МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

В.С. Ростовцев

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

УЧЕБНИК

Рекомендовано
Ученым советом ВятГУ
в качестве
учебного пособия

Киров 2014

Печатается по решению редакционно-издательского совета

Вятского государственного университета

УДК 004.891(07)

Р783

Рецензент: кандидат технических наук, доцент кафедры автоматики и телемеханики Вятского государственного университета В.И. Семёновых

Ростовцев В.С. Искусственные нейронные сети: учебник / В.С. Ростовцев. – Киров: Изд-во ВятГУ, 2014. – 208 с. Э4743

В учебном пособии рассмотрены теоретические и практические сведения по разработке, обучению и применению искусственных нейронных сетей для различных областей народного хозяйства.

Учебное пособие предназначено для студентов 1 курса магистратуры очного отделения направления 230101.68.05 «Информатика и вычислительная техника» при изучении дисциплины «Искусственные нейронные сети». Пособие может быть полезно студентам других специальностей при изучении нейросетевых технологий, а также для слушателей курсов повышения квалификации и профессиональной переподготовки.

Редактор А. В. Куликова

Подписано в печать Усл. печ. л. 13

Бумага для офисной техники Печать цифровая

Заказ № __ Тираж 50 Бесплатно

Текст напечатан с оригинала-макета, представленного автором.

610000, г. Киров, ул. Московская, 36

Оформление обложки, изготовление ПРИП ВятГУ

Оглавление

ВВЕДЕНИЕ

Нейронные сети способны обучаться распознаванию образов, пониманию речи, предсказанию погоды и управлению системами различной сложности. Эта новая технология должна сыграть важную роль в управлении энергетическими системами, индустриальными объектами, интеллектуальными роботами и во многих других практически важных приложениях. В частности нейронные сети интересны как основа для разработки параллельных архитектур.

Наряду с развитием персональных ЭВМ, сетей ЭВМ и высокопроизводительных суперЭВМ традиционной архитектуры в последние годы существенно повысился интерес к разработке и созданию компьютеров нетрадиционного типа и, прежде всего, нейрокомпьютеров. Связано это с тем, что, несмотря на высокую производительность современных суперЭВМ, приближающуюся к предельно допустимой, все ещё остается много практически важных проблем, для решения которых нужны более мощные и более гибкие вычислительные средства. Они необходимы для глобального моделирования процессов в экосистемах, при решении задач нейрофизиологии, искусственного интеллекта, метеорологии, сейсмологии и т.п. Необходимы они и при создании систем управления адаптивных интеллектуальных роботов.

Управляющие ЭВМ таких роботов должны воспринимать большие объемы информации, поступающей от многих параллельно функционирующих датчиков, эффективно обрабатывать эту информацию и формировать управляющие воздействия на исполнительные системы в реальном масштабе времени. Более того, управляющие компьютеры интеллектуальных роботов должны оперативно

решать задачи распознавания образов, самообучения, самооптимизации, самопрограммирования, т. е. те задачи, которые весьма сложны для традиционных ЭВМ и суперЭВМ. Поэтому остается актуальной необходимость в поиске новых подходов к построению высокопроизводительных ЭВМ нетрадиционной архитектуры. Среди таких подходов центральное место занимает нейрокомпьютерный подход.

Его суть состоит в разработке принципов построения новых мозгоподобных архитектур сверхпроизводительных вычислительных систем – нейрокомпьютеров. Подобно мозгу, такие системы должны обладать глобальным параллелизмом, самообучением, самооптимизацией, самопрограммированием и другими свойствами биологических систем. Ожидается, что нейрокомпьютеры в принципе смогут решить многие из тех проблем, которые сдерживают дальнейшее развитие научно-технического прогресса.

По современным представлениям нейрокомпьютер (НК) – это система, предназначенная для организации нейровычислений путем воспроизведения информационных процессов, протекающих в нейронных сетях мозга. Структурной единицей НК служит специфический процессор – нейропроцессор (НП), имитирующий информационное функционирование отдельных нервных клеток – нейронов. Нейропроцессоры связываются друг с другом в нейроподобные структуры, имитирующие нейронные сети мозга. По этой причине, чем точнее НП воспроизводит информационную деятельность нервных клеток, и чем ближе конфигурации искусственных нейронных сетей к конфигурациям сетей естественных, тем больше шансов воспроизвести в НК самообучение, самопрограммирование и другие свойства живых систем[1,4,11-14].

1. ОСНОВНЫЕ ПОНЯТИЯ И ОБЛАСТИ ПРИМЕНЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

1.1. Определения и терминология

Нейрокомпьютинг или обработка данных с помощью нейроподобных сетей, реализованных на компьютерах либо в виде программ, либо аппаратным образом, в настоящее время все более широко используется для решения многих плохо формализуемых задач.

Нейрокомпьютинг (нейровычисления) относятся к сравнительно новому направлению информационных технологий, которое получило название Machine Leaning – ML, т. е. машинное обучение.

Это важное научное направление информатики обобщает результаты и идеи, связанные с нейросетевыми вычислениями, эволюционными и генетическими алгоритмами, нечеткими множествами, итерационными самонастраивающимися алгоритмами. Появление обобщающего термина Machine Leaning связывают с именем К. Самуэля, опубликовавшего в 1963 г. в сборнике Computers and Thought (Mc Craw-Hill, N. Y.) статью под названием «Some studies in machine leaning using the game of checkers» («Некоторые проблемы обучения машины на примере игры в шахматы»).

Принцип нейроподобных вычислений отличается от привычных стандартных методов последовательной и параллельной организации вычислений отсутствием необходимости программирования и, самое главное, возможностью использования механизмов обучения нейрокомпьютера для решения задач, как правило, плохо формализуемых, с нечетко заданными начальными данными и условиями.

Искусственные нейро́нные се́ти (ИНС) — математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети Мак-Каллока и Питтса[1]. Впоследствии, после разработки алгоритмов обучения, получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для

---

[1] *Мак-Каллок У. С., Питтс В.,* Логическое исчисление идей, относящихся к нервной активности // В сб.: «Автоматы» под ред. К. Э. Шеннона и Дж. Маккарти. — М.: Изд-во иностр. лит., 1956. — с.363-384. (Перевод английской статьи 1943 г.)

робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма [1].

С точки зрения искусственного интеллекта, ИНС является основой философского течения *коннективизма* и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются, а обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами.

Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

Теория нейронных сетей включают широкий круг вопросов из разных областей науки: биофизики, математики, информатики, схемотехники и информационной технологии. Поэтому понятие "нейронные сети" детально определить сложно.

Искусственные нейронные сети (ИНС) – совокупность моделей биологических нейронных сетей.

ИНС представляют собой сеть искусственных нейронов, связанных между собой синаптическими соединениями. Сеть обрабатывает входную информацию и в процессе изменения своего состояния во времени формирует совокупность выходных сигналов.

Работа сети состоит в преобразовании входных сигналов во времени, в результате чего меняется внутреннее состояние сети и формируются

выходные воздействия. Обычно ИНС оперирует цифровыми, а не символьными величинами.

Все модели ИНС требуют обучения или расчёта весов связей. В общем случае, обучение – такой выбор параметров сети, при котором сеть лучше всего справляется с поставленной проблемой. Обучение — это задача многомерной оптимизации, и для ее решения существует множество алгоритмов.

1.2. Свойства биологических нейронных сетей

Искусственные нейронные сети набор математических и алгоритмических методов для решения широкого круга задач. Преимущества ИНС заключаются, во-первых, за счёт распараллеливания обработки информации; во-вторых, из способности самообучаться, т. е. создавать обобщения. Под термином обобщения понимается способность получать обоснованный результат на основании данных, которые не участвовали в процессе обучения [1–6].

Использование нейронных сетей обеспечивает следующие полезные свойства систем.

1. *Нелинейность (nonlinearity).* Искусственные нейроны могут быть линейными и нелинейными. Нейронные сети из нелинейных элементов являются нелинейными и могут решать более сложные задачи.

2. *Отображение входной информации в выходную (input-output mapping*). Одной из популярных парадигм обучения является обучение с учителем (*supervised learning*). Это предполагает изменение синаптических весов на основе учебных примеров, каждый из которых состоит из входных сигналов и соответствующего ему желаемого отклика нейросети.

Обучение проводится до тех пор, пока изменения синаптических весов не станут незначительными.

3. *Адаптивность (adaptivity).* ИНС обладают способностью адаптировать свои синаптические веса к изменениям окружающей среды.

4. *Очевидность ответа (evidential response).* При решении задачи классификации образов можно разработать нейронную сеть, которая повышает достоверность принимаемых решений и обеспечивает исключение сомнительных решений.

5. *Контекстная информация (contextual information).* Каждый нейрон сети потенциально может быть подвержен влиянию всех остальных её нейронов. Как следствие, существование нейронной сети непосредственно связано с контекстной информацией.

6. *Отказоустойчивость (fault tolerance).* При повреждении нейрона или его связи извлечение запомненной информации осуществляется с затруднением и потерей точности.

7. *Масштабируемость (VLSI Implementability).* Параллельная структура нейронной сети потенциально ускоряет решение некоторых задач и обеспечивает масштабируемость нейронных сетей в рамках технологии VLSI (very-large- scale-integrated), одним из преимуществ которой является возможность представить сложное поведение с помощью иерархической структуры.

8. *Единообразие анализа и проектирования (Uniformaty of analysis and desing).* Нейронные сети являются универсальным механизмом обработки информации. Одно и то же проектное решение нейронной сети может использоваться во многих предметных областях.

9. *Аналогия с нейробиологией (Neurobiological analogy*). Строение нейронных сетей определяется аналогией с человеческим мозгом, который демонстрирует отказоустойчивые параллельные вычисления для решения сложных задач.

К важнейшим свойствам биологических нейронных сетей относятся следующие:

1. *Параллельность обработки информации*. Каждый нейрон формирует свой выход только на основе своих входов и собственного внутреннего состояния под воздействием общих механизмов регуляции нервной системы.

2. *Способность к полной обработке информации*. Все известные человеку задачи решаются нейронными сетями. К этой группе свойств относятся ассоциативность (сеть может восстанавливать полный образ по его части), способность к классификации, обобщению, абстрагированию и множество других. Они до конца не систематизированы.

3. *Самоорганизация*. В процессе работы биологические НС самостоятельно, под воздействием внешней среды, обучаются решению разнообразных задач. Неизвестно никаких принципиальных ограничений на сложность задач, решаемых биологическими нейронными сетями. Нервная система сама формирует алгоритмы своей деятельности, уточняя и усложняя их в течение жизни. Человек пока не сумел создать систем, обладающих самоорганизацией и самоусложнением. Это свойство НС рождает множество вопросов. Ведь каждая замкнутая система в процессе развития упрощается, деградирует. Следовательно, подвод энергии к нейронной сети имеет принципиальное значение. Почему же среди всех диссипативных (рассеивающих энергию) нелинейных динамических систем только у живых существ, и, в частности, биологических нейросетей проявляется способность к усложнению? Какое принципиальное условие упущено человеком в попытках создать самоусложняющиеся системы?

4. *Биологические НС являются аналоговыми системами*. Информация поступает в сеть по большому количеству каналов и кодируется по пространственному принципу: вид информации определяется номером нервного волокна, по которому она передается.

Амплитуда входного воздействия кодируется плотностью нервных импульсов, передаваемых по волокну.

5. *Надежность*. Биологические НС обладают фантастической надежностью: выход из строя даже 10 процентов нейронов в нервной системе не прерывает ее работы. По сравнению с последовательными ЭВМ, основанными на принципах фон Неймана, где сбой одной ячейки памяти или одного узла в аппаратуре приводит к краху системы.

Современные искусственные НС по сложности и "интеллекту" приближаются к нервной системе таракана, но уже сейчас демонстрируют ценные свойства:

1. *Обучаемость.* Выбрав одну из моделей НС, создав сеть и выполнив алгоритм обучения, мы можем обучить сеть решению задачи, которая ей по силам. Нет никаких гарантий, что это удастся сделать при выбранных сети, алгоритме и задаче, но если все сделано правильно, то обучение бывает успешным.

2. *Способность к обобщению*. После обучения сеть становится нечувствительной к малым изменениям входных сигналов (шуму или вариациям входных образов) и дает правильный результат на выходе.

3. *Способность к абстрагированию*. Если предъявить сети несколько искаженных вариантов входного образа, то сеть сама может создать на выходе идеальный образ, с которым она никогда не встречалась.

Доступность и возросшие вычислительные возможности современных компьютеров привели к широкому распространению программ, использующих принципы нейросетевой обработки данных, о которых мы поговорим подробнее в следующих разделах, но исполняемых на последовательных компьютерах. Этот подход не использует преимуществ присущего нейро-вычислениям параллелизма, ориентируясь исключительно на способность нейросетей решать не формализуемые задачи.

1.3. История развития нейрокомпьютерных вычислений

Первые работы по формальному анализу математических моделей нейронных сетей относятся к 1943 г., имеется в виду работа Мак-Каллока и Вальтера Питтса “Логическое исчисление идей, относящихся к нервной активности“ (Mc Culloch V. S., Pitts W. H. A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophysics, 1943, v. 2, p548–558). На русский язык эта работа была переведена в 1956 г.

Авторы предложили простейшую модель работы нейрона, структура этой модели сохранилась и по сей день. Ее усовершенствование шло за счет усложнения функций активации, способов объединения нейронов в нейронные сети и динамики их работы, развития методов обучения нейронных сетей.

Фундаментальный анализ математической модели нейронной сети был проведен известнейшим специалистом в области математической логики С. К. Клини в работе “Представление событий в нервных сетях и конечных автоматах” (Труды корпорации RAND, 1951 г.).

Следующим важным шагом в развитии идей нейрокомпьютинга следует считать модель персептрона Розенблатта, которая была предложена в 1962 г. для решения некоторых задач распознавания образов (Розенблатт Ф. Принципы нейродинамики. М., 1965).

Модель персептрона Розенблатта была подвергнута всестороннему математическому анализу в работе М Минского, и С. Пейперта («Персептроны» М., 1971), которые доказали невозможность использования персептрона для решения некоторых простых задач

распознавания геометрических объектов, в частности невозможность распознать геометрически подобные образы.

В ответ на это были предложены усовершенствованные варианты нейросетей, которые снимали ряд ограничений персептрона и расширяли круг решаемых задач. Но одновременно было установлено, что мощностей существовавшей в тот период вычислительной техники явно недостаточно для решения многих реальных задач распознавания и классификации. Электроника того времени не позволяла также создать приемлемый по быстродействию и стоимости нейрокомпьютер[2].

Попытки моделирования нейросетей электрическими схемами были предприняты в начале 50-х годов. После этого остался только один шаг в направлении использования электронных моделей в качестве приборов для решения математических задач, не имевших прямого отношения к анализу работы мозга, т. е. в качестве счетно-решающих устройств. Этому шагу предшествовали многочисленные теоретические исследования принципиальных возможностей нейросетей (НС) по обработке данных. Исследовались также классы задач, которые могли бы быть решены с использованием нейросетевого подхода.

Сложность и необычность этих исследований заключалась в том, что нейросети по своей сути являются устройствами сугубо параллельного действия, в которых все узлы (нейроны) работают одновременно и одновременно обмениваются информацией между собой. Строго формализовать процессы, происходящие в нейронных сетях, оказалось не столь простым делом. При этом цель адекватного отражения процессов, которые происходят в биологической нейронной сети головного мозга, в исследованиях по созданию вычислительных устройств нейросетевого типа была отодвинута на второй план или полностью игнорировалась.

---

[2] Обзор Кузнецова А.В. Нейрокомпьютинг: история, состояние, перспективы. – www.uran.donets.ua

Главной целью становится максимальное использование параллелизма в обработке данных в нейроподобных электронных системах. Во имя достижения этой цели изобретаются модели сетей различной топологии, с различными способами формирования порога возбудимости, различными способами синхронизации.

Замечательным свойством нейроподобных вычислительных систем является их способность к обучению и самообучению.

В обычных универсальных вычислительных машинах в качестве параметров для управления аппаратурой компьютера, решающего те или иные задачи, выступает код программы, реализующий алгоритм. Алгоритм задается человеком, специалистом в конкретной области знаний. Процесс настройки компьютера сводится к составлению программы, ее отладке, к подгонке параметров программы с целью достижения правильной ее работы. В процессе обучения происходит автоматическая настройка параметров нейросети, ее весовых коэффициентов. При этом не задается перечень тех арифметических и логических операций, которые будут выполняться в сети в процессе решения задачи, как это делается при обычном программировании. Обучение, как правило, производится с использованием известных результатов решения аналогичных задач.

Реальные задачи требовали для своего решения построения нейросетей, содержащих тысячи узлов – нейронов и многие тысячи связей между ними. Процесс обучения требовал настройки нескольких тысяч весовых коэффициентов. Средствами существовавшей в то время вычислительной техники этого сделать практически было невозможно. Интерес к использованию нейросетевых вычислений упал. Последовало сокращение инвестиций в это направление исследований.

Эти обстоятельства затормозили развитие идей нейровычислений до той поры, пока не появилась вычислительная техника 80-х годов,

основанная на использовании больших интегральных схем и микропроцессоров.

К этому следует добавить, что в 70–80 годы начался бурный расцвет экспертных систем, также способных решать некоторые плохо формализуемые задачи. На экспертные системы возлагались очень большие надежды, что также повлияло на ослабление интереса к работам в области нейронных сетей.

1.4. Области применения искусственных нейронных сетей

Нейросети пригодны для решения широкого круга задач, связанных с обработкой образов. Вот список типичных постановок задач для нейросетей[3]:

- аппроксимация функций по набору точек (регрессия);
- классификация данных по заданному набору классов;
- кластеризация данных с выявлением заранее неизвестных классов-прототипов;
- сжатие информации;
- восстановление утраченных данных;
- ассоциативная память;
- оптимизация, оптимальное управление.

Этот список можно было бы продолжить и дальше. Заметим, однако, что между всеми этими внешне различными постановками задач существует глубокое родство. За ними просматривается некий единый

[3] Обзор Кузнецова А.В. Нейрокомпьютинг: история, состояние, перспективы. – www.uran.donets.ua

прототип, позволяющий при известной доле воображения сводить их друг к другу.

Рассмотрим, например, задачу *аппроксимации* функции по набору точек. Это типичный пример некорректной задачи, т. е. задачи, не имеющей единственного решения. Чтобы добиться единственности, такие задачи надо *регуляризировать* – дополнить требованием минимизации некоторого регуляризирующего функционала. Минимизация такого функционала и является целью обучения нейронной сети. Задачи *оптимизации* также сводятся к минимизации целевых функций при заданном наборе ограничений. С другой стороны, *классификация* – это ни что иное, как аппроксимация функции с дискретными значениями (идентификаторами классов), хотя ее можно рассматривать и как частный случай заполнения пропусков в базах данных, в данном случае - в колонке идентификаторов класса. Задача *восстановления* утраченных данных, в свою очередь - это *ассоциативная память*, восстанавливающая прообраз по его части. Такими прообразами в задаче *кластеризации* выступают центры кластеров. Наконец, если информацию удается восстановить по какой-нибудь ее части, значит мы добились *сжатия* этой информации, и т.д.

Многие представители разных наук, занимающихся перечисленными выше задачами и уже накопившими изрядный опыт их решения, видят в нейросетях лишь перепев уже известных им мотивов. Каждый полагает, что перевод его методов на новый язык нейросетевых схем ничего принципиально нового не дает. Статистики говорят, что нейросети - это всего лишь частный способ статистической обработки данных, специалисты по оптимизации - что методы обучения нейросетей давно известны в их области, теория аппроксимации функций рассматривает нейросети наряду с другими методами многомерной аппроксимации. Нам же представляется, что именно синтез различных методов и идей в едином

нейросетевом подходе и является неоценимым достоинством нейрокомпьютинга. Нейрокомпьютинг предоставляет единую методологию решения очень широкого круга практически интересных задач. Это, как правило, ускоряет и удешевляет разработку приложений. Причем, что обычно забывают за неразвитостью соответствующего hardware, но что, видимо, в конце концов сыграет решающую роль, нейросетевые алгоритмы решения всех перечисленных выше задач заведомо параллельны. Следовательно, все можно реализовать быстрее и дешевле.

Практически в каждой предметной области можно найти постановки нейросетевых задач [17-21].

Экономика и бизнес: предсказание рынков, оценка риска невозврата кредитов, предсказание банкротств, оценка стоимости недвижимости, выявление пере- и недооцененных компаний, автоматическое рейтингование, оптимизация портфелей, оптимизация товарных и денежных потоков, автоматическое считывание чеков и форм, безопасность транзакций по пластиковым карточкам.

Программное обеспечение компании RETEK, дочерней фирмы HNC Software, - лидер среди крупных ритейлоров с оборотом свыше $1 млрд. Ее последний продукт января 1998 года Retek Predictive Enterprise Solution включает развитые средства нейросетевого анализа больших потоков данных, характерных для крупной розничной торговли. Он также содержит прогнозный блок, чтобы можно было заранее просчитать последствия тех или иных решений. (http://www.retek.com)

Медицина: обработка медицинских изображений, мониторинг состояния пациентов, диагностика, факторный анализ эффективности лечения, очистка показаний приборов от шумов.

Группа *НейроКомп* из Красноярска (под руководством Александра Николаевича Горбаня) совместно с Красноярским межобластном

офтальмологическом центром им. Макарова разработали систему ранней диагностики меланомы сосудистой оболочки глаза. Этот вид рака составляют почти 90% всех внутриглазных опухолей и легко диагностируется лишь на поздней стадии. Метод основан на косвенном измерении содержания меланина в ресницах. Полученные данные спектрофотометрии, а также общие характеристики обследуемого (пол, возраст и др.) подаются на входные синапсы 43-нейронного классификатора. Нейросеть решает, имеется ли у пациента опухоль, и если да, то определяет ее стадию, выдавая, кроме этого, процентную вероятность своей уверенности (http://www.chat.ru/~neurocom/).

Авионика: обучаемые автопилоты, распознавание сигналов радаров, адаптивное пилотирование сильно поврежденного самолета.

Компания McDonnell Douglas Electronic Systems разработала автоматический переключатель режимов полета в реальном масштабе времени в зависимости от вида повреждения самолета. Данные от 20 сенсорных датчиков и сигналов от пилота используются нейросетью для выработки около 100 аэродинамических параметров полета. Сильной стороной является возможность сети адаптироваться к непредсказуемым аэродинамическим режимам, таким как потеря части крыла и т.д.

Связь: сжатие видеоинформации, быстрое кодирование-декодирование, оптимизация сотовых сетей и схем маршрутизации пакетов.

Нейросети уже продемонстрировали коэффициент сжатия 120:1 для черно-белого видео. Цветное видео допускает примерно вдвое большую степень сжатия 240:1 за счет специальной схемы кодирования цветов. (http://www.ee.duke.edu/~cec/JPL/paper.html)

Интернет: ассоциативный поиск информации, электронные секретари и агенты пользователя в сети, фильтрация информации в push-

системах, коллаборативная фильтрация, рубрикация новостных лент, адресная реклама, адресный маркетинг для электронной торговли.

Фирма Autonomy отделилась от родительской фирмы Neurodynamics в июне 1996 года с уставным капиталом $45 млн и идеей продвижения на рынок Internet электронных нейросетевых *агентов*. Согласно ее пресс-релизу, первоначальные вложения окупились уже через год. Компания производит семейство продуктов AGENTWARE, создающих и использующих профили интересов пользователей в виде персональных автономных нейро-агентов. Такие компактные нейро-агенты (<1 Кб) могут представлять пользователя в любом из продуктов компании. Например, агенты могут служить в качестве нейро-секретарей, фильтруя поступающую по информационным каналам информацию. Они также могут постоянно находиться на сервере провайдера, или посылаться для поиска в удаленных базах данных, осуществляя отбор данных на месте. В будущем, когда эта технология получит достаточное распространение, она позволит снизить нагрузку на трафик Сети. (http://www.agentware.com)

Автоматизация производства: оптимизация режимов производственного процесса, комплексная диагностика качества продукции (ультразвук, оптика, гамма-излучение), мониторинг и визуализация многомерной диспетчерской информации, предупреждение аварийных ситуаций, робототехника.

Ford Motors Company внедрила у себя нейросистему для диагностики двигателей после неудачных попыток построить экспертную систему, т.к. хотя опытный механик и может диагностировать неисправности он не в состоянии описать алгоритм такого распознавания. На вход нейро-системы подаются данные от 31 датчика. Нейросеть обучалась различным видам неисправностей по 868 примерам. "После полного цикла обучения качество диагностирования неисправностей сетью достигло уровня наших лучших экспертов, и значительно превосходило их в скорости (Marko K, et.

al., Ford Motors Company, Automative Control Systems Diagnostics, IJCNN 1989).

Политические технологии: анализ и обобщение социологических опросов, предсказание динамики рейтингов, выявление значимых факторов, объективная кластеризация электората, визуализация социальной динамики населения.

Уже упоминавшаяся ранее группа *НейроКомп* из Красноярска довольно уверенно предсказывает результаты президентских выборов в США на основании анкеты из 12 вопросов. Причем, анализ обученной нейросети позволил выявить пять ключевых вопросов, ответы на которых формируют два главных фактора, определяющих успех президентской кампании. Этот пример будет рассмотрен более подробно в главе, посвященной извлечению знаний с помощью нейросетей.

Безопасность и охранные системы: системы идентификации личности, распознавание голоса, лиц в толпе, распознавание автомобильных номеров, анализ аэрокосмических снимков, мониторинг информационных потоков, обнаружение подделок.

Многие банки используют нейросети для обнаружения подделок чеков. Корпорация Nestor (Providence, Rhode Island) установила подобную систему в Mellon Bank, что по оценкам должно сэкономить последнему $500,000 в год. Нейросеть обнаруживает в 20 раз больше подделок, чем установленная до нее экспертная система.

Ввод и обработка информации: Обработка рукописных чеков, распознавание подписей, отпечатков пальцев и голоса. Ввод в компьютер финансовых и налоговых документов.

Разработанные итальянской фирмой RES Informatica нейросетевые пакеты серии FlexRead, используются для распознавания и автоматического ввода рукописных платежных документов и налоговых деклараций. В первом случае они применяются для распознавания не

только количества товаров и их стоимости, но также и формата документа. В случае налоговых деклараций распознаются фискальные коды и суммы налогов.

Геологоразведка: анализ сейсмических данных, ассоциативные методики поиска полезных ископаемых, оценка ресурсов месторождений.

Нейросети используются фирмой Amoco для выделения характерных пиков в показаниях сейсмических датчиков. Надежность распознавания пиков - 95% по каждой сейсмо-линии. По сравнению с ручной обработкой скорость анализа данных увеличилась в 8 раз. (J.Veezhinathan & D.Wadner, Amoco, First Break Picking, IJCNN, 1990)

Обилие приведенных выше применений нейросетей - не рекламный трюк. Просто нейросети - это не что иное, как новый инструмент анализа данных. И лучше других им может воспользоваться именно специалист в своей предметной области. Основные трудности на пути еще более широкого распространения нейротехнологий - в неумении широкого круга профессионалов формулировать свои проблемы в терминах, допускающих простое нейросетевое решение. Данное учебное пособие призвано помочь усвоить типовые постановки задач для нейросетей. Для этого, прежде всего, нужно четко представлять себе основные особенности нейросетевой обработки информации - *парадигмы* нейрокомпьютинга.

1.5. Классификация нейронных сетей

К настоящему времени разработано несколько типов нейросетей, используемых для решения практических задач обработки данных.

Нейронные сети различают по структуре (рис.1.1) на неполносвязные, полносвязные, со случайными и регулярными связями, с симметричными и несимметричными связями.

Неполносвязные нейронные сети описываются неполносвязным ориентированным графом и разделяются на однослойные и многослойные (слоистые) с прямыми, перекрёстными и обратными связями.

В полносвязных нейронных сетях каждый нейрон передаёт свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

В многослойных нейронных сетях нейроны объединяются в слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях.

В сетях с обратными связями информация с последующих слоёв передаётся на предыдущие слои.

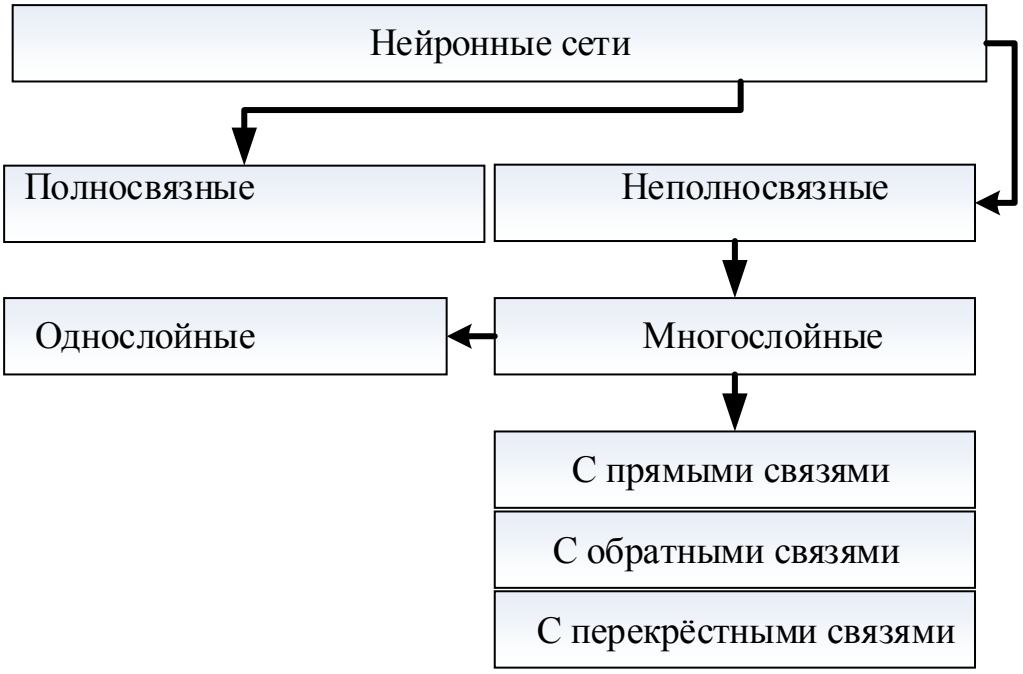


Рис. 1.1. Классификация нейронных сетей

По способу подачи информации на входы нейронные сети разделяют:

– подачу сигналов на синапсы входных нейронов;

– подачу сигналов на выходы входных нейронов;

– подачу сигналов в виде весов синапсов входных нейронов.

По способу съёма информации с выходов нейронные сети разделяют:

– съём с выходов выходных нейронов;

– съём с синапсов выходных нейронов;

– съём в виде весов синапсов выходных нейронов.

Нейронные сети классифицируются также по виду функций активации, которые могут быть разрывными, ступенчатыми и непрерывными и от наличия обратных связей (табл.1.1).

Таблица 1.1. Сравнение архитектур нейронных сетей

Сравнение сетей	*Без обратных связей (многослойные)*	*С обратными связями*
Преимущества	Простота реализации. Гарантированное получение ответа после прохождения данных по слоям	Минимизация размеров сети - нейроны многократно участвуют в обработке данных. Меньший объем сети облегчает процесс обучения
Недостатки	Требуется большее число нейронов для алгоритмов одного и того же уровня сложности. Следствие - большая сложность обучения	Требуются специальные условия, гарантирующие сходимость вычислений

В последнее время большое распространение получили так называемые радиальные сети, у которых функция активации позволяет определять близость исследуемого образа (точки пространства) к группе других точек, объединенных в кластер. В простейшем случае мера близости равна расстоянию от центра масс выделенного кластера.

Нейронные сети классифицируются по характеру входных/выходных сигналов (информации, циркулирующей между узлами).

Различают бинарные, *k*-арные сети, сети с данными и весами произвольного типа. Время в сетях может быть дискретным и непрерывным, соответственно говорят о непрерывных и дискретных сетях.

Классификация по типу связей и типу обучения (Encoding-Decoding) представлена в табл. 1.2.

Таблица 1.2. Классификация по типу связей и типу обучения (Encoding-Decoding)

Тип связей	Принцип обучения нейронной сети	
	с "учителем"	без "учителя"
Без обратных связей	Многослойные персептроны (аппроксимация функций, классификация)	Соревновательные сети, карты Кохонена (сжатие данных, выделение признаков)
С обратными связями	Рекуррентные аппроксиматоры (предсказание временных рядов, обучение в режиме on-line)	Сеть Хопфилда (ассоциативная память, кластеризация данных, оптимизация)

1.6. Элементная база для аппаратной реализации нейрокомпьютеров

В мире имеется несколько десятков специализированных фирм, выпускающих продукцию в области нейроинформатики и, кроме того, многие гиганты индустрии (IBM, Siemens Nixdorf, Mitsubishi и др.) ведут исследования и разработки в этой области [4,11-14,22]. То есть намерения здесь в настоящее время серьезные. Каковы же достигнутые результаты?

Что такое нейрокомпьютер? На самом деле сейчас между понятиями компьютер и нейрокомпьютер примерно такое же соотношение как между понятиями государь и милостивый государь. То есть сейчас любой нейрокомпьютер не претендует на звание компьютера, но создается для решения какого-то фиксированного круга задач. Похоже, что широкие приложения получат устройства, основанные на комбинированных технологиях, включающие по мере необходимости те или иные нейропроцессорные устройства.

Некоторую рекламу и, соответственно, некоторые средства в Японии получили приборы бытовой техники: пылесосы, кондиционеры, электропечки и стиральные машины, использующие нейропроцессоры в устройствах управления.

Появилась мода на применения искусственных нейронных сетей в финансово-экономических приложениях. Причиной появления моды стали успехи нейросетевых систем в области предсказания будущего. В этой и других задачах такого рода речь идет о том, чтобы по имеющемуся числовому и событийному ряду предсказать следующие его члены.

Нейросетевые конструкции порой решают эту задачу лучше, чем изощренные статистические методики. В чем здесь дело, пока непонятно, но финансистам это не важно.

Традиционной областью применения нейропроцессоров остаются задачи узнавания изображений, речи, радарных сигналов.

Одно из новых, но впечатляющих приложений - физика высоких энергий (элементарных частиц). В задачах этой науки необходимо в огромном потоке данных от многочисленных датчиков сигналов, от элементарных частиц в детекторах ускорителей разного рода, найти комбинации данных, означающих наличие определенных известных или предполагаемых событий. Предварительная обработка информации в этих случаях может быть выполнена нейропроцессорами, "натренированными" методами численного моделирования соответствующих процессов на поиск заданных событий.

И все же основной областью применения нейропроцессоров, скорее всего, станет использование в системах управления и поведения роботами. Глава фирмы, занимающей бесспорно первое место в мире по приложениям нейросетевых систем, автор термина NEUROCOMPUTER, американский профессор из калифорнийского города Сан-Диего Роберт Хехт-Нильсен полагает, что основной продукцией, производимой промышленными фирмами через 10 лет, станут "нейровычислительные роботы" (НВР). В частности, появятся разнообразные работы для выполнения домашней работы (няньки, прачки, кухарки...). Производство НВР быстро превзойдет по объему производство автомобилей и перейдет во все подавляющее производство роботов, производящих роботов...

При реализации базового элемента НК – формального нейрона, способа соединения и обучения сети разработчик может выбрать одно из трёх направлений [4]:

– *программное* – все элементы НК реализуются на программном уровне в универсальных ЭВМ с архитектурой фон Неймана;

–*аппаратно-программное* – часть элементов реализуется на аппаратном уровне, а часть – на программном уровне;

– *аппаратное* – все элементы НК выполнены на аппаратном уровне кроме специфических программ формирования синаптических коэффициентов.

Программные системы, реализующие первое направление, даже если для них требуется многопроцессорная система или многомашинная аппаратная поддержка получили название *нейроэмуляторы, нейроимитаторы*.

Второе и третье направления на аппаратном уровне реализуются на заказных кристаллах (ЗК – ASIC), встраиваемых микроконтроллерах, процессорах общего назначения, программируемых логических интегральных схемах (ПЛИС), транспьютерах, цифровых сигнальных процессорах(ЦПС) и нейрочипах [4].

Предпочтение в реализации отдается ЦПС, ПЛИС и нейрочипам.

Второе направление характеризуется тем, что аппаратная часть НК выполняется в виде платы расширения для универсальных ЭВМ. Например, реализуются операции взвешенного суммирования, нелинейного преобразования. Такие платы получили название *нейроускорители*.

Третье направление характеризуется тем, что вся аппаратная часть НК реализована на аппаратном уровне. Например, ПЛИС фирмы XILINX типа FPGA представляет собой массив конфигурируемых логических блоков (КЛБ) с полностью конфигурируемыми высокоскоростными межсоединениями. ПЛИС серии XC4000 расположена на одном кристалле и включает: конфигурируемых логических блоков (КЛБ) с полностью конфигурируемыми высокоскоростными межсоединениями; по периферии кристалла расположены блоки ввода-вывода, из которых каждый содержит два триггера (один для ввода и один для вывода); логика дешифрации; цепи контроля высокоомных состояний.

Время распространения сигналов через КЛБ составляет 0,5 нс, через блок ускоренного переноса - 0,1нс, время переключения триггера – не более 0,5 нс. Внутренние межсоединения конфигурируются пользователем и задают задержку до 5 нс. Каскадное соединение двухразрядных сумматоров позволяет построить 16-разрядный сумматор с временем суммирования двух 16-разрядных чисел в пределах 5 нс при тактовой частоте 200МГц. Каждый КБЛ серии XC4000 позволяет создать блок ОЗУ или двухпортовое ОЗУ, что позволяет строить высокопроизводительные системы.

Многоуровневая конвейерная структура НС на базе ПЛИС серии XC4000 обеспечивает время вычисления одного нейрона 6 нс (для нейрона с 8 входами и 8 разрядными кодами).

Особенности ЦПС как элементной базы нейрокомпьютеров.

Цифровые сигнальные процессоры (ЦПС) – DSP-Digital Signal Processor) появились в конце 70-х годов.

Важным преимуществом ЦПС как элементной базы нейровычислителей перед универсальными микропроцессорами является возможность работы на максимальных частотах, а также возможность выполнять операции алгоритма НС на аппаратном уровне. Это обеспечивает высокую производительность нейровычислителей. ЦПС имеют ряд архитектурных особенностей. Например реализована операция умножения с накоплением MAC (D: =A*B+D) Эта команда соответствует операции взвешенного суммирования в адаптивном сумматоре нейрона. Большинство ЦПС построено по Гарвардской архитектуре, которая предполагает наличие двух физически разделенных шин: шины команд и шины данных. Архитектура с кэш-памятью называется расширенной Гарвардской архитектурой.

Перспективным является построение нейровычислителей по новой архитектуре TrigerSHARC, сочетающей в себе высокую степень

конвейеризации и программируемость RISC –процессоров. Архитектура TrigerSHARC предполагает 3 независимых блока памяти, каждый из которых имеет 128-разрядную шину данных. Адрес доступа к данным может состоять из одного, двух, трех и четырех слов, что позволяет пользователю отказаться от сегментации памяти на память программ и память данных.

Для оценки производительности нейровычислителей используются следующие показатели:

CUPS- число изменений значений весов в секунду (оценка скорости обучения);

CPS- число соединений в секунду;

CPSPW=CPS/Nw –число соединений на один синапс (Nw –число синапсов в нейроне);

CPPS – число соединений примитивов в секунду; CPPS=CPU*Bw*Bs (Bw - разрядность весов, Bs - разрядность синапсов);

MMAC- миллион умножений с накоплением в секунду.

Нейрочипы.

Нейрочипом принято называть специализированную сверхбольшую интегральную схему (СБИС), ориентированную на реализацию нейросетевых алгоритмов.

Нейрочипы можно разделить по классификации на 3 класса: аналоговые; цифровые; гибридные.

Нейропроцессор Л1879ВМ1 (NeuroMatrix NM6403), разработанный научно-техническом центре «Модуль» (РФ, Москва). Он предназначен для аппаратной эмуляции разнообразных НС [4,22].

NeuroMatrix NM6403 предназначен для обработки 32-разрядных скалярных данных и данных программируемой разрядности, упакованных в 64-разрядные слова. Основой нейрочипа является RISC – процессор для выполнения арифметических и логических операций, операций сдвига над

32-разрядными скалярными данными. Основные функции структурных блоков NeuroMatrix NM6403:

VCP –векторный сопроцессор, предназначенный для выполнения арифметических и логических операций над 64-разрядными векторами данных программируемой разрядности;

LMI, GMI – два одинаковых блока программируемого интерфейса с локальной и глобальной 64-разрядными шинами данных. К каждой из шин может быть подключена внешняя память объёмом 2^{31} 32 –разрядных ячеек. Обмен может выполняться как 32, так и 64 –разрядными данными.

Перспективы аппаратной реализации нейрокомпьютеров.

Современные возможности аппаратной реализации НПС можно обобщенно оценить следующим образом:

число моделируемых нейронов — до 5 млн.;

число моделируемых связей — до 5 млн.;

скорость моделирования — до 500 млн. переключений связей/с.

Для аппаратной реализации НПС в настоящее время широко используются процессорные СБИС, обладающие максимальными коммуникационными возможностями и ориентированные на быстрое выполнение векторных операций. К таким СБИС относятся транспьютеры фирмы INMOS (T414, T800, A100), сигнальные процессоры фирм Texas Instruments (TMS 320C40, TMS 320C80), Motorola, Analog Device. Отечественная элементная база представлена нейрочипами на базе БМК «Исполин».

На современном этапе развития технологии микроэлектроники и других смежных областей нейронная технология стала адекватна не только различным типам микроэлектронной полупроводниковой технологии, но и оптической, оптоэлектронной, молекулярной, квантовой и некоторым другим.

Необходимо отметить, что рождение технологии систем на пластине и нанотехнологии приведет к рождению новых сверхпараллельных архитектур. Уже сейчас ясна адекватность нейросетевых архитектур технологии на пластине. Поэтому любые попытки на уровне наноэлементов делать функциональные блоки со старой архитектурой, адекватной однопроцессорным машинам, должны окончиться неудачей.

