Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

Днепродзержинский государственный технический университет

«Петрозаводский государственный университет»
Кольский филиал

«Российский химико-технологический университет им. Д.И. Менделеева»
Новомосковский институт

В.Н. БОГАТИКОВ, Л.В. ДРАНИШНИКОВ, А.Е. ПРОРОКОВ

ПОСТРОЕНИЯ СИСТЕМ УПРАВЛЕНИЯ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

Апатиты
2011

УДК 681.5
ББК 32.965в6
Б 73

Авторы:
Богатиков В. Н., заведующий кафедрой автоматики и электропривода КФ Петр ГУ
Дранишников Л. В., профессор кафедры программного обеспечения Днепродзержинского государственного технического университета
Пророков А. Е., заведующий кафедрой прикладной информатики Новомосковского института Российского химико-технологического университета им. Д. И. Менделеева

Рецензент:
д-р техн. наук, главный научный сотрудник ИИММ КНЦ РАН Горохов А.В.

Богатиков, В.Н. Построение систем управления на основе нейронных сетей: Учебно–методическое пособие / **В.Н. Богатиков, Л.В. Дранишников, А.Е. Пророков. -** Апатиты: Изд-во КФ ПетрГУ, 2011. – 41 с.

Рассматривается методология построение систем управления на основе нейронных сетей. Описывается структура искусственного нейрона, сложившаяся классификация нейронных сетей, алгоритмы обучения нейросети. Показано использование пакета Simulink в среде MatLab. Приводятся различные примеры использования нейросетей в системах управления.

Предназначено для студентов направлений бакалавриата (профилей) – «Информатика и вычислительная техника», «Информационные системы» и «Электроэнергетика и электротехника» (профиль – «Электропривод и автоматика»), а также специальностей – «Автоматизированные системы обработки информации и управления», «Информационные системы и технологии» и «Электропривод и автоматика промышленных установок и комплексов».

Учебно-методическое пособие полезно лицам занимающиеся разработкой систем управления.

Табл. 44. Ил. 39. Библиогр.: 11 назв.

УДК 681.5
ББК 32.965в6

© Петрозаводский государственный университет Кольский филиал, 2011

© Днепродзержинский государственный технический университет, 2011

© Российский химико–технологический университет им. Д. И. Менделеева Новомосковский институт, 2011

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В качестве определения нейронных сетей может быть принято: ИНС (искусственная нейронная сеть) – параллельно распределенная структура обработки информации, состоящая из отдельных элементов (нейронов), соединенных между собой связями.

Нейрокомпьютеры (компьютеры, построенные на основе нейронных сетей) обладают целым рядом свойств, привлекательных с точки зрения их практического использования:

1. Сверхвысокое быстродействие за счет использования массового параллелизма обработки информации.

2. Толерантность к ошибкам: работоспособность сохраняется при повреждении значительного числа нейронов.

3. Способность к обучению; программирование вычислительной системы заменяется обучением.

4. Способность к распознаванию образов в условиях сильных помех и искажений.

Однако, первые 2 свойства имеют место только при аппаратной реализации нейронных сетей. Аппаратно реализованные нейронные сети обеспечивают решение сложных задач за времена порядка времен срабатывания цепочек электронных и/или оптических элементов. Решение слабо зависит от неисправности отдельного нейрона. Это делает их привлекательными для использования в составе бортовых вычислительных систем.

Пакет Neural Networks Toolbox системы Matlab содержит средства для проектирования, моделирования, обучения и использования множества известных парадигм современного аппарата искусственных нейронных сетей: от базовых моделей персептрона до самых современных ассоциативных и самоорганизующихся сетей.

В настоящее время ИНС применяются для решения многих не формализуемых или трудно формализуемых задач:

- распознавания и синтеза речи;
- распознавания аэрокосмических изображений;
- прогнозирования котировки ценных бумаг и курса валют;
- предупреждения мошенничества с кредитными карточками;
- оценки стоимости недвижимости;
- оценки финансового состояния предприятий и риска невозврата кредитов;
- обработки радиолокационных сигналов;
- контроля движения на скоростных автомагистралях и железных дорогах;
- диагностики в медицине;
- добычи знаний из больших объемов данных в бизнесе, финансах и научных исследованиях.

Нейронные сети можно использовать при следующих условиях:

- Если задачу может решать человек.
- Если при решении задачи можно выделить множество входных факторов (сигналов, признаков, данных и т.п.) и множество выходных факторов.
- Если изменения входных факторов приводит к изменению выходных.

В то же время применение нейронных сетей при решении некоторых задач может оказаться эффективнее использования разума человека. Это объясняется тем, что человеческий разум ориентирован на решение задач в трехмерном пространстве. Многомерные задачи для него характеризуются значительно большей трудоемкостью.

Искусственным нейронным сетям не свойственно такое ограничение. Им все равно решать трехмерную или 10-мерную задачу.

Место нейросетевой технологии среди других методов обработки данных показано на рис.1, 2.

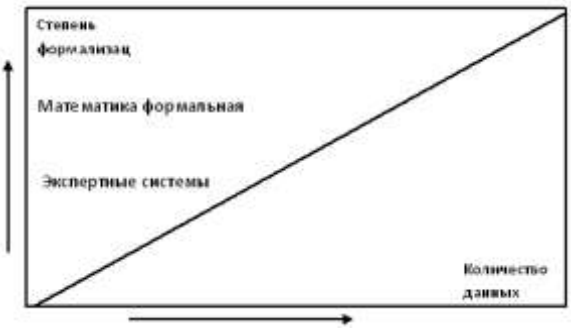


Рис.1 Область использования нейронных сетей

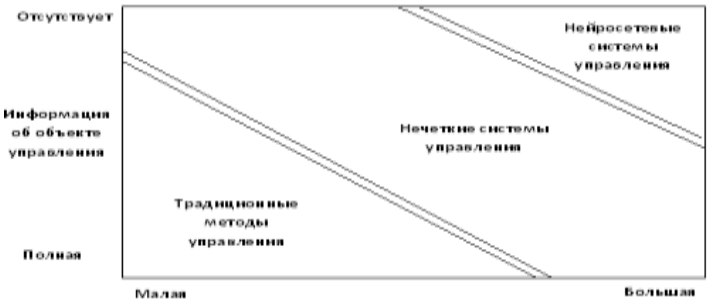


Рис.2 Области эффективного применения различных систем управления

Подобно биологической нейронной системе ИНС является вычислительной системой с огромным числом параллельно функционирующих простых процессоров с множеством связей. Модели ИНС в некоторой степени воспроизводят "организационные" принципы, свойственные мозгу человека. Моделирование биологической нейронной системы с использованием ИНС может также способствовать лучшему пониманию биологических функций. Такие технологии производства, как VLSI (сверхвысокий уровень интеграции) и оптические аппаратные средства, делают возможным подобное моделирование.

Глубокое изучение ИНС требует знания нейрофизиологии, науки о познании, психологии, физики (статистической механики), теории управления, теории вычислений, проблем искусственного интеллекта, статистики/математики, распознавания образов, компьютерного зрения, параллельных вычислений и аппаратных средств (цифровых/аналоговых/VLSI/оптических).

С другой стороны, ИНС также стимулируют эти дисциплины, обеспечивая их новыми инструментами и представлениями. Этот симбиоз жизненно необходим для исследований по нейронным сетям. Представим некоторые проблемы, решаемые в контексте ИНС и представляющие интерес для ученых и инженеров.

Классификация образов. Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), 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 $((x_1,y_1), (x_2,y_2)..., (x_n, y_n))$ (пары данных вход-выход), которая генерируется неизвестной функцией f(x), искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции f(x). Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

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

Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе

ограничений и максимизирует или минимизирует целевую функцию. Задача коммивояжера, относящаяся к классу NP-полных, является классическим примером задачи оптимизации.

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 t. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия u(t), при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление двигателем.

При применении нейронных сетей необходимо решить следующие задачи:

1. Постановка задачи, пригодной для решения с помощью нейронной сети.
2. Выбор модели ИНС.
3. Подготовка исходных данных для обучения ИНС.
4. Обучение ИНС.
5. Собственно решение задачи с помощью обученной ИНС

Кроме того, иногда нужен еще один этап – интерпретация решения, полученного нейронной сетью.

Наиболее трудоемкими процессами при использовании нейронных сетей являются подготовка исходных данных для обучения и обучение нейронной сети.

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами (рис.3). Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от нашей кожи, ушей и глаз к мозгу, процессы мышления и управления действиями - все это реализовано в живом организме как передача электрических импульсов между нейронами. Рассмотрим строение биологического нейрона. Каждый нейрон имеет отростки нервных волокон двух типов - дендриты, по которым принимаются импульсы, и единственный аксон, по которому нейрон может передавать импульс. Аксон контактирует с дендритами других нейронов через специальные образования - *синапсы*, которые влияют на силу импульса.

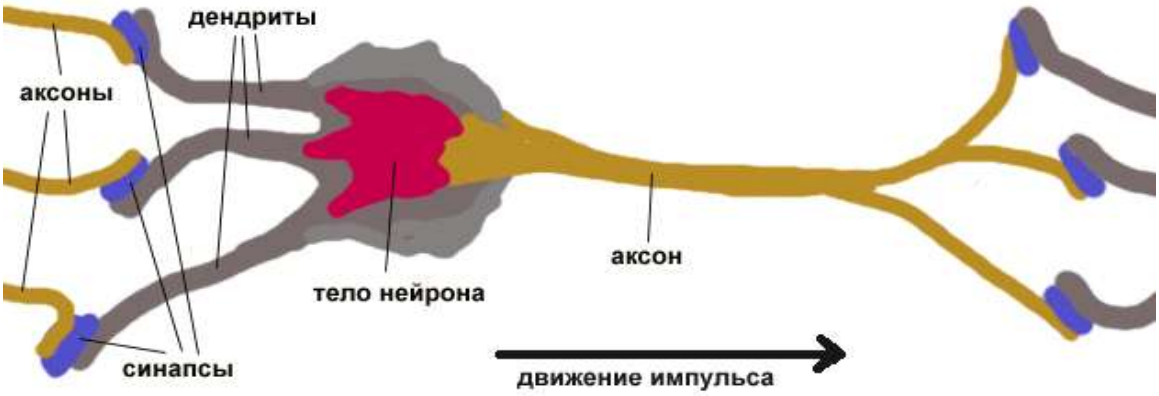


Рис.3. Строение биологического нейрона

Можно считать, что при прохождении синапса сила импульса меняется в определенное число раз, которое мы будем называть весом синапса. Импульсы, поступившие к нейрону одновременно по нескольким дендритам, суммируются. Если суммарный импульс превышает некоторый порог, нейрон возбуждается, формирует собственный импульс и передает его далее по аксону. Передача информации между нейронами осуществляется **ЭЛЕКТРИЧЕСКИМИ ИМПУЛЬСАМИ**. Под действием этих сигналов возникает процесс химической диффузии ионов натрия, калия, хлора и некоторых других элементов, изменяющий электрический **ПОТЕНЦИАЛ МЕМБРАНЫ**. Когда этот потенциал (суммарный импульс) достигает некоторой величины, называемой **ПОРОГОМ НЕЙРОНА**, нейрон возбуждается и вырабатывает импульс, который уходит по аксону. При этом потенциал мембраны резко падает и нейрон "разряжается". После некоторой паузы нейрон может опять сформировать импульс. Важно отметить, что веса синапсов могут изменяться со временем, а значит, меняется и

поведение соответствующего нейрона. Выходной сигнал проходит по ветвям аксона и достигает **СИНАПСОВ**, которые соединяют аксоны с дендритными деревьями других нейронов. Все нейроны вырабатывают сигналы только **ОДНОГО ЗНАКА**. Если импульсы, поступающие на **СИНАПС**, приводят к повышению мембранного потенциала, то такой **СИНАПС** называется **ВОЗБУЖДАЮЩИМ**. В противном случае **СИНАПС** называется **ТОРМОЗЯЩИМ**. Величина входного сигнала, генерируемого синапсом, может быть различной даже при одинаковой величине сигнала, проходящего через синапс. Эти различия определяются эффективностью или **ВЕСОМ СИНАПСА.**

Важно отметить, что веса синапсов могут изменяться со временем, а значит, меняется и поведение соответствующего нейрона.

Под нейронными сетями подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Адаптируемые и обучаемые, они представляют собой распараллеленные системы, способные к обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является *искусственный нейрон* или просто нейрон, названный так по аналогии с биологическим прототипом.

К настоящему времени предложено и изучено большое количество моделей нейроноподобных элементов и нейронных сетей.

Материалы учебного пособия составлены по источникам, представленным в Internet, в большом количестве различных журналов в виде отдельных статей, а также в help описаниях к конкретным программным системам. Использованы также материалы, перечисленные в конце учебного пособия [1-11].

1 СТРУКТУРА ИСКУССТВЕННОГО НЕЙРОНА

Структура искусственного нейрона показана на рис.4. В состав нейрона входят умножители (синапсы), сумматор и нелинейный преобразователь.

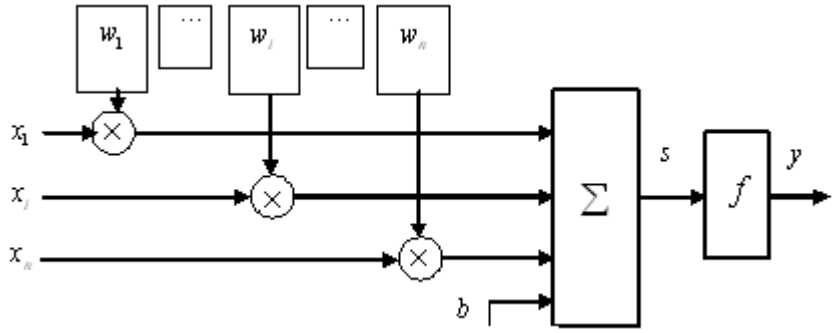


Рис.4. Структура искусственного нейрона

Синапсы осуществляют связь между нейронами и 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. Нейрон в целом реализует скалярную функцию векторного аргумента. Нетрудно построить математическую модель описанного процесса. Математическая модель нейрона описывается соотношениями:

$$s = \sum_{i=1}^{n} w_i x_i + b_i, \qquad y = f(s),$$

где w_i – вес синапса, $(i = 1,...n)$; s – результат суммирования; x_i – компонент входного вектора (входной сигнал); y – выходной сигнал нейрона; n – число входов нейрона; f – нелинейное преобразование (функция активации или передаточная функция); b – значение смещения (в нейронную сеть иногда вводя смещение, которое действует как весовой коэффициент от ячейки с активацией, равной 1; смещение увеличивает входное воздействие в сеть на единицу; вместо смещения в ряде случаев применяется фиксированный порог для функции активации).

2 ФУНКЦИИ АКТИВАЦИИ

В общем случае входной сигнал, весовые коэффициенты и значения смещения могут принимать вещественные значения. Выход y определяется видом функции активации и может быть как действительным, так и целым (рис. 5).

Синоптические веса с положительными весами называют возбуждающими, с отрицательными – тормозящими.

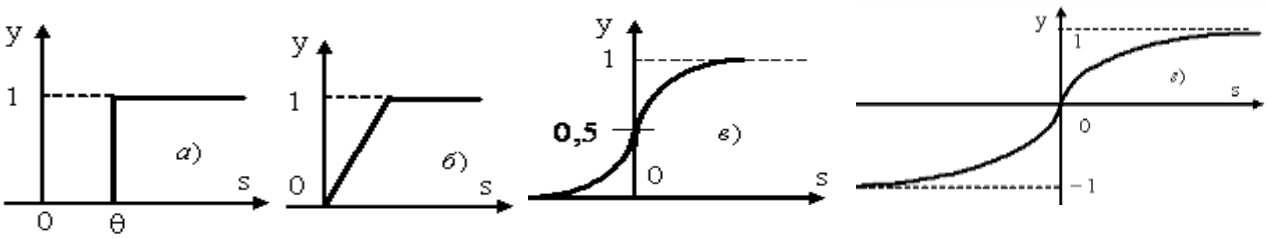


Рис.5 Примеры активационных функций: *а* – пороговая; *б* – полулинейная с насыщением; *в* – сигмоид (логистическая); *г* сигмоид (гиперболический тангенс)

Нейрон преобразует полученный суммарный импульс в соответствии с некоторой передаточной функцией $f(s)$. Таким образом, нейрон полностью описывается своими весами w_i и передаточной функцией $f(s)$. Получив набор чисел (вектор) x_i в качестве входов, нейрон выдает некоторое число у на выходе.

На входной сигнал s нелинейный преобразователь отвечает выходным сигналом $f(s)$, который представляет собой выход нейрона y. Примеры активационных функций представлены в табл. 1 и на рис.5.

