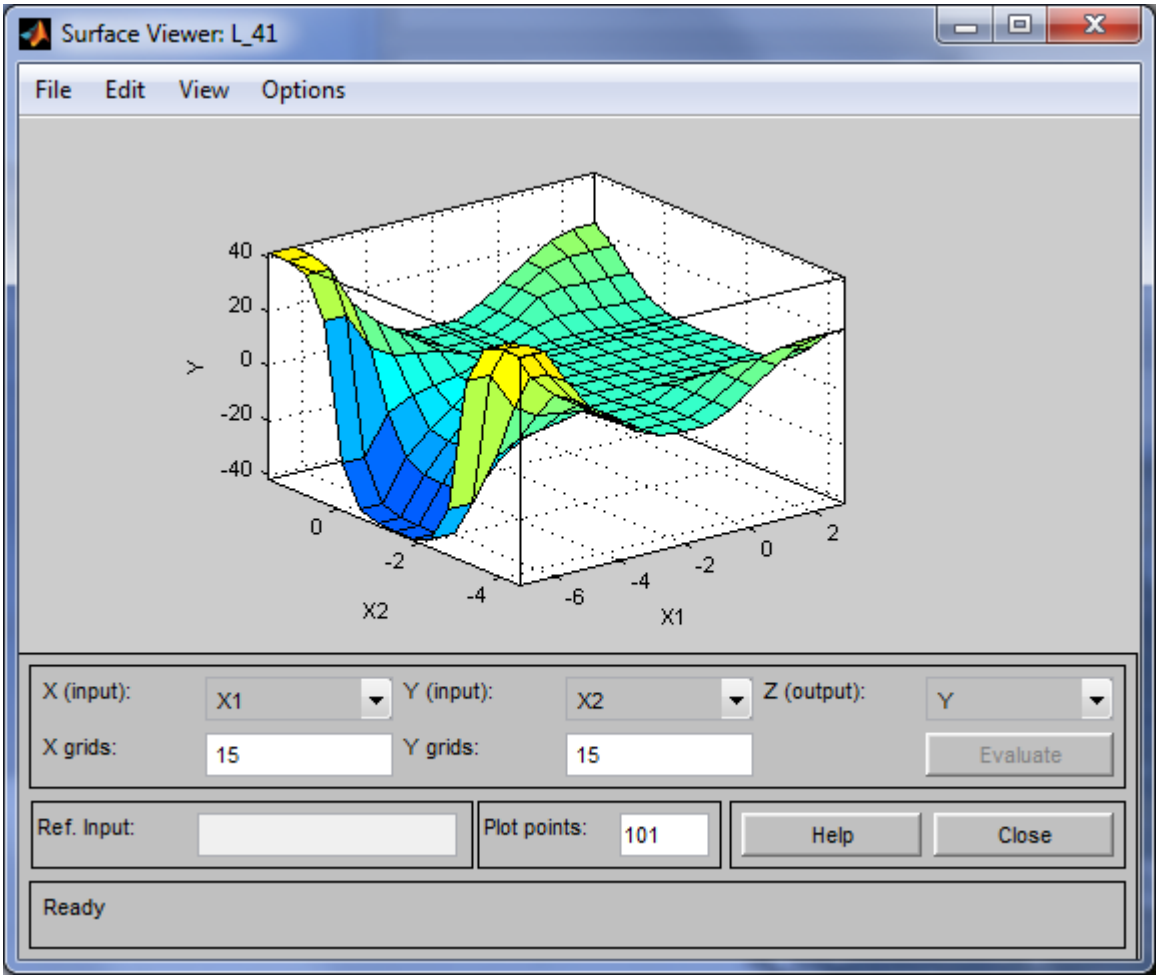


Рисунок 4.19 - Поверхность «входы-выход» в окне SurfaceViwer

Контрольные вопросы к защите

1. Для чего предназначен FIS-редактор?
2. Назовите основные элементы окна FIS-редактора?
3. Опишите редактор функций принадлежности?
4. Какого типа имеются функции принадлежности в пакете fuzzy logic?
5. Опишите окно редактора баз знаний?
6. Каким образом можно произвести визуализацию нечеткого логического вывода?
7. Что такое визуализация поверхности типа «входы-выходы»?

Лабораторная работа №5. Проектирование системы типа Сугэно средствами пакета Fuzzy Logic Toolbox на примере построения нечеткой аппроксимирующей системы

Цель работы: Проектирование систем нечеткого логического вывода типа Сугэно.

Теоретическая часть

Теоретическая часть аналогична описанию, представленному в 4 лабораторной работе.

Общая постановка задачи

Спроектировать систему типа Сугэно средствами пакета Fuzzy Logic Toolbox на примере построения нечеткой аппроксимирующей системы. Выбранную нелинейную функцию описать базой правил для лингвистических переменных, описывающих z(x,y), определенных на множестве из пяти, семи, девяти и одиннадцати термов.

Примечание. Если z выходит за заданные границы, то z принимает значение равное значению минимальной или максимальной границы, соответственно. pi = 3,14159265358979323846264338327 95

Список индивидуальных данных

№ вар.	Функция	Область изменения		
		x	y	z
1	$z = x^2 + \sin(y - pi/2)$	-2<x<2	- pi <y< pi	-1<z<4
2	$z = \cos(x) + \sin(y - pi/2)$	- pi <x< pi	- pi <y< pi	-2<z<2
3	$z = x * \sin(y)$	-pi<x<pi	-pi<y<pi	-3<z<3
4	$z = x^2 - y^2$	-4<x<4	-4<y<4	-4<z<4
5	$z = x^2 * y^2$	-3<x<3	-3<y<3	0<z<4
6	$z = x * y$	-3<x<3	-3<y<3	-4<z<4
7	$z = (x - y) * y + 1$	-3<x<3	-3<y<3	-4<z<3
8	$z = y * \sin(x + y)$	-pi/2<x<pi/2	-pi/2<y<pi/2	-0.5<z<1.5
9	$z = \sin(2*x/pi) * \sin(2*y/pi)$	-5<x<5	5<y<5	1<z<1
10	$z = y * \sin(x)$	-pi/2<x<pi/2	-pi/2<y<pi/2	1<z<1.5
11	$z = y * \cos(x)$	-pi/2<x<pi/2	-pi/2<y<pi/2	-1<z<1.5
12	$z = x * \cos(x) + y * \sin(y)$	- 2 <x< 2	- pi/2 <y< pi/2	-0.6<z<2

Пример выполнения работы

Проектирование систем типа Сугэно.

Рассмотрим основные этапы проектирования систем типа Сугэно на примере создания системы нечеткого логического вывода, моделирующей зависимость $y = x_1^2 \cdot \sin(x_2 - 1)$, $x_1 \in [-7,3], x_2 \in [-4.4,1.7]$ (рисунок 4.13).

Проектирование системы нечеткого логического вывода типа Сугэно состоит в выполнении следующей последовательности шагов.

Шаг 1. Для загрузки основного fis-редактора напечатаем слова **fuzzy** в командной строке.

Шаг 2. Выберем тип системы. Для этого в меню **File** выбираем в подменю **New fis…** команду **Sugeno**.

Шаг 3. Добавим вторую входную переменную. Для этого в меню **Edit** выбираем команду **Add input**.

Шаг 4. 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 5. 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 6. 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 7. Зададим имя системы. Для этого в меню **File** выбираем в подменю **Export** команду **To disk** и введем имя файла, например, **FirstSugeno**.

Шаг 8. Перейдем в редактор функций принадлежности. Для этого сделаем двойной щелчок левой кнопкой мыши на блоке **x1.**

Шаг 9. Зададим диапазон изменения переменной **x1**. Для этого напечатаем -7 3 в поле **Range** (рисунок 5.1) и нажмем <**Enter**>.

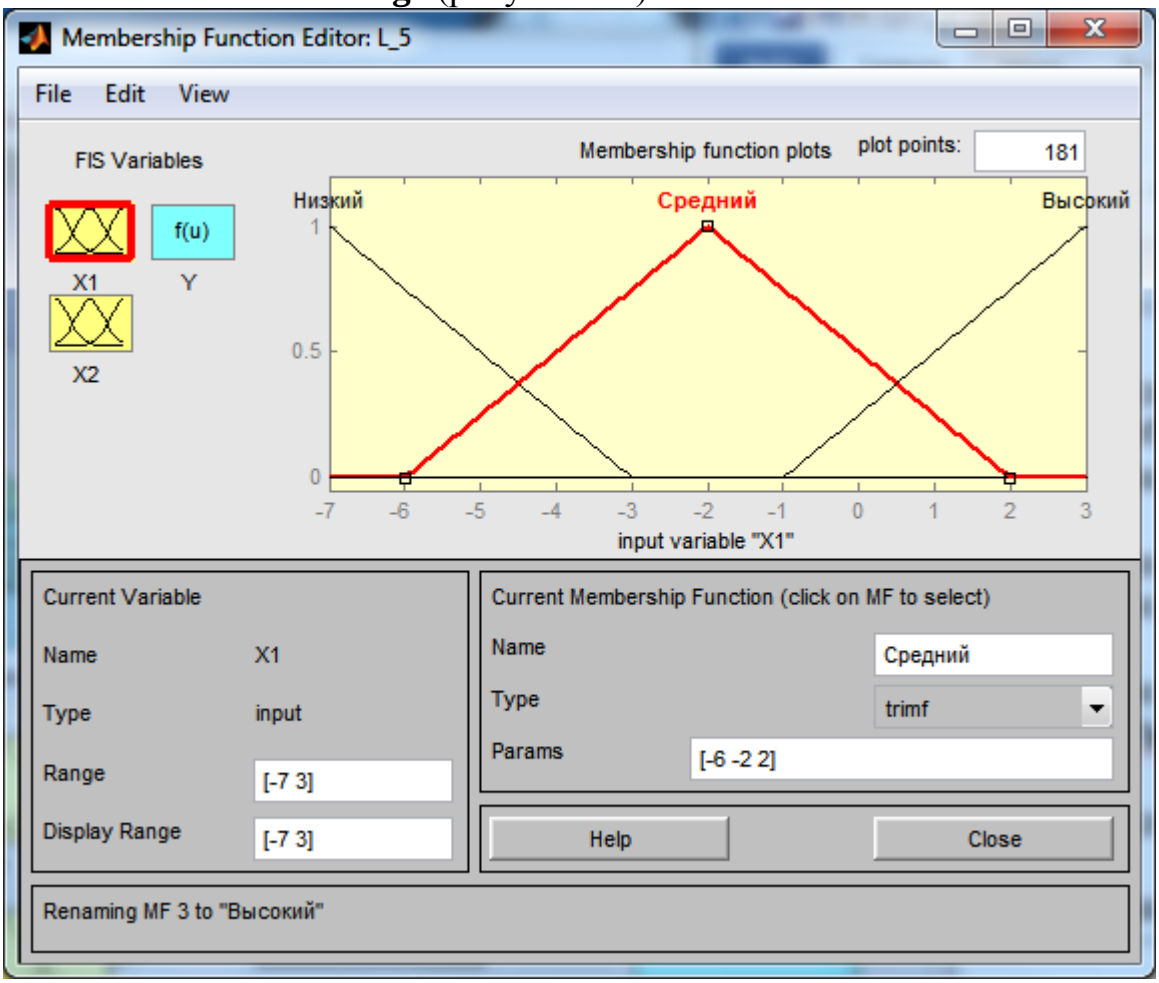


Рисунок 5.1 - Функции принадлежности переменной x1

Шаг 10. Зададим функции принадлежности переменной **x1**. Для лингвистической оценки этой переменной будем использовать, 3 терма с треугольными функциями принадлежности. Зададим наименования термов переменной **x1**. Для этого делаем один щелчок левой кнопкой мыши по графику первой функции принадлежности (рисунок 5.1). Затем напечатаем наименование терма **Низкий** в поле **Name**. Затем делаем один щелчок левой кнопкой мыши по графику второй функции принадлежности и вводим наименование терма **Средний** в поле **Name**. Еще раз делаем один щелчок левой кнопкой мыши по графику третьей функции принадлежности и вводим наименование терма **Высокий** в поле **Name** и нажмем <**Enter**>.

Шаг 11. Зададим функции принадлежности переменной **x2**. Для лингвистической оценки этой переменной будем использовать 3 терма с треугольными функциями принадлежности. Для этого активизируем переменную **x2** с помощью щелчка левой кнопки мыши на блоке **x2**. Зададим диапазон изменения переменной **x2**. Для этого напечатаем -4.4 1.7 в поле **Range** (рисунок 5.2) и нажмем <**Enter**>. По аналогии с предыдущим шагом зададим следующие наименования термов переменной **x2**: **Низкий, Средний, Высокий.**

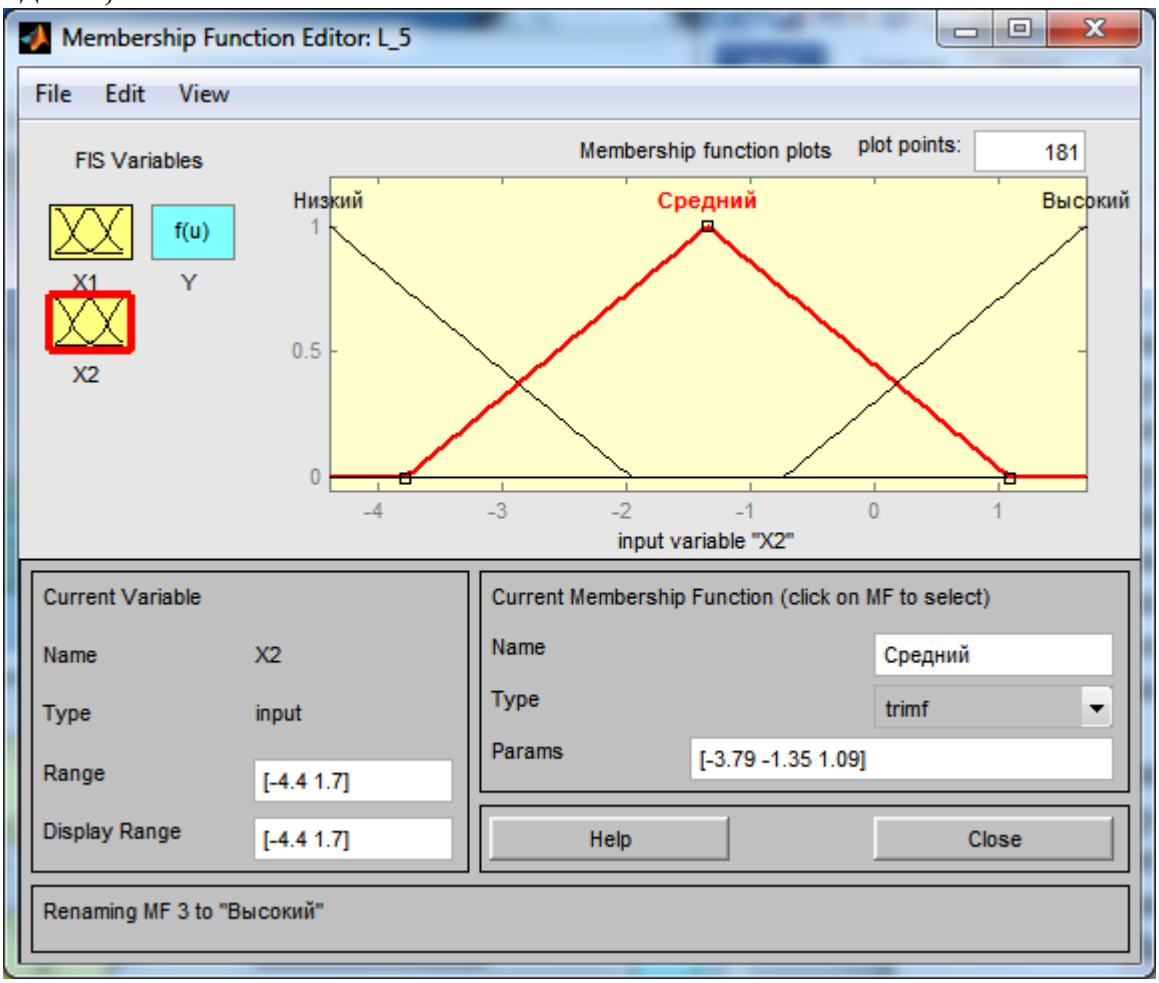


Рисунок 5.2 - Функции принадлежности переменной x2

Шаг 12. Зададим линейные зависимости между входами и выходом, приведенные в базе знаний. Для этого активизируем переменную **y** с помощью щелчка левой кнопки мыши на блоке **y**. В правом верхнем угле появилось обозначение трех функций принадлежности, каждая из которых соответствует одной линейной зависимости между входами и выходам. В базе знаний содержится 5 различных зависимостей: y=50; y=4x1-x2; y=2x1+2x2+1; y=8x1+2x2+8; y=0.