Современные технологии достигли того рубежа, когда стало возможным изготовление технической системы из 3...4 млрд. нейронов (именно такое количество их в мозгу человека). Однако их соединение продолжает оставаться проблемой.

Перечислим основные направления вычислительной техники:

– однопроцессорные ЭВМ (персональные ЭВМ, ЭВМ среднего класса);

– малопроцессорные ЭВМ;

– многопроцессорные ЭВМ (ЭВМ с массовым параллелизмом, транспьютерная ЭВМ, псевдотранспьютерная ЭВМ, ЭВМ с транспьютерным ядром и периферийными процессорами типа i860, Power PC, Alfa);

– нейрокомпьютеры.

Приоритет российской вычислительной науки и техники в указанных направлениях в ближайшие годы максимально может быть проявлен именно в области НК, поскольку она является максимально наукоемкой и менее других зависит от технологического уровня.

В России высок уровень теоретических работ и экспериментальных исследований по нейросистемам и нейроинформатике. Реальность создания нейрокомпьютерных средств уже сегодня не вызывает сомнений. Это кредиторам дает дополнительный стимул для вложения средств в развитие теоретических исследований, направленных на поиск решений широкого круга практических задач на основе нейросетевых технологий.

2. ОСНОВЫ ТЕОРИИ НЕЙРОННЫХ СЕТЕЙ

2.1. Нейросеть - виртуальная модель мультипроцессорной системы

Под нейронными сетями (НС) 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.

Хотя нейросетевые процессы по своей сути являются параллельными, проблема уменьшения потерь эффективности при моделировании нейровычислений на современных вычислительных системах с массовым параллелизмом является сложной и еще далеко не решенной задачей системного программирования. Наиболее трудоемкой по затратам вычислительных ресурсов и, следовательно, времени является задача обучения. По существу, единственным методом радикального ускорения этого процесса является его распараллеливание, и на этом направлении сосредотачиваются усилия системных программистов.

Отметим существенную разницу между нейрокомпьютером и машинами, управляемыми потоком данных. В графе потока данных в той или иной кодировке указан перечень арифметических и логических действий, порядок их выполнения и способ извлечения операндов из памяти машины. В этом смысле граф потока данных можно считать своеобразной программой вычислений. В нейрокомпьютере такого сходства с алгоритмом, заданным потоком данных, нет.

2.2. Формальная модель нейрона

2.2.1. Биологический нейрон

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. 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 [1-6].

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию (рис. 2.1).

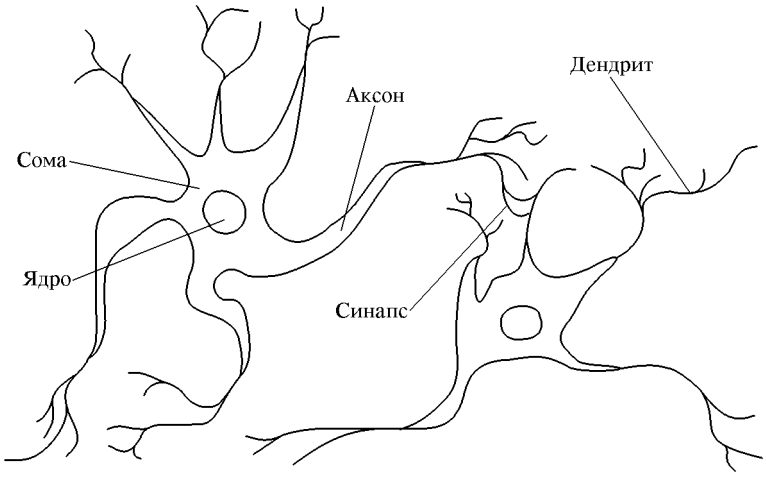


Рис. 2.1. Взаимосвязь биологических нейронов

Он состоит из тела и отростков нервных волокон двух типов — дендритов, по которым принимаются импульсы, и единственного аксона, по которому нейрон может передавать импульс. Тело нейрона включает ядро, которое содержит информацию о наследственных свойствах, и плазму, обладающую молекулярными средствами для производства

необходимых нейрону материалов. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчик), который в конце разветвляется на волокна. На окончаниях этих волокон находятся специальные образования — синапсы, которые влияют на силу импульса.

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Когда импульс достигает синаптического окончания, высвобождаются определенные химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синаптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать электрические импульсы. Результативность синапса может настраиваться проходящими через него сигналами, так что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память, которая, возможно, ответственна за память человека. Важно отметить, что веса синапсов могут изменяться со временем, что изменяет и поведение соответствующего нейрона.

2.2.2. Понятие искусственного нейрона

Искусственный нейрон (математический нейрон Мак-Каллока - Питтса, формальный нейрон [1]) - узел искусственной нейронной сети, являющийся упрощённой моделью естественного нейрона. Математически, искусственный нейрон обычно представляют, как некоторую нелинейную функцию от единственного аргумента - линейной комбинации всех входных сигналов. Данную функцию называют функцией активации[1-6] или функцией срабатывания, передаточной

функцией. Полученный результат посылается на единственный выход. Такие искусственные нейроны объединяют в сети - соединяют выходы одних нейронов с входами других. Искусственные нейроны и сети являются основными элементами идеального нейрокомпьютера.

Элементарной ячейкой нейронной сети является нейрон, структура которого с одним скалярным входом приведена на рис.2.2.

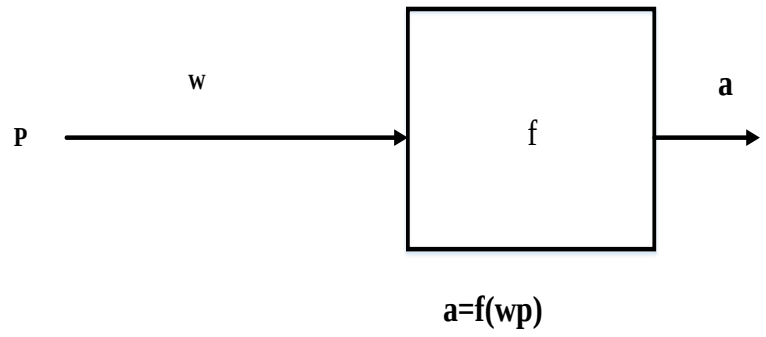


Рис. 2.2. Структура нейрона

Скалярный входной сигнал p умножается на скалярный весовой коэффициент *w*, и результирующий взвешенный вход *w*p* является аргументом функции активации нейрона *f*, которая формирует выходной сигнал *a*.

Нейрон, приведённый на рис.2.3, дополнен скалярным *смещением b*. Смещение суммируется со взвешенным входом w*p и приводит к сдвигу аргумента функции f на величину b. Смещение можно свести к схеме взвешивания с дополнительным входом равным единице.

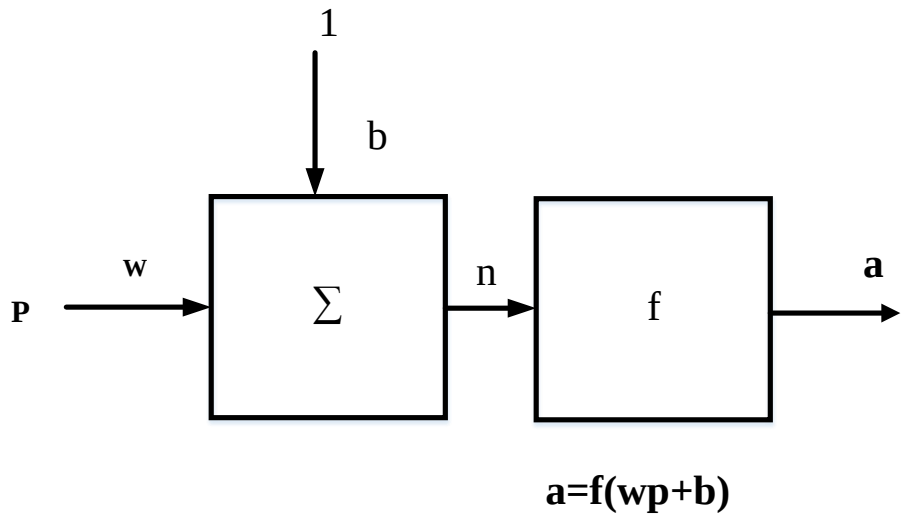


Рис.2.3. Структура нейрона со смещением

Вход функции активации n=w*p+b, а выход функции a является выходом нейрона. Константы w, b являются скалярными параметрами нейрона. Настраивая веса или параметры смещения, можно обучить нейронную сеть выполнять конкретную работу. Возможно, реализовать самообучение нейронной сети, когда она сама будет корректировать свои параметры, чтобы достичь требуемого результата. Уравнение нейрона со смещением имеет вид:

a=f (w*p+b*1).

2.2.3. Основные типы функций активации

Функции активации или передаточные функции нейрона могут иметь самый различный вид. Наиболее распространёнными функциями активации являются: единичная функция активации с жёстким ограничением; линейная функция; логистическая функция.

Единичная функция активации с жёстким ограничением, обозначаемая в MATLAB *hardlim*, приведена на рис. 2.4. Она равна нулю,

если n <0, и равна единице, если n≥0. Применяя операторы языка MATLAB можно построить график этой функции:

```
n=-5:0.1:5;
plot (n, hardlim(n), `c+:`);
```

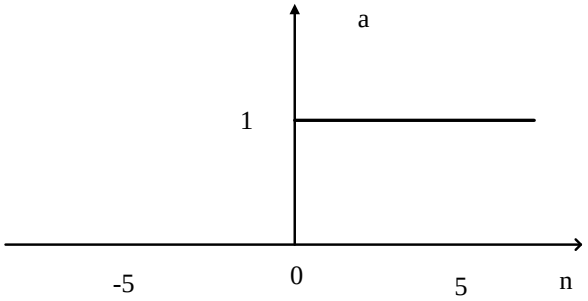


Рис.2.4. Единичная функция активации с жёстким ограничением

Недостатками единичной активационной функции является то, что она не являются дифференцируемыми на всей числовой оси, следовательно, не может быть использована в некоторых алгоритмах обучения нейронной сети.

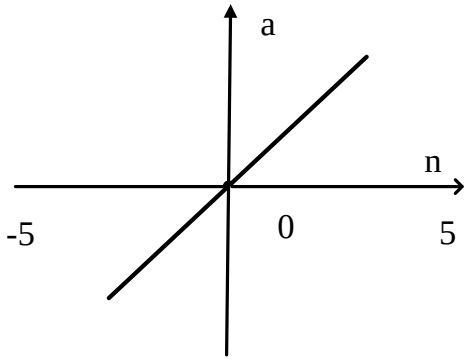


Рис.2.5. Линейная функция активации

Линейная функция активации приведена на рис. 2.5 и описывается соотношением a=purelin(n).

Линейная функция активации с насыщением определяется выражением:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x \geq 1 \\ x & \text{else} \end{cases}$$

Логистическая функция активации приведена на рис. 2.6 и описывается соотношением a=logsig(n)=1/(1+exp(-n)). Она принадлежит к классу сигмоидальных функций, и её аргумент может принимать любое значение в диапазоне от - ∞ до +∞, а выход функции изменяется в диапазоне от 0 до 1. Благодаря свойству дифференцируемости эта функция используется в сетях с обучением на основе метода обратного распространения ошибки.

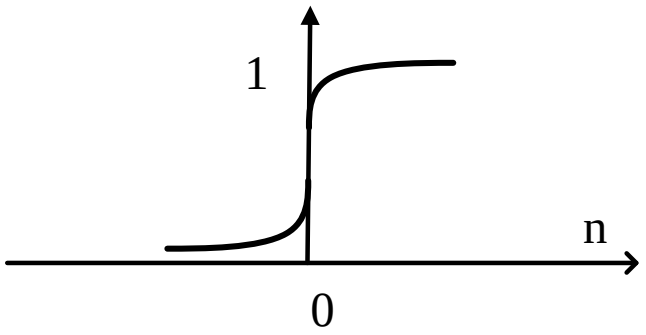


Рис.2.6. Логистическая функция активации

2.2.4. Нейрон с векторным входом

Нейрон с одним вектором входа p с R элементами $p_1,p_2,\ldots,p_R$ приведён на рис. 2.7.

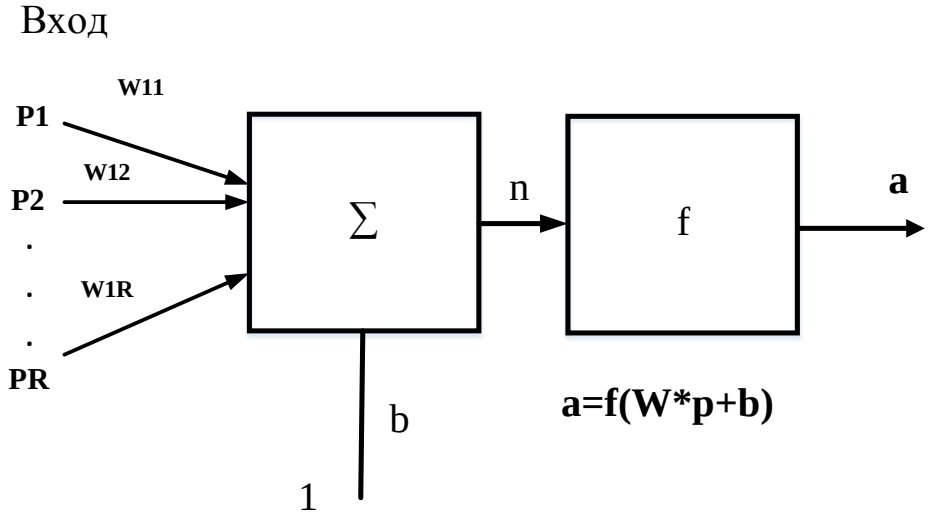


Рис.2.7. Нейрон с векторным входом

Каждый элемент входа умножается на веса $w_{11}, w_{12}, \ldots, w_{1R}$ соответственно, и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора-строки W на вектор входа p. Нейрон имеет смещение b, которое суммируется со взвешенной суммой входов и результирующая сумма n и служит аргументом функции f.

$n= w_{11}*p1+w_{12}*p2+\ldots+w_{1R}*p_R +b$

2.3. Архитектура нейронных сетей

Нейронная сеть может содержать один или более слоёв. Нейронная сеть с одним слоем называется однослойной, а с большим количеством слоёв - многослойной.

В основном, нейроны классифицируют на основе их положения в топологии сети. Входные нейроны — принимают исходный вектор, кодирующий входной сигнал. Как правило, эти нейроны не выполняют вычислительных операций, а просто передают полученный входной сигнал на входы нейронов следующего слоя.

Выходные нейроны — представляют выходы сети. В выходных нейронах могут производиться вычислительные операции суммирования и определения значения активационной функции.

Промежуточные нейроны образуют скрытые слои и выполняют основные вычислительные операции.

Структурная схема однослойной нейронной сети с R входами и S нейронами приведена на рис. 2.8.

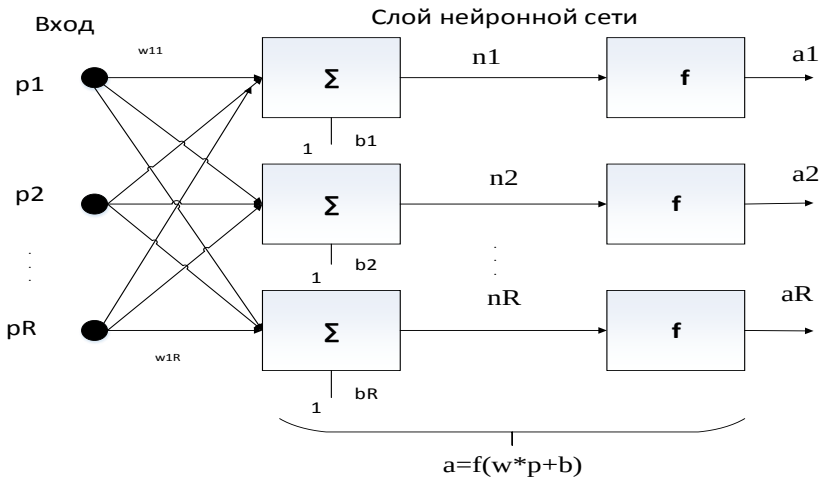


Рис. 2.8. Структура однослойной нейронной сети

В данной сети каждый элемент вектора входа соединён со всеми входами нейрона, и это соединение задаётся матрицей весов *W*; при этом каждый *i-ый* нейрон включает суммирующий элемент, который формирует скалярный выход *n(i).* Совокупность скалярных функций *n(i)* объединяется в S-элементный вектор входа n функции активации слоя.

Выходы слоя нейронов формируют вектор-столбец *a*, который имеет вид: a=f(W*p+b).

Количество входов R в слое может не совпадать с количеством нейронов S. В каждом слое используется одна и та же функция активации. Элементы вектора входа передаются в сеть через матрицу весов W, имеющую вид:

$$W=\begin{matrix} w11 & w12 & \dots & w1R \\ w21 & w22 & \dots & w2R \\ \dots & \dots & \dots & \dots \\ wS1 & wS2 & \dots & wSR \end{matrix}$$

Индексы строк матрицы W указывают адресаты (пункты назначения) весов нейронов, а индексы столбцов - какой источник является входом для этого веса. Например, элемент матрицы весов w12=W(1,2) определяет

коэффициент, на который умножается второй элемент входа при передаче его на первый нейрон.

Для однослойной нейронной сети с R входами и S нейронами укрупнённая структура приведена на рис. 2.9.

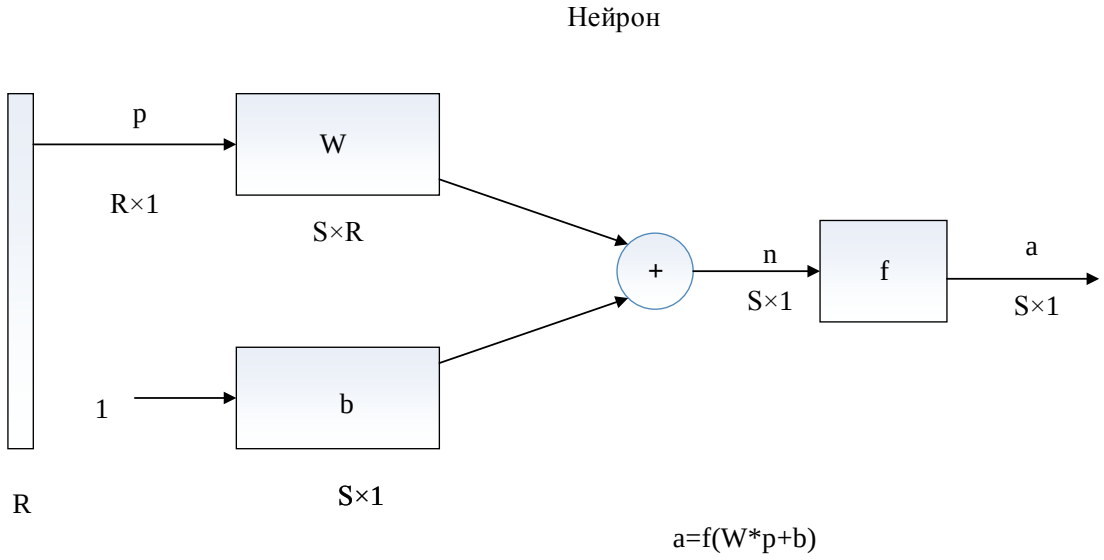


Рис. 2.9. Укрупнённая структура нейронной сети

Здесь p - вектор входа размера R×1, W- весовая матрица размера R×S, a, b, n – векторы размера S×1.

2.4. Пример моделирования однослойной нейронной сети в MATLAB

Для моделирования фрагмента схемы комбинационной, сформированной из элементов И, ИЛИ, НЕ (рис.2.10) в однослойной нейронной сети используется линейная функция активации

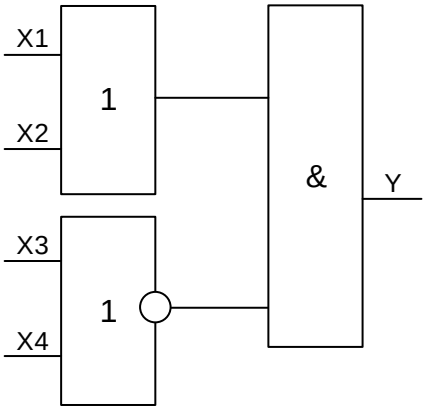


Рис.2.10. Фрагмент комбинационной схемы

Листинг программы приведён ниже, а результат моделирования представлен на рис. 2.11.

```
clear
X={[0; 0; 0; 0] [0; 0; 0; 1] [0; 0; 1; 0] [0; 0; 1; 1] [0; 1; 0; 0] [0; 1; 0; 1] [0; 1; 1; 0] [0; 1; 1; 1] [1; 0; 0; 0] [1; 0; 0; 1] [1; 0; 1; 0] [1; 0; 1; 1] [1; 1; 0; 0] [1; 1; 0; 1] [1; 1; 1; 0] [1; 1; 1; 1]};
Y={0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0};
net=newff(minmax(X),[1],{'purelin'},'traingd');
net.trainParam.goal=0
net.trainParam.epochs=5000;
[net,tr]=train(net,X,Y);
a1=sim (net, X)
```

Результаты моделирования линейной нейронной сети демонстрируют правильное обучение, но имеют большую погрешность. Значения близкие к единице выделены жирным шрифтом.

```
a1=sim (net, X)
a1 =
  Columns 1 through 11
    [0.4375]    [0.0625]    [0.0625]    [-0.3125]    [0.5625]    [0.1875]    [0.1875]    [-0.1875]    [0.5625]    [0.1875]    [0.1875]
```

Columns 12 through 16

[-0.1875] **[0.6875]** [0.3125] [0.3125] [-0.0625]

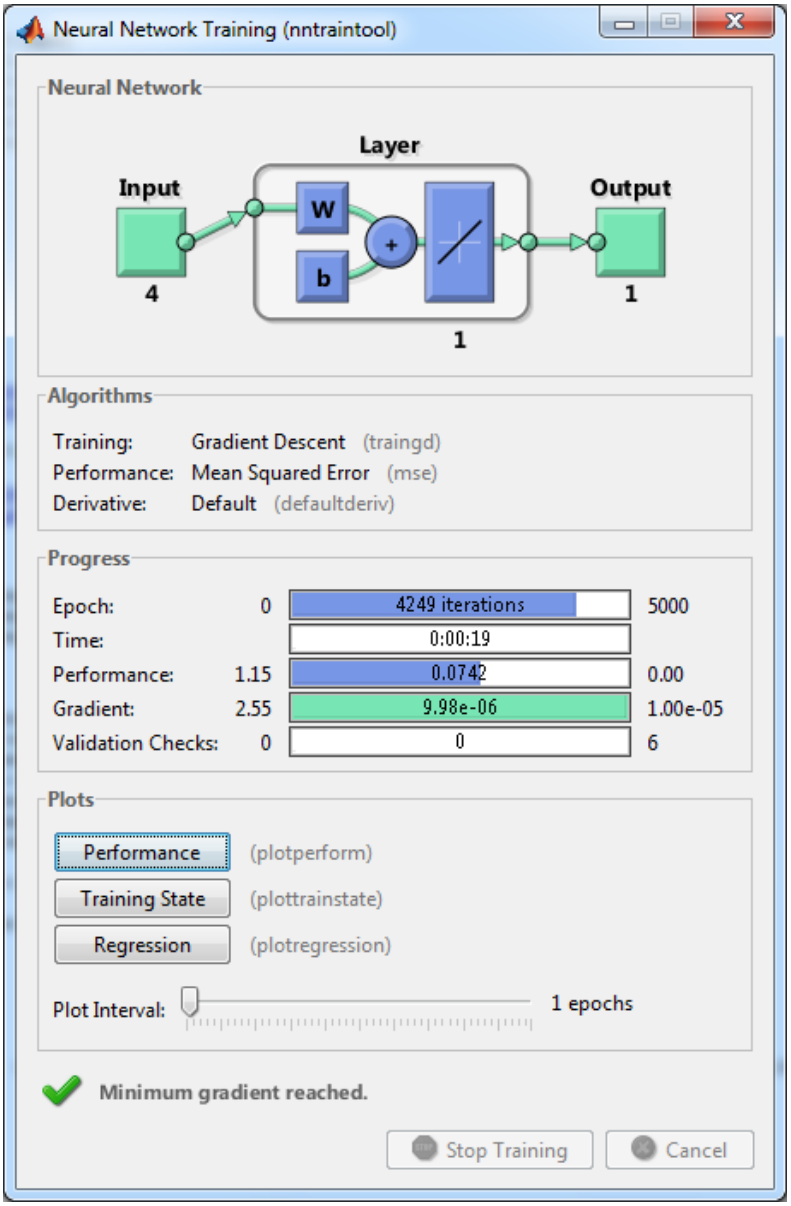


Рис.2.11. Результаты моделирования двухслойной нейронной сети

Изменив архитектуру нейронной сети на два слоя, где в первом слое будет использоваться сигмоидальная функция активации, а в выходном слое линейная функция активации, получаются достоверные результаты моделирования при низкой ошибке.

Листинг программы приведён ниже, а результат моделирования представлен на рис. 2.12.

```
clear
 X={[0; 0; 0; 0] [0; 0; 0; 1] [0; 0; 1; 0] [0; 0; 1; 1] [0; 1; 0; 0] [0; 1; 0; 1] [0; 1; 1; 0] [0; 1; 1; 1] [1; 0; 0; 0] [1; 0; 0; 1] [1; 0; 1; 0] [1; 0; 1; 1] [1; 1; 0; 0] [1; 1; 0; 1] [1; 1; 1; 0] [1; 1; 1; 1]};
Y= {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0};
net=newff (minmax(X), [30,1],{ 'tansig','purelin'},'traingd');
net.trainParam.goal=0
net.trainParam.epochs=5000;
net.trainParam.min_grad=0.000001;
 [net,tr]=train(net,X,Y);
a1=sim (net, X)
```

Результаты моделирования, приведённые ниже и на рис.2.12, показывают возможность моделирования комбинационной схемы с помощью двухслойной нейронной сети.

```
a1 =
  Columns 1 through 10
   [2.4583e-007]   [-2.8540e-006]   [-5.0679e-007]   [3.1620e-006]   [1.0000]   [6.5825e-007]   [4.3074e-008]   [-1.0129e-006]   [1.0000]   [2.1024e-006]
  Columns 11 through 16
   [-7.1937e-007]   [-1.0212e-006]   [1.0000]   [-6.7145e-007]   [1.8124e-006]   [1.1190e-007]
```

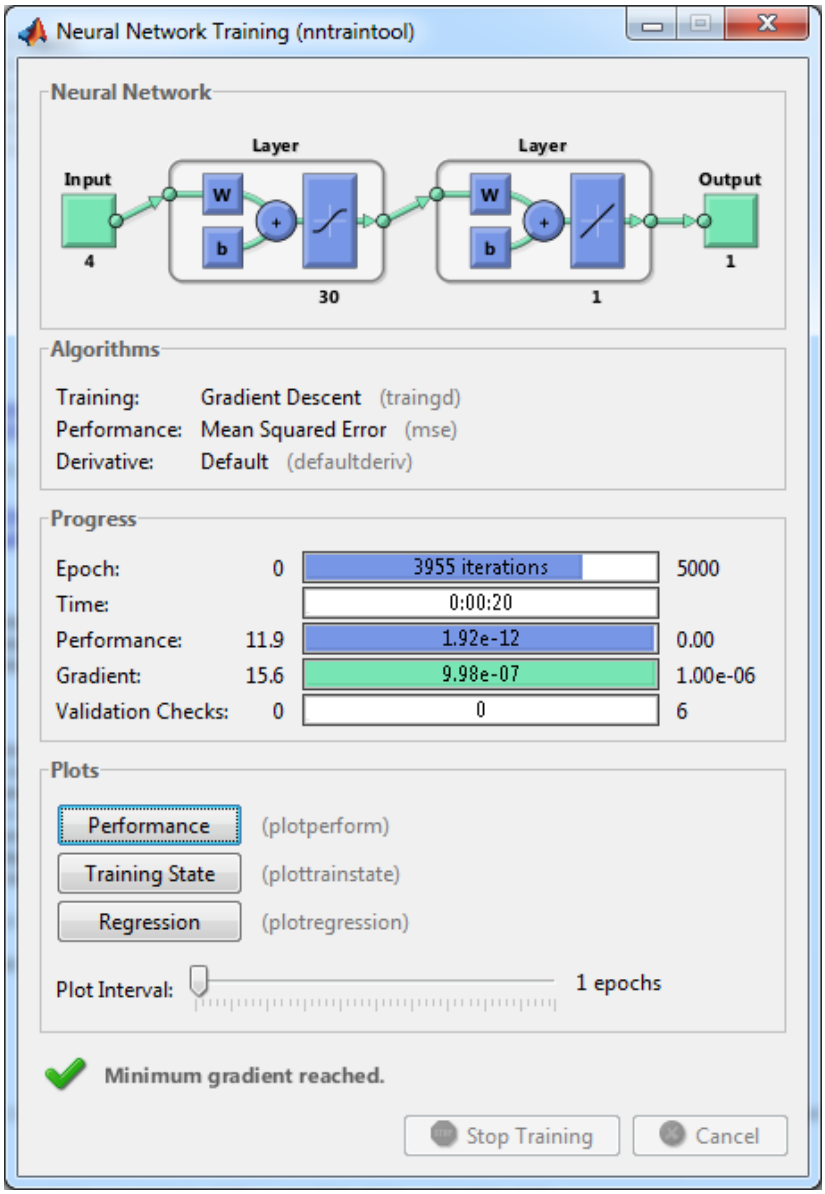


Рис.2.12. Результат моделирования двухслойной нейронной сети

3. МНОГОСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ

3.1. Принципы построения многослойных нейронных сетей

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, которое минимизирует ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного персептрона [1-6].

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

Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо.

Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод "проб и ошибок", несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, более приемлемый вариант – распространение сигналов

ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения. Именно он будет рассмотрен в дальнейшем.

3.2. Алгоритм обратного распространения ошибки

В настоящее время выпущено достаточно много различной литературы по описанию нейронных сетей. В [1,2,3,4] подробно разобран алгоритм обратного распространения ошибки (ОРО) как метод обучения НС, рассмотрены его достоинства, недостатки и способы устранения возникающих в ходе обучения проблем[4].

В книге [1,2,3,4] приведены общие сведения о НС, рассмотрены различные топологии сетей, предложены методы обучения с учителем и без учителя.

Многими исследователями были предложены улучшения и обобщения алгоритма обратного распространения.

Общепринятый от 0 до 1 динамический диапазон входов и выходов скрытых нейронов не оптимален. Так как величина коррекции веса пропорциональна выходному уровню нейрона, то нулевой уровень ведёт к тому, что вес не меняется.

При использовании двоичных входных векторах половина входов в среднем будет равна нулю, и веса, с которыми они связаны, не будут

---

[4] Короткий С., Нейронные сети: алгоритм обратного распространения. [Электронный ресурс] Режим доступа: http://www.gotai.net/documents/doc-nn-003.aspx

обучаться. Решение состоит в приведении входов к значениям ±½ и добавлении смещения к сжимающей функции, чтобы она также принимала значения ±½.

С помощью таких простых средств, время сходимости сокращается в среднем от 30 до 50%. Это является одним из примеров практической модификации, существенно улучшающей характеристику алгоритма.

Способом обратного распространения (back propogation) называется способ обучения многослойных НС. В таких НС связи между собой имеют только соседние слои, при этом каждый нейрон предыдущего слоя связан со всеми нейронами последующего слоя. Нейроны обычно имеют сигмоидальную функцию возбуждения. Первый слой нейронов называется входным и содержит число нейронов, соответствующее распознаваемому образу. Последний слой нейронов называется выходным и содержит столько нейронов, сколько классов образов распознается. Между входным и выходным слоями располагается один или более скрытых (теневых) слоев. Определение числа скрытых слоев и числа нейронов в каждом слое для конкретной задачи является неформальной задачей.

Принцип обучения такой нейронной сети базируется на вычислении отклонений значений сигналов на выходных процессорных элементах от эталонных и обратном "прогоне" этих отклонений до породивших их элементов с целью коррекции ошибки. Еще в 1974 году Поль Дж. Вербос изобрёл значительно более эффективную процедуру для вычисления величины, называемой производной ошибки по весу, когда работал над своей докторской диссертацией в Гарвардском университете. Процедура, известная теперь как алгоритм обратного распространения, стала одним из наиболее важных инструментов в обучении нейронных сетей. Однако этому алгоритму свойственны и недостатки, главный из которых - отсутствие сколько-нибудь приемлемых оценок времени обучения. Понимание того, что нейронная сеть, в конце концов обучится,

мало утешает, если на это могут уйти дни и месяцы. Тем не менее, алгоритм обратного распространения имеет широчайшее применение. Например, успех фирмы NEC в распознавании букв, был достигнут именно благодаря алгоритму обратного распространения.

Алгоритм обратного распространения ошибки (ОРО) — итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущего выхода и желаемого выхода многослойных НС. На каждый нейрон первого слоя подаются все элементы внешнего входного сигнала. Нейроны выполняют взвешенное суммирование элементов входных сигналов, прибавляя к сумме смещение нейрона. Над полученной суммой выполняется нелинейное преобразование активационной функцией. Значение функции активации и есть выход нейрона [1-6,9,10].

Среди различных структур нейронных сетей одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идёт в направлении, которое минимизирует ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного персептрона. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда

осуществимо. Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод, несмотря на свою кажущуюся простоту, требует огромного количества вычислений. И, наконец, третий, наиболее приемлемый вариант – распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения.

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2 \qquad (3.1)$$

где – реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа; d_{jp} – идеальное (желаемое) выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}$$

(3.2)

Здесь w_{ij} – весовой коэффициент синаптической связи, соединяющей i-ый нейрон слоя n-1 с j-ым нейроном слоя n, η – коэффициент скорости обучения, $0<\eta<1$.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial w_{ij}} \qquad (3.3)$$

Здесь под y_j, как и раньше, подразумевается выход нейрона j, а под s_j – взвешенная сумма его входных сигналов, то есть аргумент активационной функции. Так как множитель является производной этой функции по ее аргументу, из этого следует, что производная активационной функция должна быть определена на всей оси абсцисс. В связи с этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой. В случае гиперболического тангенса

$$\frac{dy}{ds} = 1 - s^2 \qquad (3.4)$$

Третий множитель, очевидно, равен выходу нейрона предыдущего слоя $y_i^{(n-1)}$.

Что касается первого множителя в (3), он легко раскладывается следующим образом[1-4]:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \qquad (3.5)$$

Здесь суммирование по k выполняется среди нейронов слоя n+1.

Введя новую переменную

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \qquad (3.6)$$

получается рекурсивная формула для расчетов величин $\delta_j^{(n)}$ слоя n из величин $\delta_k^{(n+1)}$ более старшего слоя n+1.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j} \qquad (3.7)$$

Для выходного же слоя

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l} \qquad (3.8)$$

Теперь мы можем записать (2) в раскрытом виде:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \qquad (3.9)$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (9) дополняется значением изменения веса на предыдущей итерации

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}) \qquad (3.10)$$

где μ – коэффициент инерционности (момента), t – номер текущей итерации.

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Напомним, что

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)} \cdot w_{ij}^{(n)} \qquad (3.11)$$

где M – число нейронов в слое n-1 с учетом нейрона с постоянным выходным состоянием +1, задающего смещение; $y_i^{(n-1)}=x_{ij}^{(n)}$ – i-ый вход нейрона j слоя n.

$$y_j^{(n)} = f(s_j^{(n)}), \text{ где } f() \text{ – сигмоид} \qquad (3.12)$$

$$y_q^{(0)}=I_q, \qquad (3.13)$$

где I_q – q-ая компонента вектора входного образа.

2. Рассчитать $\delta^{(N)}$ для выходного слоя по формуле (3.8).

Рассчитать по формуле (3.9) или (3.10) изменения весов $\Delta w^{(N)}$ слоя N.

3. Рассчитать по формулам (3.7) и (3.9) (или (3.7) и (3.10)) соответственно $\delta^{(n)}$ и $\Delta w^{(n)}$ для всех остальных слоев, n=N-1, N-2,...,1.

4. Скорректировать все веса в НС

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \qquad (3.14)$$

5. Если ошибка сети (3.1) превышает заданное пользователем значение, перейти на шаг 1. В противном случае – конец.

Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других

Из выражения (3.9) следует, что когда выходное значение $y_i^{(n-1)}$ стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов [0,1] желательно сдвинуть в пределы [-0.5, +0.5], что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой преобразуется к виду

$$f(x) = -0.5 + \frac{1}{1 + e^{-\alpha \cdot x}}. \qquad (3.15)$$

Полностью алгоритм обучения методом ОРО выглядит следующим образом:

На первом шаге на все входы сети подается один из возможных образов из обучающей выборки. В режиме нормального функционирования сети по формулам (3.2) и (3.3) вычисляются выходные значения.

На втором шаге рассчитывается по формуле (3.6) ошибка нейронов и по формуле (3.10) ошибка синаптических весов выходного слоя.

На третьем шаге вычисляются ошибки нейронов и синаптических весов для скрытых слоев в порядке, обратном нормальному выполнению суммирования.

На четвертом этапе происходит корректировка по формуле (3.10) синаптических весов всех слоев.

На пятом шаге полученная ошибка обучения сравнивается с максимально допустимой ошибкой и принимается решение об окончании или продолжении процесса обучения.

3.3. Нормализация входной и выходной информации

Входная и выходная информация для нейронной сети формируется из векторов IN (входные значения обучающей выборки), OUT (выходные значения обучающей выборки).

Нормализация значений необходима в виду того, что на веса синапсов сети обычно наложены требования принадлежности некоторому диапазону значений. Это приводит к тому, что обычно нельзя подавать сети входные сигналы в их истинном диапазоне величин и получать от сети выходные сигналы в требуемом диапазоне.

Кроме того, нормирование исключает доминирование одних входных сигналов перед другими [6,9,20-22,25-27,29].

Поэтому перед подачей входных сигналов их необходимо нормировать в одном из диапазонов: от минус 1 до 1(формула 3.17); от минус 0,5 до 0,5(формула 3.18); от 0 до 1(формула 3.19).

Наиболее простое нормирование сигналов можно выполнить следующим образом. Каждая компонента входного вектора данных x_i заменяется величиной, вычисляемой по формулам

$$x_{i0} = \frac{x_i - 0.5(\max\{x_i\} + \min\{x_i\})}{0.5(\max\{x_i\} - \min\{x_i\})}, \qquad (3.17)$$

$$x_{i0} = \frac{x_i - 0.5(\max\{x_i\} + \min\{x_i\})}{(\max\{x_i\} - \min\{x_i\})}, \qquad (3.18)$$

где x_{i0} – нормализованное значение величины;

x_i – нормализуемая величина;

max x_i и min x_i – соответственно максимальное и минимальное значения для данной компоненты, вычисленные по всей обучающей выборке.

Можно нормировать и по-другому, например, пересчитывая выборку так, чтобы разброс данным был единичным, по формуле

$$x_{i0} = \frac{x_i - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}}. \qquad (3.19)$$

Здесь имеется одна сложность. Любое изменение обучающей выборки должно соответственно менять и правило нормирования данных.

Выбор способа нормализации и значений зависит от конкретной обучающей выборки и выбирается экспериментально.

Для денормализации данных используются обратные преобразования, выполняемые по формулам 3.20, 3.21, 3.22:

Диапазон от 0 до 1.

$$x_i = \min\{x_i\} + x_{i0}(\max\{x_i\} - \min\{x_i\}); \qquad (3.20)$$

Диапазон от минус 1 до 1.

$$x_i = 0.5(\max\{x_i\} + \min\{x_i\}) + 0.5x_{i0}(\max\{x_i\} - \min\{x_i\}), \qquad (3.21)$$

Диапазон от минус 0,5 до 0,5.

$$x_i = 0.5(\max\{x_i\} + \min\{x_i\}) + x_{i0}(\max\{x_i\} - \min\{x_i\}). \qquad (3.22)$$

Обратное распространение – это самый популярный алгоритм для обучения НС с помощью изменения весов связей. Ошибка распространяется от выходного слоя к входному, т. е. в направлении противоположном направлению прохождения сигнала при нормальном функционировании сети.

Выполнение алгоритма начинается с генерации произвольных весов для многослойной сети. Затем процесс, описанный ниже, повторяется до тех пор, пока средняя ошибка на входе не будет признана достаточно малой:

1. Берётся пример входного сигнала с соответствующим правильным значением выхода, т. е. входной и выходной векторы.

2. Рассчитывается прямое распространение сигнала через сеть (определяются весовые суммы Si и активаторы ui для каждого нейрона).

3. Начиная с выходов, выполняется обратное движение через ячейки выходного и промежуточного слоя, при этом программа рассчитывает значения ошибок по формуле (3.23) и (3.24):

для нейрона выходного слоя

$$\delta_0 = (Ci-u5)*u5*(1-u5) \qquad (3.23)$$

для всех нейронов скрытого слоя

$$\delta_i = (\Sigma mm>i*w_{mj} * \delta_0)*u_i*(1- u_i) \qquad (3.24)$$

Здесь m обозначает все нейроны, связанные со скрытым узлом, w – заданный вектор веса, u – выход активационной функции.

4. Веса в сети обновляются следующим образом по формуле (3.25) и (3.26):

$$W*ij = wij - p*\delta_0*ui, \qquad (3.25)$$

для весов синапсов между скрытым и выходным слоем.

$$W*ij = wij - p*\delta i*ui, \qquad (3.26)$$

для весов синапсов между скрытым и входным слоем.

Здесь параметр p характеризует коэффициент скорости обучения (или размер шага). Это небольшое значение ограничивает изменение, которое может произойти на каждом шаге. Параметр p можно определить таким образом, чтобы он характеризовал скорость продвижения алгоритма обратного распространения к решению. Лучше начать обучение с небольшого значения (p=0,1) и затем постепенно его повышать.

Продвижение вперёд по сети соответствует активации нейронов, а продвижение назад – коррекции весов синапсов и весов смещений нейронов. Затем веса обновляются таким образом, чтобы минимизировать ошибку для данного входного вектора. Если коэффициент обучения слишком велик, сеть может никогда не сойтись, т. е. не будут найдены нужные веса синапсов при минимальной среднеквадратичной ошибке обучения. При малом значении коэффициента обучения увеличивается время обучения сети.

