Министерство общего и профессионального образования
Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Зайцев Сергей Анатольевич

Тема: Применение нейронных сетей для анализа данных.

Утверждена приказом по университету № 2016/2 от 30 августа 2001 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению высшего профессионального образования
552800 “Информатика и вычислительная техника”

Факультет Автоматики и Вычислительной Техники

Руководитель:

Гаврилов А.В.,
к.т.н., доц. каф. ВТ НГТУ

Новосибирск, 2003 г.

Министерство общего и профессионального образования
Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
д.т.н., профессор
Губарев В.В.

_____

“__”_____2003 г.

ЗАДАНИЕ

на магистерскую диссертацию

студенту *Зайцеву С. А.* факультета Автоматики и Вычислительной Техники, обучающемуся по направлению 552800 “Информатика и вычислительная техника”

Магистерская программа (специализация) 552819, “ Компьютерный анализ и интерпретация данных ”

Тема: Применение нейронных сетей для анализа данных.

Цель работы. Разработка архитектуры программного обеспечения, позволяющего использовать нейронные сети для обработки данных, а также для изучения, исследования и разработки моделей искусственных нейронных сетей, в том числе, при использовании в учебном процессе.

Руководитель МД:

Гаврилов А.В.,
к.т.н., доц. каф. ВТ НГТУ

_____
(подпись)

“__”_____2003г.

Содержание

Введение

Математический аппарат искусственных нейронных сетей был разработан достаточно давно, но широкое практическое применение его для решения прикладных задач началось сравнительно недавно. В настоящее время происходит существенное повышение интереса к искусственному интеллекту, вызванного, как развитием технических средств, так и потребностью рынка программного обеспечения в качественно новом продукте. На фоне этого процесса, а вернее, как неотъемлемая часть его, производятся многочисленные попытки применения тех или иных моделей нейронных сетей для решения различных задач. Искусственные нейронные сети получают все большее распространение за счет следующих факторов:

- искусственные нейронные сети способны решать трудно формализуемые (или не формализуемые задачи);
- искусственным нейронным сетям присущ параллельный принцип работы, что очень важно при обработке больших объемов данных, особенно мультимедийных (изображение, звук, видео);
- исследования в области искусственных нейронных сетей в восьмидесятые годы прошлого века дали успешные результаты, существенно расширившие область применения нейронных сетей.

Целью работы является разработка архитектуры программного обеспечения, позволяющего использовать нейронные сети для обработки данных, а также для изучения, исследования и разработки моделей искусственных нейронных сетей, в том числе, при использовании в учебном процессе. В соответствии со сформулированной целью была поставлена задача по созданию программы, к которой предъявлялись следующие требования.

- Программа должна позволять проводить различные виды анализа данных: прогноз, ассоциативный поиск, кластеризацию и распознавание.
- Исходные данные для анализа нужно извлекать из баз данных, как из одного из самых универсальных источников данных.
- Программа должна позволять оценить качество проведенного анализа.
- Программа должна поддерживать работу с различными моделями нейронных сетей.

В результате работы была создана программа для анализа баз данных с помощью нейронных сетей, позволяющая использовать различные модели нейронных сетей для обработки информации, извлекаемой из базы данных. Программа поддерживает четыре вида обработки данных: прогноз,

ассоциативный поиск, кластеризацию и распознавание. Однако помимо обработки данных программа может использоваться для исследования и разработки моделей искусственных нейронных сетей. Это было достигнуто за счет возможности расширения приложения новыми моделями нейронных сетей (возможно, разработанных пользователем), а также за счет возможности провести детальную настройку той или иной модели сети и оценить результат ее работы.

Как уже отмечалось, в настоящее время существует большое количество приложений, использующих искусственные нейронные сети. Но существующие приложения либо ориентированы на решение конкретных задач (распознавание речи, прогноз курса ценных бумаг …) с помощью конкретной модели нейронной сети (или набора сетей) и не позволяют использовать другие модели, либо предназначены для изучения нейронных сетей, но не для их практического применения. Приложения первого вида нельзя использовать для разработки новых моделей и алгоритмов обучения нейронных сетей – они не могут служить полноценным инструментом исследователя и разработчика нейронных сетей. Программы второго вида, наоборот, предназначены для исследования нейронных сетей и ориентированы на специалиста в области нейроинформатики. Такие программы сложны и мало пригодны для использования нейронных сетей для анализа данных конечным пользователем. Новизна разработанной мной программы заключается в объединении в ней возможности анализа данных неспециалистом и возможности исследования и разработки моделей нейронных сетей специалистом.

Для успешного применения искусственных нейронных сетей недостаточно только программной реализации модели сети. Также необходимо правильно подобрать ее параметры. Эта задача является нетривиальной и, порой, более сложной, чем реализация модели нейронной сети. В данной работе было проведено исследование одной из самых распространенных моделей нейронных сетей – многослойного перцептрона. Исследование было направлено на поиск правил выбора параметров модели для решения задачи классификации. В результате исследования были предложены рекомендации по выбору архитектуры перцептрона с пороговыми функциями активации, предназначенного для решения задачи классификации. В качестве метода исследования был выбран анализ математической модели этой сети.

Отправной точкой для создания приложения послужила программа AnalDB, разработанная на кафедре вычислительной техники факультета АВТФ НГТУ. Программа могла работать только с одной моделью нейронной сети (сетью Хопфилда). Мой авторский вклад заключается в следующем. В ходе работы программа была почти полностью переработана. Были разработаны принципы и алгоритмы программного интерфейса между оболочкой программы и моделью нейронной сети, позволяющие сравнительно легко подключать к оболочке новые модели нейронных сетей, полностью изменен

пользовательский интерфейс приложения, также было создано два модуля, реализующие модели сети Хопфилда и многослойный перцептрон. Новая версия программы была использована для проведения эксперимента по прогнозу притока реки Обь. Эксперимент показал, что программа может достаточно хорошо решать задачи прогноза, а также позволил выявить особенности применения сети Хопфилда и многослойного перцептрона.

Об основных положениях и результатах диссертационной работы были сделаны доклады на научном семинаре кафедры ВТ НГТУ "Интеллектуальные системы", на конференции “Дни науки НГТУ” (г. Новосибирск, 2001, 2002, 2003), на конкурсе грандов НГТУ (г. Новосибирск, 2001, 2002; оба раза были получены гранды), “всероссийском конкурсе программных продуктов, разработанных студентами - 2003” (г. Иркутск), на Международной конференции ИСТ-2003 (г. Новосибирск).

По теме диссертации было опубликовано две печатные работы:

1) Гаврилов А.В., Канглер В.М., Зайцев С.А. Программа анализа баз данных с помощью нейронных сетей // НАУЧНАЯ СЕССИЯ МИФИ – 2003. V Всероссийская научно-техническая конференция «Нейроинформатика-2003»: Сборник научных трудов. В 2-х частях. Ч.2. М.: МИФИ, 2003. – 236 с.
2) A.V. Gavrilov, V.M. Kangler, S.A. Zaitsev. DATA ANALYSIS PROGRAM BY MEANS OF NEURAL NETWORKS// Международная научно-техническая конференция “ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ”: Материалы конференции. В 3-х частях. Ч.3. – Новосибирск: Изд-во НГТУ, 2003. – 192 с.

В первой главе рассматривается история развития нейровычислений. Приводятся описания основных моделей нейронных сетей, их классификация, способы обучения и области применения.

Во второй главе описывается программа для анализа баз данных с помощью нейронных сетей. Приводится описание принципов ее работы и способа ее применения.

Третья глава содержит результаты исследования многослойного перцептрона. Исследуется зависимость между архитектурой сети и задачей, которая может быть представлена с помощью этой сети. Рассматривается применение перцептрона с пороговой функцией активации для решения задачи классификации. Результатом исследования являются рекомендации по выбору архитектуры сети.

В четвертой главе приводится результат применения программы для анализа баз данных с помощью нейронных сетей для прогноза притока реки Обь. Для прогноза используется многослойный перцептрон и сеть Хопфилда.

В заключении приводятся результаты и выводы, полученные в ходе работы.

Работа представлена на 69 страницах основного текста, содержит 32 иллюстрации и 5 таблиц.

1. Обзор состояния и развития нейровычислений

1.1 История развития нейровычислений

Идеи нейровычислений появились в первой половине 20 века практически одновременно с возникновением последовательных ЭВМ. Основы теории нейронных сетей были заложены в работе Мак Каллока и Питтса [1], которая появилась в 1943 году. Однако, в связи с тем, что нейровычисления требуют достаточно большой вычислительной мощности, которой не обладали машины 40-х годов, первые образцы нейрокомпьютеров появились только в 50-е годы.

Первый экспериментальный нейрокомпьютер Snark был построен Марвином Минским в 1951 году. Эта машина, не была способна решать практически интересные задачи, и первый успех нейровычислений связывают с разработкой Фрэнка Розенблатта - перцептроном (от английского perception - восприятие). Персептрон был впервые смоделирован на универсальной ЭВМ IBM-704 в 1958 году, его обучение требовало около получаса машинного времени. В 1960 году был построен аппаратный вариант - Mark I Perceptron. Он был предназначен для распознавания зрительных образов. Рецепторное поле сети состояло из 400 пикселей (матрица фотоприемников 20x20), и перцептрон успешно справлялся с решением ряда задач - мог различать некоторые буквы. Доказательство теоремы обучения персептрона [2] показало, что персептрон способен научиться всему, что он способен представлять. Эта работа усилила интерес к искусственным нейронным сетям.

Однако возможности первых персептронов были весьма ограничены. В 1969 году Минский в соавторстве с Пейпертом дал математическое обоснование принципиальной ограниченности персептронов [3]. Эта работа стала причиной начала охлаждения интереса научных кругов к нейровычислениям. Исследования в этом направлении были свернуты вплоть до 1983 года, однако несколько ученых, (Кохонен, Гроссберг, Андерсон…) продолжили исследования. Наряду с плохим финансированием и недостаточной оценкой ряд исследователей испытывал затруднения с публикациями. Поэтому исследования, опубликованные в семидесятые и начале восьмидесятых годов, размещены в массе различных журналов, некоторые из которых малоизвестны. Постепенно появился теоретический фундамент, на основе которого конструируются наиболее мощные современные многослойные сети.

В 1983 году исследования в области нейронных сетей привлекли внимание Агентства Перспективных Военных Исследований США, DARPA. Были выделены средства для проведения исследований в этой области. Это стало переломным моментом в истории нейровычислений. Интерес широкой научной общественности к нейронным сетям пробудился в начале 80-х годов

после теоретических работ физика Джона Хопфилда [4]. Он и его многочисленные последователи обогатили теорию параллельных вычислений многими идеями из арсенала физики, такими как коллективные взаимодействия нейронов, энергия сети, температура обучения и т.д.

Однако, настоящий бум практических применений нейронных сетей начался после публикации Румельхартом с соавторами метода обучения многослойного персептрона, названного ими методом обратного распространения ошибки [5]. Этот алгоритм впервые был предложен Вербосом [6]. Ограничения персептронов, о которых писали Минский и Пейперт, оказались преодолимыми, а возможности вычислительной техники - достаточными для решения широкого круга прикладных задач.

В настоящее время нейронные сети получили широкое практическое применение в самых различных областях. Приведем несколько примеров:

Автопилотируемый гиперзвуковой самолет-разведчик. Названный LoFLYTE (Low-Observable Flight Test Experiment) реактивный беспилотный самолет длиной 2,5 м был разработан для NASA и Air Force фирмой Accurate Automation Corp., Chattanooga, TN в рамках программы поддержки малого инновационного бизнеса. Это экспериментальная разработка для исследования новых принципов пилотирования, включая нейронные сети, позволяющие автопилоту обучаться, копируя приемы пилотирования летчика. Со временем нейросети перенимают опыт управления, а скорость обработки информации позволит быстро находить выход в экстремальных и аварийных ситуациях. LoFLYTE предназначен для полетов со скоростью 4-5 Махов, когда скорости реакции пилота может не хватить для адекватного реагирования на изменения режима полета.

Нейросети на финансовых рынках. Американский Citibank использует нейросетевые предсказания с 1990 года. В 1992 году, по свидетельству журнала The Economist, автоматический дилинг показывал доходность 25% годовых, что намного превышает показатели большинства брокеров. Chemical Bank использует нейро-систему фирмы Neural Data для предварительной обработки транзакций на валютных биржах 23 стран, фильтруя “подозрительные” сделки. Fidelity of Boston использует нейросети при управлении портфелями с суммарным объемом $3 миллиарда. Полностью автоматизированные системы ведения портфелей с использованием нейросетей применяют, например, Deere & Co - на сумму $100 млн и LBS Capital - на сумму $400 млн. В последнем случае экспертная система объединяется с примерно 900 нейросетями. Труды лишь одного семинара “Искусственный интеллект на Уолл-стрит” составляют шесть увесистых томов.

Распознавание краденных кредитных карт. В 1986 году известный конструктор нейрокомпьютеров профессор Роберт Хехт-Нильсен основал компанию HNC. Переключившись в 1990 году с призводства

нейрокомпьютеров на предоставление конкретных решений в различных областях, HNC Software Corp. является сейчас лидером на рынке контроля транзакций по пластиковым картам. Ее основной продукт Falcon (Сокол), выпущенный в сентябре 1992 г., контролирует сейчас более 220 млн. карточных счетов, выявляя и предотвращая в реальном времени подозрительные сделки по, возможно, краденным кредитным/дебетным картам. Искусственные нейронные сети обучаются типичному поведению клиентов, различая резкую смену характера покупок, сигнализирующую о возможной краже. Ежегодные потери крупных банков от подобных краж измеряются десятками миллионов долларов, и когда в 1994 г. впервые за всю историю пластиковых карт эти потери пошли на убыль, этот прогресс пресса связывала с успешным внедрением системы Falcon. Клиентами HNC Software являются 16 из 25 крупнейших в мире эмитентов пластиковых карт.

1.2 Основные свойства искусственных нейронных сетей

Искусственные нейронные сети представляют большой практический интерес вследствие того, что они обладают следующими свойствами:

1. Способность обучаться. Нейронные сети не программируются, а обучаются на примерах. После предъявления входных сигналов (возможно, вместе с требуемыми выходами) сеть настраивают свои параметры таким образом, чтобы обеспечивать требуемую реакцию.

2. Обобщение. Отклик сети после обучения может быть до некоторой степени нечувствителен к небольшим изменениям входных сигналов. Эта внутренне присущая способность “видеть” образ сквозь шум и искажения очень важна для распознавания образов. Важно отметить, что искусственная нейронная сеть делает обобщения автоматически благодаря своей структуре, а не с помощью использования “человеческого интеллекта” в форме специально написанных компьютерных программ.

3. Параллелизм. Информация в сети обрабатывается параллельно, что позволяет достаточно выполнять сложную обработку данных с помощью большого числа простых устройств.

4. Высокая надежность. Сеть может правильно функционировать даже при выходе из строя части нейронов, за счет того, что вычисления производятся локально и параллельно.

Последние два свойства (параллелизм и надежность) характерны только для аппаратных реализаций нейронных сетей.

1.3 Основные положения искусственных нейронных сетей

1.3.1 Модель формального нейрона

Первой моделью нейрона была модель МакКаллока и Питтса [1]. Авторы предложили использовать бинарный пороговый элемент в качестве модели искусственного нейрона. Этот математический нейрон вычисляет взвешенную сумму n входных сигналов xj, j = 1, 2... n, и формирует на выходе сигнал величины 1, если эта сумма превышает определенный порог u, и 0 - в противном случае.

В настоящее время под формальным нейроном понимается модель более общего вида. Типичный формальный нейрон производит простейшую операцию - взвешивает значения своих входов со своими же локально хранимыми весами и производит над их суммой нелинейное преобразование: $y = f(u), \quad u = w_0 + \sum_i w_i x_i$

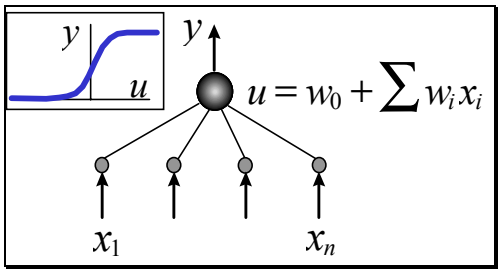

Рис.1.1. Модель формального нейрона

Нелинейность выходной функции активации $f(\cdot)$ принципиальна. Если бы нейроны были линейными элементами, то любая последовательность нейронов также производила бы линейное преобразование, и вся нейронная сеть была бы эквивалентна одному нейрону (или одному слою нейронов - в случае нескольких выходов). Нелинейность разрушает линейную суперпозицию и приводит к тому, что возможности нейронной сети существенно выше возможностей отдельных нейронов.

1.3.2 Классификация искусственных нейронных сетей по их архитектуре

Искусственная нейронная сеть может рассматриваться как направленный граф со взвешенными связями, в котором искусственные нейроны являются узлами. По архитектуре связей искусственные нейронные сети могут быть сгруппированы в два класса (рис.1.2): сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями.

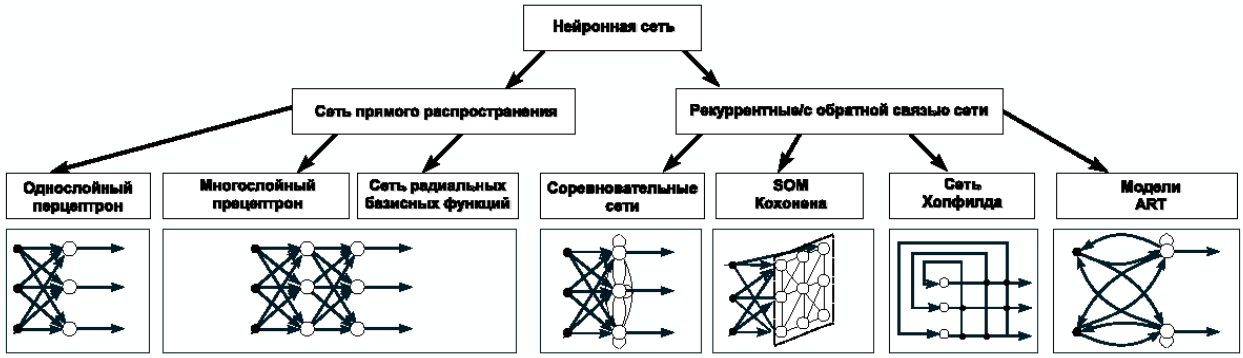

Рис. 1.2. Систематизация архитектур сетей прямого распространения и рекуррентных (с обратной связью)

В наиболее распространенном семействе сетей первого класса, называемых многослойным перцептроном, нейроны расположены слоями и имеют однонаправленные связи между слоями. На рис.1.2 представлены типовые сети каждого класса. Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети. Рекуррентные сети являются динамическими, так как в силу обратных связей в них модифицируются входы нейронов, что приводит к изменению состояния сети.

1.3.3 Обучение искусственных нейронных сетей

Способность к обучению является фундаментальным свойством мозга. В контексте искусственных нейронных сетей процесс обучения может рассматриваться как настройка архитектуры сети и весов связей для эффективного выполнения поставленной задачи. Обычно нейронная сеть должна настроить веса связей по имеющейся обучающей выборке. Функционирование сети улучшается по мере итеративной настройки весовых коэффициентов.

Для конструирования процесса обучения, прежде всего, необходимо иметь модель внешней среды, в которой функционирует нейронная сеть - знать доступную для сети информацию. Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети - какие правила обучения управляют процессом настройки. Алгоритм обучения означает процедуру, в которой используются правила обучения для настройки весов.

Существуют три парадигмы обучения: "с учителем", "без учителя" (самообучение) и смешанная. В первом случае нейронная сеть располагает правильными ответами (выходами сети) на каждый входной пример. Веса настраиваются так, чтобы сеть производила ответы как можно более близкие к известным правильным ответам. Усиленный вариант обучения с учителем

предполагает, что известна только критическая оценка правильности выхода нейронной сети, но не сами правильные значения выхода. Обучение без учителя не требует знания правильных ответов на каждый пример обучающей выборки. В этом случае раскрывается внутренняя структура данных или корреляции между образцами в системе данных, что позволяет распределить образцы по категориям. При смешанном обучении часть весов определяется посредством обучения с учителем, в то время как остальная получается с помощью самообучения.

Теория обучения рассматривает три фундаментальных свойства, связанных с обучением по примерам: емкость, сложность образцов и вычислительная сложность. Под емкостью понимается, сколько образцов может запомнить сеть, и какие функции и границы принятия решений могут быть на ней сформированы. Сложность образцов определяет число обучающих примеров, необходимых для достижения способности сети к обобщению. Слишком малое число примеров может вызвать "переобученность" сети, когда она хорошо функционирует на примерах обучающей выборки, но плохо - на тестовых примерах, подчиненных тому же статистическому распределению. Известны 4 основных типа правил обучения:

1. коррекция по ошибке:
2. машина Больцмана;
3. правило Хебба
4. обучение методом соревнования

Правило коррекции по ошибке. При обучении с учителем для каждого входного примера задан желаемый выход d. Реальный выход сети y может не совпадать с желаемым. Принцип коррекции по ошибке при обучении состоит в использовании сигнала (d-y) для модификации весов, обеспечивающей постепенное уменьшение ошибки. Обучение имеет место только в случае, когда перцептрон ошибается. Известны различные модификации этого алгоритма обучения [7].

Обучение Больцмана. Представляет собой стохастическое правило обучения, которое следует из информационных теоретических и термодинамических принципов [8]. Целью обучения Больцмана является такая настройка весовых коэффициентов, при которой состояния видимых нейронов удовлетворяют желаемому распределению вероятностей. Обучение Больцмана может рассматриваться как специальный случай коррекции по ошибке, в котором под ошибкой понимается расхождение корреляций состояний в двух режимах.

Правило Хебба. Самым старым обучающим правилом является постулат обучения Хебба [9]. Хебб опирался на следующие нейрофизиологические

наблюдения: если нейроны с обеих сторон синапса активизируются одновременно и регулярно, то сила синаптической связи возрастает. Важной особенностью этого правила является то, что изменение синаптического веса зависит только от активности нейронов, которые связаны данным синапсом.

Обучение методом соревнования. В отличие от обучения Хебба, в котором множество выходных нейронов могут возбуждаться одновременно, при соревновательном обучении выходные нейроны соревнуются между собой за активизацию. Это явление известно как правило "победитель берет все". Подобное обучение имеет место в биологических нейронных сетях. Обучение посредством соревнования позволяет кластеризовать входные данные: подобные примеры группируются сетью в соответствии с корреляциями и представляются одним элементом. При обучении модифицируются только веса "победившего" нейрона. Эффект этого правила достигается за счет такого изменения сохраненного в сети образца (вектора весов связей победившего нейрона), при котором он становится чуть ближе ко входному примеру. Можно заметить, что сеть никогда не перестанет обучаться, если параметр скорости обучения не равен 0. Некоторый входной образец может активизировать другой выходной нейрон на последующих итерациях в процессе обучения. Это ставит вопрос об устойчивости обучающей системы. Система считается устойчивой, если ни один из примеров обучающей выборки не изменяет своей принадлежности к категории после конечного числа итераций обучающего процесса. Один из способов достижения стабильности состоит в постепенном уменьшении до 0 параметра скорости обучения. Однако это искусственное торможение обучения вызывает другую проблему, называемую пластичностью и связанную со способностью к адаптации к новым данным. Эти особенности обучения методом соревнования известны под названием дилеммы стабильности-пластичности Гроссберга.

В таблице (табл.1.1) представлены различные алгоритмы обучения и связанные с ними архитектуры сетей (список не является исчерпывающим). В последней колонке перечислены задачи, для которых может быть применен каждый алгоритм. Каждый алгоритм обучения ориентирован на сеть определенной архитектуры и предназначен для ограниченного класса задач.