Табл. 1 Примеры активационных функций

Название	Формула	Область значений
Пороговая	$f(s)=\begin{cases}0, & s<\theta\\ 1, & s\geq\theta\end{cases}$	(0,1)
Знаковая	$f(s)=\begin{cases}1, & s>0\\ -1, & s\leq 0\end{cases}$	(-1,1)
Сигмоидальная (логистическая)	$f(s)=\frac{1}{1+e^{-s}}$	(0,1)
Полулинейная	$f(s)=\begin{cases}s, & s>0\\ 0, & s\leq 0\end{cases}$	$(0,\infty)$
Линейная	$f(s)=s$	$(-\infty,+\infty)$
Радиальная базисная (гауссова)	$f(s)=\exp(-s^2)$	(0,1)
Полулинейная с насыщением	$f(s)=\begin{cases}0, & s\leq 0\\ s, & 0<s<1\\ 1, & s\geq 1\end{cases}$	(0,1)
Линейная с насыщением	$f(s)=\begin{cases}-1, & s\leq -1\\ s, & -1<s<1\\ 1, & s\geq 1\end{cases}$	(-1,1)
Гиперболический тангенс (сигмоидальная)	$f(s)=\frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	(-1,1)
Треугольная	$f(s)=\begin{cases}1-\lvert s\rvert, & \lvert s\rvert\leq 1\\ 0, & \lvert s\rvert>1\end{cases}$	(0,1)

Одной из наиболее распространенных активационных функций является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид:

$$f(s)=\frac{1}{1+e^{-s}}$$

Одно из ценных свойств сигмоидальной функции – простое выражение для ее производной:

$$f'(s)=f(s)[1-f(s)],$$

которое используется в различных алгоритмах обучения.

3 КЛАССИФИКАЦИЯ НЕЙРОННЫХ СЕТЕЙ

Искусственная нейронная сеть (*ИНС*) - это набор нейронов, соединенных между собой. Как правило, передаточные функции всех нейронов в сети фиксированы, а веса являются параметрами сети и могут изменяться. Некоторые входы нейронов помечены как внешние

входы сети, а некоторые выходы - как внешние выходы сети. Подавая любые числа на входы сети, мы получаем какой-то набор чисел на выходах сети. Таким образом, работа нейросети состоит в преобразовании входного вектора X в выходной вектор Y, причем это преобразование задается весами сети.

Нейронные сети различают по структуре сети (связей между нейронами), особенностям модели нейрона, особенностям обучения сети. По структуре нейронные сети можно разделить (рис.6) на неполносвязные (или слоистые) и полносвязные, со случайными и регулярными связями, с симметричными и несимметричными связями.

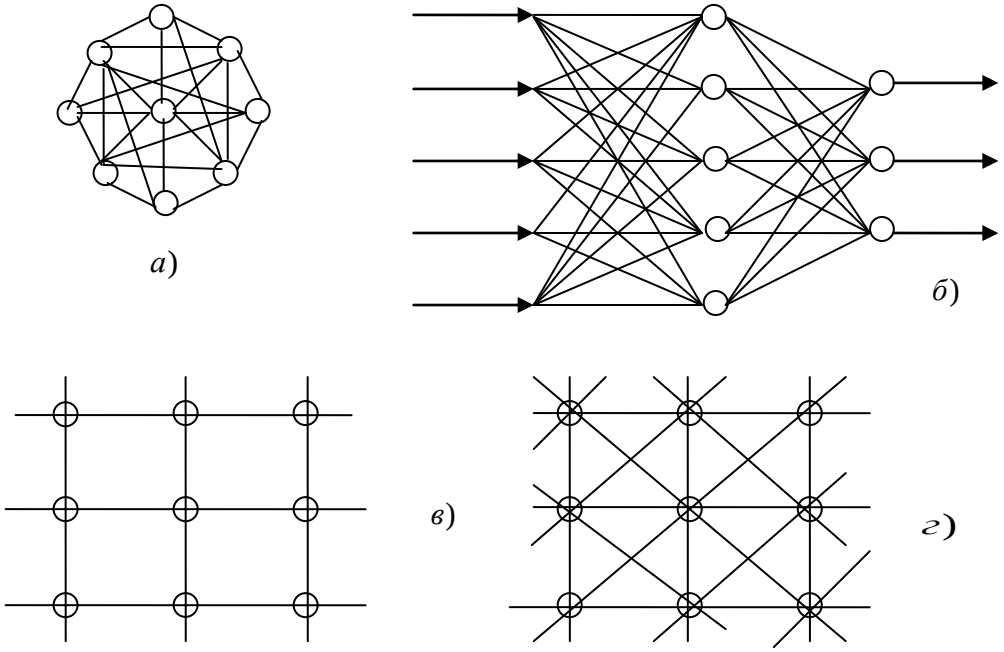


Рис.6 Архитектуры нейронных сетей: *а* – полносвязная сеть; *б* – многослойная сеть с последовательными связями; *в, г* – слабосвязные сети

Неполносвязные нейронные сети (описываемые неполносвязным ориентированным графом и обычно называемые персептронами), подразделяются на однослойные (простейшие персептроны) и многослойные, с прямыми, перекрестными и обратными связями. Классическим вариантом слоистых сетей являются сети прямого распространения (рис.7).

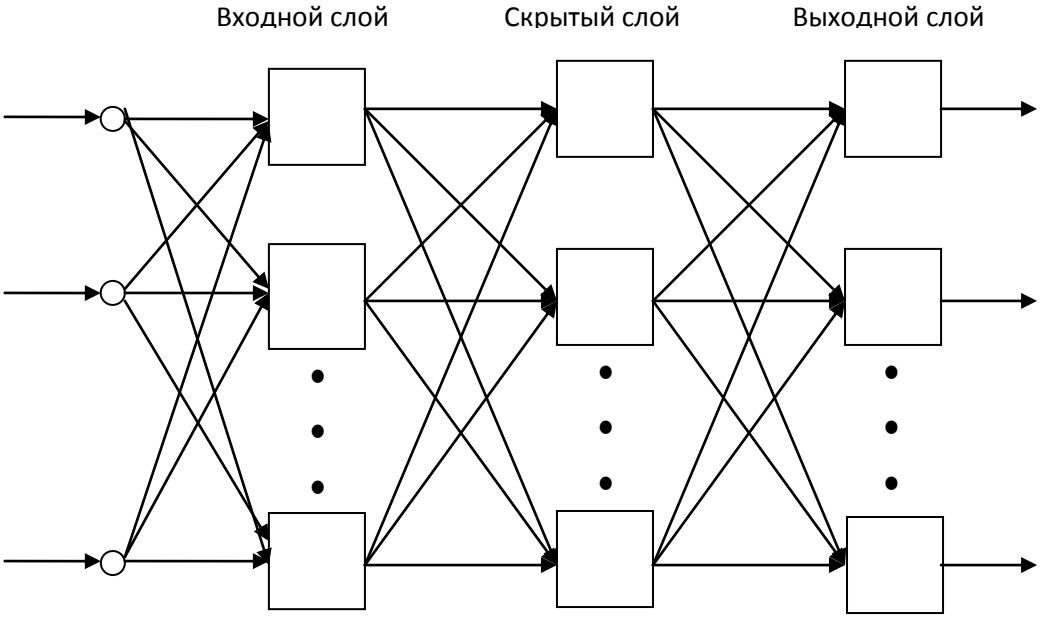


Рис.7. Многослойная (двухслойная) сеть прямого распространения

В нейронных сетях с прямыми связями нейроны *j*-ого слоя по входам могут соединяться только с нейронами i-ых слоев, где $j > i$, т.е. с нейронами нижележащих слоев. В нейронных сетях с перекрестными связями допускаются связи внутри одного слоя, т.е. выше приведенное неравенство заменяется на $j >= i$ В нейронных сетях с обратными связями используются и связи *j*-ого слоя по входам с *i*-ым при $j < i$. Кроме того, по виду связей различают персептроны с регулярными и случайными связями.

По организации обучения разделяют обучение нейронных сетей с учителем (supervised neural networks) и без учителя (nonsupervised).

При обучении с учителем предполагается, что есть внешняя среда, которая предоставляет обучающие примеры (значения входов и соответствующие им значения выходов) на этапе обучения или оценивает правильность функционирования нейронной сети и в соответствии со своими критериями меняет состояние нейронной сети или поощряет (наказывает) нейронную сеть, запуская тем самым механизм изменения ее состояния. Под состоянием нейронной сети, которое может изменяться, обычно понимается:

- веса синапсов нейронов (карта весов - map) (коннекционистский подход);
- веса синапсов и пороги нейронов (обычно в этом случае порог является более легко изменяемым параметром, чем веса синапсов);
- установление новых связей между нейронами (свойство биологических нейронов устанавливать новые связи и ликвидировать старые называется пластичностью).

По способу обучения разделяют обучение по входам и по выходам. При обучении по входам обучающий пример представляет собой только вектор входных сигналов, а при обучении по выходам в него входит и вектор выходных сигналов, соответствующий входному вектору.

По способу предъявления примеров различают предъявление одиночных примеров и "страницы" примеров. В первом случае изменение состояния нейронной сети (обучение) происходит после предъявления каждого примера. Во втором - после предъявления "страницы" (множества) примеров на основе анализа сразу их всех.

Практически любую задачу можно свести к задаче, решаемой нейросетью.

В табл. 2 показано, каким образом следует сформулировать в терминах нейросети задачу распознавания рукописных букв.

Табл. 2. Задача распознавания рукописных букв в терминах нейросети

Задача распознавания рукописных букв	
Дано: растровое черно-белое изображение буквы размером 30x30 пикселов	**Надо**: определить, какая это буква (в алфавите 33 буквы)
Формулировка для нейросети	
Дано: входной вектор из 900 двоичных символов (900=30x30)	**Надо**: построить нейросеть с 900 входами и 33 выходами, которые помечены буквами. Если на входе сети - изображение буквы "А", то максимальное значение выходного сигнала достигается на выходе "А". Аналогично сеть работает для всех 33 букв.

Поясним, зачем требуется выбирать выход с максимальным уровнем сигнала. Дело в том, что уровень выходного сигнала, как правило, может принимать любые значения из какого-то отрезка. Однако, в данной задаче нас интересует не аналоговый ответ, а всего лишь номер категории (номер буквы в алфавите). Поэтому используется следующий подход - каждой категории сопоставляется свой выход, а ответом сети считается та категория, на чьем выходе уровень сигнала максимален. В определенном смысле уровень сигнала на выходе "А" - это достоверность того, что на вход была подана рукописная буква "А". Задачи, в которых нужно отнести входные данные к одной из известных категорий, называются *задачами*

классификации. Изложенный подход - стандартный способ классификации с помощью нейронных сетей.

Затем можно переходить к следующему вопросу «как строить сеть». Это решается в два этапа:

1) выбор типа (архитектура сети);
2) подбор весов (обучение) сети.

На **первом этапе** следует выбрать следующее:

- какие нейроны мы хотим использовать (число входов, передаточные функции);
- каким образом следует соединить их между собой;
- что взять в качестве входов и выходов сети.

Эта задача на первый взгляд кажется необозримой, но необязательно придумывать нейросеть "с нуля" - существует несколько десятков различных нейросетевых архитектур, причем эффективность многих из них доказана математически. Наиболее популярные и изученные архитектуры - это многослойный перцептрон, нейросеть с общей регрессией, сети Кохонена и другие.

На **втором этапе** нам следует "обучить" выбранную сеть, то есть подобрать такие значения ее весов, чтобы сеть работала нужным образом. Необученная сеть подобна ребенку - ее можно научить чему угодно. В используемых на практике нейросетях количество весов может составлять несколько десятков тысяч, поэтому обучение - действительно сложный процесс. Для многих архитектур разработаны специальные алгоритмы обучения, которые позволяют настроить веса сети определенным образом. Наиболее популярный из этих алгоритмов - метод обратного распространения ошибки (Error Back Propagation), используемый, например, для обучения персептрона.

В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

1) *входные* нейроны – на них подается входной вектор, кодирующий входное воздействие или образ внешней среды; вычислительных процедур в этих нейронах не осуществляется, информация передается со входа на выход путем изменения его активации;

2) *выходные* нейроны – выходные значения которых представляют выход сети;

3) *промежуточные* нейроны – составляют основу ИНС, преобразования в них выполняются по вышеприведенным формулам.

Выбор структуры ИНС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач существуют оптимальные конфигурации.

Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать проблему синтеза новой конфигурации, причем необходимо руководствоваться следующими принципами:

1) возможности сети возрастают с увеличением числа ячеек сети, плотности связи между ними и числом выделенных слоев;
2) введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;
3) сложность алгоритмов функционирования сети (например, введения нескольких типов синапсов) также способствует усилению возможностей сети.

В большинстве случаев оптимальный вариант получается на основе интуитивного подбора, хотя в литературе приведены доказательства того, что для любого алгоритма существует нейронная сеть, которая может его реализовать.

Многие задачи: распознавания образов (зрительных, речевых), управления, прогнозирования, идентификации сложных систем, сводятся к следующей постановке: **необходимо построить отображение $X \to Y$ такое, чтобы в ответ на каждый возможный входной сигнал X формировался правильный выходной сигнал Y.**

Отображение задается конечным набором пар («вход», «известный выход»). Число таких пар (обучающих примеров) существенно меньше общего числа возможных сочетаний значений входных и выходных сигналов. Совокупность всех обучающих примеров носит название обучающей выборки.

В **задачах распознавания** образов X – некоторое представление образа (изображение, вектор чисел), Y – номер класса, к которому принадлежит входной образ.

В **задачах управления** X – набор контролируемых параметров управляемого объекта, Y – код, определяющий управляющее воздействие, соответствующее текущим значениям контролируемых параметров.

В задачах **прогнозирования** в качестве входных сигналов используются временные ряды, представляющие значения контролируемых переменных на некотором интервале времени. Выходной сигнал – множество переменных, которое является подмножеством переменных входного сигнала.

Вообще говоря, большая часть прикладных задач может быть сведена к реализации некоторого сложного многомерного функционального преобразования $X \to Y$.

В результате построения такого преобразования (отображения) необходимо добиваться того, чтобы обеспечивалось формирование правильных выходных сигналов в соответствии:

1) со всеми примерами обучающей выборки;

2) со всеми возможными входными сигналами, которые не вошли в обучающую выборку.

Отметим, что теоретической основой для построения НС является следующее утверждение: **для любого множества пар входных - выходных векторов произвольной размерности существует двухслойная однородная нейронная сеть с последовательными связями, с сигмоидальными передаточными функциями и с конечным числом нейронов, которая для каждого входного вектора X^k формирует соответствующий ему выходной вектор Y^k.**

Таким образом, для представления многомерных функций многих переменных может быть использована однородная нейронная сеть, имеющая всего один скрытый слой, с сигмоидальными передаточными функциями нейронов.

Для оценки числа нейронов в скрытых слоях ОНС можно воспользоваться формулой для оценки необходимого числа синаптических весов L_w (в многослойной сети с сигмоидальными передаточными функциями):

$$\frac{mN}{1+\log_2 N} \le L_w \le m\left(\frac{N}{m}+1\right)(n+m+1)+m,$$

где n – размерность входного сигнала, m –размерность выходного сигнала, N – число элементов обучающей выборки.

Число нейронов в двухслойной сети составит: $L = \frac{L_w}{n+m}$ или

$$\frac{N}{10} - n - m \le L \le \frac{N}{2} - n - m$$

Теорема о полноте. Любая непрерывная функция на замкнутом ограниченном множестве может быть равномерно приближена функциями, вычисляемыми нейронными сетями, если функция активации нейрона дважды непрерывно дифференцируема и нелинейна.

Таким образом, НС являются универсальными аппроксимирующими системами.

Задавшись определенной структурой ИНС, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех весовых коэффициентов. Этот этап называется обучением ИНС.

3.1 Обучение нейронной сети

Таким образом, нейросети составляются из простых элементов, действующих параллельно. Можно обучить нейросеть, регулируя значения весов между элементами. Обычно сеть регулируется или обучается так, чтобы частный вход вел к целевому выходу. Регулируемая сеть основана на сравнении выхода и цели, пока сетевой выход не будет соответствовать цели. Обычно имеется много обучающих пар вход/цель. Количество обучающих пар выбирается в зависимости от постановки задачи. Переменные, выступающие в роли входных и выходных сигналов НС, могут представлять собой экспериментальные данные, получаемые измерением выходных параметров объекта при задании определенных входных.

Обучить нейронную сеть - значит, сообщить ей, чего мы от нее добиваемся. Этот процесс очень похож на обучение ребенка алфавиту. Показав ребенку изображение буквы "А", мы спрашиваем его: "Какая это буква?" Если ответ неверен, мы сообщаем ребенку тот ответ, который мы хотели бы от него получить: "Это буква А". Ребенок запоминает этот пример вместе с верным ответом, то есть в его памяти происходят некоторые изменения в нужном направлении. Мы будем повторять процесс предъявления букв снова и снова до тех пор, когда все 33 буквы будут твердо запомнены. **Такой процесс называют "обучение с учителем"** (рис.8).