Пример расчёта.

Проход вперёд. Сначала выполняется расчёт движения входного сигнала по сети для примера, приведенного на рис.3.1.

U3=f(w3,1*u1+w3,2*u2+wb*смещение)

U3=f(0*1+0,5*1+ 1*1=f(1,5)

$f(s)=1/(1+e^{-\alpha*s})$

U3=0,81757 (при α=1)

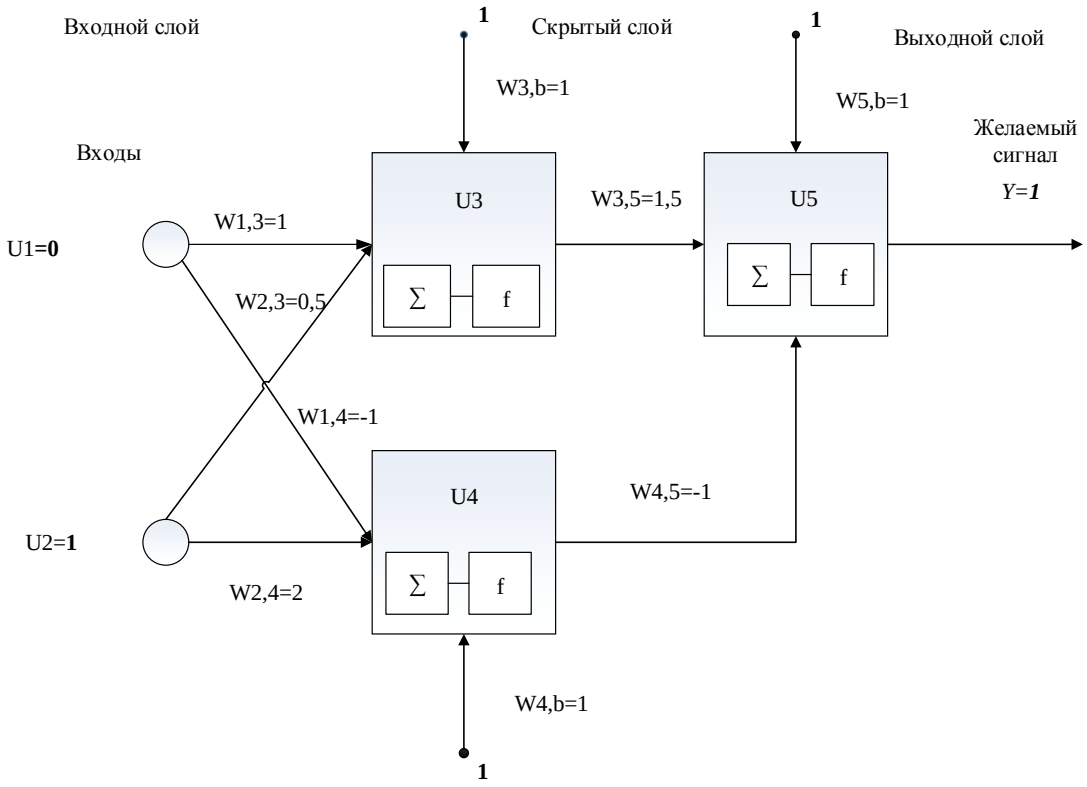


Рис.3.1. Пример архитектуры нейронной сети

Пример расчёта.

Проход вперёд. Сначала выполняется расчёт движения входного сигнала по сети (рис.3.1).

U3=f(w3,1*u1+w3,2*u2+wb*смещение)

U3=f(0*1+0,5*1+ 1*1=f(1,5)

$f(s)=1/(1+e^{-\alpha*s})$

U3=0,81757 (при α=1)

U4=f(w4,1*u1+w4,2*u2+wb*смещение)

U4=f (-1*0+2*1+ 1*1=f (3)

$f(s)=1/(1+e^{-\alpha*s})$

U4=0, 952574 (при α=1)

Теперь сигнал дошёл до скрытого слоя. Конечный шаг – переместить сигнал в выходной слой и рассчитать значение на выходе сети:

U5=f(w5,3*u3+w5,4*u4+wb*смещение)

U5=f (0, 81757*1, 5+0, 952574*(-1) + 1*1=f (1, 2195)

$f(s)=1/(1+e^{-\alpha*s})$

U5=0, 78139 (при α=1)

Правильный реакцией НС на входной сигнал является значение 1,0; значение рассчитанное сетью, составляет 0, 78139. Для коррекции весовых коэффициентов обычно используется среднеквадратичная ошибка, вычисляемая по формуле среднеквадратичной ошибки $E=0,5*\Sigma(y^{эталонное} - y^{расчётное})^2$. Здесь $y^{эталонное}$ берётся из обучающей выборки, а $y^{расчётное}$ вычисляется после подачи на входы нейронной сети входного вектора.

$E=0,5*(1.0 - 0,78139)^2=0,023895$.

Обратный проход.

По формуле (1) рассчитаем ошибку в выходном узле:

δ_0= (Ci-u5) *u5*(1-u5) =(1,0-0,78139) * 0,78139*(1- 0,78139) =0,0373

Теперь следует рассчитать ошибку для двух скрытых нейронов.

$\delta_{u4}= (\delta_0 * w5,4)*u4 *(1- u4)$

δ_{u4}=(0,0373* (-1)* 0,952574*(1 - 0,952574)= - 0,0016851

$\delta_{u3}= (\delta_0 * w5,3)*u3 *(1- u3)$

δ_{u3}= (0,0373* 1,5* 0,81757 * (1 - 0,81757)= 0, 0083449

Изменение весов соединений.

1. Сначала обновляются веса между выходным и скрытым слоем.

w*ij= wij + p*δ_0*ui

w5,4 = w 5,4 + (p*0,0373*u4)

w5, 4 = -1, 0 + (0, 5*0, 0373*0, 952574) = -0, 9882

w5, 3 = w 5, 3 + (p*0, 0373*u3)

w5, 3 = 1, 5 + (0, 5*0, 0373*0, 81757) = 1, 51525

2. Теперь нужно обновить смещение для выходного нейрона

w5,b = w 5,b + (p*0,0373*смещение5)

w5, b = 1 + (0, 5*0, 0373*1) =1, 01865

Для w5,4 вес изменён с -1 до -0,9882, а для w5,3 увеличен с 1,5 до 1,51525.

Смещение обновлено для повышения возбуждения нейрона.

3. Теперь обновляются веса между входным и скрытым слоем:

w4,2 = w 4,2 + (p*- 0,0016851*u2)

w4,2 = 2 + (0,5*- 0,0016851*1)=1,999916

w4,1 = w4,1 + (0,5*- 0,0016851*u1)

w4,1 = -1,0 + (0,5*- 0,0016851*0)= -1,0

w3,2 = w 3,2 + (p*0, 0083449*u2)

w3, 2 = 0, 5 + (0, 5*0, 0083449*1) = 0, 50417

w3, 1 = w3, 1 + (0, 5*0, 0083449*u1)

w3, 1 = 1, 0 + (0, 5*0, 0083449*0) = 1, 0

4. Теперь обновляются смещения для скрытых нейронов:

W4,b = w 4,b + (p*- 0,0016851*смещение4)

W4,b = 1 + (0,5*- 0,0016851*1)=0,99915

W3,b = w 3,b + (p*0, 0083449 *смещение3)

W3, b = 1 + (0, 5*0, 0083449*1) =1, 00417

5. Обновление весов для скрытого слоя завершается. Выполним ещё один проход вперёд.

U3=f(w3,1*u1+w3,2*u2+wb*смещение)

U3=f(0*1+0,50417*1+ 1,00417*1=f(1,50834)

$f(s)=1/(1+e^{-\alpha*s})$

U3=0,8188 (при α=1)

U4=f(w4,1*u1+w4,2*u2+wb*смещение)

U4=f(-1*0+1,99916*1+ 1,99916*1=f(2,99831)

U4=0,952497

U5=f(w5,3*u3+w5,4*u4+wb*смещение)

U5=f(1,51525*0,81888+(-0,9822)*0,952497 + 1,01865*1=f(1,32379)

$f(s)=1/(1+e^{-\alpha*s})$

U5=0, 7898 (при α=1)

$E=0,5*(1,0 – 0,7898)^2= 0,022$

На первой итерации ошибка была равна 0,023895, а на второй сократилась до 0,022.

Следовательно, алгоритм обратного распространения ошибки обеспечивает уменьшение среднеквадратичной ошибки обучения НС.

3.5. Недостатки алгоритма обратного распространения ошибки

Рассматриваемая НС имеет несколько **"узких мест**" [1-6].

Во-первых, в процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых

коэффициентов сместят рабочую точку на сигмоидах многих нейронов в область насыщения. Малые величины производной от логистической функции приведут к остановке обучения, что парализует НС.

Во-вторых, применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Эта проблема связана еще с одной, а именно – с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращения весов и, следовательно, скорость обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве p обычно выбирается число меньше 1, но не очень маленькое, например, 0.1, и оно, вообще говоря, может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов стабилизируются, p кратковременно сильно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

Существует и иной метод исключения локальных минимумов, а заодно и паралича НС, заключающийся в применении стохастических НС, но о них лучше поговорить отдельно.

3.6. Параметры, влияющие на обучение многослойной нейронной сети

Исследование влияния параметров на обучение многослойной нейронной сети может быть проведено на учебной инструментальной

системе, разработанной на кафедре ЭВМ [7]. Главное меню программы приведено на рис.3.2.

Для выявления параметров, влияющих на обучение многослойной нейронной сети, рассматривается задача аппроксимации функции F (x1, x2, x3) =x1*x1-x2/x3.

Сформировано 807 примеров обучающей выборки, отвечающих требованиям полноты, равномерности и непротиворечивости.

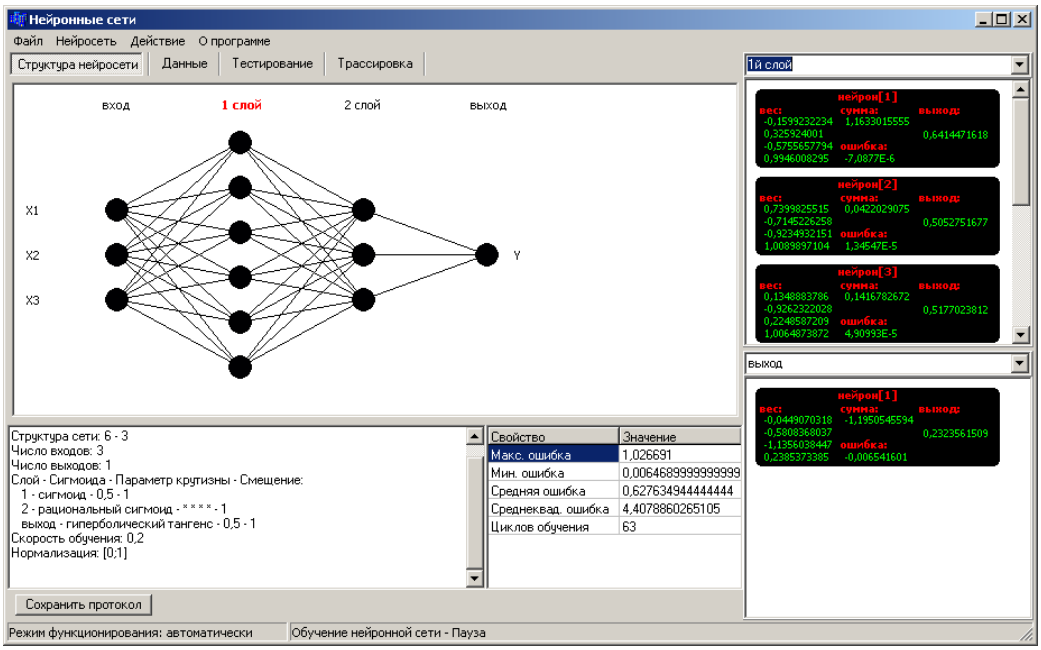


Рис.3.2. Главное меню программы

Нормализация очень сильно влияет на эффективность алгоритма ОРО. Если данные подаются на вход сети в ненормализованном виде, то среднеквадратичная ошибка получается очень большой. Для данной выборки большой ошибка получается большой при нормализации на интервалах[-0,5;0,5], [-1;1]. Намного меньшей ошибка получается при нормализации [0;1].

Вторым исследуемым параметром является порядок выбора примеров из обучающей выборки (последовательный или случайный). Установлено, что при случайном выборе и при нормализации [0;1]

среднеквадратичная ошибка меньше. Случайный метод выбора примеров из выборки не вызывает привыкания НС к выбираемым примерам.

Далее было выявлено, что сигмоидальная функция с большим коэффициентом крутизны, обеспечивает наименьшую среднеквадратичную ошибку.

Исследования демонстрировали следующую закономерность: чем больше величина смещения, тем среднеквадратичная ошибка меньше. В результате было выбрано смещение близкое к единице.

Скорость обучения сети влияет на скорость обучения и сходимость алгоритма обучения. По результатам экспериментов выбран параметр скорости обучения, равный единице, обеспечивающий меньшую среднеквадратичную ошибку и, естественно, меньшее время обучения сети.

Архитектура нейронной сети является одним из важнейших характеристик. Были исследованы НС с различным числом слоёв и нейронов в слое и выбрана структура 3-5-1(рис.3.3), предусматривающая 3 нейрона во входном слое, 5 нейронов – в скрытом слое и один нейрон – в выходном слое.

Одним из способов выхода из локального экстремума является применение параметра момента, значение которого выбрано 0,5.

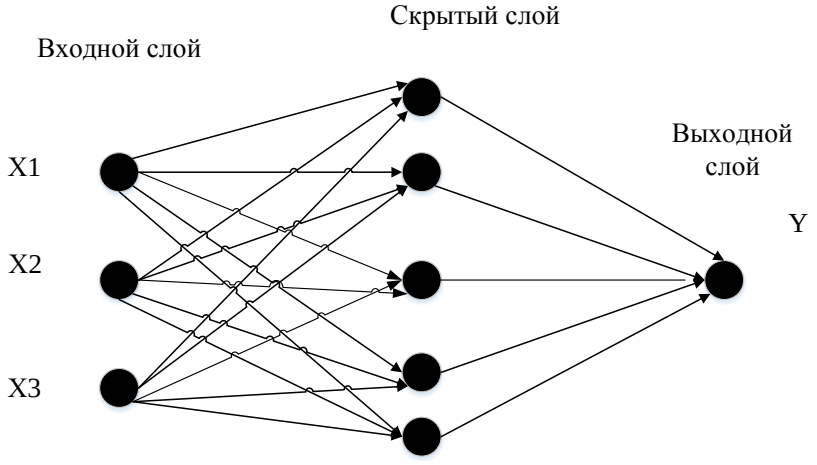


Рис.3.3. Структура нейронной сети (3-5-1)

3.7. Выполнение режима трассировки нейронной сети

Условия останова обучения нейронной сети:

Циклов обучения: 1000

Инициализация весов синапсов случайным образом...

Нейрон[1][1]

W [1, 1, 1] = 0,392

W [1, 1, 2] = 0,696

W [1, 1, 3] = -0,746

Вес смещения:

W [1, 1, 4] = 1

Нейрон[1][2]

W [1, 2, 1] = 0,128

W [1, 2, 2] = -0,528

W [1, 2, 3] = 0,178

Вес смещения:

W [1, 2, 4] = 1

Нейрон[1][3]

W [1, 3, 1] = 0,356

W [1, 3, 2] = 0,332

W [1, 3, 3] = -0,208

Вес смещения:

W [1, 3, 4] = 1

Нейрон[1][4]

W [1, 4, 1] = -0,962

W [1, 4, 2] = 0,102

W [1, 4, 3] = 0,522

Вес смещения:

W [1, 4, 4] = 1

Нейрон[1][5]

W [1, 5, 1] = 0,756

W [1, 5, 2] = 0,922

W [1, 5, 3] = 0,184

Вес смещения:

W [1, 5, 4] = 1

Нейрон[2][1]

W [2, 1, 1] = 0,238

W [2, 1, 2] = -0,464

W [2, 1, 3] = -0,34

W [2, 1, 4] = -0,406

W [2, 1, 5] = -0,598

Вес смещения:

W [2, 1, 6] = 1

Выбирается допустимый образ из обучающего множества.

0,246

0,254

0,260572

0,255454

Подаем сигнал на вход нейронной сети...

Нейрон[0][1]

Аксон = 0,246

Нейрон[0][2]

Аксон = 0,254

Нейрон[0][3]

Аксон = 0,260572

Прямая волна алгоритма. Выходы нейронов скрытого слоя.

Нейрон[1][1]

Взвешенная сумма = 1,078829288

Аксон = 0,6316762385

Нейрон[1][2]

Взвешенная сумма = 0,943757816

Аксон = 0,6158283722

Нейрон[1][3]

Взвешенная сумма = 1,117705024

Аксон = 0,6361869922

Нейрон[1][4]

Взвешенная сумма = 0,925274584

Аксон = 0,6136396283

Нейрон[1][5]

Взвешенная сумма = 1,468109248

Аксон = 0,6756944005

Нейрон[2][1]

Взвешенная сумма = -0,004911937868

Аксон = 0,4993860081

Обратная волна алгоритма. Подсчёт локальной ошибки нейронов.

Подсчет локальной ошибки нейронов на выходе нейронной сети...

Желаемый сигнал на выходе:

0,255454

Прогнозируемый сигнал на выходе нейронной сети:

0,4993860081

Нейрон[2][1]

Локальная ошибка = 0,03049145503

Подсчет локальной ошибки нейронов в скрытом слое нейронной сети...

Нейрон[1][1]

Локальная ошибка = 0,0008442078539

Нейрон[1][2]

Локальная ошибка = -0,001673597874

Нейрон[1][3]

Локальная ошибка = -0,001199748121

Нейрон[1][4]

Локальная ошибка = -0,001467506929

Нейрон[1][5]

Локальная ошибка = -0,001997809641

Коррекция весов синапсов...

W [1, 1, 1] = 0,3923182664

W [1, 1, 2] = 0,6963148895

W [1, 1, 3] = -0,7456878845

Вес смещения:

W [1, 1, 4] = 1

W [1, 2, 1] = 0,1273690536

W [1, 2, 2] = -0,528624252

W [1, 2, 3] = 0,1773812474

Вес смещения:

W [1, 2, 4] = 1

W [1, 3, 1] = 0,355547695

W [1, 3, 2] = 0,331552494

W [1, 3, 3] = -0,2084435637

Вес смещения:

W [1, 3, 4] = 1

W [1, 4, 1] = -0,9625532501

W [1, 4, 2] = 0,1014526199

W [1, 4, 3] = 0,5214574421

Вес смещения:

W [1, 4, 4] = 1

W [1, 5, 1] = 0,7552468258

W [1, 5, 2] = 0,921254817

W [1, 5, 3] = 0,1832613818

Вес смещения:

W [1, 5, 4] = 1

W [2, 1, 1] = 0,2436153637

W [2, 1, 2] = -0,458143024

W [2, 1, 3] = -0,334453406

W [2, 1, 4] = -0,4001096551

W [2, 1, 5] = -0,5930557252

Вес смещения:

W [2, 1, 6] = 1

Выполняется вычисление синапсов НС (табл.3.1) аналогично алгоритму, проведённому в п 3.4. В качестве активационной функции использована сигмоидальная функция с коэффициентом крутизны $\alpha=0,5$

$$F(s_i) = \frac{1}{1 + e^{-\alpha s_i}} \ ,$$

где α=0,5 – оптимальный для конкретной задачи коэффициент крутизны функции;

s_i – взвешенная сумма входов i-го нейрона.

Вес смещения и значение самого смещения взяты равные единице.

Среднеквадратичная ошибка выходного слоя рассчитывается по формуле

$$\delta_0 = \frac{1}{2}\sum_{i=1}^{N_i}(y_i - d_i)^2 \ ,$$

где y_i – рассчитанное значение выхода i-го нейрона в выходном слое;

d_i – требуемое значение выхода i-го нейрона в выходном слое;

N_i – количество нейронов в выходном слое.

В нашем случае.

В качестве активационной функции использована сигмоидальная функция с коэффициентом крутизны α=0,5.

Для расчёта обратной волны алгоритма (табл.3.2) необходимо вычислить производную активационной функции, формула которой приведена ниже.

$$F'(s_i) = \frac{\alpha \cdot e^{-\alpha s_i}}{(1 + e^{-\alpha s_i})^2} \ .$$

Расчёт весов и смещений выполняется по формулам

$$w_{ij}(t+1) = w_{ij}(t) - p\delta_i F'(s_i) x_j \ ,$$

$$T_j(t+1) = T_j(t) - p\delta_i F'(s_i) \ .$$

Параметр скорости обучения принят равный 0,5, вес смещения равный единице.

Таблица 3.1. Расчет параметров для «прямой волны» алгоритма

Номер слоя	Номер нейрона	Номер входа	Входной сигнал, xj	Весовой коэффициент wij	wij*xj	Взвешенная сумма, Si	Выход нейрона, yi = F(Si)
Входной слой	1	1	0,246	-	-	-	0,246
	2	1	0,254	-	-	-	0,254
	3	1	0,26057	-	-	-	0,260572
Скрытый слой	1	1	0,246	0,392	0,0964	1,0788292	0,631676
		2	0,254	0,696	0,1767		
		3	0,26057	-0,746	-0,1943		
	2	1	0,246	0,128	0,0314	0,9437578	0,615828
		2	0,254	-0,528	-0,1341		
		3	0,260572	0,178	0,0463		
	3	1	0,246	0,356	0,0875	1,117705	0,636186
		2	0,254	0,332	0,0843		
		3	0,260572	-0,208	-0,0541		
	4	1	0,246	-0,962	-0,2366	0,925274	0,613639
		2	0,254	0,102	0,0259		
		3	0,260572	0,522	0,1360		
	5	1	0,246	0,756	0,1859	1,468109	0,675694
		2	0,254	0,922	0,2341		
		3	0,260572	0,184	0,0479		
Выходной слой	1	1	0,631676	0,238	0,1503	-0,004911	0,499386
		2	0,615828	-0,464	-0,2857		
		3	0,636186	-0,34	-0,2163		
		4	0,613639	-0,406	-0,2491		
		4	0,675694	-0,598	-0,4040		

Все расчеты совпали с результатами работы программы в режиме трассировки с точностью до 10^{-3}–10^{-6}, лишь при расчете весов смещений точность снизилась до 10^{-3} из-за погрешности округления ручных вычислений.

Таблица 3.2. Расчет параметров для «обратной волны» алгоритма

Эталонный результат, на выходе				0,25545			
Среднеквадратичная ошибка:				0,02975			
Слой	Номер нейрона	Эталон di	Прогноз, yi	$\delta m^{(k+1)}$	$\varepsilon i^{(k)}$	Si	F'(Si)
Выход	1	0,2554	0,499	-	0,316503	2,186540	0,094000
1	1	-	-	0,02975	0,007080	1,078829	0,116330
	2	-	-	0,02975	-0,01380	0,943757	0,118291
	3	-	-	0,02975	-0,01011	1,117705	0,115726
	4	-	-	0,02975	-0,01207	0,925274	0,118543
	5	-	-	0,0297	-0,01779	1,468109	0,109565

Из расчёта параметров весов синапсов на обратной «волне» следует, что ошибка (по модулю) уменьшается в направлении от выходного слоя к первому скрытому. После расчёта ошибок было выполнено изменение весовых коэффициентов и весов смещения.

Таблица 3.3. Расчет весов синапсов и смещений обратной «волны»

Номер нейрона	Номер входа	Входной сигнал xj	Весовой коэффициент wij(t)	Si	F'(Si)	δi	Вес wij(t+1)	Вес смещения Tj(t+1)
1	1	0,246	0,392	1,078	0,11633	0,00082	0,3920	0,9999
	2	0,254	0,696				0,6959	
	3	0,2605	-0,746				-0,746	
2	1	0,246	0,128	0,943	0,11829	-0,00163	0,1280	1,0000
	2	0,254	-0,528				-0,527	
	3	0,2605	0,178				0,1780	
3	1	0,246	0,356	1,117	0,11572	-0,00117	0,3560	1,0000
	2	0,254	0,332				0,3320	
	3	0,2605	-0,208				-0,207	
4	1	0,246	-0,962	0,9252	0,118543	-0,00143	-0,961	1,0000
	2	0,254	0,102				0,1020	
	3	0,2605	0,522				0,5220	
5	1	0,246	0,756	1,468	0,10956	-0,00194	0,7560	1,0001
	2	0,254	0,922				0,9220	
	3	0,2605	0,184				0,1840	
1	1	0,6316	0,238	2,186	0,09400	0,02975	0,2371	0,9986
	2	0,6158	-0,464				-0,464	
	3	0,6361	-0,34				-0,340	
	4	0,6136	-0,406				-0,406	
	4	0,6756	-0,598				-0,598	

4. НЕЙРОННЫЕ СЕТИ С РАДИАЛЬНО-БАЗИСНЫМИ ФУНКЦИЯМИ

4.1. Общие сведения о нейронных сетях с радиальными базисными функциями

Многослойные нейронные сети выполняют аппроксимацию функции нескольких переменных путем преобразования множества входных переменных во множество выходных переменных.

Существует способ отображения входного множества в выходное множество, заключающийся в преобразовании путем адаптации нескольких одиночных аппроксимирующих функций к ожидаемым значениям, причем эта адаптация проводится только в ограниченной области многомерного пространства.

При таком подходе отображение всего множества данных представляет собой сумму локальных преобразований. С учётом роли, которую играют скрытые нейроны, преобразования составляют множество базисных функций локального типа. Выполнение одиночных функций (при ненулевых значениях) регистрируется только в ограниченной области пространства данных - отсюда и название локальная аппроксимация.

Особое семейство образуют сети с радиальной базисной функцией, в которых нейроны реализуют функции, радиально изменяющиеся вокруг выбранного центра и принимающие ненулевые значения только в окрестности этого центра. Подобные функции, определяемые в виде

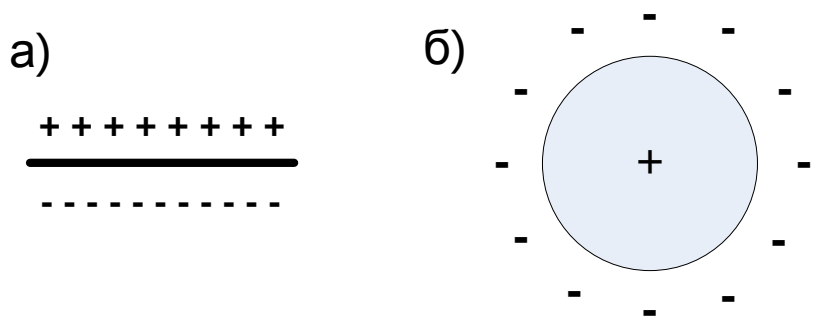
$$H(x) = H(||x-c||), \qquad (4.1)$$

где x – выходы нейронов первого слоя;

c – центр радиальной базисной функции.

Такие функции называются радиальными базисными функциями. В них роль нейрона заключается в отображении радиального пространства вокруг одиночной заданной точки (центра), либо вокруг группы таких точек, образующих кластер. Суперпозиция сигналов, поступающих от всех таких нейронов, которая выполняется выходным нейроном, позволяет получить отображение всего многомерного пространства.

Сети радиального типа представляют собой естественное дополнение сигмоидальных сетей. Сигмоидальный нейрон представляется в многомерном пространстве гиперплоскостью, разделяющей это пространство на две категории (два класса), в которых выполняется одно из двух условий: либо (u, x)> 0, либо (u, x) <0. Такой подход продемонстрирован на рис. 4.1а.



а) сигмоидальным нейроном; б) радиальным нейроном

Рис. 4.1. Иллюстрация способов разделения пространства данных

В свою очередь, радиальный нейрон представляет собой гиперсферу, которая осуществляет шаровое разделение пространства вокруг центральной точки в соответствии с рис. 4.1б. Именно с этой точки зрения он является естественным дополнением сигмоидального нейрона, поскольку в случае круговой симметрии данных позволяет заметно уменьшить количество нейронов, необходимых для разделения различных классов. Поскольку нейроны могут выполнять различные функции, в

радиальных сетях отсутствует необходимость использования большого количества скрытых слоев.

Структура типичной радиальной сети включает входной слой, на который подаются сигналы, описываемые входным вектором, скрытый слой с нейронами радиального типа и выходной слой, состоящий, как правило, из одного или нескольких линейных нейронов. Функция выходного нейрона сводится исключительно к взвешенному суммированию сигналов, генерируемых скрытыми нейронами [3].

На рис.4.2 представлена структура нейронной сети с радиальными базисными функциями.

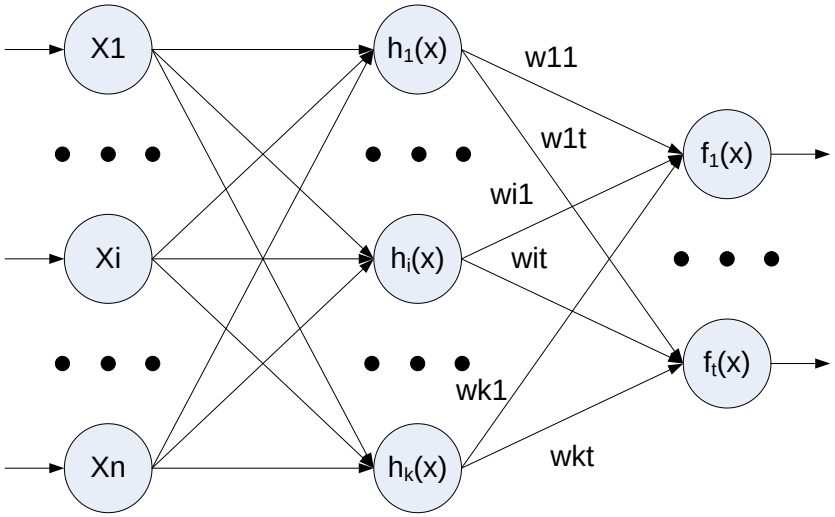


Рис.4.2. Структура нейронной сети с радиальными базисными функциями

В математической статистике в качестве радиальных базисных функций часто используют полиномиальные функции

$$h_j(x) = x^j, \qquad (4.2)$$

где x – выходы нейронов первого слоя;

j – коэффициент искривления.

Комбинация синусоидальных функций (ряды Фурье) часто используется при обработке сигналов

$$h(x) = \sin\left(\frac{2\pi j(x-\theta_j)}{m}\right), \tag{4.3}$$

где θ_j – центр функции;

m – радиус функции.

Логистические функции наиболее популярны в многослойных искусственных нейронных сетях

$$h(x) = \frac{1}{1+\exp(bx-b_0)}, \tag{4.4}$$

где b – центр функции;

b_0 – коэффициент подстройки.

Наиболее распространенной функцией, применяемой для распознавания образов, является функция Гаусса

$$h(x,c) = e^{\frac{-\sum_{j=1}^{n}\left(x_j-c_j\right)^2}{\delta^2}}, \tag{4.5}$$

где c = (c_1,c_2,…,c_n) - вектор координат центра активационной функции нейрона скрытого шара;

δ - ширина окна активационной функции нейрона скрытого шара [4].

4.2. Математические основы функционирования радиальных нейронных сетей

Математическую основу функционирования радиальных сетей составляет теорема Т. Ковера о распознаваемости образов, в соответствии с которой нелинейные проекции образов в некоторое многомерное пространство могут быть линейно разделены с большей вероятностью, чем при их проекции в пространство с меньшей размерностью [1].

Если вектор радиальных функций в N-мерном входном пространстве обозначить h(x), то это пространство является нелинейно h - разделяемым на два пространственных класса X^+ и X^- тогда, когда существует такой вектор весов, что

$$\begin{aligned} u^T h(x) > 0 \ , \ x \in X^+, \\ u^T h(x) < 0 \ , \ x \in X^-. \end{aligned} \qquad (4.6)$$

Граница между этими классами определяется уравнением

$$u^T h(x) = 0. \qquad (4.7)$$

Доказано, что каждое множество образов, случайным образом размещенных в многомерном пространстве, является h - разделяемым с вероятностью 1 при условии соответственно большой размерности этого пространства. На практике это означает, что применение достаточно большого количества скрытых нейронов, реализующих радиальные функции h(x), гарантирует решение задачи классификации при построении всего лишь двухслойной сети: скрытый слой должен реализовать вектор h(x), а выходной слой может состоять из единственного линейного

нейрона, который выполняет суммирование выходных сигналов от скрытых нейронов с весовыми коэффициентами, заданными вектором w [2].

Простейшая нейронная сеть радиального типа функционирует по принципу многомерной интерполяции, состоящей в отображении *p* различных входных векторов x_i *(i* = 1, 2, ... *p)* из входного N-мерного пространства во множество из *p* рациональных чисел y_i *(i* = 1, 2, ... *p).* Для реализации этого процесса необходимо использовать *p* скрытых нейронов радиального типа и задать такую функцию отображения *F(x),* для которой выполняется условие интерполяции

$$F(x_i)=y_i. \qquad (4.8)$$

Использование *p* скрытых нейронов, соединяемых связями с весами w_i с выходными линейными нейронами, означает формирование выходных сигналов сети путем суммирования взвешенных значений соответствующих базисных функций. Рассмотрим радиальную сеть с одним выходом и *p* обучающими парами (x_i,y_i). Примем, что координаты каждого из *p* центров узлов сети определяются одним из векторов x_i. В этом случае взаимосвязь между входными и выходными сигналами сети может быть определена системой уравнений, линейных относительно весов w.

Если предположить, что параметры функции Гаусса, смещение с и радиус δ фиксированы, то есть каким-то образом уже определены, то задача нахождения весов решается методами линейной алгебры. Этот метод называется методом псевдо обратных матриц, и он минимизирует средний квадрат ошибки. Суть этого метода основана на вычислении интерполяционной матрицы H [1,4].

$$H = \begin{bmatrix} h_1(x_1) & h_2(x_1) & \dots & h_m(x_1) \\ h_1(x_2) & h_2(x_2) & \dots & h_m(x_2) \\ \dots & \dots & \dots & \dots \\ h_1(x_p) & h_2(x_p) & \dots & h_m(x_p) \end{bmatrix}. \quad (4.9)$$

На следующем этапе вычисляется инверсия произведения матрицы H на транспонированную матрицу H^T

$$A^{-1} = \left(H^T H\right)^{-1}. \quad (4.10)$$

Окончательный результат, матрица весов, рассчитывается по формуле

$$W = A^{-1} H^T y. \quad (4.11)$$

Полученная архитектура радиальных сетей имеет структуру, аналогичную многослойной структуре сигмоидальных сетей с одним скрытым слоем. Роль скрытых нейронов в ней играют базисные радиальные функции, отличающиеся своей формой от сигмоидальных функций. Несмотря на отмеченное сходство, сети этих типов принципиально отличаются друг от друга.

Радиальная сеть имеет фиксированную структуру с одним скрытым слоем и линейными выходными нейронами, тогда как сигмоидальная сеть может содержать различное количество слоев, а выходные нейроны бывают как линейными, так и нелинейными. Используемые радиальные функции могут иметь весьма разнообразную структуру.

Нелинейная радиальная функция каждого скрытого нейрона имеет свои значения параметров c_i и s_i, тогда как в сигмоидальной сети применяются, как правило, стандартные функции активации с одним и тем же для всех нейронов параметром β. Аргументом радиальной функции является эвклидово расстояние образца x от центра c_i, а в сигмоидальной сети это скалярное произведение векторов $w^T x$.

Еще большие отличия между этими сетями можно заметить при детальном сравнении их структур. Сигмоидальная сеть имеет многослойную структуру, в которой способ упорядочения нейронов повторяется от слоя к слою. Каждый нейрон в ней выполняет суммирование сигналов с последующей активацией.

Структура радиальной сети несколько иная. На рис.4.2 изображена подробная схема сети РБФ с радиальной функцией при классическом понимании эвклидовой метрики. Первый слой составляют нелинейные радиальные функции, параметры которых (центры c_i и коэффициенты s_i) уточняются в процессе обучения. Первый слой не содержит линейных весов в понимании, характерном для сигмоидальной сети.

4.3. Нелинейная модель расчёта параметров радиальной базисной функции

Если предыдущее предположение о фиксированных параметрах функции активации не выполняется, то есть помимо весов необходимо настроить параметры активационной функции каждого нейрона (смещение функции и её радиус), задача становится нелинейной. Решать ее приходится с использованием итеративных численных методов оптимизации, например, градиентных методов.

Расположение центров должно соответствовать кластерам, реально присутствующим в исходных данных. Рассмотрим два наиболее часто используемых метода.

Выборка из выборки. В качестве центров радиальных элементов берутся несколько случайно выбранных точек обучающего множества. В силу случайности выбора они "представляют" распределение обучающих данных в статистическом смысле. Однако, если число радиальных элементов невелико, такое представление может быть неудовлетворительным.

Алгоритм К-средних. Этот алгоритм стремится выбрать оптимальное множество точек, являющихся центроидами кластеров в обучающих данных. При K радиальных элементах их центры располагаются таким образом, чтобы каждая обучающая точка "относилась" к одному центру кластера и лежала к нему ближе, чем к любому другому центру и каждый центр кластера был центроидом множества обучающих точек, относящихся к этому кластеру.

После того, как определено расположение центров, нужно найти отклонения. Величина отклонения (ее также называют сглаживающим фактором) определяет, насколько "острой" будет гауссова функция.

Если эти функции выбраны слишком острыми, сеть не будет интерполировать данные между известными точками, и потеряет способность к обобщению. Если же гауссовы функции взяты чересчур широкими, сеть не будет воспринимать мелкие детали.

На самом деле сказанное - еще одна форма проявления дилеммы переобучения и недообучения. Как правило, отклонения выбираются таким образом, чтобы колпак каждой гауссовой функций захватывал "несколько" соседних центров. Для этого имеется несколько методов: явный (отклонения задаются пользователем); изотропный; метод К-средних.

В изотропном методе (отклонение берется одинаковым для всех элементов и определяется эвристически с учетом количества радиальных элементов и объема покрываемого пространства);

В методе К-средних (отклонение каждого элемента устанавливается (индивидуально) равным среднему расстоянию до его «К ближайших соседей», тем самым отклонения будут меньше в тех частях пространства, где точки расположены густо, - здесь будут хорошо учитываться детали, - а там, где точек мало, отклонения будут большими и будет производится интерполяция).

В последнее время получили распространение методы обучения нейронных сетей с радиальными базисными функциями, в которых используется сочетание генетических алгоритмов для подбора параметров активационных функций и методов линейной алгебры для расчета весовых коэффициентов выходного слоя. На каждой итерации поиска генетический алгоритм самостоятельно выбирает в каких точках пространства входных сигналов сети разместить центры активационных функций нейронов скрытого слоя и назначает для каждой из них ширину окна. Для полученной таким образом совокупности параметров скрытого слоя вычисляются веса выходного слоя и получающаяся при этом ошибка аппроксимации, которая служит для генетического алгоритма целевой функцией. На следующей итерации генетический вариант отбрасывает «плохие» варианты и использует наборы, показавшие наилучшие результаты на предыдущей итерации [1,3,4,9,10].

5. САМООРГАНИЗУЮЩИЕСЯ НЕЙРОННЫЕ СЕТИ

5.1. Принцип работы сети Кохонена

Примером сети, использующей алгоритм обучения без учителя, является самоорганизующая карта Кохонена (1982 год). Другое название сети Кохонена –KNC (Kohonen Clastering Networks). KNC используют для отображения нелинейных зависимостей на двумерные (чаще всего) сетки, представляющие метрические и топологические зависимости входных векторов, объединяемых в кластеры.

Нейронная сеть Кохонена имеет один слой нейронов. Количество входов каждого нейрона равно размерности входного вектора. Количество нейронов непосредственно определяет сколько различных кластеров сеть может распознать.

Кора человеческого мозга разбита на участки. Например, участок, ответственный за ступни ног, примыкает к участку, контролирующему движение всей ноги и т.д. Все элементы человеческого тела отображаются на эту поверхность.

Основная цель обучения в KNC состоит в выявлении структуры в n-мерных входных данных и предоставлении ее на карте в виде распределенных нейронных активностей. Каждый нейрон несет информацию о кластере, объединяющем в группу схожие по критерию близости входные вектора, формируя для данной группы собирательный образ.

Подобные вектора активизируют подобные нейроны, т. е. KNC *способна к обобщению.* Конкретному кластеру может соответствовать и несколько нейронов с близкими значениями векторов весов, поэтому

выход из строя одного нейрона не так критичен к ошибке распознавания, как это имеет место в сети Хемминга [1,3,4].

В большинстве случаев каждый выходной нейрон связан со своими соседями. Эти внутрислойные связи играют важную роль в процессе обучения, так как корректировка весов происходит не для всех весов сети, а только в окрестности этого элемента.

Сеть Кохонена использует состязательный конкурентный алгоритм обучения. Выигрывает тот нейрон, чей вектор весов наиболее близок к текущему входному вектору. Например, в смысле расстояния определяемой евклидовой метрикой.

Пример. В качестве близости двух образов X1 и X2 можно принять евклидово расстояние между ними

$$D(X1, X2) = \sqrt{\sum(X1-X2)^2} . \qquad (5.1)$$

У нейрона – победителя это расстояние будет меньше, чем у остальных. В другом варианте обучения победителем считается нейрон, весовой вектор которого имеет наибольшее скалярное произведение со входным вектором, так как скалярное произведение – это проекция входного вектора на вектор весов. В этом методе оба вектора должны быть нормализованы по длине.

5.2. Алгоритм обучения сети Кохонена

Вход каждого нейрона характеризуется весовым вектором той же размерности, что и входной. Число параметров сети определяется n*k, где n –число входов; k –число нейронов. Свойства сети определяются размерностью массива нейронов, числом нейронов в каждом измерении, формой окрестности выигравшего нейрона, законом сжатия окрестности и

скоростью обучения. Начальный размер окрестности выбирается в пределах от 1/2 до 2/3 размера сети и сокращается, например, по обратно пропорциональной зависимости.

