ФГБОУ ВО «Воронежский государственный
технический университет»

Л.В. Хливненко

ПРАКТИКА НЕЙРОСЕТЕВОГО МОДЕЛИРОВАНИЯ

Воронеж 2015

УДК 004.032.26

Хливненко Л.В. Практика нейросетевого моделирования: монография / Л.В. Хливненко. – Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2015. – 214 с.

ISBN 978-5-7731-0429-2

В монографии рассмотрены теоретические основы моделирования искусственных нейронных сетей различной архитектуры. Приведены алгоритмы обучения однослойных и многослойных сетей прямого распространения, самоорганизующихся и рекуррентных сетей. Рассмотрено моделирование многоагентных систем на основе эволюционирующих нейронных сетей. Приводятся оригинальные методики визуализации внутреннего состояния обученной нейронной сети и решения задач классификации, категоризации, прогнозирования, восстановления зашумленной информации. Даны методологические основы проектирования нейросетевых модулей решения задач в виде компьютерных приложений. Приведены описания структур, интерфейсов и компьютерные коды основных блоков нейросетевых приложений. Описаны методы комбинирования градиентных и стохастических алгоритмов обучения для повышения эффективности решения практических задач. Приводятся оригинальные методики решения задач распознавания образов, прогнозирования курсов валют, задач медицинской диагностики. Рассмотрены методы и способы оценки эффективности разработанных нейросетевых моделей.

Издание предназначено для научных работников, специализирующихся в области разработки автоматизированных систем искусственного интеллекта и когнитивного моделирования процессов принятия решений, может быть полезно студентам и аспирантам соответствующих специальностей.

Табл. 4. Ил. 120. Библиогр.: 102 назв.

Рецензенты: кафедра нелинейных колебаний Воронежского государственного университета (д-р физ.-мат. наук, проф. В.Г. Задорожний); д-р техн. наук, проф. И.Ф. Астахова

ISBN 978-5-7731-0429-2

ПРЕДИСЛОВИЕ

Искусственные нейронные сети (ИНС) вызывают интерес у широкого круга исследователей. Технологию нейросетевого моделирования используют при изучении плохо структурируемых и формализуемых систем.

При исследовании сложных систем (биологических, социальных, политических, экономических, исторических и др.) исследователя часто интересуют общие закономерности и тенденции развития самой системы, а не отдельных ее элементов. Выделение эмерджентных свойств системы сопряжено с рядом сложностей, которые могут быть связаны с излишней детализацией и большим количеством связей между элементами системы.

Воздействие на одну часть системы приводит к изменениям в других ее частях и почти всегда сопровождается побочными эффектами. Возникающие при этом петли обратной связи сопряжены с задержками во времени. Результат управляющего воздействия не проявляется мгновенно и часто не соответствует цели управления. Чем сложнее организована система, тем более она инертна и устойчива, тем сложнее найти ее точки бифуркации.

Решение парадокса моделирования надежной системы из большого количества ненадежных элементов подсказано самой природой. Биологические нейронные сети при некритическом поражении способны самостоятельно восстанавливаться. При этом сохранившие жизнеспособность нейроны берут на себя функции потерянной части системы.

Способность к самовозрождению сложной системы обусловлена распределением хранения знаний, децентрализованной стратегией управления и коллективным принятием решений. Всеми перечисленными свойствами обладают математические модели, в основу которых положена технология искусственных нейронных сетей.

Многие вопросы, связанные с практическим применением ИНС, исследованы недостаточно полно. Большинство работ узко специализированы и связаны с конкретной предметной областью. Теоретические обзоры по нейросетевому моделированию, как правило, не содержат практических рекомендаций по реализации нейросетевых алгоритмов.

Практически отсутствуют обзоры различных модификаций ИНС с методиками разработки компьютерных приложений для их апробации и исследования.

Зачастую исследователи выбирают готовые инструментальные средства моделирования, которые, как правило, являются зарубежными коммерческими продуктами с закрытыми исходными кодами. В этом случае исследователю достается пассивная роль манипулятора настройками конфигурации и обучения сети.

Целью данной работы является обобщение накопленного автором опыта в области компьютерного моделирования ИНС и выработка методики разработки нейросетевых приложений для решения широкого круга прикладных задач.

В соответствии с указанной целью в рамках данной монографии поставлены и решались следующие задачи:

1. Обзор проблематики нейросетевого моделирования.

2. Разработка программ для реализации однослойных, многослойных, стохастических, самоорганизующихся, рекуррентных и эволюционирующих ИНС в свободно распространяемой среде программирования Lazarus.