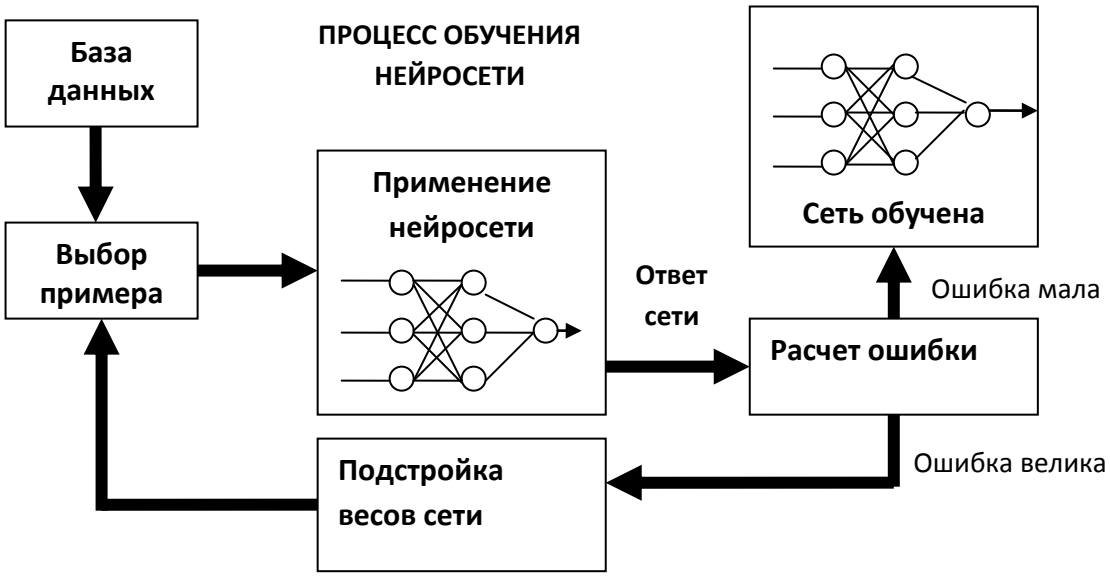


Рис.8 Иллюстрация процесса обучения НС

При обучении сети мы действуем совершенно аналогично. У нас имеется некоторая база данных, содержащая примеры (набор рукописных изображений букв). Предъявляя изображение буквы "А" на вход сети, мы получаем от нее некоторый ответ, не обязательно верный. Нам известен и верный (желаемый) ответ - в данном случае нам хотелось бы, чтобы на выходе с меткой "А" уровень сигнала был максимален.

Обычно в качестве желаемого выхода в задаче классификации берут набор (1, 0, 0, ...), где 1 стоит на выходе с меткой "А", а 0 - на всех остальных выходах. Вычисляя разность между желаемым ответом и реальным ответом сети, мы получаем 33 числа - *вектор ошибки*. Алгоритм обратного распространения ошибки - это набор формул, который позволяет по вектору ошибки вычислить требуемые поправки для весов сети.

Одну и ту же букву (а также различные изображения одной и той же буквы) мы можем предъявлять сети много раз. В этом смысле обучение скорее напоминает повторение упражнений в спорте - тренировку.

Оказывается, что после многократного предъявления примеров веса сети стабилизируются, причем сеть дает правильные ответы на все (или почти все) примеры из базы данных. В таком случае говорят, что "сеть выучила все примеры", " сеть обучена", или "сеть натренирована". В программных реализациях можно видеть, что в процессе обучения величина

ошибки (сумма квадратов ошибок по всем выходам) постепенно уменьшается. Когда величина ошибки достигает нуля или приемлемого малого уровня, тренировку останавливают, а полученную сеть считают натренированной и готовой к применению на новых данных.

Важно отметить, что вся информация, которую сеть имеет о задаче, содержится в наборе примеров. Поэтому качество обучения сети напрямую зависит от количества примеров в обучающей выборке, а также от того, насколько полно эти примеры описывают данную задачу. Так, например, бессмысленно использовать сеть для предсказания финансового кризиса, если в обучающей выборке кризисов не представлено. Считается, что для полноценной тренировки требуется хотя бы несколько десятков (а лучше сотен) примеров.

Повторим еще раз, что обучение сети - сложный и наукоемкий процесс. Алгоритмы обучения имеют различные параметры и настройки, для управления которыми требуется понимание их влияния.

3.2 Алгоритмы обучения нейронных сетей

Рассмотрим идею одного из самых распространенных алгоритмов обучения – **алгоритма обратного распространения ошибки**. Это итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущего выхода от желаемого выхода в многослойных нейронных сетях. Этот алгоритм используется для обучения многослойных НС с последовательными связями вида рис.9. Нейроны в таких сетях делятся на группы с общим входным сигналом - слои. На каждый нейрон первого слоя подаются все элементы внешнего входного сигнала. Все выходы нейронов *m*-го слоя подаются на каждый нейрон слоя *m+1*. Нейроны выполняют взвешенное суммирование элементов входных сигналов. К сумме элементов входных сигналов, помноженных на соответствующие синаптические веса, прибавляется смещение нейрона. Над результатом суммирования выполняется нелинейное преобразование - функция активации (передаточная функция). Значение функции активации есть выход нейрона.

В многослойных сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, неизвестны, и трех- или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Наиболее приемлемым вариантом обучения в таких условиях оказался градиентный метод поиска минимума функции ошибки с рассмотрением сигналов ошибки от выходов НС к ее входам, т.е. в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения.

В данном алгоритме функции ошибки представляют собой сумму квадратов рассогласования (ошибки) желаемого выхода сети и реального. При вычислении элементов вектора градиента использован своеобразный вид производных функций активации сигмоидного типа. Алгоритм действует циклически и его циклы принято называть эпохами. На каждом этапе на вход сети поочередно подаются все обучающие наблюдения, выходные значения сети сравниваются с целевыми значениями и вычисляется ошибка. Значение ошибки, а также градиента поверхности ошибок используется для корректировки весов, после чего все действия повторяются. Начальная конфигурация сети выбирается случайным образом, и процесс обучения прекращается либо когда пройдено определенное количество эпох, либо когда ошибка достигнет некоторого определенного уровня малости, либо когда ошибка перестанет уменьшаться.

Обучение сети методом ОРО включает в себя три этапа:

- прямое распространение входного обучающего образа;
- вычисление ошибки и ее обратное распространение;
- регулирование весов.

Этот метод основан на вычислении вектора градиента поверхности ошибок, который указывает направление кратчайшего спуска по поверхности из данной точки. Последовательность шагов приводит после ряда итераций к минимуму поверхности ошибок. Основная трудность здесь представляет собой выбор длины шага. На практике величина шага

принимается пропорциональной крутизне склона с постоянной, называемой скоростью обучения. Рассмотрим алгоритм для многослойного прямонаправленного персептрона (МПП) (рис.9).

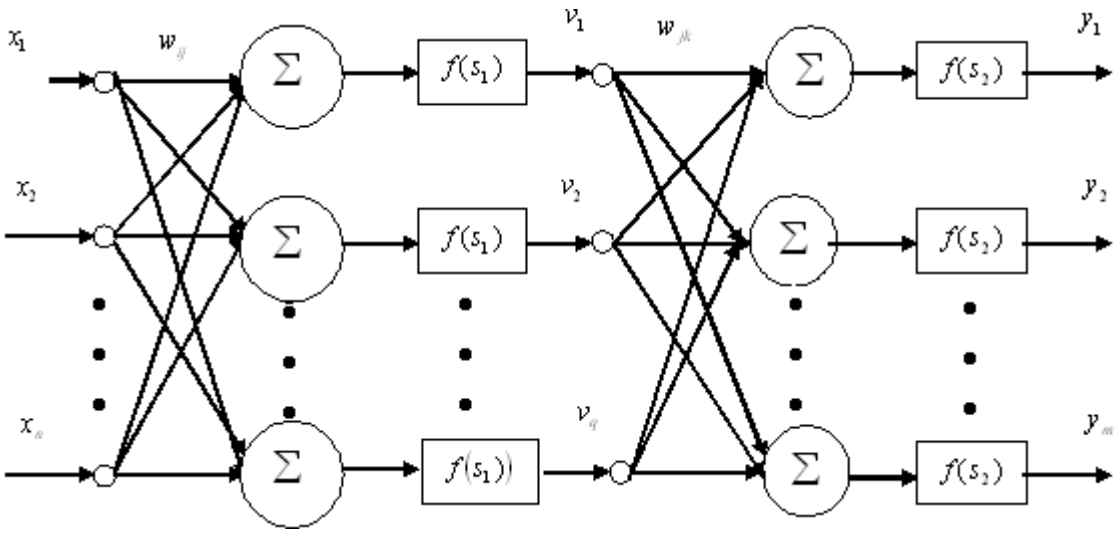


Рис.9 Схема трехслойного персептрона

Обозначим через $x_i (i = 1,2,...,n)$ входные нейроны; $v_j (j = 1,2,...,q)$ – нейроны скрытого слоя; $y_k (k = 1,2,...,m)$ – выходные нейроны; w_{ij} – веса от ячеек входного слоя к нейронам скрытого слоя; w_{jk} – веса от нейронов скрытого слоя к выходным нейронам. Индексом p $(p = 1,2,...,P)$ обозначим различные образы, предъявляемые на вход. Входные сигналы могут быть **бинарными, биполярными** или **непрерывными.**

Поведение сети определяется на основе ряда пар «вход-выход». Каждый обучающий пример состоит из n входных сигналов x_i и m требуемых выходных сигналов t_k . Обучение МПП для конкретной задачи эквивалентно нахождению таких значений всех синаптических весов, при которых для соответствующего входа формируется требуемый выход или обучение многослойной сети заключается в регулировании всех весов таким образом, чтобы ошибка между требуемыми выходными t_k и действительными выходными y_k сигналами, усредненная по всем обучающим примерам, была минимальна. При предъявлении образа p на вход сети скрытая ячейка j принимает сигнал:

$$s_j^p = \sum_i w_{ij} x_i^p$$

и на своем выходе с помощью функции активации вырабатывает такой сигнал:

$$v_j^p = f(s_j^p) = f\left(\sum_i w_{ij} x_i^p\right).$$

Выходная ячейка с номером k суммирует сигналы от нейронов скрытого слоя, образуя сигнал, равный:

$$s_k^p = \sum_j w_{jk} v_j^p = \sum_j w_{jk} f\left(\sum_i w_{ij} x_i^p\right),$$

и после воздействия функции активации формирует выходной сигнал (пороговые значения отброшены, т.к. они всегда могут быть введены в сеть):

$$y_k^p = f\left(s_k^p\right) = f\left(\sum_j w_{jk} v_j^p\right) = f\left[\sum_j w_{jk} f\left(\sum_i w_{ij} x_i^p\right)\right].$$

В качестве функции ошибок примем функцию вида:

$$E[w] = 0{,}5\sum_p \sum_k \left(t_k^p - y_k^p\right)^2,$$

где t_k^p – требуемое значение выхода.

Подставляя в последнее выражение значение для y_k^p, получим:

$$E[w] = 0{,}5\sum_p \sum_k \left\{ t_k^p - f\left[\sum_j w_{jk} f\left(\sum_i w_{ij} x_i^p \right) \right] \right\}^2.$$

Функция $E[w]$ является непрерывно дифференцируемой функцией от каждого веса, входящего в него выражения, поэтому можно воспользоваться алгоритмом градиентного спуска для нахождения весов.

Для весов между скрытыми и выходными слоями правило градиентного спуска дает:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = \eta \sum_p \left(t_k^p - y_k^p\right) \cdot f'\left(s_k^p\right) \cdot v_j^p = \eta \sum_p \delta_k^p v_j^p,$$

где $\delta_k^p = f'\left(s_k^p\right)\left(t_k^p - y_k^p\right)$, η – коэффициент скорости обучения ($0 < \eta < 1$).

Для весов между входным и скрытым слоями нужно продифференцировать выражение для $E[w]$, воспользовавшись цепным правилом:

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} = -\eta \sum_p \frac{\partial E}{\partial v_j^p} \cdot \frac{\partial v_j^p}{\partial w_{ij}} = \eta \sum_p \sum_k \left(t_k^p - y_k^p\right) f'\left(s_k^p\right) w_{jk} f'\left(s_j^p\right) x_i^p =$$
$$= \eta \sum_p \sum_k \delta_k^p w_{jk} f'\left(s_j^p\right) x_i^p = \sum_p \delta_j^p x_i^p,$$

где $\delta_j^p = f'\left(s_j^p\right) \sum_k w_{jk} d_k^p$.

Отметим, что уравнения для Δw_{jk} и Δw_{ij} имеют одинаковую форму, но различаются значениями параметра δ. В общем случае при произвольном числе слоев правило изменения весов в методе ОРО имеет вид:

$$\Delta w_{\alpha\beta} = \eta \sum_p \delta_{out} v_{in},$$

где суммирование производится по всем предъявляемым образам; выход и вход относятся к двум концам α и β синаптического соединения; v_{in} – активация от скрытой ячейки или реального входы. Значения δ зависят от рассматриваемого слоя; для последнего слоя эта величина определяется по выражению $\delta_k^p = f'\left(s_k^p\right)\left(t_k^p - y_k^p\right)$, а для других слоев – формулой подобной $\delta_j^p = f'\left(s_j^p\right) \sum_k w_{jk} d_k^p$.

Выражения, определяющие правила изменения весов, записаны в виде сумм по предъявляемым образам, однако обычно образы поступают на вход последовательно: образ p предъявляется на вход, и по окончании прохода «вперед-назад» по сети все веса изменяются перед предъявлением следующего образа. Это уменьшает функцию ошибок *E* на каждом шаге. Если образы выбираются в случайном порядке, то движение по пространству весов происходит стохастически, что позволяет более широко исследовать поверхность ошибок. Альтернативная версия изменения весов (групповое обучение) заключается в минимизации функции ошибки таким образом, что весовые изменения накапливаются по всем обучающим примерам, и только затем происходит модификация весов.

Этот алгоритм представляет собой следующую последовательность шагов.

1. Инициировать веса, приняв их малыми случайными величинами.
2. Если условие остановки не выполняется, делать шаги 3-10.

3. Выбирается очередная обучающая пара (X,Y) из обучающего множества; вектор X подается на вход сети. Для каждой обучающей пары выполнять шаги 4-9.

Прямой проход по сети

4. Каждая входная ячейка x_i принимает входной сигнал и распространяет его ко всем нейронам скрытого слоя.

5. Каждая ячейка скрытого слоя $v_j, j=1,2,..,q$ суммирует свои взвешенные входные сигналы $s_j = \sum w_{ij} x_i$ применяет к полученной сумме функцию активации, формируя выходной сигнал $v_j = f(s_j)$, который посылается ко всем ячейкам выходного слоя.

6. Каждая выходная ячейка $y_k, k=1,2,...,m$ суммирует взвешенные сигналы: $s_k = \sum w_{jk} v_j$, формируя после применения функции активации выходной сигнал сети: $y_k = f(s_k)$.

Обратное распространение ошибки

7. Каждая выходная ячейка сопоставляет свое значение выхода с требуемой целевой величиной и вычисляет параметр $\delta_k : \delta_k = (t_k - y_k) f'(s_k)$, после чего определяется корректировочный член для весов: $\Delta w_{jk} = \eta \delta_k v_j$, а параметры δ_k посылаются в нейроны скрытого слоя.

8. Каждая скрытая ячейка v_j суммирует свои $\delta-$входы от нейронов выходного слоя: $s_j = \sum_k \delta_k w_{jk}$

Результат умножается на производную от функции активации для определения δ_j: $\delta_j = f'(s_j) \sum_k \delta_k w_{jk}$, и вычисляется поправочный член: $\Delta w_{ij} = \eta \delta_j x_i$.

Изменение весов

9. Веса между скрытым и выходными слоями модифицируются следующим образом: $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$.

Аналогичным образом изменяются веса между входным и скрытыми слоями: $w_{ij}(new) = w_{ij}(old) + \Delta w_{ij}$.

10. Проверка условия остановки: минимизация ошибки между требуемым и реальным выходными сигналами.

Схема алгоритма ОРО представлена на рис.10.