Если x1=Средний, то y=0;

Если x1=Высокий и x2=Высокий, то y=2x1+2x2+1;

Если x1=Высокий и x2=Низкий, то y=4x1-x2;

Если x1=Низкий и x2=Средний, то y=8x1+2x2+8;

Если x1=Низкий и x2=Низкий, то y=50;

Если x1=Низкий и x2=Высокий, то y=50.

Поэтому добавим еще две зависимости путем выбора команды **Add Mfs…** меню Edit. В появившимся диалоговом окне в поле **Number of MFs** выбираем 2 и нажимаем кнопку **OK**.

Шаг 13. Зададим наименования и параметры линейных зависимостей. Для этого делаем один щелчок левой кнопкой мыши по наименованию первой зависимости **mf1**. Затем печатаем наименование зависимости, например 50, в поле **Name**, и устанавливаем тип зависимости – константа путем выбора опции **Constant** в меню **Type**. После этого вводим значение параметра – 50 в поле **Params**.

Аналогично для второй зависимости **mf2** введем наименование зависимости, например **8x1+2x2+8**. Затем укажем линейный тип зависимости путем выбора опции **Linear** в меню **Type** и введем параметры зависимости **8 2 8** в поле **Params**. Для линейной зависимости порядок параметров следующий: первый параметр – коэффициент при первой переменной, второй – при второй и т.д., и последний параметр – свободный член зависимости.

Аналогично для третьей зависимости **mf3** введем наименование зависимости, например **1+2x1+2x2**, укажем линейный тип зависимости и введем параметры зависимости **2 2 1**.

Для четвертой зависимости **mf4** введем наименование зависимости, например **4x1-x2**, укажем линейный тип зависимости и введем параметры зависимости **4 -1 0**.

Для пятой зависимости **mf5** введем наименование зависимости, например **0**, укажем тип зависимости - константа и введем параметр зависимости **0**.

В результате получим графическое окно, изображенное на рисунке 5.3.

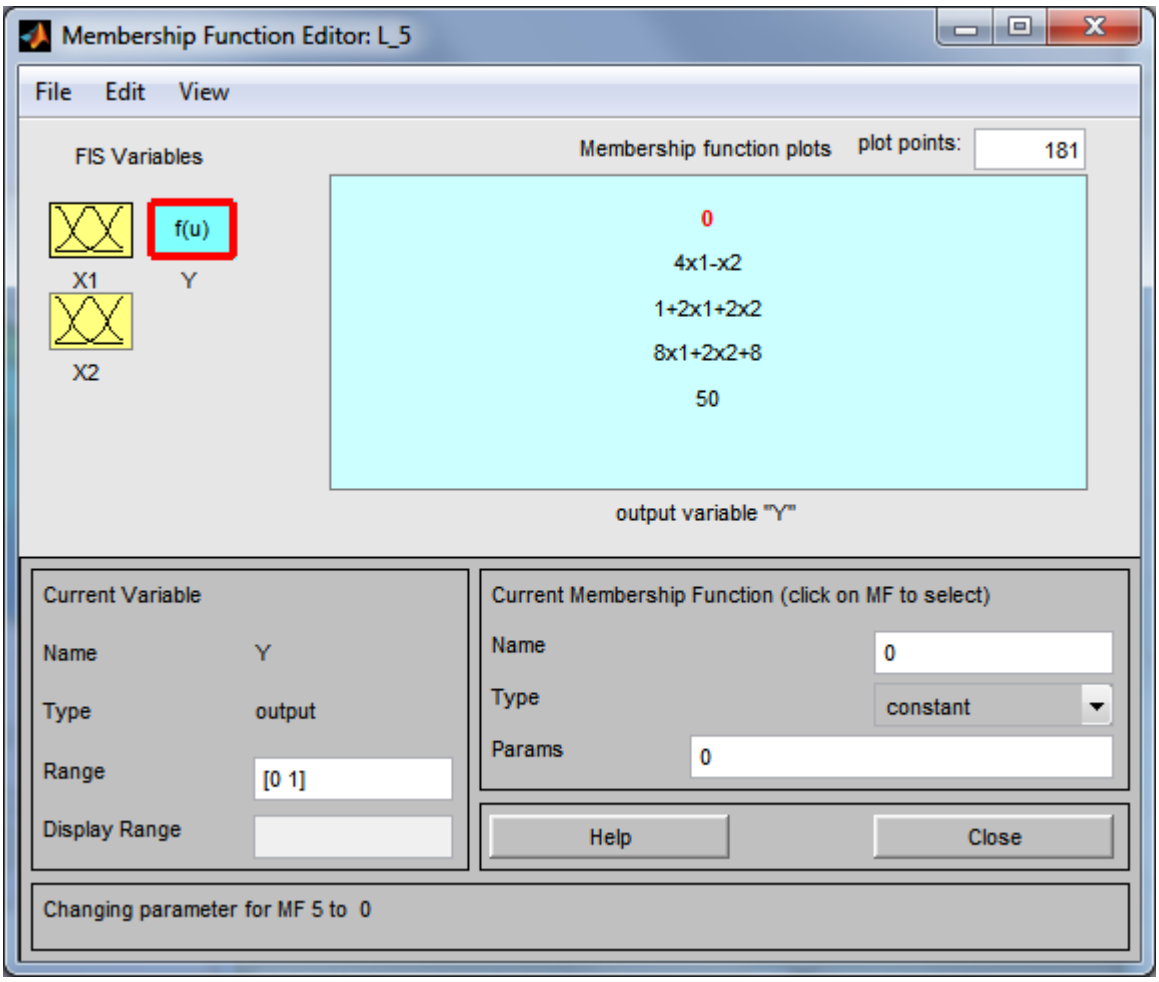


Рисунок 5.3 - Окно линейных зависимостей «входы-выход»

Шаг 14. Перейдем в редактор базы знаний **RuleEditor**. Для этого выберем в меню **Edit** команду **Edit rules....** и введем правила базы знаний. Для ввода правила необходимо выбрать соответствующую комбинацию термов и зависимостей и нажать кнопку **Add rule**. На рисунке 5.4 изображено окно редактора базы знаний после ввода всех шести правил.

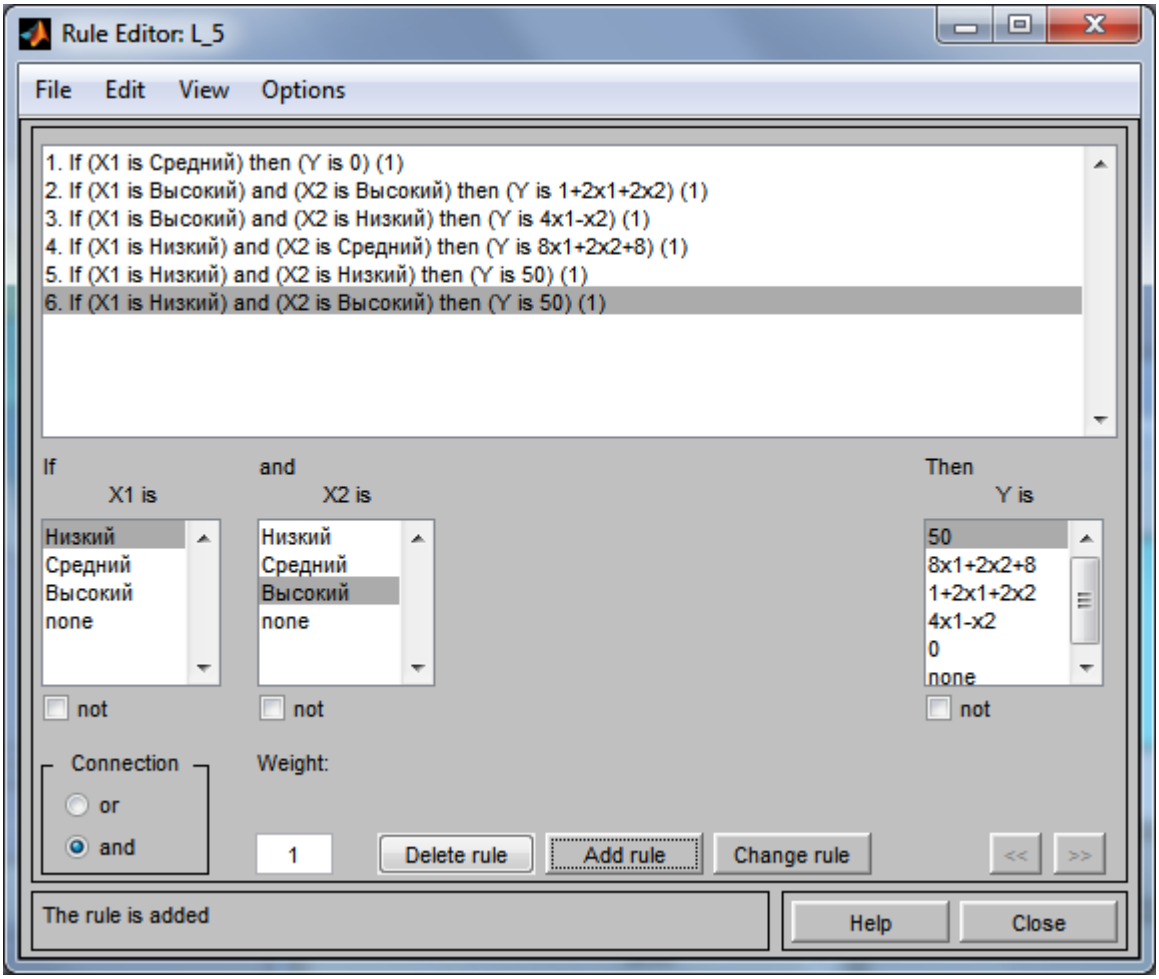


Рисунок 5.4 - Нечеткая база знаний для системы типа Сугэно

На рисунке 5.5 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. Как видно из этого рисунка значение выходной переменной рассчитывается как среднее взвешенное значение результатов вывода по каждому правилу.

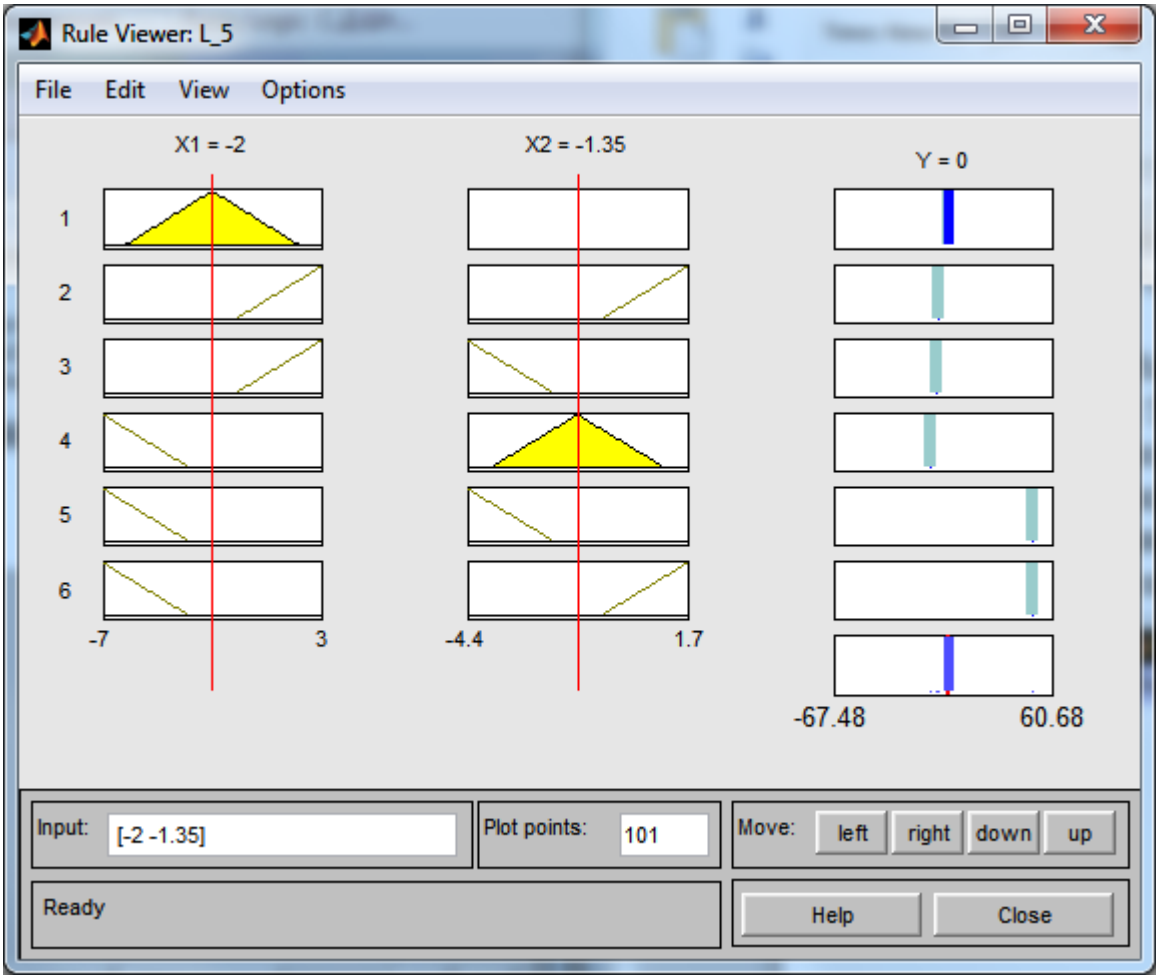


Рисунок 5.5 - Визуализация нечеткого логического вывода для системы типа Сугэно

На рисунке 5.6 приведена поверхность «входы-выход», соответствующая синтезированной нечеткой системе. Для вывода этого окна необходимо использовать команду **View surface...** меню **View**. Сравнивая поверхности на рисунке 4.13, рисунке 4.19 и на рисунке 5.6 можно сделать вывод, что нечеткие правила достаточно хорошо описывают сложную нелинейную зависимость. При этом, модель типа Сугэно более точная. Преимущество моделей типа Мамдани состоит в том, что правила базы знаний являются прозрачными и интуитивно понятными, тогда как для моделей типа Сугэно не всегда ясно какие линейные зависимости «входы-выход» необходимо использовать.