Таблица 1.1. Известные алгоритмы обучения

Парад	Обучающее	Архитектура	Алгоритм обучения	Задача

	игма	правило			
С учителем		Коррекция ошибки	Однослойный и многослойный перцептрон	Алгоритмы обучения перцептрона Обратное распространение Adaline и Madaline	классификация образов; аппроксимация функций; предсказание, управление
		Больцман	Рекуррентная	Алгоритм обучения Больцмана	классификация образов
		Хебб	Многослойная прямого распространения	Линейный дискриминантный анализ	анализ данных; классификация образов
		Соревнование	Соревнование	Векторное квантование	категоризация внутри класса; сжатие данных
			Сеть ART	ARTMap	классификация образов
Без учителя		Коррекция ошибки	Многослойная прямого распространения	Проекция Саммона	категоризация внутри класса; анализ данных
		Хебб	Прямого распространения или соревнование	Анализ главных компонентов	анализ данных; сжатие данных
			Сеть Хопфилда	Обучение ассоциативной памяти	ассоциативная память
		Соревнование	Соревнование SOM Кохонена Сети ART	Векторное квантование SOM Кохонена ART1, ART2	категоризация; сжатие данных категоризация.; анализ данных категоризация.
Смешанная		Коррекция ошибки и соревнование	Сеть RBF	Алгоритм обучения RBF	классификация образов; аппроксимация функций; предсказание, управление

1.4 Основные модели искусственных нейронных сетей.

1.4.1 Многослойные сети прямого распространения

Стандартная L-слойная сеть прямого распространения состоит из слоя входных узлов (будем придерживаться утверждения, что он не включается в сеть в качестве самостоятельного слоя), (L-1) скрытых слоев и выходного слоя, соединенных последовательно в прямом направлении и не содержащих связей между элементами внутри слоя и обратных связей между слоями. На рис.1.3 приведена структура трехслойной сети.

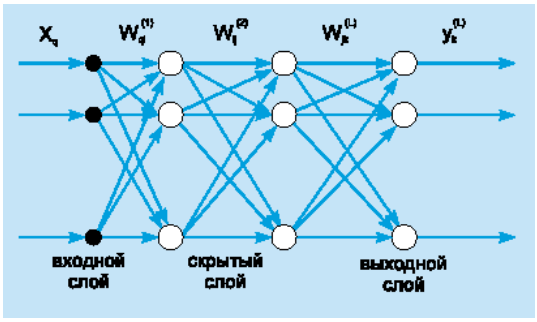

Рис.1.3. Типовая архитектура трехслойной сети прямого распространения

1.4.2 Многослойный перцептрон

Наиболее популярный класс многослойных сетей прямого распространения образуют многослойные перцептроны, в которых каждый вычислительный элемент использует пороговую или сигмоидальную функцию активации. Многослойный перцептрон может формировать сколь угодно сложные границы принятия решения и реализовывать произвольные булевы

функции [3]. Разработка алгоритма обратного распространения для определения весов в многослойном перцептроне сделала эти сети наиболее популярными у исследователей и пользователей нейронных сетей. Подавляющее большинство приложений связано именно с применением таких многослойных перцептронов. Как правило, используются именно сети, состоящие из последовательных слоев нейронов. Хотя любую сеть без обратных связей можно представить в виде последовательных слоев, именно наличие многих нейронов в каждом слое позволяет существенно ускорить вычисления используя матричные ускорители.

В немалой степени популярность перцептронов обусловлена широким кругом доступных им задач. В общем виде они решают задачу аппроксимации многомерных функций, т.е. построения многомерного отображения $F: \mathbf{x} \Rightarrow \mathbf{y}$, обобщающего заданный набор примеров $\{\mathbf{x}^\alpha, \mathbf{y}^\alpha\}$.

В зависимости от типа выходных переменных (тип входных не имеет решающего значения), аппроксимация функций может принимать вид

- *Классификации* (дискретный набор выходных значений), или
- *Регрессии* (непрерывные выходные значения)

Многие практические задачи распознавания образов, фильтрации шумов, предсказания временных рядов и др. сводится к этим базовым постановкам. Причина популярности персептронов кроется в том, что для своего круга задач они являются во-первых универсальными, а во-вторых - эффективными с точки зрения вычислительной сложности устройствами.

1.4.3 RBF-сети

Сети, использующие радиальные базисные функции (RBF-сети), являются частным случаем двухслойной сети прямого распространения. Каждый элемент скрытого слоя использует в качестве активационной функции радиальную базисную функцию типа гауссовой. Радиальная базисная функция (функция ядра) центрируется в точке, которая определяется весовым вектором, связанным с нейроном. Как позиция, так и ширина функции ядра должны быть обучены по выборочным образцам. Обычно ядер гораздо меньше, чем обучающих примеров. Каждый выходной элемент вычисляет линейную комбинацию этих радиальных базисных функций. С точки зрения задачи аппроксимации скрытые элементы формируют совокупность функций, которые образуют базисную систему для представления входных примеров в построенном на ней пространстве.

Существуют различные алгоритмы обучения RBF-сетей. Основной алгоритм использует двушаговую стратегию обучения, или смешанное обучение. Он оценивает позицию и ширину ядра с использованием алгоритма кластеризации "без учителя", а затем алгоритм минимизации среднеквадратической ошибки "с учителем" для определения весов связей

между скрытым и выходным слоями. Поскольку выходные элементы линейны, применяется неитерационный алгоритм. После получения этого начального приближения используется градиентный спуск для уточнения параметров сети.

Этот смешанный алгоритм обучения RBF-сети сходится гораздо быстрее, чем алгоритм обратного распространения для обучения многослойных перцептронов. Однако RBF-сеть часто содержит слишком большое число скрытых элементов. Это влечет более медленное функционирование RBF-сети, чем многослойного перцептрона. Эффективность (ошибка в зависимости от размера сети) RBF-сети и многослойного перцептрона зависят от решаемой задачи.

1.4.4 Самоорганизующиеся карты Кохонена

Самоорганизующиеся карты Кохонена (SOM) обладают благоприятным свойством сохранения топологии, которое воспроизводит важный аспект карт признаков в коре головного мозга высокоорганизованных животных. В отображении с сохранением топологии близкие входные примеры возбуждают близкие выходные элементы. На рис.1.2 показана основная архитектура сети SOM Кохонена. По существу она представляет собой двумерный массив элементов, причем каждый элемент связан со всеми n входными узлами.

Такая сеть является специальным случаем сети, обучающейся методом соревнования, в которой определяется пространственная окрестность для каждого выходного элемента. Локальная окрестность может быть квадратом, прямоугольником или окружностью. Начальный размер окрестности часто устанавливается в пределах от 1/2 до 2/3 размера сети и сокращается согласно определенному закону (например, по экспоненциально убывающей зависимости). Во время обучения модифицируются все веса, связанные с победителем и его соседними элементами.

Самоорганизующиеся карты Кохонена могут быть использованы для проектирования многомерных данных, аппроксимации плотности и кластеризации. Эта сеть успешно применялась для распознавания речи, обработки изображений, в робототехнике и в задачах управления. Параметры сети включают в себя размерность массива нейронов, число нейронов в каждом измерении, форму окрестности, закон сжатия окрестности и скорость обучения.

1.4.5 Модели теории адаптивного резонанса

Дилемма стабильности-пластичности является важной особенностью обучения методом соревнования. Она заключается в том, чтобы обучать сеть новым явлениям (пластичность) и в то же время сохранить стабильность, чтобы существующие знания не были стерты или разрушены.

Карпентер и Гроссберг, разработавшие модели теории адаптивного резонанса (ART1, ART2 и ARTMAP) [10], сделали попытку решить эту дилемму. Сеть имеет достаточное число выходных элементов, но они не

используются до тех пор, пока не возникнет в этом необходимость. Будем говорить, что элемент распределен (не распределен), если он используется (не используется). Обучающий алгоритм корректирует имеющийся прототип категории, только если входной вектор в достаточной степени ему подобен. В этом случае они резонируют. Степень подобия контролируется параметром сходства k, 0<k<1, который связан также с числом категорий. Когда входной вектор недостаточно подобен ни одному существующему прототипу сети, создается новая категория, и с ней связывается нераспределенный элемент со входным вектором в качестве начального значения прототипа. Если не находится нераспределенного элемента, то новый вектор не вызывает реакции сети.

Чтобы проиллюстрировать модель, рассмотрим сеть ART1, которая рассчитана на бинарный (0/1) вход. Упрощенная схема архитектуры ART1 представлена на рис.1.4.

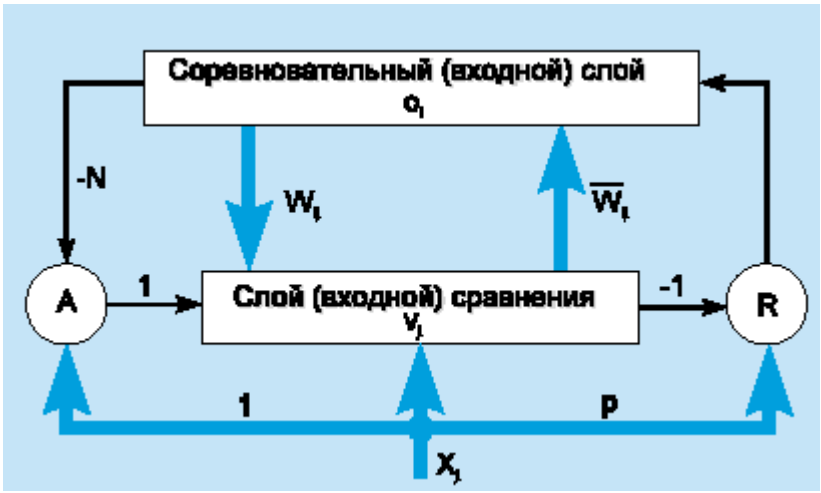

Рис.1.4. Сеть ART1

Сеть содержит два слоя элементов с полными связями. Направленный сверху вниз весовой вектор w_j соответствует элементу j входного слоя, а направленный снизу вверх весовой вектор i связан с выходным элементом i; i является нормализованной версией w_i . Векторы w_j сохраняют прототипы кластеров. Роль нормализации состоит в том, чтобы предотвратить доминирование векторов с большой длиной над векторами с малой длиной. Сигнал сброса R генерируется только тогда, когда подобие ниже заданного уровня.

Модель ART1 может создать новые категории и отбросить входные примеры, когда сеть исчерпала свою емкость. Однако число обнаруженных сетью категорий чувствительно к параметру сходства.

1.4.6 Сеть Хопфилда

Хопфилд использовал функцию энергии как инструмент для построения рекуррентных сетей и для понимания их динамики [4]. Формализация

Хопфилда сделала ясным принцип хранения информации как динамически устойчивых аттракторов и популяризовала использование рекуррентных сетей для ассоциативной памяти и для решения комбинаторных задач оптимизации.

Динамическое изменение состояний сети может быть выполнено, по крайней, мере двумя способами: синхронно и асинхронно. В первом случае все элементы модифицируются одновременно на каждом временном шаге, во втором - в каждый момент времени выбирается и подвергается обработке один элемент. Этот элемент может выбираться случайно. Главное свойство энергетической функции состоит в том, что в процессе эволюции состояний сети согласно уравнению она уменьшается и достигает локального минимума (аттрактора), в котором она сохраняет постоянную энергию. Если хранимые в сети образцы являются аттракторами, она может использоваться как ассоциативная память. Любой пример, находящийся в области притяжения хранимого образца, может быть использован как указатель для его восстановления.

Ассоциативная память обычно работает в двух режимах: хранения и восстановления. В режиме хранения веса связей в сети определяются так, чтобы аттракторы запомнили набор p n-мерных образцов $\{x_1, x_2,..., x_p)$, которые должны быть сохранены. Во втором режиме входной пример используется как начальное состояние сети, и далее сеть эволюционирует согласно своей динамике. Выходной образец устанавливается, когда сеть достигает равновесия.

Емкость памяти сети Хопфилда конечна, так как сеть с n бинарными элементами имеет максимально 2^n различных состояний, и не все из них являются аттракторами. Более того, не все аттракторы могут хранить полезные образцы. Ложные аттракторы могут также хранить образцы, но они отличаются от примеров обучающей выборки. Максимальное число случайных образцов, которые может хранить сеть Хопфилда, составляет 0.15*n. Если запоминаемые образцы представлены ортогональными векторами (в отличие от случайных), то количество сохраненных в памяти образцов будет увеличиваться.

1.5 Интеллектуальный анализ данных

Интеллектуальный анализ данных (ИАД, data mining, KDD - knowledge discovery in databases) представляет собой новейшее направление в области информационных систем, ориентированное на решение задач поддержки принятия решений на основе количественных и качественных исследований сверхбольших массивов разнородных ретроспективных данных. Принципиальное отличие ИАД от известных методологий, используемых в существующих системах поддержки принятия решений (DSS) состоит в переходе от технологии оперативного экспресс-анализа текущих ситуаций, характерной для традиционных систем обработки данных, к фундаментальным

методам исследований, опирающимся на мощный аппарат современной математики. В качестве примеров рабочего инструментария ИАД можно назвать такие разделы математики, как многомерный статистический анализ, нелинейные дифференциальные уравнения, нейронные сети, эволюционное программирование, теория наблюдения динамических систем, временные ряды и другие.

Выделяют пять стандартных типов закономерностей, которые позволяют выявлять методы ИАД: ассоциация, последовательность, классификация, кластеризация и прогнозирование. Основными задачами ИАД являются:

- краткосрочный и долгосрочный прогноз развития ситуаций;
- комплексный системный анализ, включающий в себя обнаружение и идентификацию скрытых закономерностей, ранее неизвестных взаимосвязей, значимых факторов развития самого объекта анализа и среды, в которую он погружен, визуализацию полученных результатов, подготовку предварительных отчетов и проектов допустимых решений с оценками их достоверности и эффективности возможных реализаций.

ИАД может иметь множество самых разнообразных практических приложений: в экономике, торговле, системах здравоохранения, страхования, в различных областях, связанных с контролем и прогнозированием состояния сложных динамических систем. Однако наиболее интенсивное развитие и внедрения данная методология нашла в сфере финансов и бизнеса [11-14].

Главными требованиями, предъявляемыми к методам извлечения знаний, являются эффективность и масштабируемость. Работа с очень большими базами данных требует эффективности алгоритмов, а неточность и, зачастую, неполнота данных порождают дополнительные проблемы для извлечения знаний. Нейронные сети имеют здесь неоспоримое преимущество, поскольку именно они являются наиболее эффективным средством работы с зашумленными данными. По сравнению с традиционными методами математической статистики, классификации и аппроксимации, нейросетевые технологии обеспечивают достаточно высокое качество решений при меньших затратах. Они позволяют выявлять нелинейные закономерности в сильно зашумленных неоднородных данных, дают хорошие результаты при большом числе входных параметров. По этим причинам ИАД является одним из важнейших применений нейронных сетей.

Сфера применения Data Mining ничем не ограничена — она везде, где имеются какие-либо данные. Но в первую очередь методы Data Mining сегодня, мягко говоря, заинтриговали коммерческие предприятия, развертывающие

проекты на основе информационных хранилищ данных (Data Warehousing). Опыт многих таких предприятий показывает, чтоотдача от использования Data Mining может достигать 1000%. Например, известны сообщения об экономическом эффекте, в 10–70 раз превысившем первоначальные затраты от $350,000 до $750,000 [15]. Известны сведения о проекте в $20,000,000, который окупился всего за 4 месяца. Другой пример — годовая экономия $700,000 за счет внедрения Data Mining в сети универсамов в Великобритании.

1.6 Выводы

В результате развития нейровычислений было создано большое количество эффективных моделей нейронных сетей, ориентированных на решение разнообразных задач. Благодаря этому искусственные нейронные сети успешно применяются для решения широкого класса практических задач. Поэтому при решении конкретной задачи необходимо решить вопрос выбора наиболее подходящей модели нейронной сети, ее параметров и способа обучения.

Одной из самых перспективных и быстро развивающихся областей применения нейронных сетей является интеллектуальный анализ данных. Успехи в применении систем, реализующих интеллектуальный анализ данных, породили большой интерес к такого рода такие системам. Таким образом, инструмент, обладающий основными свойствами data mining - систем (работа с базами данных, решение задач поиска ассоциаций, прогнозирования, классификации и кластеризации), соответствует современным потребностям рынка программного обеспечения.

2. Программа для анализа баз данных с помощью нейронных сетей

2.1 Цели и задачи разработки программы

Программа разрабатывалась, как инструмент, позволяющий достичь следующих целей:

- Использовать нейронные сети для решения различных задач анализа данных.
- Изучать различные модели нейронных сетей.

Исходя из названных выше целей, были сформулированы следующие задачи:

- Программа должна позволять проводить различные виды анализа данных: прогноз, ассоциативный поиск, кластеризацию и распознавание. Другими словами, программа должна выполнять роль тренажера, с помощью которого будут изучаться и создаваться различные модели нейронных сетей.
- Нужно разработать архитектуру приложения, поддерживающую работу с различными моделями нейронных сетей, их динамическое подключение

и как можно полное использование возможностей подключенных моделей.

- Исходные данные для анализа нужно извлекать из баз данных, как из удобного и универсального источника данных.

Основными требованиями к программе являются:

- простота и удобство использования;
- программа не должна ограничивать возможности моделей нейронных сетей;
- высокая скорость работы;
- поддержка широкого спектра баз данных.

Дополнительными, но не второстепенными требованиями являются:

- обеспечение устойчивости программы к сбоям – не фатальные ошибки не должны приводить к прекращению работы программы с потерей данных;
- простота реализации алгоритмов.

Входными данными для программы является база данных, а также команды пользователя. Нужная часть информации из базы данных должна извлекаться с помощью SQL – запроса, введенного пользователем. Программа должна предоставлять пользователю следующую информацию:

- прогноз значений столбцов на выбранное пользователем число записей вперед;
- степень соответствия предсказанных значений и действительных данных (если они есть);
- графики, характеризующие точность прогноза (должна быть возможность распечатки графиков и сохранения в графических файлах);
- кластер, к которому относится выбранная запись;
- тип записи (при решении задачи распознавания);
- запись, ассоциативно связанную с введенным шаблоном.

Различие между типом записи и кластером заключается в том, что при обучении тип записи указывается пользователем, а кластер определяется самой нейронной сетью.

2.2 Разработка интерфейса взаимодействия приложения с моделями нейронных сетей

Основным требованием к приложению является поддержка различных моделей нейронных сетей. Для этого необходимо разработать универсальный способ взаимодействия программы-оболочки и модулей нейронных сетей. Поскольку программа-оболочка должна поддерживать работу с модулями, которые будут созданы позже нее, то связь с модулями нейронных сетей должна устанавливаться динамически во время работы программы-оболочки. Самым простым и эффективным способом установления такой связи является использование динамически линкуемых библиотек (DLL). В качестве альтернативы можно было использовать механизмы DDE или OLE, но они значительно сложнее, чем использование DLL, хотя особых преимуществ для решения задачи не предоставляют.

Идея использования DLL заключается в следующем. Модули нейронных сетей оформляются в виде динамических библиотек и помещаются в один каталог с программой-оболочкой. В процессе работы (при создании нейронной сети) программа-оболочка ищет в текущем каталоге библиотеки с модулями сетей (для этого необходим простейший протокол, обеспечивающий обнаружение и идентификацию модулей нейронных сетей).

2.2.1 Требования к интерфейсу взаимодействия оболочки и библиотек нейронных сетей

Поскольку оболочка должна решать несколько задач анализа (прогнозирование, классификацию, распознавание и поиск ассоциаций), и различные модели нейронных сетей могут быть предназначены для решения только некоторых классов задач, то нужен механизм, позволяющий определить, для каких целей может использоваться библиотека сети.

Интерфейс должен позволять полностью использовать возможности как можно большего числа моделей нейронных сетей. То есть должна быть обеспечена возможность получения модулем нейронной сети исчерпывающей информации о решаемой задаче (объем этой информации естественным образом ограничивается командами пользователя, которые получила программа-оболочка, и информацией, хранимой в базе данных).

Интерфейс должен учитывать различие в форматах данных, которые использует нейронная сеть. Это могут быть двоичные данные 0 - 1, или вещественные числа, строка может рассматриваться сетью как единое целое и кодироваться с помощью идентификатора, но если сеть решает задачу распознавания фраз на естественном языке, строка должна рассматриваться как совокупность символов, которая, возможно, потребует предварительной обработки (например, разбиение на слова). Некоторые модели нейронных сетей для качественной работы требуют предварительной обработки входных данных (нормализация, код Грея для представления интервальных величин в двоичном виде).

Вместе с тем желательно не передавать сети данные, которые не нужны для ее работы, так как это приведет к неоправданному снижению производительности приложения.

1.1 Сравнение различных вариантов реализации интерфейса взаимодействия оболочки и библиотек нейронных сетей

Любой вариант реализации интерфейса должен содержать ответы на два вопроса.

1) Каким образом производится управление моделями нейронных сетей?

2) Как производится обмен данными между оболочкой и библиотеками нейронных сетей?

Приложение использует несколько форм представления данных, поэтому при обработке информации различными модулями требуется ее преобразование (рис.2.1). Второй вопрос должен определять, какой модуль и каким образом будет выполнять это преобразование.

Рассмотрим различные варианты реализации интерфейса, по–разному отвечающие приведенные выше вопросы.

- Управление моделями нейронных сетей осуществляется из оболочки, с использованием некоторого, заранее определенного набора функций, экспортируемых из модуля DLL. Функции реализуют операции низкоуровневого управления сетью, например, прочитать/записать состояние нейрона, установить значение на входах сети, прочитать значение выходных синапсов сети, обучить сеть, выполнить обработку данных. Оболочка должна содержать в себе логику передачи внутренних данных на входы нейронной сети. Предварительная обработка входных данных (например, нормализация) осуществляется экспортируемой их библиотеки сети функцией. Модель нейронной сети является пассивным компонентом – вся работа, связанная с чтением данных из БД, преобразованием ее во внутреннее представление (интервальное кодирование, использование тренда), а затем в формат конкретной модели нейронной сети, чтение результатов с выходов сети и преобразование ее во внутренний формат осуществляется оболочкой. Оболочка должна поддерживать несколько форматов данных, используемых нейронными сетями, различные способы обучения сетей (обучение одним образцом, пакетом образцов, в случае пакета, возможно, потребуется случайный порядок представления образцов). Чтобы оболочка могла работать с моделью сети, нужно знать, что это за модель. Для этого в библиотеке должна быть функция, которая возвращала бы всю необходимую информацию (название сети, класс решаемых ей задач, способ представления данных, особенности использования). Главное достоинство этого варианта заключается в простоте создания библиотек (все функции преобразования данных содержаться в оболочке)

Недостатки:

1) Оболочка будет работать только с теми моделями сетей, которые она “знает”. Для новых моделей потребуется либо изменять оболочку, либо создавать интерфейс взаимодействия с оболочкой, который будет размещаться в библиотеке, что перечеркивает преимущество этого варианта. Если же модели будет не достаточно данных, предоставляемых оболочкой оболочка, то модель будет невозможно подключить к оболочке.

2) Сложность оболочки.

3) Используются только те возможности сети, которые поддерживаются оболочкой.