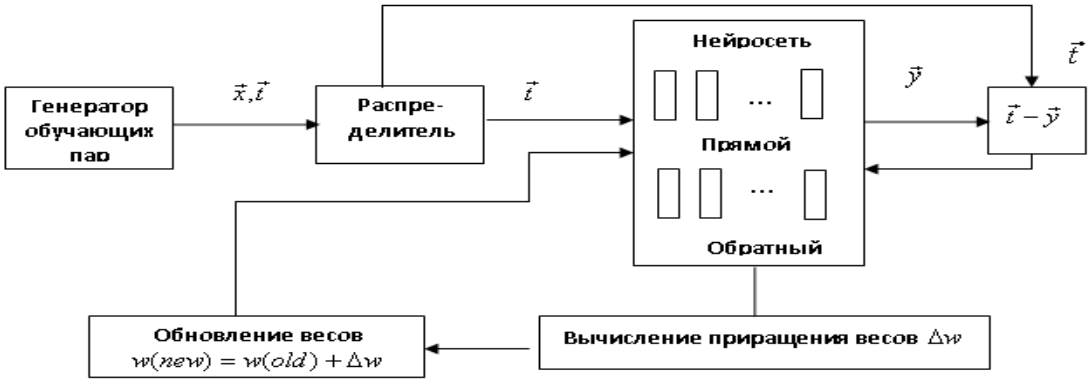


Рис.10 Вычислительная схема метода ОРО

Практические рекомендации при использовании алгоритма ОРО

1. **Выбор начальных значений.** Этот этап оказывает влияние на достижение сетью глобального минимума функции ошибки и на скорость схождения к минимуму. С одной стороны, значения начальных весов не должны быть очень большими, иначе начальные входные сигналы в каждую скрытую или входную ячейку попадут в диапазон, где производная сигмоидной функции активации имеет очень малую величину. С другой стороны, если начальные значения взять достаточно малыми, то сетевой вход в скрытый или выходной нейрон будет ближе к нулю, что приведет к очень медленному обучению. **Общее правило** инициирования начальных весов заключается в выборе их значений из равномерно распределенных величин в интервале (-0,5…0,5) – иногда этот интервал может быть несколько меньше или больше, но не превышать ± 1. Значения весов могут положительными или отрицательными, поскольку окончательные веса после обучения также могут иметь любой знак.

2. **Продолжительность обучения сети.** Предлагается использовать две серии данных во время обучения: серию обучающих образов и серию контрольных образов. Эти две серии являются раздельными. Регулирование веса основано на обучающих образах, однако, во время обучения ошибка вычисляется с использованием контрольных образов. До тех пор пока ошибка для контрольных образов уменьшается, процесс обучения продолжается. При возрастании этой ошибки сеть начинает терять свою способность к обобщению, и в этот момент обучение прекращается.

3. **Количество требуемых обучаемых пар**. Для соотношения между числом обучаемых образов *P*, количеством регулируемых весов w и точностью классификации ε предложено использовать следующее выражение: $w/P = \varepsilon$. Например, для $\varepsilon = 0{,}1$ многослойная сеть с 80 регулируемыми весами потребует 800 обучаемых образов, чтобы быть уверенным в правильной классификации 90% предъявляемых контрольных образов.

4. **Представление данных.** Для нейронной сети легче обучиться набору различных состояний, чем отклику с непрерывным значением.

   Во многих задачах входные и выходные векторы имеют составляющие в одном и том жее диапазоне величин. Вследствие того что один из членов в выражении для корректировки весов является активацией ячейки предыдущего слоя, нейроны, имеющие нулевую активацию, обучаться не будут. Обучение может быть улучшено и в том случае, если входной вектор представлен в биполярной форме, а в качестве функции активации используется биполярная сигмоида (биполярная сигмоида очень близка к функции гиперболического тангенса).

5. **Введение инерционной поправки.** Показано, что т поиск минимума функции ошибок методом градиентного спуска оказывается достаточно медленным, если скорость обучения η мала, и приводить к значительным осцилляциям при большой скорости η.

6. **Модификация функции активации.** Диапазон функции активации должен соответствовать диапазону целевых значений конкретной задачи. Бинарная сигмоидная функция вида $f(x) = [1 + \exp(-x)]^{-1}$ с производной $f'(x) = f(x)[1 - f(x)]$ может быть изменена для перекрытия любого требуемого диапазона с центром при любом значении *x* и необходимом наклоне.

   Сигмоида может иметь расширенный диапазон, чтобы отображать значения в интервале $[a,b]$ для любых a и b. Для этого нужно ввести параметры

   $$\gamma = b - a; \qquad \rho = -a$$

   Тогда сигмоидная функция:

   $$g(x) = \gamma f(x) - \rho$$

   имеет требуемые свойства, т.е. диапазон $[a,b]$. Ее производная выражается как:

   $$g'(x) = \gamma^{-1}[\rho + g(x)][\gamma - \rho - g(x)].$$

Наклон сигмоидной функции может быть изменен с помощью введенного параметра γ.

7. **Число нейронов в скрытом слое.** Подход при выборе скрытых нейронов заключается в том, что на первом этапе число таких нейронов берется заведомо большим, чем требуется , а далее, по мере обучения сети, излишние нейроны убираются. Все нейроны, которые не вносят вклад в решение или дают информацию, не требующуюся в последующем слое, рассматриваются как лишние и и удаляются из скрытого слоя. На практике считается, что сеть достигла сходимости, если разность между требуемым и действительным выходами не превышает 0,1. Если два скрытых нейрона дают приблизительно одинаковый выход для всех обучающих примеров, то только один из них действительно необходим, так как оба нейрона переносят одинаковую информацию. После удаления тех ячеек, которые не дают вклада в решение, значения веса уменьшения сети должны быть изменены путем переобучения сети для получения требуемых характеристик.

Классический метод ОРО относится к алгоритмам с линейной сходимостью. Его известными **недостатками** являются: невысокая скорость сходимости (большое число итераций), возможность сходится не к глобальному оптимуму, а к локальным решениям. Возможен также паралич сети – большинство нейронов функционируют при очень больших значениях аргумента функции активации, т.е. на пологом участке (т.кт ошибка пропорциональна производной, которая на данных участках мала, то процесс обучения практически замирает). Для устранения этих недостатков были предложены многочисленные модификации алгоритма ОРО.

Обучение без учителя. Главная черта, делающая обучение без учителя привлекательным, - это его «самостоятельность». Процесс обучения, как и в случае с учителем, заключается в подстройке весов синапсов. Очевидно, что подстройка весов синапсов может проводиться только на основании информации, доступной в нейроне, т.е. информации о его состоянии, уже имеющихся весовых коэффициентов и поданном векторе X. Исходя их этого и по аналогии с известными принципами самоорганизации нервных клеток, построены алгоритмы обучения Хебба и Кохонена. Общая идея данных алгоритмов заключается в том, что в процессе самообучения путем соответствующей коррекции весовых коэффициентов усиливаются связи между возбужденными нейронами.

3.3 Типы нейронных сетей

Многослойные нейронные сети – эти сети с сигмоидными передаточными функциями являются наиболее общими, универсальными сетевыми архитектурами. Имеются различные структуры многослойных сетей: с последовательными, перекрестными и обратными связями, с фиксированной и переменной структурой. С ростом числа слоев возрастают как сложность построения сети, так и качество ее работы. Многослойные нейронные сети являются универсальными аппроксиматорами.

Сеть Кохонена – кластерный анализ, распознавание образов, классификация. Слой Кохонена самообучается.

Нейронные сети встречного распространения – распознавание и восстановление образов, сжатие данных (с потерей информации), статистический анализ. Сеть содержит два слоя с последовательными связями: первый слой Кохонена, второй Гроссберга. **Преимущества**: 1) сеть проста; 2) сеть быстро обучается; сеть дает возможность строить функцию и обратную к ней функцию, что находит применение при решении практических задач. **Недостатки.** Слабая теоретическая проработка. Сеть не дает возможность строить точные аппроксимации.

Нейронные сети Хопфилда и Хемминга – позволяют просто и эффективно разрешить задачу воссоздания образов по неполной и искаженной информации.

Сети с радиальными базисными функциями (RBF) – двухслойная сеть без обратных связей, которая содержит скрытый слой радиально-симметричных скрытых нейронов (шаблонный слой). Сети RBF моделируют произвольную нелинейную функцию с помощью всего одного промежуточного слоя. Параметры линейной комбинации в выходном слое можно

полностью оптимизировать с помощью хорошо известных методов линейной оптимизации, которые работают быстро и не испытывают трудностей с локальными минимумами, так мешающими при обучении с использованием алгоритма ОРО. Поэтому сеть RBF обучается на порядок быстрее, чем с использованием алгоритма ОРО. **Недостатками** сетей RBF являются: сети обладают плохими экстраполирующими свойствами и получаются весьма громоздкими при большой размерности вектора входов. Модификация сети RBF является радиальная базисная *сеть с нулевой ошибкой*.

Другими модификациями являются вероятностные нейронные сети (PNN) и обобщенно-регрессионная нейронная сеть (GRNN).

Вероятностная нейронная сеть (PNN). Предназначены для решения вероятностных задач и, в частности, задач классификации. Архитектура сети PNN базируется на архитектуре базисной сети, но в качестве второго слоя используют так называемый конкурирующий слой, который подсчитывает вероятность принадлежности входного вектора к тому или иному классу, и, в конечном счете, сопоставляет вектор с тем классом, вероятность принадлежности к которому выше. Выходное значение имеет вероятностный смысл. Сеть быстро обучается. **Недостатком** таких сетей является их объем – сети требуют много памяти и могут медленно работать.

Обобщенно-регрессионная нейронная сеть (GRNN). Данная сеть аналогична вероятностной нейронной сети, но она предназначена для решения задач регрессии. Как и сеть RBF, сеть GRNN не обладает способностью экстраполировать данные.

Линейные нейронные сети. Согласно общепринятому в науке принципу, если более сложная модель не дает лучших результатов, чем более простая, то из них следует предпочесть вторую. В терминах аппроксимации отображений самой простой моделью будет линейная, в которой аппроксимирующая (подгоночная) функция определяется гиперплоскостью. В задаче классификации гиперплоскость размещается таким образом, чтобы она разделяла собой два класса (линейная дискриминантная функция); в задаче регрессии гиперплоскость должна проходить через заданные точки. Линейная модель задается уравнением:

$$Y = X \cdot W + B,$$

где W – матрица весов сети, B – вектор смещений.

На языке нейронных сетей линейная модель представляется сетью без промежуточных слоев, которая в выходном слое содержит только линейные элементы, т.е. элементы с линейной функцией активации. Веса соответствуют элементам матрицы, а пороги – компонентам вектора смещения.

4 ПРИМЕРЫ СОЗДАНИЯ И ИСПОЛЬЗОВАНИЯ НЕЙРОННЫХ СЕТЕЙ

Пакет Neural Networks Toolbox

Пакет Neural Networks Toolbox (нейронные сети) содержит средства для проектирования, моделирования, обучения и использования множества известных парадигм аппарата искусственных нейронных сетей, от базовых моделей персептрона до современных ассоциативных и самоорганизующихся сетей. Пакет может быть использован для решения множества разнообразных задач, таких как обработка сигналов, нелинейное управление, финансовое моделирование и т.п.

Для каждого типа архитектуры и обучающего алгоритма ИНС имеются функции инициализации, обучения, адаптации, создания и моделирования, демонстрации и примеры применения.

1. Функции создания нейронных сетей. Функции данной группы позволяют создать НС заданной структуры (необученную).

Синтаксис: **Имя сети** = **функция** (параметры сети).

Например запись вида
net = newlind (p,t),

Описание: **newlind** – функция проектирования линейной НС. Данная функция по матрицам входных и выходных векторов методом наименьших квадратов определяет веса и смещения линейной НС.

net = newgrnn (p,t,spread) –функция создания обобщенно-регрессионной сети;

net = newrbe (p,t,spread) – функция создания сети с радиальными базисными элементами с нулевой ошибкой на обучающей выборке;

net = newpnn (p,t,spread) - функция создания вероятностной НС;

net = newp (pr,s,tf,lf) - функция создания персептрона (**pr** –матрица минимальных и максимальных значений входных элементов, **s** – число нейронов, **tf** – функция активации, по умолчанию пороговая, **lf** – функция, реализующая алгоритм обучения пресептрона.

Функции моделирования НС

1 Sim - функция, моделирующая работу НС позволяет рассчитать выходы обученной НС при заданных векторах входов.

2 Gensim – функция формирует S-модель нейронной сети для ее запуска в среде Simulink.

Синтаксис: Gensim (net,st), где net – имя созданной сети, st – интервал дискретизации (если НС не имеет задержек, ассоциированных с ее входами или слоями, значение данного аргумента устанавливается равны -1). Функция генерирует нейросетевой блок Simulink для последующего моделирования НС средствами этого пакета.

Пример 1 Создать обобщенно-регрессионную НС (типа GRNN) с именем a, реализующую функциональную зависимость между входом и выходом вида $y = x^2$. Синтаксис: net = newgrnn(p,t,spread), где p – матрица входных векторов, t – матрица целевых векторов, spread – отклонение (по умолчанию 1,0). Использовать следующие экспериментальные данные:

x=[-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9 1];

y=[1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1].

Проверку качества восстановления приведенной зависимости осуществить, используя данные контрольной выборки x1=[-0.9 -0.7 -0.3 0.4 0.8].

Процедура создания и использования данной НС описывается следующим образом:

>> x=[-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9 1]; *задание входных значений*

>> y=[1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1]; *задание целевых значений*

>> a=newgrnn(x,y,0.01); *создание НС с отклонением 0,01*

>> y1=sim(a,[-0.9 -0.7 -0.3 0.4 0.8]) *опрос сети*

y1 =

0.8200 0.5049 0.0316 0.0710 0.6390

Как видно, точность аппроксимации в данном случае получилась не очень высокой. Попробуем использовать сеть с радиальными базисными элементами типа newrbe:

>> a=newrbe(x,y);

>> y1=sim(a,[-0.9 -0.7 -0.3 0.4 0.8])

y1 =

0.8100 0.4900 0.0900 0.1600 0.6400

Нетрудно видеть, что применение сети данного типа приводит к точному восстановлению заданной зависимости.

Пример 2 Рассмотрим задачу восстановления некоторой, вообще говоря, неизвестной зависимости по имеющимся экспериментальным данным с использованием линейной НС. Пусть экспериментальная информация задана значениями

x = [1.0 1.5 3.0 -1.2],

y = [0.5 1.1 3.0 -1.0].

Создадим векторы входа и целей:

>> x = [1.0 1.5 3.0 -1.2];

>> y = [0.5 1.1 3.0 -1.0];

Создадим линейную нейронную сеть:

>> b = newlind(x,y); *Создание НС с именем b.*

Проведем опрос сети для значения входа, равного 3,0 (этому, согласно экспериментальным данным, соответствует целевое значение 3,0):

>> y1=sim(b, 3.0) *Опрос сети*

y1 =

2.7003

Погрешность восстановления по данным обучающей выборки в данном случае - 10%.

5 ИСПОЛЬЗОВАНИЕ SIMULINK ПРИ ПОСТРОЕНИИ НС

Пакет Neural Networks содержит ряд блоков, которые могут быть либо непосредственно использованы для построения НС в среде **Simulink**, либо применяться вместе с рассмотренной функцией gensim.

Для вызова набора блоков в командной строке необходимо набрать команду **neural**, после выполнения, которой появится окно (рис.11).

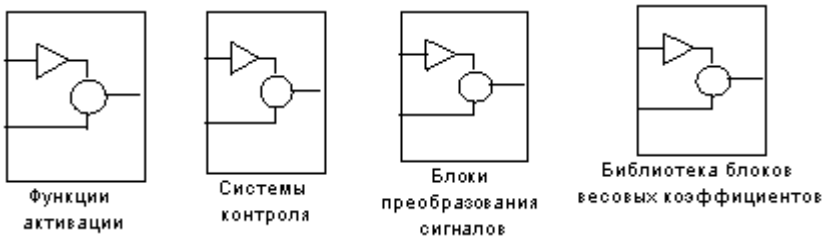


Рис.11 Основные нейросетевые блоки Simulink

Каждый из представленных на рис.11 блоков, в свою очередь, является набором некоторых блоков.

Блоки функции активации. Двойной щелчок на блоке приводит к появлению библиотеки блоков (рис.12). Каждый из этих блоков данной библиотеки преобразует подаваемый на него вектор в соответствующий вектор той же размерности.

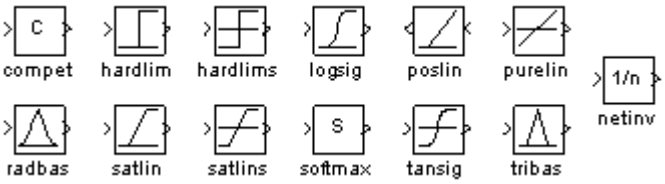


Рис.12 Библиотека блоков функций активации

Блоки преобразования входов сети. Функции данной группы реализуют функции накопления потенциала нейрона в виде поэлементного произведения или сумм взвешенных входов нейрона, а также вычисляют производные от таких произведений или сумм (рис.13).

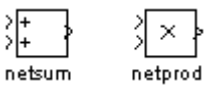


Рис.13. Окно библиотеки блоков преобразования сигналов

Блоки весовых коэффициентов (рис.14**).** Функции этой группы выполняют следующие операции: взвешивание и вычисление расстояний в сетях с топологией (dotprod – функция придания входам P некоторых весов W. Возвращает матрицу $Z = W \cdot P$; normprod – функция вычисления нормированного скалярного произведения (каждый элемент дополнительно делится на сумму элементов соответствующего столбца-сомножителя); dist – функция вычисления евклидова расстояния; negdist – отрицательное евклидово расстояние. Векторы в **Simulink** необходимо представлять как столбцы.