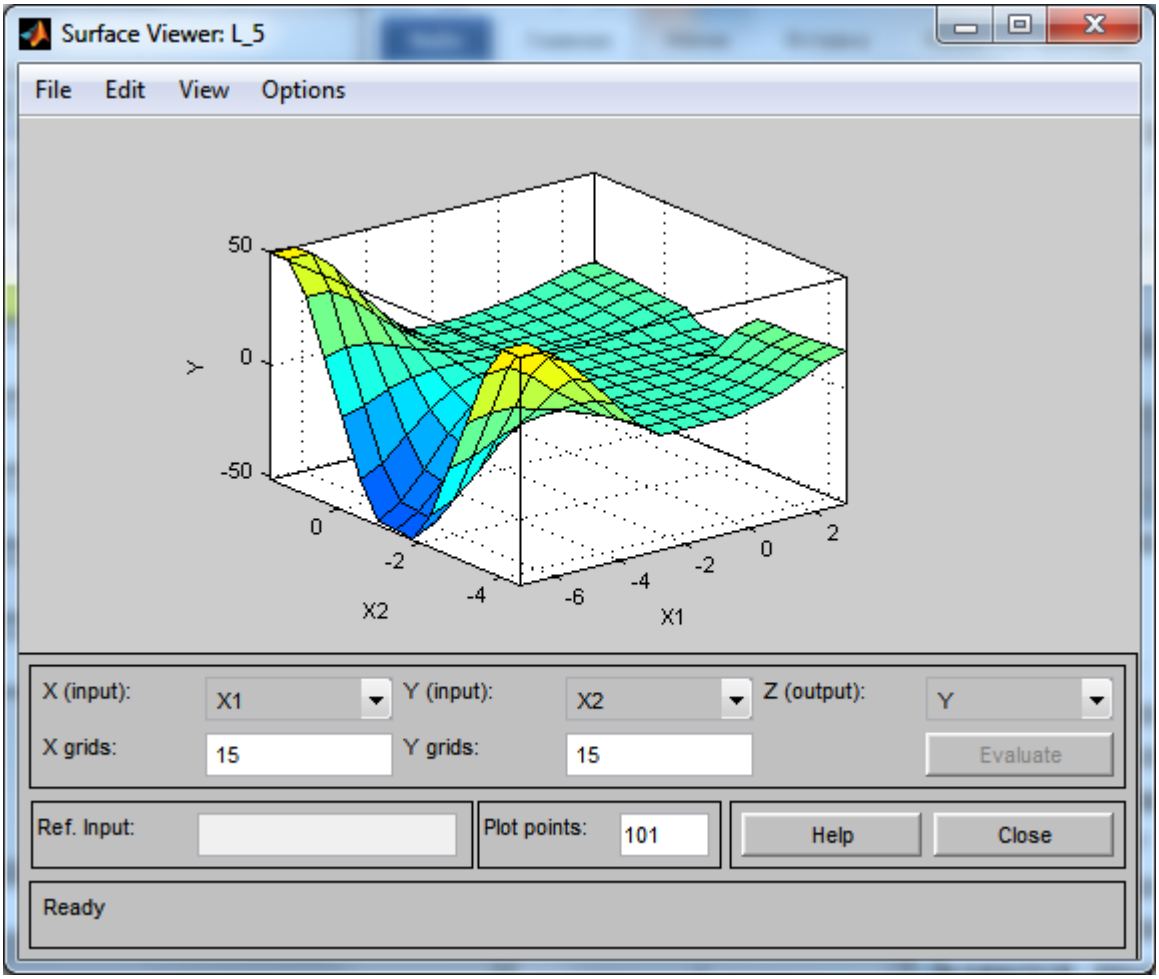


Рисунок 5.6 - Поверхность «входы-выход» для системы типа Сугэно

Контрольные вопросы к защите

1. Для чего предназначен FIS-редактор?
2. Назовите основные элементы окна FIS-редактора?
3. Опишите редактор функций принадлежности?
4. Какого типа имеются функции принадлежности в пакете fuzzy logic?
5. Опишите окно редактора баз знаний?
6. Каким образом можно произвести визуализацию нечеткого логического вывода?
7. Что такое визуализация поверхности типа «входы-выходы»?
8. Назовите ключевые отличия подходов проектирования нечетких систем Мамдани и Сугэно?

Лабораторная работа №6. Проектирование интеллектуальной системы на основе нечетких знаний.

Цель работы: Разработать компьютерную модель нечеткой экспертной системы и исследовать ее работу.

Теоретическая часть

Для моделирования многомерных зависимостей "входы - выход" целесообразно использовать иерархические системы нечеткого логического вывода. В этих системах выходная переменная одной базы знаний является входной для другой базы знаний. На рисунке 6.1 приведен пример иерархической нечеткой базы знаний, моделирующей зависимость y=f(x1,x2,x3,x4,x5,x6) с использованием трех баз знаний. Эти базы знаний описывают такие зависимости: y1=f1(x1,x2), y2=f2(x4,x5,x6) и y=f3(y1,x3,y2).

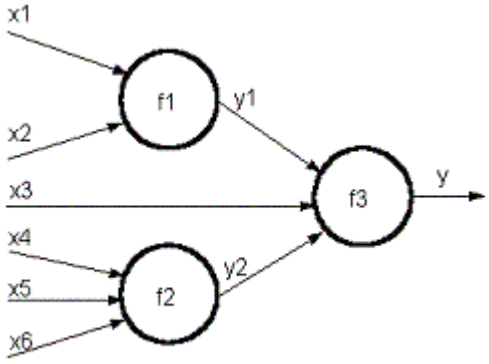


Рисунок 6.1 - Пример иерархической нечеткой базы знаний

Применение иерархических нечетких баз знаний позволяет преодолеть "проклятие размерности". При большом количестве входов эксперту трудно описать причинно-следственные связи в виде нечетких правил. Это обусловлено тем, что в оперативной памяти человека может одновременно хранится не более 7±2 понятий-признаков. Следовательно, количество входных переменных в одной базе знаний не должно превышать это магическое число. *Более поздние исследования показали, что хорошие базы знаний получаются, когда количество входов не превышает пяти шести*. Поэтому, при большем количестве входных переменных необходимо их иерархически классифицировать с учетом приведенных выше рекомендаций. Обычно, выполнение такой классификации не составляет трудностей для эксперта, так как при принятии решений человек иерархически учитывает влияющие факторы.

Преимущество иерархических баз знаний заключается еще и в том, что они позволяют небольшим количество нечетких правил адекватного описать многомерные зависимости "входы - выход". Пусть, для лингвистической оценки переменных используется по пять термов. Тогда, максимальное количество правил для задания зависимости y=f(x1,x2,x3,x4,x5,x6) с помощью одной базы знаний будет равным 5^6=15625 (конечно, для адекватного описания зависимости "входы - выход" необходимо значительно меньше нечетких правил). Для иерархической базы

знаний (рисунок 6.1), описывающую ту же зависимость, максимальное количество нечетких правил будет равным 52+53+53=275 . Причем, это "короткие" правила с двумя - тремя входными переменными.

Особенностью нечеткого логического вывода по иерархической базе знаний является отсутствие процедур дефаззификации и фаззификаци для промежуточных переменных (y_1 и y_2 на рисунке 6.1). Результат логического вывода в виде нечеткого множества напрямую передается в машину нечеткого логического вывода следующего уровня иерархии. Поэтому, для описания промежуточных переменных в иерархических нечетких базах знаний достаточно задать только терм-множества, без определения функций принадлежностей.

Общая постановка задачи

Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи согласно варианта, проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

Список индивидуальных данных

1) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи закупок (соотношения цены, качества, объема закупок и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

2) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи распределения нагрузок спортсмена (соотношение нагрузок, физического состояния, потребляемых калорий и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

3) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи управления транспортным средством (регулировка скорости с учетом передачи, погодных условий, интенсивности потока и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

4) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи управления транспортным средством (управление рулем, газом, тормозом при въезде в гараж), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

5) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования теплоснабжения (соотношение среднесуточной температуры, ветра, размера здания и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

6) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования реверсного движения на мосту (учитывать время, интенсивность потока, день недели и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

7) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора специй для блюда (соотношение количества и остроты специй, рецептуры, предпочтений едока, объема пищи и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

8) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора объема блюд (учитывать калорийность, вкусовые предпочтения, количество едоков и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

9) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подачи электроэнергии в условиях экономии (учет времени суток, типа помещений, количества людей, типа оборудования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

10) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора интенсивности занятий (учитывать начальный уровень подготовки, объем учебного материала, количество человек в группе, необходимый уровень усвоения и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

11) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи расчета потребления бензина (учитывать тип совершаемых маневров, уровень подготовки водителя, состояние автомобиля, тип автомобиля и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

12) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования системы орошения (учитывать время года, количество выпадающих озадков, вид орошаемой культуры и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

13) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи настройки аудиосистемы (мощность колонок, их количество, размер помещения, назначение установки и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

14) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора дозы снотворного (количество препарата, действие препарата, восприимчивость к выбранному препарату, цель и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

15) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи планирования объема производства продукции (с учетом возможной прибыли, необходимых ресурсов, платежеспособности населения, рынка сбыта и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

16) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования кондиционера (учитывать его мощность, объем помещения, температуру окружающей среды, необходимую температуру в помещении и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

17) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи распределения нагрузки между компьютерами при использовании их в кластерах (учитывать характеристики компьютеров, их количество, количество параллельного кода, характеристики сети и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

18) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора складского помещения (учитывать площадь склада, количество и размеры продукции, удаленность от места производства и точек реализации, свойства продукции и характеристики помещений и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

19) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора комплектующих для компьютера (учитывать цену, потребности пользователя, совместимость, сроки использования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

20) Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи определения количества линий в службе поддержки (учитывать количество обслуживаемых клиентов, среднюю частоту обращения в службу одного клиента, среднее время обслуживания одной заявки, квалификацию персонала и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

Пример выполнения работы

Рассмотрим пример того, как обрабатываются нечеткие правила вывода в экспертной системе, управляющей вентилятором комнатного кондиционера. Задача кондиционера - поддерживать оптимальную температуру воздуха в комнате, охлаждая его, когда жарко, и нагревая, когда холодно. Пусть, изменяя скорость вращения вентилятора, прогоняющего воздух через охлаждающий элемент, мы можем менять температуру воздуха, тогда алгоритм работы кондиционера может быть задан следующими правилами:

1) если температура воздуха в комнате высокая, то скорость вращения вентилятора высокая;

2) если температура воздуха в комнате средняя, то скорость вращения вентилятора средняя;

3) если температура воздуха в комнате низкая, то скорость вращения вентилятора низкая.

Для того чтобы система могла обрабатывать эти правила, надо задать функции принадлежности для нечетких подмножеств, определенных на значениях температуры (**t**) и скорости вращения вентилятора (**v**). Пусть температура воздуха в комнате находится в пределах от 0°C до 60°C - в противном случае кондиционер вряд ли поможет. Функцию принадлежности для нечеткого подмножества **низкая**, определенную на интервале изменения температуры, можно задать, например, так (рисунок 6.2):

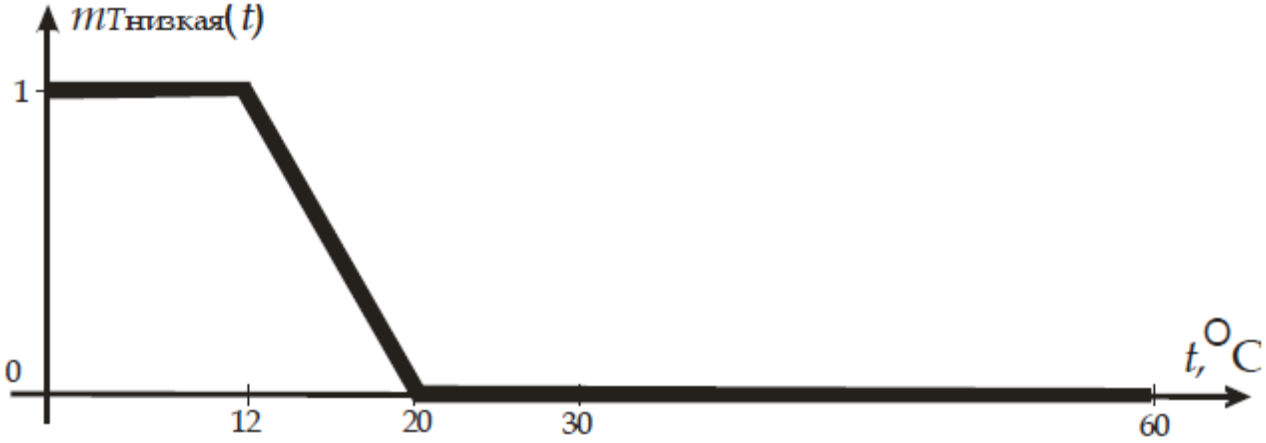


Рисунок 6.2 - Нечеткое подмножество «низкая», определенное на множестве значений температуры

Если температура меньше 12°C, то это - определенно низкая температура для комнаты ($m_{низкая}(t)=1$, t 12). Температуру выше 20°C никак нельзя назвать низкой ($m_{низкая}(t)=0$, t 20). В промежутке между этими значениями функция принадлежности линейно убывает - с увеличением температуры уменьшается истинность утверждения «температура воздуха в комнате низкая». Аналитически $m_{низкая}(t)$ выражается следующим образом:

$$m_{T\ \text{низкая}}(t)=\begin{cases}1, t\le 12\\ \dfrac{20-t}{8}, 12<t<20\\ 0, t\ge 20\end{cases}$$

Сходные рассуждения позволяют нам задать функции принадлежности для оставшихся подмножеств: **средняя** (рисунок 6.3) и **высокая** (рисунок 6.4).

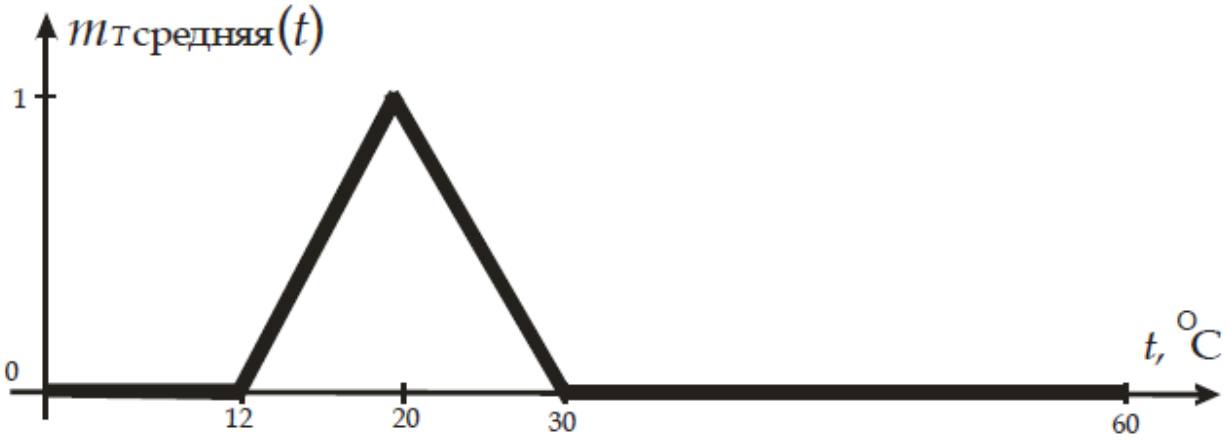


Рисунок 6.3 – Нечеткое подмножество «средняя», определенное на множестве значений температуры

$$m_{T\ \text{средняя}}(t)=\begin{cases}0, t<12 \text{ или } t\ge 30\\ \dfrac{t-12}{8}, 12\le t<20\\ \dfrac{30-t}{10}, 20\le t<30\end{cases}$$

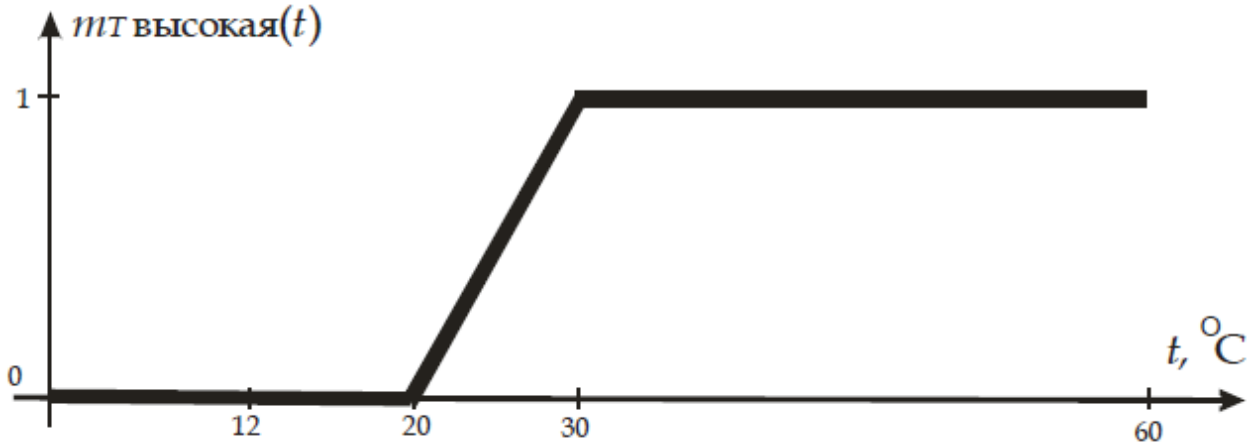


Рисунок 6.4 - Нечеткое подмножество «высокая», определенное на множестве значений температуры

$$m_{T\ \text{высокая}}(t)=\begin{cases}0, t<20\\ \dfrac{t-20}{10}, 20\le t<30\\ 1, t>30\end{cases}$$

Определим нечеткие подмножества для скорости вращения вентилятора. Пусть она может изменяться от 0 до 1000 об/мин. Вполне допустимым будет следующий вариант определения функций принадлежности для нечетких подмножеств **низкая, средняя** и **высокая** (рисунок 6.5, 6.6, 6.7).