3. Применение нейросетевых алгоритмов к решению задач медицинской диагностики, распознавания образов, восстановления данных, классификации, категоризации, прогнозирования.

Автор будет признателен читателям за отзывы и рекомендации по улучшению данной работы, которые можно направлять по адресу: *hlivnenko_lv@mail.ru*

ГЛАВА I. ПРОБЛЕМАТИКА НЕЙРОСЕТЕВОГО МОДЕЛИРОВАНИЯ

ИССКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ: что это такое и зачем они нужны?

Компьютерные технологии столь стремительно развиваются, что если бы также изменялось автомобилестроение, то обыватели ездили бы на одноразовых автомобилях со скоростью звука.

Элементная база традиционного компьютера настолько измельчала, что на скорость и производительность новых модификаций компьютерных устройств начинают влиять такие неустранимые факторы как, например, размеры и скорость движения электронов.

Автором первой научной работы по цифровым ЭВМ принято считать Джона фон Неймана, который в 1945 г. сформулировал основные принципы работы и определил компоненты современных компьютеров, другими словами – описал их архитектуру, которая остается классической до сих пор.

Компьютеры, считающиеся фон-неймановскими, построены на принципах двоичного кодирования, программного управления, хранимой программы, однотипности представления чисел и команд, иерархичности и адресности памяти. Таким образом, традиционный компьютер представляет собой, образно говоря, набор устройств, которым управляет набор программ.

О том, как нужно составлять программы и организовывать вычислительный процесс, впервые написал Алан Тьюринг в 1943 г. в концепции абстрактной вычислительной машины.

Главным недостатком традиционного компьютера является то, что он является всего лишь быстрым исполнителем, а

не творцом. Традиционный компьютер не способен сделать больше того, что заложено в него программистом.

Известен парадокс брадобрея, который бреет всех тех, кто не бреет себя. Парадокс заключается в том, что на вопрос: бреет ли брадобрей сам себя? – нет однозначного ответа. Точно также нет и универсального алгоритма, который служил бы генератором алгоритмов решения других задач.

А как же тогда решать на компьютере плохо формализуемые задачи, с которыми без труда справляется человек, и часто не может объяснить, как он это делает.

Попробуйте объяснить, по каким логическим правилам вы отличаете одного своего знакомого от другого? Почему люди другой национальности вам кажутся похожими друг на друга? Почему одна музыкальная композиция вам кажется приятной, а другая нет? Что является эталоном симпатичного человека? Чем отличается запах персика от запаха груши?

Сравнение нейрокомпьютера и традиционного компьютера

Прототип	Архитектура	Вычислительный процесс	Примечание
Нейро-компьютер	У.МакКаллок, У.Питтс (1943 г.)	Д. Хебб (1949 г.) Правило обучения	Моделирование работы правого полушария мозга
Традиционный компьютер	Дж. фон Нейман (1945 г. - EDVAC)	А. Тьюринг (1936 г.) Машина Тьюринга	Моделирование работы левого полушария мозга

В решении когнитивных (познавательных) задач принимают участие оба полушария человеческого мозга. Благодаря работам американского врача Орнстайна стало общеизвестно, что левое полушарие специализируется на абстрактном мышлении, которое опирается на законы логики. Правое полушарие мозга оперирует образами – матрицами или шаблонами, формирующимися в процессе опыта, обучения, тренировки.

Традиционные фон-неймановские компьютеры и нейрокомпьютеры отличаются так же, как отличаются два полушария человеческого мозга.

Искусственная нейронная сеть представляет собой распределенный параллельный процессор, состоящий из элементарных единиц обработки информации (искусственных нейронов), накапливающих экспериментальные знания и представляющий их для последующей обработки [2].

Искусственные нейронные сети являются популярным сегодня направлением в распознавании образов. Хотя зародилось это направление в тоже время, когда появились первые работы по традиционным компьютерам. Зададимся вопросом: почему же всплеск интереса к искусственным нейронным сетям наблюдается только в последние десятилетия?

Первой работой по нейрокомпьютерам считается статья Уоррена МакКаллока и Уолтера Питтса «Логическое исчисление идей, относящихся к нервной активности», опубликованная в 1943 г.

В статье была описана простейшая математическая модель искусственного нейрона с пороговой функцией активации.

Рис. 1.1. Портрет Д. Хебба, описавшего первое правило обучения искусственных нейронов

Рис. 1.2. Портрет Ф. Розенблатта, создателя первого персептрона

Первое правило обучения искусственных нейронов описал в 1949 г. канадский биолог Дональд Хебб, заметив, что сила связи между биологическими нейронами изменяется в соответствии с их активностью.