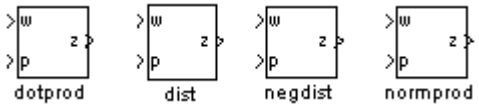


Рис.14 Библиотека блоков весовых коэффициентов

Блоки нейросетевых регуляторов. На рис.15 показан набор блоков, объединенных в библиотеку системы контроля. Данные блоки реализуют нейросетевые регуляторы трех различных структур – регулятор с предсказанием, регулятор, основанный на использовании модели нелинейной авторегрессии со скользящим средним (NARMA 1,2 Controller) и регулятор на основе эталонной модели, которые удобны при построении и исследовании моделей систем автоматического управления, а также блок просмотра результатов.

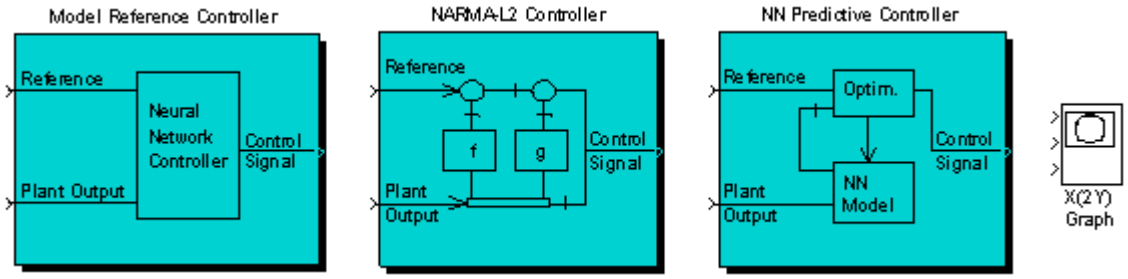


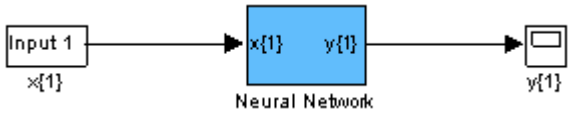
Рис.15. Окно библиотеки нейросетевых регуляторов

Формирование нейросетевых моделей. Основной функцией для формирования нейросетевых моделей в **Simulink** является функция gensim, записываемая в форме

Gensim (net,st), где net – имя созданной сети, st – интервал дискретизации (если НС не имеет задержек, ассоциированных с ее входами или слоями, значение данного аргумента устанавливается равны -1). Функция генерирует нейросетевой блок Simulink (рис.34) для последующего моделирования НС средствами этого пакета.

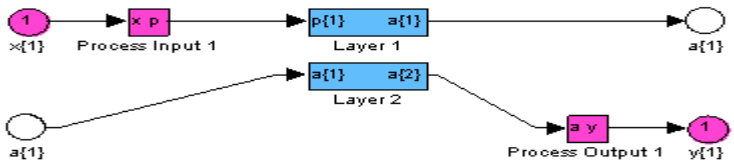
Пример 3. Создать двухслойную НС прямой передачи сигнала (функция создания многослойной НС записывается как **newff**) с вектором минимальных и максимальных значений входов [0,1], пятью нейронами в первом (скрытом) слое и одним нейроном – в выходном слое, а затем сформировать S-модель такой НС. Данная задача решается следующим образом:

```
>> net=newff([0 1],[5 1]); % Создание новой НС
>> gensim(net)
```



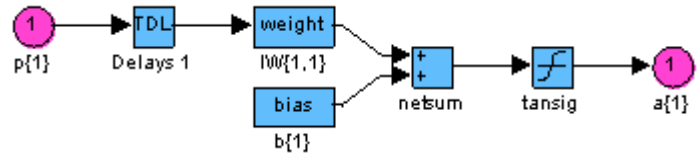
Результат выполнения функции gensim отображен на рис.16.

Рис.16. Результат выполнения функции gensim

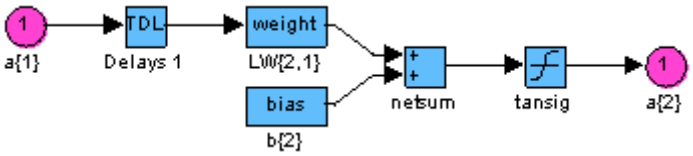


Дважды щелкая на блоке Neural Network, а затем на блоках Layer1 и Layer2, можно получить детальную информацию о структуре сети (рис.17)

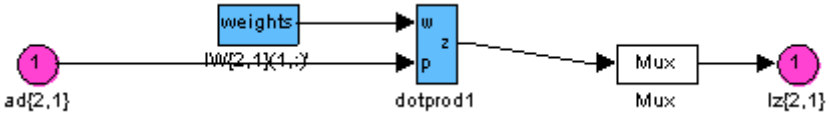
Структура созданной НС



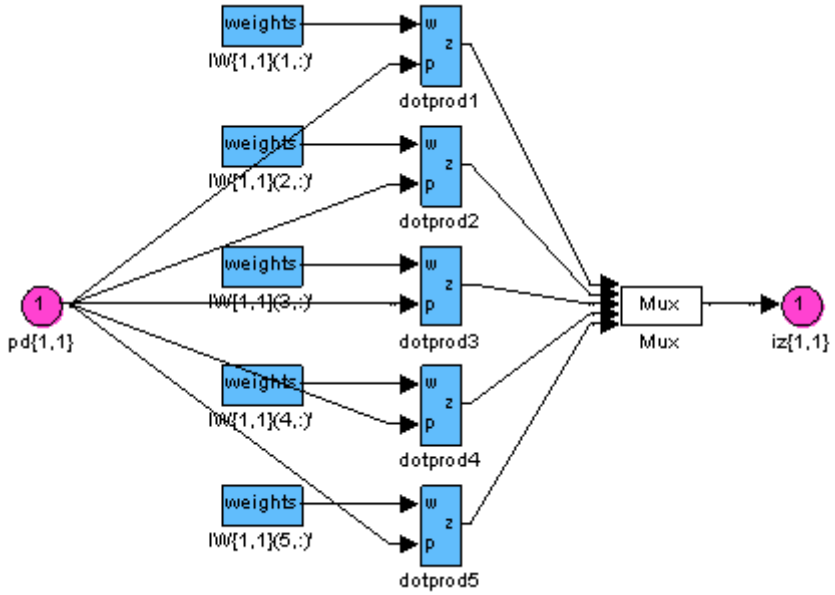
Структура 1-го слоя



Структура 2-го слоя



Структура выходного слоя



Структура скрытого слоя

Рис.17 Структура созданной НС с детальной графической информацией

Пример 4. Пусть входной и целевой векторы имеют вид

p=[1 2 3 4 5];

t=[1 3 5 7 9];

Создадим линей НС и протестируем ее по данным обучающей выборки:

>> p=[1 2 3 4 5];

>> t=[1 3 5 7 9];

>> net=newlind(p,t); *создание НС*

>> y=sim(net,p) *опрос сети*

y =

1.0000 3.0000 5.0000 7.0000 9.0000

Запустим **Simulink** командой

>> gensim(net,-1)

Это приведет к открытию блок-диаграммы, показанной на рис.34. В данном случае блок Input1является стандартным боком задания константы (Constant). Изменим значение по умолчанию на 4 и нажмем кнопку ОК, затем кнопку Start в панели инструментов. Для вывода нового значения необходимо дважды щелкнуть на правом значке блока y(1), оно будет равно 7.

С созданными сетями можно проводить различные эксперименты, возможные в среде **Simulink**, например, возможно встраивание нейросетевого регулятора в систему управления и моделирование последней.

6 СИСТЕМА АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ С НЕЙРОСЕТЕВЫМ РЕГУЛЯТОРОМ НА ОСНОВЕ ЭТАЛОННОЙ МОДЕЛИ

В качестве примера моделирования в среде Simulink систем управления с использованием нейросетевых регуляторов используем пример, иллюстрирующий систему управления с эталонной моделью реактора полного перемешивания (рис.18).

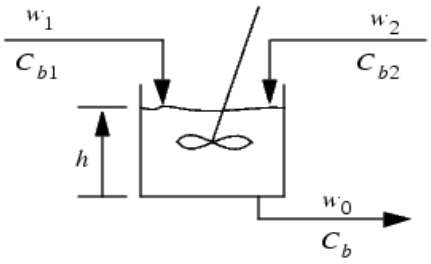


Рис.18 Реактор полного перемешивания (w_1, w_2, w_0 – расходы, h – уровень, C_{b1}, C_{b2}, C_b – концентрации компонентов на входе и выходе из реактора, соответственно).

Изменения концентрации в реакторе и на выходе из него, а также уровень жидкости в реакторе (динамическая модель системы) представлены следующей системой дифференциальных уравнений (получены из уравнений материального баланса):

$$\frac{dC_b(t)}{dt} = (C_{b1} - C_b(t))\frac{w_1(t)}{h(t)} + (C_{b2} - C_b(t))\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{(1 + k_2 C_b(t))^2}$$

$$\frac{dh(t)}{dt} = w_1(t) + w_2(t) - 0.2\sqrt{h(t)}$$

где $k_1 = 1$ и $k_2 = 1$ - константы, $C_{b1} = 24.9$, $C_{b2} = 0.1$.

Цель автоматического управления - стабилизировать концентрацию на выходе, регулируя поток $w_1(t)$(расход $w_2(t) = 0.1$ является величиной постоянной). Уровень резервуара $h(t)$ не является управляемым параметром.

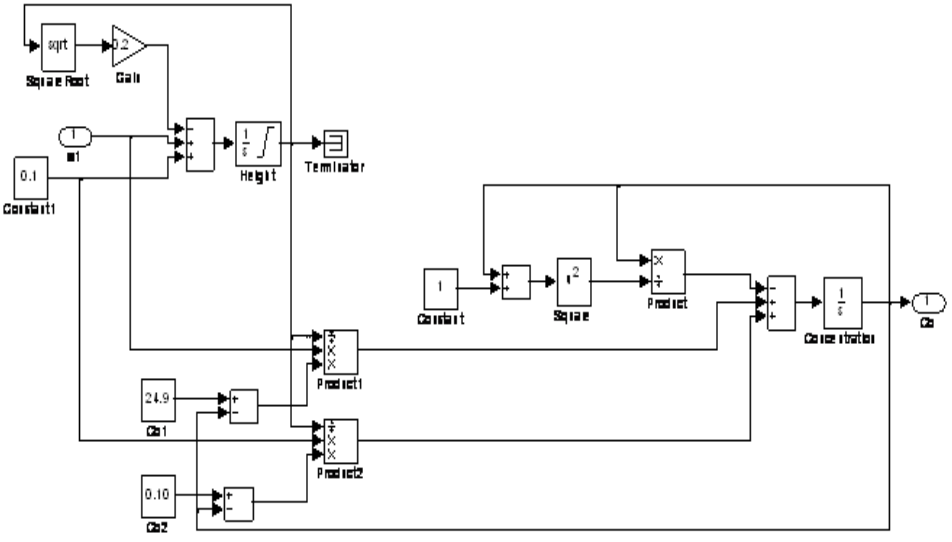


Рис.19. Структурная динамическая модель реактора

Соответствующая динамическая модель объекта регулирования приведена на рис.19.

Структурная схема, поясняющая принцип построения системы управления показана на рис.20.

Автоматический регулятор состоит из нейронной сетевой модели объекта и блока оптимизации рис.21).

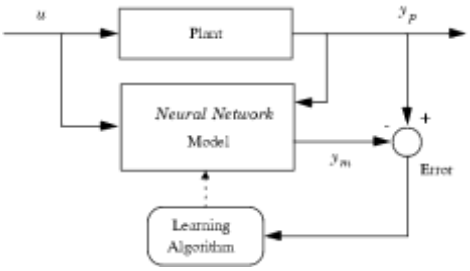


Рис.20. Система управления (Plant – объект управления, Learning algorithm – обучающий алгоритм

Блок оптимизации определяет оптимальное значение управляющего параметра (u), который подается на вход объекта (рис. 22).

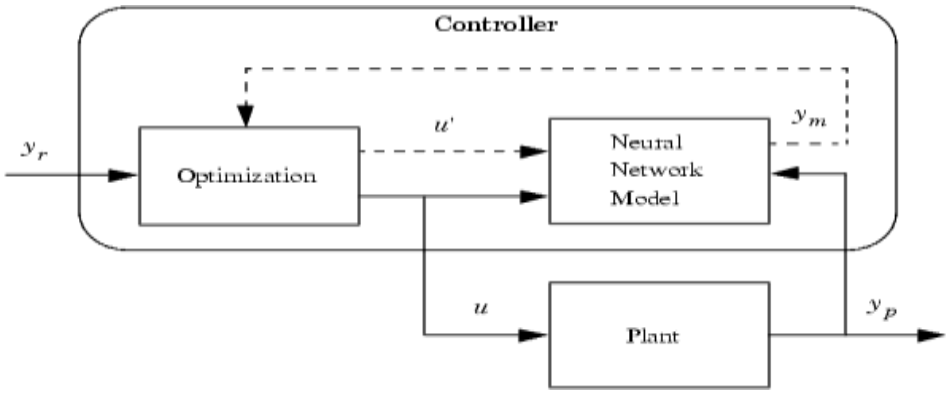


Рис.21. Блок оптимизации

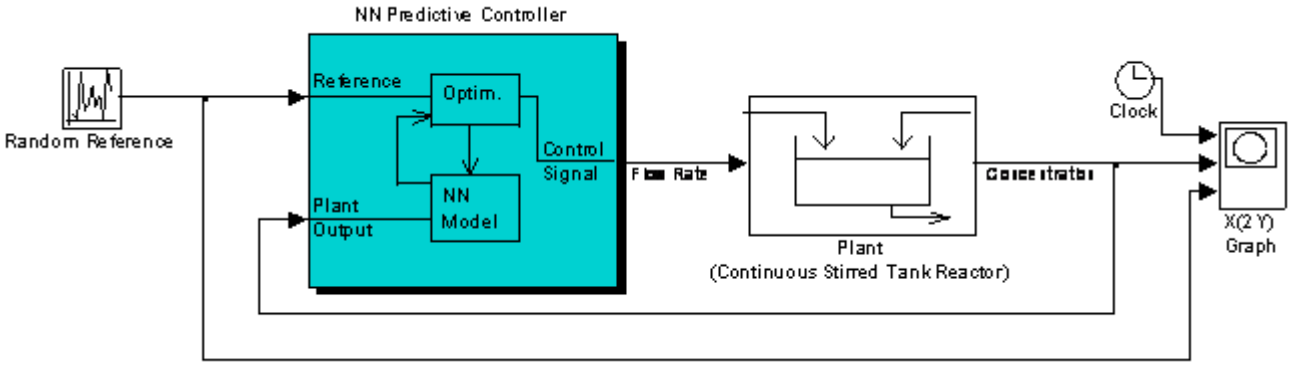


Рис.22. Структура нейросетевой системы управления ректора полного перемешивания (random reference - программа генерирует обучающиеся данные, применяя случайные сигналы к Simulink модели объекта).

На рис.23 показано как строится НС модели регулятора в окне Neural Network Predictive Control и окне Plant Identification (ρ – весовой фактор, α – параметр оптимизации).

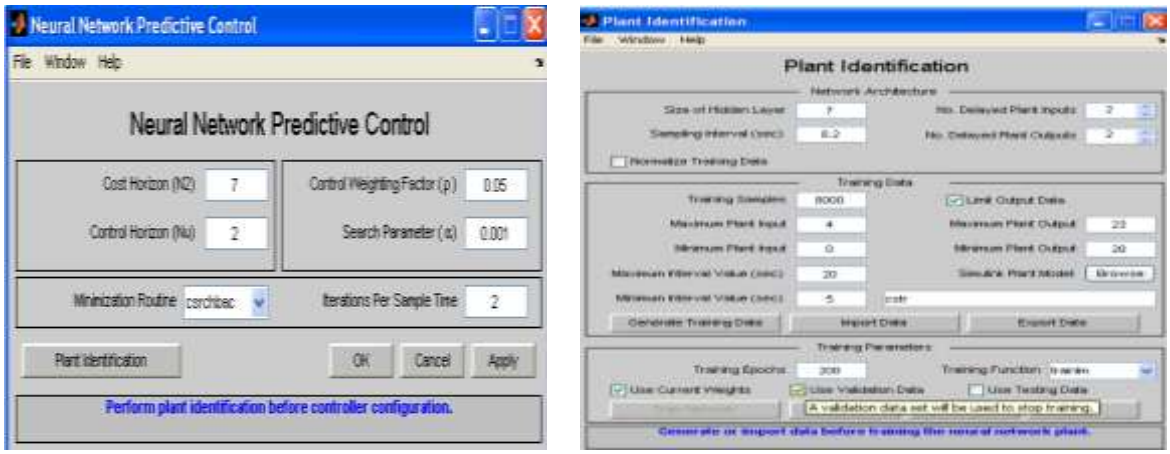


Рис.23 Построение нейросетевой модели регулятора

На рис.24 показан график регулирования концентрации продукта на выходе реактора.