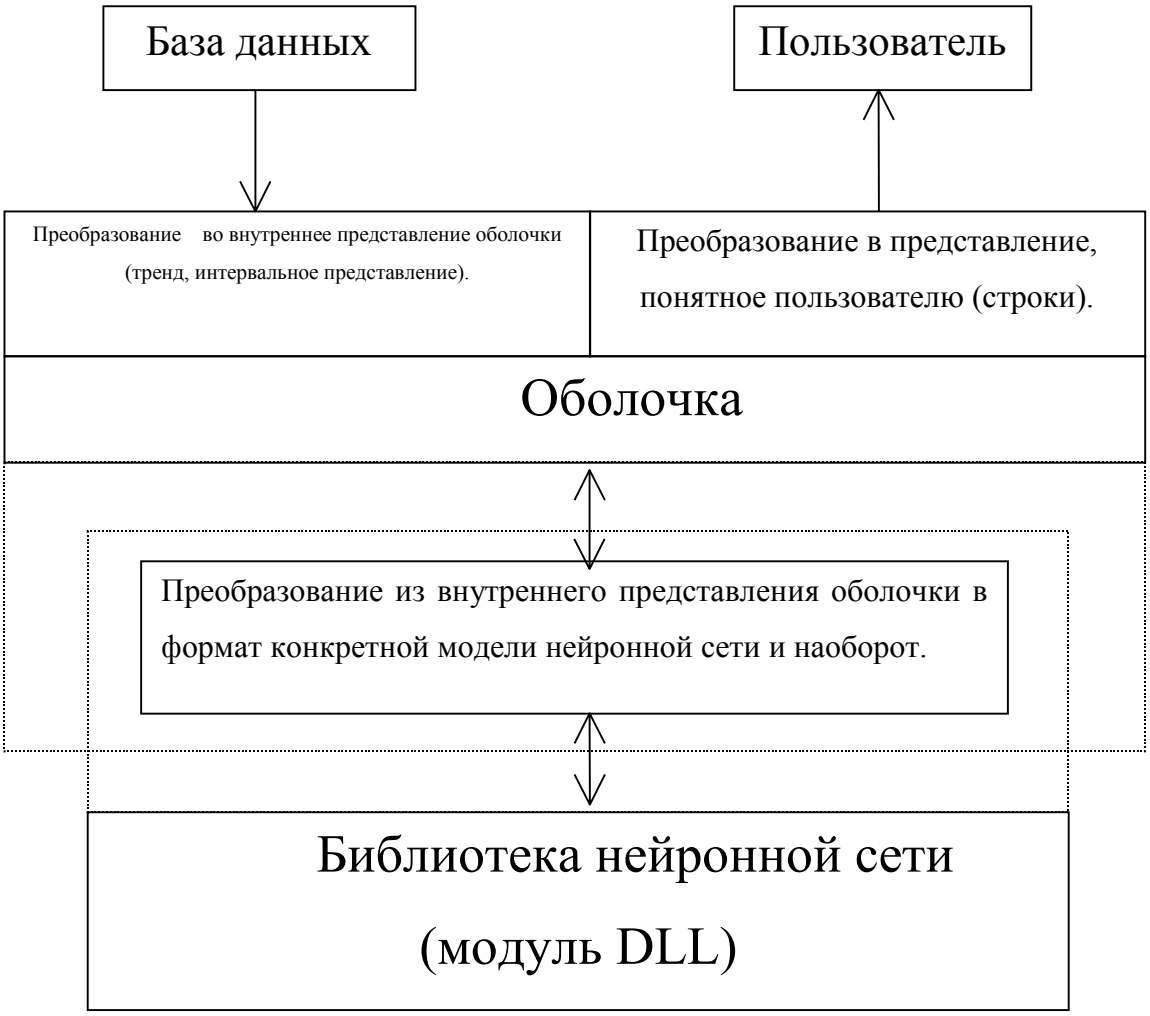

Рис.2.1. Потоки данных приложения

- Управление моделью производится с помощью небольшого числа функций высокоуровневого управления нейронной сетью (создание сети, уничтожение, обучение, работа, запись в файл, чтение из файла …). Данные во внутреннем формате оболочки, необходимые для работы сети, вместе с

информацией о их типах и способах кодирования (интервал, тренд, значение) помещаются в блок памяти, указатель на который передается функции. Модель сети должна преобразовать полученные данные в свой формат. Для передачи данных из модели в оболочку используется блок памяти такого же формата – библиотека сети должна преобразовать сигналы с выходов сети во внутреннее представление оболочки. Данный вариант обладает большей гибкостью за счет усложнения библиотеки сети. Проблема преобразования данных в формат конкретной сети решаются довольно просто: при передаче данных используется только один формат - он известен, а способ представления информации в нейронной сети известен создателю библиотеки сети. Проблема поддержки большого числа способов работы с различными типами сетей, решается с помощью использования функций управления высокого уровня – оболочка просто дает команду, не заботясь о деталях ее выполнения, этим занимается библиотека сети, которая содержит средства выполнения команды, оптимизированные для работы с конкретным типом сети. Этот вариант позволяет работать с большим числом моделей нейронных сетей, гибкость интерфейса ограничивает вопрос: ”Какие именно данные нужно предоставлять библиотеке”? В простейшем случае это записи из базы данных плюс их тип и способ представления. Для большинства моделей сетей таких данных достаточно. А если нужно будет исследовать модель сети, которая выбирает записи из БД нестандартным образом (например, при обучении сети пакетом образцов с использованием алгоритма имитации отжига требуется предъявление образцов в случайном порядке)? Тогда начинаются проблемы. Для работы с сетью оболочке необходимо знать только классы задач, которые она может решать. Эту информацию предоставляет экспортируемая функция, которая решает две задачи: определяет “специализацию” модели и позволяет отличить библиотеку нейронной сети от других динамически линкуемых библиотек. Достоинства этого варианта:

1) сравнительная простота оболочки

2) возможность работы с большим числом моделей нейронных сетей

3) возможность полного использования специфических особенностей модели

Недостатки:

1) для передачи информации о типах записей и способах их представления требуется дополнительный объем памяти, что особенно заметно при передаче большого числа коротких записей.

2) усложнение библиотеки нейронной сети

3) некоторым моделям нейронных сетей может не хватить информации, предоставляемой оболочкой

- Предыдущий вариант удовлетворяет практически всем требованиям, предъявляемым к интерфейсу. Этот вариант был выбран в качестве основного. Но в процессе его детальной разработки был создан третий вариант интерфейса. Когда решался вопрос: “Какие данные предоставлять библиотеке и в как их кодировать?”, возникла идея, что, по сути, решается вопрос не о том, как передать информацию в оболочку, а о том, какую часть от имеющейся в оболочке информации предоставить библиотеке. Для передачи сведений о типах передаваемых записей и способе их представления, производится копирование данных из внутренних структур данных оболочки. Так как библиотека при загрузке проецируется в адресное пространство оболочки, то из библиотеки можно получить доступ к структурам данных и методам оболочки. Так возникла идея о взаимном связывании библиотеки и оболочки. Вот схема ее реализации: оболочка загружает библиотеку и запоминает указатели на экспортируемые функции, затем вызывается функция библиотеки, параметрами которой являются указатели на структуры данных и на методы, доступ к которым нужен библиотеке. На рис.2.2 изображена схема реализации данного варианта интерфейса.

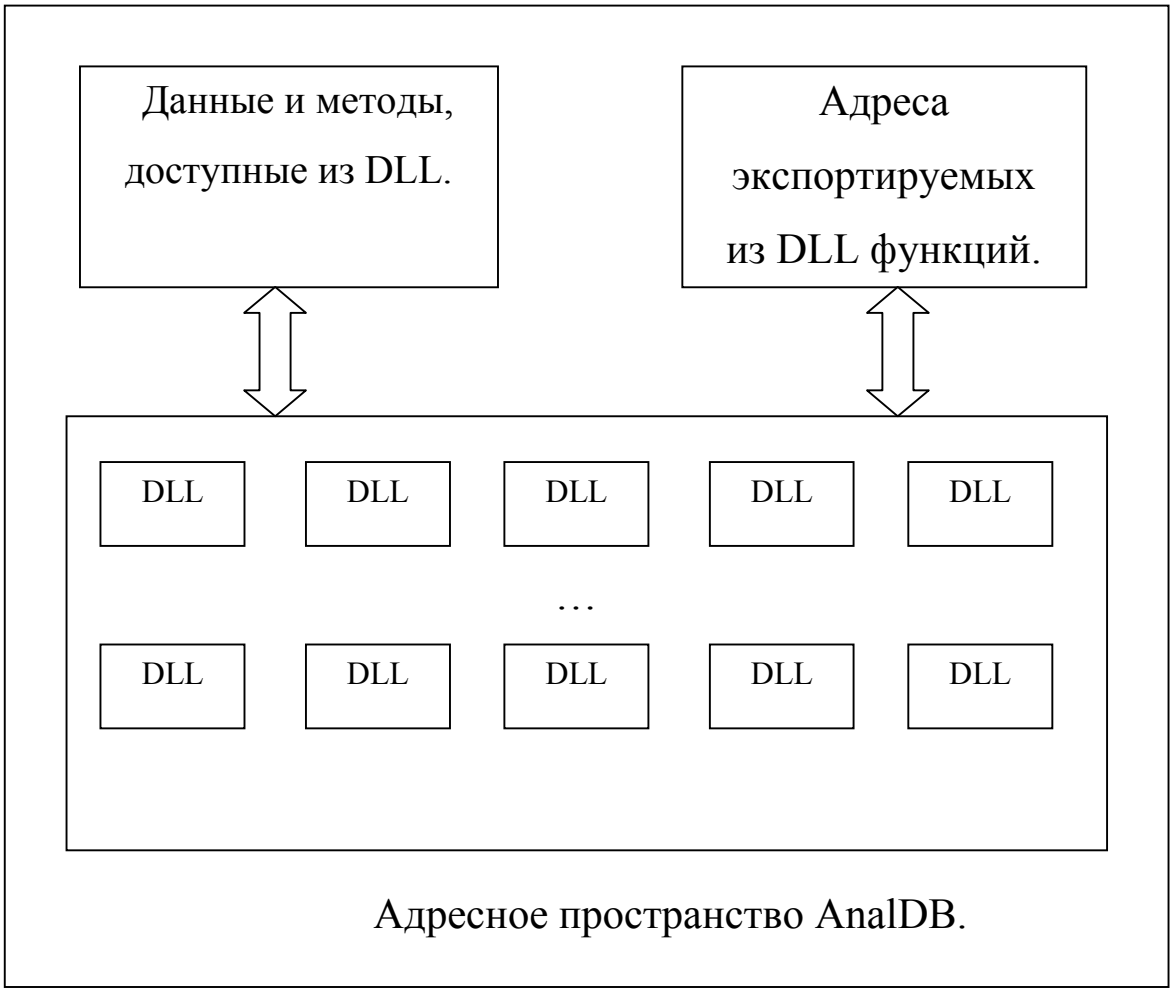

Рис.2.2. Принцип организации взаимодействия оболочки с моделями нейронных сетей

- Когда оболочка передает команду библиотеке, например “обучить сеть”, библиотека вызывает функции оболочки, которые осуществляют чтение нужных данных, причем сведения о типах данных не передается, они получаются непосредственно из структур данных оболочки.

- Этот вариант предоставляет наибольшую гибкость интерфейса – оболочка может работать с любыми моделями нейронных сетей. Единственным недостатком этого варианта является усложнение библиотеки нейронной сети.

- Последний вариант полностью удовлетворяет требованиям к интерфейсу, поэтому выбран именно он. Потенциальной проблемой на уровне реализации данного способа организации интерфейса является работа с памятью, выделенной в DLL и основной программе. Если для управления памятью использовать только функции Windows API , то сложностей не будет, но эти функции неудобны в использовании. Намного привлекательней выглядит возможность использования средств управления динамической памятью языка программирования, но в этом случае нужно принять меры по предотвращению возникновения ошибок при работе с динамической памятью.

2.3 Проектирование структуры приложения

2.3.1 Обобщенная структура приложения

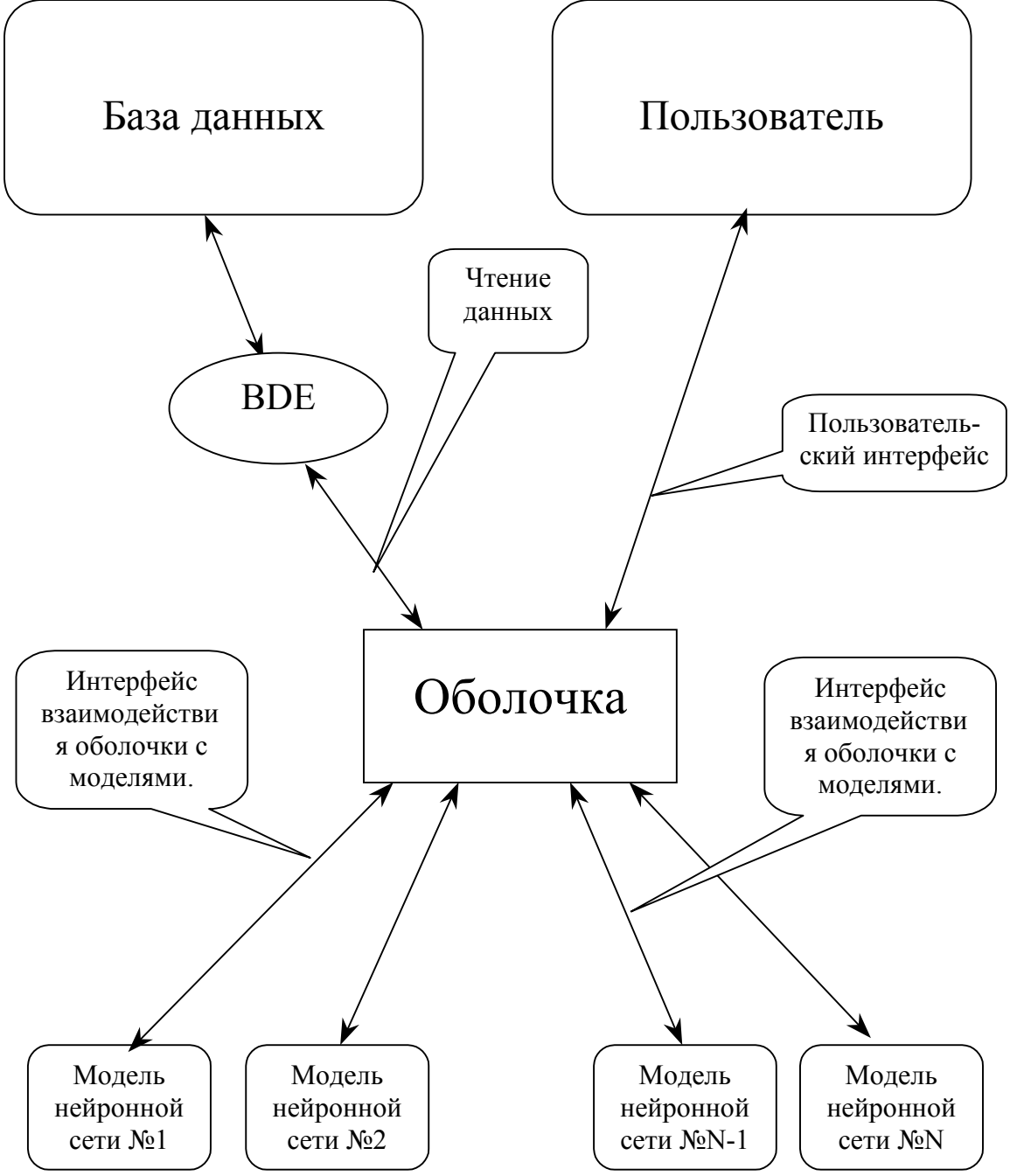

Рис.2.3. Структура приложения

Как можно увидеть из рис.2.3, приложение должно содержать реализацию трех типов взаимодействий:

- Приложение – база данных. Требуется извлекать данные из БД с помощью SQL запроса. Для этого используются средства BDE, которые значительно упрощают работы с базой данных.
- Оболочка – библиотека нейронной сети. Этот тип взаимодействия был рассмотрен в предыдущем разделе. Реализация этого взаимодействия является основной частью разработки приложения.

- Оболочка – пользователь. По сути, это пользовательский интерфейс приложения. Основная задача – разработка удобных средств ввода команд пользователем, а также наглядное представление результатов работы приложения. Реализация данного типа взаимодействия не содержит никаких потенциальных проблем, так как в Delphi предоставляет удобные средства создания интерфейса.

2.3.2 Укрупненная структура организации данных приложения

Поскольку любое приложение прежде всего занимается обработкой данных, то способ их организации оказывает определяющее влияние на структуру приложения в целом. Данном приложение создано на основе архитектуры SDI (Single Document Interface). SDI ориентирует приложение на обработку одного документа. В AnalDB таким документом является проект. Структура проекта изображена на рис.2.4. В основе проекта лежит описание данных, которые будут обрабатываться приложением. Данные описываются с помощью задания базы данных и SQL – запроса, возвращающего курсор с нужными данными. Проект содержит следующие компоненты:

Словари представляют собой контейнеры, содержащие множество значений полей для каждого столбца курсора, созданного с помощью SQL – запроса. Словари необходимы для решения следующих задач:

- интервальное кодирование данных
- кодирование любых типов данных с помощью целочисленного идентификатора – номера значения в словаре
- обнаружение во входных данных незнакомых значений – некоторые типы нейронных сетей плохо работают с незнакомыми данными

Содержимое словаря не должно меняться после его создания, иначе нейронные сети, использующие словарь будут давать некорректные результаты.

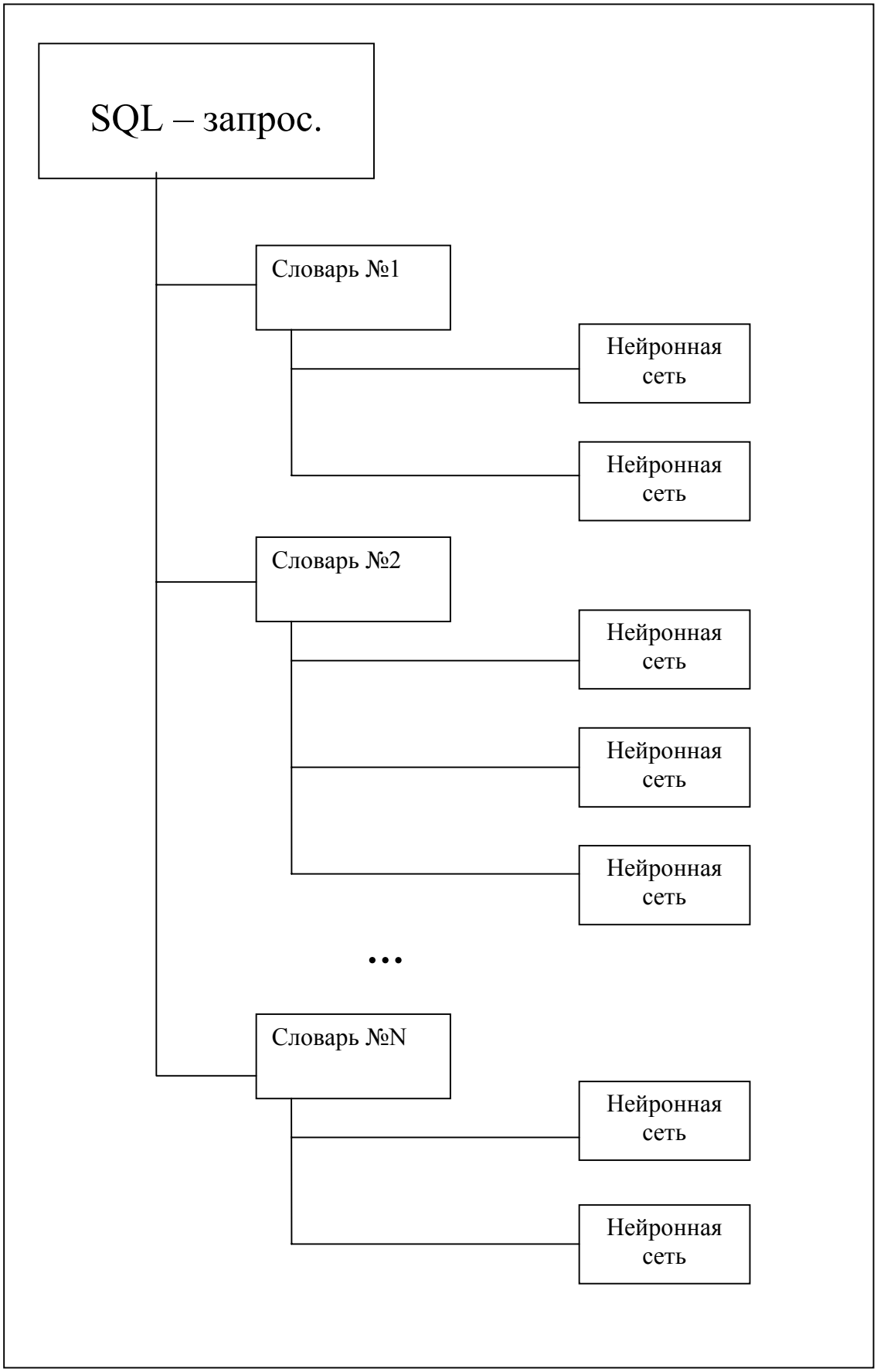

Рис.2.4. Структура проекта

Так как при изменении содержимого базы данных множество значений столбцов может измениться, то для возможности работы с различными временными срезами данных в приложении используется множество словарей. Использование множества словарей также позволяет исследовать влияние объема словаря на качество работы сети.

Нейронные сети. Каждая нейронная сеть создается на основе какого – либо словаря. Нейронная сеть с точки зрения пользователя является инструментом анализа. Каждая сеть при создании получает “специализацию” (класс решаемых задач) и при дальнейшем использовании решает только узкий класс задач. Для возможности решения различных классов задач одним типом сетей или сравнения различных сетей при решении одинаковых задач к словарю можно подключить произвольное количество нейронных сетей.

2.4 Реализация программы

Программа создана с использованием технологии объектно-ориентированного программирования. Приложение состоит из набора классов, каждый класс разработан для решения конкретной задачи. Рассмотрим классы приложения.

2.4.1 Класс TProject

Программной моделью проекта является класс TProject. Этот класс содержит общую информацию о проекте (имя проекта, примечание, дату создания, текущий каталог проекта, признак изменения, SQL-запрос …) а также методы для управления проектом.

- Инициализация проекта.
- Освобождение ресурсов.
- Создание нового проекта.
- Сохранение проекта в файле.
- Чтение проекта из файла.
- Добавление словаря к проекту.
- Удаление словаря из проекта.
- Вывод информации о проекте.

В программе используется только один объект этого класса. Поскольку в данном классе реализуется только общее управление проектом, то особых сложностей при реализации данного класса нет. Для сокращения числа дисковых операций в проект сохраняется на диск выборочно – каждый компонент содержит флаг изменения, при команде пользователя “Save” производится запись только тех элементов, которые содержат изменения.

В проекте используется иерархический способ управления: TProject управляет словарями, словари управляют нейронными сетями, а нейронные сети – библиотеками нейронных сетей. Операции ввода/вывода обладают разработаны таким образом, что повреждение части проекта не приведет к потере всех данных. При возникновении ошибки производится отсечение поддерева, содержащего ошибку.

2.4.2 Класс Dict

Этот класс содержит методы и данные используемые для управления словарем. Класс выполняет две основные задачи: обеспечивает работу со множеством значений столбцов и осуществляет управление множеством нейронных сетей, подключенных к словарю.

Первая задача решается следующим образом:

В словаре содержится массив столбцов, содержащихся в курсоре. Каждый элемент массива является ссылкой на класс Pole, который содержит информацию о типе столбца, его названии, массив со множеством значений элементов этого столбца. Для хранения списка столбцов используется динамический массив. Такой выбор обусловлен тем, что операции чтения списка производятся намного чаще, чем добавления или удаления, а массив характеризуется высокой скоростью чтения данных, но низкой скоростью добавления и удаления. Для хранения множества значений для конкретного столбца используется виртуальный упорядоченный массив. Для работы с этим массивом используется класс dyn_array. Класс использует двоичный поиск для поиска информации и для добавления новых данных. Это обеспечивает максимальную скорость поиска и приемлемую скорость добавления информации. Так как в массиве не должно быть повторяющихся элементов, то добавлению элемента предшествует поиск в массиве, поэтому применение двоичного поиска значительно уменьшит время создания словаря. Для данного массива самой часто используемой операцией является операция поиска – она используется при обучении и работе нейронной сети, поэтому использование двоичного поиска ускорит и эти операции. Следует заметить, что операции обучения сети и создания словаря являются самыми трудоемкими и применение двоичного поиска позволило увеличить скорость их выполнения в несколько раз (производилось сравнение с обычным поиском).