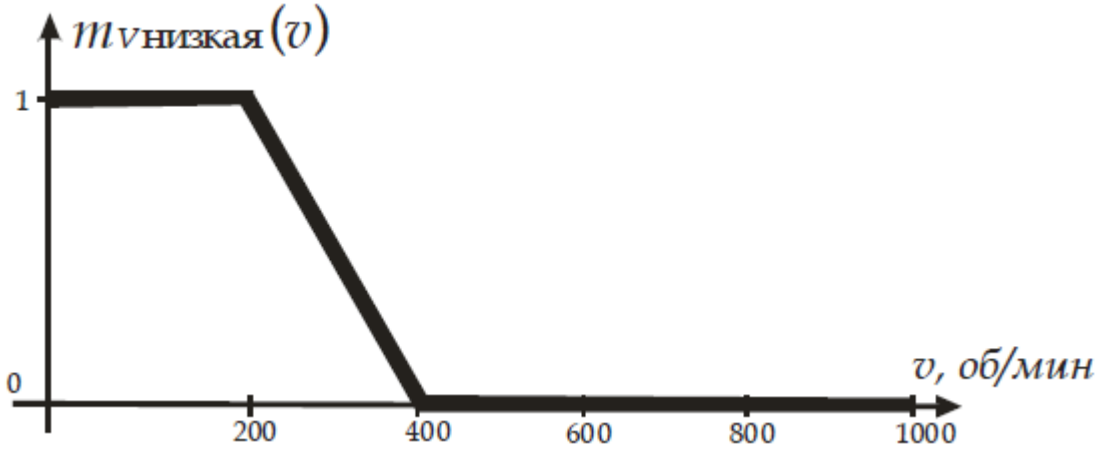


Рисунок 6.5 – Нечеткое подмножество «низкая», определенное на множестве значений скорости вращения вентилятора

$$m_{V\text{ низкая}}(v)=\begin{cases}1, v<200\\ \dfrac{400-v}{200}, 200\le v\le 400\\ 0, v>400\end{cases}$$

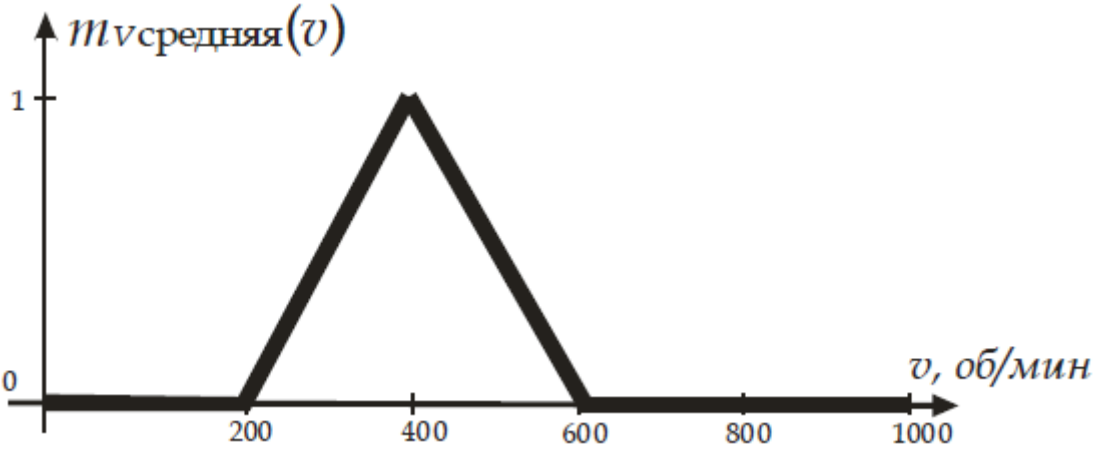


Рисунок 6.6 – Нечеткое подмножество «средняя», определенное на множестве значений скорости вращения вентилятора

$$m_{V\text{ средняя}}(v)=\begin{cases}0, v<200 \text{ или } v>600\\ \dfrac{v-200}{200}, 200\le v<400\\ \dfrac{600-v}{200}, 400\le v\le 600\end{cases}$$

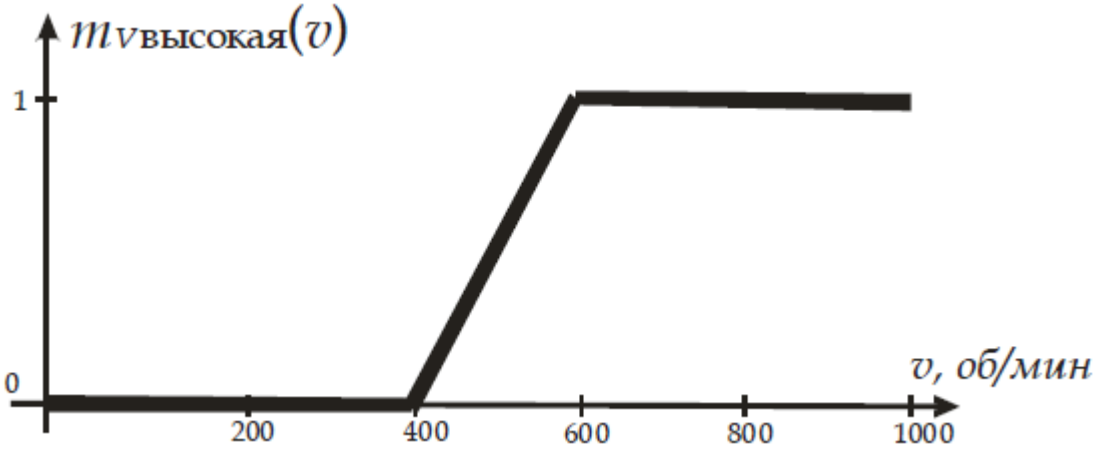


Рисунок 6.7 – Нечеткое подмножество «высокая», определенное на множестве значений скорости вращения вентилятора

$$m_{V\ \text{высокая}}(v)=\begin{cases}0, v<400\\ \dfrac{v-400}{200}, 400\le v<600\\ 1, v\ge 600\end{cases}$$

Рассмотрим теперь, как нечеткая экспертная система определяет скорость вращения вентилятора в зависимости от температуры воздуха в комнате. Пусть эта температура равна 22°С.

Сначала экспертной системе надо определить истинность левых частей правил вывода при подстановке в них текущего значения температуры. Для этого она должна найти степень вхождения t=22°С в каждое из указанных слева нечетких подмножеств. В левых частях правил указаны три подмножества, заданных на интервале значений температуры: **высокая, низкая** и **средняя**. Степень вхождения находим, вычисляя значение функций принадлежности каждого из подмножеств от t=22°С:

$$m_{T\ \text{высокая}}(22)=0.2;$$
$$m_{T\ \text{средняя}}(22)=0.8;$$
$$m_{T\ \text{низкая}}(22)=0.$$

Значения истинности левой части каждого правила используются для модификации нечеткого множества, указанного в его правой части. Модификацию будем производить описанным выше методом «произведения». На рисунке 6.8 изображено, как трансформируются находящиеся в правых частях правил нечеткие подмножества высокая, средняя и низкая.

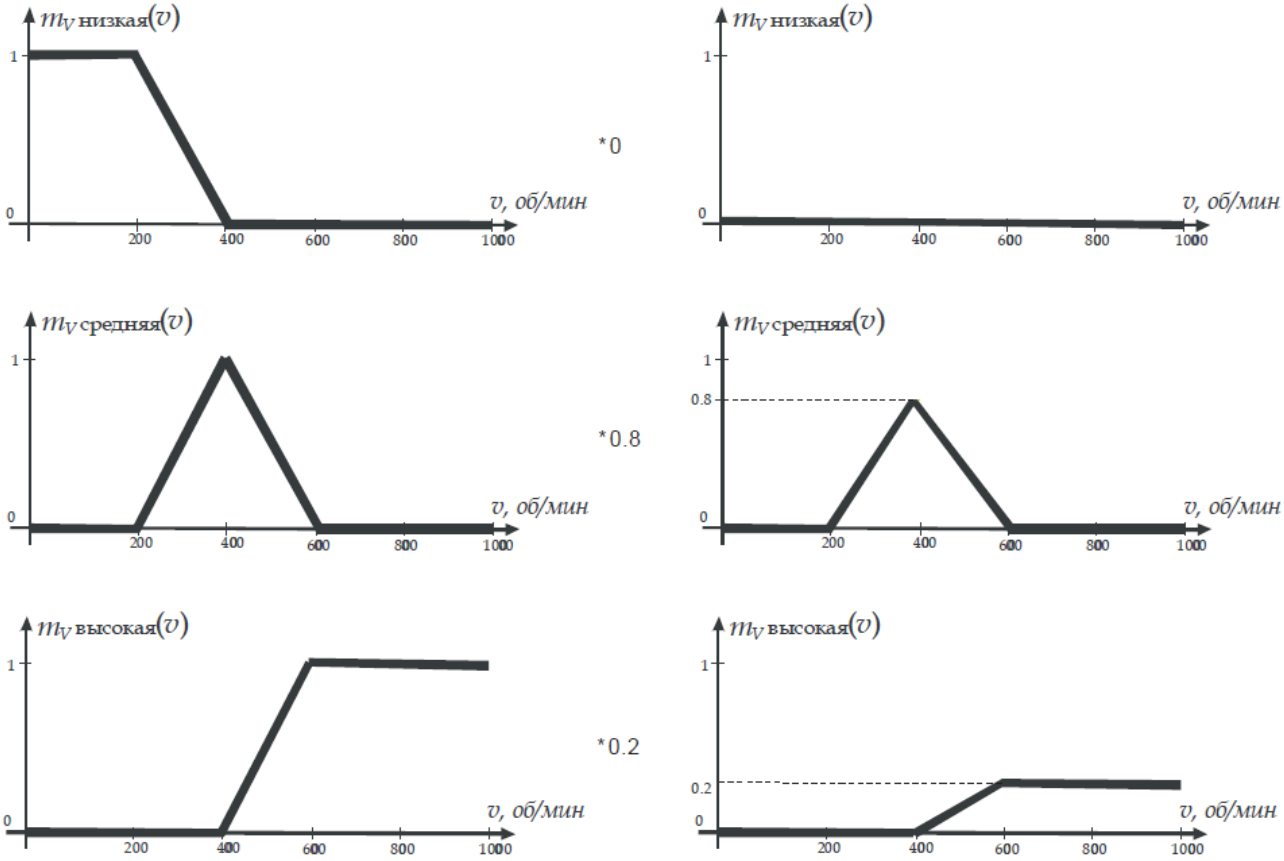


Рисунок 6.8 – Модификация нечетких подмножеств, определенных на интервале изменения скорости вращения вентилятора

Далее нечеткой экспертной системе необходимо обобщить результаты действия всех правил вывода, то есть произвести суперпозицию полученных нечетких множеств. Результат объединения нечетких множеств показан на рисунке 6.9.

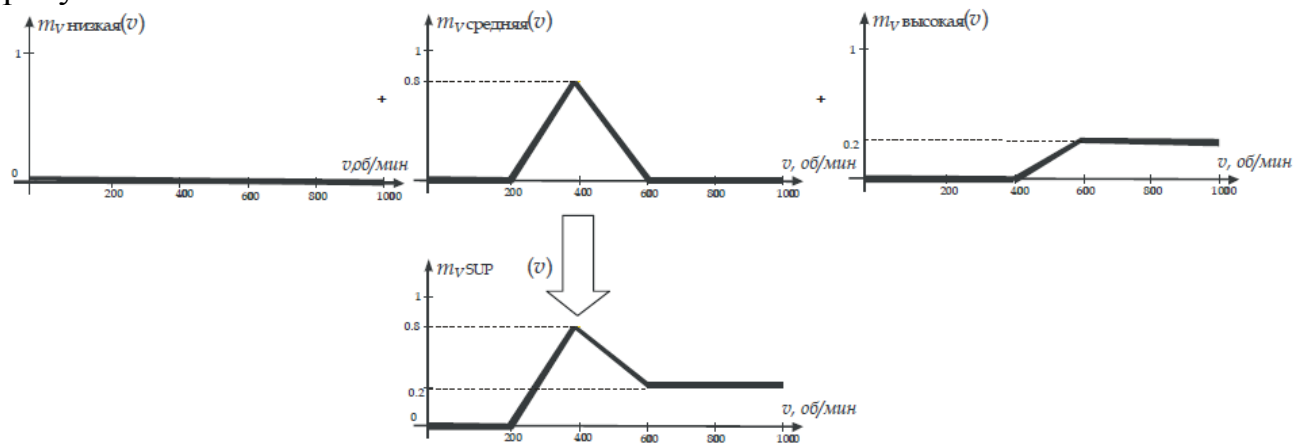


Рисунок 6.9 – Результат суперпозиции нечетких множеств

Теперь необходимо осуществить переход от суперпозиции множеств к скалярному значению. Скаляризацию произведем методом "центра тяжести". Иллюстрация того, как получается результат, представлена на рисунке 6.10.

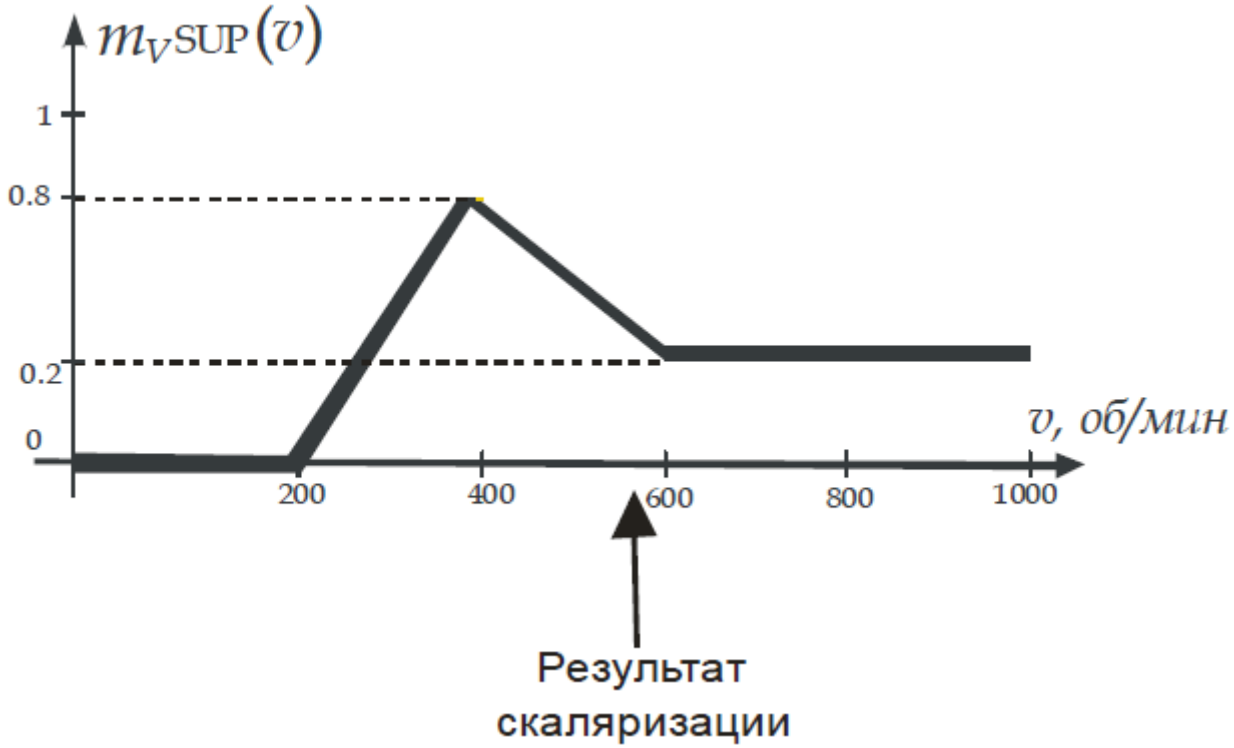


Рисунок 6.10 – Получение скалярного значения скорости вращения вентилятора методом «центра тяжести»

Центр тяжести фигуры на рисунке 6.10 находится в точке v=520. Это и будет значением скорости вращения вентилятора, которое выдаст экспертная система при температуре воздуха в комнате равной 22°С. При других значениях температуры функция принадлежности обобщенного результата выполнения всех правил, изображенная на рисунке 6.10, будет меняться.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Задние нечетких функций принадлежности. В прикладном пакете Fuzzy logic toolbox программы MATLAB создадим новый проект и зададим нечеткие функции принадлежности для температуры как входные параметры (рисунок 6.11).