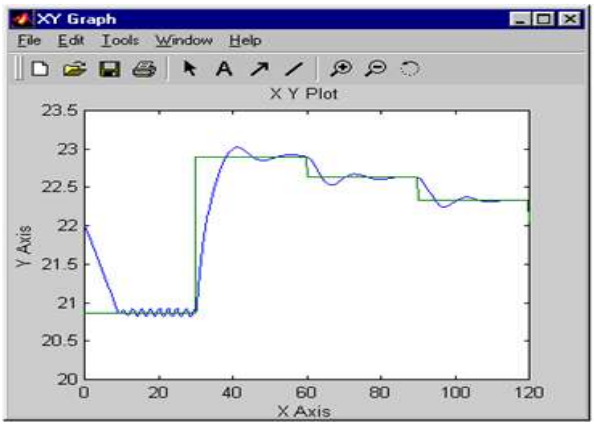


Рис.24 График регулирования концентрации продукта на выходе реактора

7 НЕЧЕТКИЕ НЕЙРОННЫЕ СЕТИ

Нечеткие нейронные сети или гибридные сети призваны объединить в себе достоинства нейронных сетей и систем нечеткого вывода. **С одной стороны**, они позволяют разрабатывать и представлять модели систем в форме правил нечетких продукций, которые обладают наглядностью и простотой содержательной интерпретации. **С другой стороны**, для построения правил нечетких продукций используются методы нейронных сетей, что является более удобным и менее трудоемким процессом для системных аналитиков. В последнее время аппарат гибридных сетей повсеместно признается специалистами как один из наиболее перспективных для решения слабо или плохо структурированных задач прикладного системного анализа.

Ниже рассмотрены особенности построения и использования гибридных сетей, реализованные в пакете Fuzzy Logic Toolbox.

Как уже говорилось выше в настоящее время предложены различные схемы классификации нейронных сетей и соответствующие алгоритмы их обучения. Одним из самых распространенных алгоритмов обучения является так называемый алгоритм обратного распространения ошибки (back propagation). Этот алгоритм представляет собой итеративный градиентный алгоритм минимизации среднеквадратичного отклонения значений выхода от желаемых значений (минимизации ошибки) в многослойных нейронных сетях.

Нечеткая нейронная сеть – это многослойная НС, в которой слои выполняют функции элементов системы нечеткого вывода. Характерной особенностью этих сетей является возможность использования нечетких правил вывода для расчета выходного сигнала. Нейроны данной сети характеризуются набором параметров, настройка которых производится в процессе обучения, как у обычных НС.

Рассмотрим пример реализации нечеткой нейронной сети на базе алгоритма Сугено:

1-й слой осуществляет фазификацию, нелинейные функции $\mu_{r,j}(x_j)$, где r – номер продукционного правила, j – номер компонента входного вектора $\bar{x}$ соответствуют функциям принадлежности предпосылок правил. Настраиваемые параметры данного слоя – параметры используемых функций принадлежности.

2-й слой осуществляет вычисление результирующих функций принадлежности предпосылок нечетких правил.

3-й слой, состоящий из двух нейронов-сумматоров, осуществляет суммирование и взвешенное суммирование выходных сигналов слоя 2. Параметрами данного слоя являются весовые коэффициенты w_r.

4-й слой, состоящий из единственного выходного нейрона, реализует операцию деления $z = f_1 / f_2$ и не содержит настраиваемых параметров.

Обучение нечетких сетей, так же как и классических сетей, может проводиться либо по алгоритму с учителем, основанному на минимизации целевой функции, либо по алгоритму

самоорганизации.

Выбор вида и структуры нейронной сети предопределяется спецификой решаемой задачи. При этом для решения отдельных типов практических задач разработаны оптимальные конфигурации нейронных сетей, которые наиболее адекватно отражают особенности соответствующей проблемной области. Дальнейшим развитием нейронных сетей являются так называемые гибридные сети.

7.1 Гибридная сеть как адаптивная система нейро-нечеткого вывода

Гибридная сеть представляет собой многослойную нейронную сеть специальной структуры без обратных связей, в которой используются обычные (не нечеткие) сигналы, веса и функции активации, а выполнение операции суммирования $s = \sum_{i=1}^{n} w_i x_i + b_i$ основано на использовании фиксированной Т-нормы, Т-конормы или некоторой другой непрерывной операции. При этом значения входов, выходов и весов гибридной нейронной сети представляют собой вещественные числа из отрезка [0,1].

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

В пакете Fuzzy Logic Toolbox системы MATLAB гибридные сети реализованы в форме так называемой адаптивной системы нейро-нечеткого вывода ANFIS. С одной стороны, гибридная сеть ANFIS представляет собой нейронную сеть с единственным выходом и несколькими входами, которые представляют собой нечеткие лингвистические переменные. При этом термы входных лингвистических переменных описываются стандартными для системы MATLAB функциями принадлежности, а термы выходной переменной представляются линейной или постоянной функцией принадлежности. С другой стороны, гибридная сеть ANFIS представляет собой систему нечеткого вывода FIS типа Сугено нулевого или первого порядка, в которой каждое из правил нечетких продукций имеет постоянный вес, равный 1. В системе MATLAB пользователь имеет возможность редактировать и настраивать гибридные сети ANFIS аналогично системам нечеткого вывода, используя все рассмотренные ранее средства пакета Fuzzy Logic Toolbox.

7.2 Реализация ANFIS в среде MATLAB

В пакете Fuzzy Logic Toolbox гибридные сети реализованы в форме адаптивных систем нейро-нечеткого вывода ANFIS. При этом разработка и исследование гибридных сетей оказывается возможной:

- в интерактивном режиме с помощью специального графического редактора адаптивных сетей, получившего название редактора ANFIS;
- в режиме командной строки с помощью ввода имен соответствующих функций с необходимыми аргументами непосредственно в окно команд системы MATLAB. Для работы в режиме командной строки предназначены специальные функции.

Редактор ANFIS позволяет создавать или загружать конкретную модель адаптивной системы нейро-нечеткого вывода, выполнять ее обучение, визуализировать ее структуру, изменять и настраивать ее параметры, а также использовать настроенную сеть для получение результатов нечеткого вывода. Графический интерфейс редактора ANFIS вызывается функцией anfisedit из командной строки (рис.25).

С помощью данного редактора осуществляется создание или загрузка структуры гибридной системы, просмотр структуры, настройка ее параметров, проверка качества функционирования такой системы.

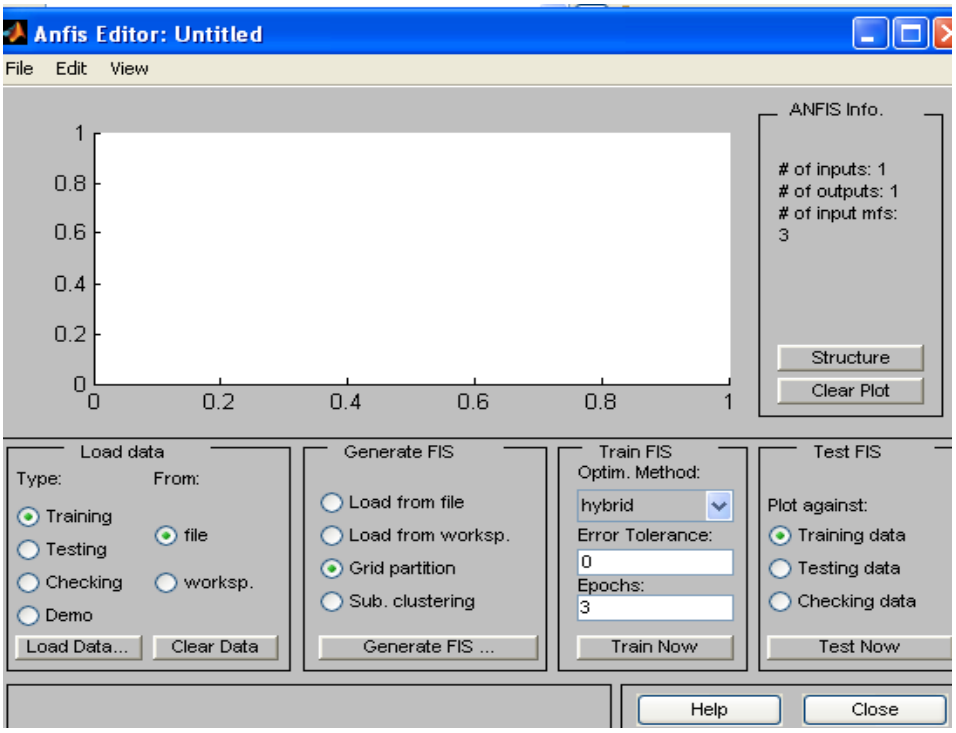


Рис. 25 Окно редактора гибридных систем

Основную часть графического интерфейса занимает окно визуализации данных, которое расположено ниже главного меню. Для вновь создаваемой гибридной сети это окно не содержит никаких данных.

Для создания гибридной сети необходимо загрузить данные. Для этой цели следует воспользоваться кнопкой Load Data в левой нижней части графического окна. При этом данные могут быть загружены из внешнего файла (disk) или из рабочей области (worksp.). В первом случае необходимо предварительно создать файл с исходными данными (файл с расширением .dat), который представляет собой обычный текстовый файл. При этом исходные данные представляют собой обычную числовую матрицу размерности $m \times (n+1)$, в которой количество строк m соответствует объему выборки, первые n столбцов значениям входных переменных модели, а последний столбец значению выходной переменной. Согласно правилам системы MATLAB отдельные значения матрицы отделяются пробелам, а каждая строка матрицы завершается символом "перевод каретки" (клавиша <Enter».

Загружаемые исходные данные могут быть одного из следующих типов:

- обучающие данные (Training) - обязательные данные, которые используются для построения гибридной сети;
- тестовые данные (Testing) - необязательные данные, которые используются для тестирования построенной гибридной сети с целью проверки качества функционирования построенной гибридной сети;
- проверочные данные (Checking) - необязательные данные, которые используются для про верки построенной гибридной сети с целью выяснения факта переобучения сети;
- демонстрационные данные (Demo) - позволяют загрузить один из демонстрационных примеров гибридной сети.

После загрузки обучающих данных их структура будет отображена в рабочем окне редактора ANFIS. При этом каждой строке данных соответствует отдельная точка графика, которая для обучающих данных изображается кружком. На горизонтальной оси указываются порядковый номер (индекс) отдельной строки данных, а вертикальная ось служит для указания значений выходной переменной.

После подготовки и загрузки обучающих данных можно сгенерировать структуру

системы нечеткого вывода FIS типа Сугено, которая является моделью гибридной сети. Для этой цели следует воспользоваться кнопкой Generate FIS в нижней части рабочего окна редактора. При этом две первые опции относятся к предварительно созданной структуре гибридной сети, а две последних к форме разбиения входных переменных модели.

Загрузить структуру уже созданной FIS можно либо с диска (Load from disk), либо из рабочей области (Load from worksp.). При создании структуры новой FIS можно независимо разбить все входные переменные на области их значений (Grid partition) или воспользоваться процедурой субтрактивной кластеризации для предварительного разбиения значений входных переменных на кластеры близких значений (Sub. clustering).

После нажатия кнопки Generate FIS вызывается диалоговое окно с указанием числа и типа функций принадлежности для отдельных термов входных переменных и выходной переменной (рис.26). В этом случае можно выбрать любой тип функций принадлежности из реализованных в системе MA TLAB.

После генерации структуры гибридной сети можно визуализировать ее структуру, для чего следует нажать кнопку Structure в правой части графического окна. Структура полученной в результате системы нечеткого вывода FIS отображается в отдельном окне (рис.27).

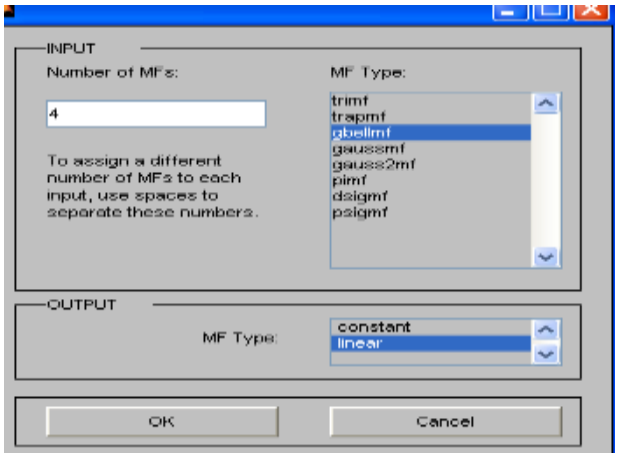


Рис.26 Диалоговое окно для задания количества и типа функций принадлежности

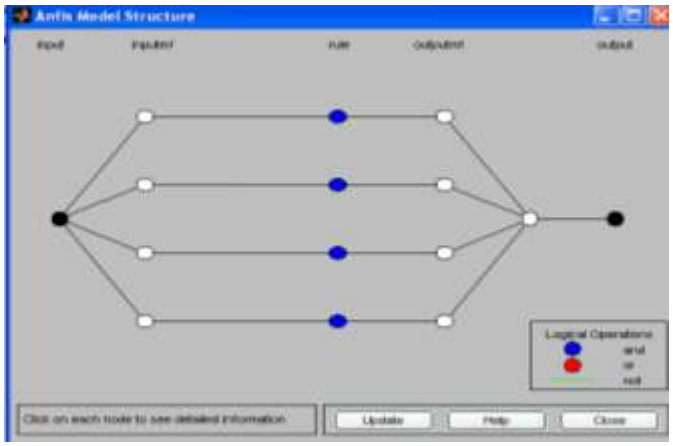


Рис.27 Структура сгенерированной системы нечеткого вывода

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

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

2. Установить уровень ошибки обучения (Error Tolerance) по умолчанию значение 0 (изменять не рекомендуется).

3. Задать количество циклов обучения (Epochs) по умолчанию значение 3 (рекомендуется увеличить и для рассматриваемого примера задать его значение равным 40).

Для обучения сети следует нажать кнопку Train Now. При этом ход процесса обучения иллюстрируется в окне визуализации в форме графика зависимости ошибки от количества циклов обучения.

В этом случае на рис.30 изображена зависимость ошибки проверки от количества циклов обучения, а на нижнем графике зависимость ошибки обучения от количества циклов обучения (знаком "*").

Аналогично могут быть выполнены дополнительные этапы тестирования и проверки гибридной сети, для которых необходимо предварительно загрузить соответствующие данные.

7.3 Примеры решения задачи нейро-нечеткого вывода

Для иллюстрации процесса разработки гибридной сети в системе MATLAB рассмотрим задачу построения адаптивной системы нейро-нечеткого вывода для аппроксимации некоторой зависимости, которая описывается математической функцией $y = x^3$. Этот пример позволяет не только уточнить содержание и последовательность этапов разработки, но и оценить точность полученной нечеткой модели посредством сравнения прогнозируемых модельных значений с известными заранее значениями соответствующей функции.

Общая последовательность процесса разработки модели гибридной сети может быть представлена в следующем виде.

1. Для начала с помощью редактора отладчика m-файлов подготовим обучающие данные, которые содержат 9 строк пар "значение входной переменной - значение выходной переменной" следующего вида (рис.28). Сохраним обучающие данные во внешнем файле с именем function1.dat.

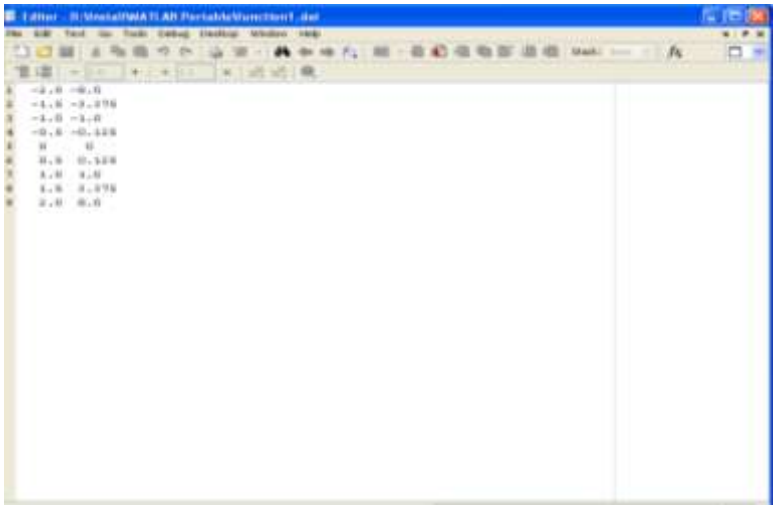


Рис.28 Обучающие данные для примера построения гибридной сети ANFIS, представляющей функцию $y = x^3$.

2. Далее загрузим этот файл с обучающими данными в редактор ANFIS (рис.29). В рабочем окне редактора будет изображен график, форма которого аналогична исходной математической функции.