Кроме того Dict содержит методы для загрузки и записи словаря. Для того чтобы окно приложения могло обрабатывать сообщения, длительная операция создания словаря выполняется в отдельном потоке. Поддерживается возможность прекращения создания словаря, не дожидаясь его нормального завершения.

Вторая задача содержит меньше особенностей:

В словаре содержится массив со списком нейронных сетей, подключенных к словарю (массив используется из-за того, что операции добавления/удаления выполняются довольно редко). Словарь содержит методы для добавления и удаления нейронных сетей.

Этот класс также содержит методы для просмотра и изменения свойств словаря, а также для записи/чтения словаря.

2.4.3 Класс TNet

Помимо операций, специфических для SDI (создание нового экземпляра, уничтожение, запись на диск, загрузка из файла) и операций поддержки целостности древовидной структуры проекта, класс TNet содержит методы для работы с библиотеками нейронных сетей. Этот класс реализует часть интерфейса взаимодействия оболочки с библиотеками нейронных сетей, рассмотренного в разделе 4. Рассмотрим эти методы подробнее.

- Поиск библиотек нейронных сетей – в текущем каталоге проверяются все файлы с расширением .dll. В каждой библиотеке сначала ищутся функции с определенными именами, если все нужные функции обнаруживаются, то это библиотека нейронной сети.

- Идентификация типа нейронной сети. Каждая библиотека нейронной сети должна содержать метод Who_Are_You, который должен возвращать указатель на название модели нейронной сети и битовую маску, указывающую, какие задачи может решать данная модель.

- Загрузка библиотеки и установление взаимного связывания. Оболочка запоминает дескриптор загруженной библиотеки и адреса импортируемых функций. Затем оболочка вызывает метод библиотеки PreCreateNet, которому передается дескриптор главного окна оболочки, указатель на объект класса TProject, а также указатели на методы классов TProject, Dict, TNet Кроме того передаются указатель на функцию освобождения памяти, выделенной оболочкой.

- Позиционирование в курсоре. Так как чтение данных из курсора осуществляется под управлением библиотеки нейронной сети, то оболочка должна предоставлять ей возможность выбирать позицию чтения

- Чтение текущей записи из курсора. При чтении содержится преобразование данных согласно способу представления. Все данные помещаются в блок памяти, указатель на который передается библиотеки. Этот метод вызывается по инициативе библиотеки нейронной сети. Библиотека должна сама освободить эту память с помощью функции оболочки FreeRtesul.

Класс поддерживает работу с четырьмя типами данных для которых может использоваться до четырех способов представления. Рассмотрим сначала методы представления данных:

- Непосредственное – значение передается без какого – либо преобразования

- Тренд – передается разность между текущим и предыдущим значением поля. При этом последовательность записей в курсоре рассматривается как ряд, упорядоченный по значению какого – либо параметра (например,

времени – тогда мы имеем временной ряд). Тренд можно рассматривать как некое подобие первой производной от величины. Тренд только для решения задачи прогноза и только для числовых типов данных. Тренд эффективен для прогнозирования монотонно изменяющихся величин, для которых периодичность значений наблюдается только на уровне первой производной.

- Качественный тренд. В некоторых применениях абсолютное значение тренда не нужно, важен лишь сам факт изменения величины и направление этого изменения. В этом случае используется качественный тренд. Для того чтобы небольшие случайные изменения величины не оказывали влияния на результаты прогноза, используется порог срабатывания тренда – это процент на который должна измениться величина, чтобы факт изменения был зафиксирован. Этот тип тренда также применяется только для числовых значений.

- Интервальное кодирование. Рекомендуется для применения с вещественными данными. Множество значений столбца разбивается на N интервалов, при передаче данных вместо самого значения используется номер интервала, в который оно попадает. Применение этого способа представления данных позволяет повысить качество анализа для вещественных данных, так как излишняя точность представления данных может сильно ухудшить качество работы нейронной сети. Данный способ представления не применяется для строк.

В оболочке все данные из курсора преобразуются к четырем основным типам данных

- Целые числа со знаком – размер 4 байта
- Вещественные числа с плавающей точкой – 8 байт
- Вещественные числа с фиксированной точкой – 8 байт
- Строки

Столбцы, не сводимые к названным типам данных, оболочкой игнорируются.

Тип данных определяется оболочкой автоматически, а способ представления и его параметры выбирает пользователем при создании нейронной сети. Как входные, так и выходные данные нейронной сети могут иметь различные способы представления, более того один и тот же столбец может присутствовать на входе и выходе несколько раз, если каждое вхождение отличается способом представления данных или его параметрами. Класс TNet содержит методы для удаления дублирующихся входов или выходов.

Класс поддерживает следующие типы анализа:

- Прогноз

- Поиск ассоциаций
- Классификация
- Распознавание.

По сути, класс TNet является шлюзом между оболочкой и библиотеками нейронных сетей. Поскольку обучение нейронной сети может занять много времени, то эта операция выполняется в отдельном потоке и предусмотрена возможность ее принудительного завершения.

2.4.4 Класс TPack

Данный класс предназначен для преобразования информации, полученной от нейронной сети в вид, понятный пользователю. Класс может записывать результаты в двух форматах: в строки и в поля таблиц. Класс на использует информацию о способах представления данных (она есть в классе TNet) для преобразования потока байтов в строки (или в запись в таблице). Класс TPack автоматически производит преобразования тренда (не качественного) в значение, для этого используется предыдущее значение, которое может быть получено либо из исходных данных, либо из предыдущих результатов прогноза.

2.4.5 Класс TPrognoz – решение задачи прогноза

Задача прогноза формулируется следующим образом: требуется получить (спрогнозировать) N записей на основе известных M записей. Поскольку чтением входных данных занимается библиотека нейронной сети, то требуется передать в нее информацию о том, с какой записи нужно начать прогноз, передать управление нейронной сети и вывести результат прогноза. Для решения последней задачи используется класс TPack, номер записи, с которой нужно начать прогноз передается просто – эта запись должна быть текущей записью в курсоре. Для решения задачи прогноза нужно предоставить пользователю возможность изменять номер текущей записи в курсоре и запускать процесс прогнозирования, после этого вывести результаты работы в таблице. Но поскольку программа должна иметь средства для исследования моделей нейронных сетей, есть возможность проверить качество прогноза. Для этого нейронная сеть прогнозирует записи, которые имеются в базе данных, после чего строится таблица, в которой можно сравнить действительные значения с предсказанными. При этом также рассчитывается относительная ошибка прогноза, ее математическое ожидание и дисперсия. Также предусмотрена возможность построения графиков и диаграмм плотностей распределения относительных погрешностей прогноза. Можно построить несколько графиков на одном рисунке. Все результаты прогноза можно сохранить на диске. Результаты прогноза сохраняются в текстовом виде, а таблица тестового прогноза сохраняется в формате dbf, для того чтобы можно было произвести обработку этих данных в других программах (например, в

Excel). Графики можно сохранить в форматах bmp, wmf, emf или скопировать в буфер обмена.

2.4.6 Класс TCluster – решение задач кластеризации и распознавания.

Работа с нейронными сетями, настроенными на решение задач кластеризации и распознавания имеет много общего. Задача кластеризации и распознавания очень похожи: требуется определить, к какому классу образцов относится набор входных данных, отличие в том, что при распознавании образцы задаются явно в процессе обучения (это значение поля, которое выбрано в качестве выходного), а при кластеризации нейронная сеть сама разбивает образцы на кластеры. В данной программе кластеры идентифицируются целыми числами. В обоих случаях пользователь должен ввести интервал записей (номера первой и последней записей), которые будут обрабатываться. Нейронная сеть обрабатывает записи по одной (обрабатывается текущая запись). Позиционированием занимается оболочка. Результатом работы сети является либо значение поля (распознавание) или целое число – номер кластера. Результаты выводятся на экран в виде таблицы, их можно сохранить в формате dbf. Качество работы нейронной сети можно оценить с помощь просмотра исходной базы данных. Поскольку формат результата кластеризации и классификации не очень сложен (нет тренда, интервальное представление также не применяется), то класс TPack не используется.

2.4.7 Класс TAssoc – решение задачи ассоциативного поиска

Задача ассоциативного поиска заключается в следующем: нейронной сети предъявляется образец, содержащий неполную информацию (некоторые элементы отсутствует), сеть в процессе работы должна определить, на какой из известных образцов, он больше всего похож.

Реализация данного класса несколько отличается от других. В данном случае библиотека нейронной сети не может сама прочитать входную информацию из базы данных, так как она там отсутствует (образец вводится пользователем). Поэтому при решении задачи ассоциативного поиска оболочка должна создать блок памяти, содержащий входные данные. Для проверки качества работы нейронной сети предусмотрена возможность осуществления поиска с помощью SQL. Пользователь может ввести запрос, учитывающий все условия поиска и сравнить результаты, полученные различными методами.

2.5 Использование программы

2.5.1 Подготовка проекта

Работа с программой состоит из двух этапов: подготовительного и основного. Подготовительный этап состоит из создания и обучения нейронной сети (а также из шагов, необходимых для создания сети), второй этап - это использование сети.

Подготовительный этап состоит из следующих шагов:

1. Создание нового проекта.

Начинается с выполнения команды "Новый проект". В основе проекта лежит SQL – запрос, определяющий данные, на которых будут обучаться нейронные сети, и которые будут служить исходными данными для работы сети. Для создания проекта нужно выбрать источник данных – это может быть BDE – алиас или папка, в которой расположены таблицы DBF. После того как источник данных (база данных) выбран, нужно создать SQL – запрос, извлекающий нужные данные. Для этого можно воспользоваться мастером SQL-запросов или “вручную” ввести запрос на языке SQL. Первый вариант более прост и не требует серьезного знания SQL. Второй вариант позволяет использовать все возможности SQL. Отметим, что это не взаимоисключающие варианты, так как мастер порождает текст SQL-запроса, который при необходимости можно отредактировать.

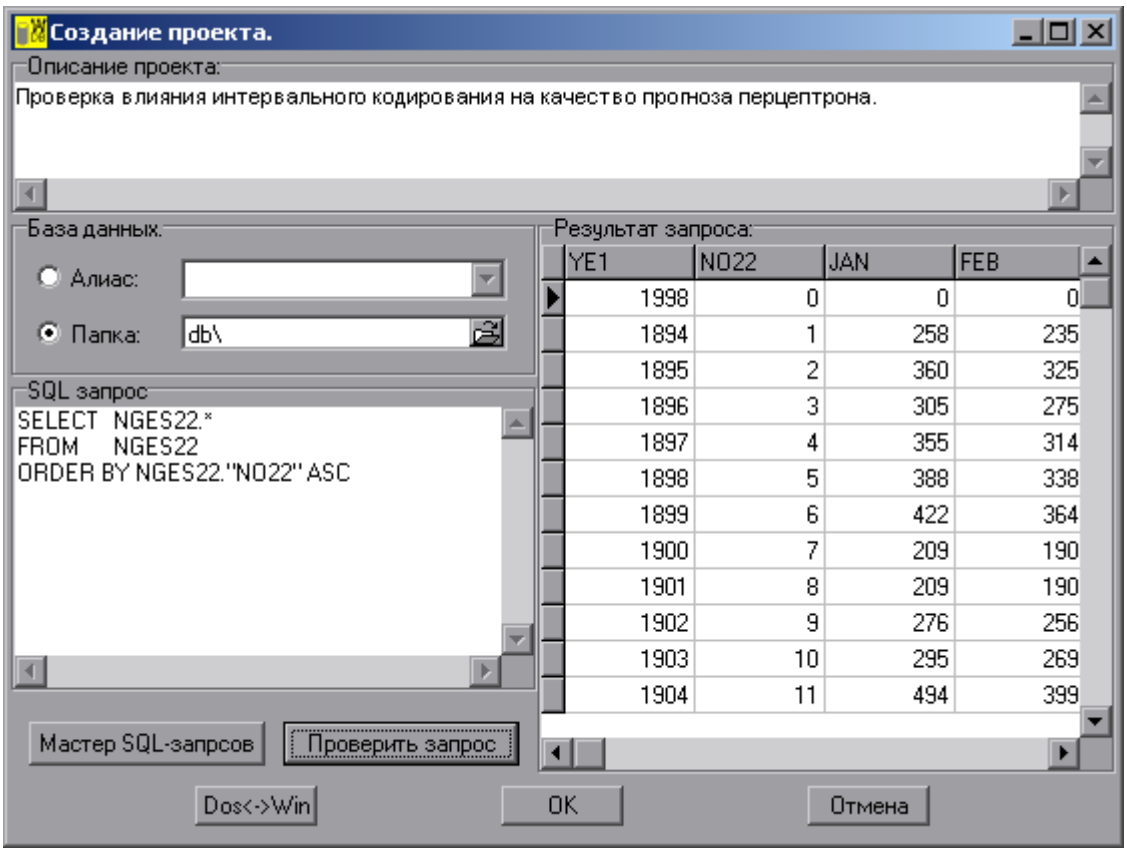

Рис.2.5. Создание проекта

2. Создание словаря.

Начинается с выполнения одноименной команды меню. При этом нужно ввести имя файла, в котором будет храниться словарь, и примечание, описывающее словарь.

3. Создание нейронной сети.

Выполняется при выполнении команды "Добавить нейронную сеть". Для создания нейронной сети необходимо выбрать нужную модель сети (перед отображением окна создания сети программа ищет в текущем каталоге библиотеки нейронных сетей; все найденные библиотеки отображаются в правой верхней части окна создания сети).

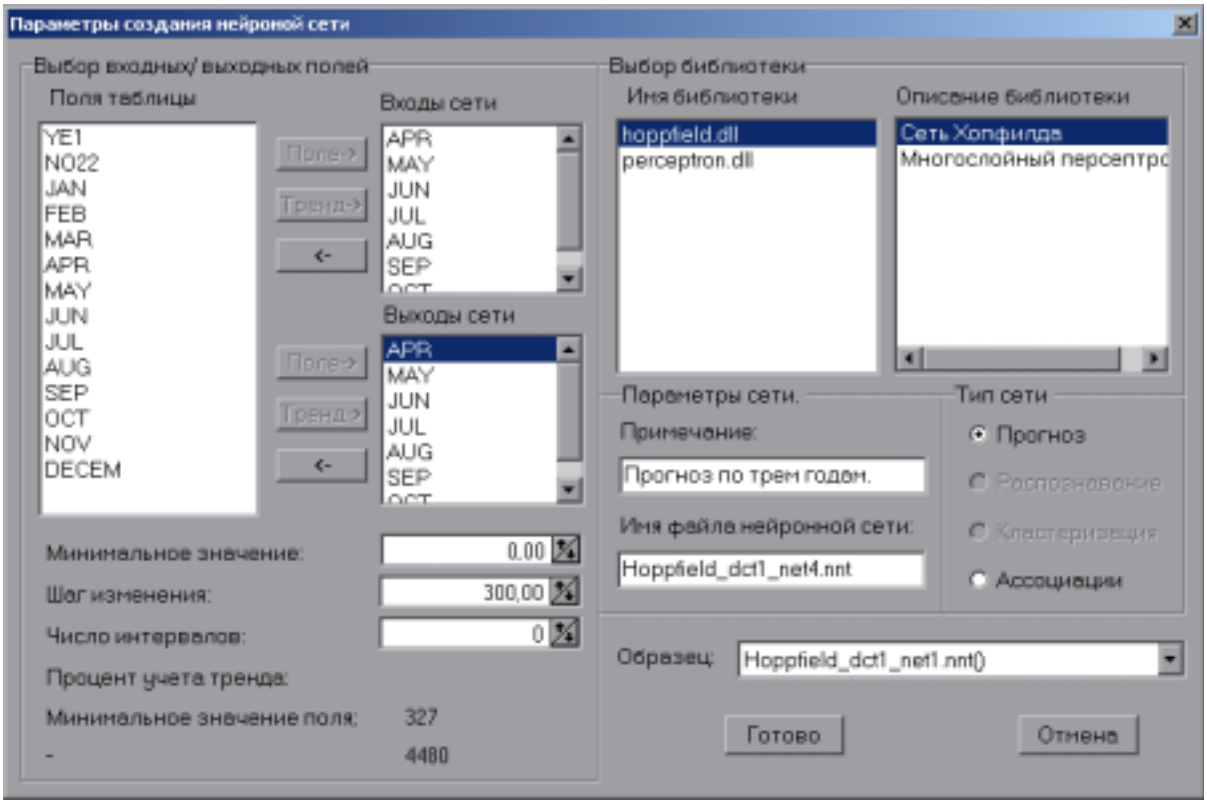

Рис.2.6. Создание нейронной сети

Затем нужно выбрать тип задачи, которую будет решать модель (прогноз, распознавание, кластеризация или ассоциативный поиск). После этого нужно выбрать поля, которые будут подаваться на входы и/или выходы нейронной сети. Выделив входное или выходное поле, можно выбрать тип предварительной обработки данных (количественный или качественный тренд, интервальное кодирование). Тренд считается качественным, если процент учета тренда не равен нулю. Признаком интервального кодирования является отличное от нуля число интервалов. Для упрощения задания параметров интервального кодирования (число интервалов, размер интервала – шаг изменения и минимальное значение поля) можно воспользоваться информацией о диапазоне изменения значений поля (отображается в левой нижней части окна). Кроме того, при ненулевом значении шага изменения или числа интервалов двойной щелчок на входном/выходном поле сформирует параметры кодирования на основе одного из этих значений и информации о максимальном и минимальном значении поля. Для работы с входными и выходными полями поддерживается операция Drag and Drop. Описание входов/выходов сети можно скопировать из любой сети текущего проекта,

решающую ту же задачу анализа. Для этого нужно просто выбрать сеть-образец из выпадающего списка в правой нижней части окна.

После задания входов и выходов сети нужно ввести имя файла, в котором она будет сохранена, а также примечание. Когда вся нужная информация будет введена, нужно нажать кнопку “Готово”. После этого нужно будет ввести дополнительные параметры, характерные для конкретной модели сети и типа задачи, решаемой моделью. После создания сети сразу же следует ее обучение. Для обучения нужно задать множество записей, на которых будет производиться обучение.

После обучения сеть готова к использованию. Следует отметить, что для получения качественно работающих сетей требуется создать несколько сетей, варьируя параметры модели, провести проверку их работы на обучающем и тестирующем множествах и на основе ее результатов выбрать лучшее значение параметров сети.

Для того чтобы начать анализ данных с помощью обученной сети, нужно выделить одну из сетей проекта и либо дважды кликнуть на ней левой кнопкой мыши или выдрать команду меню “Запустить сеть”. После этого в зависимости от типа задачи, решаемой моделью сети, появится окно, в котором можно проводить анализ. Рассмотрим, каким образом, проводятся различные типы анализа.

2.5.2 Прогнозирование

Прогнозирование позволяет получить значение одного или нескольких полей для одной или нескольких записей, следующих за текущей. Для получения прогноза нужно в верхней таблице выбрать текущую запись и нажать кнопку “Прогноз”. После этого в нижней таблице появятся прогнозируемые значения.

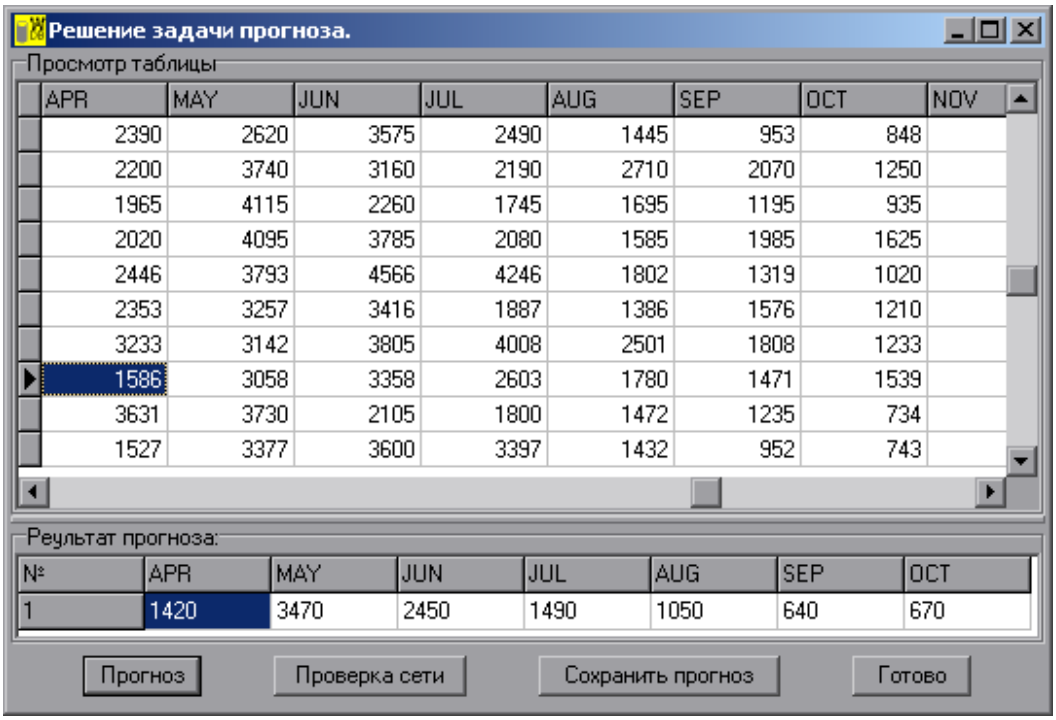

Рис.2.7. Решение задачи прогноза

Результаты прогноза можно сохранить, нажав кнопку “Сохранить прогноз”. Для проверки качества прогноза нужно нажать кнопку “Проверка сети” и выбрать интервал, на котором будет провеяться сеть. Проверка заключается в сравнении результата прогноза со значениями, хранящимися в базе данных. По окончании проверки формируется отчет, в котором содержатся реальные значения (из базы данных), прогнозируемые значения и величина относительной ошибки (отношения разности этих значений к значению из базы данных). Также подсчитывается математическое ожидание и среднеквадратичное отклонение относительной ошибки.

Для более наглядного представления результатов проверки можно представить в графическом виде (диаграмма или график). На графике отображается плотность распределения относительной погрешности прогноза.

Отчет.