Фрэнк Розенблатт в 1957 г. описал архитектуру искусственной нейронной сети, названную персептроном, которая до сих пор остается основной «рабочей лошадкой» в практике нейросетевого моделирования. В начале шестидесятых годов прошлого века на базе персептрона были построены действующие системы распознавания образов.

Проведем параллели с развитием элементной базы компьютеров того времени. В основном используются ЭВМ первого поколения на электронных лампах. В СССР создаются и эксплуатируются БЭСМ и МЭСМ (большие и малые электронные счетные машины). Справедливости ради отметим, что расчеты, выполненные на этих ЭВМ, позволили запустить первый искусственный спутник земли и первого человека в космос.

Появляются транзисторные компьютеры с полупроводниковой элементной базой, которые приходят на смену ламповым ЭВМ. Быстродействие увеличилось до целых 100-500 тысяч (!) операций в секунду.

Одна интегральная схема третьего поколения (1970-е – 1980-е годы) оказалась способна заменить тысячи транзисторов. Первую интегральную схему Джек Килби собрал в 1953 г., а Нобелевскую премию за это изобретение получил в 2000-м году вместе с нашим соотечественником Жоресом Ивановичем Алферовым за вклад в развитие информационных технологий.

В 1969 г. была опубликована работа Марвина Минского и Сеймура Пейперта, в которой однослойные искусственные нейронные сети подверглись критике. Научный поиск в области нейросетевого моделирования был приостановлен на два десятилетия.

Основной причиной угасания интереса к искусственным нейронным сетям, на наш взгляд, было отсутствие компьютерной платформы, на базе которой можно было реализовы-

вать эффективные нейросетевые алгоритмы распознавания образов.

С середины 80-х годов прошлого века направление «искусственные нейронные сети» возродилось заново. Появились эффективные методы обучения многослойных сетей, разработанные к 1986 г. различными авторами независимо друг от друга. Джон Хопфилд опубликовал в 1982-1986 гг. работы по рекуррентым многослойным сетям, способным решать задачи восстановления зашумленной информации [94].

В настоящее время нейроэмуляторы, нейрочипы, нейроплаты и нейрокомпьютеры разрабатываются в лабораториях ведущих университетов и многих известных фирм.

Для каких целей? Какие задачи могут решаться с помощью искусственных нейронных сетей? В каких областях человеческой жизнедеятельности могут быть использованы нейросетевые решения?

Очертим некоторый круг задач, решаемых искусственными нейронными сетями.

1. Аппроксимация функций по набору точек (регрессия). В математической статистике хорошо изучены линейные регрессионные математические модели, а также модели, допускающие линеаризацию. Искусственные нейронные сети позволяют приближать неизвестные нелинейные законы взаимосвязи изучаемых признаков.

2. Классификация данных по заданному набору классов. В задачах классификации требуется отнести новый образец к некоторой группе. Искусственные нейронные сети позволяют формировать правила принятия решений в процессе обучения. В задачах такого сорта, как правило, нужно иметь обучающую выборку данных, которые были расклассифицированы экспертом в изучаемом вопросе.

3. Кластеризация данных с выявлением заранее неизвестных классов. Решение задачи кластеризации данных по-

зволяет разделить имеющуюся неупорядоченную информацию на несколько групп (кластеров, классов) с образцами, сходными по некоторым параметрам.

4. Сжатие информации. Искусственные нейронные сети позволяют выделить из имеющейся информации значимую часть (меньшую по размеру) и, передав ее по каналам связи, на принимающей стороне восстановить из части целое (большее по размеру), возможно, с небольшими искажениями.

5. Восстановление утраченных данных. Искусственные нейронные сети позволяют запоминать в памяти некоторое количество эталонов информации, а затем распознавать и восстанавливать зашумленное информационное сообщение по имеющимся образцам.

6. Оптимизация, оптимальное управление. Процесс обучения искусственных нейронных сетей часто связан с поиском оптимальных значений целевых функций, что позволяет применять их в однокритериальных и многокритериальных задачах принятия оптимальных решений [28].

Перечисленные задачи находят своё применение в многочисленных отраслях жизнедеятельности. Отметим некоторые области применения искусственных нейронных сетей.

В экономике и бизнесе технология искусственных нейронных сетей может быть использована для предсказания рынков, оценки рисков невозврата кредитов, предсказания банкротств, оценки стоимости недвижимости, автоматического считывания и распознавания чеков и документов [17].

В медицине искусственные нейронные сети применяются при диагностике, в процессе обработки медицинских изображений, для мониторинга состояния пациента, для факторного анализа эффективности лечения, для обобщения экспериментальных данных, при фильтрации показаний приборов от шумов.