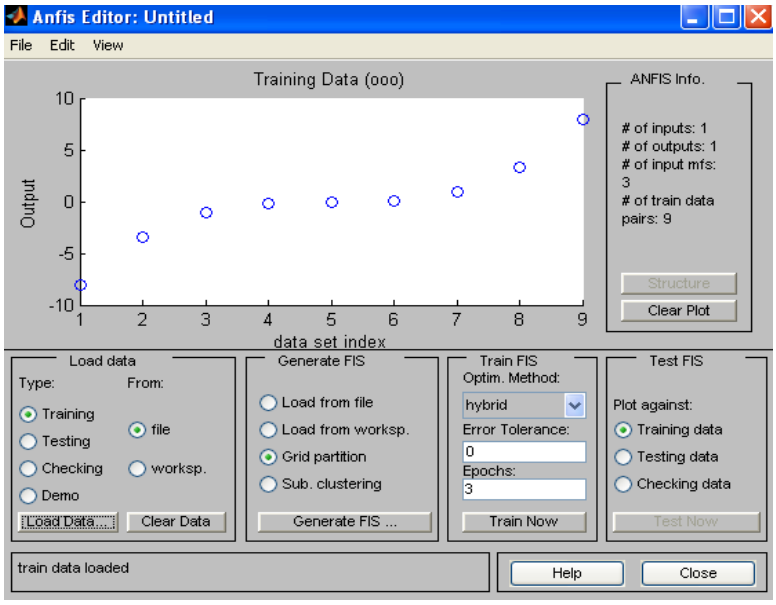
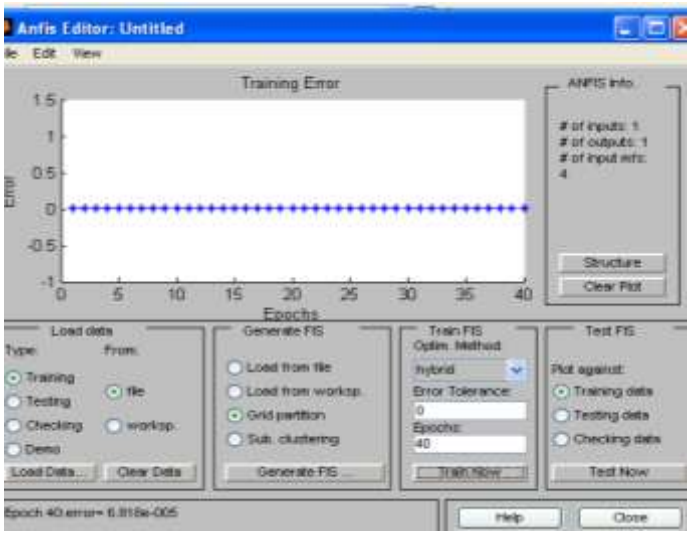


Рис.29 Графический интерфейс редактора ANFIS после загрузки файла function.dat с обучающими данными

3. Поскольку в данном случае отсутствуют тестовые и проверочные данные, можно сразу приступить к генерации структуры системы нечеткого вывода FIS. Установив параметры генерации, получим структуру FIS, вид которой также совпадает с изображенной на рис.27.
4. Теперь можно перейти к обучению сгенерированной системы нечеткого вывода. Для этого оставим без изменения предложенные системой MATLAB по умолчанию метод обучения (гибридный) и уровень ошибки (0), а количество циклов обучения изменим на 40. После обучения сети в рабочем окне редактора ANFIS будет изображен график изменения ошибки в ходе выполнения отдельных циклов обучения (рис.30).
5. Выполнить анализ точности построенной нечеткой модели гибридной сети можно с помощью просмотра поверхности соответствующей системы нечеткого вывода (рис.31).



Визуальный анализ изображенного графика с точным графиком функции $y = x^3$ позволяет судить о достаточно высокой степени их совпадения, поможет свидетельствовать об адекватности построенной нечеткой модели гибридной сети.

Рис.30. График зависимости ошибки обучения от количества циклов обучения

Анализ адекватности построенной модели можно выполнить с помощью просмотра правил соответствующей системы нечеткого вывода (рис.32).

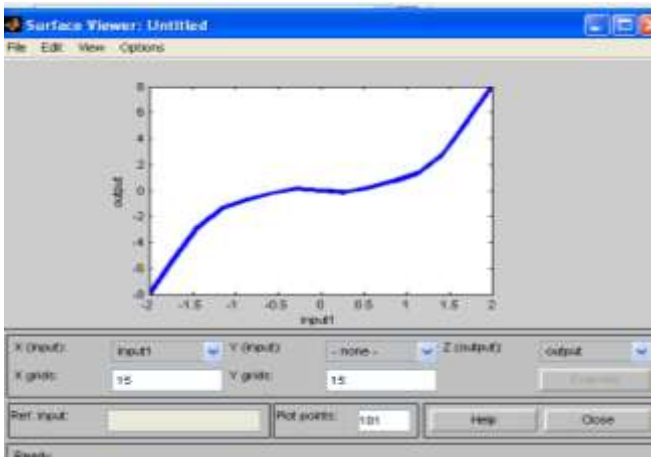


Рис.31. Графический интерфейс просмотра поверхности сгенерированной системы нечеткого вывода

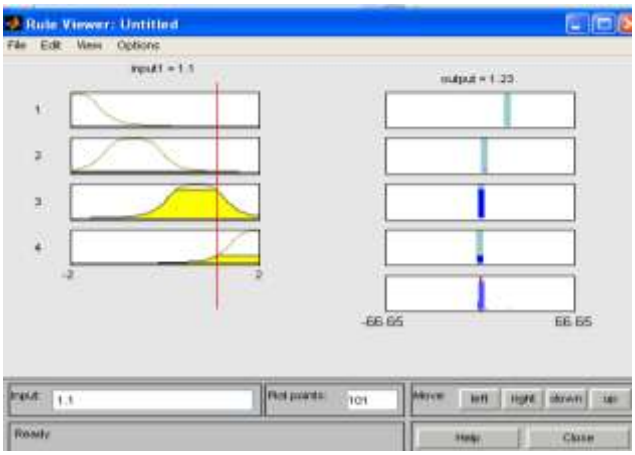


Рис.32. Графический интерфейс просмотра правил сгенерированной системы нечеткого вывода

Проверка построенной модели гибридной сети может быть выполнена для нескольких значений выходной переменной. С этой целью необходимо ввести конкретное значение в поле ввода Input (например, значение 1.1), после нажатия клавиши <Enter> с помощью построенной модели будет получено соответствующее значение выходной переменной (в данном случае значение 1.23). Сравнивая полученное значение с точным значением функции 1.331, получим относительную ошибку порядка 8%.

Менее удачной оказывается проверка для значения входной переменной 0.1, для которого построенная модель предлагает отрицательное значение -0,148. Очевидно, данный факт свидетельствует не в пользу адекватности построенной нечеткой модели и требует ее дополнительной настройки.

В общем случае дополнительная настройка модели может быть выполнена несколькими возможными способами. Наиболее приемлемыми из них представляются следующие.

1. Подготовка и загрузка большего по объему выборки файла с обучающими исходными данными.

2. Подготовка и загрузка дополнительного файла с проверочными исходными данными, сформированными для пар значений рассматриваемой математической функции, отсутствующих В выборке обучающих данных.

3. Редактирование типов и значений параметров функций принадлежности термов входной и выходной переменных с помощью редактора функций принадлежности системы MATLAB.

Проиллюстрируем третий способ дополнительной настройки построенной нечеткой модели гибридной сети. На первый взгляд он представляется наиболее естественным с точки зрения возможности визуального контроля выполняемых изменений параметров. С этой целью откроем редактор функций принадлежности и методом подбора изменим количественные значения параметров второй и третьей функции принадлежности входной переменной, поскольку именно они "работают" при получении некорректного значения выходной переменной: 0.148 для значения входной переменной 0.2. (рис.33).

После изменения параметров в поле Params для второй функции принадлежности на значения [0.55 2 0.437] и для третьей функции принадлежности на значения [0.65 1.9 0.667] получим более точное значение выходной переменной для исходного значения входной переменной 0.2 (рис.34).

Более эффективным способом настройки, а точнее модификации, данной нечеткой модели оказался первый. С этой целью необходимо увеличить объем обучающей выборки до 17 пар значений.

В заключение следует отметить, что даже простейшие рассмотренные примеры отражают творческий характер процесса построения и анализа моделей гибридных сетей. При этом выбор того или иного способа дополнительной настройки нечетких моделей зависит не только от специфики решаемой задачи, но и от объема доступной выборки обучающих и проверочных данных. В случае недостаточной информации обучающих данных использование гибридных сетей может оказаться вообще нецелесообразным, поскольку получить адекватную нечеткую модель, а значит и точный прогноз значений выходной переменной не представляется возможным.

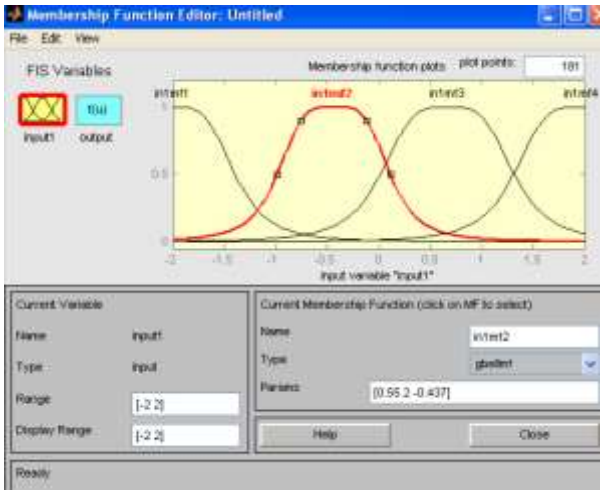


Рис.33 Графический интерфейс редактора функций принадлежности построенной системы нечеткого вывода

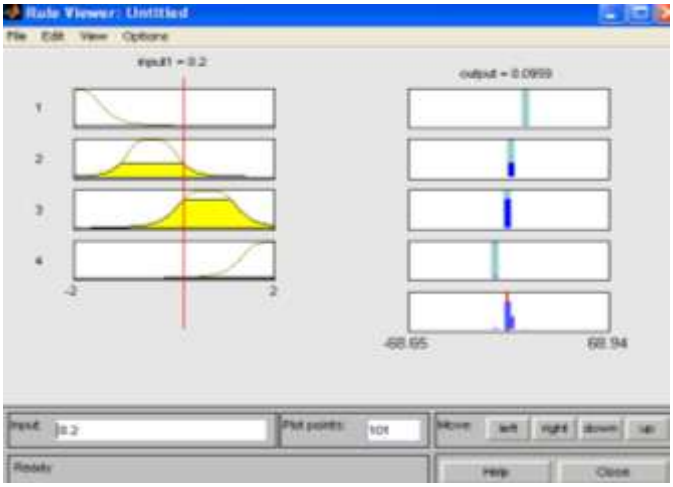


Рис.34 Результат ручной настройки параметров функций принадлежности входной переменной для рассматриваемой системы нечеткого вывода

Именно по этим причинам необходим предварительный анализ всех возможностей применяемых нечетких моделей для решения конкретных задач в той или иной проблемной области. Подобный анализ необходимо выполнять с системной точки зрения и с учетом всех складывающихся на данный момент обстоятельств. Только всесторонняя и полная оценка проблемной ситуации позволит разработать адекватную модель решения той или иной конкретной задачи нечеткого управления или принятия решений.

Анализ и прогнозирование валютных цен на финансовом рынке

В качестве второго примера построения и использования адаптивной системы нейро-нечеткого вывода рассмотрим процесс разработки нечеткой модели гибридной сети для решения задачи прогнозирования валютных цен на финансовом рынке.

Суть данной задачи состоит в том, чтобы, зная динамику изменения курсовой стоимости продажи некоторой валюты за фиксированный интервал времени, предсказать значение ее курсовой стоимости на определенный момент времени в будущем. При этом характерной

особенностью динамики изменения курса (тренда) является наличие двух основных тенденций в колебаниях соответствующих цен.

С одной стороны, наблюдается общее долгосрочное повышение курсовой стоимости, связанное с величиной инфляции. С другой стороны, наблюдается краткосрочное колебание цен, связанное с целым рядом случайных факторов, адекватное представление которых в той или иной формальной модели вряд ли возможно.

Традиционно для решения данной задачи применяются различные модели технического анализа, основанные на использовании различных индикаторов. В то же время наличие неявных тенденций в динамике изменения курсовой стоимости валют позволяет применить модель адаптивных нейронечетких сетей.

В качестве исходных данных можно воспользоваться информацией о динамике курса ЦБ РФ по валюте "Доллар США" (USD) за некоторый временной интервал. Возьмем значения курсовой стоимости USD за I единицу в период с I октября 2002 с. по 10 декабря 2002 с. Данную информацию для удобства дальнейшей работы представим в табличной форме (табл. 3).

Табл. 3. Динамика курса иностранной валюты (USD) в период с 1.10.02 по 10.12.02

Дата	Курс USD	Дата	Курс USD	Дата	Курс USD
01.10.02	31.6827	01.11.02	31.7701	03.12.02	31.8547
02.10.02	31.6919	02.11.02	31.7646	04.12.02	31.8584
03.10.02	31.6982	05.11.02	31.7744	05.12.02	31.8596
04.10.02	31.6809	06.11.02	31.7909	06.12.02	31.8578
05.10.02	31.68	07.11.02	31.7756	07.12.02	31.86
08.10.02	31.6795	11.11.02	31.7756	10.12.02	31.8597
09.10.02	31.6799	12.11.02	31.7756		
10.10.02	31.6803	13.11.02	31.8226		
11.10.02	31.6685	14.11.02	31.8157		
12.10.02	31.6703	15.11.02	31.8203		
15.10.02	31.6703	16.11.02	31.8225		
16.10.02	31.6762	19.11.02	31.8231		
17.10.02	31.6767	20.11.02	31.8224		
18.10.02	31.6761	21.11.02	31.823		
19.10.02	31.6727	22.11.02	31.8248		
22.10.02	31.6973	23.11.02	31.8222		
23.10.02	31.7272	26.11.02	31.8416		
24.10.02	31.7159	27.11.02	31.8382		
25.10.02	31.7109	28.11.02	31.84		
26.10.02	31.7314	29.11.02	31.84		
29.10.02	31.7411	30.11.02	31.8424		
30.10.02	31.6977				
31.10.02	31.7408				

Предположим, что нечеткая модель гибридной сети будет содержать 4 входных переменных. При этом первая входная переменная будет соответствовать курсу USD на текущий банковский день, вторая курсу USD на предыдущий банковский день, т. е. на день (i-1), где через i обозначен текущий банковский день. Тогда третья входная переменная будет соответствовать курсу USD на (i-2) банковский день, а четвертая курсу USD на (i-3) банковский день.

Соответствующие обучающие данные могут быть сведены в отдельную таблицу. Объем полученной таким образом обучающей выборки равен 40 (табл. 4), что соответствует динамике

курса USD в период с 4 октября 2002 по 29 ноября 2002 г.

При этом данные за декабрь 2002 не вошли в состав обучающей выборки и могут быть использованы для проверки адекватности построенной нечеткой модели.

Сохраним обучающую выборку во внешнем файле под именем priceUSD.dat. После этого откроем редактор ANFIS, в который загрузим этот файл с обучающими данными. Внешний вид редактора ANFIS с загруженными обучающими данными изображен на рис.35. Перед генерацией структуры системы нечеткого вывода типа Сугено после вызова диалогового окна свойств зададим для каждой из входных переменных по 3 лингвистических терма, а в качестве типа их функций принадлежности выберем треугольные функции (установленные системой MATLAB по умолчанию).