1. Инициализировать весовые коэффициенты для всех выходных нейронов (матрица W^k_n) случайными малыми величинами. Вычислить усредненное начальное расстояние между обучающими векторами D_0.

$$D_0 = 1/N * \sum_{i=1}^{N} \min \text{dist}(Xi, W^j_n), \qquad (5.2)$$

где N -число примеров обучающей выборки;

j- номер нейрона для которого расстояние dist min.

Установить размер окрестности для выигравшего нейрона r (радиус стимуляции).

2. Положить $D_0 = Dt$

3. Подать на вход сети очередной входной вектор Xt(t), где t-номер итерации.

4. Для каждого нейрона j определить его расстояние $\text{dist}(Xi, W^j_n)$ по формуле (1) для любого j.

5. Выбрать нейрон–победитель m=m*, для которого расстояние минимально. Поиск ведется по величине отклонения входного вектора от весового вектора каждого нейрона.

6. Модифицировать весовые коэффициенты выигравшего нейрона

$$W^{m*}(t+1) = W^m(t) + \mu * \text{dist}(Xi(t), W^m_n(t)) \qquad (5.3)$$

и тех нейронов s , которые находятся в окрестности выигравшего

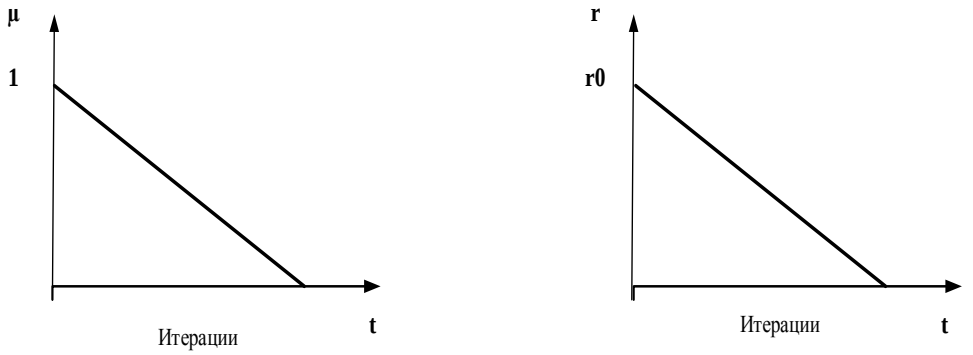
$$W^j(t+1) = W^j(t) + \mu * \text{dist}(Xi(t), W^m_n(t)) \qquad (5.4)$$

7. Повторить п.3-6 для векторов обучающей выборки.

8. Положить $W^k_n(t)\,(t+1) = W^k_n(t)$ и вычислить D_t по формуле (5.2). Подсчитать $\varepsilon = (D_{t+1,} D_{t)} / D_{t.}$

Если ε= <η, где η –априорно заданная положительная пороговая величина, то перейти к п.9, иначе t=(t+1), изменить параметр скорости обучения μ, параметр размера окрестности r_0 (рис.5.1) и перейти к п.2

9. Конец.



а) изменение скорости обучения б) изменение радиуса окрестности

Рис.5.1. Графики изменения параметров обучения сети Кохонена

5.3. Сети на встречного распространения

Одной из наиболее популярных комбинированных сетей является двухслойная сеть встречного распространения CPN (Counterpropagation Network), первым слоем которой является сеть Кохонена, а второй – выходная звезда (нейронная сеть) Гроссберга [1,3,4].

Нейрон в форме входной звезды имеет n входов, которым соответствуют весовые коэффициенты W= (w1, w2,…,wn), один выход Y, являющийся взвешенной суммой этих входов. Таким образом, звезда

является детектором состояния входов и реагирует только на свой входной вектор.

Подстройка весов производится по формуле

$$Wi(t+1)=Wi(t)+ \mu*(\{Xi- Wi(t)), \quad (5.5)$$

где Wi(t) –весовой вектор i-ой входной звезды на t-м такте обучения;

μ- параметр скорости обучения (выбирается в начале 0,1-0,2 и затем постепенно уменьшается);

Xi- входной вектор.

Выходная звезда Гроссберга выполняет противоположную функцию- при поступлении сигнала на вход выдается определенный вектор. Нейрон этого типа имеет один вход и m выходов с весами W= (w1, w2,…,wn), которые подстраиваются по формуле

$$Wi(t+1)=Wi(t)+\alpha*(\{ Yi- Wi(t)), \quad (5.6)$$

где Wi(t) – весовой вектор i-ой выходной звезды на t-м такте обучения;

Yi –выходной вектор; α - скорость обучения. Рекомендуется начать обучение с α =1 и постепенно уменьшается до 0.

Особенностью нейронов в форме звезд Гроссберга является избирательность памяти. Каждый нейрон помнит свой входной образ и игнорирует остальные. Каждой выходной звезде соответствует конкретная командная функция. Образ памяти связывается с определенным нейроном и а не возникает вследствие взаимодействия нейронов.

Сеть со встречным распространением CPN представляет собой соединение самоорганизующейся сети Кохонена и выходной звезды Гроссберга. Топология сети представлена на рис. 5.2. Все компоненты входного вектора X соединены со всеми нейронами сети Кохонена, а все его выходы соединены с нейронами слоя Гроссберга. Для упрощения на рис. 5.2 показаны не все связи.

Здесь выходы сети Кохонена не являются выходами сети, а служат лишь входами для выходного слоя – *выходной звезды Гроссберга.*

Создатель этой сети З. Хехт-Нильсен рекомендует использовать эти архитектуры для решения задач аппроксимации функций и заполнения пробелов в таблице данных.

Обучение сети встречного распространения состоит из двух шагов:

1. На первом шаге весовые векторы слоя Кохонена настраиваются таким образом, чтобы провести распределение входных векторов по классам, каждый из которых соответствует одному нейрону победителю. Обучение проводится без учителя. Точность кластеризации гарантируется только при условии представительной выборки.

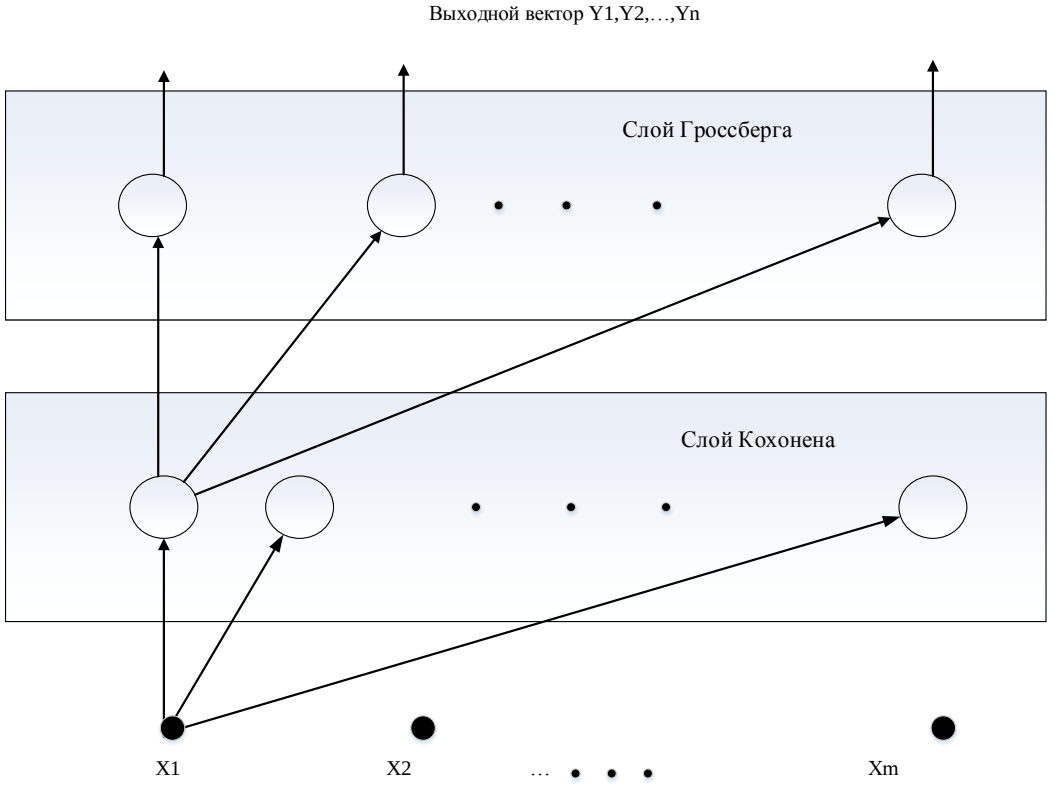


Рис.5.2. Топология нейронной сети встречного распространения

2. На втором шаге осуществляется обучение с учителем. Проводится подстройка весовых коэффициентов выходного слоя Гроссберга на примерах с заданным выходом по формуле (5.5).

При этом настраиваются только веса, соответствующие связям с теми элементами слоя Кохонена, которые являются победителями в текущем такте обучения (выигравшие элементы посылают выходной сигнал равный 1).

Темпы обучения нейронов слоя Кохонена и Гроссберга должны быть согласованы. Кроме того, в слое Кохонена подстраиваются также веса всех нейронов в окрестности победителя, которая постепенно сужаться до одного нейрона.

При функционировании сети в режиме распознавания нейроны слоя Гроссберга по сигналу «нейрона-победителя» в слое Кохонена воспроизводят на выходах сети образ в соответствии со значениями его весовых коэффициентов. В случае, когда слой Гроссберга состоит из одного элемента, полученный *скалярный выход равен одному из весов,* соответствующих связям этого элемента (с выигравшим нейроном).

Одной из особенностей *сети встречного распространения* является возможность восстановления пары векторов (X, Y) по одной известной компоненте. При предъявлении сети на этапе распознавания только входного вектора (с нулевым значением Y) восстанавливается Y, и наоборот, при предъявлении сети на этапе распознавания только выходного вектора Y, может быть восстановлен X.

Обученная НС может функционировать в режиме интерполяции, когда в слое Кохонена может быть несколько победителей. В этом случае уровни их нейронной активности нормируются таким образом, чтобы в сумме получалась единица; тогда выходной вектор определяется по сумме выходных векторов каждой из активных звезд Гроссберга. Тогда *нейронная сеть встречного распространения осуществляет* линейную интерполяцию между значениями выходных векторов, отвечающих нескольким кластерам [1,3,4].

Разработанная для учебного процесса программная модель нейронной сети встречного распространения [15] позволяет исследовать функционирование нейронной сети при различных параметрах и демонстрирует эффективность для создания нейросетевых экспертных систем.

6. НЕЙРОННЫЕ СЕТИ АДАПТИВНОЙ РЕЗОНАНСНОЙ ТЕОРИИ

6.1. Сети на основе теории адаптивного резонанса

Восприятие внешнего мира связано с решением дилеммы *«стабильности – пластичности».* Является ли воспринимаемый образ новой информацией или представляет вариант старой и ее не требуется запоминать. Таким образом, память человека должна оставаться *пластичной,* способной к восприятию новых образов, и в то же время сохранять *стабильность*, гарантирующую не разрушение старых образов [1].

Ранее рассмотренные нейронные сети не приспособлены к решению этой задачи. Например, в многослойном персептроне, после предъявления нового входного вектора изменяются весовые коэффициенты, и нет гарантии, что старые образы не разрушаться.

Аналогичная ситуация имеет место в сетях Кохонена, обучающихся на основе самоорганизации. Данные сети всегда выдают положительный результат при классификации и не способны отделить новые образы от искаженных образов.

Семейство сетей на основе теории адаптивного резонанса, разработанное Гроссбергом, применительно к биологическим структурам, обладает свойством *«стабильности – пластичности».* Adaptive Resonance Theory Network (ART). Такие нейронные сети называются сетями адаптивной резонансной теории (АРТ).

6.2. Нейронная сеть ART-1

Сеть ART-1 ориентирована на обработку образов, содержащих двоичную информацию. Сети ART-2 и ART-3 могут работать с непрерывной информацией [1,4].

Сеть ART -1 является классификатором входных двоичных образов, сформированным сетью по нескольким классам прототипов. В данной сети используется алгоритм обучения без учителя на основе конкуренции.

Сеть ART -1 состоит из 5 блоков (рис.6.1): слоя сравнения F1; слоя распознавания F2; трёх управляющих блоков: блока управления G1, блока управления G2, блока сброса G3.

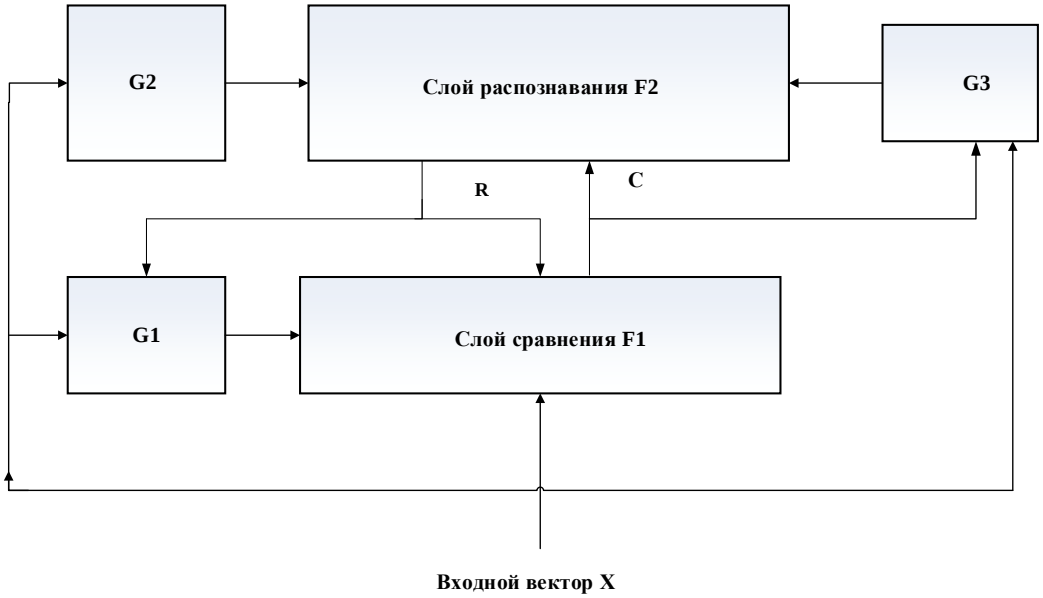


Рис.6.1. Структурная схема нейронной сети АРТ-1

Начальное значение нейрона управления G1 равно 1. Входной вектор X поступает на слой сравнения, который по правилу 2 из 3 пропускает его на слой распознавания и C=X. Каждый из нейронов слоя сравнения имеет 3 входа: входной сигнал X, сигнал соответствующего нейрона управления G1, сигнал обратной связи из слоя распознавания R в начальный момент равен нулю.

Каждый нейрон слоя распознавания имеет вектор весов Bj. При этом возбуждается только один нейрон слоя распознавания, вектор весов которого наиболее близок к вектору C. Это обеспечивается латеральным торможением остальных нейронов слоя.

Сигнал обратной связи нейрона- победителя поступает обратно в слой сравнения через весовые коэффициенты Tj. По существу этот вектор является носителем критических черт образа. Важным понятием в теории резонанса является шаблон критических черт. Не все черты (признаки), представленные в образе, являются существенными для восприятия. Результат распознавания определяется присутствием критических черт в представленном образе.

Выход нейрона управления G1=1, когда образ X имеет ненулевые компоненты. Он выполняет функции детектора новизны поступающего входного образа. Однако, когда возникает ненулевой отклик R из слоя распознавания, значение нейрона G1становится равным нулю.

Сигнал нейрона G2 устанавливается в единицу при ненулевом векторе X. Задачей оттого нейрона является погашение активности в слое распознавания, если в сеть не поступило никакой информации.

При генерации отклика R из слоя распознавания выход G1=0 и нейроны слоя сравнения активизируются сигналами X и R. Правило 2 из 3 приводит к активизации тех нейронов слоя сравнения, для которых и X и R являются единичными. В результате выход C теперь уже не равен вектору X, а содержит только те компоненты, которые определяют критические черты входного образа. Этот механизм в теории ART -1 получил название *адаптивной фильтрации* образа X.

Теперь задачей является установить, достаточен ли набор критических черт для окончательного отнесения образа X к классу нейрона-победителя. Эту функцию выполняет нейрон сброса, который измеряет сходство между векторами X и C. Выход нейрона сброса

определяется отношением числа единичных компонент в векторе С к числу единичных компонент в векторе Х. Если это отношение ниже определенного уровня сходства, нейрон вырабатывает сигнал сброса в слой распознавания, т. е. уровень резонанса недостаточен. Параметр ρ <1 определяет уровень сходства.

$\rho = (Q \cap H) / H$,

где ∩ – знак пересечения;

Н –количество единиц в векторе Х;

Q- количество единиц в векторе С.

Пример Х=110011100; Н=4; С=11001000; Q=3.

Тогда $\boldsymbol{\rho} = (Q \cap H) / H = ¾ = 0{,}75$

Данный параметр изменяется от 1 (полное соответствие) до 0 (наихудшее соответствие).

6.3. Процесс функционирования сетей ART -1

Начальное состояние сети. Нулевые значения компонент входного вектора Х устанавливают нейрон G2=0 и одновременно устанавливаются в ноль выходы нейронов слоя распознавания. При возникновении ненулевых значений в векторе Х нейроны управления G1 и G2 устанавливаются в единицу. По правилу 2 из 3 выходы С равны вектору Х.

Вектор С поступает в слой распознавания, где в конкурентной борьбе выигрывает нейрон-победитель. В итоге вектор R слоя распознавания содержит ровно одну единичную компоненту, а остальные равны нулю. Ненулевой выход нейрона-победителя устанавливает в ноль нейрон управления G1. По обратной связи нейрон-победитель посылает в слой сравнения сигналы и начинается фаза сравнения.

Фаза сравнения. Вектор отклика сравнивается с входным вектором X и выход C слоя сравнения теперь содержит единичные значения только в тех позициях, в которых единицы имеются и вектора X и у вектора обратной связи P. Если уровень сходства будет меньше заданного, то вырабатывается сигнал сброса, нейрон-победитель исключается из дальнейшей конкуренции и начинается фаза поиска. Если уровень сходства достаточный, то нейрон сброса остается неактивным. В этом случае вектор C вновь возбуждает нейрон-победитель.

Фаза поиска. После сигнала сброса все нейроны слоя распознавания получают нулевые выходы и нейрон управления G1=1. Снова вектор C полностью идентичен вектору X как в начале работы сети. Однако предыдущий нейрон-победитель исключается из конкурентной борьбы. После чего будет найден новый нейрон-победитель и повторяется фаза сравнения. Этот процесс завершается одним из двух способов:

1. Найден запомненный класс, сходство которого достаточно для успешной классификации и затем происходит обучение, в котором модифицируются веса bi и ti векторов B и T возбужденного нейрона;
2. Все классы проверены и ни один из них не дал требуемого сходства. В этом случае образ X объявляется новым в сети и ему из резерва назначается новый нейрон. Весовые векторы этого нейрона B и T устанавливаются равными вектору X.

Фаза поиска нужна затем, что обучение и функционирование сети выполняется одновременно. Нейрон-победитель определяет в пространстве входных векторов ближайший к заданному входному образу вектор памяти, и если все черты исходного вектора были критическими, это и была бы верная классификация.

После относительной стабилизации процесса обучения классификация выполняется без фазы поиска.

6.4. Обучение сети ART -1

В начале функционирования все веса **B** и **T** и параметр сходства принимают начальные значения

bij < L / (L-1+m);

ti=1 для всех i,j,

где bij –вес связи, соединяющий i-ый нейрон в слое сравнения с j-ым нейроном в слое распознавания;

m – число компонент входного вектора X; L – константа (например равна 2 при L>1,).

Такой выбор весов приводит к устойчивому обучению. Начальная установка весов bij в малые величины гарантирует, что несвязанные нейроны не будут получать больше возбуждения, чем обученные нейроны в слое распознавания. Параметр сходства выбирается на основе требований решаемой задачи. При высоких значениях **ρ** формируется большое число классов, к каждому из которых относятся только очень похожие векторы. При низком уровне **ρ** сеть формирует небольшое число классов с высокой степенью обобщения (*«грубая»* классификация). Для повышения гибкости сети необходима динамическое изменение параметра **ρ** во время процедуры обучения.

Обучение происходит без учителя и проводится для весов нейрона-победителя в случае успешной, так и неуспешной классификации. При этом веса вектора **B** стремятся к нормализованной величине компонент вектора **C**.

$$bij = \frac{L * ci}{L - 1 + \sum_{k} ck},$$

bij – вес связи, соединяющий i-ый нейрон в слое сравнения с j-ым нейроном в слое распознавания;

ci – i-ая компонента выходного вектора C слоя сравнения;

j – номер выигравшего нейрона в слое распознавания;

L – константа.

При этом роль нормализации компонент очень важна. Вектора с большим количеством единиц приводят к небольшим значениям b, и наоборот.

Таким образом, произведение (b*c) =∑bi*ci оказывается масштабированным и сеть может правильно различать вектора, даже если один из них является подмножеством другого.

Пусть вектор X1=100000, а вектор X2=111100. Эти образы различные. При обучении без нормализации (bi →ci) при поступлении в сеть первого образа он даст одинаковые скалярные произведения, равные единице, как с весами вектора X1, так и весами вектора X2. Вектор X2 при наличии шума может выиграть конкуренцию. При этом веса его вектора T установятся равными (100000), и образ (111100) будет «забыт». В случае применения нормализации исходные скалярные произведения будут равны 1 для вектора **X** и 2/5 для вектора **X2** (при L=2). Здесь вектор X1 выиграет конкуренцию.

Компоненты вектора T устанавливаются равными компонентам вектора C. tij=cij, где tij – вес связи между выигравшем нейроном j в слое распознавания и нейроном i в слое сравнения.

Если какая-то tj равна 0, то при дальнейшем обучении на фазах сравнения соответствующая компонента cj никогда не получит подкрепление по правилу 2 из 3.

Таким образом, нейронные сети АРТ решают дилемму «стабильности-пластичности» и могут быть весьма эффективными при реализации практических задач.

Разработанная для учебного процесса программная модель АРТ-сети [16] позволяет исследовать функционирование нейронной сети при различных показателях схожести и зашумлённости образов.

7. РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

7.1. Основные типы рекуррентных нейронных сетей

Отдельную группу нейронных сетей составляют сети с обратной связью между различными слоями нейронов. Это так называемые рекуррентные сети [1-9]. Их общая черта состоит в передаче сигналов с выходного либо скрытого слоя во входной слой, то есть рекуррентные сети характеризуются прямым и обратным распространением информации между слоями нейронной сети.

Главная особенность, выделяющая эти сети среди других нейронных сетей, – динамические зависимости на каждом этапе функционирования. Изменение состояния одного нейрона отражается на всей сети вследствие обратной связи типа "один ко многим". В сети возникает некоторый переходный процесс, который завершается формированием нового устойчивого состояния, отличающегося в общем случае от предыдущего.

Рекуррентные НС характеризуются прямым и обратным распространением информации между слоями нейронной сети. Обратная связь может присутствовать в нейронных сетях в виде локальной обратной связи (то есть на уровне одного нейрона) или глобальной ОС на уровне всей НС. На практике в основном используются два класса рекуррентных сетей:

– ассоциативная память[1-5];

– сети отображения вход-выход (RMLP, RTRN, НС Элмана)[3].

Подклассом рекуррентных НС являются релаксационные НС (РНС). В основе функционирования таких сетей лежит итеративный принцип работы. На каждой итерации процесса происходит обработка данных,

полученных на предыдущем шаге. Такая циркуляция информации продолжается до тех пор, пока не установится состояние равновесия (релаксация).

7.2. Модели релаксационных нейронных сетей

Благодаря обратной связи при подаче сигнала на входы сети, в ней возникает переходный процесс, который завершается формированием нового устойчивого состояния. В основе функционирования таких сетей лежит итеративный принцип работы. На каждой итерации процесса происходит обработка данных, полученных на предыдущем шаге. Такая циркуляция информации продолжается до тех пор, пока не установится состояние равновесия. При этом состояния нейронной сети перестают изменяться и характеризуются стационарными значениями.

Если функцию активации нейрона обозначить, где u – взвешенная сумма его возбуждений, то состояние нейрона можно определить выходным сигналом, где n – размерность входного вектора и количество нейронов в первом слое. Изменение состояния i-го нейрона можно описать системой дифференциальных уравнений для $i = \overline{1,n}$, где b_i – пороговое значение.

Рекуррентной сети можно поставить в соответствие энергетическую функцию Ляпунова:

$$E = -\frac{1}{2}\sum_{j}\sum_{i\neq j} w_{ij} y_i y_j + \sum_{i=1}^{n} \frac{1}{R_i} \int_0^x f_i^{-1}(y_i) dy_i + \sum_{i=1}^{n} b_i y_i \,. \qquad (7.1)$$

Изменение состояния какого-либо нейрона инициализирует изменение энергетического состояния (7.1) сети в направлении минимума ее энергии вплоть до его достижения. В пространстве состояний локальные энергетические минимумы E представлены точками

стабильности, называемыми аттракторами из-за тяготения к ним ближайшего окружения. Благодаря наличию аттракторов, рекуррентные сети могут быть использованы как устройства ассоциативной памяти.

Классическими примерами РНС являются нейронные сети Хопфилда [1-5], Хемминга [1-4], двунаправленная ассоциативная память (ДАП) [3,4,8,9] и машина Больцмана (МБ) [1,3-5].

Среди различных конфигураций искусственных нейронных сетей (НС) встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем [1], ни обучение без учителя [1]. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации можно расценивать, как помощь учителя, но с другой – сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с "миром" (учителем) не приходится. Подобная логика обучения характерна для релаксационных НС – сетей Хопфилда, Хемминга и двунаправленной ассоциативной памяти [1,3,4].

7.3. Нейронная сеть Хопфилда

В 1982 году американский биофизик Д. Хопфилд представил математический анализ релаксационных сетей с обратными связями. Поэтому такие НС получили название сетей Хопфилда. НС Хопфилда реализует существенное свойство авто ассоциативной памяти – восстановление по искаженному (зашумленному) образцу ближайшего к нему эталонного. В этом случае входной вектор используется как

начальное состояние сети, и далее сеть эволюционирует согласно своей динамике. Причем любой пример, находящийся в области притяжения хранимого образца, может быть использован как указатель для его восстановления. Выходной восстановленный образец формируется, когда сеть достигает равновесия.

Как видно на рисунке 1 структура сети Хопфилда представляется в виде системы с непосредственной обратной связью выхода со входом. Выходные сигналы нейронов являются одновременно входными сигналами сети: $x_i(k) = y_i(k-1)$. В классической сети Хопфилда отсутствует связь выхода нейрона с его собственным входом, что соответствует, а матрица весов является симметричной: $W = W^T$. Отсутствие авто связи и симметричность матрицы весов являются достаточными, но не необходимыми условиями сходимости переходных (итерационных) процессов в сети Хопфилда.

Наиболее часто используется в качестве функции активации используется биполярная ступенчатая функция активации со значениями, то есть выходной сигнал i-го нейрона определяется функцией:

$$y_i = \mathrm{sgn}(\sum_{j=0}^{n} w_{ij} x_j + b_i), \tag{7.2}$$

где sgn – функция определения знака.

Если порог срабатывания функции (7.2) является компонентой вектора X. Тогда основную зависимость, определяющую сеть Хопфилда, можно представить в виде:

$$y_i(k) = \mathrm{sgn}(\sum_{j=0}^{n} w_{ij} y_j(k-1)). \tag{7.3}$$

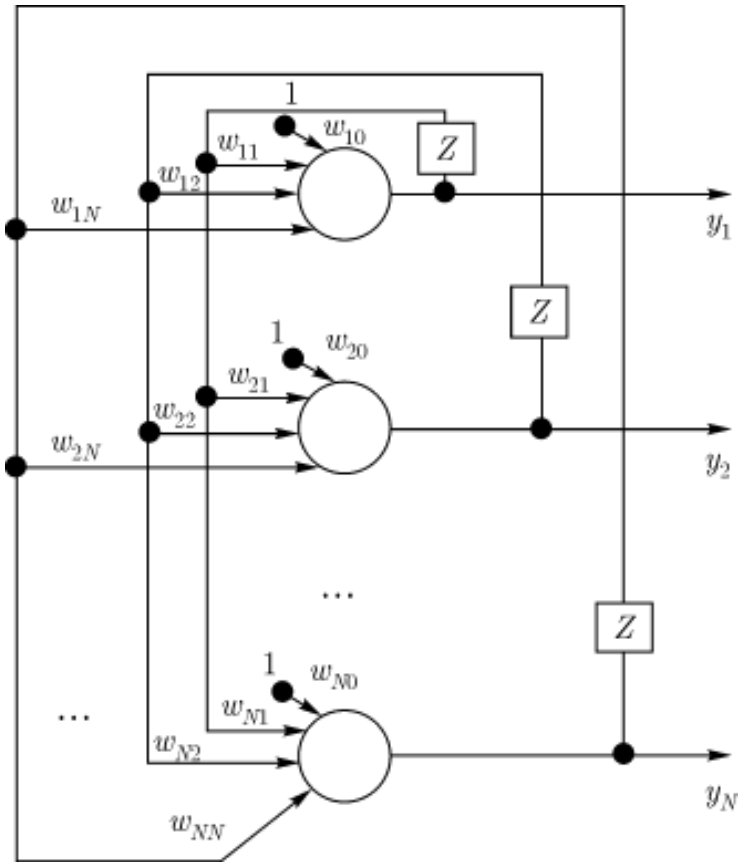


Рис.7.1. Структура нейронной сети Хопфилда

В процессе функционирования сети Хопфилда можно выделить два режима: обучения и классификации. В режиме обучения на основе известных векторов подбираются весовые коэффициенты сети. В режиме классификации при фиксированных значениях весов и вводе конкретного начального состояния нейронов возникает переходный процесс вида (7.3), завершающийся в одном из локальных минимумов, для которого.

Важным параметром ассоциативной памяти является ее емкость. Под емкостью понимается максимальное число запомненных образов, которые классифицируются с допустимой погрешностью. Показано, что при использовании для обучения правила Хебба и при $e_{\max}$ =0,01 (1% компонентов образа отличается от нормального состояния) максимальная емкость памяти составит всего лишь около 13,8% от количества нейронов, образующих ассоциативную память. Столь малая емкость обусловлена

тем, что сеть Хебба хорошо запоминает только взаимно ортогональные векторы или близкие к ним.

Три наиболее часто используемых метода обучения сети Хопфилда: правило Хебба, метод проекций, метод дельта проекций.

Для одного обучающего вектора значения весов могут быть вычислены по правилу Хебба: так как вследствие биполярных значений элементов вектора X, то есть всегда. При вводе большего количества обучающих векторов веса подбираются согласно обобщенному правилу Хебба:

$$w_{ij} = \frac{1}{n}\sum_{k=0}^{p} x_i^{(k)} x_j^{(k)} . \qquad (7.4)$$

Лучшие результаты, чем при использовании правила Хебба, можно получить, если для обучения использовать псевдо инверсию. В основе этого подхода лежит предположение, что при правильно подобранных весах каждый поданный на вход сети вектор вызывает генерацию самого себя на выходе сети. В матричной форме это можно представить в виде, где **W** матрица весов сети размерностью, а X – прямоугольная матрица размерностью $n \times p$, составленная из p обучающих векторов $X(k), k = \overline{1, p}$. Решение такой линейной системы уравнений имеет вид, где знак + обозначает псевдо инверсию. Если обучающие векторы линейно независимы, последнее выражение можно упростить и представить в виде:

$$W = X(X^T X)^{-1} X^T . \qquad (7.5)$$

В выражении (7.4) псевдо инверсия заменена обычной инверсией квадратной матрицы $X^T X$ размерностью. Выражение (7.4) можно записать в итерационной форме, не требующей расчета обратной матрицы. В этом случае (7.4) принимает вид итерационной зависимости от последовательности обучающих векторов $X(k), k = \overline{1, p}$:

$$y^{(k)} = (W^{(k-1)} - E)x^{(k)}, \qquad (7.6)$$

при начальных условиях $W^{(0)} = 0$. В результате предъявления p векторов матрица весов сети принимает значение. Такое обучение увеличивает максимальную емкость сети Хопфилда до. Увеличение емкости обусловлено тем, что в методе проекций требование ортогональности векторов заменено гораздо менее жестким требованием их линейной независимости.

Модифицированный вариант метода проекций – метод Δ-проекций представляет из себя градиентную форму алгоритма минимизации. В соответствии с этим методом веса подбираются с помощью процедуры, многократно повторяемой на всем множестве обучающих векторов:

$$W = W + \frac{h}{n}(x^{(k)} - Wx^{(k)})(x^{(k)})^T. \qquad (7.7)$$

Процесс (7.7) повторяется многократно по всем векторам вплоть до стабилизации значений весов.

После обучения сети Хопфилда по одному из алгоритмов (7.5), (7.6) или (7.7), сеть способна распознавать вектора, подаваемые на её вход. Алгоритм функционирования обученной сети состоит из трех следующих этапов.

На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов: (в данном случае индекс в скобках указывает номер итерации), поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер.

Второй этап работы заключается в смене состояния сети: рассчитывается новое состояние нейронов $s_j^{(m+1)} = \sum_{i=1}^{n} w_{ij} y_i^{(m)}\ j = \overline{1,n}$ и новые значения аксонов.

На третьем этапе выполняется проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы стабилизировались) – конец. При этом выходной вектор представляет собой образец, наилучшим образом (в смысле сети Хопфилда) сочетающийся с входными данными.

Недостатком классического варианта сети Хопфилда является тенденция к стабилизации в точках локального, а не глобального минимума энергии сети E. Одним из вариантов устранения этого недостатка является применение стохастических методов задания состояний нейронов.

7.4. Машина Больцмана

При решении технических и экономических задач неизвестна даже приблизительная оценка глобального экстремума. Это обуславливает применение методов глобальной оптимизации. НС Хопфилда находит локальный минимум задачи оптимизации. Для устранения этого недостатка можно использовать машину Больцмана, которая является расширением сети Хопфилда. В основе сети Больцмана лежит метод имитационного отжига (управляемого охлаждения), который является разновидностью процедуры случайного поиска. В своей базовой форме машина Больцмана является сетью Хопфилда и их структуры полностью совпадают. Метод имитации отжига представляет собой алгоритмический аналог физического процесса управляемого охлаждения. Он был предложен Метрополисом в 1953 году. Данный метод позволяет находить глобальный минимум функции нескольких переменных.

При отвердевании расплавленного металла его температура должна уменьшаться постепенно до момента полной кристаллизации. Если процесс остывания протекает слишком быстро, то образуются

нерегулярности структуры металла, которые вызывают внутренние напряжения. В результате общее энергетическое состояние тела, зависящее от внутренней напряженности, остается более высоким, чем при медленном охлаждении. Быстрая фиксация энергетического состояния тела на уровне выше нормального соответствует сходимости оптимизационного алгоритма к точке локального минимума. Энергия состояния тела соответствует целевой функции, а абсолютный минимум – точке глобального минимума. Метод имитации отжига представляет собой алгоритмический аналог физического процесса управляемого охлаждения. Это метод позволяет находить глобальный минимум функции нескольких переменных.

Алгоритм обучения Больцмана:

1. Определить переменную T, представляющую искусственную температуру. Придать T большое начальное значение.

2. Предъявить сети множество входов и вычислить выходы и целевую функцию.

3. Придать случайное изменение весу и пересчитать выход сети и изменение целевой функции в соответствии со сделанным изменением веса.

4. Если целевая функция уменьшилась (улучшилась), то сохранить изменение веса. Если изменение веса приводит к увеличению целевой функции, то вероятность сохранения этого изменения вычисляется с помощью распределения Больцмана:

$$P(c) = e^{\frac{-c}{kT}} \quad (8)$$

где P(c) — вероятность изменения c в целевой функции; k — константа, аналогичная константе Больцмана, выбираемая в зависимости от задачи; T — искусственная температура.

Выбирается случайное число r из равномерного распределения от нуля до единицы. Если, то изменение сохраняется, в противном случае

величина веса возвращается к предыдущему значению. Это позволяет системе делать случайный шаг в направлении, увеличивающем целевую функцию, и дает ей тем самым возможность вырываться из локальных минимумов, где любой малый шаг увеличивает целевую функцию.

Для завершения обучения машины Больцмана повторяются шаги 3 и 4 для каждого из весов сети, с постепенным уменьшением температуры T, пока не будет достигнуто допустимо низкое значение целевой функции. В этот момент предъявляется другой входной вектор, и процесс обучения повторяется. Сеть обучается на всех векторах обучающего множества, с возможным повторением, пока целевая функция не станет допустимой для всех обучающих векторов.

Величина случайного изменения веса на шаге 3 может определяться различными способами. Например, подобно тепловой системе, весовое изменение может выбираться в соответствии с гауссовским распределением: $P(w') = e^{\frac{-(w')^2}{T^2}}$ где — вероятность изменения веса на величину.

Машина Больцмана, учась на высокой температуре, ведет себя как случайная модель, а на низких температурах проявляет себя как детерминированная. Из-за случайной компоненты в процессе обучения, нейрон может принять новое значение состояния, которое увеличивается быстрее, чем уменьшается общее пространство состояний. Имитация физического отжига позволяет продвигаться к глобальному минимуму, избегая локальный. Для достижения сходимости к глобальному минимуму энергии скорость уменьшения температуры должна быть обратно пропорциональна логарифму времени.

Как и в сети Хопфилда, сети может быть представлен частичный образ для восстановления отсутствующей информации. Ограничение на

число распознаваемых образов оценивается, как и в сети Хопфилда классов – менее 15 % от общего количества элементов в слое.

7.5. Нейронная сеть Хемминга

Когда нет необходимости, чтобы сеть в явном виде выдавала образец, то есть достаточно, скажем, получать номер образца, ассоциативную память успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений.

Достоинством сети Хемминга считается небольшое количество взвешенных связей между нейронами. Многочисленные эксперименты доказали, что сеть Хемминга дает лучшие результаты, чем сеть Хопфилда. Единственная проблема, связанная с сетью Хемминга, проявляется в случае, когда зашумленные образы находятся на одинаковом (в смысле Хемминга) расстоянии от двух или более эталонов. В этом случае выбор сетью Хемминга одного из эталонов становится случайным.

Как видно на рисунке 2 сеть Хемминга включает в себя два слоя. Первый слой имеет однонаправленное распространение сигналов от входа к выходу и фиксированные значения весов. Второй слой состоит из нейронов, связанных обратными связями по принципу "каждый с каждым", при этом в каждом нейроне слоя существует авто связь (связь входа нейрона со своим собственным выходом). Количество нейронов в каждом слое сети равно количеству запоминаемых векторов.

Разные нейроны во втором слое связаны отрицательной (тормозящей) обратной связью с весом, при этом величина обычно обратно пропорциональна количеству образов. С собственным входом нейрон связан положительной (возбуждающей) обратной связью с весом, равным

+1. Пороговые веса нейронов приняты равными нулю. Нейроны этого слоя функционируют в режиме WTA (англ.: Winner Takes All - "победитель получает все"), при котором в каждой фиксированной ситуации активизируется только один нейрон, а остальные пребывают в состоянии покоя [1,3,4].

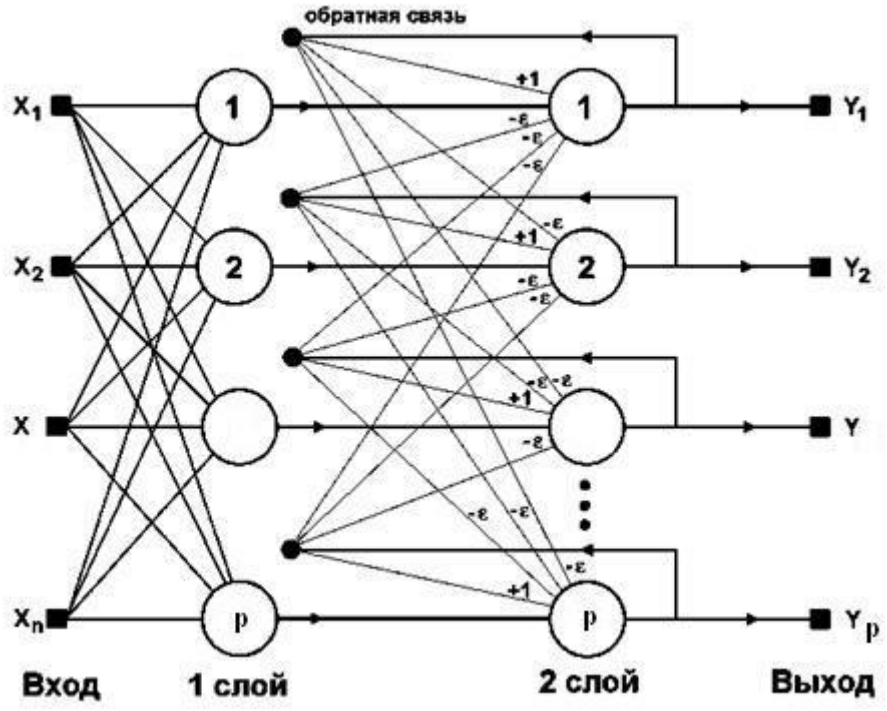


Рис.7.2. Структура нейронной сети Хемминга

Обучение сети Хемминга заключается в предварительном расчете значений весовых коэффициентов и порогов активации нейронов по следующим правилам. Весам первого слоя присваиваются значения, рассчитываемые по выражению:

$$w^1{}_{ik} = \frac{x_i^{\,k}}{2}, i = \overline{1,n}, k = \overline{1,p} \ , \qquad (7.9)$$

при этом пороги активационных функций равны:

$$T_k = \frac{n}{2}, k = \overline{1, p}\,. \tag{7.10}$$

Как отмечалось ранее, веса связи i-го и j-го нейронов второго слоя равны:

$$w^2_{ij} = \begin{cases} -\varepsilon, & i \neq j \\ 1, & i = j \end{cases}, \quad 0 < \varepsilon < \frac{1}{p}\,. \tag{7.11}$$

После обучения сети по формулам (7.9-7.11), сеть способна распознавать вектора, подаваемые на её вход. На входы сети подается неизвестный вектор X, исходя из которого, рассчитываются состояния нейронов первого слоя. Значения выходных сигналов нейронов определяются по формуле:

$$y_j{}^1 = \sum_{i=1}^{n} w_{ji} x_i + T_j,\ j = \overline{1, m}\,. \tag{7.12}$$

Эти сигналы становятся начальными состояниями нейронов второго слоя. Этот слой определяет "победителя", то есть нейрон, выходной сигнал которого близок к 1. Такой нейрон указывает на вектор образа с минимальным расстоянием Хемминга до входного вектора X. Нейрон-победитель определяется итерационным процессом расчета состояний нейронов второго слоя за счет ослабления весами входных сигналов слоя:

$$y^2{}_j(z+1) = f(y^2{}_j(z) - \varepsilon \sum_{k=1}^{p} y^2{}_k(z)), k \neq j,\ j = \overline{1, p}\,. \tag{7.13}$$

Итерационный процесс (13) во втором слое завершается, когда активным остается только один нейрон (его выход имеет значение близкое к 1), тогда как остальные нейроны пребывают в близком к нулю состоянии.