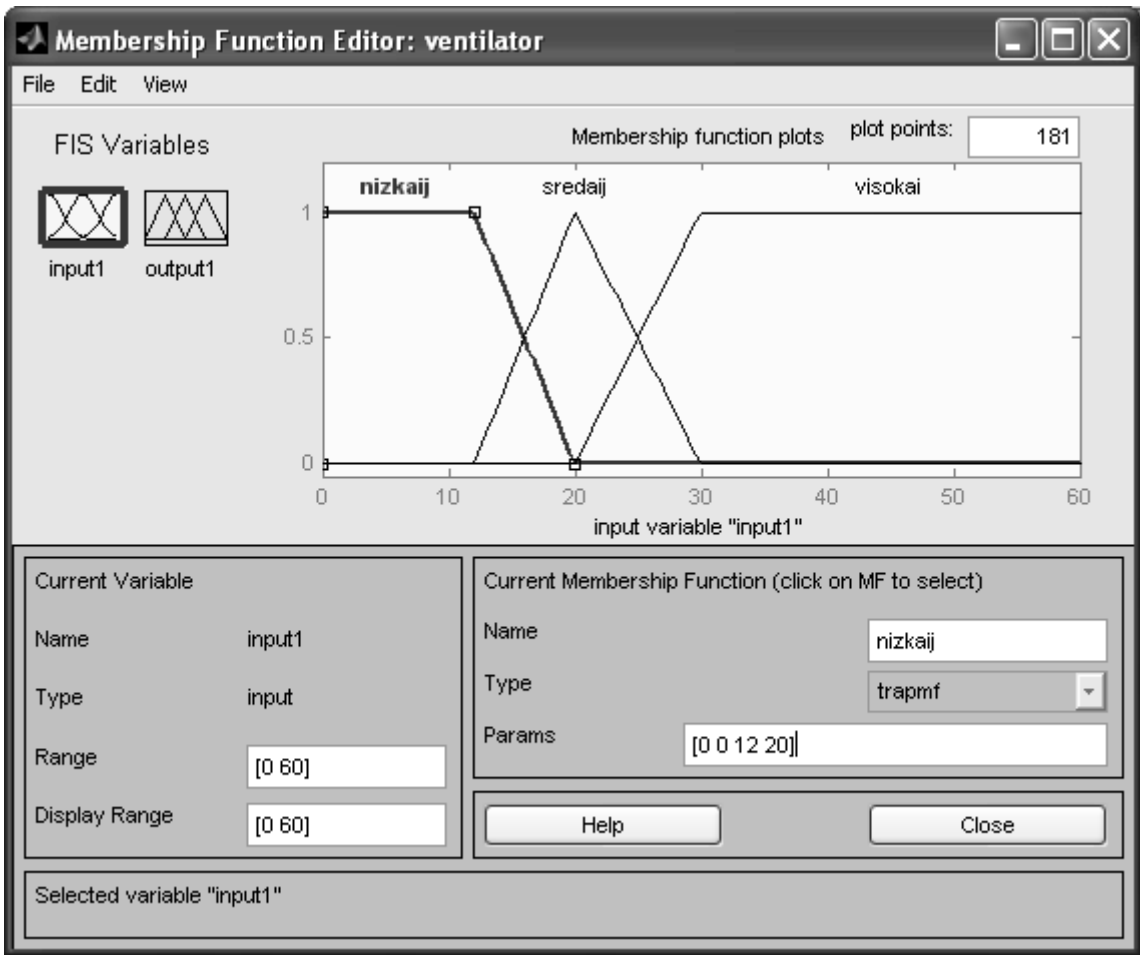


Рисунок 11 – Окно задания нечетких функций принадлежности для значений температуры

Зададим нечеткие функции принадлежности для температуры как входные параметры (рисунок 6.12).

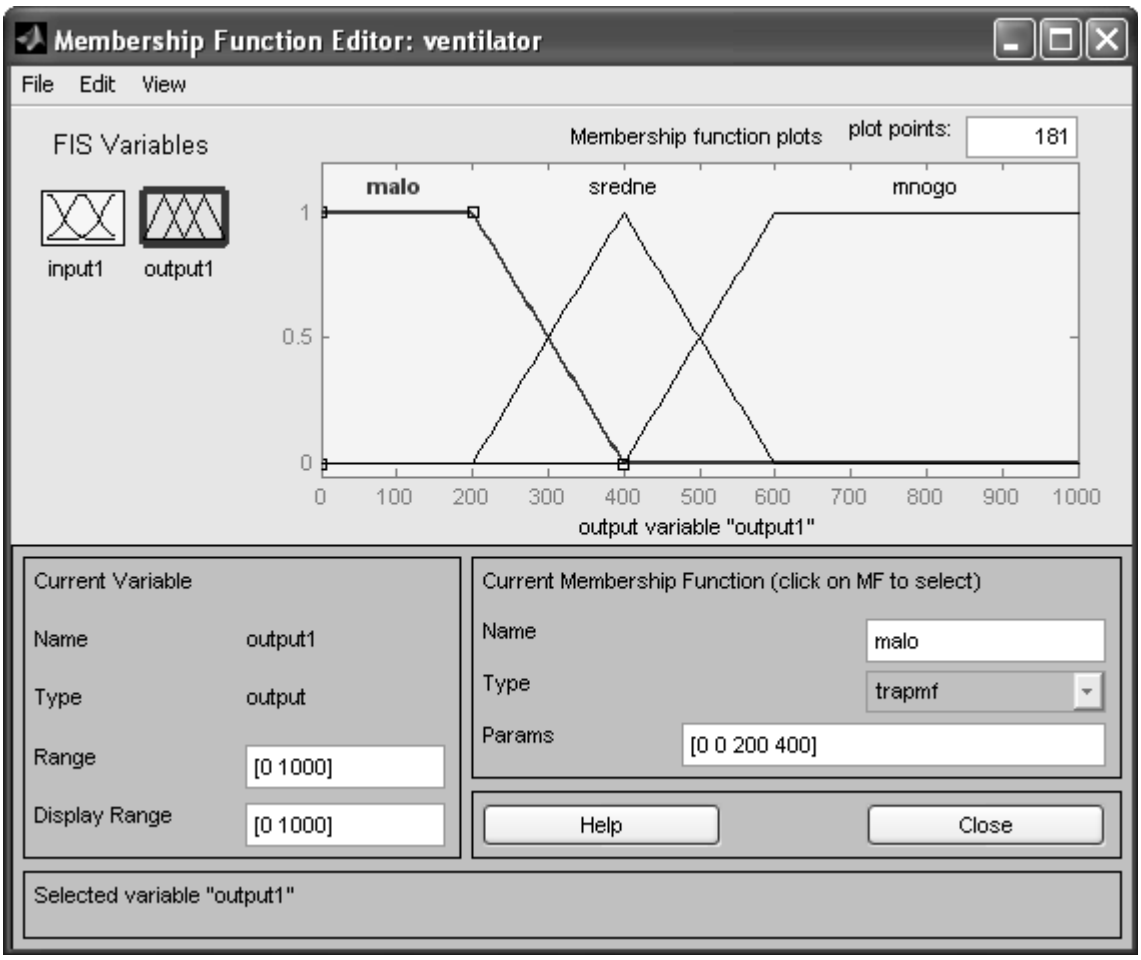


Рисунок 6.12 – Окно задания нечетких функций принадлежности для значений скорости вращения вентилятора

2. Задание правил вывода. Правила вывода в созданной компьютерной модели нечеткие правила вывода задаются при помощи вкладки **Edit→Rules**. Окно редактора после задания правил вывода будет иметь вид, представленный на рисунке 6.13.

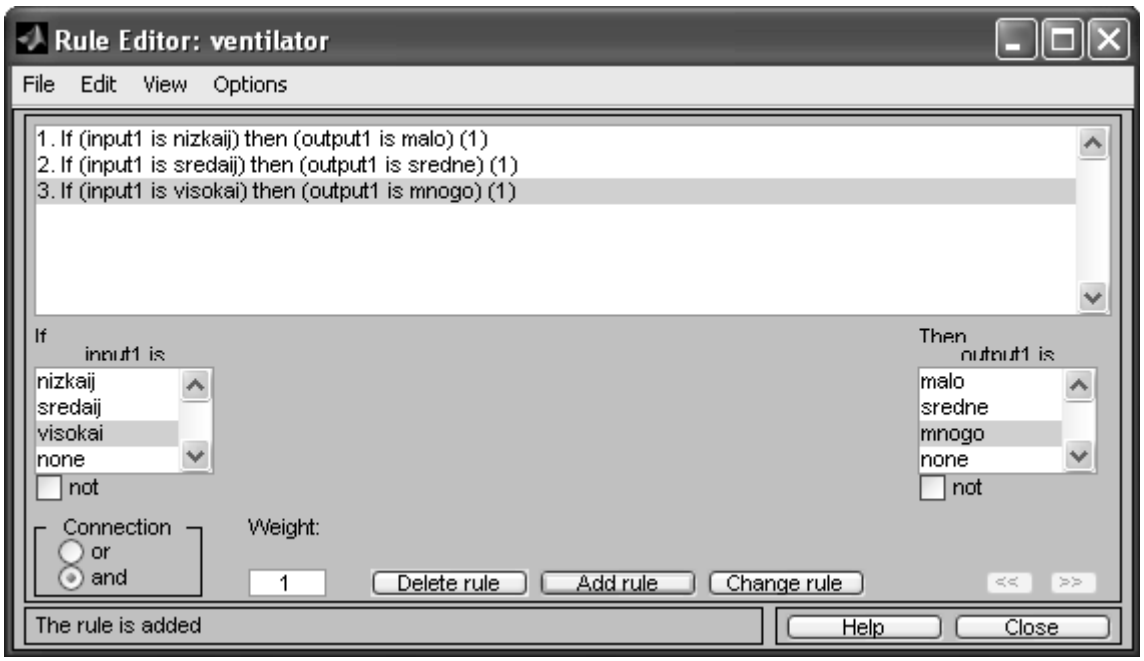


Рисунок 6.13 – Задание правил вывода

3. Получение отклика системы. Последовательность обработки нечетких для конкретного значения температуры можно просмотреть в окне просмотра правил **View→Rules**. На рисунке 6.14 представлена последовательность обработки нечетких знаний для температуры t=22 °C, при этом вентилятор кондиционера должен вращаться со скоростью v=520 об/мн.

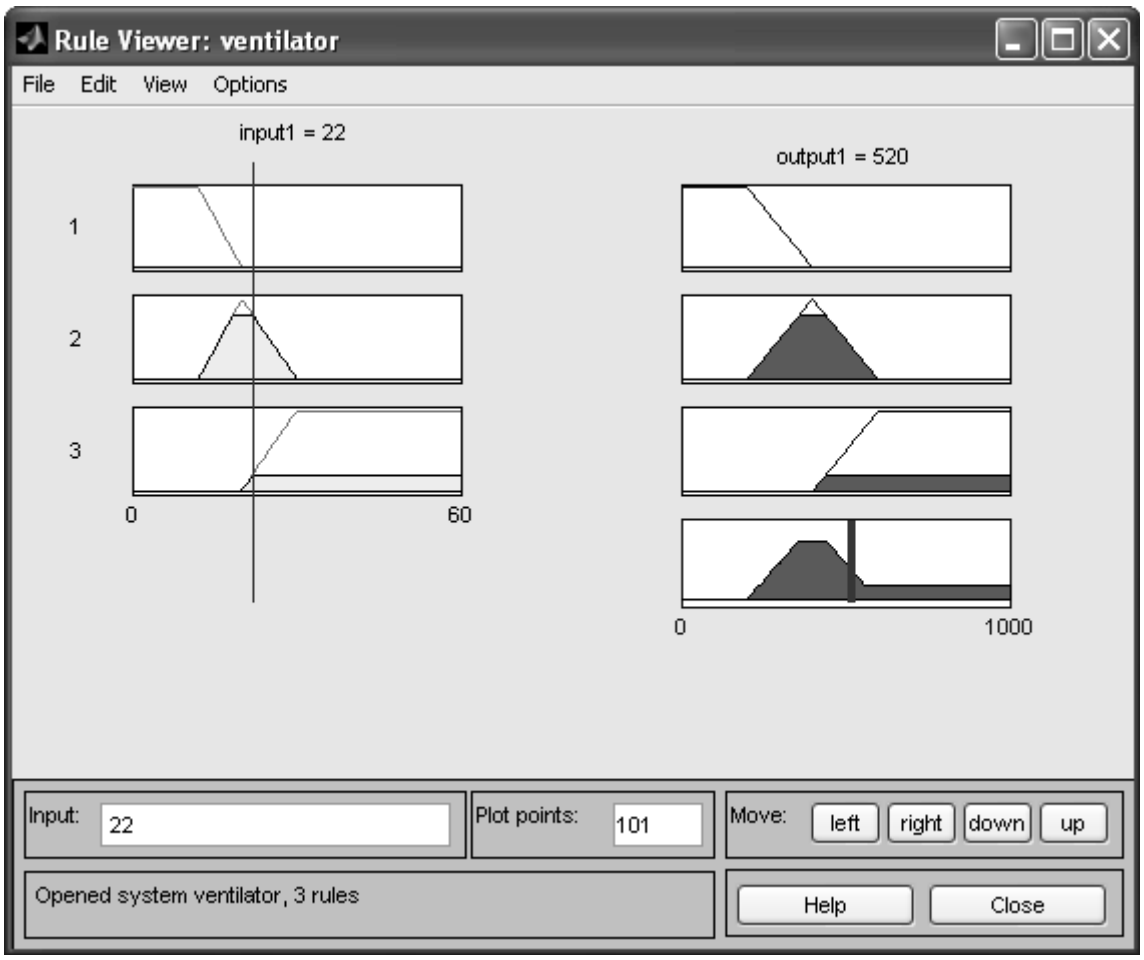


Рисунок 6.14 – Обработка нечетких знаний в экспертной системе

Передаточную характеристику системы, т.е. зависимость скорости вращения кондиционера от температуры для рассматриваемой модели можно просмотреть при помощи команды **View→Suface** (рисунок 6.15).