Файл

N	OCT_100_	OCT_100__1	e_OCT_100__1
94	2045	848 - 873	-0,57921760391198
95	960	1073 - 1098	0,130729166666667
96	848	1323 - 1348	0,574882075471698
97	1250	1098 - 1123	-0,1116
98	935	1123 - 1148	0,214438502673797
99	1625	1148 - 1173	-0,285846153846154
100	1020	1148 - 1173	0,137745098039216
101	1210	1173 - 1198	-0,0202479338842975
102	1233	1173 - 1198	-0,0385239253852393
103	1539	948 - 973	-0,375893437296946
104	734	1323 - 1348	0,819482288828338
105	743	923 - 948	0,259084791386272
M			0,198827400703573
Sqrt(D			0,447758354715435

Рис.2.8. Результат проверки результатов прогноза в табличном виде

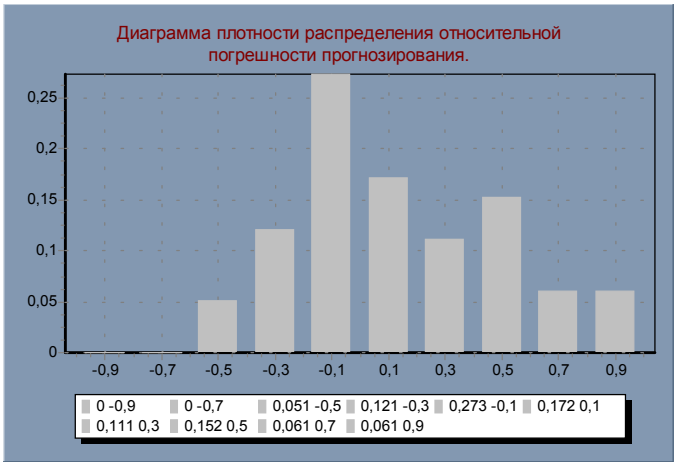

Рис.2.9. Результат проверки результатов прогноза в графическом виде

2.5.3 Ассоциативный поиск

При решении задачи поиска ассоциаций нужно заполнить несколько полей в таблице “Маска поиска/результаты поиска” признаками, по которым будет производиться поиск. После нажатия кнопки “Запустить сеть” в таблице будут отображены результаты поиска.

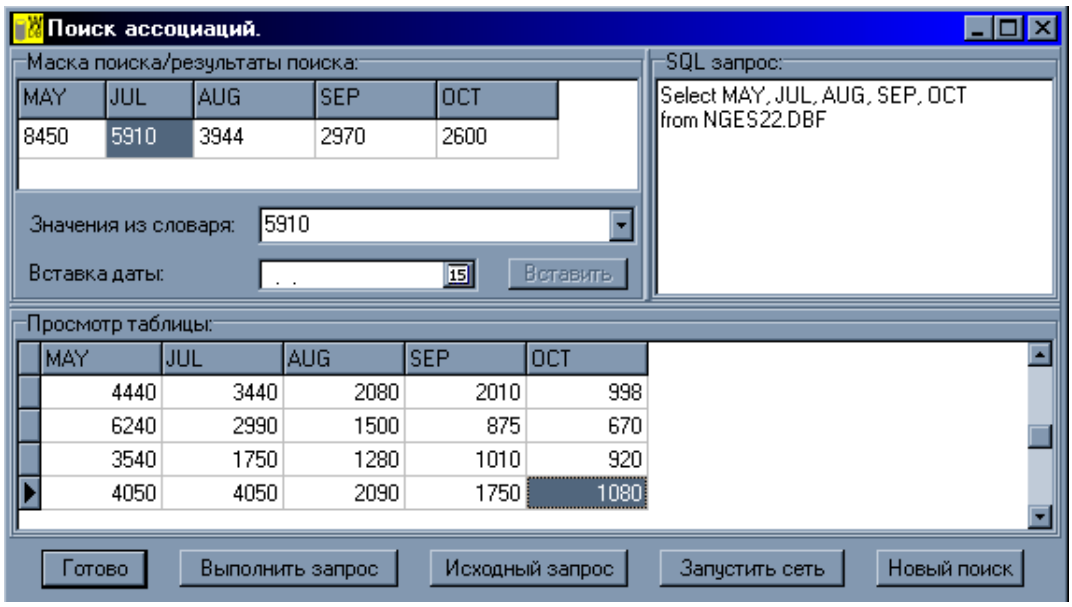

Рис.2.10. Поиск ассоциаций

Результат можно сравнить с содержимым базы данных. Для этого нужно ввести SQL – запрос и нажать кнопку “Выполнить запрос”. После этого в нижней части таблицы будет отображен результат обработки запроса.

2.5.4 Кластеризация и распознавание

Для кластеризации и распознавания в программе используется одно и то же окно. В верхней части окна расположена таблица, содержащая исходные данные – она может использоваться для проверки результатов работы сети. Для начала работы сети нужно нажать кнопку “Пуск” и задать множество записей, которые будут обрабатываться.

Рис.2.11. Решение задач распознавания и кластеризации

В результате работы сети формируется таблица, содержащая обработанные записи и соответствующие им типы (или номера кластеров). Результаты работы сети могут быть сохранены – для этого нужно нажать кнопку “Сохранить”.

2.6 Рекомендации по использованию программы

2.6.1 Применение сети Хопфилда

Модель Хопфилда, прежде всего, следует применять для ассоциативного поиска. Для решения других задач, также стоит сначала использовать эту модель, поскольку она очень проста в применении и очень быстро обучается. Особенностью этой модели является то, что она обрабатывает бинарные данные и для каждого бита обрабатываемых чисел используется отдельный нейрон. А так как ошибка сети в любом бите равновероятна, то возможны достаточно большие погрешности. По этой же причине сеть очень чувствительна даже к небольшим погрешностям во входных данных, для нее ошибки, например, в старшем и младшем битах числа неразличимы, то есть биты не взвешены. Поэтому при работе с вещественными числами настоятельно рекомендуется использовать интервальное кодирование, таким образом, часть работы по обобщению образов будет выполняться внешними по отношению к сети средствами. Однако интервальное кодирование следует применять осторожно, так как оно приводит к уменьшению количества разрядов, необходимых для кодирования данных (вместо номера значения используется номер интервала), что влечет уменьшение числа нейронов в сети и, следовательно, ее информационной емкости.

При решении задачи прогнозирования имеет смысл использовать тренд вместо абсолютного значения параметров. Дело в том, что использование нейронной сети как ассоциативной памяти (чем, по сути, и является сеть Хопфилда) не позволяет точно прогнозировать монотонно изменяющиеся значения. В таком случае можно использовать тренд (или изменение) величины. Тренд является аналогом первой производной от значения величины. Аналогом интервального кодирования для тренда является качественный тренд, он делит изменение величины на три интервала: значение уменьшилось, не изменилось или увеличилось. Качественный тренд следует применять для сети Хопфилда, а также в случаях, когда из контекста задачи ясно, что абсолютное значение изменения не важно, а важен лишь факт изменения.

2.6.2 Применение перцептрона

Если с помощью сети Хопфилда задача не решается (или решается, но плохо), то следует использовать перцептрон. При этом рекомендуется число входов сети сделать на 1 больше, чем число входных переменных. На дополнительный вход всегда подавать одну и ту же ненулевую константу (значение константы принципиально не важно, но оно не должно быть очень большим или малым по модулю). Константу можно добавить либо в таблицу с исходными данными, либо сгенерировать константный столбец средствами SQL. В любом случае, использование константы должно быть запланировано до создания проекта.

При задании параметров перцептрона не имеет смысла делать сигмоиду слишком крутой, иначе на выходе сети будут формироваться практически двоичные сигналы, что не всегда приемлемо, особенно при работе сети в режиме аппроксимации функциональной зависимости.

Пожалуй, самым важным вопросом является вопрос выбора архитектуры перцептрона. Естественно, архитектура сети сильно зависит от решаемой задачи (имеется ввиду не тип задачи, а конкретная задача, состав обучающей и тестовой выборок). Поэтому, не изучив конкретную задачу, нельзя сказать, какой должна быть архитектура, но можно дать рекомендации по подбору архитектуры.

2.7 Выводы и результаты

Была разработана архитектура приложения, позволяющая работать с широким классом моделей нейронных сетей, динамически подключать новые модели, а также решать с их помощью различные задачи анализа данных.

На основе разработанной архитектуры было создано приложение – программа для анализа баз данных с помощью нейронных сетей. В данной главе были рассмотрены детали внутренней реализации этой программы, а также ее пользовательский интерфейс и способ использования.

Работа с созданной программой позволило сделать следующие выводы:

1) Разработанная архитектура позволяет без особых усилий подключать к программе новые модули нейронных сетей. Подключенные модули могут успешно использоваться оболочкой для решения различных задач. Таким образом, разработанная архитектура получила практическое подтверждение своей работоспособности и соответствия поставленным задачам.

2) Работа с моделью многослойного перцептрона показала, что для успешного его применения недостаточно только правильной реализации этой модели и корректного взаимодействия оболочки с модулем многослойного перцептрона. Очень важным вопросом является выбор архитектуры перцептрона.

3) Простой подбор архитектуры перцептрона мало эффективен из-за большой длительности процесса обучения перцептрона.

3. Определение достаточной конфигурации перцептрона для решения задачи классификации

3.1 Введение в задачу

Искусственные нейронные сети находят практическое применение для решения различных задач (распознавание образов, выполнение прогнозов, оптимизация, реализация ассоциативной памяти, управление ...). При всем многообразии применения нейронных сетей можно выделить четыре основных режима работы нейронной сети:

Память, адресуемая по содержанию. В модели вычислений фон Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержанию, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержанию. Ассоциативная память важна при создании мультимедийных информационных баз данных.

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

Аппроксимация функций. Предположим, что имеется обучающая выборка $\{(x_1, y_1), (x_2, y_2)..., (x_n, y_n)\}$ (пары данных вход-выход), которая генерируется неизвестной функцией F(x), искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции F(x). Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

Кластеризация. При решении задачи кластеризации, которая известна также как классификация образов "без учителя", отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и размещает близкие образы в один кластер. Кластеризации может применяться для извлечения знаний, сжатия данных и исследования свойств данных.

Для решения первых трех типов задач очень часто используется многослойный перцептрон (задачу кластеризации перцептрон решать не может). Применение перцептрона также стимулируется наличием программных средств (в том числе свободно распространяемых), успешно реализующих модель перцептрона. Основными причинами популярности перцептрона являются простота реализации и универсальность модели. Под универсальностью понимается способность сети решать задачи различной

сложности. Это свойство достигается за счет того, что у перцептрона есть ряд настраиваемых параметров (прежде всего, это *архитектура* сети), которые определяют сложность задачи, которую может решить сеть. Возможность настройки параметров несколько усложняет работу с моделью сети, так как неправильный выбор параметров может привести к неэффективному (с точки зрения затрат вычислительных ресурсов) и даже некорректному применению перцептрона. Например, при использовании сети со слишком большим числом нейронов, для хранения лишних нейронов тратится память, а также тратится дополнительное время на их обучение, если же используется сеть с архитектурой, принципиально не позволяющей решить задачу, то тратится время на обучение, которое заведомо не даст желаемого результата. Примером некорректного применения сети является “жесткое” кодирование архитектуры, не позволяющей решить задачу.

3.2 Постановка задачи

Перед пользователем модели многослойного перцептрона стоит достаточно важная задача выбора архитектуры сети. Прежде всего, эта задача заключается в выборе количества слоев и нейронов в них – то есть *архитектуры* сети. Существует закономерность: чем больше слоев и нейронов, тем более сложную задачу сможет решать перцептрон, но такое правило не конструктивно, поскольку при задании архитектуры конкретной сети точные указания более важны, чем рекомендации общего характера.

Одним из вариантов выбора архитектуры сети является перебор возможных вариантов (например, постепенное наращивание сети в процессе ее обучения). Но для реализации этого варианта нужно знать, каким образом наращивать сеть (в какой слой нужно добавлять нейроны). Полный перебор всех (приемлемых с точки зрения решаемой задачи) вариантов может занять слишком много времени. Для ускорения перебора необходимы правила подбора архитектуры сети, хотя бы эвристические. Для разработки таких правил необходимо исследовать зависимость между архитектурой сети и перцептронной *представляемостью* задачи. Понятие *представляемости* относится к способности перцептрона (или другой сети) моделировать определенную функцию. Представляемость следует отличать от *обучаемости,* которая означает наличие систематической процедуры настройки весов сети для реализации заданной функции. Поскольку перцептрон может работать в различных режимах (поиск ассоциаций, аппроксимация, классификация), которые существенно отличаются друг от друга, то исследование должно быть ориентировано на конкретный режим работы сети. В данной работе этим режимом работы является классификация.

Итак, задачу можно сформулировать следующим образом. Известны следующие данные:

- совокупность пар (<вход>, <выход>), которая называется обучающей выборкой; <вход> и <выход> в общем случае являются векторами.
- модель искусственной нейронной сети (перцептрон), которая должна реализовать такое отображение, чтобы на каждый возможный сигнал <вход> формировался правильный сигнал <выход>;
- точность, с которой сеть должна моделировать отображение;
- моделируемое сетью отображение является классификацией входных данных.

Требуется найти правила выбора архитектуры перцептрона, позволяющей с заданной точностью моделировать нужное отображение.

3.3 Математическая модель

Введем следующие обозначения.

N_L – количество слоев в сети без учета входного слоя (этот слой не оказывает влияния на моделируемую перцептроном функцию, он просто хранит входные данные).

$n^{(k)}$ – количество нейронов в слое k ($0 \le k \le N$). Для входного слоя k=0, для выходного k= N_L.

$w_{ij}^{(k)}$ - вес связи j-го входа i-го нейрона слоя k с выходом j-го нейрона слоя k-1 ($1 \le k \le N$, $1 \le i \le n^{(k)}$, $1 \le j \le n^{(k-1)}$).

$y_i^{(k)}$ - выход i-го нейрона ($0 \le k \le N$, $1 \le i \le n^{(k)}$). $y_i^{(0)}$ - значение подаваемое на i-ый вход перцептрона. $y_i^{(N_L)}$ - значение i-го выхода перцептрона.

$f(x)$ - функция активации нейрона.

$X = \begin{pmatrix} x_1 \\ ... \\ x_{n^{(0)}} \end{pmatrix}$ - вектор значений признаков, подаваемый на вход сети.

$Y = \begin{pmatrix} y_1 \\ ... \\ y_{n^{(N_L)}} \end{pmatrix}$ - вектор значений выходов сети. Он является идентификатором класса, к которому относится X (при решении задачи классификации).

$F(X) = Y$ - функция классификации, которую должна моделировать нейронная сеть.

$F_P(X) = Y$ - функция, реализуемая перцептроном.

N_C - количество классов или мощность множества $\{Y^{(i)}\}$, где $Y^{(i)} = F(X)$ для любого X.

C_i - множество векторов признаков, принадлежащих к i-му классу. $C_i = \{X \mid F(X) = Y^{(i)}\}$ $(1 \le i \le N_C)$.

N_T - количество образцов в обучающей выборке.

$T = \{(X^{(i)}, Y^{(i)}) \mid Y^{(i)} = F(X^{(i)})\}$ - обучающая выборка $(1 \le i \le N_T)$.

Задача классификации с помощью перцептрона сводится к настройке параметров перцептрона, таким образом, чтобы реализуемая им функция $F_P(X)$ с заданной точностью моделировала функцию $F(X)$. Значение точность моделирования может вычисляться различными способами. Выбор конкретного способа определяется разработчиком нейронной сети. Как правило, в основе оценки точности лежит мера близости вектора значений выходов нейронной сети и ожидаемого на выходе вектора (например, Евклидово расстояние). Обычно эта мера вычисляется для всех элементов обучающей выборки и либо усредняется, либо выбирается наибольшее из полученных значений. Под параметрами понимаются следующие величины: N_L, $n^{(k)}$, $f(x)$, $w_{ij}^{(k)}$, где $0 \le k \le N_L$.

Функция $F(X)$ неизвестна. Известен лишь некоторый набор из N_T точек и значений этой функции в этих точках – обучающая выборка $T = \{(X^{(i)}, Y^{(i)}) \mid Y^{(i)} = F(X^{(i)})\}$. Обучающая выборка используется для оценки точности моделирования перцептроном функции $F(X)$. Из параметров всегда известны число входов и выходов сети ($n^{(0)}$ и $n^{(N_L)}$). Как правило, также заранее заданной является функция активации $f(x)$. Имея обучающую выборку, значения $w_{ij}^{(k)}$ можно вычислить с помощью ряда методов, самым распространенным из которых является алгоритм обратного распространения ошибки. В основе этого алгоритма лежит градиентный метод поиска.

Задача определения архитектуры сети состоит в определении параметров N_L и $n^{(k)}$, позволяющих точно смоделировать функцию классификации на обучающей выборке.

Перцептрон реализует функцию вида: $F_P(X, w_{ij}^{(k)}, N_L, n^{(k)}) = Y$, где $1 \le k \le N$, $1 \le i \le n^{(k)}$, $1 \le j \le n^{(k-1)}$, $N_L > 0$.

$F(X, w_{ij}^{(k)}, N_L, n^{(k)})_i = y_i^{(N_L)}$, где $1 \le i \le n^{(N_L)}$.

Значения выходов перцептрона вычисляются рекурсивно по следующей формуле:

$$y_i^{(k)} = f\left(\sum_{j=1}^{n^{(k-1)}} w_{ij}^{\{k\}} \cdot y_j^{(k-1)}\right), \text{ где } 1 \le k \le N,\ 1 \le i \le n^{(k)},\ 1 \le j \le n^{(k-1)}. \quad (1)$$

Формально задача поиска N_L и $n^{(k)}$ записывается следующим образом:

$$\begin{aligned}
&\sum_{i=1}^{N_L} (F_P(X^{(i)}, w_{ij}^{(k)}, N_L, n^{(k)}) - Y^{(i)})^2 \to \min, \text{где} 1 \le k \le N_L, 1 \le i \le n^{(k)}, 1 \le j \le n^{(k-1)}, \\
&N_L \text{ - целое}, \\
&N_L \ge 1, \\
&n^{(k)} \text{ - целое, где } 0 \le k \le N_L, \\
&n^{(k)} > 0, \text{где } 0 \le k \le N_L
\end{aligned} \quad (2)$$

3.4 Выбор метода решения задачи

Несмотря на то, что поставленная задача напоминает задачу нелинейного целочисленного программирования, для ее решения в общем виде методы нелинейного программирования практически неприменимы из-за переменной размерности задачи ($n^{(k)}$).

Практическая невозможность реализации идеального алгоритма обучения сети (то есть алгоритма, который находил бы глобальный минимум) также затрудняет решение задачи. Существующие методы вычисления $w_{ij}^{(k)}$, не гарантируют нахождение глобального минимума. Этот факт важен для данной задачи, так как нельзя отличить две ситуации: когда задача не решается из-за ее не представляемости с помощью выбранной архитектуры сети, и когда алгоритм поиска не смог найти решение. По этим причинам наиболее перспективным способом решения является изучение принципа работы перцептрона, с целью выявления влияния архитектуры сети на представляемость ею задачи классификации образов.

3.5 Обзор существующих решений

3.5.1 Влияние линейной разделимости на перцептронную представляемость функции

Вопрос выбора архитектуры перцептрона возник достаточно давно. С перцептронной представляемостью очень тесно связано понятие линейной разделимости. В работе [16] рассматривается связь перцептронной представляемости и линейной раделимости функции, моделируемой перцептроном, применительно к решению решении задач классификации. Ниже приводятся наиболее важные (с точки зрения решаемой задачи) фрагменты этой работы.

Функции, не реализуемые однослойной сетью, называют линейно неразделимыми. Линейная разделимость ограничивает однослойные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. Так как линейная разделимость ограничивает возможности перцептронного представления, то

важно знать, является ли данная функция разделимой. Не существует простого способа определить это, если число переменных велико. Вероятность того, что случайно выбранная функция окажется линейно разделимой, весьма мала даже для умеренного числа переменных. По этой причине однослойные перцептроны на практике ограничены простыми задачами.

К концу 60-х годов проблема линейной разделимость была хорошо понята. К тому же было известно, что это серьезное ограничение представляемости однослойными сетями можно преодолеть, добавив дополнительные слои. Многослойные сети способны выполнять более общие классификации, отделяя те точки, которые содержатся в выпуклых ограниченных или неограниченных областях.

Чтобы уточнить требование выпуклости, рассмотрим простую двухслойную сеть с двумя входами, подведенными к двум нейронам первого слоя, соединенными с единственным нейроном в слое 2 (рис.3.1). Пусть порог выходного нейрона равен 0,75, а оба его веса равны 0,5. В этом случае для того, чтобы порог был превышен и на выходе появилась единица, требуется, чтобы оба нейрона первого уровня на выходе имели единицу. Таким образом, выходной нейрон реализует логическую функцию И.

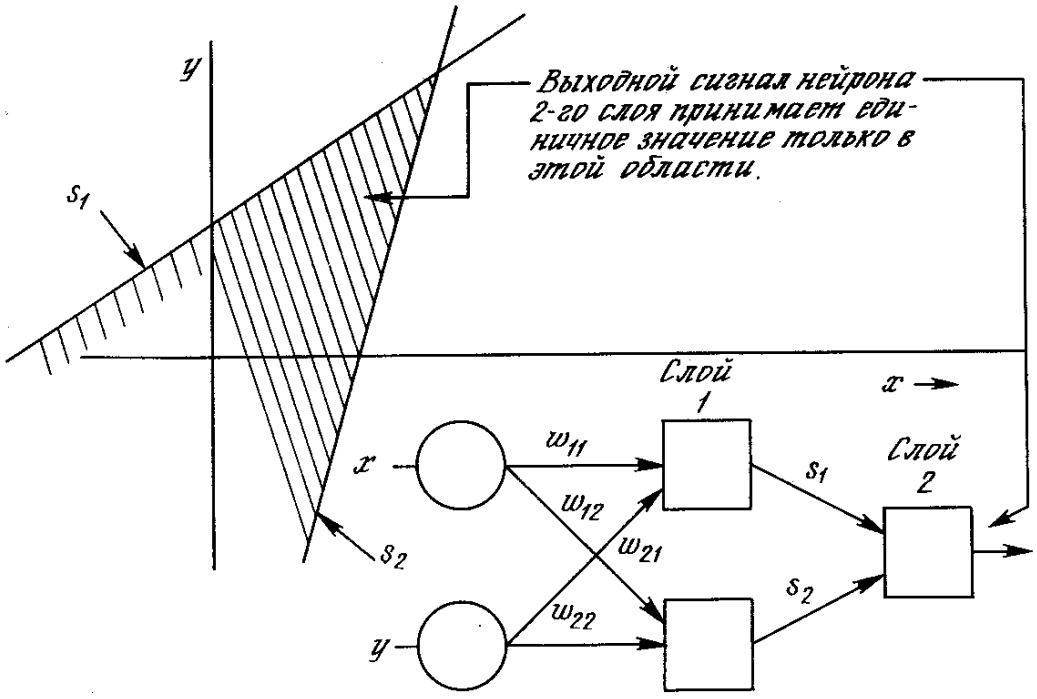

Рис.3.1. Выпуклая область решений, задаваемая двухслойной сетью

На (рис.3.1) каждый нейрон слоя 1 разбивает плоскость x-y на две полуплоскости, один обеспечивает единичный выход для входов ниже верхней прямой, другой – для входов выше нижней прямой. На (рис.3.1) показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V-образной области. Аналогично во втором

слое может быть использовано три нейрона с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован выпуклый многоугольник любой желаемой формы. Так как они образованы с помощью операции ‘И’ над областями, задаваемыми линиями, то все такие многогранники выпуклы, следовательно, только выпуклые области и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

Входы не обязательно должны быть двоичными. Вектор непрерывных входов может представлять собой произвольную точку на плоскости x-y. В этом случае мы имеем дело со способностью сети разбивать плоскость на непрерывные области, а не с разделением дискретных множеств точек. Для всех этих функций, однако, линейная разделимость показывает, что выход нейрона второго слоя равен единице только в части плоскости x-y, ограниченной многоугольной областью. Поэтому для разделения плоскостей P и Q необходимо, чтобы все P лежали внутри выпуклой многоугольной области, не содержащей точек Q (или наоборот).