7.6. Двунаправленная ассоциативная память

Память человека часто является ассоциативной; один предмет напоминает нам о другом, а этот другой о третьем. Если позволить нашим мыслям, они будут перемещаться от предмета к предмету по цепочке умственных ассоциаций. Кроме того, возможно использование способности к ассоциациям для восстановления забытых образов. Рене рассмотренные модели ассоциативной памяти являются строго говоря, авто ассоциативными, это означает, что образ может быть завершен или исправлен, но не может быть ассоциирован с другим образом. Данный факт является результатом одноуровневой структуры ассоциативной памяти, в которой вектор появляется на выходе тех же нейронов, на которые поступает входной вектор.

Обобщением сети Хопфилда на случай двухслойной рекуррентной структуры, позволяющей кодировать множества пар взаимосвязанных векторов (X^k, Yk), считается двунаправленное ассоциативное запоминающее устройство, называемое Двунаправленной Ассоциативной Памятью (ДАП, англ.: Bidirectional Associative Memory (BAM)). В общем случае размерности n и m соответственно векторов X и Y не совпадают. В публикациях [8,9] представлено несколько форм реализации двунаправленной ассоциативной памяти: дискретная, непрерывная, адаптивная, и конкурирующая ДАП. Структура простейшей ДАП изображена на рис.7.3. Сигналы в такой сети распространяются в двух направлениях. В первом цикле сигналы вначале проходят в одну сторону для задания состояний нейронов-получателей, то в следующем цикле эти нейроны сами становятся источниками, высылающими сигналы в обратную сторону. При этом выбор начального направления распространения сигналов не регламентирован и может произвольно

выбираться пользователем. Процесс повторяется до достижения состояния равновесия.

Функция активации нейронов имеет пороговый характер. Для обеспечения лучших характеристик сети на этапе обучения используются только биполярные сигналы, в таком случае компоненты векторов (X^k, Yk) могут принимать только значения,

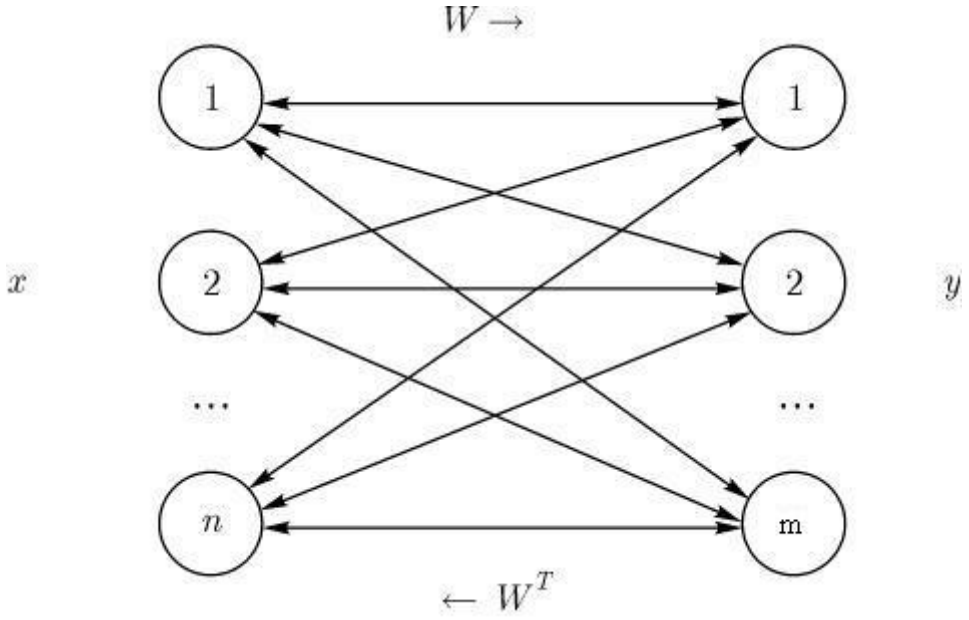


Рис. 7.3. Структура нейронной сети ДАП

Обучение ДАП заключается в предварительном расчете матрица весов W. Входные обучающие данные представляют собой множество пар биполярных векторов: $\left\{(X^k = (x_1^k, x_2^k, ...x_n^k), Y^k = (y_1^k, y_2^k, ..., y_m^k)), k = \overline{1, p}\right\}$.

На основе этого множества формируется матрица:

$$W = \sum_{k=1}^{p} (X^k)^T * Y^k . \qquad (7.14)$$

Рассчитанная по формуле (7.14) матрица весов W, связывающая обе части сети, является действительной и в общем случае несимметричной.

При прямом распространении сигналов веса описываются матрицей W, а при обратном — матрицей W^T.

Если принять за основное направление распространения сигнала направление $X \to Y$, то процесс функционирования сети выглядит следующим образом. На вход сети подают вектор X(0). Он обрабатывается матрицей весов W сети, в результате чего вырабатывается вектор выходных сигналов нейронов Y(1). Вектор Y(1) затем обрабатывается транспонированной матрицей W^T весов, которая вырабатывает новые выходные сигналы, представляющие собой новый входной вектор X(1). Процесс повторяется до тех пор, пока сеть не достигнет стабильного состояния, в котором ни вектор X(f), ни вектор Y(f) не изменяются. Заметим, что нейроны в слоях 1 и 2 функционируют, как и в других парадигмах, вычисляя сумму взвешенных входов и вычисляя по ней значение функции активации F. Этот процесс в векторной форме может быть выражен следующим образом:

$$\begin{aligned} &i = 0 \\ &Y(i+1) = F(X(i)W) \\ &X(i+1) = F(Y(i+1)W^T) \end{aligned} \quad , \qquad (7.15)$$

где i – номер итерации.

В результате процесса (7.15) двунаправленной обработки сигналов формируются два стабильных вектора X(f) и Y(f), удовлетворяющих уравнениям: и.

В режиме распознавания при начальных значениях векторов, совпадающих с использованными векторами при обучении, сеть распознает их безошибочно. При искажении векторов и сеть ВАМ не всегда способна откорректировать эти векторы и распознает их с определенными погрешностями.

Как и сети Хопфилда, ДАП имеет ограничения на максимальное количество ассоциаций, которые она может точно воспроизвести. Если этот лимит превышен, сеть может выработать неверный выходной сигнал,

воспроизводя ассоциации, которым не обучена. В работе [8] приведены оценки, в соответствии с которыми количество запомненных ассоциаций не может превышать количества нейронов в меньшем слое. При этом предполагается, что емкость памяти максимизирована посредством специального кодирования, при котором количество компонент со значениями +1 равно количеству компонент со значениями –1 в каждом биполярном векторе. На практике не редко используют ещё более осторожную оценку емкости памяти: если размерности векторов X и Y обозначить соответственно n и, m то удовлетворительное качество распознавания можно получить при выполнении зависимости, где – число запоминаемых в сети пар векторов.

Ограничение количества единиц во входных векторах представляет серьезную проблему, тем более, что теория, которая позволяет перекодировать произвольный набор векторов в такой “разреженный” набор, отсутствует. Возможно, однако, что еще более серьезной является проблема некорректной сходимости. Суть этой проблемы заключается в том, что сеть может не производить точных ассоциаций вследствие природы поля притяжения; об ее форме известно очень немногое. Это означает, что ДАП не является ассоциатором по отношению к ближайшему соседнему образу. В действительности она может производить ассоциации, имеющие слабое отношение ко входному вектору. Как и в случае гомогенных ДАП, могут встречаться ложные стабильные состояния и немногое известно об их количестве и природе. Несмотря на эти проблемы, ДАП остается объектом интенсивных исследований. Основная привлекательность ДАП заключается в ее простоте.

7.7. Многопрофильная модель релаксационной нейронной сети

Многопрофильная модель релаксационной нейронной сети [8] является лабораторной установкой (ЛУ), предназначенной для создания релаксационных сетей Хопфилда, Хемминга, Двунаправленной ассоциативной памяти и Машины Больцмана с варьируемыми параметрами, создания и редактирования обучающих выборок, обучения сетей различными методами, тестирования сетей.

По желанию пользователя могут устанавливаться следующие параметры релаксационной сети: метод обучения, вид активационной функции, коэффициент обратной связи, начальная температура и скорость охлаждения (для Машины Больцмана).

Программа имеет два режима работы: стандартный режим и режим формирования отчета. Стандартный режим работы позволяет формировать и редактировать обучающие выборки, произвольно устанавливать тип и параметры, проводить обучение и тестирование сети. Режим формирования отчета предполагает наличие заранее созданной обучающей выборки и позволяет протестировать все типы сетей с вариацией всех параметров и всеми возможными видами модификации эталонных данных обучающей выборки.

Работа с ЛУ начинается с окна выбора одного из двух режимов работы: обычный режим или режим формирования отчета. В обычном режиме ЛУ позволяет создавать и модифицировать обучающие выборки, произвольно выставлять тип, параметры РНС, выполнять их обучение и тестирование. Режим формирования отчета предназначен для выполнения лабораторной работы.

Главное окно программы моделирования содержит пять рабочих областей (рис.7.4):

– строка меню;

– панель быстрого доступа;

– область отображения обучающей выборки и результатов тестирования;

– область модификации и тестирования отдельных образов;

– строка состояния.

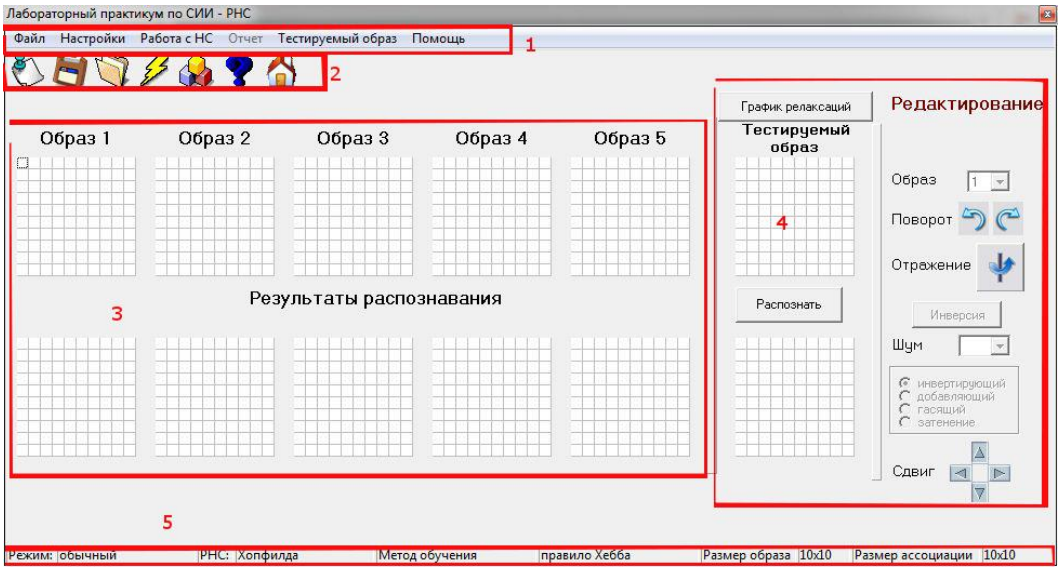


Рис.7.4. Главное окно программы моделирования

Строка меню предоставляет доступ к большинству функций ЛУ, для большинства пунктов меню предусмотрены комбинации горячих клавиш. Все функции сгруппированы в пяти разделах. Элемент меню "Файл" предоставляет доступ к стандартным функциям работы с файлами обучающих выборок.

Элемент "Настройки" осуществляет вызов окна настроек, в которых можно задать параметры обучающих выборок и моделируемых сетей. Элемент меню "Тестируемый образ" предоставляет доступ к функциям модификации тестовых образов и тестирования сети с использованием модифицированных образов.

8. ПРИМЕРЫ МОДЕЛИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ В СРЕДЕ MATLAB

8.1. Моделирование многослойных нейронных сетей для аппроксимации функций sin(x) и cos(x)

Листинг программы моделирования многослойной нейронной сети в среде MATLAB[5] приведён ниже.

```
%Задание интервала значений для функций sin(x) и cos(x).
X = [-3.14*5:0. 01:3.14*5];
%Формирование значений функций в этом интервале
Ys = sin(X);
Yc = cos(X);
%Создание многослойных нейронных сетей для аппроксимации
nets = newff(minmax(X),[13,1,1],{'tansig','tansig','tansig'},'traingd');
netc = newff(minmax(X),[13,3,1],{'tansig','tansig','tansig'},'traingd');
%Тренировка сети и получение результатов работы сети
nets.trainParam.epochs = 1000;
netc.trainParam.epochs = 1000;
[nets,trs] = train(nets,X,Ys);
[netc,trc] = train(netc,X,Yc);
Yns=sim(nets,X);
Ync=sim(netc,X);
%Построение графика функций sin(x)
```

[5] Справочник по Matlab http://radiomaster.ru/cad/matlab/index.php

```
figure('NumberTitle','off','Name','Функция
Sin(x)','ToolBar','none','MenuBar','none');
plot(X,Ys,'-k','LineWidth',1);hold on; plot(X,Yns,'--k','LineWidth',2);hold
on;
Xin=[-8.53 -7.34 -6.21 -5.15 -4.91 -3.82 -2.76 -1.63 -0.53 0.45 1.34 2.22
...
   3.15 4.01 5.95 6.85 7.77 8.66];
Yout=sim(nets,Xin);
plot(Xin,Yout,'ok','MarkerSize',7,'LineWidth',1);
%Построение графика функций cos(x)
figure('NumberTitle','off','Name','Функция
Cos(x)','ToolBar','none','MenuBar','none');
plot(X,Yc,'-k','LineWidth',1);hold on; plot(X,Ync,'--k','LineWidth',2);hold
on;
Yout=sim(netc,Xin);
plot(Xin,Yout,'ok','MarkerSize',7,'LineWidth',1);
```

Для обучения НС использовалось три функции:

– traingd – функция обучения НС с использованием алгоритма градиентного спуска GD.

– trainlm – функция обучения НС с использованием алгоритма Левенберга-Марквардта LM.

– trainbr – функция обучения НС с использованием алгоритма Левенберга-Марквардта, дополненная регуляцией по Байесу BR.

Для аппроксимации функций sin(x), cos(x), в зависимости от применяемой функции обучения НС, подобрано разное количество нейронов, приведённых в табл. 8.1.

Таблица 8.1

Функции обучения нейронной сети	Минимальное количество нейронов аппроксимации функций					
	sin(x)			cos(x)		
	1-й слой	2-ой слой	3-й слой	1-й слой	2-ой слой	3-й слой
traingd	13	1	1	13	3	1
trainlm	7	1	1	7	1	1
trainbr	7	1	1	7	1	1

При увеличении количества нейронов с функциями обучения trainlm и trainbr до такого же количества, как и с traingd, качество аппроксимации функций sin(x) и cos(x) улучшается.

При использовании функций обучения trainlm и trainbr результаты получаются более точные, чем при использовании функции traingd.

Наилучшей функцией обучения оказалась функция trainlm, так как её среднеквадратичная ошибка уменьшается быстрее, чем в traingd и trainbr.

При использовании trainlm (как и trainbr), требуется меньшее количество нейронов для аппроксимации функций sin(x) и cos(x).

Использование функций trainlm и trainbr уменьшает количество необходимых итераций (эпох) для обучения сети, что сокращает время обучения.

Оптимальная структура многослойной НС для функции sin(x) имеет структуру с 15 нейронами во входном слое, с 5 нейронами в скрытом слое и одним нейроном в выходном слое. Сокращённое обозначение структуры НС (15-5-1).

newff (minmax(X), [15, 5, 1], {'tansig','tansig','tansig'},'trainlm').

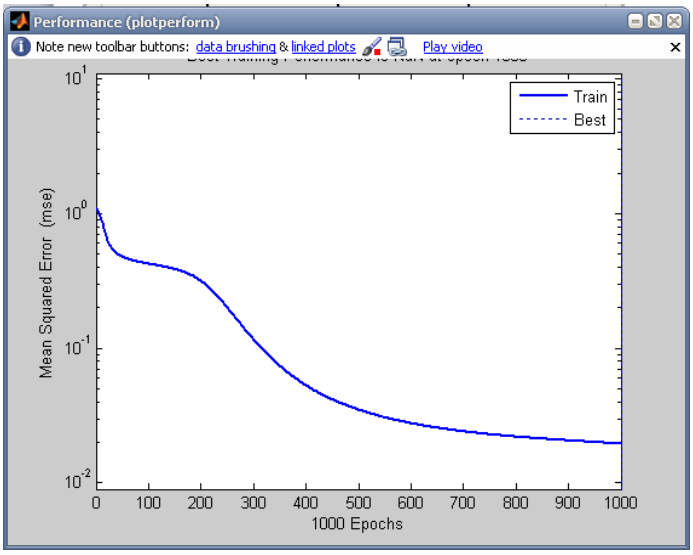


Рис.8.1. График зависимости среднеквадратичной ошибки от количества итераций (traingd 13-1-1)

Оптимальная структура многослойной НС для моделирования функции cos(x) включает 15 входных, 8 скрытых и один выходной нейрон (15-8-1).

newff (minmax(X), [15, 8, 1], {'tansig','tansig','tansig'},'trainlm').

На рис. 8.1 ниже приведён график зависимости среднеквадратичной ошибки от количества итераций (traingd 13-1-1) при аппроксимации функции sin(x).

Основная ошибка аппроксимации наблюдается для функции sin(x) в области от 0,8 до 1 и от минус 1 до минус 0,8. На рис. 8.2 ниже приведён график зависимости среднеквадратичной ошибки от количества итераций (trainbr 7-1-1) при аппроксимации функции sin(x), но при других параметрах архитектуры НС и обучения.

Наилучшие результаты аппроксимации функции sin(x) были получены с параметрами (trainlm 15-5-1). На рис. 8.3 представлена оптимальная архитектура НС.

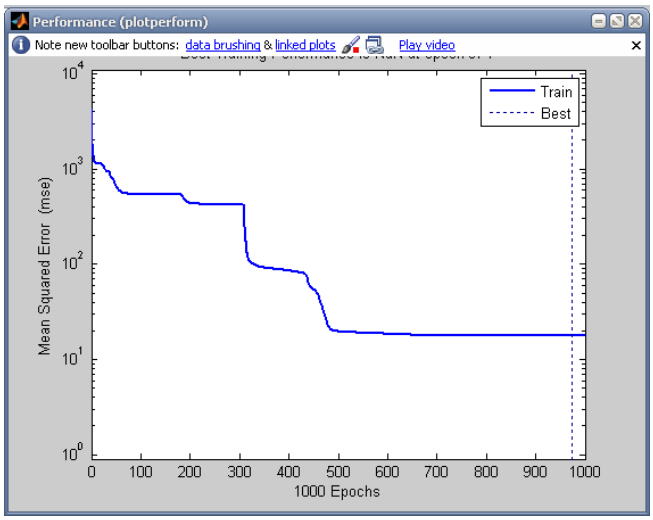


Рис. 8.2. График зависимости среднеквадратичной ошибки от количества итераций (trainbr 7-1-1)

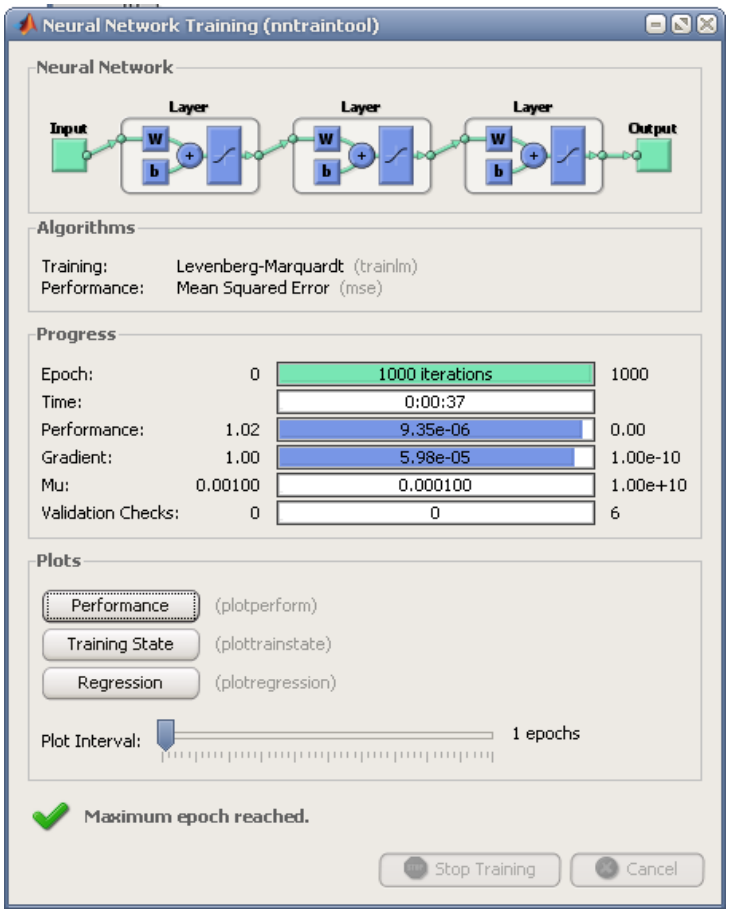


Рис.8.3 Нейронная сеть аппроксимации функции sin(x) (trainlm 15-5-1)

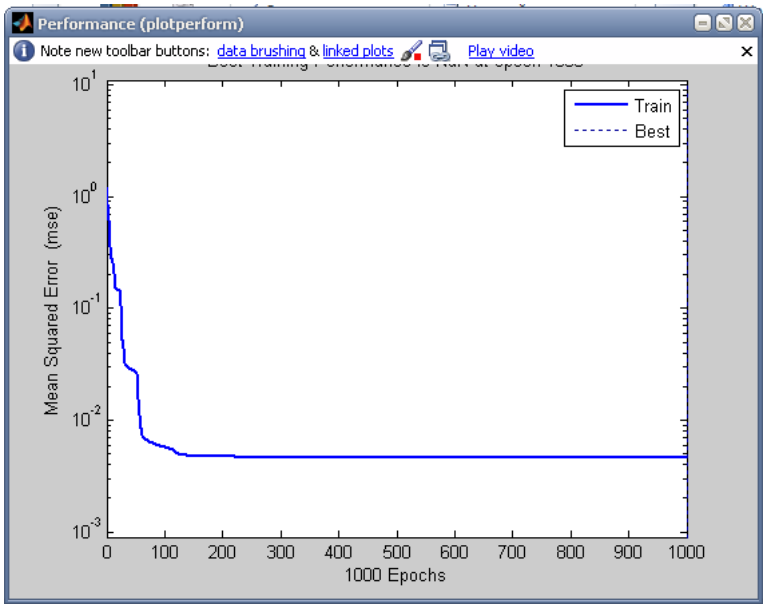


Рис. 8.4. График зависимости среднеквадратичной ошибки от количества итераций (trainlm 13-3-1)

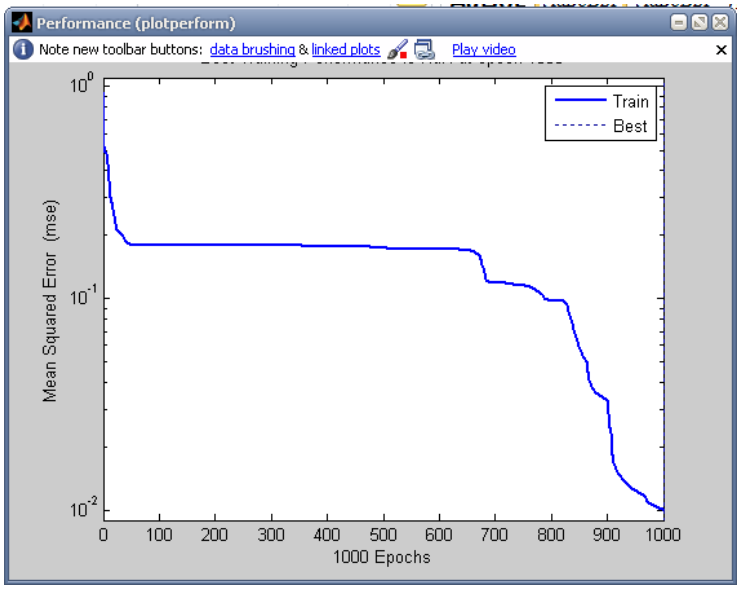


Рис.8.5. График зависимости среднеквадратичной ошибки от количества итераций (trainlm 7-1-1)

На рис. 8.4 ниже приведён график зависимости среднеквадратичной ошибки от количества итераций (traingd 13 1 1) при аппроксимации функции cos(x). Основная ошибка аппроксимации наблюдается для функции cos(x) в области от 0,8 до 1 и от минус 1 до минус 0,8. На рис. 8.5 ниже приведён график зависимости среднеквадратичной ошибки от количества итераций (trainbr 7-1-1) при аппроксимации функции cos(x), но

при других параметрах архитектуры НС и обучения. Наилучшие результаты аппроксимации функции sin(x) были получены с параметрами (trainlm 15-8-1).

8.2. Моделирование комбинационной логической схемы

При создании гибридных интеллектуальных систем могут быть использованы логические схемы, схемы резервирования и т.п.

Пример моделируемой схемы, реализующей логическое выражение вида ~ ((b & d) v (a & c)) &d = y, приведён на рис. 8.6.

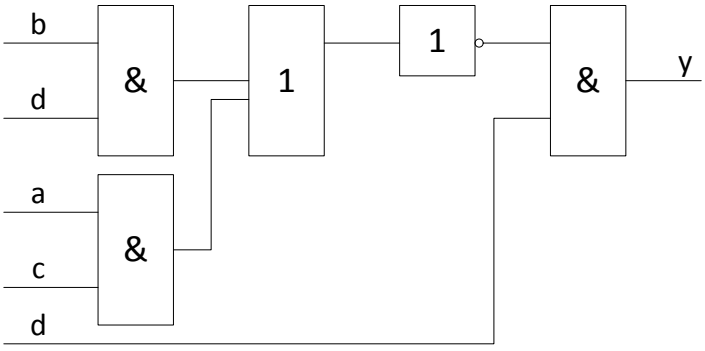


Рис. 8.6. Фрагмент моделируемой логической схемы

Листинг программы моделируемой схемы представлен ниже.

%Создание сети НС, состоящей из 1 нейрона и четырех входов

net=newp ([0 1; 0 1;0 1;0 1],1);

%Задание таблицы истинности входных и выходных данных

P={[0;0;0;0] [0;0;0;1] [0;0;1;0] [0;0;1;1] [0;1;0;0] [0;1;0;1] [0;1;1;0]

...

[0;1;1;1] [1;0;0;0] [1;0;0;1] [1;0;1;0] [1;0;1;1] [1;1;0;0] [1;1;0;1]...

[1;1;1;0] [1;1;1;1]};

T={0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0};

```
%Обучение сети
net.trainParam.epochs = 100;
Net = train (net, P, T);
%Проверка правильности полученной НС
Y = sin (net, P)
W = net.iw{1}
B = net.b{1}
```

При подаче тестового вектора обученная НС выдает ошибочные результаты. Для настройки сети необходимо провести несколько циклов адаптации. После каждого цикла адаптации НС проверяется на наличие ошибок. Если в результате работы НС присутствуют ошибки, производится следующий цикл адаптации и так до тех пор, пока ошибки не будут равны нулю.

```
net.adaptParam.passes=1;
[net,a,e] = adapt(net,P,T);
%Проверка ошибок
e
```

Результаты работы программы приведены на рис. 8.7 и рис.8.8.

```
e =
  Columns 1 through 9
    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]    [0]
  Columns 10 through 16
    [0]    [0]    [0]    [0]    [0]    [0]    [0]
Y =
  Columns 1 through 9
    [0]    [1]    [0]    [1]    [0]    [0]    [0]    [0]    [0]
  Columns 10 through 16
    [1]    [0]    [0]    [0]    [0]    [0]    [0]
```

W =

[-3,-5,-2,4]

B =

[-1]

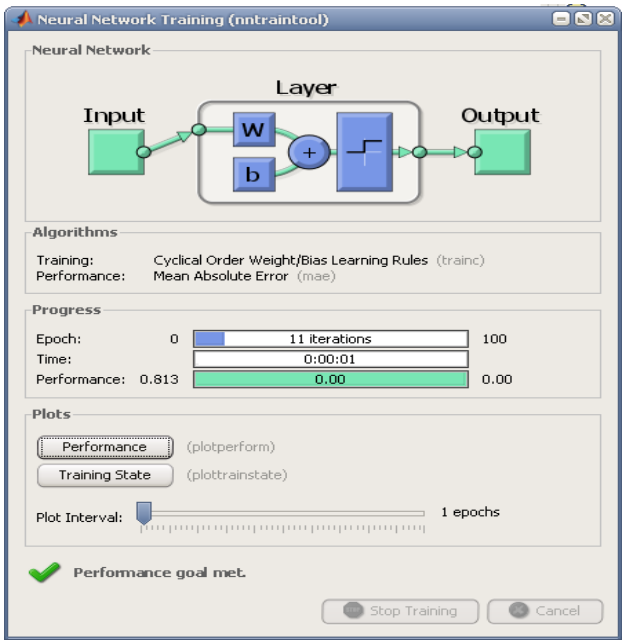


Рис. 8.7. Результаты обучения нейронной сети

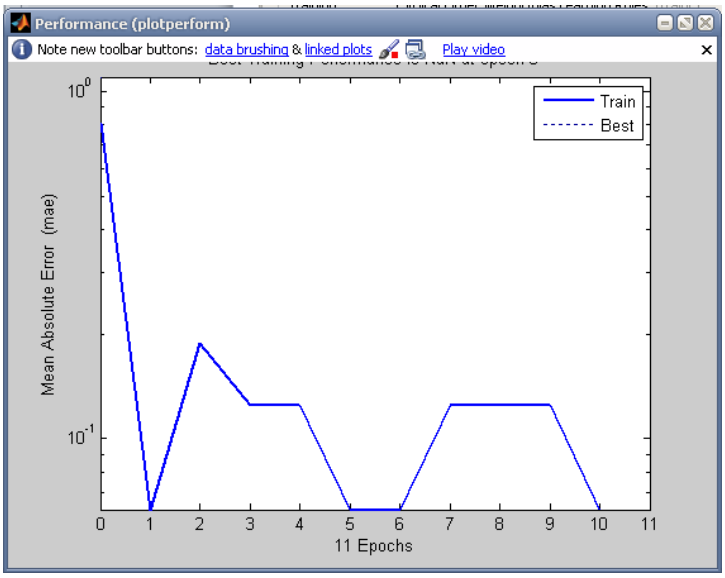


Рис. 8.8. График зависимости средней абсолютной ошибки обучения от количества итераций

Листинг программ реализующих элементарные логические функции (2И, 2ИЛИ, НЕ) приведён ниже.

Файл LR2_and

```
%создание и обучение персептрона, реализующего логическую
%функцию 2И, с пороговой активационной функцией,
net=newp([0 1; 0 1],1);
P={[0; 0] [0; 1] [1; 0] [1;1]};
T={0, 0, 0, 1};
net.trainParam.goal=0;
net.trainParam.epochs=20;
net=train(net, P, T);
Y=sim(net, P);
```

Файл LR2_or

```
%создание и обучение персептрона, реализующего логическую
%функцию 2ИЛИ, с пороговой активационной функцией
net = newp([0 1; 0 1],1);
P={[0; 0] [0; 1] [1; 0] [1;1]};
T = {0, 1, 1, 1};
net.trainParam.goal=0;
net.trainParam.epochs= 20;
net=train(net, P, T);
Y=sim(net, P);
```

Файл LR2_not

%создание и обучение персептрона, реализующего логическую %функцию НЕ, с пороговой активационной функцией

net = newp([0 1],1);

P = {0, 1};

T = {1, 0};

net.trainParam.goal=0;

net.trainParam.epochs= 20;

net=train(net, P, T);

Y=sim(net, P);

Анализ результатов аппроксимации позволил отобрать оптимальные параметры многослойной НС:

– для функции sin(x):

newff(minmax(X), [15, 5,1],{'tansig','tansig','tansig'},'trainlm');

– для функции cos(x):

newff(minmax(X),[15,8,1],{'tansig','tansig','tansig'},'trainlm').

При создании нейронной сети для аппроксимации функции на её конфигурацию влияют следующие факторы:

– Интервал, на котором аппроксимируется функция. Чем больше интервал входных значений, тем больше требуется нейронов для её аппроксимации.

– Вид аппроксимируемой функции влияет на выбор активационных функций нейронной сети каждого слоя.

– Количество нейронов и функции активации для каждого слоя НС подбираются экспериментально. Для аппроксимации тригонометрических функций sin(x), cos(x) выбрана гиперболическая тангенциальная функция активации tansig, которая позволяет наиболее точно аппроксимировать тригонометрические функции sin(x), cos(x).

При моделировании фрагмента комбинационной схемы использовался персептрон с четырьмя входами (a, b, c, d). Входы персептрона могут принимать значения от 0 до 1. Для обучения персептрона потребовалось 11 итераций (эпох). При обучении персептрона получены веса (-3,-5,-2,4) и смещение (-1).

8.3. Моделирование радиально-базисных нейронных сетей для аппроксимации функций sin(x) и cos(x)

Радиально-базисная нейронная сеть создается добавлением на каждом шаге нового нейрона в скрытом слое, пока допустимая среднеквадратическая ошибка не станет равна установленному значению или не будет достигнуто максимальное количество нейронов. Радиально-базисную сеть целесообразно применять, в случае достаточного количества векторов обучающего множества, а также в задачах, которые не требуют высокой точности и допускают снижение точности для ускорения выполнения задачи.

Листинг программы моделирования радиально-базисных нейронных сетей для аппроксимации функций sin(x) и cos(x) приведён ниже.

```
% Создание радиально-базисных нейронных сетей для аппроксимации % функций sin(x) и cos(x)
%Задание интервала, на котором исследуются целевые
 % функции sin(x) и cos(x)
X = [-pi: 0.1: pi];
%Получение значения функций
Ys = sin(X);
Yc = cos(X);
%Создание радиально-базисных нейронные сети
%GOAL=0, SPREAD=1
```

```
nets = newrb(X,Ys,0,1);
netc = newrb(X,Yc,0,1);
% Симуляция
Yns=sim(nets,X);
Ync=sim(netc,X);
%Построение графика аппроксимации
%Построение графика функции sin(x)
figure('NumberTitle','off','Name','Функция
Sin(x)','ToolBar','none','MenuBar','none');
plot(X,Ys,'-k','linewidth',1);hold on;
plot(X,Yns,'-ok','markersize',4,'LineWidth',1); hold on;
title('Sin(x)');
xlabel('X');
ylabel('Y');
Xin=[-pi:0.43:pi];
Yout=sim(nets,Xin);
plot(Xin,Yout,'sk','markersize',8,'LineWidth',2);
%Построение графика функции cos(x)
figure('NumberTitle','off','Name','Функция
Cos(x)','ToolBar','none','MenuBar','none');
plot(X,Yc,'-k','linewidth',1);hold on;
plot(X,Ync,'-ok','markersize',4,'LineWidth',1); hold on;
title('Cos(x)');
xlabel('X');
ylabel('Y');
Xin=[-pi:0.43:pi];
Yout=sim(netc,Xin);
plot(Xin,Yout,'sk','markersize',8,'LineWidth',2);
%Количество нейронов в скрытом слое нейронной сети
```

NumNeuronsSin=nets.layers{1}.size

NumNeuronsCos=netc.layers{1}.size

%Вычисление значения среднеквадратичной ошибки обучения

% нейросети

Es=Ys-Yns;

Es = mse(Es)

Ec=Yc-Ync;

Ec = mse(Ec)

NumNeuronsSin =63

NumNeuronsCos =63

Количество нейронов для аппроксимации функций sin(x), cos(x) оказалось сравнительно большим (63). Это связано с количеством обучающих векторов и значением параметра цели. В нашем случае количество векторов равнялось 63, а цель = 0. В табл.8.2 приведена зависимость количества нейронов, средней ошибки от параметра цели. Цель (GOAL) – допустимая среднеквадратичная ошибка сети.

Таблица 8.2. Зависимость количества нейронов и средней ошибки от значения цели (Goal)

Цель (Goal)	Количество нейронов		Среднеквадратичная ошибка	
	sin(x)	cos(x)	sin(x)	cos(x)
0	63	63	$2.2966*10^{-16}$	$8.8417*10^{-19}$
0.01	2	3	0.0045	$9.8484*10^{-4}$
0.0001	8	6	$6.1257*10^{-4}$	$2.4159*10^{-5}$
0.000001	12	10	$4.2085*10^{-7}$	$1.1312*10^{-7}$
0.00000001	15	12	$6.9561*10^{-9}$	$3.0663*10^{-9}$
0.0000000001	17	14	$3.2617*10^{-11}$	$1.1821*10^{-11}$

Анализ таблицы 8.2 показывает, что при уменьшении значения цели количество нейронов в скрытом слое увеличивается, и, следовательно,

увеличивается точность аппроксимации. Аппроксимация функции cos(x) в целом требует меньшее количество нейронов в скрытом слое и имеет меньшую среднеквадратичную ошибку обучения, по сравнению с аппроксимацией функции sin(x).

На рис. 8.9-8.12 показаны графики функций sin(x), cos(x) (сплошной линией), графики аппроксимаций этих функций через НС (маркеры (атрибут ‘o’) указывают на входные вектора обучения НС). Также квадратными маркерами (атрибут ‘s’) отмечены результаты работы НС при поступлении на вход тестового вектора.