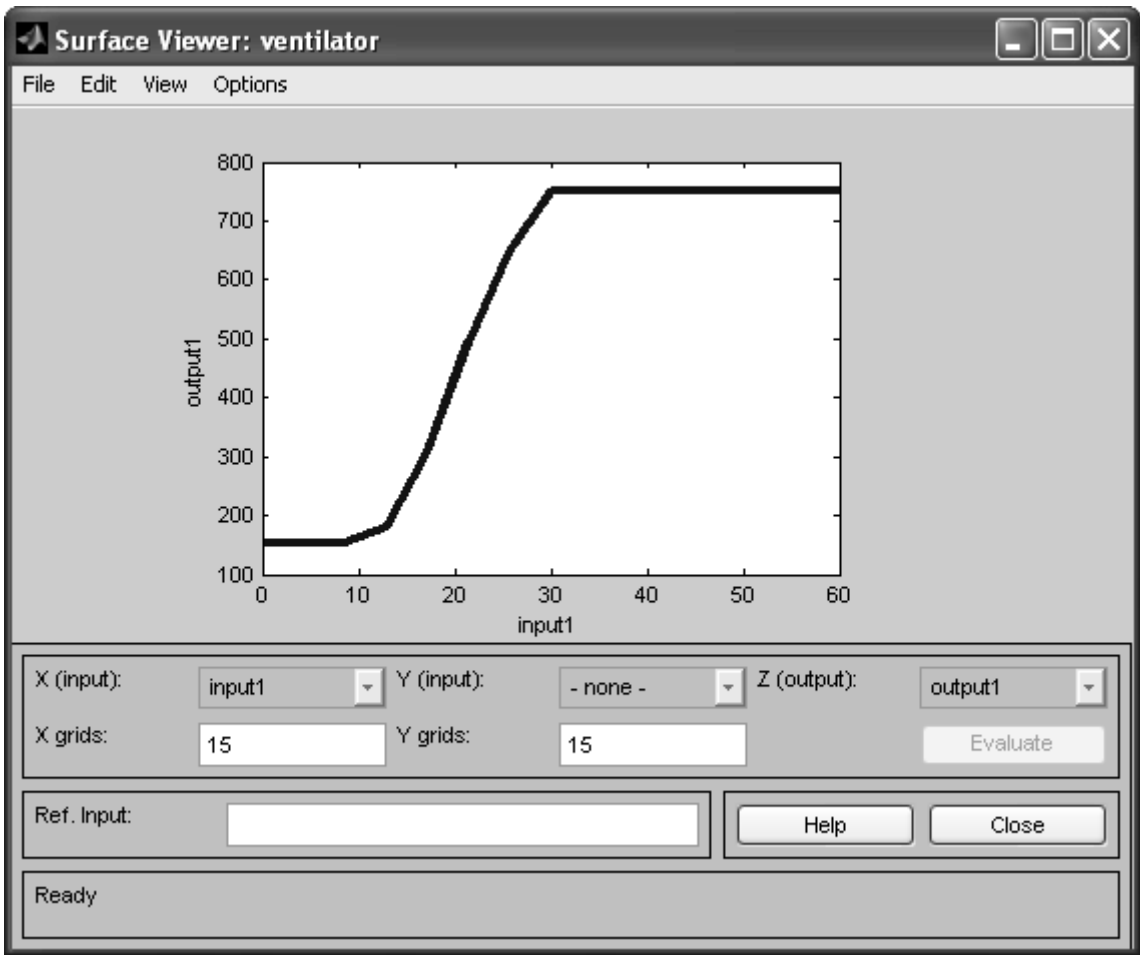


Рисунок 6.15 – Отклик системы

Контрольные вопросы к защите

1. Назовите и охарактеризуйте этапы осуществления нечеткого логического вывода?
2. Что такое фаззификация?
3. Каким набором параметров характеризуется лингвистическая переменная?
4. В чем отличие нечеткой переменной от лингвистической?
5. Назовите логические связки?
6. Приведите примеры лингвистических переменных?
7. Какие задачи необходимо решать, используя аппарат нечеткой логики?

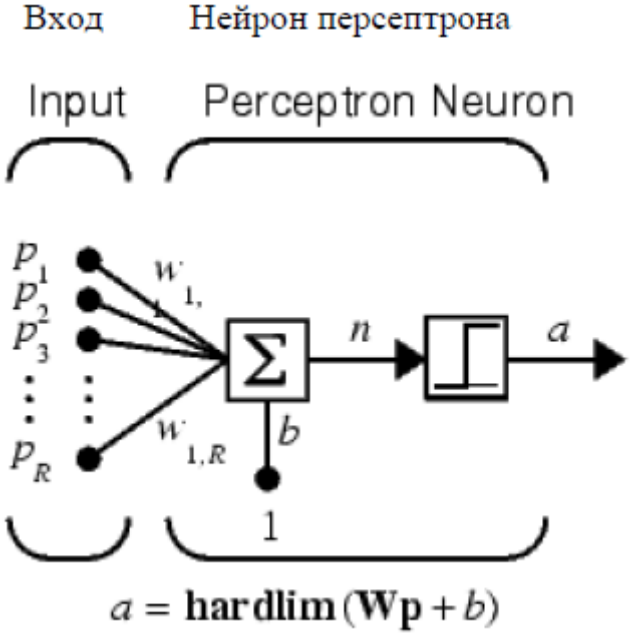
Лабораторная работа №7. Персептроны и однослойные персептронные нейронные сети

Цель работы: изучение модели нейрона персептрона и архитектуры персептронной однослойной нейронной сети; создание и исследование моделей персептронных нейронных сетей в системе MATLAB.

Теоретическая часть

Персептроном называется простейшая нейронная сеть, веса и смещения которого могут быть настроены таким образом, чтобы решить задачу классификации входных векторов. Задачи классификации позволяют решать сложные проблемы анализа коммутационных соединений, распознавания образов и других задач классификации с высоким быстродействием и гарантией правильного результата.

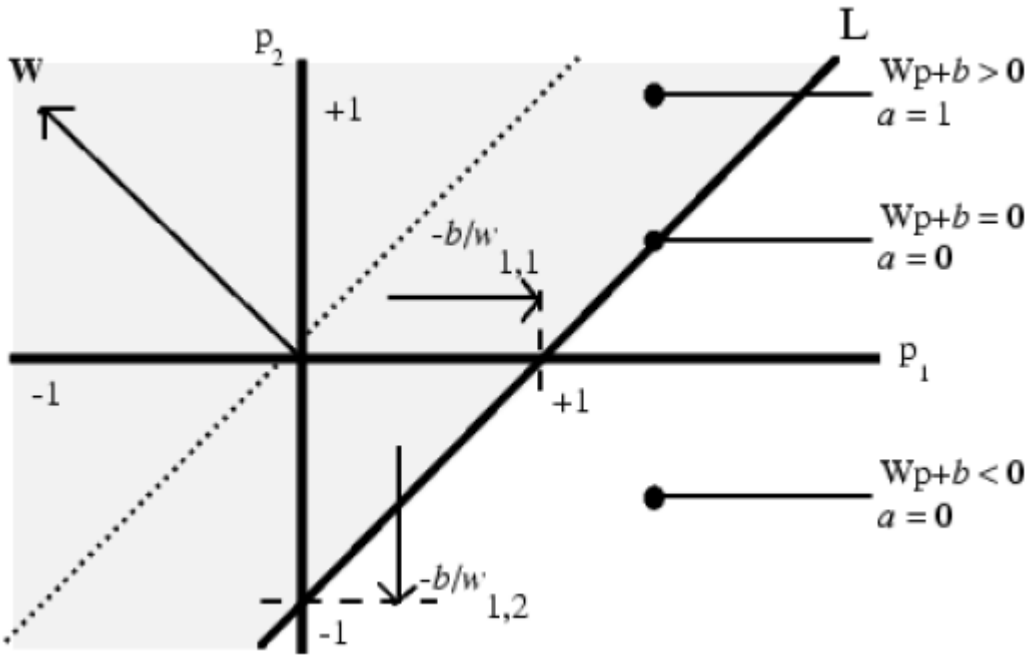
Нейрон персептрона. Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активации hardlim с жесткими ограничениями



Каждое значение элемента вектора входа персептрона умножено на соответствующий вес w_{1j}, и сумма полученных взвешенных элементов является входом функции активации.

Если вход функции активации $n \geq 0$, то нейрон персептрона возвращает 1, если $n < 0$, то 0.

Функция активации с жесткими ограничениями придает персептрону способность классифицировать векторы входа, разделяя пространство входов на две области, как это показано на рисунке ниже, для персептрона с двумя входами и смещением.



Пространство входов делится на две области разделяющей линией L, которая для двумерного случая задается уравнением:

$$W^T p + b = 0 \qquad (1)$$

Эта линия перпендикулярна к вектору весов w и смещена на величину b. Векторы входа выше линии L соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1; векторы входа ниже линии L соответствуют выходу персептрона, равному 0.

При изменении значений смещения и весов граница линии L изменяет свое положение.

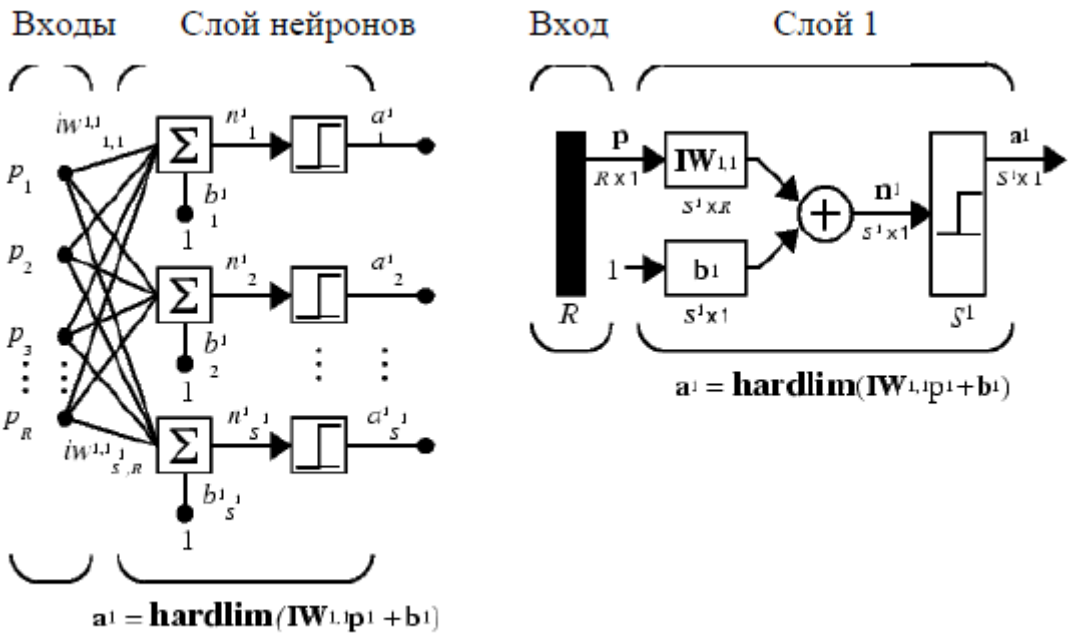
Персептрон без смещения всегда формирует разделяющую линию, проходящую через начало координат; добавление смещения формирует линию, которая не проходит через начало координат.

В случае, когда размерность вектора входа превышает 2, т. е. входной вектор P имеет более 2 элементов, разделяющей границей будет служить гиперплоскость.

Архитектура сети. Персептрон состоит из единственного слоя, включающего S нейронов, как это показано на рисунках а и б в виде развернутой и укрупненной структурных схем соответственно; веса w_{ij} – это коэффициенты передачи от j-го входа к i-му нейрону.

Уравнение однослойного персептрона имеет вид

$$a = f(Wp + b).$$



а *б*

Модель персептрона. Для формирования модели однослойного персептрона в системе MATLAB предназначена функция newp:

net = newp(PR, S)

со следующими входными аргументами: PR – массив минимальных и максимальных значений для R элементов входа размера Rx2; S – число нейронов в слое.

Например, функция **net = newp([0 2], 1)** создает персептрон с одноэлементным входом и одним нейроном; диапазон значений входа – [0 2].

В качестве функции активации персептрона по умолчанию используется функция **hardlim**.

Общая постановка задачи

Провести моделирование однослойной персептронной сети.

Список индивидуальных данных

Номер варианта	Количество входов	Диапазоны значений входов	Количество нейронов в слое
1	2	–9…+9	3
2	2	–7…+7	2
3	2	–5…+5	3
4	2	–3…+3	2
5	2	–6…+6	3
6	2	–3…+3	2
7	2	–1…+1	3
8	2	–4…+4	2
9	2	–2…+2	3
10	2	–8…+8	2

Пример выполнения работы

Описание процесса решения.

1. Для заданного варианта разработать структурную схему персептронной нейронной сети.

2. Разработать алгоритм создания и моделирования персептронной нейронной сети.

3. Реализовать разработанный алгоритм в системе MATLAB.

4. Определить параметры созданной нейронной сети (веса и смещение) и проверить правильность работы сети для последовательности входных векторов (не менее 5).

5. Построить график функции активации с указанием областей активации.

6. Переустановить значения матриц весов и смещений с помощью рассмотренных функций инициализации.

Рассмотрим однослойный персептрон с одним двухэлементным вектором входа, значения элементов которого изменяются в диапазоне от –2 до 2 (p_1 = [–2 2], p_2 = [–2 2], число нейронов в сети S = 1):

clear, net = newp([-2 2;-2 2],1); %Создание персептрона net

По умолчанию веса и смещение равны нулю, и для того, чтобы установить желаемые значения, необходимо применить следующие операторы:

net.IW{1,1} = [-1 1]; % Веса w11= -1; w12 = 1

net.b{1} = [1]; % Смещение b = 1

Запишем уравнение (1) в развернутом виде для данной сети:

$$\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} \begin{bmatrix} p_1 & p_2 \end{bmatrix} + b_1 = 0,$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} p_1 & p_2 \end{bmatrix} + 1 = 0.$$

В этом случае разделяющая линия имеет вид

L: -p1 + p2 + 1 = 0

Определим реакцию сети на входные векторы p_1 и p_2, расположенные по разные стороны от разделяющей линии:

```
p1 = [1; 1];
a1 = sim(net,p1) % Моделирование сети net с входным вектором p1
a1 =
     1
p2 = [1; -1];
a2 = sim(net,p2) % Моделирование сети net с входным вектором p2
a2 =
     0
```

Персептрон правильно классифицировал эти два вектора.

Заметим, что можно было бы ввести последовательность двух векторов в виде массива ячеек и получить результат также в виде массива ячеек

```
p3 = {[1; 1] [1; -1]}
a3 = sim(net,p3) % Моделирование сети net при входном сигнале p3
p3 =
    [2x1 double] [2x1 double]
a3 =
    [1] [0]
```

Инициализация параметров. Для однослойного персептрона в качестве параметров нейронной сети в общем случае выступают веса входов и смещения. Допустим, что создается персептрон с двухэлементным вектором входа и одним нейроном

```
clear, net = newp([-2 2;-2 2],1);
```

Запросим характеристики весов входа

```
net.inputweights{1, 1}
ans =
    delays:   0
    initFcn: 'initzero'
    learn:    1
    learnFcn: 'learnp'
    learnParam: []
    size: [1 2]
    userdata: [1x1 struct]
```

weightFcn: 'dotprod'

Из этого списка следует, что в качестве функции инициализации по умолчанию используется функция initzero, которая присваивает весам входа нулевые значения. В этом можно убедиться, если извлечь значения элементов матрицы весов и смещения:

```
wts = net.IW{1,1}, bias = net.b{1}
wts =
    0 0
bias =
    0
```

Теперь переустановим значения элементов матрицы весов и смещения:

```
net.IW{1,1} = [3, 4]; net.b{1} = 5;
wts = net.IW{1,1}, bias = net.b{1}
wts =
    3 4
bias =
    5
```

Для того чтобы вернуться к первоначальным установкам параметров персептрона, предназначена функция init:

```
net = init(net); wts = net.IW{1,1}, bias = net.b{1}
wts =
    0 0
bias =
    0
```

Можно изменить способ, каким инициализируется персептрон с помощью функции init. Для этого достаточно изменить тип функций инициализации, которые применяются для установки первоначальных значений весов входов и смещений. Например, воспользуемся функцией инициализации rands, которая устанавливает случайные значения параметров персептрона:

```
% Задать функции инициализации весов и смещений
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
% Выполнить инициализацию ранее созданной сети с новыми функциями
net = init(net);
wts = net.IW{1,1}, bias = net.b{1}
wts =
    -0.1886 0.8709
bias =
    -0.6475
```

Видно, что веса и смещения выбраны случайным образом.

Контрольные вопросы к защите

1. Что такое персептрон?
2. Что необходимо для формального описания нейрона?
3. Какого вида бывают функции активации?
4. Какую функцию выполняет функция активации?
5. Сколько может быть входов у персептрона?
6. Каким уравнение описывается персептрон?
7. Каким образом классифицируются нейронные сети?
8. Что подразумевается под слоем?