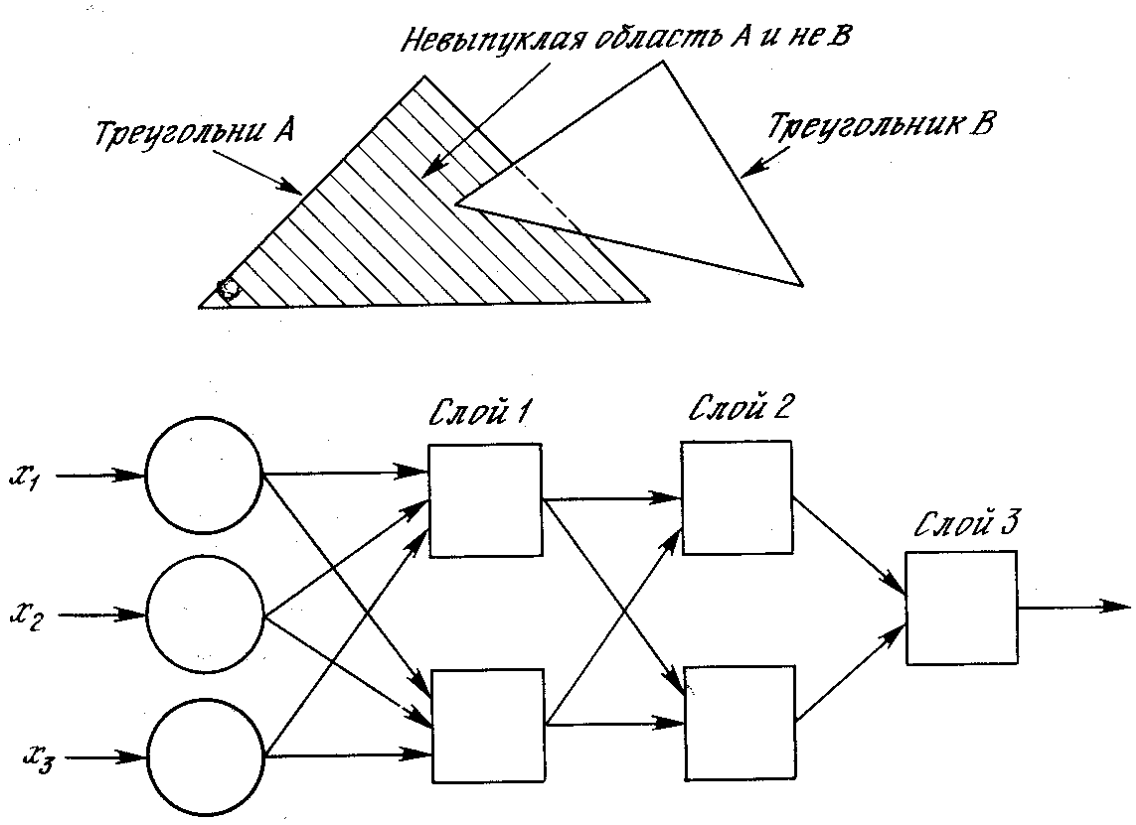

Рис.3.2. Невыпуклая область решений, задаваемая трехслойной сетью

Трехслойная сеть является более общей. Ее классифицирующие возможности ограничены лишь числом искусственных нейронов и весов. Ограничения на выпуклость отсутствуют. Теперь нейрон третьего слоя

принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть невыпуклой. На Рис.3.2. иллюстрируется случай, когда два треугольника A и B, скомбинированные с помощью функций “A и не B”, задают невыпуклую область. При добавлении нейронов и весов число сторон многоугольников может неограниченно возрастать. Это позволяет аппроксимировать область любой формы с любой точностью. Вдобавок не все выходные области второго слоя должны пересекаться. Возможно, следовательно, объединять различные области, выпуклые и невыпуклые, выдавая на выходе единицу всякий раз, когда входной вектор принадлежит одной из них.

3.5.2 Теорема Хехт-Нильсена

Теорема Колмогорова–Арнольда является основой теоремы Хехт-Нильсена. Работа Колмогорова–Арнольда посвящена проблеме представления функции многих переменных в виде суперпозиции функций меньшего числа переменных. Ниже приводятся тезисы этой работы:

- Возможность представления непрерывных функций суперпозициями.
- Теорема о представлении любой непрерывной функции 3-х переменных в виде функции не более 2-х переменных.
- Теорема о представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одной переменной и сложения.

Теорема Хехт-Нильсена доказывает представляемость функции многих переменных достаточно общего вида с помощью двухслойной нейронной сети с прямыми и полными связями. На Рис.3.3 показана архитектура такой сети.

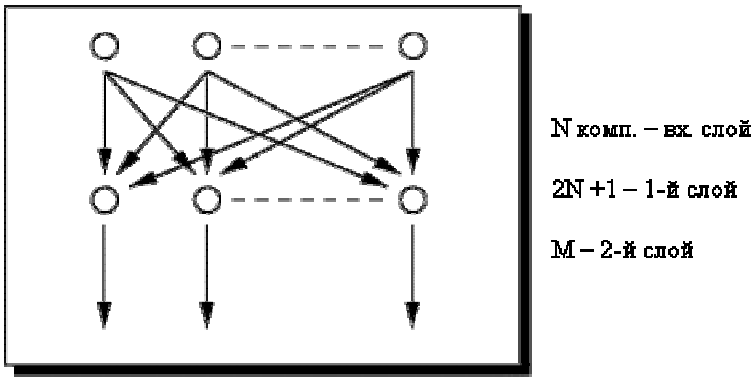

Рис.3.3. Архитектура сети по теореме Хехт-Нильсена

Теорема указывает минимальное количество нейронов в сети, необходимое для решения. Однако для практического применения теорема мало пригодна, поскольку в ней определены функции активации только для части нейронов.

3.6 Анализ модели перцептрона

3.6.1 Роль слоев перцептрона

Задачу классификации можно разделить на две задачи. Первая задача состоит в разбиении пространства признаков на области, соответствующие различным классам. Вторая задача заключается в формировании нужного идентификатора класса (выходного вектора Y).

В данном разделе рассматривается задача разделения пространства признаков многослойным перцептроном с пороговой функцией активации $f(x) = \begin{cases} 1, \text{если } x \geq 0 \\ 0, \text{если } x < 0 \end{cases}$. Доказываются следующие утверждения:

1) Независимо от числа скрытых слоев и количества нейронов в них перцептрон разбивает пространство признаков на некоторые непересекающиеся множества, объединение которых образует пространство признаков. Все точки, принадлежащие к такому множеству, преобразуются перцептроном одинаково. Далее такие множества будут называться элементарными.

2) Разбиение на элементарные множества полностью определяется весами входов нейронов первого слоя и не зависит от нейронов других слоев. Нейроны следующих слоев не влияют на решение первой задачи, они реализуют преобразование информации о принадлежности входного вектора к тому или иному классу в нужный формат. Этот формат зависит от условий задачи (значений, ожидаемых на выходе сети).

Введем следующие обозначения:

E - пространство значений признаков. Размерность пространства равна количеству входов сети ($n^{(0)}$).

$D_i \subset E$ - элементарное множество номер i, где M - количество элементарных множеств, $i = 1,...,M$.

После введения обозначений можно формализовать часть утверждений. Итак, требуется доказать:

1) $\exists D_i, D_j \subset E$ такие, что $D_i \cap D_j = \phi \; \forall i \neq j$ и $\bigcup_{k=1}^{M} D_k$, где $i, j = 1,...,M$. (3)

2) $\forall x^{(1)}, x^{(2)} \in D_i \quad \forall j = 1,...,n^{(1)}; \; y_j^{(n^{(N_L)})}(x^{(1)}) = y_j^{(n^{(N_L)})}(x^{(2)})$, где $i = 1,...,M$ (4)

3) Разбиение на элементарные множества зависит только от весов входов нейронов первого скрытого слоя.

Докажем эти утверждения.

Рассмотрим работу первого слоя нейронов. Для каждого нейрона первого слоя $y_i^{(1)}(x) = f(\sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot x_j)$, где $x \in E$ - вектор значений признаков, подаваемый на вход сети, $i = 1,...n^{(1)}$. График функции $\sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot x_j = 0$ в пространстве значений признаков представляет собой гиперплоскость, разделяющую пространство признаков на два полупространства $G_i = \{x \in E \mid \sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot x_j \geq 0\}$ и $\overline{G}_i = \{x \in E \mid \sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot x_j < 0\}$, где $i = 1,...n^{(1)}$. Отсюда следует, что $\forall x \in G_i \;\; y_i^{(1)}(x) = 1$ и $\forall x \in \overline{G}_i \;\; y_i^{(1)}(x) = 0$ (5)

Каждый нейрон первого скрытого слоя разделяет пространство признаков гиперплоскостью на два полупространства. Выход этого нейрона показывает, к какой из частей принадлежит входной сигнал *x*. Элементарные множества формируются следующим образом.

$$\begin{aligned}
&D_1 = G_1 \cap ... \cap G_{n^{(1)}-1} \cap G_{n^{(1)}}, \\
&D_2 = G_1 \cap ... \cap G_{n^{(1)}-1} \cap \overline{G}_{n^{(1)}}, \\
&D_3 = G_1 \cap G_2 \cap ... \cap \overline{G}_{n^{(1)}-1} \cap G_{n^{(1)}}, \\
&... \\
&D_{M-1} = \overline{G}_1 \cap \overline{G}_2 \cap ... \cap \overline{G}_{n^{(1)}-1} \cap G_{n^{(1)}}, \\
&D_M = \overline{G}_1 \cap \overline{G}_2 \cap ... \cap \overline{G}_{n^{(1)}-1} \cap \overline{G}_{n^{(1)}}
\end{aligned} \quad , \quad \text{где } M = 2^{n^{(1)}} . \qquad (6)$$

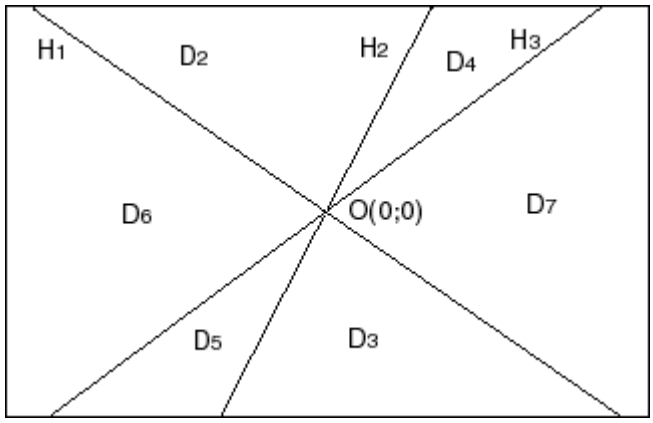

Рис.3.4. Разбиение двумерного пространства E тремя гиперплоскостями (H_1, H_2, H_3) на элементарные множества D_1-D_8 ($D_1 = D_8 = \phi$)

На (рис.3.4) приводится пример разбиения двумерного пространства признаков тремя гиперплоскостями. Этот рисунок иллюстрирует две важные особенности разбиения на элементарные множества. Во-первых, часть элементарных множеств, построенных по формулам (6), могут быть пустыми. Например, для двумерного пространства признаков только $2 \cdot n^{(1)}$ из $2^{n^{(1)}}$ возможных множеств могут быть непустыми. Во-вторых, все гиперплоскости, разделяющие пространство признаков, содержат начало координат – точку O (0, 0, …, 0). При этом пространство *E* разбивается на неограниченные

множества. Действительно, точка O (0, 0, …, 0) является корнем любого уравнения вида $\sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot x_j = 0$, где i – номер нейрона первого слоя ($1 \le i \le n^{(1)}$).

Докажем свойства (3) и (4).

1. $D_\mathrm{i} \cap D_j = \phi \; \forall \mathrm{i} \neq j$, где $\mathrm{i}, j = 1,...,M$. Данное свойство следует из (6). Запишем выражения для D_i и D_j следующим образом: $D_\mathrm{i} = \bigcap_{p=1}^{p<P_i} G_{l_p^i} \cap \bigcap_{n=1}^{n<N_i} \overline{G}_{m_n^i}$, $D_\mathrm{j} = \bigcap_{p=1}^{p<P_j} G_{l_p^j} \cap \bigcap_{n=1}^{n<N_i} \overline{G}_{m_n^j}$, где P_t - количество множеств вида G_k (при этом $l_p^t = k$), а N_t - вида $\overline{G}_k$ (при этом $m_n^t = k$), входящих в выражение (6).

   $D_\mathrm{i} \cap D_\mathrm{j} = \bigcap_{p=1}^{p<P_i} G_{l_p^i} \cap \bigcap_{n=1}^{n<N_i} \overline{G}_{m_n^i} \cap \bigcap_{p=1}^{p<P_j} G_{l_p^j} \cap \bigcap_{n=1}^{n<N_j} \overline{G}_{m_n^j}$. Поскольку выражения (6) построены таким образом, что любая пара выражений отличается хотя бы одним операндом. Отсюда следует, что существует такой индекс $m_p^i = l_n^j$ или $l_n^i = m_p^j$, а это значит, что пересечение D_i и D_j равно пересечению множеств, среди которых есть хотя бы одна пара вида G_k, $\overline{G}_k$. Пересечение таких множеств всегда пустое, таким образом $D_\mathrm{i} \cap D_\mathrm{j} = \phi$.

2. $\bigcup_{\mathrm{k}=1}^{\mathrm{M}} D_\mathrm{k}$. Это свойство практически очевидно. Доказывается выполнением операции объединения, при этом используется свойство дистрибутивности и тождество $\overline{G}_l \cup G_l \equiv E$.

3. $\forall x^{(1)}, x^{(2)} \in D_i \; \forall j = 1,...,n^{(1)} \; y_j^{(1)}(x^{(1)}) = y_j^{(1)}(x^{(2)})$, где $i = 1,...,M$. Поскольку $x^{(1)}, x^{(2)} \in D_i$, то оба вектора значений признаков $(x^{(1)}, x^{(2)})$ одновременно принадлежат множеству G_j ($j = 1,...,n^{(1)}$) (или одновременно не принадлежат – это зависит от того, какое множество (G_j или $\overline{G}_j$ использовалось при построении D_i в (6)). Отсюда следует, что
   $x^{(1)}, x^{(2)} \in D_i \Rightarrow (x^{(1)} \in G_j \Leftrightarrow x^{(2)} \in G_j)$, где $j = 1,...,n^{(1)}$. (7)

Из (5) и (7) следует $y_j^{(1)}(x^{(1)}) = y_j^{(1)}(x^{(2)})$. Поскольку на выходе первого слоя для векторов $x^{(1)}, x^{(2)}$ формируются одинаковые сигналы, являющиеся входными сигналами для второго слоя, то на выходе сети также сформируются одинаковые выходные векторы. Итак, второе утверждение доказано. Утверждение о том, что разбиение на элементарные множества зависит только от весовых коэффициентов входов нейронов первого слоя, справедливо, поскольку разбиение пространства производится гиперплоскостями,

соответствующих нейронам первого слоя. Для нейронов следующих слоев векторы признаков, принадлежащих одному элементарному множеству, неразличимы, таким образом, в этих слоях элементарные области не могут быть разделены (но могут объединяться).

3.6.2 Анализ полученных результатов

Доказанные в предыдущем пункте утверждения справедливы и при менее строгих условиях. Приведенные выше рассуждения справедливы при использовании пороговой функции вида: $\varphi(s) = \begin{cases} a, \text{если s} \geq \text{d} \\ b, \text{если s} < \text{d} \end{cases}$, где a, b и d – константы, одинаковые для всех нейронов. Причем использование такой функция активации обязательно только для нейронов второго слоя – нейроны других слоев могут использовать произвольные функции активации.

Применение во втором слое непрерывных функций, близких к пороговым (например, достаточно крутой сигмоиды) даст похожий результат. Разбиение пространства признаков будет осуществлять первый слой, только границы областей будут размытыми (на степень размытости будут влиять вид пороговой функции). Нейроны других слоев также будут влиять на разбиение, но только в области ”размытых” границ.

Элементарное множество является пересечением полупространств и поэтому представляет собой выпуклое множество. Все гиперплоскости содержат начало координат – точку O (0, 0, …, 0) и делят пространство E на неограниченные множества (рис.4). Использование только неограниченных элементарных областей не всегда удобно. Для того чтобы сместить гиперплоскости из начала координат, можно воспользоваться простым приемом. Нужно ввести дополнительный признак, который всегда равен ненулевой константе. Это приведет к увеличению размерности пространства признаков. Элементарные области полученного расширенного пространства E' также будут неограниченными, но проекции этих областей на оригинальное (не расширенное) пространство значений признаков E могут быть как ограниченными, так и неограниченными выпуклыми областями. При подаче на дополнительный вход сети константного значения производится проекция пространства E' на пространство E. Пространство E в E' задается уравнением $y_{n^0}^{(0)} = const$, где $y_{n^0}^{(0)}$ дополнительный признак.

При введении дополнительного признака уравнения гиперплоскостей в пространстве E примут вид $\sum_{j=1}^{n^{(0)}-1} w_{ij}^{(1)} \cdot x_j = w_{in^{(0)}}^{(1)} \cdot const$.

Обобщение, проведенное в данном разделе, позволяет применить результаты данной работы не только для анализа перцептрона при решении задачи классификации образов, но и при решении других задач. Особенно важным моментом является решающая роль нейронов второго слоя.

Действительно, если для разбиения пространства признаков использовать недостаточное количество гиперплоскостей, то увеличение числа слоев и нейронов в них не сможет улучшить способность перцептрона разделять пространство признаков.

3.7 Определение достаточного числа слоев и числа нейронов в них для решения задачи классификации

Для решения задачи классификации перцептроном с пороговой функцией активации достаточно трехслойной сети, рассмотренной в разделе 5.1. Каждый слой такой сети выполняет различные задачи. Первый слой разделяет пространство признаков множество полупространств. Роль этого слоя очень важна, поскольку она не может выполняться нейронами других слоев. Второй и третий слом производят кодирование элементарных областей в соответствии с форматом выходного вектора сети. Функции этих слоев похожи и часть их может реализовываться как во втором так ив третьем слое (в простом случае сеть может успешно решать задачу классификации с помощью двух слоев). Второй слой собирает из них ограниченные и неограниченные выпуклые области. Третий слой комбинирует эти выпуклые области и может собрать из них не выпуклые области. Итак, N_L можно ограничить значениями от 1 до 3.

При выборе количества нейронов в скрытых слоях перцептрона рекомендуется:

1. Увеличить размерность входного образца на 1. На дополнительный вход нужно всегда подавать одно и тоже ненулевое значение. Если размерность не увеличить, то сеть возможности сети окажутся достаточно ограниченными. Например, сеть сможет разделить точки, лежащие на прямой, проходящей через начало координат, только в том случае, если начало координат находится на отрезке, соединяющем эти точки. Пусть даны две точки $A = (x_1 ..., x_{n^{(0)}})$, $B = (a \cdot x_1 ..., a \cdot x_{n^{(0)}})$, принадлежащие E, где $a > 0$ (если A не совпадает с началом координат, то при $a > 0$ начало координат не лежит на отрезке AB). Ни одна гиперплоскость вида $\sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot y_j^{(0)} = 0$ не сможет разделить точки A и B, так как при таких условиях система неравенств $\begin{cases} \sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot x_j \geq 0, \\ \sum_{j=1}^{n^{(0)}} w_{ij}^{(1)} \cdot a \cdot x_j < 0 \end{cases}$ не имеет решений. Введение дополнительного входа решает эту проблему, поскольку константный вход позволяет сместить гиперплоскости из начала координат.

2. Для того чтобы сеть могла различать N_C классов, в каждом ее слое должно быть не менее $n^{(1)} = \log_2 N_C$ нейронов. Меньшее количество

нейронов, имеющих двоичные выходы, в принципе не смогут передать информацию идентифицирующую класс.

3. Для разделения N_T образцов в $n^{(0)}$-мерном пространстве признаков достаточно $N_T \cdot (n^{(0)}+1)$ нейронов первого слоя (если введен дополнительный константный признак; то есть если в нулевом слое содержится $n^{(0)}$ + 1 нейрон). При этом каждая точка, соответствующая образцу обучающей выборке, будет выделена простейшей (с точки зрения количества необходимых для ее отсечения гиперплоскостей) ограниченной выпуклой областью.

4. Количество нейронов во втором слое определяет количество выпуклых областей, из которых будут формироваться классы. Для правильной классификации обучающей выборки, содержащей N_T образцов во втором слое сети достаточно $n^{(2)} = N_T$ нейронов (при условии, что первый слой правильно разделил пространство признаков). При этом каждый нейрон выделит элементарную область, к которой принадлежит образец.

5. При итеративном подборе архитектуры сети лучше начинать с небольшого количества нейронов, так как это позволит сети обобщить обучающую выборку, а не просто запомнить ее. Использование предельного числа нейронов ($n^{(1)} = N_T \cdot (n^{(0)}+1), n^{(2)} = N_T$) может привести к простому запоминанию обучающих образцов. Для получения сети, способной к обобщению рекомендуется начать с архитектуры, содержащей небольшое число нейронов (но не меньшее $\log_2 N_C$). Затем количество нейронов можно постепенно увеличивать. При этом можно использовать различные варианты наращивания сети. Например, можно добавлять в первый слой по $n^{(0)}+1$ нейрону на каждый нейрон, добавленный во второй слой. Таким образом, сеть получает возможность выделить в пространстве признаков дополнительную ограниченную (или неограниченную) выпуклую область.

3.8 Выводы и результаты

В ходе исследования были получены следующие результаты.

1. Было показано, что за разбиение пространства значений признаков перцептроном с пороговой функцией активации отвечает только первый слой нейронов.

2. Были предложены практические рекомендации по выбору архитектуры перцептрона:

   - для решения задачи классификации рекомендуется использовать трехслойную сеть;

- были приведены формулы для расчета минимально необходимого и минимально достаточного количество нейронов в каждом слое;
- был предложен вариант итеративного выбора архитектуры сети.

4. Эксперимент по использованию разработанной программы для решения задачи прогнозирования притока реки Обь

4.1 Описание задачи

Разработка программы для анализа баз данных с помощью нейронных сетей тесно связана с решением практической задачи - прогнозом притока реки Обь. Эта задача имеет практической значение и рассматривается в [17], как часть задачи управления водохранилищем. Помимо классического математического аппарата для решения этой задачи предполагалось применить нейронные сети. Первой сетью, используемой для прогноза притока реки, была сеть Хопфилда. Прогноз проводился с помощью первой версии программы для анализа баз данных с помощью нейронных сетей. Результат ее работы, а также постановка задачи прогнозирования подробно рассматриваются в ряде работ ([18], [19], [20]). Прогноз, полученный с помощью модели Хопфилда, оказался недостаточно точным. Для повышения точности прогноза было решено применить многослойный перцептрон. При разработке новой версии программы это предположение было учтено, и в состав приложения была включена модель многослойного перцептрона. В данном разделе приводится отчет о применении новой версии программы для прогнозирования притока реки.

Прогнозирование проводилось с использованием и перцептрона и сети Хопфилда. Сеть Хопфилда использовалась в качестве эталона для оценки качества работы многослойного перцептрона.

4.2 Предварительное изучение исходных данных

Прежде чем применять нейронные сети, было проведено ознакомительное изучение исходных данных. Эти данные представляют собой таблицу, в строках которой содержится значение притока реки Обь за каждый из 12 месяцев года. Всего в таблице 14 столбцов (12 месяцев, номер года и порядковый номер записи) и 104 строки (данные, полученные с 1894 года по 1998 год). Для визуального изучения характера зависимости притока реки от времени были построены графики зависимости притока реки за каждый месяц от номера года. Чтобы сделать рисунки более наглядным и упростить сравнение графиков, соответствующих различным месяцам, расположим графики на четырех рисунках (рис.4.1 – рис.4.4) – по три графика на каждом.