Табл. 4. Обучающие данные для построения модели гибридной сети

Первая входная переменная	Вторая входная переменная	Третья входная переменная	Четвертая входная переменная	Выходная переменная
31.6809	31.6982	31.6919	31.6827	31.68
31.68	31.6809	31.6982	31.6919	31.6795
31.6795	31.68	31.6809	31.6982	31.6799
31.6799	31.6795	31.68	31.6809	31.6803
31.6803	31.6799	31.6795	31.68	31.6685
31.6685	31.6803	31.6799	31.6795	31.6703
31.6703	31.6685	31.6803	31.6799	31.6703
31.6703	31.6703	31.6685	31.6803	31.6762
31.6762	31.6703	31.6703	31.6685	31.6767
31.6767	31.6762	31.6703	31.6703	31.6761
31.6761	31.6767	31.6762	31.6703	31.6727
31.6727	31.6761	31.6767	31.6762	31.6973
31.6973	31.6727	31.6761	31.6767	31.7272
31.7272	31.6973	31.6727	31.6761	31.7159
31.7159	31.7272	31.6973	31.6727	31.7109
31.7109	31.7159	31.7272	31.6973	31.7314
31.7314	31.7109	31.7159	31.7272	31.7411
31.7411	31.7314	31.7109	31.7159	31.6977
31.6977	31.7411	31.7314	31.7109	31.7408
31.7408	31.6977	31.7411	31.7314	31.7701
31.7701	31.7408	31.6977	31.7411	31.7646
31.7646	31.7701	31.7408	31.6977	31.7744
31.7744	31.7646	31.7701	31.7408	31.7909
31.7909	31.7744	31.7646	31.7701	31.7756
31.7756	31.7909	31.7744	31.7646	31.7756

31.7756	31.7756	31.7909	31.7744	31.7756
31.7756	31.7756	31.7756	31.7909	31.8226
31.8226	31.7756	31.7756	31.7756	31.8157
31.8157	31.8226	31.7756	31.7756	31.8203
31.8203	31.8157	31.8226	31.7756	31.8225
31.8225	31.8203	31.8157	31.8226	31.8231
31.8231	31.8225	31.8203	31.8157	31.8224
31.8224	31.8231	31.8225	31.8203	31.823
31.823	31.8224	31.8231	31.8225	31.8248
31.8248	31.823	31.8224	31.8231	31.8222
31.8222	31.8248	31.823	31.8224	31.8416
31.8416	31.8222	31.8248	31.823	31.8382
31.8382	31.8416	31.8222	31.8248	31.84
31.84	31.8382	31.8416	31.8222	31.84
31.84	31.84	31.8382	31.8416	31.8424

В качестве типа функции принадлежности выходной переменной зададим линейную функцию (рис.36).

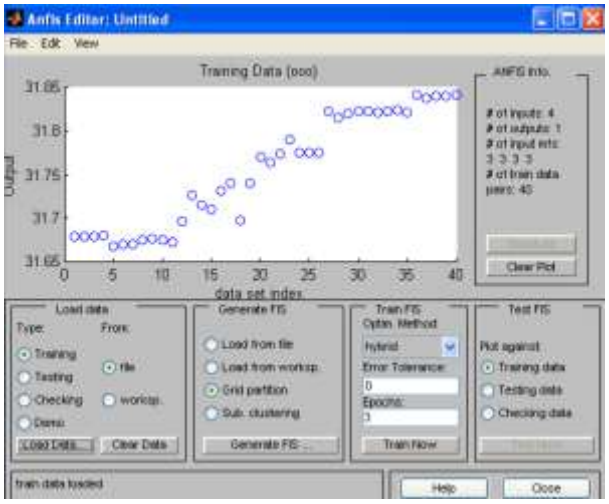


Рис.35 Графический интерфейс редактора ANFIS после загрузки обучающих данных

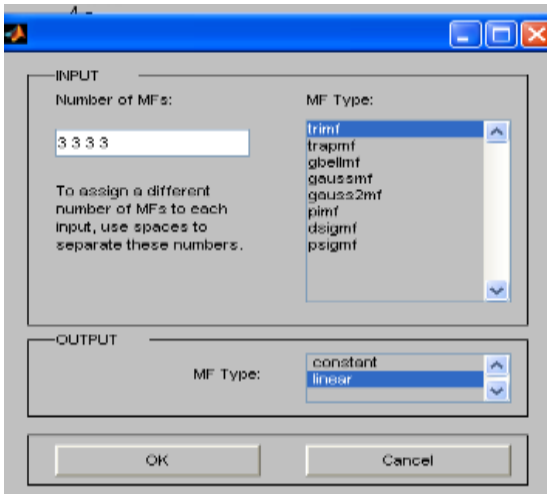


Рис.36 Диалоговое окно для задания количества и типа функций принадлежности

Для обучения гибридной сети воспользуемся гибридным методом обучения с уровнем ошибки О, а количество циклов обучения зададим равным 10. После окончания обучения данной гибридной сети может быть выполнен анализ графика ошибки обучения (рис.37), который показывает, что обучение практически закончилось после 3-го цикла.

После обучения гибридной сети можно визуально оценить структуру построенной нечеткой модели (рис.38). Очевидно, графическая наглядность данной модели оставляет желать лучшего, поскольку общее количество правил в разработанной адаптивной системе нейронечеткого вывода равно 81, (по затрудняет их визуальный контроль и оценку.

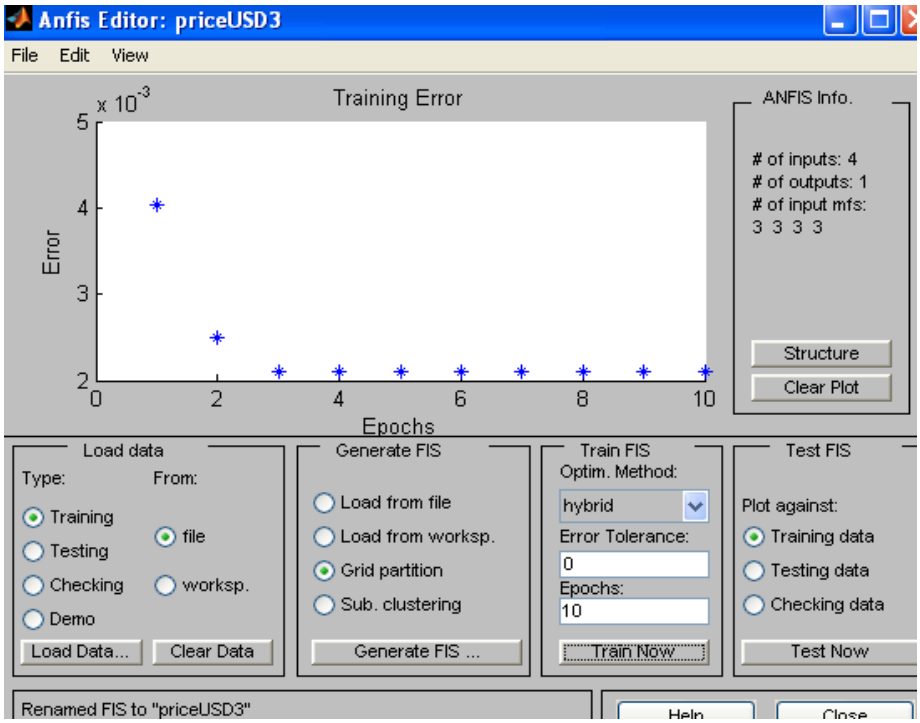


Рис.37. График зависимости ошибки обучения от количества циклов обучения

С помощью графических средств системы MATLAB можно выполнить контроль и настройку параметров функций принадлежности входных переменных и правил нечетких продукций. Для выполнения соответствующих операций можно воспользоваться редактором функций принадлежности (рис.39). Однако до проверки адекватности построенной нечеткой модели оставим все параметры функций принадлежности без изменений.

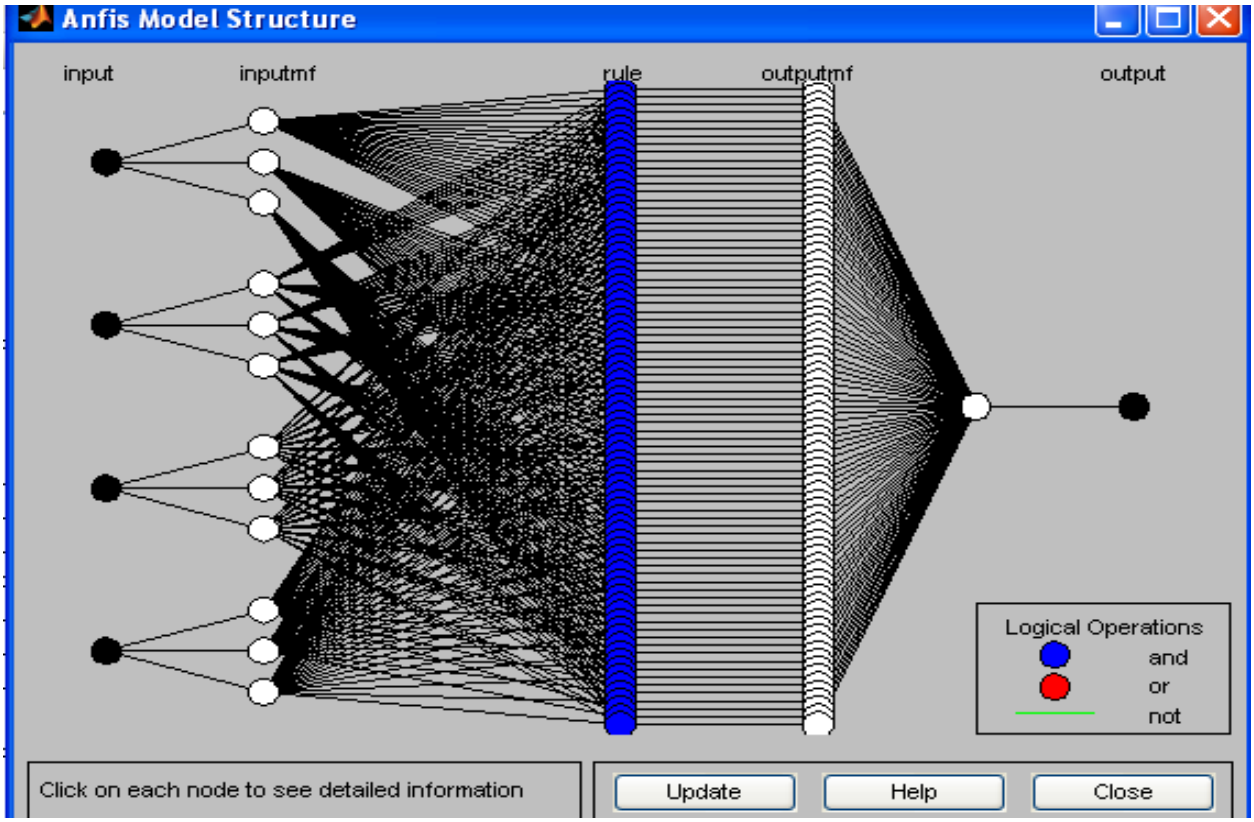


Рис.38. Структура сгенерированной системы нечеткого вывода

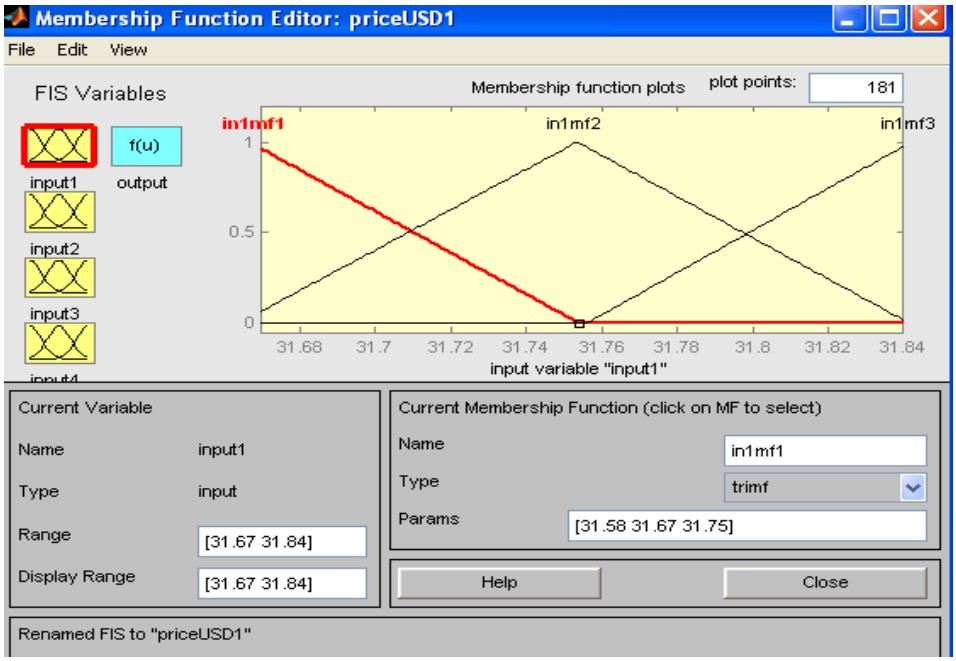


Рис.39 Графический интерфейс редактора функций принадлежности построенной системы нечеткого вывода для проверки первой входной переменной

Выполним проверку адекватности построенной нечеткой модели гибридной сети. Для этой цели сделаем ретроспективный прогноз значения курсовой стоимости USD на следующий банковский день, например, на 3 декабря 2002 г., считая для этого случая текущим банковским днем 30 ноября 2002 г. Поскольку точность количественных значений, обеспечиваемая графическими средствами пакета Fuzzy Logic ToolBox, является недостаточной для решения данной задачи, воспользуемся функцией командной строки evalfis. В качестве аргументов этой функции укажем вектор значений курсовой стоимости USD на текущий и 3 предшествующих банковских дня. Полный формат вызова этой функции будет следующим:

out = evalfis([31.8424 31.84 31.84 31.8382), priceUSD1),

где out - условное имя выходной переменной; evalfis – функция, выполняющая вывод в системе FIS; 31. 8424 - значение курсовой стоимости USD на 30.11.02; 31. 84 - значение курсовой стоимости USD на 29.11.02; 31.84 - значение курсовой стоимости USD на 28.11.02; 31.8382 - значение курсовой стоимости USD на 27.11.02; priceUSD1 - имя структуры FIS, предварительно загруженной в рабочую область системы MA TLAB.

```
>> out=evalfis([31.8424 31.84 31.84 31.8382],priceUSD1)

out =

31.8547
```

После выполнения этой команды с помощью разработанной нечеткой модели будет получено значение выходной переменной для 3.12.02, равное 31.8547. Сравнивая полученное значение с соответствующим значением из табл. 17.2, можно констатировать совпадение этих значений.

Таким образом, проверка построенной нечеткой модели гибридной сети показывает достаточно высокую степень ее адекватности реальным исходным данным, что позволяет сделать вывод о возможности ее практического использования для прогнозирования курсовой стоимости USD на финансовом рынке валют. В этом случае нечеткие модели адаптивных систем нейро-нечеткого вывода могут считаться новым и конструктивным инструментом технического

анализа финансовых рынков.

Рассмотренный подход является перспективным направлением для построения и использования соответствующих нечетких моделей прогнозирования цен других финансовых инструментов, таких как курсы других валют, акций компаний, фьючерсов и опционов. Действительно, общим для всех этих инструментов с позиций технического анализа является отсутствие априорных предположений о динамике колебаний соответствующих курсов цен, что вполне согласуется с исходными предпосылками построения нечетких моделей адаптивных систем нейро-нечеткого вывода.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Терехин, В.В. Основы моделирования в MATLAB. Часть 2. Simulink. Учебное пособие. / В.В. Терёхин. - Новокузнецк, 2004. - 304 с.
2. Дьяконов, В.П. Математические пакеты расширения MATLAB. Спец. справочник. / В.П. Дьяконов, В.В. Круглов. – СПб.: Питер, 2001. - 480 с.
3. Потёмкин, В.Г. Инструментальные средства MATLAB 5.x. / В.Г. Потёмкин. – М.: Диалог-МИФИ, 2000. - 336 с.
4. Дьяконов, В.П. Математические пакеты расширения MATLAB. Спец. справочник. / В.П. Дьяконов, В.В. Круглов. – СПб.: Питер, 2001. - 480 с.
5. Леоненков, А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. / А. В. Леоненков. – Санкт-Петербург, 2003. - 736 с.
6. Дьяконов, В. П. MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6 в. Инструменты искусственного интеллекта и биоинформатики. / В. П. Дьяконов, В. В. Круглов. – М.: Солон-Пресс, 2006. - 456 с.
7. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. - М.: Финансы и статистика, 2002. - 344 с.
8. Тэрано, Т. Прикладные нечеткие системы. / Т. Тэрано, К. Асаи, М. Сугено. – М.: Мир, 1993. - 368 с.
9. Кричевский, М.Л. Интеллектуальные методы в менеджменте. Нейронные сети. Нечеткая логика. Генетические алгоритмы. Динамические системы. / М.Л. Кричевский. – Питер, 2005. - 304 с.
10. Усков, А.А. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика. / А.А. Усков, А.В. Кузьмин. – М., 2004. - 143 с.
11. Круглов, В.В. Нечеткая логика и искусственные нейронные сети. / В.В. Круглов, М.И. Дли, Р.Ю. Голунов. – М.: Физматлит, 2001. - 224 с.

Учебно-методическое издание

БОГАТИКОВ Валерий Николаевич
ДРАНИШНИКОВ Леонид Васильевич
ПРОРОКОВ Анатолий Евгеньевич

ПОСТРОЕНИЯ СИСТЕМ УПРАВЛЕНИЯ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

Подписано в печать 29.11.11. Формат 60*841_1/_16
Бумага типографская. Офсетная печать. 2,4 уч.-изд.л.
Тираж 100 экз. Изд. № 26 КФ

Издательство Петрозаводского государственного университета
Петрозаводск, пр. Ленина, 33

Отпечатано подразделением оперативной полиграфии
Кольского филиала Петр ГУ
Апатиты, ул. Космонавтов, 3