Лабораторная работа №8. Модель нейрона. Графическая визуализация вычислений в системе Matlab

Цель работы: изучение структурных схем модели нейрона и средств системы MATLAB, используемых для построения графиков функций активации нейрона.

Теоретическая часть

Элементарной ячейкой нейронной сети является нейрон. Структура нейрона с единственным скалярным входом показана на рисунок 8 1,а

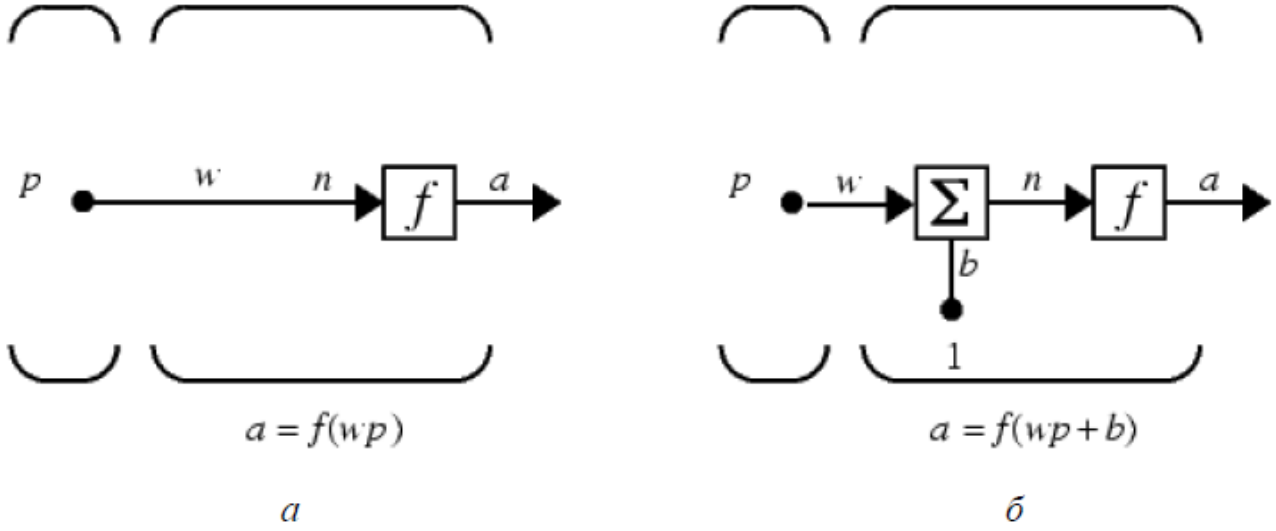


Рисунок 8.1

Скалярный входной сигнал *p* умножается на скалярный весовой коэффициент *w*, и результирующий взвешенный вход *w*p* является аргументом функции активации нейрона *f*, которая порождает скалярный выход *a*.

Нейрон, показанный на рисунок 8 1,б, дополнен скалярным смещением *b*. Смещение суммируется со взвешенным входом *w*p* и приводит к сдвигу аргумента функции *f* на величину *b*. Действие смещения можно свести к

схеме взвешивания, если представить, что нейрон имеет второй входной сигнал со значением, равным 1 (*b*1*). Вход *n* функции активации нейрона по-прежнему остается скалярным и равным сумме взвешенного входа и смещения *b*. Эта сумма (*w*p + b*1*) является аргументом функции активации *f,* а выходом функции активации является сигнал *a*. Константы *w* и *b* являются скалярными параметрами нейрона. Основной принцип работы нейронной сети состоит в настройке параметров нейрона таким образом, чтобы поведение сети соответствовало некоторому желаемому поведению. Регулируя веса и параметры смещения, можно обучить сеть выполнять конкретную работу; возможно также, что сеть сама будет корректировать свои параметры, чтобы достичь требуемого результата.

Уравнение нейрона со смещением имеет вид

$$\mathbf{a = f\,(w*p + b*1)} \qquad (1)$$

Как уже отмечалось, смещение *b* – настраиваемый скалярный параметр нейрона, который не является входом. В этом случае *b* – вес, а константа 1, которая управляет смещением, рассматривается как вход и может быть учтена в виде линейной комбинации векторов входа

$$n = \begin{bmatrix} w & b \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = w * p + b * 1 \qquad (2)$$

Нейрон с векторным входом. Нейрон с одним вектором входа *p* с *R* элементами $p_1, p_2, \ldots, p_R$ показан на рисунок 8 2. Здесь каждый элемент входа умножается на веса $w_{11}, w_{12}, \ldots, w_{1R}$ соответственно, и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора строки *W* на вектор-столбец входа *p*.

Нейрон имеет смещение *b*, которое суммируется со взвешенной суммой входов. Результирующая сумма

$$\mathbf{n = w_{11}p_1 + w_{12}\,p_2 + \ldots + w_{1R}\,p_R + b*1}$$

или

$$\mathbf{n = w_{11}p_1 + w_{12}\,p_2 + \ldots + w_{1R}\,p_R + b}$$

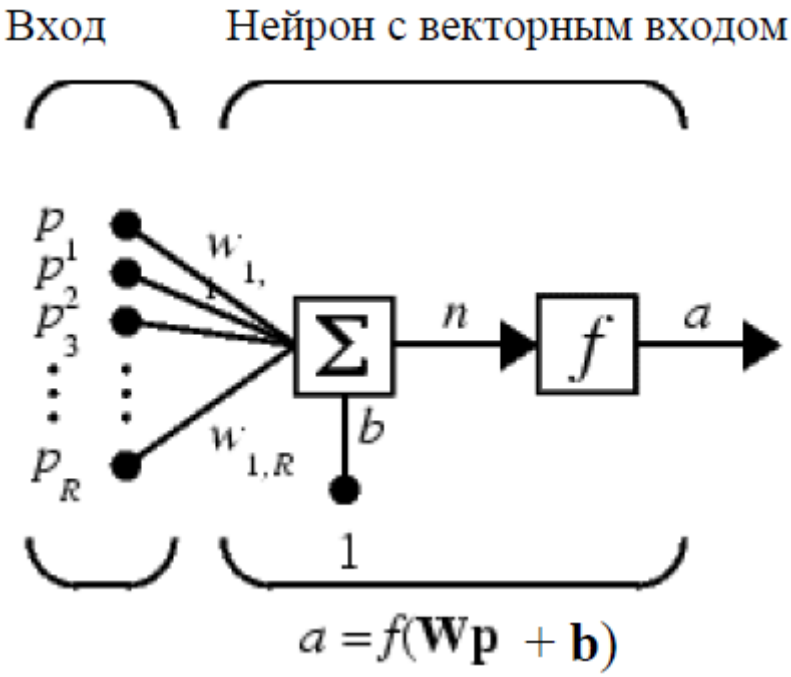


Рисунок 8.2

и служит аргументом функции активации *f*. В нотации языка MATLAB это выражение записывается так:

n = W*p + b.

Структура нейрона, показанная выше, является развернутой. При рассмотрении сетей, состоящих из большого числа нейронов, обычно используется укрупненная структурная схема нейрона (рисунок 8.3).

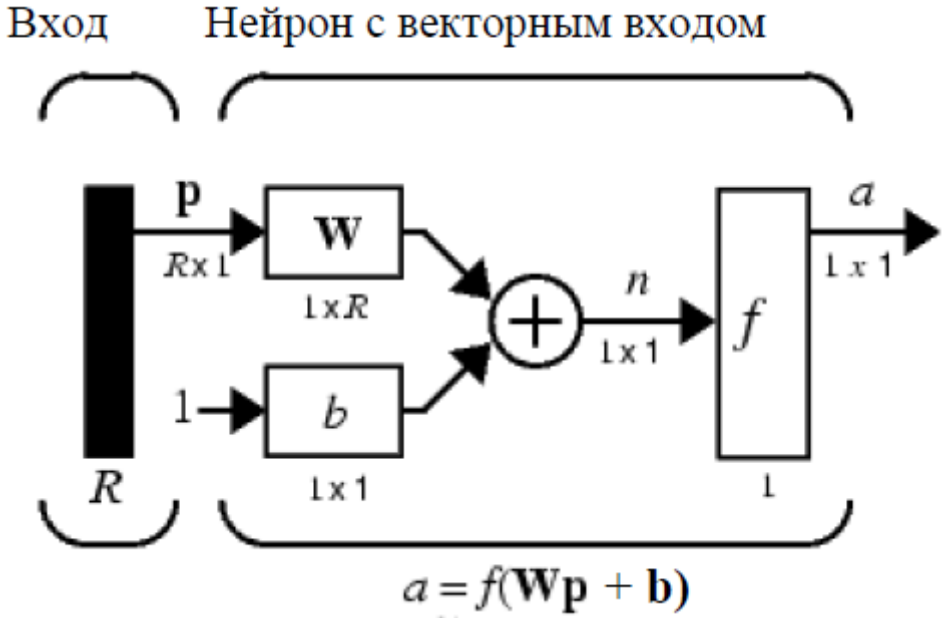


Рисунок 8.3

Вход нейрона изображается в виде темной вертикальной черты, под которой указывается количество элементов входа *R*. Размер вектора входа *p* указывается ниже символа *p* и равен *Rx1*.

Вектор входа умножается на вектор-строку *W* длины *R*. Как и прежде, константа 1 рассматривается как вход, который умножается на скалярное смещение *b*.

Входом *n* функции активации нейрона служит сумма смещения *b* и произведение *W*p*. Эта сумма преобразуется функцией активации *f*, на выходе которой получаем выход нейрона *a*, который в данном случае является скалярной величиной.

Структурная схема, приведенная на рисунке 8.3, называется слоем сети. Слой характеризуется матрицей весов *W*, смещением *b*, операциями умножения *W*p*, суммирования и функцией активации *f*. Вектор входов *p* обычно не включается в характеристики слоя.

Каждый раз, когда используется сокращенное обозначение сети, размерность матриц указывается под именами векторно-матричных переменных (рисунок 8.3). Эта система обозначений поясняет строение сети и связанную с ней матричную математику.

Функции активации. Функции активации (передаточные функции) нейрона могут иметь самый различный вид. Функция активации *f*, как правило, принадлежит к классу сигмоидальных функций, которые имеют две горизонтальные асимптоты и одну точку перегиба, с аргументом функции *n* (входом) и значением функции (выходом) *a*.

Рассмотрим три наиболее распространенные формы функции активации.

Единичная функция активации с жестким ограничением hardlim

Эта функция описывается соотношением *a = hardlim(n) = 1(n)* и показана на рисунке 8.4.

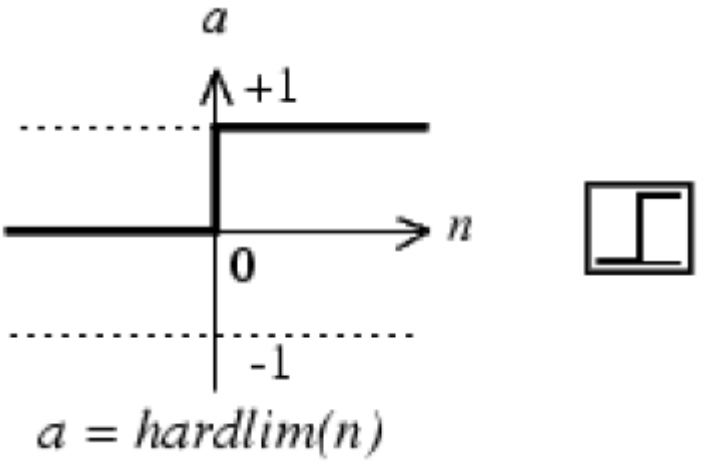


Рисунок 8.4

Она равна 0, если $n < 0$, и равна 1, если $n \geq 0$.

Чтобы построить график этой функции в диапазоне значений входа от –5 до +5, необходимо ввести следующие операторы языка MATLAB в командном окне:

n = -5:0.1:5;

plot(n,hardlim(n),'b+:');

Линейная функция активации purelin

Эта функция описывается соотношением *a = purelin(n) = n* и показана на рисунке 8.5.

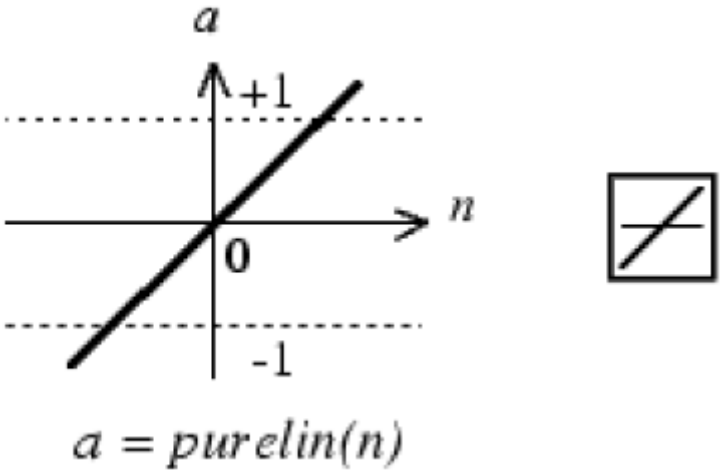


Рисунок 8.5

Чтобы построить график этой функции в диапазоне значений входа от –5 до +5, необходимо ввести следующие операторы языка MATLAB в командном окне:

n=-5:0.1:5;

plot(n,purelin(n),'b+:');

Логистическая функция активации logsig

Эта функция описывается соотношением *a = logsig(n) = = 1/(1 + exp(-n))* и показана на рисунке 8.6.

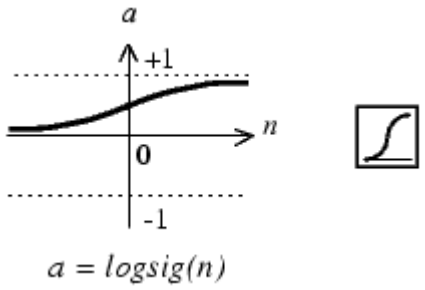


Рисунок 8.6

Данная функция принадлежит к классу сигмоидальных функций, и ее аргумент может принимать любое значение в диапазоне от –∞ до +∞, а выход изменяется в диапазоне от 0 до 1. Благодаря свойству дифференцируемости (нет точек разрыва) эта функция часто используется в сетях с обучением на основе метода обратного распространения ошибки.

Чтобы построить график этой функции в диапазоне значений входа от –5 до +5, необходимо ввести следующие операторы языка MATLAB в командном окне:

n=-5:0.1:5;

plot(n,logsig(n),'b+:');

На укрупненной структурной схеме для обозначения типа функции активации применяются специальные графические символы; некоторые из них приведены на рисунке 8.7, где а – ступенчатая, б – линейная, в – логистическая.

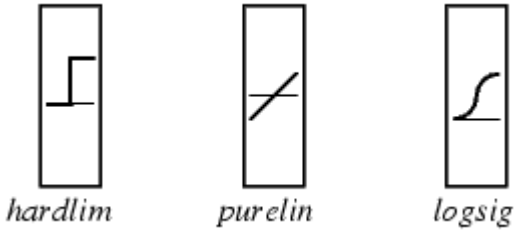


Рисунок 8.7

Общая постановка задачи

1. Построить графики функций активации в заданных диапазонах значений в соответствии с вариантом (таблица), используя функцию *plot*.

2. Используя функцию *plot*, построить графики всех заданных функций, согласно варианту, в одном графическом окне.