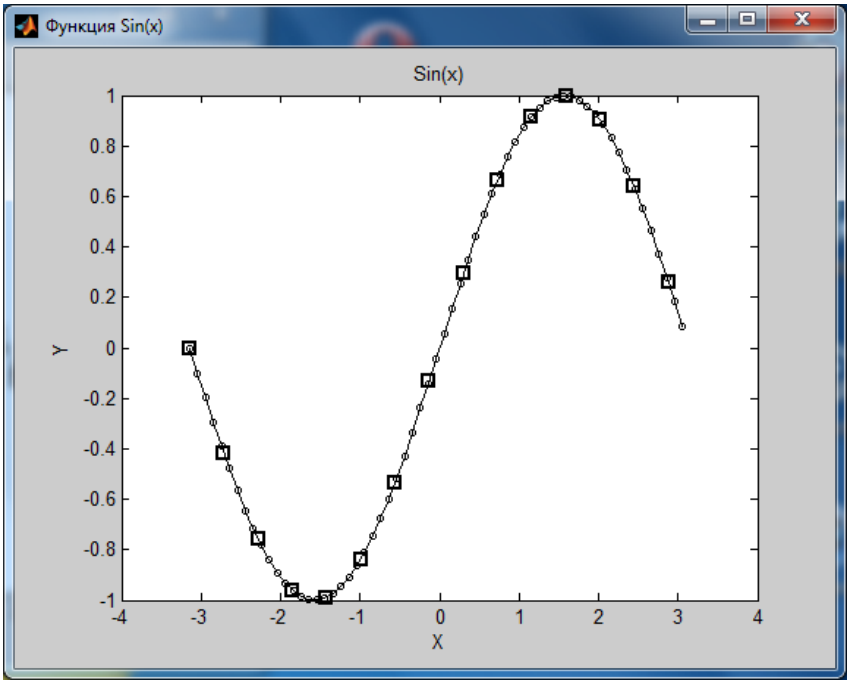


Рис. 8.9. График функции sin(x) и аппроксимации с помощью НС (newrb, goal=0,spread=1, numneurons=63)

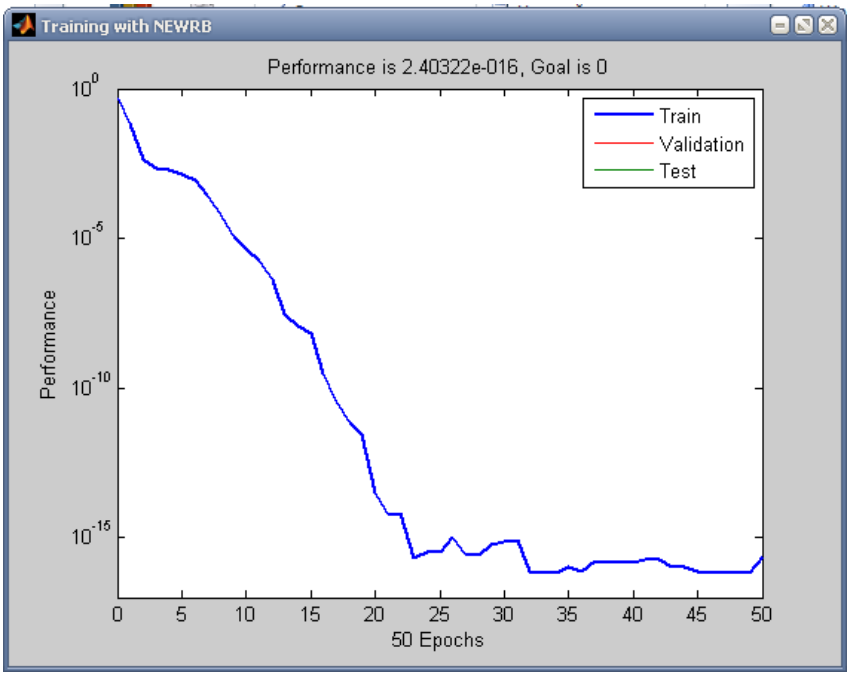


Рис. 8.10. График зависимости среднеквадратичной ошибки от количества итераций (sin(x), newrb, goal=0,spread=1, numneurons=63)

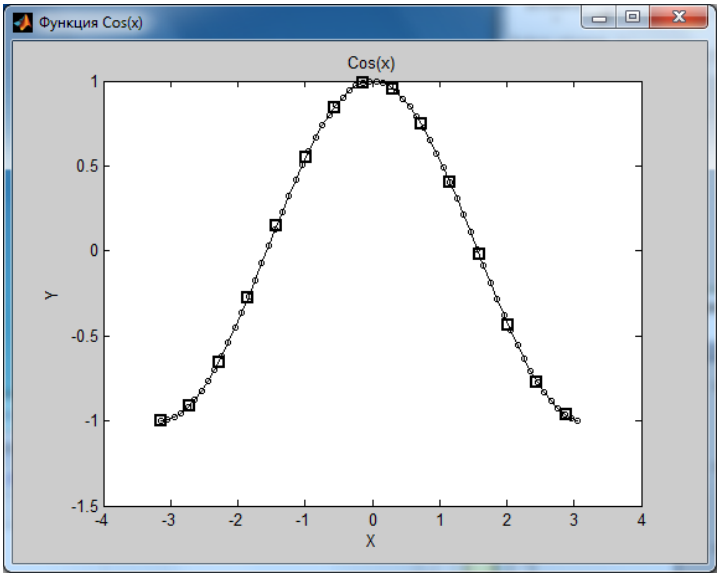


Рис. 8.11. График функции cos(x) и аппроксимации с помощью НС (newrb, goal=0,spread=1, numneurons=63)

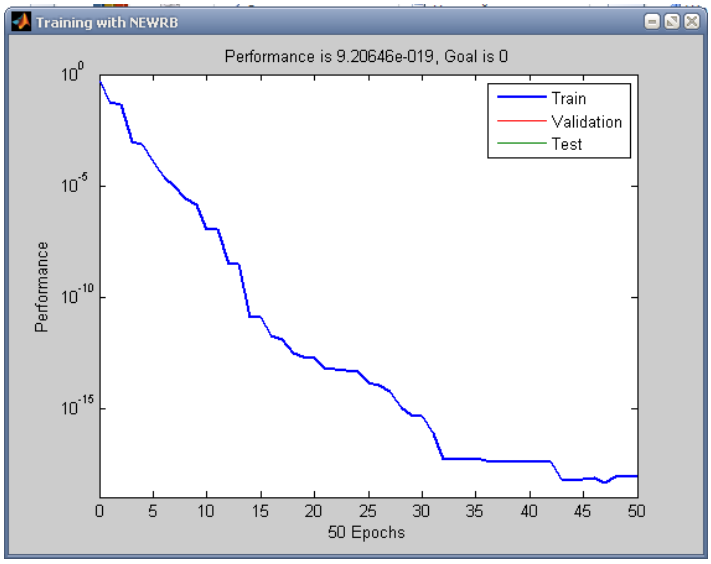


Рис. 8.12. График зависимости среднеквадратичной ошибки от количества итераций (cos(x), newrb, goal=0,spread=1, numneurons=63)

8.4. Моделирование обобщенных регрессионных нейронных сетей (GRNN) для аппроксимации функций sin(x) и cos(x)

Обобщённые регрессионные сети являются разновидностью радиальных базисных сетей и используются для анализа временных рядов, решения задач обобщенной регрессии. Характерной особенностью этих сетей является высокая скорость их обучения. Поэтому GRNN сети целесообразно применять в задачах, где требуется обеспечить минимальное время обучения сети с невысокими требованиями к точности.

Листинг программы моделирования обобщенных регрессионных нейронных сетей для аппроксимации функций sin(x) и cos(x) приведён ниже.

%Создание обобщенных регрессионных нейронных сетей для %аппроксимации функций sin(x) и cos(x)

%Задание интервала на котором моделируются целевые функции %sin(x) и cos(x)

```
X = [-pi:0.1:pi];
%Получение значения функций
Ys = sin(X);
Yc = cos(X);
%Создание обобщённых регрессионных НС
%SPREAD=1
nets = newgrnn(X,Ys,1);
netc = newgrnn(X,Yc,1);
Percentage Производится симуляция
Yns=sim(nets,X);
Ync=sim(netc,X);
%Построение графика аппроксимации
%Построение графика функции sin(x)
figure('NumberTitle','off','Name','Функция
Sin(x)','ToolBar','none','MenuBar','none');
plot(X,Ys,'-k','linewidth',3);hold on;
plot(X,Yns,'-ok','markersize',8,'LineWidth',1); hold on;
title('Sin(x)');
xlabel('X');
ylabel('Y');
Xin=[-pi:0.82:pi];
Yout=sim(nets,Xin);
plot(Xin,Yout,'sk','markersize',10,'LineWidth',2);
%Построение графика функции cos(x)
figure('NumberTitle','off','Name','Функция
Cos(x)','ToolBar','none','MenuBar','none');
plot(X,Yc,'-k','linewidth',3);hold on;
plot(X,Ync,'-ok','markersize',8,'LineWidth',1); hold on;
title('Cos(x)');
```

```
xlabel('X');
ylabel('Y');
Xin=[-pi:0.82:pi];
Yout=sim(netc,Xin);
plot(Xin,Yout,'sk','markersize',10,'LineWIdth',2);
%Количество нейронов в скрытом слое созданных НС
NumNeuronsSin=nets.layers{1}.size
NumNeuronsCos=netc.layers{1}.size
%Вычисление значения среднеквадратичной ошибки обучения НС
Es=Ys-Yns;
Es = mse(Es)
Ec=Yc-Ync;
Ec = mse(Ec)
NumNeuronsSin =63
NumNeuronsCos =63
Es = 0.0514
Ec = 0.0573
```

На рис. 8.13-8.14 показаны графики функций sin(x), cos(x) (черная сплошная линия), графики аппроксимации этих функций (маркеры (‘o’) указывают на входные вектора обучения НС).

Анализ рис. 8.13 и 8.14 демонстрирует низкую точность аппроксимации при большом (63) количестве нейронов. В обобщенной регрессионной нейронной сети количество нейронов равно количеству элементов входного вектора.

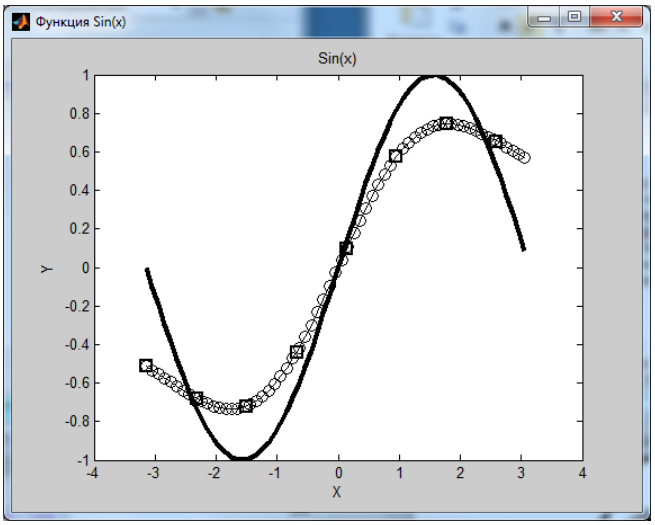


Рис. 8.13. График функции sin(x) и аппроксимации с помощью НС (newgrnn, spread=1, numneurons=63)

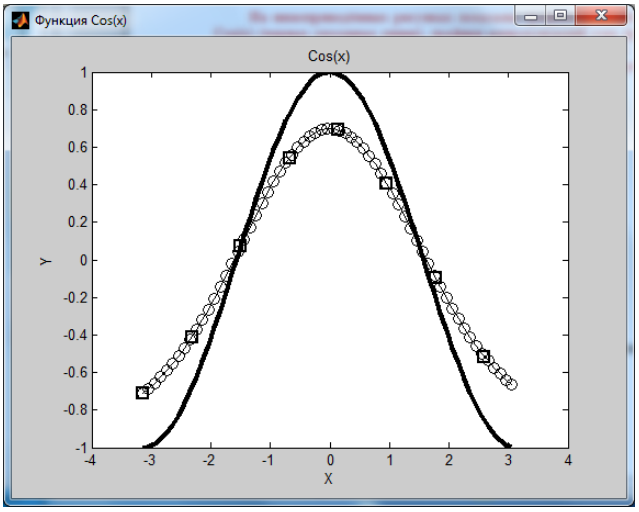


Рис. 8.14. График функции cos(x) и аппроксимации с помощью НС (newgrnn, spread=1, numneurons=63)

Точность аппроксимации зависит от значения параметра SPREAD. Чтобы выполнить точную аппроксимацию следует использовать значение параметра SPREAD меньшее, чем расстояние (шаг) между входными векторами. При шаге между входными векторами 0,1 выбирается значение параметра SPREAD равное 0,09.

В результате получены следующие значения:

NumNeuronsSin = 63

NumNeuronsCos = 63

Es =4.0157*10^{-5}

Ec =4.4414*10^{-6}

На рис. 8.15 и 8.16 представлены графики аппроксимации функций sin(x), cos(x) с использованием значения параметра SPREAD=0.09 при обучении НС.

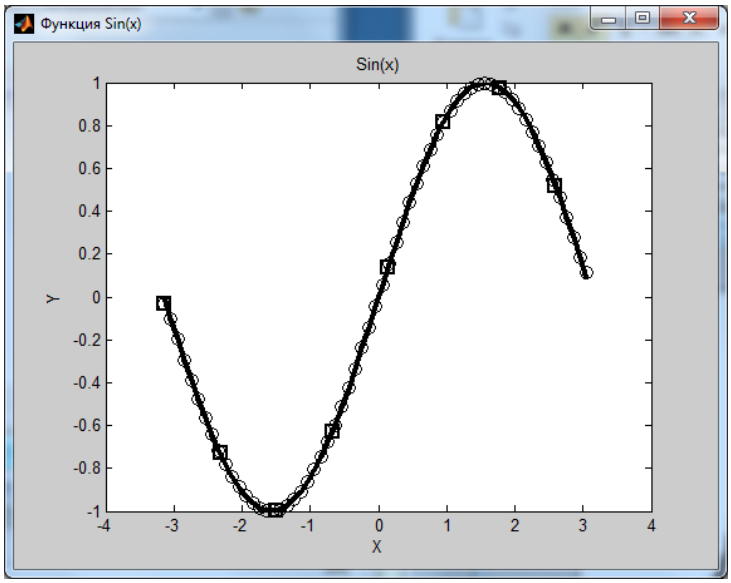


Рис. 8.15. График функции sin(x) и аппроксимации с помощью НС (newgrnn, spread=0.09, numneurons=63)

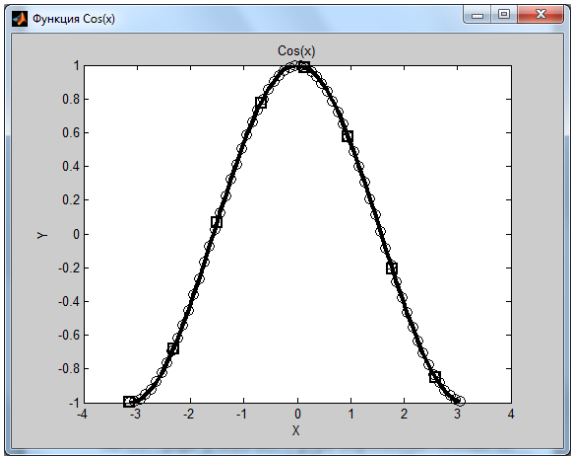


Рис. 8.16. График функции cos(x) и аппроксимации с помощью НС (newgrnn, spread=0.09, numneurons=63)

8.5. Влияние параметра SPREAD нейронной сети на результаты аппроксимации

Для исследования параметра SPREAD будет использована радиально-базисная нейронная сеть, созданная ранее. Для параметра GOAL устанавливается в значение 0.000001, которое обеспечит достаточно точную аппроксимацию функций sin(x), cos(x) при снижении времени обучения сети за счёт меньшего количества нейронов в скрытом слое (чем при GOAL=0).

Интервал значений входных векторов [-pi:01: pi].

Параметр влияния SPREAD существенно влияет на качество аппроксимации функции: чем его значение больше, тем более «гладкой» будет аппроксимация. Слишком большое значение параметра SPREAD приведет к тому, что для получения «гладкой» аппроксимации быстро изменяющейся функции потребуется большое количество нейронов скрытого слоя. Малое значение параметра SPREAD потребует большого количества нейронов для аппроксимации «гладкой» функции.

Интервал значений тестовой выборки [-pi: 0.05: pi].

Установим значение параметра SPREAD=0.01.

На нижеприведённых рис. 8.17-8.18 показаны графики функций sin(x), cos(x) (сплошная линия), графики аппроксимаций этих функций через НС (маркеры (‘o’) указывают на входные вектора обучения НС). Также точечными маркерами (‘.’) и линиями отмечены результаты работы НС при поступлении на вход тестового вектора.

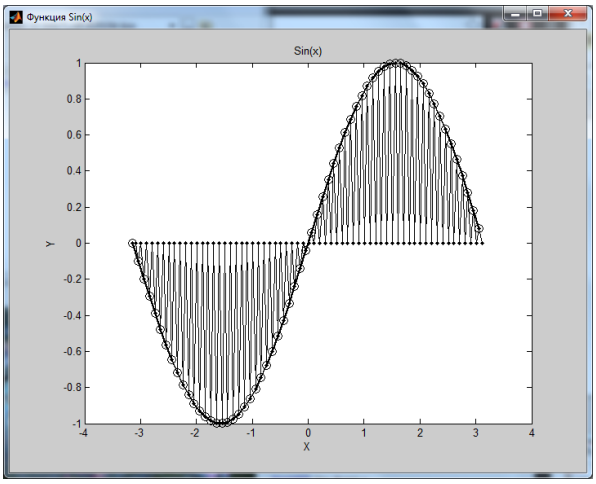


Рис. 8.17. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=0.01, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

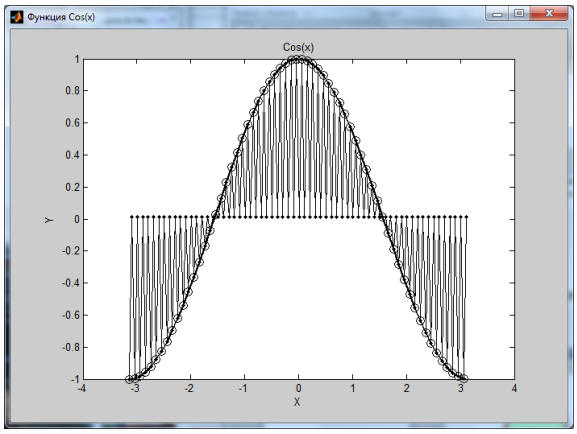


Рис. 8.18. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=0.01, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

Рис. 8.19-8.32 демонстрируют, что НС не обеспечивает необходимой гладкости аппроксимируемой функции, за счёт влияния параметра SPREAD.

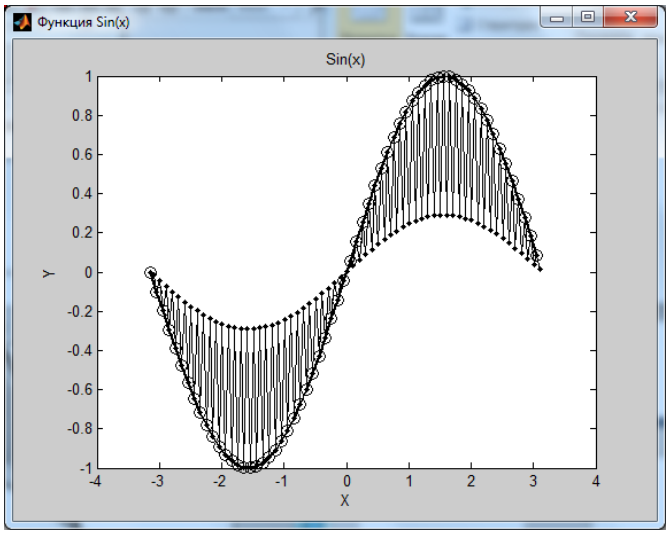


Рис. 8.19. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=0.03, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

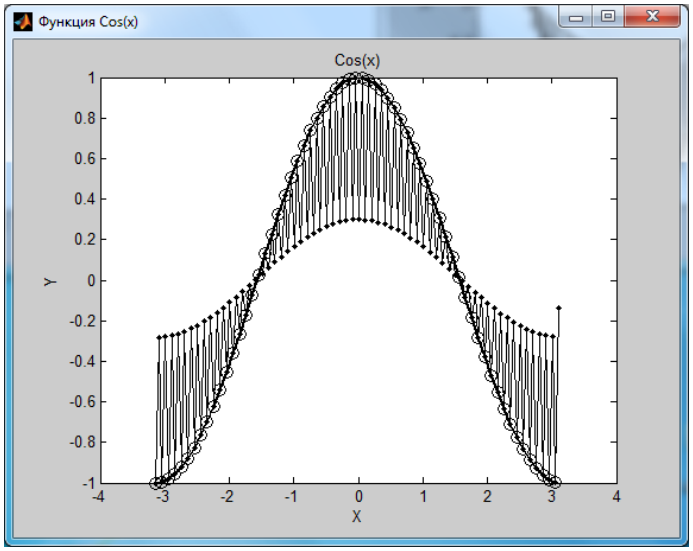


Рис. 8.20. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=0.03, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

Так как SPREAD=0.01, то интервал перекрытия входных значений составляет плюс-минус 0.01. При обучении НС векторами с шагом 0.1 значения тестового вектора не перекрываются функциями активации и не обеспечивается необходимая гладкость функции. При приближении параметра SPREAD к значению шага изменения входного вектора, аппроксимируемая функция приобретает необходимую гладкость.

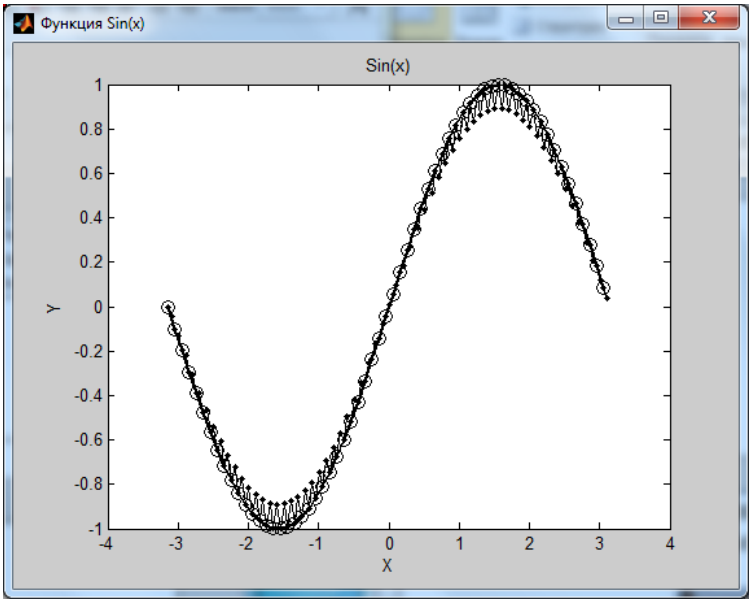


Рис. 8.21. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=0.05, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

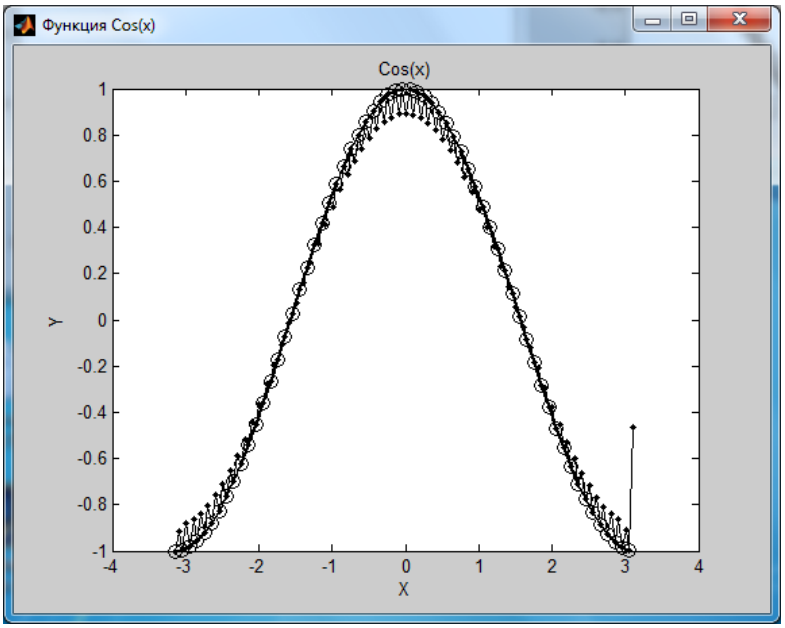


Рис. 8.22. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=0.05, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

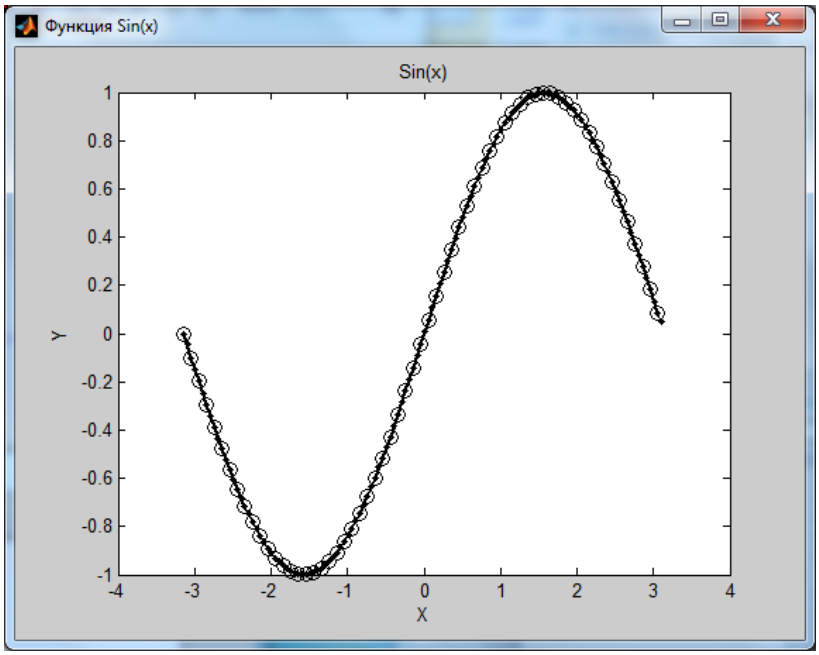


Рис. 8.23. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=0.07, numneurons=62, StepInVector=0.1, StepTestVector=0.05)

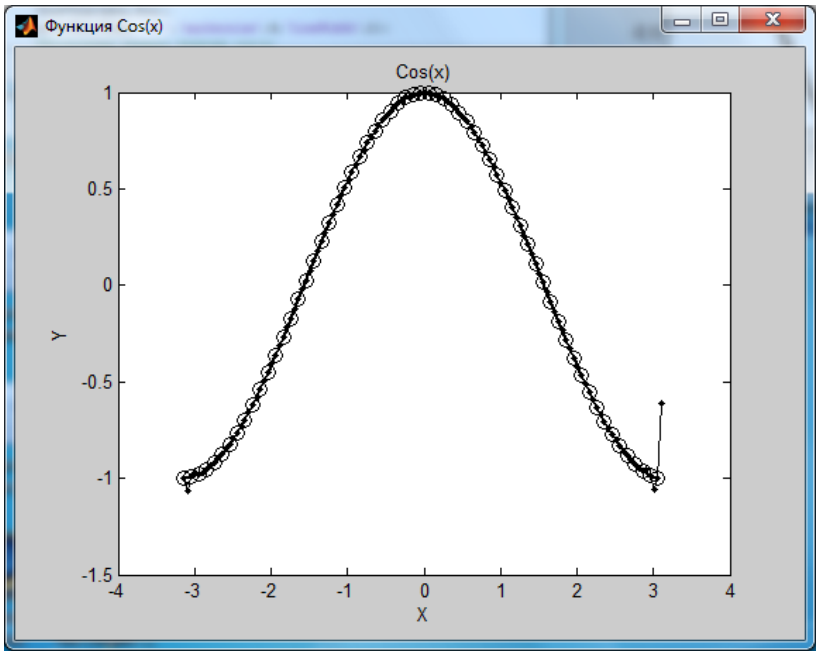


Рис. 8.24. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=0.07, numneurons=61, StepInVector=0.1, StepTestVector=0.05)

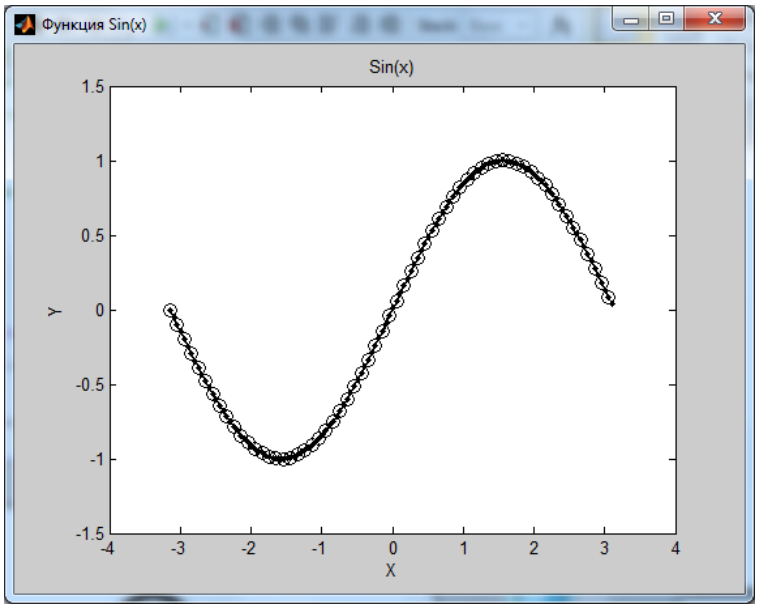


Рис. 8.25. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=0.5, numneurons=18, StepInVector=0.1, StepTestVector=0.05)

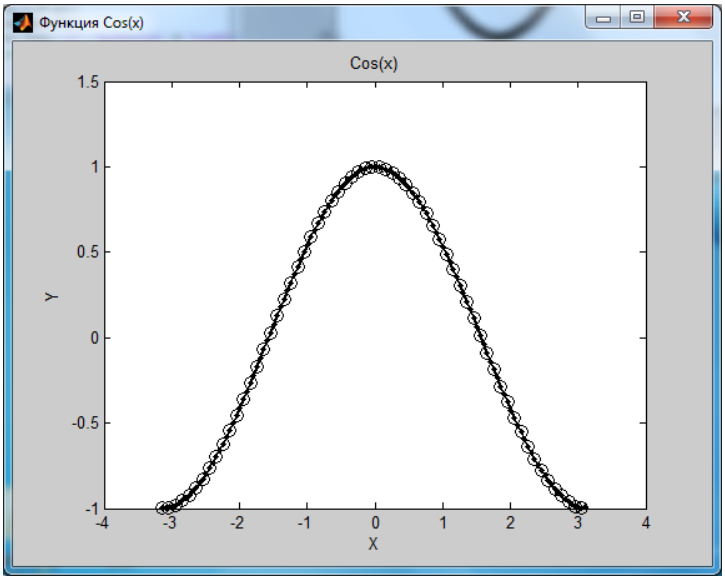


Рис. 8.26. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=0.5, numneurons=19, StepInVector=0.1, StepTestVector=0.05)

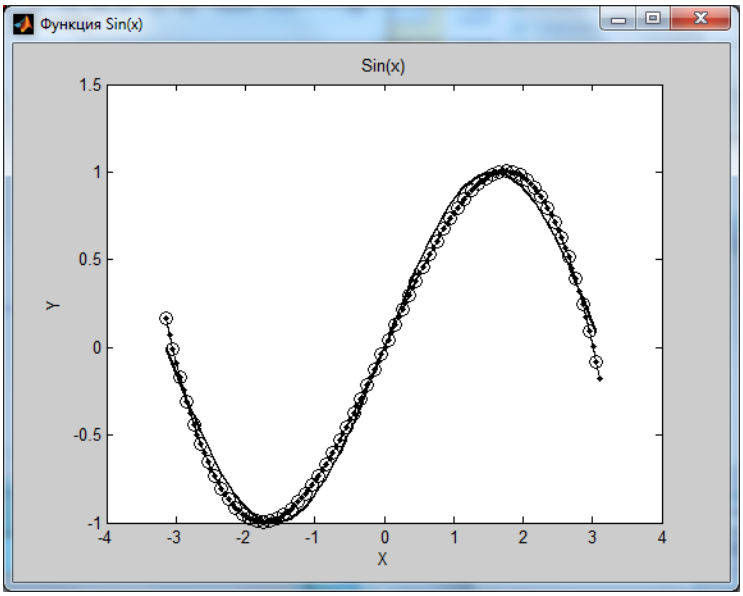


Рис. 8.27. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=250, numneurons=63, StepInVector=0.1, StepTestVector=0.05)

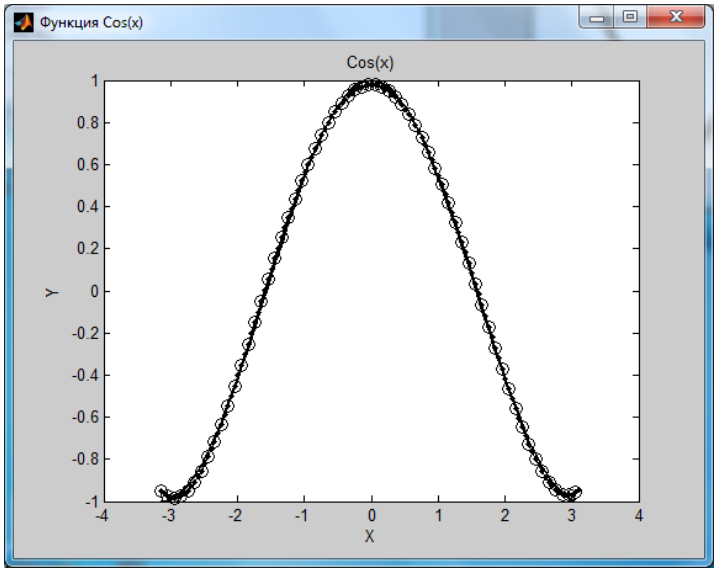


Рис. 8.28. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=250, numneurons=63, StepInVector=0.1, StepTestVector=0.05)

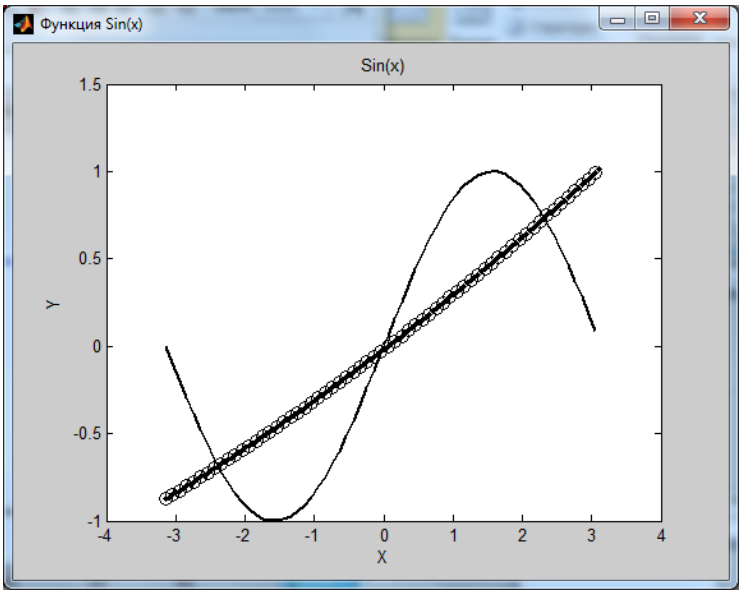


Рис. 8.29. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=400, numneurons=63, StepInVector=0.1, StepTestVector=0.05)

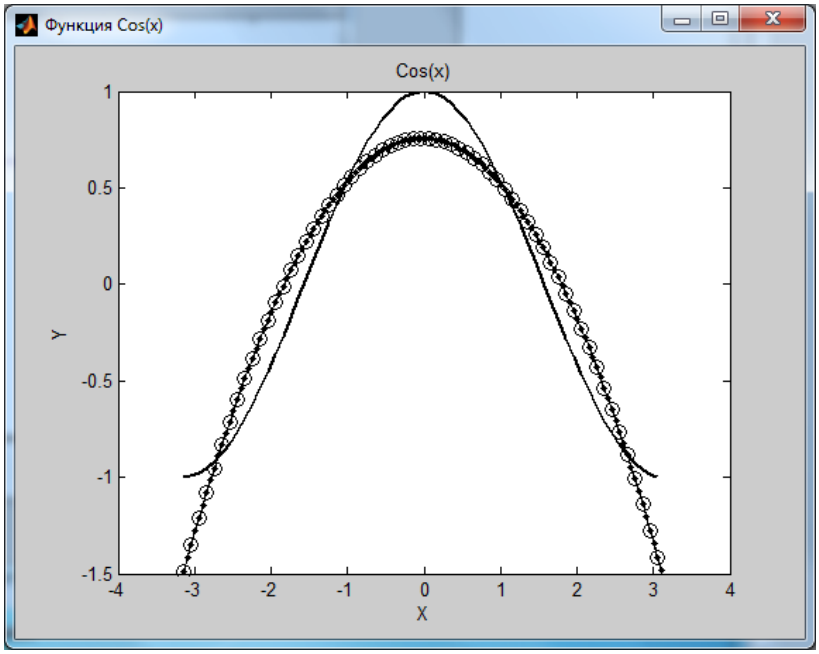


Рис. 8.30. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=400, numneurons=63, StepInVector=0.1, StepTestVector=0.05)

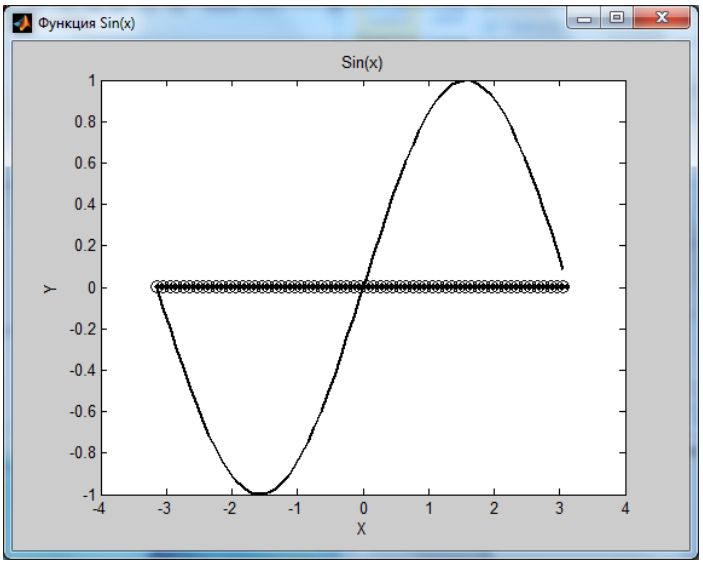


Рис. 8.31. График функции sin(x) и аппроксимации с помощью НС (newrb, spread=100000000, numneurons=63, StepInVector=0.1, StepTestVector=0.05)

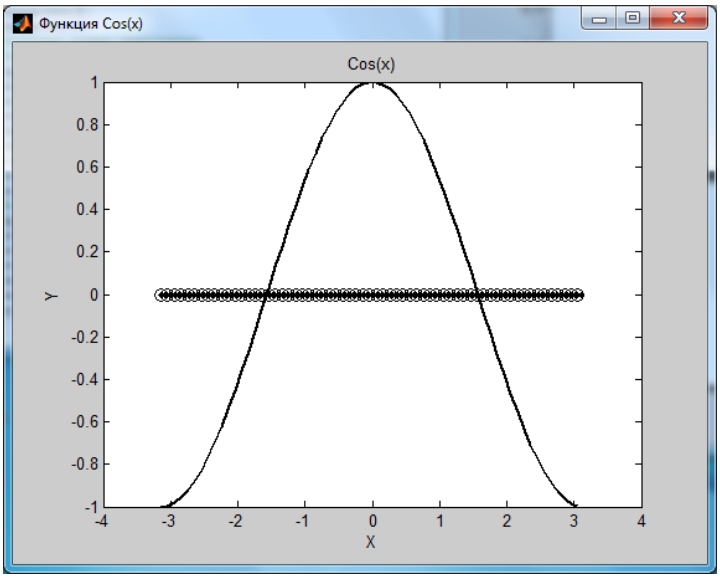


Рис. 8.32. График функции cos(x) и аппроксимации с помощью НС (newrb, spread=100000000, numneurons=63, StepInVector=0.1, StepTestVector=0.05)

При больших значениях SPREAD=100 000 000 (табл. 8.3) все функции активации перекрывают друг друга, и каждый базисный нейрон

выдает значение близкое к нулю для всех входных значений и сеть перестает реагировать на входные значения.

Таблица 8.3. Зависимость количества нейронов и средней ошибки от значения параметра влияния SPREAD

SPREAD	Количество нейронов		Среднеквадратичная ошибка	
	sin(x)	cos(x)	sin(x)	cos(x)
0.01	62	62	0	0
0.03	62	62	$4.7850*10^{-33}$	$6.5239*10^{-33}$
0.05	62	62	$9.5930*10^{-33}$	$7.6335*10^{-33}$
0.5	18	19	$9.5791*10^{-7}$	$6.3002*10^{-7}$
250	63	63	0.0044	$3.4797*10^{-4}$
400	63	63	0.1973	0.0387
100 000 000	63	63	0.4987	0.5013

Листинг программы исследования влияния параметра SPREAD на структуру и качество аппроксимации приведён ниже.

Файл LR3_p3_spread:

```
%Создание радиально-базисных нейронных сетей для аппроксимации
%функций
%sin(x) и cos(x)
%Задание интервала на котором моделируются целевые функции
%sin(x) и cos(x)
X = [-pi: 0.1: pi];
%Получение значения функций
Ys = sin(X);
Yc = cos(X);
%Создание радиально-базисных нейронных сетей
%SPREAD=1
nets = newrb(X, Ys, 0.000001, 0.01);
```

```
netc = newrb(X, Yc, 0.000001, 0.01);
% Симуляция
Yns=sim(nets,X);
Ync=sim(netc,X);
%Построение графика аппроксимации функции
%Построение графика функции sin(x)
figure('NumberTitle','off','Name','Функция
Sin(x)','ToolBar','none','MenuBar','none');
plot(X,Ys,'-r','linewidth',6);hold on;
plot(X,Yns,'-o','markersize',5); hold on;
title('Sin(x)');
xlabel('X');
ylabel('Y');
Xin=[-pi:0.05:pi];
Yout=sim(nets,Xin);
plot(Xin,Yout,'-g.','markersize',5);
%Построение графика функции cos(x)
figure('NumberTitle','off','Name','Функция
Cos(x)','ToolBar','none','MenuBar','none');
plot(X,Yc,'-r','linewidth',6);hold on;
plot(X,Ync,'-o','markersize',5); hold on;
title('Cos(x)');
xlabel('X');
ylabel('Y');
Xin=[-pi:0.05:pi];
Yout=sim(netc,Xin);
plot(Xin,Yout,'-g.','markersize',5);
%Количество нейронов в скрытом слое созданной НС
NumNeuronsSin=nets.layers{1}.size
```

```
NumNeuronsCos=netc.layers{1}.size
%Вычисление значения средней квадратичной ошибки обучения НС
Es=Ys-Yns;
Es = mse(Es)
Ec=Yc-Ync;
Ec = mse(Ec)
```

В результате исследований выявлено влияние изменения параметров нейронных сетей GOAL и SPREAD на качество аппроксимации функций. Зависимости количества нейронов и средней ошибки от значений параметров GOAL и SPREAD приведены в табл. 8.2 и табл.8.3.

Значение параметра влияния SPREAD с одной стороны должно быть достаточно большим, чтобы покрыть весь диапазон значений входного вектора. С другой стороны, параметр SPREAD не должен быть слишком большим, тогда сеть перестанет реагировать на различные элементы входного множества.

Увеличение параметра GOAL, т. е. уменьшение точности обучения сети приводит к уменьшению количества необходимых нейронов в скрытом слое радиально-базисной сети.

8.6. Моделирование вероятностных нейронных сетей (PNN) для задач классификации векторов

Вероятностные нейронные сети (PNN) являются разновидностью радиальных базисных сетей и применяются для решения вероятностных задач и задач классификации. Такие сети имеют высокую скорость обучения, но работают относительно медленно.

Листинг программы моделирования приведён ниже.

```
%Моделирование с помощью вероятностной нейронной сети PNN
```

```
%Задание множества двухэлементных векторов
P = [1 1 2 2 2 3 3 3 4 4 4 5 5 5 6;...
     1 2 1 2 3 2 3 4 3 4 5 4 5 6 5];
%Задание множества классов, каждый вектор соответствует %одному из 5 классов
Tclass = ...
    [1 1 1 2 2 2 3 3 3 4 4 4 5 5 5];
%Преобразование вектора в разряженную матрицу, определяющую %принадлежность векторов своим классам
Tmat = ind2vec(Tclass)
%Преобразование разреженной матрицы в полную матрицу
Tf = full(Tmat)
%Создание вероятностной нейронной сети PNN
net = newpnn(P,Tf);
%Задание тестовых векторов
testP = [5 4 3 2 1;...
         6 5 4 3 2];
%Моделирование полученной вероятностной НС
%векторами, принадлежащими разным классам
Y = sim(net,testP);
%Преобразование матрицы в индексный вектор
class = vec2ind(Y);
class
```

Результаты работы программы:

```
Tmat =
  (1,1)       1
  (1,2)       1
  (1,3)       1
  (2,4)       1
```

```
    (2,5)       1
    (2,6)       1
    (3,7)       1
    (3,8)       1
    (3,9)       1
    (4,10)      1
    (4,11)      1
    (4,12)      1
    (5,13)      1
    (5,14)      1
    (5,15)      1
Tf =
  Columns 1 through 10
     1     1     1     0     0     0     0     0     0     0
     0     0     0     1     1     1     0     0     0     0
     0     0     0     0     0     0     1     1     1     0
     0     0     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     0
  Columns 11 through 15
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     1     1     0     0     0
     0     0     1     1     1
class =
     5     4     3     2     1
```

Результаты работы программы показывают, что тестовые векторы правильно классифицировались вероятностной нейронной сетью (PNN).