В авионике к областям применения искусственных нейронных сетей можно отнести обучаемые автопилоты, распознавание сигналов радаров, адаптивное пилотирование сильно поврежденного самолета, автоматическую навигацию беспилотных летательных аппаратов.

В связи искусственные нейронные сети применяются в процессе сжатия информации, быстрого кодирования-декодирования, оптимизации маршрутов навигации пакетов [23].

В интернет-технологиях искусственные нейронные сети могут быть использованы для ассоциативного поиска информации, в качестве агентов пользователя в сети, для фильтрации информации и распознавания спама, для адресной рекламы и маркетинга в электронной торговле.

В процессах автоматизации производства искусственные нейронные сети применяются для оптимизации режимов производственного процесса, при контроле качества продукции, для предупреждения аварийных ситуаций, в робототехнике [88].

В политологических и социологических исследованиях искусственные нейронные сети используются для предсказания результатов выборов, анализа социологических опросов, выявления значимых факторов, для объективной кластеризации электората [66, 78].

В безопасности и охранных системах искусственные нейронные сети применяются для идентификации личности по отпечаткам пальцев, голосу, подписи, лицу. С помощью этой технологии возможно в автоматическом режиме распознавать голоса [90], лица в толпе, автомобильные номера по изображению с видеокамеры [10, 11]. Искусственные нейронные сети используются также при анализе аэрокосмических снимков, мониторинге информационных потоков в компьютерных сетях и обнаружении вторжений.

Приведенный перечень областей применения искусственных нейронных сетей и решаемых ими задач далеко не является полным. Он даёт представление о спектре практических приложений этой технологии компьютерного моделирования, достоинствами которой являются универсальность и робастность [37].

БИОЛОГИЧЕСКИЕ НЕЙРОННЫЕ СЕТИ: чем обусловлено название?

Название «искусственные нейронные сети» связано с попыткой моделирования биологических процессов обработки сигналов, происходящих в нервных клетках.

Нервная система человека представляет собой трехступенчатую структуру. Центром этой системы является мозг, представленный сетью нейронов. Мозг получает информацию, анализирует ее и принимает решения.

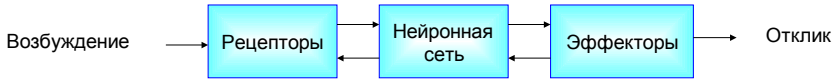


Рис. 1.3. Блочная диаграмма нервной системы

Входами биологической нейронной сети служат рецепторы, воспринимающие сигналы внешнего мира. Сигналы, полученные рецепторами, обрабатываются нейронной сетью. В результате генерируется управляющий сигнал и эффекторный орган выполняет некоторое действие, служащее откликом на внешнее раздражение. Примерно по такой схеме, мы рефлекторно отдёргиваем руку от горячего предмета при случайном соприкосновении с ним.

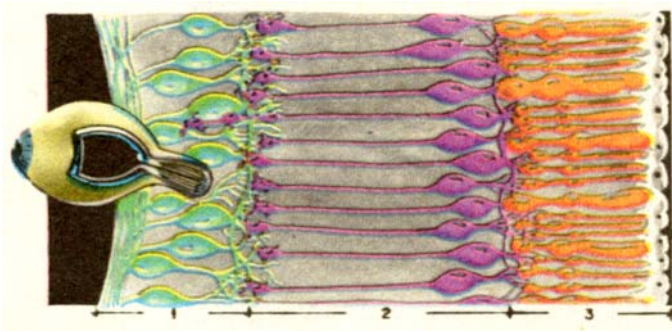


Рис. 1.4. Многослойная структура сетчатки глаза

Многослойной биологической нейронной сетью является сетчатка глаза, которая состоит из трех слоев:

1. колбочки и палочки играют роль рецепторов и реагируют на световые фотоны;
2. промежуточный слой нейронов служит для сбора информации с первого слоя нейронов;
3. наружный слой нейронов связан с нейронами промежуточного слоя. Отростки наружных нейронов формируют зрительный нерв, который передает информацию в головной мозг.

Биологический нейрон состоит из тела клетки (*сомы*) и отростков: дендритов и аксона, служащих каналами входа и выхода информации.

Дендритами называют элементы нервной клетки, выполняющие роль зоны приема информации от других клеток. Нейрон имеет множество дендритов, объединенных в дендритные деревья.

Аксоном называется отросток, играющий роль выходного канала информации. Аксон имеет множество ответвлений, служащих для контакта с дендритами других нейронов.