Список индивидуальных данных

Номер варианта	Диапазоны значений входа	Имя функции
1	–3…+3	hardlim
2	–1…+1	hardlims
3	–4…+4	purelin
4	–2…+2	poslin
5	–8…+8	satlin
6	–9…+9	satlins
7	–7…+7	radbas
8	–5…+5	tribas
9	–3…+3	logsig
10	–6…+6	tansig

Пример выполнения работы

Для построения графика функции одной переменной в системе MATLAB используется оператор *plot*. При этом графики строятся в

отдельных масштабируемых и перемещаемых окнах. Например, для построения графика функции *sin x* достаточно вначале задать диапазон и шаг изменения аргумента, а затем использовать оператор *plot* (рисунок 8.8):

```
x=-5:0.1:5;
plot(x,sin(x))
```

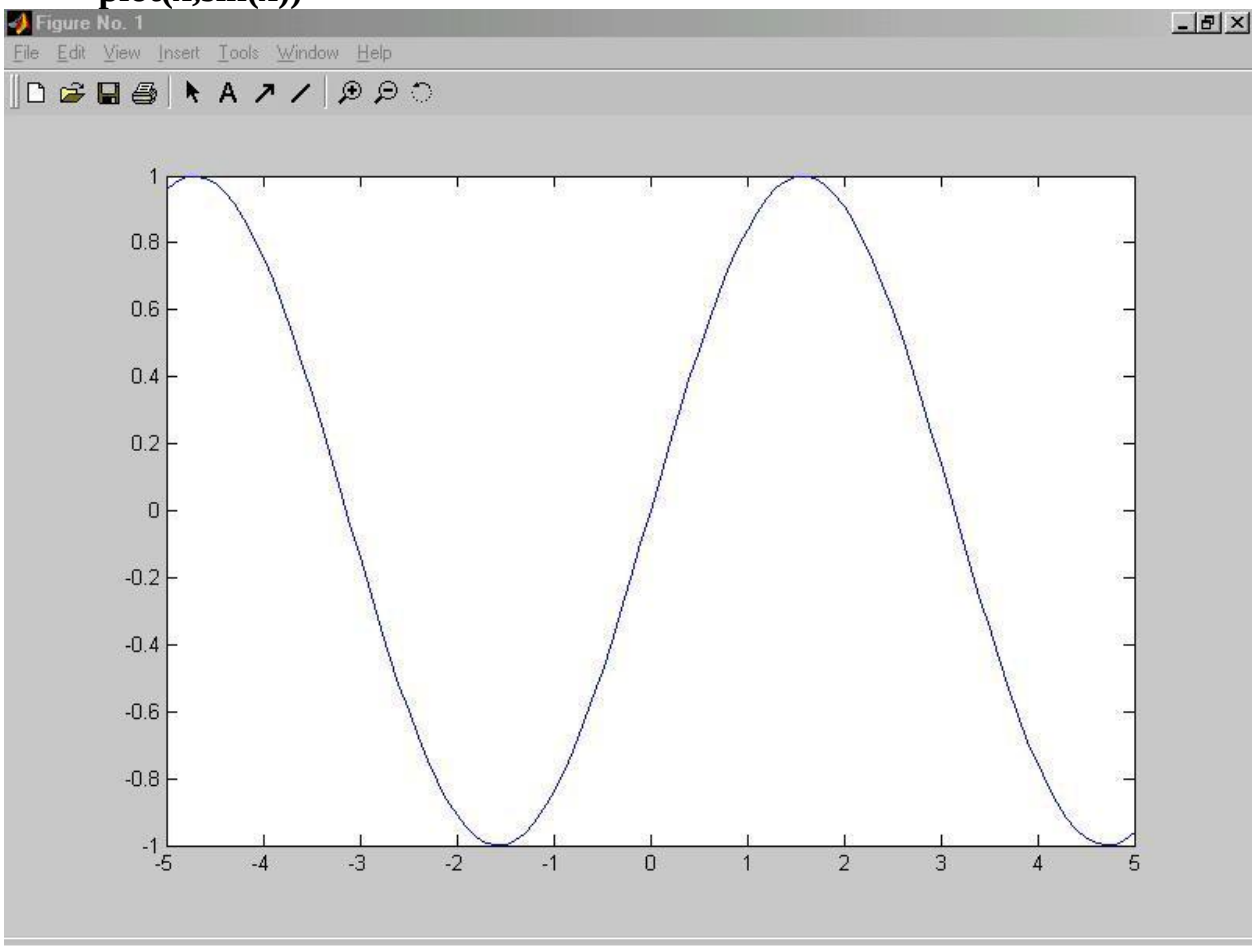


Рисунок 8.8

Оператор *plot* является мощным инструментом для построения графиков функций одной переменной. Он позволяет строить графики сразу нескольких функций и имеет различные формы, синтаксис которых можно узнать, воспользовавшись командой *help plot*.

Контрольные вопросы к защите

1. Что такое функция активации?
2. Для чего предназначен оператор plot?
3. Для чего предназначен оператор *help plot*?
4. Как строится логистическая функция активации?
5. Как строится линейная функция активации?
6. Как строится единичная функция активации?
7. Почему наиболее часто используется логистическая функция активации?

Лабораторная работа №9. Процедура настройки параметров персептронных нейронных сетей. Правила настройки

Цель работы: изучение процедуры настройки параметров персептронных нейронных сетей и реализация правил настройки в системе MATLAB.

Теоретическая часть

Определим процесс обучения персептрона как процедуру настройки весов и смещений с целью уменьшить разность между желаемым (целевым) и истинным сигналами на его выходе, используя некоторое правило настройки (обучения). Процедуры обучения делятся на 2 класса: с учителем и без учителя.

При обучении с учителем задается множество примеров требуемого поведения сети, которое называется обучающим множеством

$$\{p_1, t_1\}, \{p_2, t_2\},\ldots,\{p_Q, t_Q\} \qquad (1)$$

Здесь p_1, p_2, ..., p_Q – входы персептрона, а t_1, t_2 ,..., t_Q – требуемые (целевые) выходы.

При подаче входных сигналов выходы персептрона сравниваются с целями. Правило обучения используется для настройки весов и смещений персептрона так, чтобы приблизить значения выхода к целевому значению. Алгоритмы, использующие такие правила обучения, называются алгоритмами обучения с учителем. Для их успешной реализации необходимы эксперты, которые должны предварительно сформировать обучающие множества. Разработка таких алгоритмов рассматривается как первый шаг в создании систем искусственного интеллекта.

В этой связи ученые не прекращают спора на тему, можно ли считать алгоритмы обучения с учителем естественными и свойственными природе, или они созданы искусственно. Например, обучение человека, на первый взгляд, происходит без учителя: на зрительные, слуховые, тактильные и прочие рецепторы поступает информация извне, и внутри мозга происходит некая самоорганизация. Однако нельзя отрицать и того, что в жизни человека немало учителей – и в буквальном, и в переносном смысле, которые координируют реакции на внешние воздействия. Вместе с тем как бы ни развивался спор приверженцев этих двух концепций обучения, представляется, что обе они имеют право на существование. И рассматриваемое нами правило обучения персептрона относится к правилу обучения с учителем.

При обучении без учителя веса и смещения изменяются только в связи с изменениями входов сети. В этом случае целевые выходы в явном виде не задаются. Главная черта, делающая обучение без учителя привлекательным,

– это его самоорганизация, обусловленная, как правило, использованием обратных связей. Что касается процесса настройки параметров сети, то он организуется с использованием одних и тех же процедур. Большинство алгоритмов обучения без учителя применяется при решении задач кластеризации данных, когда необходимо разделить входы на конечное число классов.

Общая постановка задачи

1. Для заданного преподавателем варианта задания (таблица) выполнить ручной расчет настройки весов и смещений персептронной нейронной сети.
2. Разработать алгоритм создания и моделирования персептронной нейронной сети.
3. Реализовать разработанный алгоритм в системе MATLAB.
4. Определить параметры созданной нейронной сети (веса и смещение) и проверить правильность работы сети для последовательности входных векторов (не менее 5).
5. Сравнить результаты ручных расчетов и расчетов, выполненных в системе MATLAB.

Список индивидуальных данных

Номер варианта	Количество входов – 2; количество нейронов – 1		
	Диапазоны значений входов	Входы персептрона	Целевые выходы
1	–4…+4	{[–3; 1] [2; –1] [2; 2] [3; –1]}	{1 1 0 0}
2	–3…+3	{[–2; –1] [1; –1] [0; 1] [2; 0]}	{1 0 1 0}
3	–2…+2	{[0; 0] [1; 1] [–1; 1] [–1; 0]}	{0 0 1 1}
4	–4…+4	{[–2; 2] [1; 2] [0; 0] [3; –2]}	{0 1 0 1}
5	–3…+3	{[–1; 1] [–2; –1] [1; –2] [2; 0]}	{1 0 0 1}
6	–2…+2	{[0; 0] [–1; 1] [–1; 0] [1; 1]}	{0 1 1 0}
7	–4…+4	{[–2; 1] [1; –2] [3; –1] [2; 2]}	{0 0 0 1}
8	–3…+3	{[–2; 1] [0; 1] [2; –1] [–2; –1]}	{0 0 1 0}
9	–2…+2	{[–1; –1] [0; 0] [1; –1] [1; 1]}	{1 1 1 0}
10	–4…+4	{[0; 0] [2; –2] [1; –2] [–2; –1]}	{0 1 0 0}

Пример выполнения работы

Правила настройки персептронных нейронных сетей. Настройка параметров (обучение) персептрона осуществляется с использованием обучающего множества. Обозначим через p вектор входов персептрона, а через t – вектор соответствующих желаемых выходов. Цель обучения –

уменьшить погрешность $e = a - t$, которая равна разности между реакцией нейрона a и вектором цели t.

Правило настройки (обучения) персептрона должно зависеть от величины погрешности e. Вектор цели t может включать только значения 0 и 1, поскольку персептрон с функцией активации *hardlim* может генерировать только такие значения.

При настройке параметров персептрона без смещения и с единственным нейроном возможны только три ситуации:

1. Для данного вектора входа выход персептрона правильный ($a = t$ и $e = t - a = 0$), и тогда вектор весов w не претерпевает изменений.

2. Выход персептрона равен 0, а должен быть равен 1 ($a = 0, t = 1$ и $e = t - 0 = 1$). В этом случае вход функции активации $w^T p$ отрицательный и его необходимо скорректировать. Добавим к вектору весов w вектор входа p , и тогда произведение $(w^T + p^T)\, p = w^T\, p + p^T\, p$ изменится на положительную величину, а после нескольких таких шагов вход функции активации станет положительным и вектор входа будет классифицирован правильно. При этом изменяется настройка весов.

3. Выход персептрона равен 1, а должен быть равен 0 ($a = 1, t = 0$ и $e = t - 0 = -1$). В этом случае вход функции активации $w^T p$ положительный и его необходимо скорректировать. Вычтем из вектора весов w вектор входа p, и тогда произведение $(w^T - p^T)\, p = w^T\, p - p^T\, p$ изменится на отрицательную величину, а после нескольких таких шагов вход функции активации станет отрицательным и вектор входа будет классифицирован правильно. При этом изменяется настройка весов.

Теперь правило настройки (обучения) персептрона можно записать, связав изменение вектора весов Δw с погрешностью $e = t - a$:

$$\Delta \mathbf{w} = \begin{cases} 0, \text{ если } e = 0; \\ \mathbf{p}^{\mathrm{T}}, \text{ если } e = 1; \\ -\mathbf{p}^{\mathrm{T}}, \text{ если } e = -1. \end{cases}$$

Все три случая можно описать одним соотношением

$$\mathbf{\Delta w = (t - a)\, p^T = e\, p^T.}$$

Можно получить аналогичное выражение для изменения смещения, учитывая, что смещение можно рассматривать как вес для единичного входа:

$$\mathbf{\Delta b = (t - a)\, 1 = e.}$$

В случае нескольких нейронов эти соотношения обобщаются следующим образом:

$$\begin{cases} \Delta \mathbf{W}^{\mathrm{T}} = (\mathbf{t} - \mathbf{a})\mathbf{p}^{\mathrm{T}}; \\ \Delta \mathbf{b} = (\mathbf{t} - \mathbf{a}) = \mathbf{e}. \end{cases}$$

Тогда правило настройки (обучения) персептрона можно записать в следующей форме:

$$\begin{cases} \mathbf{W}^{\mathrm{T}new} = \mathbf{W}^{\mathrm{T}old} + \mathbf{e}\mathbf{p}^{\mathrm{T}}; \\ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}. \end{cases}$$

Описанные соотношения положены в основу алгоритма настройки параметров персептрона, который реализован в ППП Neural Network Toolbox в виде функции *learnp*. Каждый раз при выполнении функции *learnp* будет происходить перенастройка параметров персептрона, и, если решение существует, то процесс обучения персептрона сходится за конечное число итераций. Если смещение не используется, то функция *learnp* ищет решение, изменяя только вектор весов *w*. Это приводит к нахождению разделяющей линии, перпендикулярной вектору *w*, которая должным образом разделяет векторы входа.

Рассмотрим простой пример персептрона с единственным нейроном и двухэлементным вектором входа

clear, net = newp([-2 2;-2 2],1);

Определим смещение *b* равным 0, а вектор весов *w* равным [1 –0.8]

net.b{1} = 0;

w = [1 -0,8]; net.IW{1,1} = w;

Обучающее множество зададим следующим образом:

p = [1; 2]; t = [1];

Моделируя персептрон, рассчитаем выход и ошибку на первом шаге настройки:

a = sim(net,p), e = t-a

a =

0

e =

1

Используя функцию настройки параметров *learnp*, найдем требуемое изменение весов:

dw = learnp(w,p,[ ],[ ],[ ],[ ],e,[ ],[ ],[ ])

dw =

1 2

Тогда новый вектор весов примет вид

w = w + dw

w =

2.0000 1.2000

Описанные выше правило и алгоритм настройки (обучения) персептрона гарантируют сходимость за конечное число шагов для всех задач, которые могут быть решены с использованием персептрона. Это в

первую очередь задачи классификации векторов, которые относятся к классу линейно отделимых, когда все пространство входов можно разделить на две области некоторой прямой линией, в многомерном случае – гиперплоскостью.

Контрольные вопросы к защите

1. Каким образом осуществляется процесс обучения персетрона?
2. Опишите процесс обучения персептрона с учителем?
3. Опишите процесс обучения персептрона без учителя?
4. Для чего предназначена функция learnp?
5. Какую роль в процессе обучения играет ошибка?

Литература

1. Галушкин А.И. Нейронные сети : основы теории : для научных работников, аспирантов и студентов / Галушкин Александр Иванович ; рец.: Ю.В. Гуляев, Э.Д. Аведьян. - Москва: Горячая линия-Телеком, 2010. - 496 с.
2. Донской Д.А. Моделирование искусственных нейронных сетей в системме MatLab: часть первая / Д. А. Донской, Б. Д. Шашков, Д. М. Деревянчук, Н.В. Деревянчук, Ю. Г. Квятковский. – Пенза: ПГУ, 2005. – 36с.
3. Избранные главы теории нечетких множеств : учеб. пособие / В. И. Ухоботов. Челябинск : Челяб. гос. ун-т, 2011. 245 с.
4. Кизим Н.А. Нейронные сети : теория и практика применения : монография / Кизим Николай Александрович, Ястремская Елена Николаевна, Сенчуков Виктор Фёдорович ; рец.: З.Я Заруба, Т.С. Клебанова, Я.Г. Берсуцкий; Национальная академия наук Украины, научно-исследовательский центр индустриальных проблем развития. - Харьков: ИНЖЭК, 2006. - 234 с.
5. Круглов В.В. Нечеткая логика и искусственные нейронные сети: Учебные пособия для студентов вузов, обучающихся по специальности "Прикладная информатика (по областям)" / В.В. Круглов, М.И. Дли, Р.Ю. Голунов; Рец.: В.П. Дьяконов, В.В. Борисов. - Москва: Физматлит, 2001. - 224 с.
6. Макеев А.В. Основы нечеткой логики. Учебное пособие для вузов. – Н.Новгород:ВГИПУ, 2009. – 59с.
7. Портал искусственного интеллекта [Электронный ресурс] – http://www.aiportal.ru/.
8. Редько В.Г. Эволюция, нейронные сети, интеллект: модели и концепции эволюционной кибернетики / В.Г. Редько. - 8-е изд. - Москва: ЛИБРОКОМ, 2013. - 220 с.
9. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы = Sieci neuronowe, algorytmy genetyczne i systemy rozmyte / Д. Рутковская, М. Пилиньский, Л. Рутковский; пер. с польск. И.Д. Рудинского. - 2-е изд. - Москва: Горячая линия - Телеком, 2013. - 383 с.
10. Центр компетенций MathWorks [Электронный ресурс] - http://matlab.ru/