8.7. Моделирование радиально-базисных нейронных сетей для аппроксимации функции tang(x)

Результаты аппроксимации радиально-базисной сети для аппроксимации функции tang(x) представлены в табл. 8.4 и 8.5.

Таблица 8.4. Радиально-базисная сеть с нулевой ошибкой для аппроксимации функции tang(x)

Количество нейронов в скрытом слое	Точные значения функции tang(x)					Средняя ошибка
	14.101	0.69891	-0.309	-0.7602	-5.7978	
1	0.01	0,72046	-0,303	-0,8025	-5,7336	2.478
2	1,7141	0,72046	-0,303	-0,8025	-5,7336	2.478

Таблица 8.5. Изменение значений ошибки в зависимости от параметров радиально-базисной сети

Допустимое значение функционала	Значение параметра влияния	Точные значения функции tang(x)					Средняя ошибка
		-0.995	-0.96334	-0.862	0.0407	0.819	
0.01	1,71417	1,7141	0,72046	-0,3038	-0,8025	-5,733	2.478
4	0.001	1,7141	0,72046	-0,3038	-0,8025	-5,733	2.478
0.0001	10	1,7141	0,72046	-0,3038	-0,8025	-5,733	2.478
10	0.0000001	1,7141	0,72046	-0,3038	-0,8025	-5,733	2.478

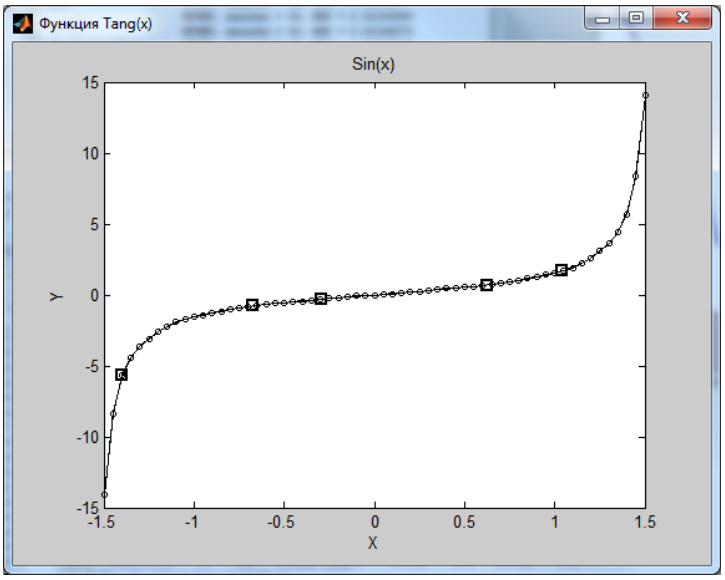


Рис. 8.33. Результаты аппроксимации функции tang(x)

При аппроксимации функции тангенса с помощью радиально-базисных функций значения получились намного точнее, чем в случае с многослойной сетью.

Изменение параметров сети также не повлияло на результаты, и при любых значениях входных параметров сети ошибка получилась не более 2.478, что говорит о точности полученных результатов.

Сравнительный анализ результатов моделирования многослойной и нейронной сети с радиально-базисными функциями представлен в табл. 8.6 и 8.7.

Сравнительный анализ данных, приведённых в табл. 6.6 и 6.7, показывает, что результаты аппроксимирования функций с помощью РБФ-сетей на порядок выше, чем при использовании многослойных сетей. Это можно объяснить тем фактом, что в основе РБФ-сетей лежат радиально-базисные функции, которые являются радиально-симметричными функциями относительно начала координат.

Таблица 8.6. Результаты моделирования функции tang (x)

tang (x)					
Многослойная сеть	0,99810	0,98586	-0,29193	-0,96162	-0,99704
РБФ- сеть	1,71417	0,72046	-0,30384	-0,80251	-5,73363
Точные значения	14.10142	0.698919	-0.30934	-0.7602	-5.79788

Таблица 8.7. Результаты моделирования функции cos (x)

cos (x)					
Многослойная сеть	-0.99952	-0.95787	-0.86069	0.04016	0.82201
РБФ-сеть	-0,99580	-0,96334	-0,86200	0,040785	0,819648
Точные значения	-0.99581	-0.96334	-0.862	0.040785	0.819648

Применение РБФ-сетей для аппроксимации функций обеспечивает решение задачи, алгоритм обучения РБФ сети сходится гораздо быстрее, чем алгоритм обратного распространения ошибки при обучении многослойной сети.

Для обеспечения высокой точности РБФ-сеть требует большое число скрытых нейронов, что приводит к замедлению функционирования RBF сети по сравнению с многослойной нейронной сетью.

Перечисленные примеры демонстрируют, что нейронные сети обеспечивают решение различных задач аппроксимации тригонометрических функций.

8.8. Пример моделирования задачи кластеризации с помощью нейронной сети Кохонена

Нейронная сеть Кохонена применяется в задачах кластеризации. Обучение сети Кохонена происходит без учителя. Для создания сети необходимо использовать функцию newsom. Сокращение SOM (self-organization map) в названии функции соответствует самоорганизующейся карте.

Созданная нейронная сеть Кохонена предназначена для разделения входных данных на несколько кластеров. Количество формируемых кластеров зависит от количества нейронов в сети.

Для исследования карт будем использовать различные гексагональные сетки, изменяя количество нейронов при создании сети с помощью функции newsom.

Исследование задачи кластеризации проведено на примере технической диагностики персонального компьютера. Предположим, что у нас имеется 20 компьютеров, в каждом из которых может возникнуть 3 вида отказа или компьютер может быть исправен. Эту задачу можно представить массивом из 20 векторов, каждый вектор содержит по 2 элемента. Каждый компьютер можно отнести к одному из четырёх кластеров:

1. нет отказа, компьютер исправен;
2. отказ в блоке питания;
3. отказ в материнской плате;
4. отказ в жестком диске (HDD).

Сеть Кохонена решает задачу технической диагностики, используя метод конкуренции. Чтобы смоделировать задачу технической диагностики для четырёх ситуаций необходимо использовать четыре нейрона.

Листинг программы, моделирующей задачу технической диагностики персональных компьютеров с помощью нейронной сети Кохонена, приведён ниже.

```
%Процедура очистки переменных среды
clear;
%Задание координат центров кластеров (-5; -5), (5; -5),
%(-5;5), (5;5) плюс случайный разброс от 0 до 1.
X1=-5+rand (1, 5);
y1=-5+rand (1, 5);
X2=5+rand (1, 5);
y2=-5+rand (1, 5);
X3=-5+rand (1, 5);
y3=5+rand (1, 5);
X4=5+rand(1,5);
Y4=5+rand(1,5);
%Объединение вектора в один массив
X (1:5) =x1;
X (6:10) =x2;
X (11:15) =x3;
X (16:20) =x4;
Y (1:5) =y1;
Y(6:10) =y2;
Y(11:15) =y3;
Y(16:20) =y4;
```

```
%Создание обучающего вектора
Z (1, 1:20) =x;
Z (2, 1:20) =y;
%Создание нейронной сети Кохонена с гексагональной сеткой 2x2
net = newsom(z,[2 2]);
%Построение топологии двумерной карты Кохонена
figure('NumberTitle','off','Name','Карта размещения нейронов','ToolBar','none','MenuBar','none');
plotsom(net.layers{1}.positions);
%Настройка параметров обучения и обучение сети
net.trainparam.epochs = 1000;
net = train(net, z);
%Построение входных векторов с их отображениями, которые
% формируются весами SOM
figure('NumberTitle','off','Name','Векторы весов','ToolBar','none','MenuBar','none');
plot(x1,y1,'ob');hold on;grid on;
plot(x2,y2,'or');
plot(x3,y3,'og');
plot(x4,y4,'ok');
plotsom(net.iw{1,1},net.layers{1}.distances);hold off;
%Моделирование и проверка получившейся нейронной сети на %входном массиве
Outnet = sim(net,z)
OutIndex = vec2ind(Outnet)
figure('NumberTitle','off','Name','Расположение векторов по кластерам','ToolBar','none','MenuBar','none');
bar(sum(Outnet'));
```

Результаты работы программы представлены ниже.

```
Outnet =
  Columns 1 through 14
    0    0    0    0    0    1    1    1    1    1    0    0    0    0
    1    1    1    1    1    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    1    1    1    1

  Columns 15 through 20
    0    0    0    0    0    0
    0    0    0    0    0    0
    0    1    1    1    1    1
    1    0    0    0    0    0
OutIndex =
  Columns 1 through 14
    2    2    2    2    2    1    1    1    1    1    4    4    4    4
  Columns 15 through 20
    4    3    3    3    3    3
```

Проанализировав полученные результаты, можно сделать вывод о том, что нейронная сеть Кохонена обучилась и работает правильно, так как каждый элемент вектора был кластеризован правильно. Ниже на рис. 8.34-8.42 приведены графики работы нейронной сети Кохонена.

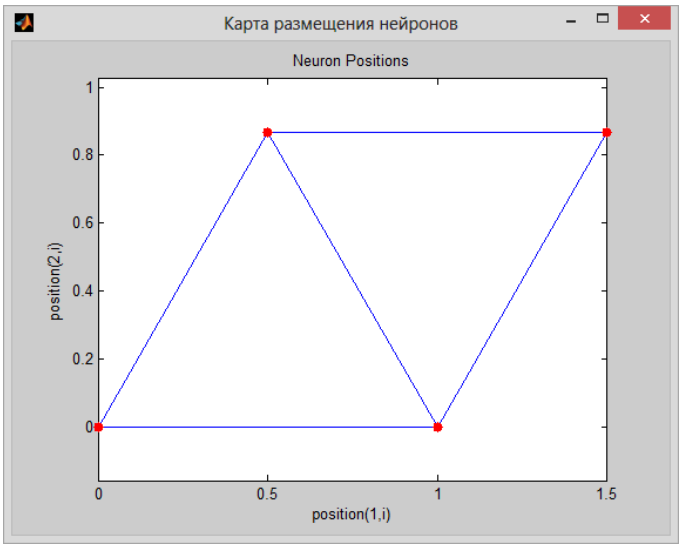


Рис. 8.34. Топология двумерной карты Кохонена (гексагональная сетка 2x2)

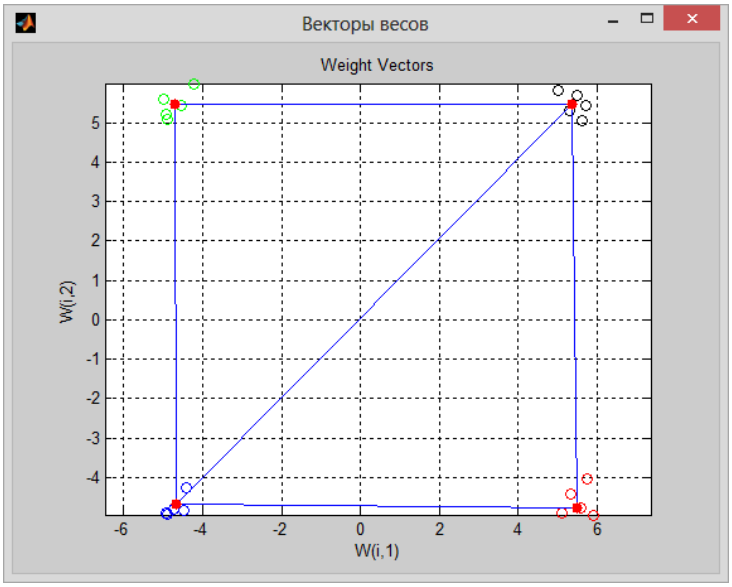


Рис. 8.35. Входные вектора с их отображениями, которые формируются весами SOM (гексагональная сетка 2x2)

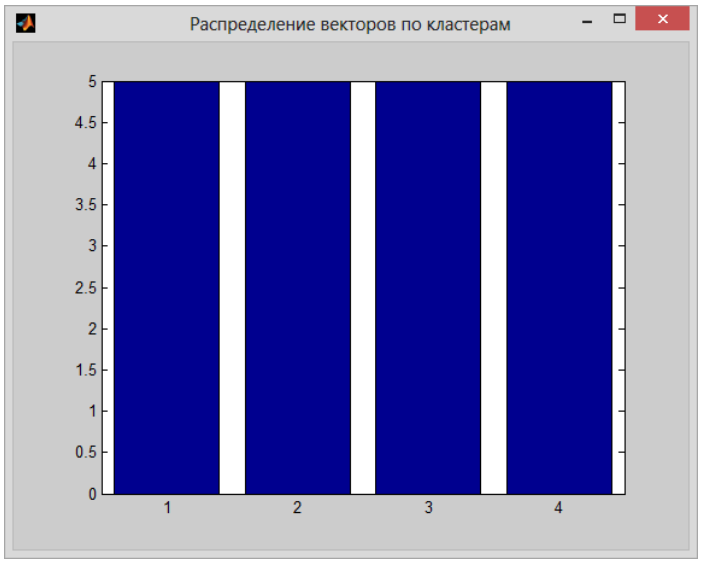


Рис. 8.36. Распределение векторов по кластерам

(гексагональная сетка 2x2)

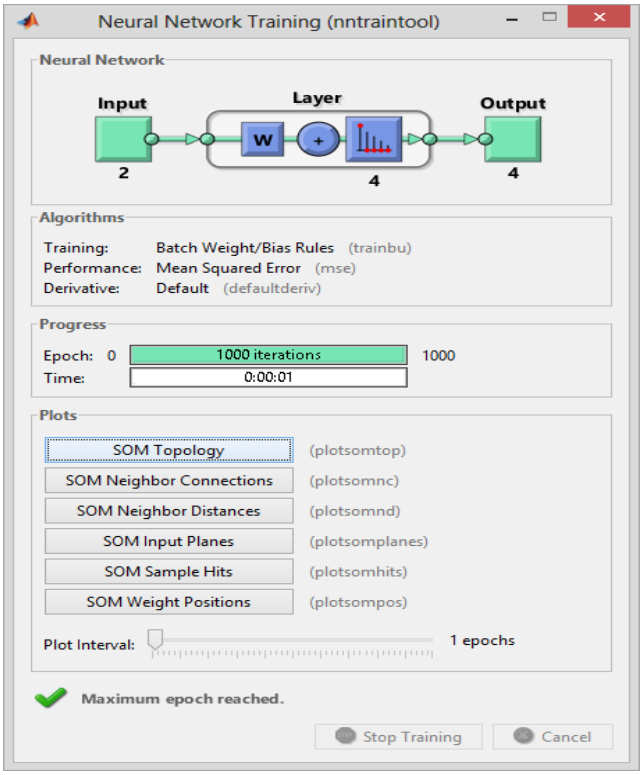


Рис. 8.37. Результаты обучения нейронной сети Кохонена

(гексагональная сетка 2x2)

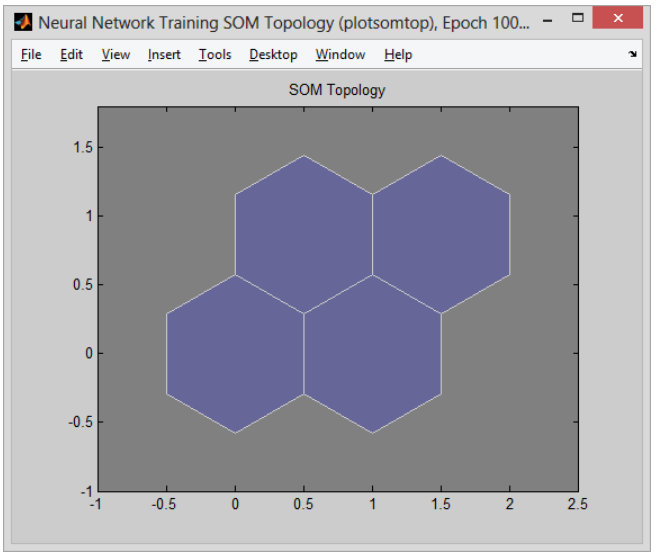


Рис. 8.38. Топология обученной нейронной сети

(окно по ссылке SOM Topology, гексагональная сетка 2x2)

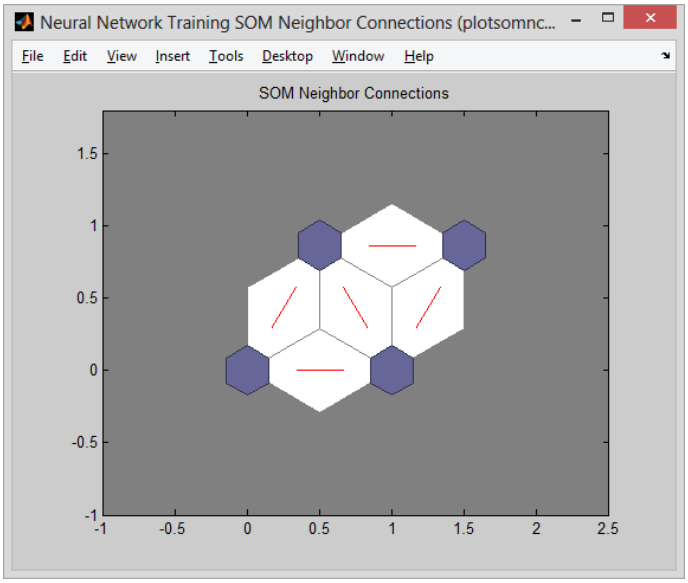


Рис. 8.39. Соседние соединения НС

(окно по ссылке SOM Neighbor Connections, гексагональная сетка 2x2)

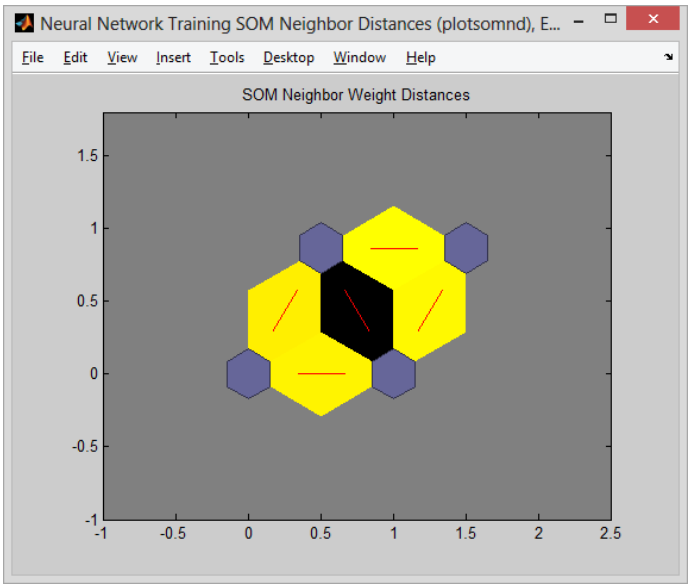


Рис. 8.40. Соседние расстояния НС

(окно по ссылке SOM Neighbor Distances, гексагональная сетка 2x2)

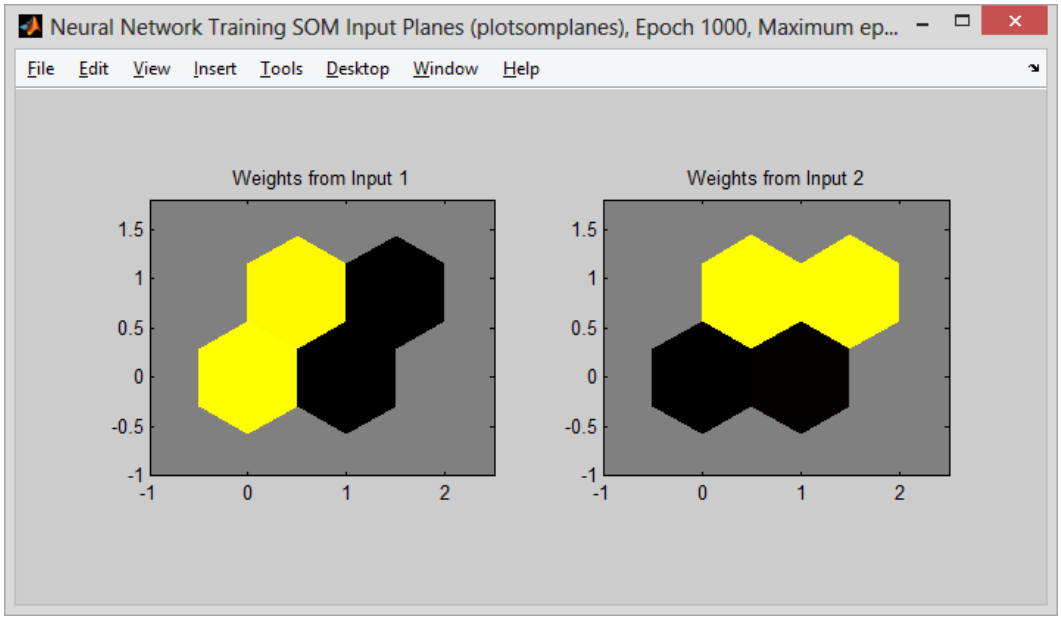


Рис. 8.41. Входные плоскости

(окно по ссылке SOM Input Planes, гексагональная сетка 2x2)

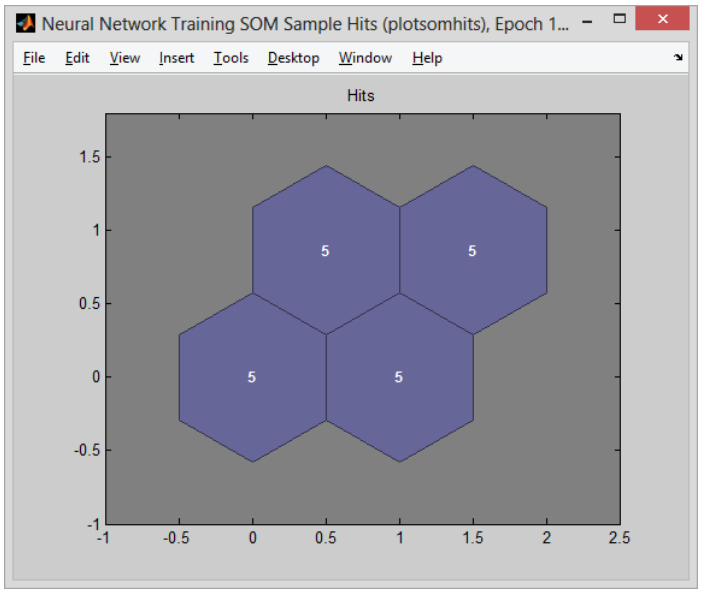


Рис. 8.42. Распределение векторов по плоскостям
(окно по ссылке SOM Sample Hits, гексагональная сетка 2x2)

Сравнительный анализ моделируемых сетей Кохонена в зависимости от различной размерности гексагональной сетки позволяет сделать следующие выводы:

1. Для решения задачи технической диагностики четырьмя выходными значениями, необходимо четыре нейрона. Топология расположения этих нейронов (1x4, 4x1, 2x2) ни как не влияет на решение задачи.

2. При использовании количества нейронов меньше четырёх, задача не может быть решена правильно, так как этого количества недостаточно для описания четырёх ситуаций.

3. При использовании количества нейронов больше четырёх задача также не решается правильно, так как появляется излишняя точность в кластеризации (4x4). При этом распределение количества

элементов вектора по нейронам варьируется от 0 до 5. Такое количество нейронов оказывается избыточным.

4. Необходимое и достаточное количество нейронов для решения задачи кластеризации в нейронной сети Кохонена четырёх состояний персональных компьютеров равно четырём.

8.9. Пример моделирования задачи кластеризации с помощью LVQ нейронной сети

Исследование работы LVQ нейронной сети проводилось на той же задаче технической диагностики. Результирующая нейронная LVQ сеть была обучена и продемонстрировала правильные результаты. Для решения задачи потребовалось также четыре нейрона.

Листинг программы, моделирующей задачу технической диагностики персональных компьютеров с помощью LVQ нейронной сети, приведён ниже.

```
%Процедура очистки переменных среды
clear;
%Задание координат центров кластеров (-5;-5),(5;-5),
%(-5;5),(5;5) плюс случайный разброс от 0 до 1.
x1=-5+rand(1,5);
x2=5+rand(1,5);
y2=-5+rand(1,5);
y1=-5+rand(1,5);

x3=-5+rand(1,5);
y3=5+rand(1,5);
x4=5+rand(1,5);
y4=5+rand(1,5);
```

```
%Задаём цели
T1(1:5)=1;
T2(1:5)=2;
T3(1:5)=3;
T4(1:5)=4;
%Объединение вектора в один массив
x(1:5)=x1;
x(6:10)=x2;
x(11:15)=x3;
x(16:20)=x4;
y(1:5)=y1;
y(6:10)=y2;
y(11:15)=y3;
y(16:20)=y4;
Tc(1:5)=T1;
Tc(6:10)=T2;
Tc(11:15)=T3;
Tc(16:20)=T4;
%Создание обучающего вектора и целевого вектора
z(1,1:20)=x;
z(2,1:20)=y;
%Преобразование целевых классов в целевые векторы
T= ind2vec(Tc);
%Создание нейронной сети LVQ сеть с 4 нейронами и процентной долей %каждого кластера по 25%
net = newlvq(z,4,[.25 .25 .25 .25]);
%Настройка параметров обучения и обучение сети
net.trainparam.goal=0;
net.trainparam.epochs = 1000;
```

```
net = train(net,z,T);
%Построение входных векторов на координатной плоскости
figure('NumberTitle','off','Name','Входные
векторы','ToolBar','none','MenuBar','none');
plot(x1,y1,'ob');hold on;grid on;
plot(x2,y2,'or');
plot(x3,y3,'og');
plot(x4,y4,'ok');hold off;
%Моделирование и проверка получившейся нейронной сети на
входном %массиве
Outnet = sim(net,z)
OutIndex = vec2ind(Outnet)
figure('NumberTitle','off','Name','Распределение векторов по
кластерам','ToolBar','none','MenuBar','none');
bar(sum(Outnet'));
```

Результаты работы программы**:**

```
Outnet =

  Columns 1 through 14

     1     1     1     1     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     1     1     1     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     1     1     1     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 15 through 20

     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
     1     0     0     0     0     0
     0     1     1     1     1     1
OutIndex =
  Columns 1 through 14
     1     1     1     1     1     2     2     2     2     2     3     3     3     3
  Columns 15 through 20
     3     4     4     4     4     4
```

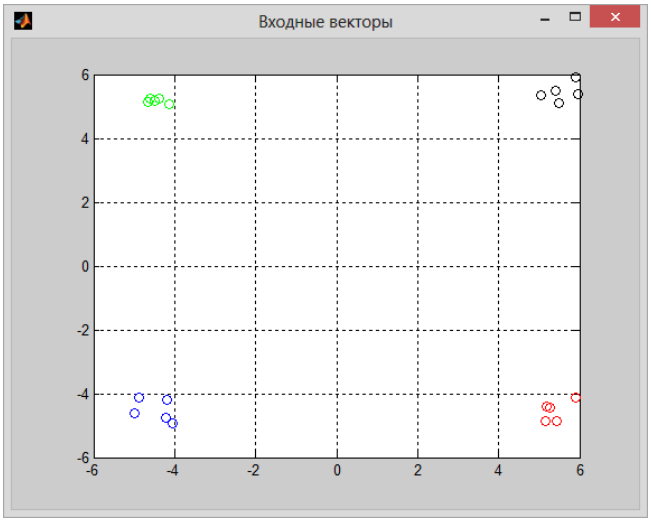


Рис. 8.43. Расположение входных векторов на координатной плоскости

(заданные значения вероятностей распределения 25%,25%,25%,25%)

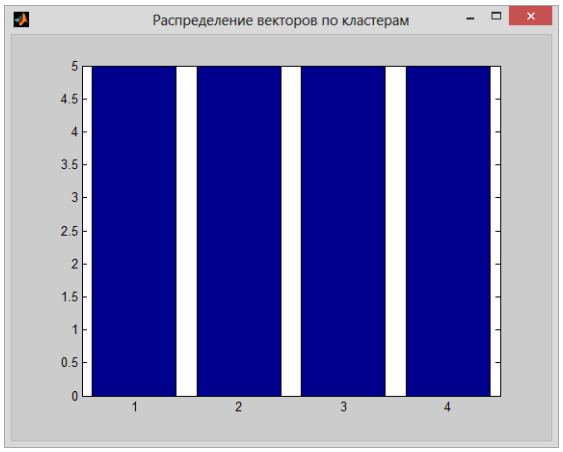


Рис. 8.44. Распределение векторов по кластерам

(заданные значения вероятностей распределения 25%,25%,25%,25%)

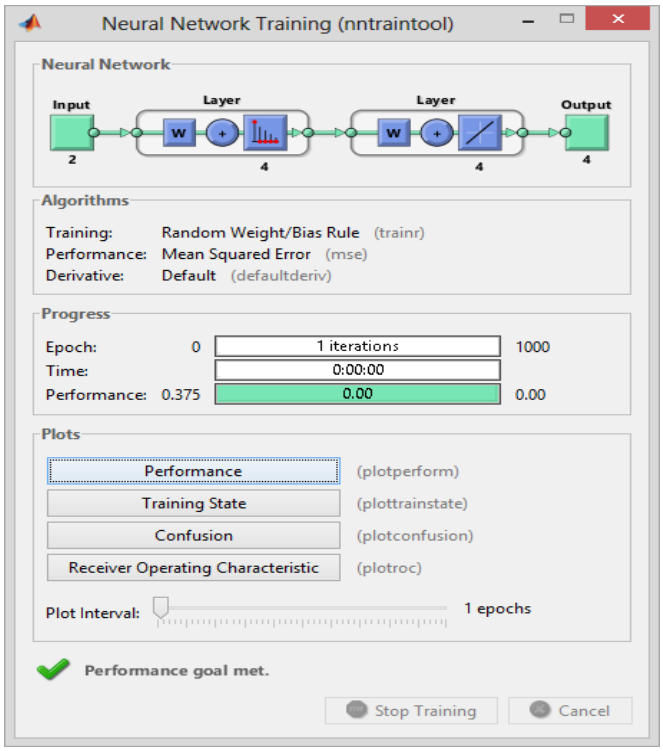


Рис. 8.45. Окно процесса обучения нейронной сети LVQ

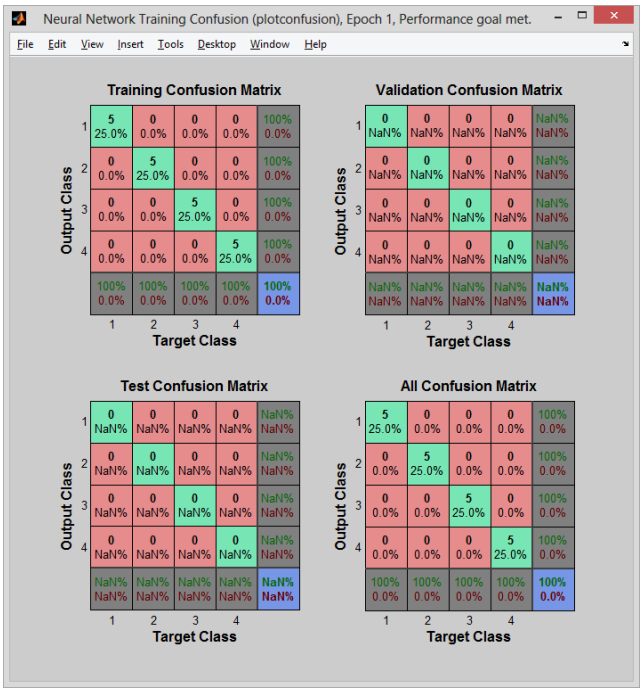


Рис. 8.46. Вероятности распределения (окно по ссылке Confusion)(заданные значения вероятностей распределения 25%,25%,25%,25%)

При изменении значений вероятности на 40%,40%,10%,10% нейронная сеть классифицирует входные векторы неправильно (см. графики на рис.8.47-8.54).

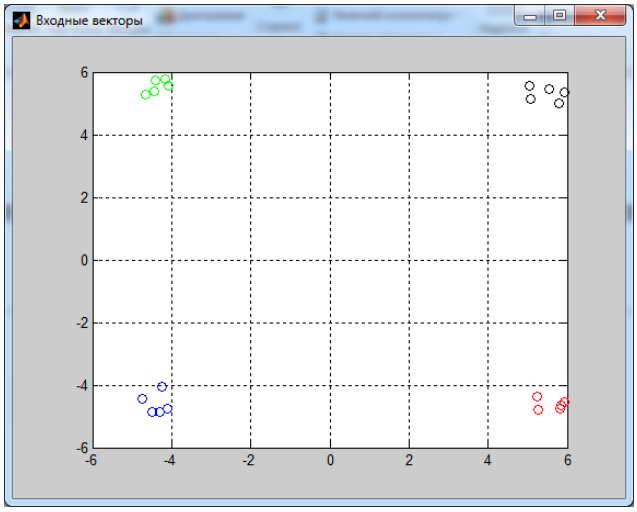


Рис. 8.47. Расположение входных векторов на координатной плоскости (заданные значения вероятностей распределения 40%, 40%,10%,10%)

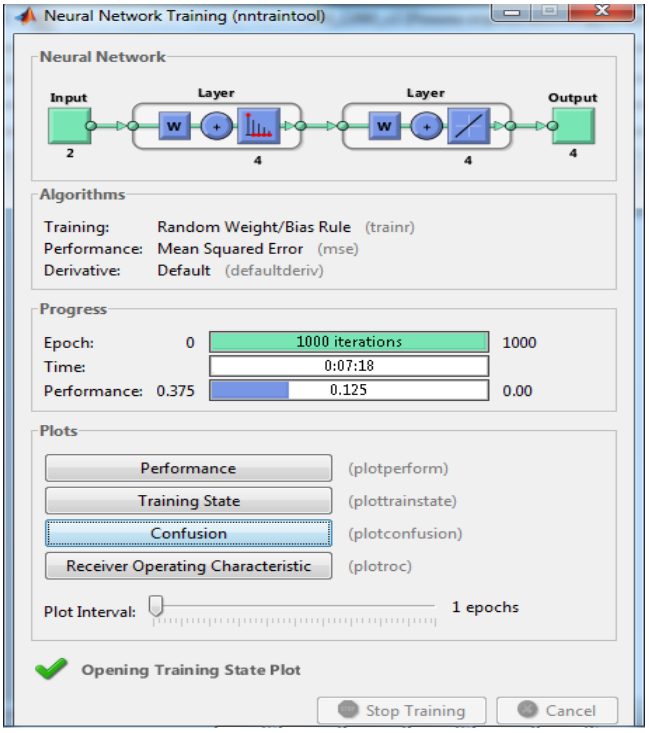


Рис. 8.48. Окно процесса обучения нейронной сети LVQ(заданные значения вероятностей распределения 40%,40%,10%,10%)

Рис. 8.49. Вероятности распределения (окно по ссылке Confusion) (заданные значения вероятностей распределения 40%,40%,10%,10%)

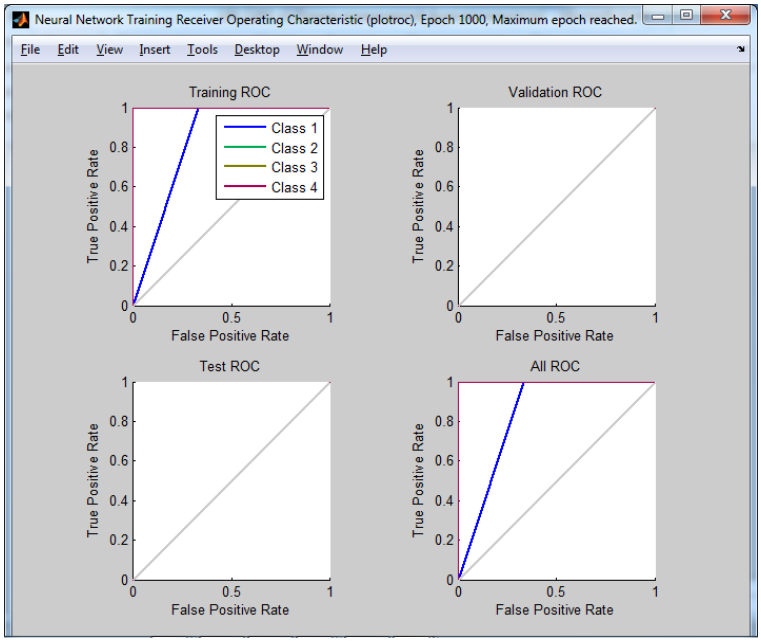


Рис. 8.50. Характеристика управления передатчиком НС (окно Reciver Operating Characteristic) (заданные значения вероятностей распределения 40%,40%,10%,10%)

Нейронная сеть Кохонена содержит один слой нейронов и является самоорганизующейся сетью, обучается без учителя. Количество входов каждого нейрона равно размерности входного вектора. Количество

нейронов непосредственно определяет, сколько различных кластеров сеть может распознать.

В результате исследования влияния размерности гексагональной сетки можно сделать вывод, что необходимое и достаточное количество нейронов для решения задачи с четырьмя классами должно быть равно четырём.

LVQ нейронная сеть является развитием сети Кохонена и в отличие от самоорганизующихся сетей выполняет не только кластеризацию, но и классификацию. Первый слой сети обучается без учителя, а второй слой обучается с учителем, обеспечивая более определённые результаты, чем в сети Кохонена.

9. МОДЕЛИРОВАНИЕ РЕШЕНИЯ СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

9.1. Объект моделирования - система дифференциальных уравнений

В [33,35] рассмотрено применение нейронных сетей для решения системы дифференциальных уравнений следующих объектов моделирования:

- система линейных дифференциальных уравнений;
- система нелинейных дифференциальных уравнений;
- нелинейное дифференциальное уравнение с разрывной функцией в правой части уравнений.

Нейронные сети можно использовать для аппроксимации как непрерывных, так и разрывных решений дифференциальных уравнений [42,43].

Трудности возникают в случае бесконечного разрыва, так как на выходе нейронной сети может появиться только конечное число, то в точке бесконечного разрыва искомого решения возникают конечные значения, получаемые нейронной сетью.

Система нелинейных дифференциальных уравнений не имеет аналитического решения. Поэтому обучающая выборка строилась с помощью численного метода Рунге-Кутта с автоматическим выбором шага.

Для проверки результатов моделирования нелинейного дифференциального уравнения с разрывной правой частью было получено аналитическое решение.

В качестве оценки функционирования нейронной сети выбрана сумма квадратичных отклонений выходов сети от эталонов.

С помощью пакета программ MATLAB рассмотрены и исследованы многослойные нейронные сети обратного распространения и нейронные RBF-сети.

В качестве обучающего алгоритма использованы алгоритм обучения Левенберга-Марквардта и байесовский метод [43].

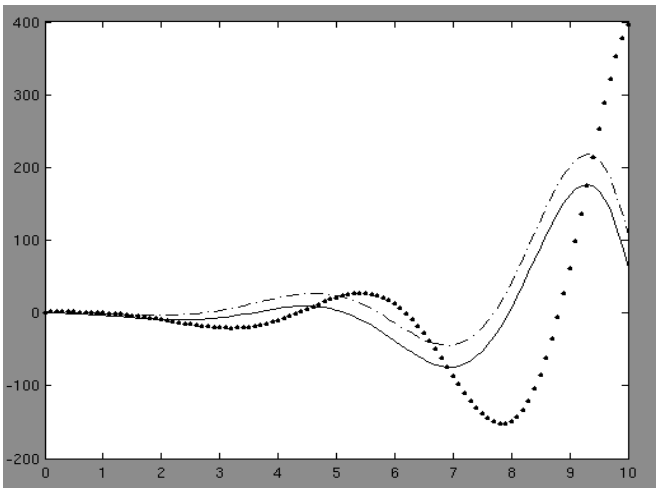
Для аппроксимации нелинейного отображения с помощью многослойного персептрона может потребоваться меньшее число параметров, чем для сети RBF при одинаковой точности вычислений.

9.2. Объект моделирования - система линейных дифференциальных уравнений

В качестве примера рассматривается система линейных дифференциальных уравнений

$$\begin{cases} y_1' = (y_2 - y_3) - 5 \cdot t, \\ y_2' = y_1 - y_3, \\ y_3' = -0.51 \cdot t + y_1 + y_2, \\ y_1(0) = 0, \\ y_2(0) = 1, \\ y_3(0) = 1. \end{cases}$$

Данная система была решена аналитически (рис. 9.1) и на его основе сформирована обучающая выборка из 100 примеров. Значения независимой переменной t выбирались равномерно в пределах от 0 до 10 с шагом 0,1.



Сплошная линия –; штрихпунктирная линия –;

Пунктирная линия – $y_3(t)$

Рис. 9.1. График решения системы линейных дифференциальных уравнений

9.3. Выбор алгоритма обучения и параметров трехслойной сети обратного распространения

С помощью пакета программ MATLAB создана и обучена трехслойная сеть обратного распространения, включающая один нейрон во входном слое, три нейрона в выходном слое. Число нейронов в скрытом слое варьировалось с целью подбора наиболее оптимальной архитектуры.

В качестве обучающего алгоритма выбран алгоритм Левенберга-Марквардта и байесовский метод [38,44]. Другие алгоритмы обучения на данной выборке продемонстрировали худшие результаты по критерию качества функционирования сети.