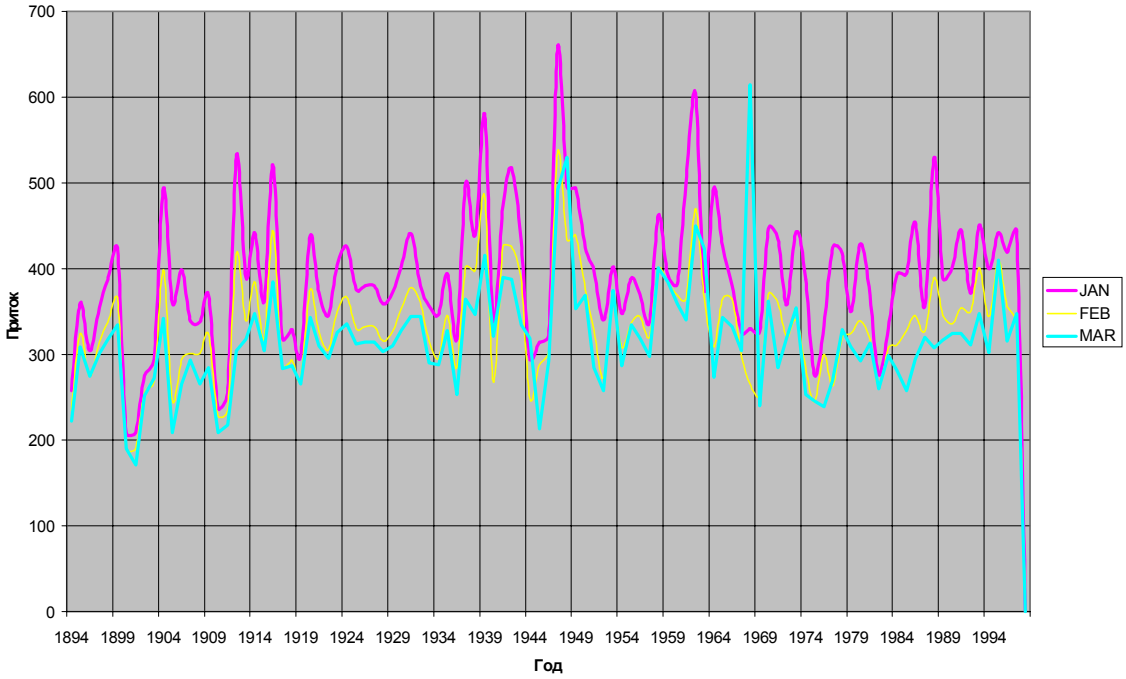

Рис. 4.1 Приток реки Обь в январе – марте (данные за 1894 – 1998 гг.)

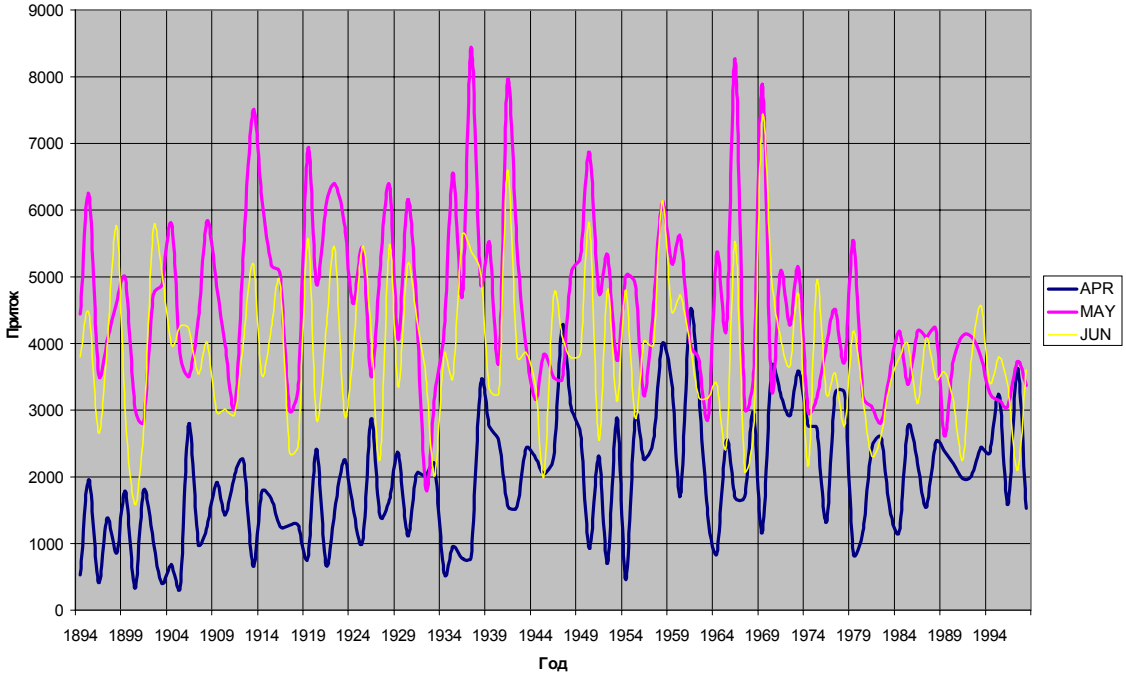

Рис.4.2 Приток реки Обь в апреле – июне (данные за 1894 – 1998 гг.)

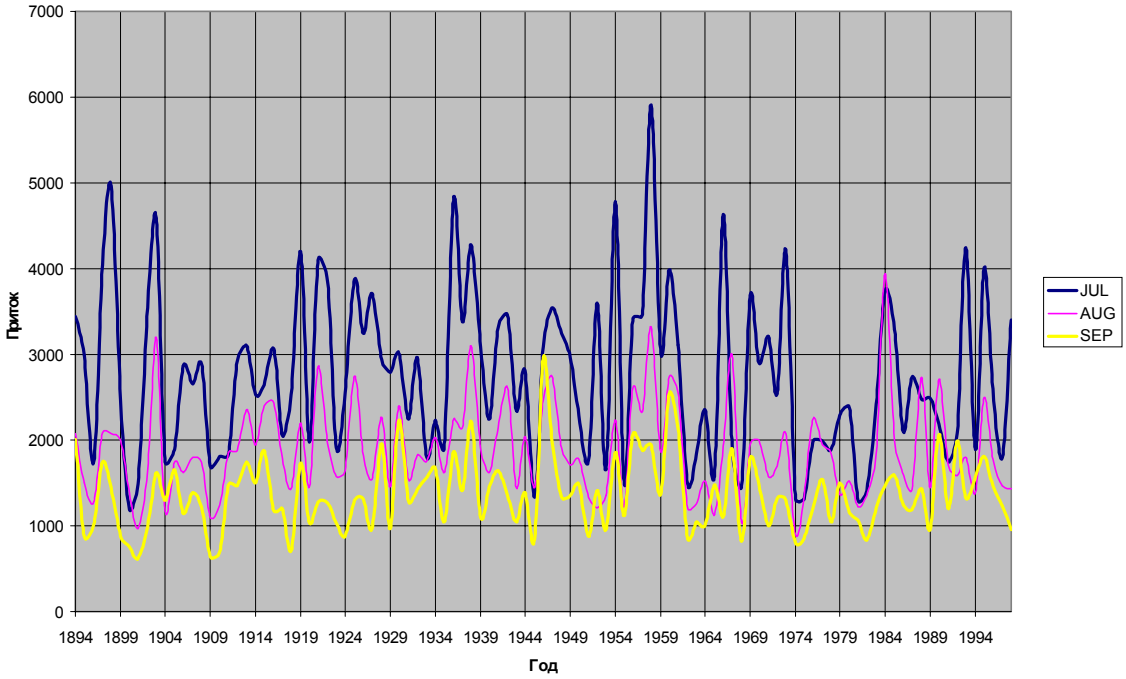

Рис.4.3 Приток реки Обь в июле – сентябре (данные за 1894 – 1998 гг.)

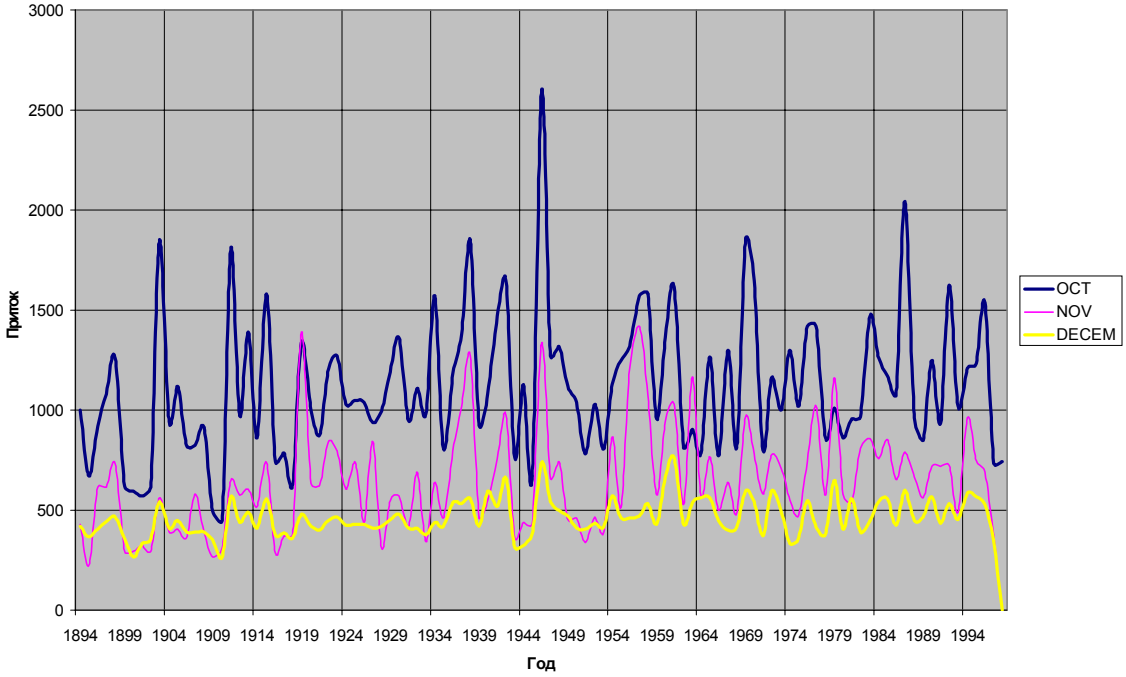

Рис.4.4 Приток реки Обь в октябре – декабре (данные за 1894 – 1998 гг.)

По графикам трудно выявить точную закономерность изменения притока реки. Зависимости притока от номера года, соответствующие различным месяцам, отличаются друг от друга. Например, в декабре приток от года к году изменяется слабо, а в мае амплитуда изменения притока очень велика. Зависимость притока реки для одного месяца достаточно сложна. В ней отсутствует явная периодичность, а также характерны резкие изменения.

Особенно сильно это проявляется для месяцев весеннего паводка. Но все-таки определенная закономерность в исходных данных прослеживается:

1. Кривые, соответствующие соседним месяцам, имеют схожую форму (рис.4.1). Это позволяет предположить, что на приток реки основное влияние оказывают факторы, общие для соседних месяцев одного года.

2. Можно заметить, что для многих месяцев проявляется схожая зависимость притока реки от года. Изменение притока состоит из циклов, имеющих похожую форму. В циклах можно выделить три части: резкое возрастание притока, постепенное его уменьшение, сопровождаемое 2-3 небольшими выбросами, и резкое снижение притока. Амплитуда циклов не постоянна и в зависимости от года и месяца меняется в широких пределах. Отметим, что это очень грубая оценка, но важен лишь сам факт выделения человеком закономерности.

Закономерности в исходных данных притока хоть и сложные, но есть. Еще раз отметим, что они слабо обоснованы и почти не формализованы. Но факт того, что человек может выделить некоторые закономерности позволяет предположить, что нейронные сети им обучиться.

Между притоком реки в соседние месяцы существует сильная зависимость, ее легко проследить на графиках. Зависимость притока за определенный месяц от года намного сложнее. Вторая закономерность более сложна. Попробуем обучить нейронные сети обеим закономерностям. Обучение сети первой закономерности является решением задачи краткосрочного прогнозирования (предсказывается приток реки на следующий месяц; исходными данными являются значение притока за предыдущие месяцы). Вторая закономерность соответствует долгосрочному прогнозированию (прогнозируется приток реки в определенном месяце в следующем году; исходными данными является приток реки за тот же месяц в предыдущие годы).

Для обучения сетей будем использовать не более четверти исходных данных (приток реки за 25 лет). Остальные данные будем использовать для проверки обученных сетей.

4.3 Решение задачи краткосрочного прогнозирования

4.3.1 Решение с помощью сети Хопфилда

Использование сети Хопфилда для краткосрочного прогнозирования показало ограниченность данной модели. Главной трудностью оказалась сильная зависимость емкости памяти сети от размерности задачи. Дело в том, что число входных и выходных переменных задачи однозначно определяет число нейронов в сети. А согласно [21] число образов, которое может запомнить сеть Хопфилда, составляет около 15% от количества нейронов в ней.

При задании небольшой глубины анализа (числа значений, подаваемых на вход сети) сеть могла обучиться лишь небольшому количеству образцов. Например, при анализе значений за 12 месяцев сеть могла “обучиться” данным за 6 лет. Слово “обучиться” взято в кавычки, поскольку сеть практически ничего не смогла запомнить. Поэтому под термином “обучиться” здесь понимается факт того, что после обучения была получена не парализованная сеть, то есть сеть (под параличом сети понимается нечувствительность ее ко входным данным). Точных предсказаний было получено не боле 5. Наращивание глубины анализа до 5 лет позволило увеличить число верных предсказаний до 15. Ошибки прогноза получались очень большими. Такое поведение сети является следствием того, что в ней содержалось недостаточно нейронов для запоминания образов. Интервальное кодирование только усугубило ситуацию. Использование интервалов вместо значений уменьшило количество нейронов, необходимых для кодирования входного значения, что и привело к уменьшению объема памяти сети.

Для увеличения объема памяти есть два пути: увеличить глубину анализа и отказаться от использования интервального кодирования (или использовать большое количество интервалов, что снижает эффективность кодирования). Но первый вариант уже при глубине анализа в 5 лет приводит к заметному замедлению работы сети. Второй вариант сильно снижает способность сети к обобщению.

Сеть Хопфилда дала плохие результаты даже при использовании большого числа нейронов. Для прогнозирования сети, обученные в результате экспериментов, практически непригодны, но их можно использовать для сравнения с перцептроном. Приведем лучшие результаты работы сети Хопфилда.

Сеть была обучена на данных за первые 150 месяцев, при этом на вход одновременно подавались значения за 60 месяцев. Выбор такой конфигурации был обусловлен двумя факторами: скоростью работы сети и качеством ее работы. 150 месяцев – это наибольшая выборка, которой сеть смогла “обучиться” при приемлемом времени работы (тест на обучающей выборке длился около 30 минут). Глубина анализа была подобрана таким образом, чтобы емкости сети хватило для избежания паралича сети. В таблице (табл.4.1)

приводятся результаты работы этой сети на обучающей и тестовой выборках. Отметим, чти при получении данных для таблицы (табл.4.1) из обучающей выборки были исключены данные за первые 12 месяцев, поскольку для полноценного прогноза необходимо иметь данные за 12 месяцев до прогнозируемого. Если нужных данных нет, сеть заменяет их 0, что приводит к неудовлетворительному прогнозу. Данное замечание справедливо для всех экспериментов с обучающей выборкой, рассматриваемых в данной главе.

Таблица 4.1 Результат краткосрочного прогнозирования с помощью сети Хопфилда

Выборка.	Среднее значение относительной ошибки прогноза.	Среднеквадратичное отклонение относительной ошибки.	Ссылка на иллюстрацию.
Обучающая	0.18	2.09	рис.4.5
Тестовая	0.33	2.04	рис.4.6

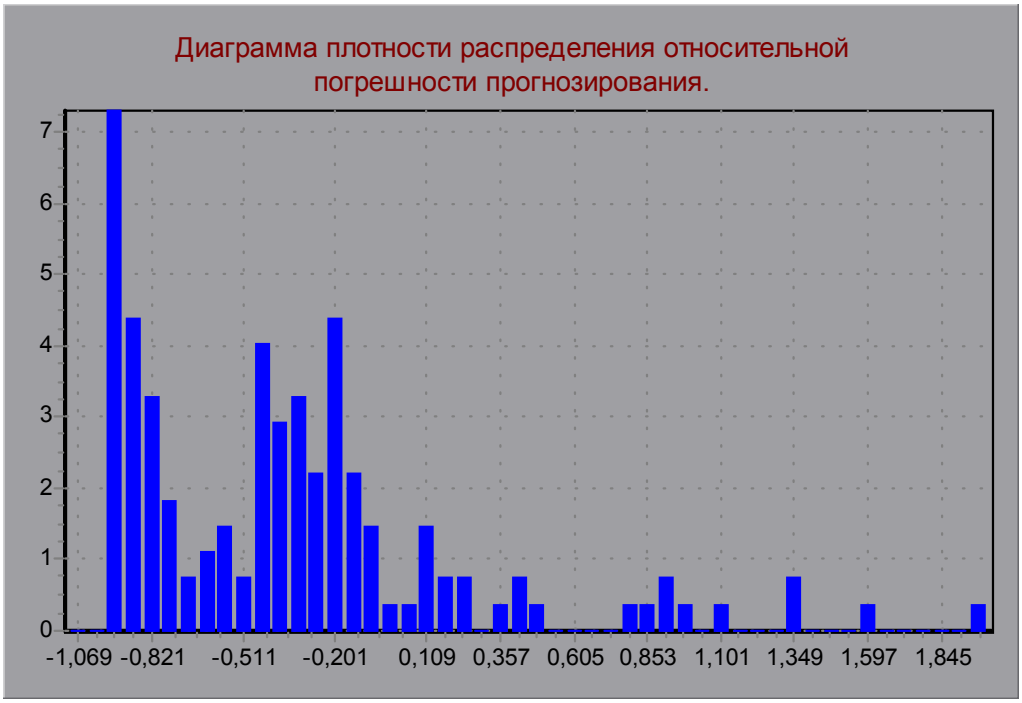

Рис.4.5 Плотность распределения относительной ошибки краткосрочного прогноза сети Хопфилда на обучающей выборке

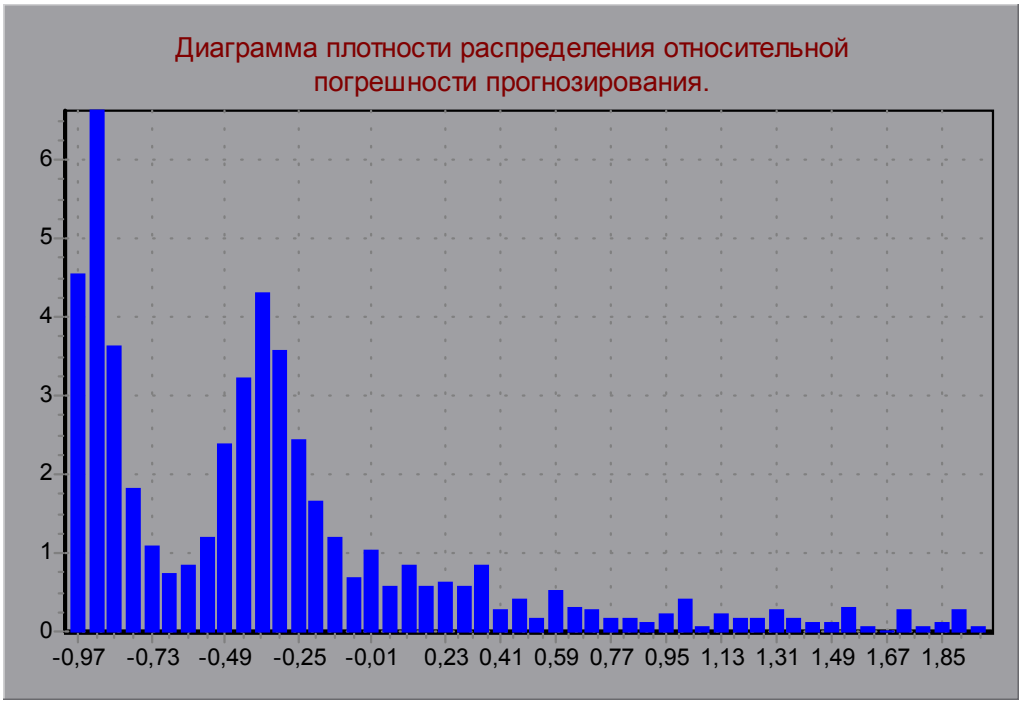

Рис.4.6 Плотность распределения относительной ошибки краткосрочного прогноза сети Хопфилда на тестовой выборке

Результаты экспериментов– это среднее значение относительной ошибки прогноза, среднеквадратичное отклонение относительной ошибки прогноза, а также диаграммы плотности распределения относительной ошибки прогноза. Первый параметр позволяет оценить смещенность прогноза. Но этого параметра недостаточно для того чтобы оценить результат работы сети. Если независимо от входных данных сеть будет всегда прогнозировать одно и то же значение, равное математическому ожиданию результата, то прогноз будет несмещенным, но некачественным. Для выявления такой ситуации нужно учитывать и второй параметр - среднеквадратичное отклонение относительной ошибки прогноза. Большое значение среднеквадратичного отклонения указывает на низкое качество анализа. При обучении сети нужно стремиться к получению слабосмещенного (в идеале несмещенного) прогноза с небольшой дисперсией. В дополнение к числовым характеристикам для оценки характера работы сети можно использовать диаграмму плотности распределения относительной ошибки прогноза. Диаграмма предназначена для визуальной оценки числовых характеристик (математического ожидания, дисперсии), а также для оценки формы распределения ошибки. Распределение, похожее на нормальное, при слабой смещенности указывает на то, что сеть чаще дает хороший (точный) прогноз. Такой результат, полученный на тестовой выборке, позволяет предположить, что сеть смогла достичь главной задачи – обобщить обучающую выборку, выявить в ней закономерность. Если же закон похож на равномерное распределение, то сеть с равной вероятностью дает точные и неточные прогнозы. Такой результат для тестовой выборки указывает на то, что сеть не смогла выявить закономерность.

Подытожим полученные результаты. Эффективность применения сети Хопфилда для краткосрочного прогнозирования оказалось достаточно низкой. Прогноз получился смещенным, ошибка очень большой. Ограничение емкости памяти сети размерностью задачи часто приводило к параличу сети. В итоге сеть Хопфилда даже не смогла запомнить обучающую выборку. Сеть Хопфилда, быстро обучается, но очень долго работает.

4.3.2 Решение с помощью перцептрона

В перцептроне, в отличие от сети Хопфилда, размерность решаемой сетью задачи не вносит принципиальных ограничений на сложность последней. Прежде всего, это следует из многослойности перцептрона. Число и размер скрытых слоев играет очень важную роль при обучении. Выбор подходящей архитектуры сети является одной из главных задач при использовании перцептрона. В ходе многочисленных экспериментов и изучения зависимости перцептронной представимости задачи от архитектуры перцептрона полуэмпирически было выяснено, что наилучшим образом задачу прогнозирования решает перцептрон с одним скрытым слоем.

Сначала эксперименты велись с перцептроном, имеющим два скрытых слоя. Число нейронов в слоях выбиралось в соответствии с рекомендациями, приведенными в предыдущей главе. Однако, второй скрытый слой оказался лишним. При уменьшении числа нейронов в нем перцептрон обучался лучше. При удалении второго скрытого слоя перцептрон мог почти без ошибок воспроизводить обучающую выборку. Данный результат устойчиво проявлялся при проведении многократных экспериментов. Прежде чем привести их результаты опишем параметры используемого перцептрона и приведем обоснование их выбора.

Глубина анализа была выбрана равной 12. Такой выбор был вызван следующими соображениями. Поскольку влияние притока реки в месяцы, близкие к прогнозируемому месяцу, очень велико, то имеет смысл использовать небольшое число месяцев. Таким числом стало 12 благодаря тому, что естественный период изменения притока реки равен году.