В качестве оценки функционирования сети выбрана сумма квадратичных отклонений выходов сети от их эталонных значений. Обучение считается законченным при условии, что сумма квадратичных отклонений выходов сети от их эталонных значений не превышает 0,001. Максимальное количество циклов обучения - 15000. В среде MATLAB

описанные действия представлены кодом для нейронной сети с 18 нейронами в скрытом слое и обучающим алгоритмом Левенберга-Марквардта, листинг которого приведён ниже.

```
net=newff(minmax(t),[1,18,3],{'logsig' 'logsig' 'purelin'},'trainlm');
net.performFcn='sse';
net.trainParam.goal=0.001;
net.trainParam.epochs=15000;
[net,tr]=train(net,t,y);
```

В результате варьирования функций активации для трехслойной сети с 18 нейронами в скрытом слое было выявлено, что данная нейронная сеть обучается лишь в том случае, если в скрытом слое выбрана сигмоидальная логистическая функция, а в выходном слое – линейная функция. При этом для входного слоя лучше выбрать нелинейную сигмоидальную логистическую функцию активации.

9.4. Варьирование параметра SPREAD для нейронной радиально-базисной сети

На той же обучающей выборке была обучена нейронная радиальная базисная сеть (RBF-сеть) с точностью 0,0000001. Проведено варьирование значения параметра SPREAD.

Тестирование проводилось на выборке в 100000 примеров (табл. 9.1).

Анализ данных табл. 9.1 показывает, что время обучения RBF-сети и время работы, принимают минимальные значения при изменении параметра SPREAD из интервала от одного до двух.

В качестве оптимального значения параметра SPREAD выбрана единица, которая обеспечивает минимальное время работы.

По критерию минимального времени обучения, средней ошибки лучше выбрать параметр SPREAD равный двум.

Таблица 9.1. Варьирование параметра SPREAD для нейронной RBF-сети

SPREAD	Время обучения, с	Время работы, с	Максимальная ошибка	Средняя ошибка	Число нейронов
0,2	25,34	2,30	1,407979401	0,00559038	88
0,4	15,59	1,53	0,007366237	0,00024368	48
0,6	12,37	1,73	0,002655288	0,00016707	36
0,8	8,52	1,75	0,000632534	0,00014276	28
1	8,42	1,14	0,001402572	0,00018918	23
1,5	8,01	1,11	0,001402572	0,00018918	20
2	8,25	1,16	0,000721534	0,00010629	22
2,5	14,63	1,48	0,000858799	0,00014250	43

9.5. Выбор оптимальной архитектуры нейронной сети

Время работы и ошибки рассчитывались для нейронных сетей разной архитектуры на 1000000 точек с параметрами, выбранными выше в качестве оптимальных. Ошибки рассчитывались в сравнении с аналитическим решением, результаты которых приведены в табл. 9.2.

Время работы трехслойной нейронной сети возрастает по мере роста числа нейронов скрытого слоя. Сети, обученные по алгоритму Байеса, работают медленнее, чем сети, обученные по алгоритму Левенберга-Марквардта. Это связано с тем, что веса синапсов при обучении задаются случайным образом.

Таблица 9.2. Результаты моделирования нейронной сети

Архитектура сети		Время обучения, с	Время работы, с	Максимальная ошибка	Средняя ошибка
RBF		7,89	4,87	0,000911859	0,000214449
1-9-3	trainlm	731,66	2,09	0,538380957	0,090512478
	trainbr	800,45	3,22	0,221002567	0,013998969
1-15-3	trainlm	480,65	2,83	0,010466897	0,001339655
	trainbr	995,24	3,32	0,014147684	0,002023587
1-18-3	trainlm	190,77	3,33	0,011569533	0,001390274
	trainbr	140,45	4,68	0,012510434	0,001337011
1-21-3	trainlm	237,81	3,45	0,020310202	0,001154658
	trainbr	242,36	5,32	0,015167542	0,001428948
1-25-3	trainlm	139,11	4,43	0,019714370	0,001406030
	trainbr	103,28	6,80	0,021075418	0,001230780
1-30-3	trainlm	118,49	5,40	0,023770362	0,000988233
	trainbr	282,82	6,62	0,012567312	0,001163680

Наименьшую среднюю ошибку на 1000000 точек из трехслойных сетей дает сеть с 30 нейронами в скрытом слое для обоих алгоритмов обучения. Однако данная сеть имеет самое большое время работы и сравнительно большую максимальную погрешность. Наименьшую максимальную ошибку дает сеть с 15 нейронами в скрытом слое, но она

имеет большое время обучения. В среднем наиболее оптимальной является сеть с 18 нейронами в скрытом слое. К данной сети можно применять оба алгоритма обучения. В дальнейшем будет рассматриваться именно эта сеть.

Достаточно хороший результат решения системы дифференциальных уравнений получен с помощью RBF-сети. Однако, время работы в режиме тестирования RBF-сети увеличивается по сравнению с трехслойной сетью, а время обучения RBF-сети во много раз меньше.

Таким образом, для решения заданной системы дифференциальных уравнений с помощью нейросетевых технологий рекомендуется использовать трехслойной сети с 18 нейронами в скрытом слое и радиальные базисные нейронные сети.

9.6. Варьирование количества примеров обучающей выборки

Для указанных сетей варьировалось количество примеров в обучающей выборке. Значения независимой переменной t выбирались равномерно в пределах от 0 до 10 с шагом 0,1.

В результате варьирования количества примеров обучающей выборки для сети с 18 нейронами в скрытом слое и алгоритмом обучения Левенберга-Марквардта при работе сети на 1000000 точек выявлено, что с ростом количества примеров в обучающей выборке уменьшаются максимальная и средняя ошибки, но при этом растет время обучения. Наименьшее время обучения наблюдается при 100 примерах в обучающей выборке.

В аналогичном эксперименте с использованием алгоритма обучения Байеса обнаружено, что с ростом количества примеров в обучающей выборке уменьшаются максимальная и средняя ошибки, но при этом также

растет время обучения. Сеть не удалось обучить на выборке из 500 примеров. Наименьшее время обучение наблюдается при 100 примерах в обучающей выборке. Время работы при разном количестве примеров в выборке также почти одинаковое.

В результате варьирования выборки RBF-сети при работе сети на 1000000 точек можно заметить, для данного типа сети время обучения намного меньше, чем для трехслойных нейронных сетей. При этом минимальная средняя ошибка достигается при обучающей выборке из 100 примеров.

Варьирование количества примеров в тестовой выборке для трехслойной сети с алгоритмом обучения Левенберга-Марквардта показало, что время работы растёт практически линейно с ростом количества примеров в тестовой выборке. Максимальная ошибка незначительно возрастает, средняя ошибка незначительно убывает.

В результате варьирования количества примеров в тестовой выборке для сети RBF выявлено, что время работы растет практически линейно с ростом количества примеров в тестовой выборке. Максимальная ошибка и средняя ошибки практически не меняются. Время работы RBF-сети больше, чем трехслойной сети, однако, максимальная и средняя ошибки меньше.

На разном количестве точек проводилось вычисление значений решения системы дифференциальных уравнений численным методом Рунге-Кутта с автоматическим выбором шага.

Для рассмотренной системы дифференциальных уравнений ошибки, получаемые при использовании нейронных сетей, гораздо меньше, чем ошибки, получаемые численным методом.

При увеличении количества точек время работы численного метода возрастает экспоненциально и намного превышает время моделирования с помощью нейронных сетей.

Например, для 4000000 точек численный метод находит решение в 1478,5 раз медленнее, чем трехслойная нейронная сеть и в 834,2 раза медленнее, чем нейронная сеть RBF.

Время работы трёх рассмотренных выше методов нахождения решения системы дифференциальных уравнений представлены в табл. 9.3.

При анализе решения системы линейных дифференциальных уравнений аналитически, с помощью трехслойных нейронных сетей обратного распространения, с помощью радиально-базисных нейронных сетей и с помощью численного метода Рунге-Кутта с автоматическим выбором шага было выявлены следующие результаты:

– время работы численного метода растёт экспоненциально по мере роста примеров тестовой выборки;

– время работы нейронных сетей характеризуется линейной зависимостью;

– точность решения рассмотренной системы дифференциальных уравнений с помощью нейросетевых технологий выше по сравнению с численными методами.

Таблица 9.3. Варьирование количества примеров в тестовой выборке для различных методов решения

Количество примеров	Время работы трехслойной сети, с	Время работы RBF сети, с	Время работы численного метода, с
1000	0,07	0,12	0,04
100000	0,29	0,50	5,54
300000	0,74	1,60	61,83
500000	1,17	2,50	199,84
700000	1,60	3,61	415,92
900000	2,07	4,66	714,70
2000000	4,55	8,31	2820,13
4000000	10,63	18,84	15716,09

Таким образом, по затратам времени и по точности для решения рассмотренной системы дифференциальных уравнений эффективнее использовать нейронные сети, предварительно обучив их на достаточном количестве примеров (не менее 100). При этом значение независимой переменной *t* выбрано из интервала от 0 до 10.

В качестве оптимальных архитектур нейронных сетей выбраны трехслойная сеть обратного распространения с 18 нейронами в скрытом слое и алгоритмом обучения Левенберга-Марквардта и сеть RBF. Сеть RBF обучается быстрее и приводит к меньшим ошибкам, чем трехслойная сеть обратного распространения. Однако время ее тестирования больше, чем время тестирования трехслойного персептрона. Если имеются ограничения на время обучения нейронной сети, то целесообразно использовать RBF-сеть, если на время тестирования, то многослойную НС.

9.7. Моделирование системы нелинейных дифференциальных уравнений

Рассматривается система нелинейных дифференциальных уравнений

$$\begin{cases} y_1' = \cos(y_2) + \sin(y_3), \\ y_2' = y_1 \cdot y_3, \\ y_3' = \sin(-0.51 \cdot t) - y_1 \cdot y_2, \\ y_1(0) = 0, \\ y_2(0) = 1, \\ y_3(0) = -1. \end{cases}$$

Данная система не имеет аналитического решения. Поэтому обучающая выборка строится с помощью численного метода Рунге-Кутта с автоматическим выбором шага, и результаты сравниваются с результатами работы этого метода. Как и в предыдущем примере, размер обучающей

выборки 100 примеров. Значения независимой переменной *t* взяты из интервала от 0 до 10.

Для анализа была создана трехслойная сеть обратного распространения, включающая один нейрон во входном слое с сигмоидальной логистической функцией активации и три нейрона в выходном слое с линейной функцией активации.

Число нейронов в скрытом слое с сигмоидальной логистической функцией активации варьировалось с целью подбора наиболее оптимальной архитектуры сети. В качестве обучающего алгоритма был выбран алгоритм Левенберга-Марквардта.

В качестве оценки функционирования выбрана сумма квадратичных отклонений выходов сети от эталонов. При этом значение отклонения, при котором обучение считается законченным - 0,001. Максимальное количество циклов обучения - 15000.

На этой же выборке была обучена сеть с радиальными базисными функциями с точностью 0,0000001 на обучающей выборке.

Время работы и ошибки рассчитывались при работе сетей разной архитектуры на 100000 точек. Ошибки рассчитывались в сравнении с решением, полученным численным методом.

Таблица 9.4. Сравнительные характеристики для нейронных сетей

Архитектура сети	Время обучения, с	Время работы, с	Максимальная ошибка	Средняя ошибка
RBE	15,38	2,02	0,021354475	0,002782473
1-10-3	751,48	0,83	0,134491640	0,011535858
1-15-3	692,64	1,04	0,022203863	0,003520186
1-18-3	61,87	0,79	0,025334404	0,002900122
1-21-3	22,07	0,83	0,024508904	0,003088053
1-25-3	*11,22*	*0,95*	*0,021440149*	*0,003389760*
1-30-3	15,47	0,95	0,025778263	0,003609650
1-40-3	31,12	1,05	0,050063986	0,007294015

Результаты экспериментов, приведенных в табл. 9.4 показывают, что время работы трехслойной сети почти не меняется по мере роста числа нейронов в скрытом слое. При этом время обучения нейронной сети уменьшается и достигает наименьшего значение при 25 нейронах в скрытом слое. При дальнейшем увеличении числа нейронов время обучения возрастает.

По совокупности показателей в среднем наиболее оптимальной является сеть с 25 нейронами в скрытом слое.

В данном случае RBF-сеть имеет время обучения немного большее, чем время обучения трехслойной сети с 25 нейронами в скрытом слое. Время работы сети RBF при этом в два раза выше, а ошибки незначительно меньше.

Таким образом, в данном примере в качестве оптимальной архитектуры сети можно однозначно выбрать трехслойный персептрон с 25 нейронами в скрытом слое.

Следует отметить, что для численного метода Рунге-Кутта с автоматическим выбором шага время работы на 100000 точек составляет 5,62 с, что почти в шесть раз больше времени работы нейронных сетей. Таким образом, использовать нейронные сети эффективно. Можно использовать численный метод для построения обучающей выборки и, обучив на этой выборке нейронную сеть, получать результаты за более короткое время.

9.8. Моделирование системы нелинейных дифференциальных уравнений с разрывной правой частью

Рассматривается нелинейное дифференциальное уравнение с разрывной правой частью

$$\begin{cases} y_1' = -2 \cdot e^{-t}, t \in [0;5), \\ y_1' = \dfrac{0.5}{(t-7.5)^2}, t \in [5;10], \\ y_1(0) = 0, \\ y_1(5) = 0. \end{cases}$$

Аналитическое решение данного уравнения имеет вид

$$\begin{cases} y_1 = 2 \cdot e^{-t} - 2, t \in [0;5), \\ y_1 = \dfrac{-1}{(2 \cdot t - 15)} - 0.2, t \in [5;10]. \end{cases}$$

На основе этого аналитического решения была построена обучающая выборка, включающая 100 примеров. При этом значения независимой переменной *t* выбирались равномерно в пределах от 0 до 10 с шагом 0,1.

Один разрыв в данном уравнении задан явно через разрывность правой части, а второй разрыв возникает из-за гиперболы в качестве решения на втором промежутке.

Уравнение было решено численным методом Рунге-Кутта. При этом решение, полученное численным методом, и аналитическое решение практически совпадают. Но численный метод останавливается, когда достигает второго разрыва. В этом его недостаток.

Для анализа была создана трехслойная сеть обратного распространения, включающая один нейрон во входном слое с сигмоидальной логистической функцией активации и три нейрона в выходном слое с линейной функцией активации. Число нейронов в скрытом слое с сигмоидальной логистической функцией активации варьировалось с целью подбора наиболее оптимальной архитектуры сети. В качестве обучающего алгоритма был выбран алгоритм Левенберга-Марквардта.

Было произведено обучение сетей. В качестве оценки функционирования выбрана сумма квадратичных отклонений выходов сети от эталонов. При этом значение отклонения, при котором обучение

считается законченным - 0,001. Максимальное количество циклов обучения - 15000.

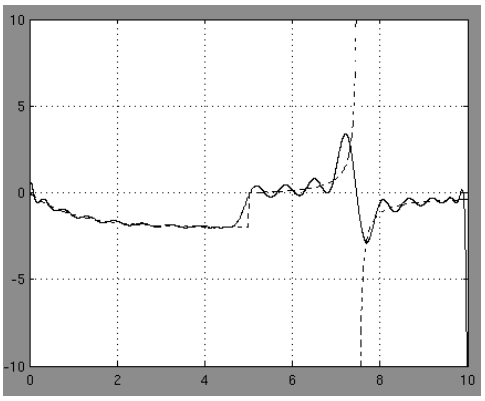
На этой же выборке была обучена сеть с радиальными базисными функциями с точностью 0,0000001 на обучающей выборке.

Время обучения RBF-сети меньше, чем для трехслойного персептрона, а время работы намного больше. Время работы трехслойного персептрона увеличивается линейно с ростом количества нейронов в скрытом слое.

На рис.9.2 изображено «гладкое» решение, без разрывов и резких скачков. На участке до первого разрыва решение аппроксимируется довольно хорошо, но на втором и третьем участках ошибки значительные.

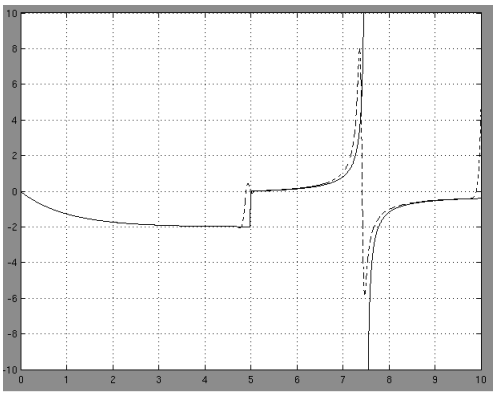
Рис. 9.2 иллюстрирует решение, полученное с помощью трехслойного персептрона с 15 нейронами в скрытом слое, без разрывов и без резких скачков. На всех трёх участках решение аппроксимируется сетью достаточно точно.

Получены достаточно точные решения с помощью трехслойного персептрона с 15, 21, 25 и 30 нейронами в скрытом слое. При этом время работы сети растет с увеличением количества нейронов в скрытом слое, а наименьшее время обучения было получено у сети с 21 нейроном в скрытом слое. Но поскольку время обучения зависит от весов синапсов, которые инициализируются случайным образом, то в данном случае лучше ориентироваться на время работы и в качестве оптимальной архитектуры выбрать трехслойный персептрон с 15 нейронами в скрытом слое.



Пунктирная линия – аналитическое решение; сплошная линия – решение с помощью сети RBF

Рис. 9.2. График решения дифференциального уравнения полученного с помощью сети RBF



Сплошная линия – аналитическое решение; пунктирная линия – решение с помощью трехслойного персептрона

Рис. 9.2. График решения, полученного с помощью трехслойного персептрона

Трехслойный персептрон может дать хорошее приближение даже в случае разрывной правой части дифференциального уравнения, однако, вблизи бесконечных разрывов ошибки могут быть значительными именно потому, что нейронные сети не могут обеспечить бесконечный разрыв.

9.9. Аппаратная поддержка моделирования системы дифференциальных уравнений на ПЛИС

Многие прикладные задачи моделирования могут быть представлены в виде системы дифференциальных уравнений, для решения которых широко применяются численные методы. В [38] описаны примеры решения системы дифференциальных уравнений с использованием нейросетевых технологий. Время решения системы дифференциальных уравнений численным методом в несколько раз превышает время решения с помощью нейронной сети. При этом средняя ошибка решения с

помощью нейронной сети на порядок меньше, чем ошибка, полученная с помощью численного метода.

В [38] приведены результаты аппаратной реализации нейронных сетей на базе ПЛИС Xilinx для экстраполяции функций.

Реализация на ПЛИС нейронной сети для решения конкретной задачи позволяет не только повысить скорость и точность обработки информации, а также открывает возможности для создания отказоустойчивых систем специального назначения за счёт некоторой избыточности нейронной сети и её обучения с учётом возможного выхода из строя элементов или обрыва цепей.

Выбранная трехслойная нейронная сеть с архитектурой (1,25,3), содержит один нейрон в первом слое, 25 нейронов во втором слое и 3 нейрона в третьем слое. Структурная схема нейрона (рис.9.3) включает блок выбора пары «вход-вес» (Xi; Wi); блок умножения, с помощью которого выполняется умножение входа нейрона на соответствующий ему вес синапса с сохранением результата в буферном регистре; сумматор, предназначенный для сложения произведений весов синапсов и входов нейрона, и сохранением взвешенной суммы в буферном регистре; нелинейный преобразователь, с выхода которого снимается выходное значение нейрона.

Рис.9.3. Структурная схема нейрона

Нейроны входного и выходного слоев могут иметь пороговую или линейную (с насыщением) функцию возбуждения. Особый интерес и сложность реализации представляет нелинейный преобразователь нейронов скрытого (второго) слоя НС и, особенно, сигмоидальной функции активации. Функция возбуждения нейронов скрытого и выходного слоёв представлена на рис.9.4.

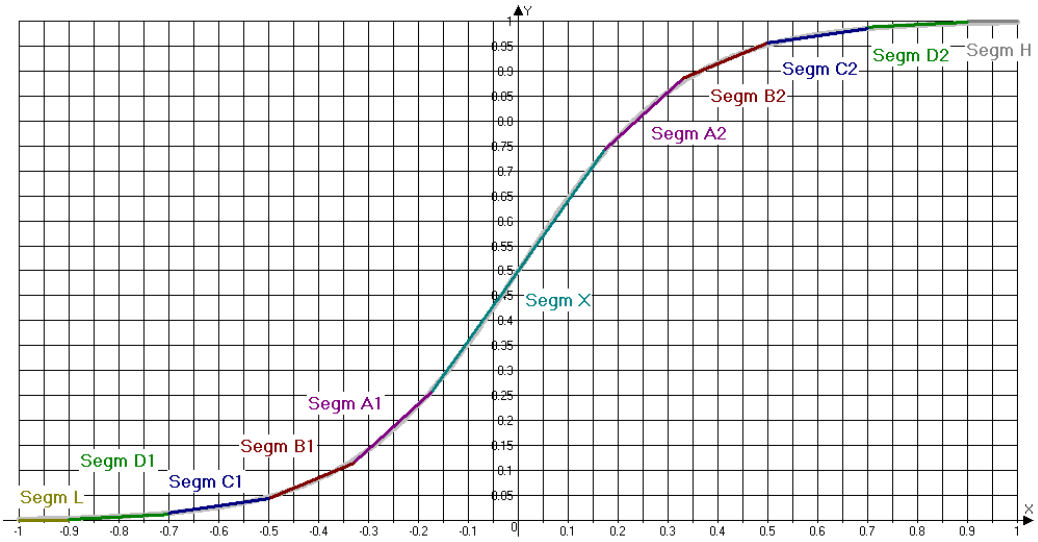


Рис.9.4. Функция возбуждения нейронов

Реализация нелинейного преобразователя возможна табличным способом или путём кусочно-линейной аппроксимации.

Табличный способ характеризуется тем, что в памяти ПЛИС должны храниться аргументы и соответствующие им значения функции. Главным преимуществом этого способа является быстродействие, за которое надо платить большими затратами памяти при заданной точности. Вторым недостатком табличного способа реализации является вероятность «попасть» между табличными значениями. В этом случае функция вычислена не будет.

В случае применения кусочно-линейной аппроксимации функция разбивается на сегменты, которые, в свою очередь, задаются линейной функцией, имеющей вид f(x)=kx+b. Значение x поступает с выхода блока сумматора произведений «вход-вес», значение k задаёт угловой коэффициент. В этом случае в памяти хранится только массив аргументов границ отрезков. Недостатком является меньшее, по сравнению с табличным методом, быстродействие при следующих основных достоинствах:

1. Малые аппаратные затраты. Для вычисления значения необходимы только сумматор и устройство умножения.

2. Непрерывность функции. Отсутствует вероятность «промахов» и возможно вычисление функции с любым шагом.

3. Потенциал точности. Относительно небольшие аппаратные затраты для значительного повышения точности аппроксимации и, следовательно, вычислений в целом.

Выбранный метод кусочно-линейной аппроксимации предполагает разбиение функции на ряд сегментов (на рис. 9.4 изображено 11 сегментов).

Основным принципом функционирования нелинейного преобразователя нейронов слоя является кусочно-линейная аппроксимация функции активации с последующей параллельной выборкой сегмента и вычислением функции в точке по линейному закону. Взвешенная сумма одновременно поступает на входы необходимого количества компараторов (для рассматриваемой реализации 10), после чего однозначно определяется, к какому сегменту принадлежит аргумент с помощью комбинационной схемы. Каждой симметричной паре сегментов соответствует свой угловой коэффициент k и каждому сегменту функции активации соответствуют свои линейные коэффициенты b. Определив, к какому сегменту относится аргумент (взвешенная сумма), вычисление функции сводится к двум действиям: умножению аргумента на угловой коэффициент с помощью устройства умножения и сложению получившегося результата с линейным коэффициентом b посредством сумматора.

Функционирование нейрона выполняется следующим образом. Выбирается пара синапс-вход (Wi; Xi) и перемножаются между собой на устройстве умножения. Взвешенная сумма накапливается в регистре и на каждой итерации цикла складывается с новым произведением Wi*Xi. Как

только все пары «вход-вес» будут перебраны, результирующая взвешенная сумма, являясь аргументом функции активации, анализируется на необходимость вычисления функции. Необходимость в вычислениях пропадает, если взвешенная сумма принадлежит одному из двух крайних сегментов функции, где она (функция) будет равна, либо нулю в этом случае алгоритм сразу же завершается, поскольку регистр был сброшен в состояние нуля на первом шаге алгоритма, либо единице соответственно. Если же необходимость в вычислении функции все же присутствует, то необходимые действия производятся за два такта с помощью устройства умножения и сумматора. Результат работы нейрона записывается в выходной регистр.

Повышение точности вычислений возможно посредством уменьшения длины и увеличения количества отрезков, на которые разбивается функция активации нейрона. Таким образом, точность вычислений напрямую зависит от точности аппроксимации.

Увеличение разрядности регистра взвешенной суммы повышает точность аппроксимации, но приводит к увеличению аппаратных затрат при реализации на ПЛИС.

Значительный прирост быстродействия, возможно, получить, реализовав параллельное перемножение весов на входы нейронов. Это позволит ускорить работу этих нейронов в несколько раз, но приведёт к увеличению аппаратных затрат. Вычисление взвешенной суммы остается последовательным процессом, поэтому для ощутимого ускорения можно конвейеризовать расчёт взвешенной суммы и процедуру перемножения весов на входные сигналы.

Возможно использование отдельных устройств перемножения и сложения для вычисления функции на каждом участке. В этом случае удастся избежать задержки на мультиплексорах (выбора линейного и углового коэффициента попросту не будет, выбирать нужно будет только

источник результата), но потребуются дополнительные управляющие сигналы для выбора источника результата.

При реализации прототипа нейронной сети на базе ПЛИС необходимо выполнить два важных требования: высокая тактовая частота и достаточное количество логических элементов для реализации нейронной сети.

Для решения системы дифференциальных уравнений [44] путём аппаратной реализации прототипа трёхслойной нейронной сети [1,25,3] был выбран кристалл Altera Cyclone III EP3C120, содержащий около 120 000 логических элементов и способный устойчиво работать на частоте до 400 МГц.

Моделирование схемы ПЛИС проводится на базе системы автоматизированного проектирования ПЛИС Altera Cyclone III Quartus II.

Интересная разработка CM1K представлена на сайте www.cognimam.com. CM1K - первая версия ASIC нейронной сети CogniMem, содержащей 1024 нейрона работающих параллельно и способных к распознаванию образцов до 256 байтов за несколько микросекунд.

10. ПРИМЕРЫ РЕАЛИЗАЦИИ ПАКЕТОВ НЕЙРОСЕТЕВЫХ ПРОГРАММ

10.1 Анализ преимуществ коммерческих нейросетевых программ

Несмотря на значительное число специализированных аппаратных разработок нейросетей, основное применение в настоящее время получили программные реализации различных нейросетевых парадигм [4,10,11,12,20,25-28], называемые нейропакетами (нейроэмуляторами). Иначе говоря, нейропакет - это программная оболочка, эмулирующая для пользователя нейросреду на обычном персональном компьютере [23-28].

Преимущества таких "виртуальных" нейрокомпьютеров для относительно небольших задач очевидны:

– во-первых, не надо тратиться на новую аппаратуру, если можно загрузить уже имеющиеся компьютеры общего назначения;

– во-вторых, пользователь не должен осваивать особенности программирования на специализированных процессорах и способы их сопряжения с базовым компьютером;

– универсальные ЭВМ не накладывают никаких ограничений на структуру сетей и способы их обучения, тогда как специализированные процессоры зачастую имеют ограниченный набор "зашитых" в них функций активации и достигают пиковой производительности лишь на определенном круге задач.

Таблица10. 1. Секторы рынка нейросетевых программных продуктов

Сегмент рынка нейронных продуктов	Преимущества	Недостатки
Нейронные пакеты общего назначения	Не требуют самостоятельного программирования, легко осваиваются, пригодны для быстрого и дешёвого решения прикладных задач	Не способны к расширению, не могут использоваться для разработки сложных систем или их подсистем
Системы разработки нейронных приложений	Могут использоваться для создания сложных систем обработки данных в реальном времени	Требуют навыков программирования, более глубокого знания нейросетей
Готовые решения на основе нейросетей	Предоставляют комплексное решение проблемы	Высокая стоимость

10.2. Универсальный нейропакет *NeuroSolutions*

Универсальный нейропакет *NeuroSolutions* фирмы NeuroDimension, Inc. предназначен для моделирования широкого круга искусственных нейронных сетей. Основное достоинство описываемого нейропакета состоит в его гибкости: помимо традиционно используемых нейросетевых парадигм (типа полносвязных многослойных нейронных сетей или

самоорганизующихся карт Кохонена) нейропакет включает в себя мощный редактор визуального проектирования нейронной сети, позволяющий создавать практически любые собственные нейронные структуры и алгоритмы их обучения. Особо следует отметить, что данный нейропакет позволяет пользователю вводить собственные критерии обучения нейронной сети, не ограничивая его только широко распространённым, но далеко не самым оптимальным критерием минимума среднеквадратичной ошибки.

Нейропакет NeuroSolutions снабжен мощными и хорошо продуманными средствами визуализации: отображать и визуально контролировать можно многое, начиная от структуры нейронной сети и кончая процессом и результатом обучения. Наличие мощных средств визуализации выводит нейропакет на уровень CAD-систем (систем автоматизированного проектирования), т. е. NeuroSolutions можно считать системой проектирования и моделирования нейронной сети.

10.3. NeuralWorks Professional II/Plus

В отличие от NeuroSolutions в пакете *NeuralWorks Professional II/Plus* (фирма NeuralWare, Inc.) основной упор сделан на применение стандартных нейронных парадигм и алгоритмов обучения и в этом данный пакет превосходит все остальные. В нем реализованы 28 стандартных нейронных парадигм, используемых при решении прикладных задач.

Также как и NeuroSolutions, NeuralWorks Professional имеет хорошо организованную систему визуализации данных. Можно просмотреть структуру нейронной сети, изменение весовых коэффициентов в процессе обучения, изменение ошибки обучения, а также корреляцию весов нейронной сети при обучении. Последнее является уникальной возможностью, представляемой только пакетом NeuralWorks Professional и

весьма полезной при анализе поведения нейронной сети при обучении и работе.

Также как и NeuroSolutions, NeuralWorks Professional представляет собой открытую систему, в которую можно интегрировать внешние программные модули, написанные пользователями. Пакет имеет встроенный генератор кода, поддерживающий компилятор Microsoft Visual C++.

10.4. Нейропакет Process Advisor

Нейропакет *Process Advisor* (фирма Al Ware, Inc) создавался для решения задач управления динамическими процессами (в частности, технологическими). Однако разработчикам удалось создать программу, которая может считаться универсальным нейропакетом. В нейропакете реализована только многослойная нейронная сеть прямого распространения, обучаемая с помощью модифицированного алгоритма обратного распространения ошибки (backpropagation error). Введены возможность работы с динамическими процессами. В частности, в Process Advisor возможна работа с входными сигналами как с функциями времени, а не только как с дискретным набором точек. Помимо Process Advisor, такую возможность предоставляет только пакет NeuroSolutions. Кроме того нейропакет Process Advisor позволяет осуществлять управление внешними аппаратными контроллерами, подключаемыми к компьютеру.

10.5. Нейропакет NeuroShell 2

Нейропакет *NeuroShell 2* (фирма Ward Systems Group) является одной из трех программ, входящих в состав пакета The AI Trilogy. Он

представляет собой универсальный нейропакет, предназначенный для моделирования нескольких наиболее известных нейронных парадигм: многослойных нейронных сетей, сети Кохонена и других. Пакет NeuroShell 2 сильно проигрывает по сравнению с NeuroSolutions и NeuralWorks.

Система для профессионала позволяет опытным пользователям создавать 16 различных видов нейросетей со значительно большими возможностями установки и контроля их параметров, чем в Системе для начинающего.

Средства автономного использования, входящие в состав пакета NeuroShell 2, позволяют пользователю использовать свою созданную нейронную сеть как динамическую библиотеку (DLL), которая может быть вызвана из других программ или из Microsoft Excel. Вы можете также осуществить генерацию программного кода на Си или Visual Basic для созданных Вами сетей. Нейросети, которые Вы создали с помощью NeuroShell 2, Вы можете распространять без каких-либо ограничений и уплаты роялти. Для использования нейронной сети вне NeuroShell 2 используется модуль Генератор автономных файлов.

Нейропакет NeuroShell 2 имеет и недостаточно продуманную систему визуализации данных: контролировать можно многие параметры, но в разных режимах работы нейропакета. Из-за отсутствия единого интегрального контроля данных в процессе обучения или работы нейронной сети часто приходится переключаться из одного режима в другой, что отнимает много времени и весьма неудобно в использовании.

К особенностям нейропакета следует отнести жестко реализованную последовательность действий при работе с нейронной сетью. Так, невозможно определить структуру нейронной сети до того, как заданы входные данные. С одной стороны, это очень удобно, особенно для начинающих пользователей, поскольку сразу становится ясно, что и в какой последовательности следует делать. С другой стороны, более

опытного пользователя такая жесткая зафиксированная последовательность действий утомляет: для того, чтобы внести в нейронную сеть небольшое изменение, приходится выполнять всю цепочку действий. Таким образом, можно сказать, что пакет NeuroShell 2 удобен для начинающих пользователей.

Обычно процесс анализа данных начинается с подготовки данных. Пользователь может ввести данные вручную или импортировать данные из файлов.

Обзор нейросетевых архитектур. Реализованные нейросетевые архитектуры (классификация нейронных сетей) приведены на рис. 10.1

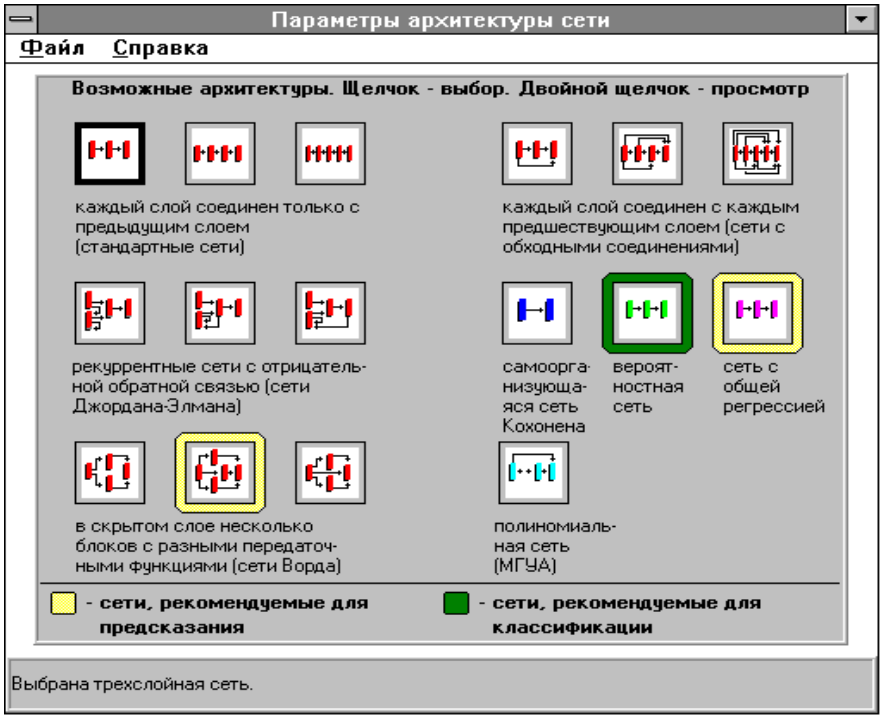


Рис.10.1 Выбор типа архитектуры в программе NeuroShell 2

Модуль "Проектирование" NeuroShell 2 предлагает на выбор 16 различных нейросетей, приведённых на рис.10.1. Для каждой из них возможны несколько методов обучения. Чтобы установить параметры нейросети, Вам нужно просто указать курсором на блок или связь и щелкнуть мышкой.

В NeuroShell 2 можно использовать сети, реализующие следующие нейросетевые парадигмы: сети с обратным распространением ошибки; сети Кохонена; вероятностные нейронные сети; нейронные сети с общей регрессией; полиномиальные сети.

Список сокращений и обозначений

АРТ - адаптивная резонансная теория

ДАП-двунаправленная ассоциативная память

ИНС-искусственная нейронная сеть

НК- нейрокомпьютер

НС-нейронная сеть

ОРО-обратное распространение ошибки

ПЛИС- программируемая логическая интегральная схема

СБИС- сверхбольшая интегральная схема

ЦПС -цифровой сигнальный процессор

Библиографический список

1. Хайкин, Саймон. Нейронные сети: полный курс, 2е издание [Текст].: Пер. с англ. М. Издательский дом "Вильямс", 2006. 1104 с. : ил.
2. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика[Текст]. - 1-е. - М.: Горячая линия - Телеком, 2001. - С. 382.
3. Осовский, С. Нейронные сети для обработки информации [Текст] / С. Осовский – М.: Финансы и статистика, 2002. – 344с.
4. Комарцова, Людмила Георгиевна. Нейрокомпьютеры[Текст]: Учеб. пос. / Комарцова, Людмила Георгиевна, Максимов, Александр Викторович. - М.: Изд-во МГТУ им. Н. Э. Баумана, 2004. - 320с. - (Информатика в техническом университете).
5. Головко В. А. Нейронные сети: обучение, организация и применение [Текст]. Кн. 4: Учеб. Пособие для вузов М.: ИПРЖР, 2001. - 256с.
6. Круглов В. В. Нечеткая логика и искусственные нейронные сети [Текст]/ В. В. Круглов, М. И. Дли, Р. Ю. Голунов, Физматлит 2001. - 224 с.
7. Ростовцев В.С., Исупов К.С. Свидетельство о государственной регистрации программы для ЭВМ №2010610489. Система моделирования многослойной нейронной сети. Зарег, в государственном Реестре программ 11 января 2010г.
8. Ростовцев В.С., Новокшонов Е.В. Многопрофильная модель релаксационной нейронной сети. Свидетельство о регистрации программы для ЭВМ №2010614670 от 16 июля 2010 года.
9. Ярушкина, Н. Г. Основы теории нечётких и гибридных систем [Текст]: учеб, пособие/Н. Г. Ярушкина. - М.: Финансы и статистика, 2004. – 320 с.: ил.
10. Яхъяева, Г. Э. Нечеткие множества и нейронные сети [Текст] / Г.Э. Яхъяева; Лаб. знаний, Интернет-ун-т информ. технологий - ИНТУИТ.ру. – М.: БИНОМ, 2006.
11. Комашинский, В. И. Нейронные сети и их применение в системах управления и связи [Текст]/ Комашинский, В. И., Смирнов, Д. А. - М.: Горячая линия-Телеком, 2003. - 94с. - Библиогр.: с. 88.
12. Нейронные сети: история развития теории [Текст]: Учеб. пос. Кн. 5 / Под ред. А. И. Галушкина, Я. З. Цыпкина. - М.: ИПРЖР, 2001. - 840с.: ил. - (Нейрокомпьютеры и их применение)

13. Нейрокомпьютеры и их применение [Текст]. Кн.9: Нейрокомпьютеры в системах обработки сигналов / Под ред. Ю. В. Гуляева, А. И. Галушкина. - М.: Радиотехника, 2003. - 224с.: ил. - Библиогр.: с. 190.
14. Нейрокомпьютеры и их применение [Текст]. Кн.7: Нейрокомпьютеры в системах обработки изображений / Под ред. Ю. В. Гуляева, А. И. Галушкина. - М.: Радиотехника, 2003. - 192с.: ил. - Библиогр.: с. 190.
15. Ростовцев В.С., [Рапопорт А.Н.], Куимов В.И. Свидетельство о государственной регистрации программы для ЭВМ № 2010614222. Модель нейронной сети встречного распространения. Зарег, в государственном Реестре программ 11 января 2010г.
16. Ростовцев В.С., Лукин М.А. Свидетельство о государственной регистрации программы для ЭВМ №2009615454. Программа моделирования нейронных сетей адаптивной резонансной теории. Зарег, в государственном Реестре программ 1 октября 2009г.
17. Смоленцев, Н. К. Основы теории вейвлетов. Вейвлеты в MATLAB[Текст]: учеб. пособие / Н. К. Смоленцев. - 2-е изд., перераб. и доп. - М.: ДМК Пресс, 2005. - 304 с.: ил. - Библиогр.: с. 302
18. Поршнев, Сергей Владимирович. Компьютерное моделирование физических процессов в пакете MATLAB[Текст]: учеб. пособие / С. В. Поршнев. - 2-е изд., испр. - СПб. М.; Краснодар: Лань, 2011. - 726 с. + 1 эл. опт. диск (CD-ROM).
19. Солонина, А. И. Цифровая обработка сигналов. Моделирование в MATLAB[Текст] / А. И. Солонина. - СПб. [б. и.], 2008. - 806 с.

Интернет-источники

20. Тора-Центр. Аналитические программы, решения и технологии. Режим доступа: http://www.tora-centre.ru
21. MATLAB.Exponenta.Ru. Spline Toolbox. Обзор средств MATLAB и ToolBox'ов для приближения данных[Электронный ресурс].- Режим доступа: http://MATLAB.exponenta.ru/spline/book1/6.php
22. Сайт НТЦ «Модуль». Режим доступа: www.module.ru.
23. Генетические алгоритм. Нечеткая логика. Нейронные сети. Режим доступа: http://ami.nstu.ru/~vms/lecture/lecture16/lek_bd2001_72web.htm
24. Справочник по нейронным сетям. Режим доступа: http://www.neuroshell.forekc.ru/

25. А.А.Ежов, С.А.Шумской. Нейрокомпьютинг и его применение в экономике и бизнесе, [Электронный ресурс] - Режим доступа:http:\\ www.neuroproject.ru
26. Обзор нейропакетов. Режим доступа: http://www.neuroproject.ru/aboutproduct.php?info=nstinfo-forecasts
27. Инструментальные средства разработки нейросетевых приложений. Режим доступа: http://www.madi.ru/study/kafedra/asu/metod/nero/5_2.shtml
28. С. А. Терехов Лекции по теории и приложениям искусственных нейронных сетей (электронная версия). Лабораторотория Искусственных Нейронных Сетей НТО-2, ВНИИТФ, Снежинск, 1998. http://alife.narod.ru/lectures/neural/Neu_ch04.htm