Чтобы сеть могла обучаться с приемлемой скоростью, функция активации не должна быть слишком пологой. Слишком крутая функция активации не может использоваться в силу ограничений представления дробных чисел в ЭВМ (происходит переполнение при очень большом значении производной). Учитывая эти два требования, в качестве функции активации была выбрана сигмоида с показателем, равным пяти.

Число нейронов в скрытом слое было выбрано в соответствии с рекомендациями, приведенными в предыдущей главе. Поскольку размерность пространства признаков равна 12, то для отсечения в нем замкнутого выпуклого множества нужно не менее 13 гиперплоскостей. Первый эксперимент проводился с перцептроном, имеющим 13 нейронов в скрытом

слое. Затем число нейронов в этом слое итеративно увеличивалось на 13 до тех пор, пока не был получен приемлемый результат. Критерием остановки итераций было качество прогноза на обучающей выборке. При этом основными параметрами были средняя относительная ошибка и среднеквадратичное отклонение этой ошибки. Чем меньше значение этих параметров, тем лучше обучилась сеть. Сеть с 130 нейронами в скрытом слое дала неплохой прогноз (табл.4.2)

При разбиении исходных данных на две выборки: обучающую, состоящую из данных за первые 300 месяцев, и тестовую (оставшиеся 948 записей) были получены следующие результаты.

Таблица 4.2 Результат краткосрочного прогнозирования с помощью перцептрона

Выборка.	Среднее значение относительной ошибки прогноза.	Среднеквадратичное отклонение относительной ошибки.	Ссылка на иллюстрацию.
Обучающая	0.05	0.31	рис.4.7
Тестовая	0.10	0.53	рис.4.8

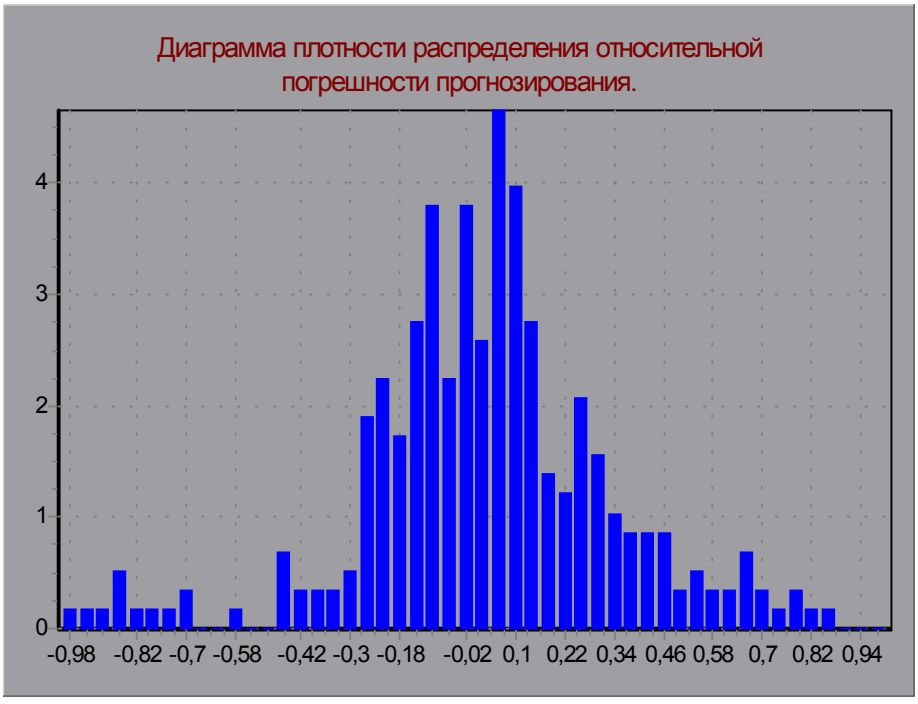

Рис.4.7 Плотность распределения относительной ошибки краткосрочного прогноза перцептрона на обучающей выборке

Рис.4.8 Плотность распределения относительной ошибки краткосрочного прогноза перцептрона на тестовой выборке

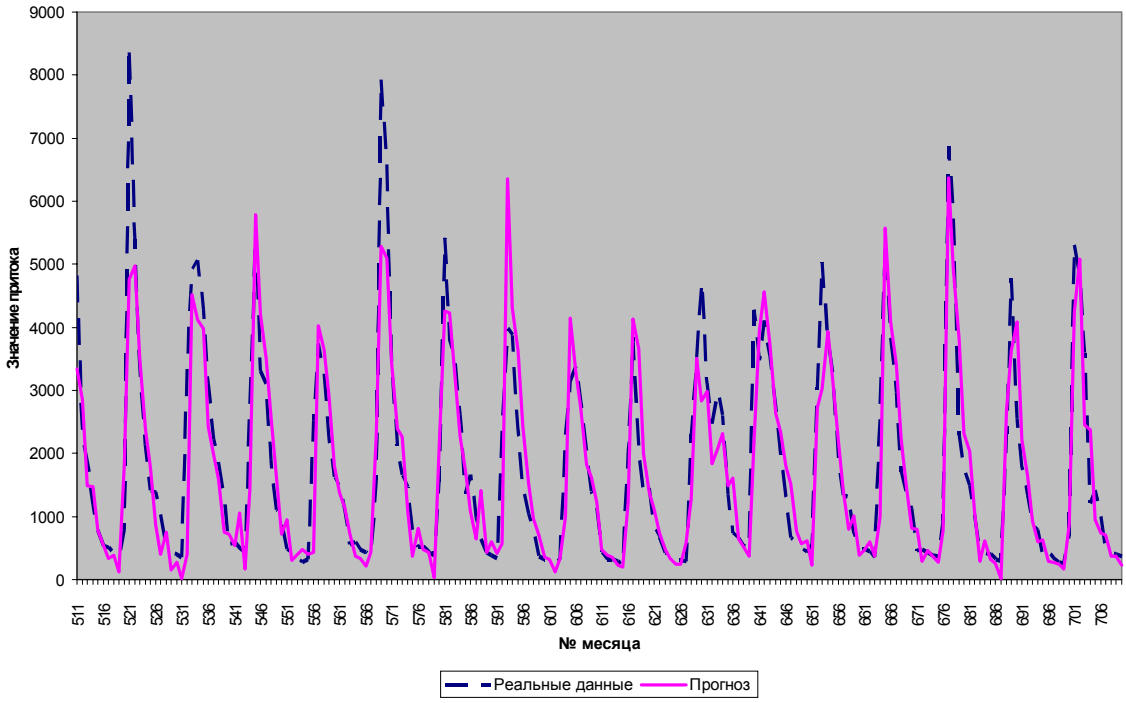

Рис.4.9 Прогнозируемые и реальные значения притока реки Обь. Данные за месяцы с 500 по 699

Перцептрон достаточно хорошо смог воспроизвести обучающую выборку. Распределение относительной ошибки получилось слабо смещенным, дисперсия небольшой, закон распределения оказался похож на нормальное распределение.

Самым важным результатом является то, что сеть смогла выявить закономерность в исходных данных, то есть обобщить обучающую выборку. На это указывает диаграмма, приведенная на (рис.4.8). На тестовой выборке относительная ошибка прогноза получилась несколько больше, чем на обучающих данных, но результат получился намного лучше, чем результат сети Хопфилда. Прогноз воспроизводил реальный закон изменения притока реки. Погрешность в прогнозе была, но тенденция изменения притока, как правило, прогнозировалась верно – большим реальным значениям соответствовало большое значение прогноза, малым – малое (рис.4.9).

Помимо большей точности предсказания перцептрон оказался более удобным средством прогнозирования. Хотя эта сеть долго обучается, работает она очень быстро (особенно по сравнению с сетью Хопфилда), так как в режиме использования перцептрона отсутствуют итеративные шаги. Поскольку сеть предполагается чаще использовать, чем обучать, то высокая скорость работы является более важной, чем высокая скорость обучения.

4.4 Решение задачи долгосрочного прогнозирования

4.4.1 Решение с помощью сети Хопфилда

Негативные свойства сети Хопфилда, проявившиеся при краткосрочном прогнозировании, имели место и при выполнении долгосрочного прогноза. Рассмотрим полученные результаты. Обучающая выборка состояла из данных за 25 лет, а тестовая – из данных за оставшиеся 79 лет. На вход сети предъявлялся приток реки за все месяцы пяти лет, предшествующих прогнозируемому году. Выходом являлся приток реки за апрель. В табл.4.3 приводятся результаты эксперимента с сетью Хопфилда.

Таблица 4.3 Результат долгосрочного прогнозирования с помощью сети Хопфилда

Выборка.	Среднее значение относительной ошибки прогноза.	Среднеквадратичное отклонение относительной ошибки.	Ссылка на ил-люстрацию.
Обучающая	-0.59	0.59	рис.4.10
Тестовая	-0.61	0.24	рис.4.11

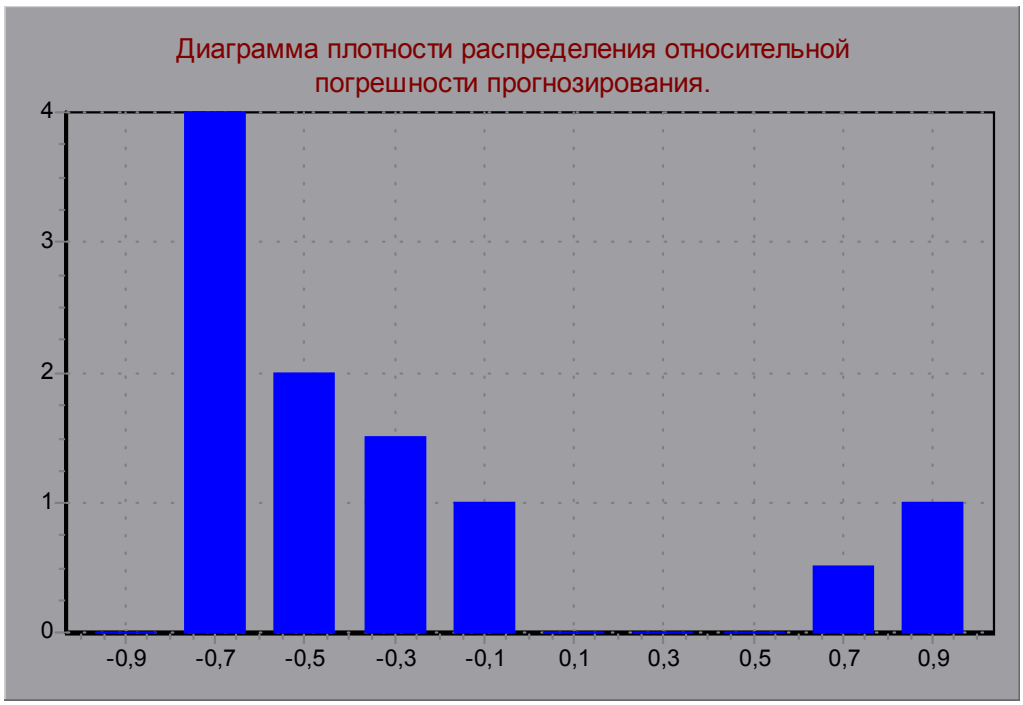

Рис.4.10 Плотность распределения относительной ошибки долгосрочного прогноза сети Хопфилда на обучающей выборке

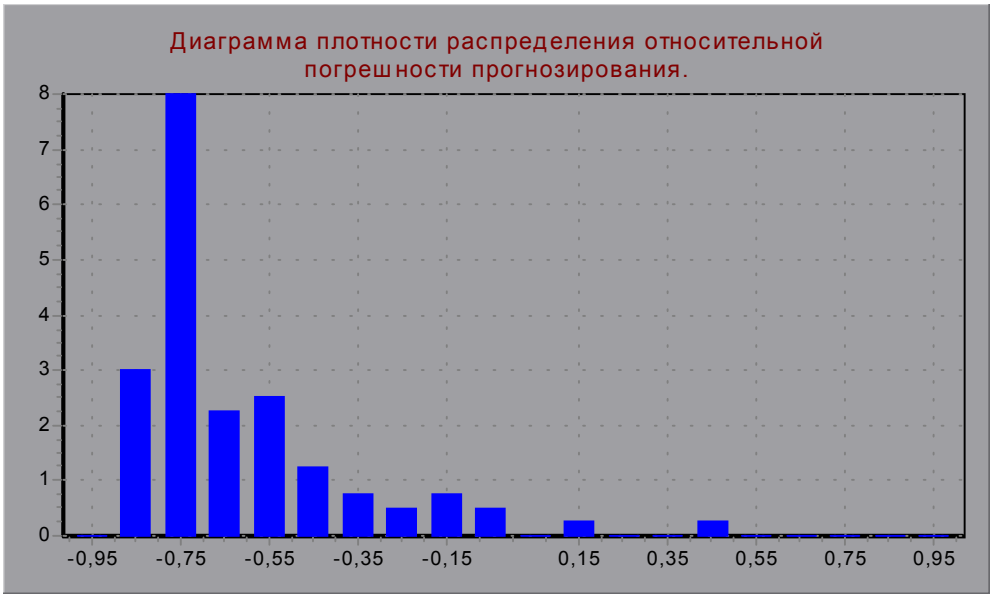

Рис.4.11 Плотность распределения относительной ошибки долгосрочного прогноза сети Хопфилда на тестовой выборке

Сеть Хопфилда очень плохо справилась с долгосрочного прогнозирования. Результат оказался хуже, чем при краткосрочном прогнозировании: прогноз получился более смещенным, несмотря на то, что число образов, которые предъявлялись сети при обучении намного меньше.

4.4.2 Решение с помощью перцептрона

Как уже отмечалось, долгосрочное прогнозирование намного сложнее краткосрочного. Результаты экспериментов подтвердили это предположение. В испытаниях использовались перцептроны с различной архитектурой, но лучше всего обучился перцептрон с одним скрытым слоем. Обучающая выборка состояла из данных за 25 лет, а тестовая – из данных за оставшиеся 80 лет. Входными данными являлся приток реки за все месяцы пяти лет, предшествующих прогнозируемому году. Выходом являлся приток реки за апрель. Результаты экспериментов приведены в таблице (табл.4.4).

Таблица 4.4 Результат долгосрочного прогнозирования с помощью перцептрона

Выборка.	Среднее значение относительной ошибки прогноза.	Среднеквадратичное отклонение относительной ошибки.	Ссылка на иллюстрацию.
Обучающая	0.09	0.4	рис.4.12
Тестовая	0.19	0.81	рис.4.13

Рис.4.12 Плотность распределения относительной ошибки долгосрочного прогноза перцептрона на обучающей выборке

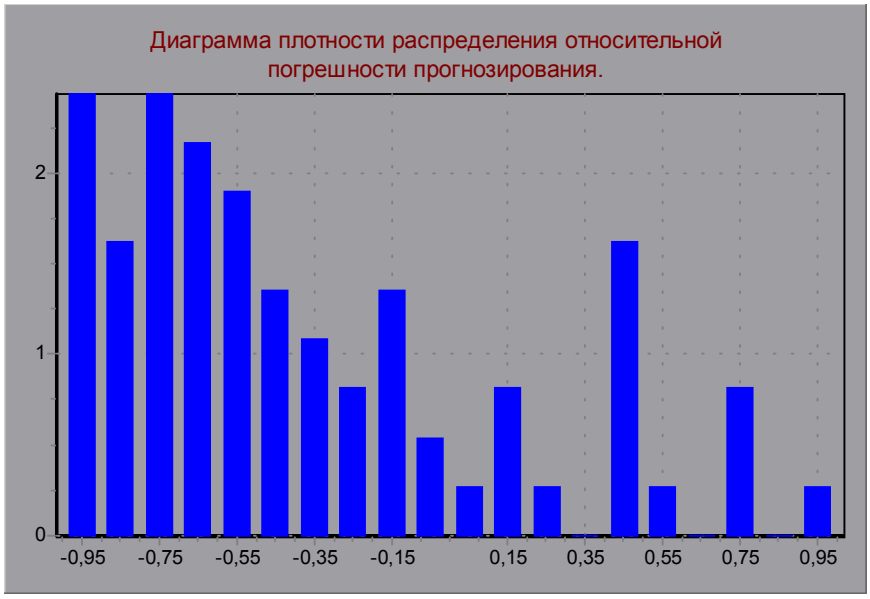

Рис.4.13 Плотность распределения относительной ошибки долгосрочного прогноза перцептрона на тестовой выборке

Полученные результаты показывают, что сеть успешно запомнила обучающую выборку (прогноз на ней получился почти несмещенным и точным). Прогноз на тестовой выборке оказался неудовлетворительным. Числовые характеристики и диаграмма (рис.4.12) говорят о смещенном и неточном прогнозе. Вид распределения позволяет предположить, что сеть не смогла выявить закономерность изменения притока, так как не прослеживается тенденции к точному прогнозу.

4.5 Анализ полученных результатов и выводы

Применение программы для анализа баз данных с помощью нейронных сетей для прогнозирования притока реки Обь показало, что сеть Хопфилда практически непригодна для решения этой задачи. Ни краткосрочный, ни долгосрочный прогноз этой сети не является точным. Сеть работает очень медленно и имеет склонность к параличу.

Предположение о том, что перцептрон даст более качественный прогноз, подтвердилось, но только для краткосрочного предсказания. Долгосрочный прогноз оказался неудовлетворительным. Возможно, причиной тому является более сложная закономерность долгосрочного изменения притока реки, отмеченная при предварительном изучении исходных данных. Обучаемость перцептрона оказалась достаточно высокой. Перцептрон легко запоминает обучающую выборку и воспроизводит ее с высокой точностью (это справедливо и для краткосрочного и для долгосрочного прогнозирования). Однако для успешного обучения сети необходимо подобрать подходящую архитектуру сети, что представляет собой нетривиальную задачу - процесс выбора архитектуры сети носит итеративный характер, поскольку искомое решение очень сильно зависит от условий решаемой задачи.

Помимо выводов о задаче прогнозирования притока реки Обь и особенностей перцептрона, можно сделать определенные выводы и об инструменте, используемом для прогнозирования. Встроенная в программу для анализа баз данных модель перцептрона оказалась легко обучаемой и способной к обобщению. Полезность приложения, по сравнению с версией, располагающей только сетью Хопфилда, повысилась, что подтверждает правильность решения о реализации новой модели нейронной сети (перцептрона). Таким образом, приложение может быть использовано и для решения различных задач. Например, для обработки результатов тестов с открытым ответом в системе тестирования “Intertest” [22].

Заключение

При выполнении работы была разработана архитектура программного обеспечения и разработана программа для анализа баз данных с помощь нейронных сетей. Разработанная архитектура позволяет сравнительно легко подключать новые модели нейронной сети к оболочке и имеет удобные средства для работы с базами данных, нейронными сетями и результатами работы обученных нейронных сетей.

В соответствии с поставленной целью программа позволяет, как проводить различные типы анализа (прогноз, классификация, кластеризация, поиск ассоциаций), так и исследовать модели нейронных сетей. Программа может извлекать исходные данные из любой базы данных, поддерживаемой BDE, формировать отчеты по результатам анализа, которые позволяют оценить качество работы сети, экспортировать результаты своей работы, а также динамически подключать различные модели нейронных сетей. Таким образом, все поставленные задачи были решены.

Также было проведено исследование зависимости между архитектурой многослойного перцептрона и задачей, которая может быть представлена с помощью этой сети. В результате был получен набор рекомендаций, позволяющих выбрать архитектуру сети, достаточную для решения конкретной задачи классификации.

Разработанная программа была испытана на примере решения задачи прогнозирования притока реки Обь. В результате эксперимента было показано, что программу можно использовать для решения такого рода задач, и что перцептрон обладает большими возможностями, чем сеть Хопфилда

По результатам исследования перцептрона и применения программы для прогноза реки Обь были сделаны следующие выводы:

1) Для решения задачи классификации с помощью многослойного перцептрона с пороговыми функциями активации достаточно двух скрытых слоев нейронов.

2) При использовании в сети нейронов, не имеющих параметра, задающего порог срабатывания, введение в сеть дополнительного входа позволит расширить возможности сети.

3) Применение сети Хопфилда ограничивается сильной зависимостью между количеством входов сети и информационной емкостью. Чтобы повысить емкость требуется увеличить число входов, которое определяется условиями задачи.

Литература

1) Мак-Каллок У.С., Питтс У. Логическое исчисление идей, относящихся к нервной активности // Автоматы, под ред. Шеннона К.Э. и Маккарти Дж. М.: ИЛ, 1956. С. 362 - 384.

2) Розенблатт Ф. Принципы нейродинамики. Перцептрон и теория механизмов мозга. М.: Мир, 1965. 480 с.

3) Минский М., Пейперт С. Персептроны. - М.: Мир, 1971. -261 с.

4) J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", in Proc. National Academy of Sciencies, USA 79, 1982, pp. 2554-2558.

5) Rumelhart D. E., Hinton G. E., Williams R. J. Learning representations by back-propagating errors // Nature (London). - 1986. - N 323. - P. 533-536.

6) Werbos P. J. Beyond regression: New tools for prediction and analysis in the behavioral science: Ph.D. Thesis. — Harvard University, Cambridge, MA, 1974. —120 p.

7) J. Hertz, A. Krogh, and R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, Mass., 1991. – 293 p.

8) J.A. Anderson and E. Rosenfeld, "Neurocomputing: Foundation of Research", MIT Press, Cambridge, Mass., 1988. – 267 p.

9) Hebb D. O. The Organization of Behavior: A Neuropsychological Theory. — New York: Wiley, 1949. - 358 p.

10) G.A.Carpenter and S. Grossberg, Pattern Recognition by SelfOrganizing Neural Networks, MIT Press, Cambridge, Mass., 1991. – 328 p.

11) Киселев М., Соломатин Е. Средства добычи знаний в бизнесе и финансах. - // Открытые системы, 1997, N4, стр. 41-44.

12) Johnston S.J. Анализ данных увеличивает доход банков. - // Банковские системы, 1997, N04, с.36.

13) Шапот М. Интеллектуальный анализ данных в системах поддержки принятия решений. - // Открытые системы, 1998, N1, с. 30-35.

14) Продукты для интеллектуального анализа данных. - //Рынок программных средств, 1997, N14-15, с.32-39.

15) Кречетов Н.. Продукты для интеллектуального анализа данных. — Рынок программных средств, № 14–15, 1997, с. 32–39.

16) Ф. Уоссермен. Нейрокомпьютерная техника: Теория и практика. – М.: Мир, 1992. – 240 с.

17) Губарев В.В., Альсова О.К., Беленький А.И., Гаврилов А.В., Голованский А.П., Давыдова Т.Н., Канглер В.М. Управление Новосибирским водохранилищем на основе прогнозирования притока. - // Водное хозяйство России. Проблемы, технологии, управление, Екатеринбург, Изд-во РосНИИВХ, 2000, т. 2, № 5. - С. 484-499.

18) Гаврилов А.В. Гибридные интеллектуальные системы. – Новосибирск: Изд-во НГТУ,2003. – 164 с.

19) Гаврилов А.В., Канглер В.М. Использование искусственных нейронных сетей для анализа данных. - // Сб. научн. трудов НГТУ. - Новосибирск: Изд-во НГТУ, 1999. - № 3(16). - С. 56-63.

20) Гаврилов А.В., Губарев В.В. Применение модели Хопфилда для решения задачи прогнозирования на примере анализа притока реки Обь. // 2-я Всероссийская научно-техн. конф. "Нейроинформатика-2000", М., 2000. - С. 33-38.

21) Круглов В.В. Борисов В.В. Искусственные нейронные сети. Теория и практика. – М.: Горячая линия – Телеком, 2001 – 382 с.

22) Гаврилов А.В., Зайцев С.А., Макаревич Л.Г., Романов Е.Л. Автоматизированная система тестирования знаний в среде Internet/Intranet. – //Открытое и дистанционное образование, №1(3), 2001. - С. 49-51.