Место контакта называется *синапсом*, через него передается электрохимический импульс между нейронами. Чем шире место контакта, тем сильнее импульс.

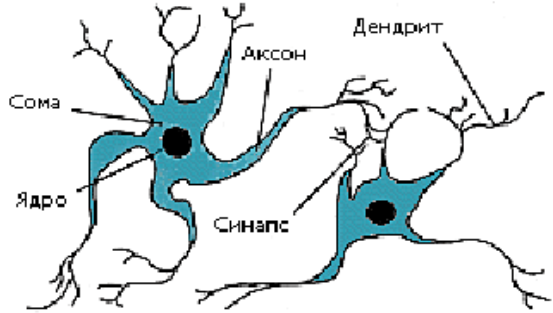


Рис. 1.5. Структурные элементы биологического нейрона

Таким образом, входы нейрона неравноправны и определяются силой синаптической связи, которая изменяется в процессе жизнедеятельности.

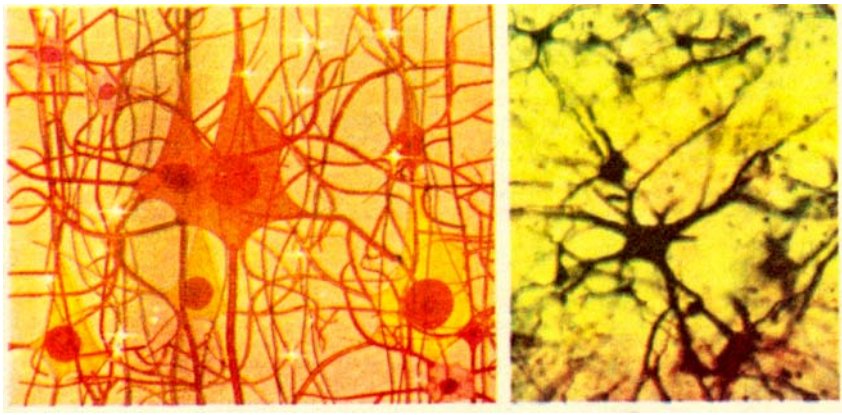


Рис. 1.6. Фотография биологической нейронной сети

Сигналы, подводимые к телу нейрона через синапсы и дендриты, могут возбуждать или затормаживать его. На мембране нейрона подводимые электрохимические сигналы суммируются с учетом их полярности. Положительные сигналы стремятся возбудить нейрон, отрицательные — воспрепятствовать возбуждению. Если суммарное возбуждение в теле нейрона превышает пороговый уровень, происходит «вклю-

чение» нейрона. При этом на аксоне генерируется электрохимический импульс.

Размеры нейронов варьируются от 5 до 120 мкм, время отклика составляет около 2 мс.

Нервная система представляет собой самоорганизующуюся иерархическую структуру, которая на нижнем уровне представлена молекулами. На следующем уровне иерархии находятся синапсы, формирующие нейронные микроконтуры, играющие роль биологических чипов, служащих для сбора информации.

Следующий уровень в иерархии представлен дендритными деревьями, подводящими информационные сигналы к нейронам. Нейроны со схожими функциями формируют локальные контуры. Дирижерами разных функций служат межрегиональные контуры. На высшей ступени иерархии находится феномен, называемый центральной нервной системой.

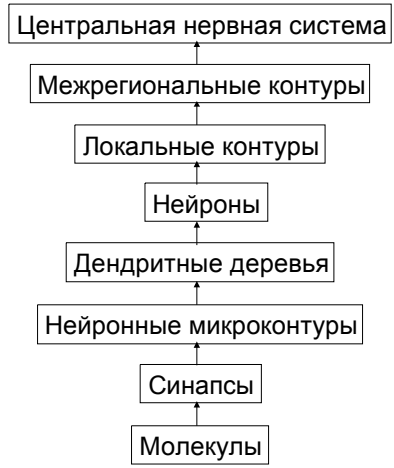


Рис. 1.7. Структурная организация уровней мозга

Приведенное описание является сильно упрощенным и далеко не полным. Оно дано с целью пояснения смысла био-

логических терминов и понятий, используемых при описании искусственных нейронных сетей.

Чем больше узнают нейрофизиологи о работе головного мозга, тем более эффективные модели могут быть построены на базе искусственных нейронных сетей.

Кора больших полушарий головного мозга состоит из различных цитоархитектонических полей. В пределах отдельного цитоархитектонического поля кора имеет близкое клеточное строение. Функции конкретной зоны коры определяются обработкой на ее колонках различных сигналов: поступающих из других областей коры или от других структур мозга. Колонки могут временно объединяться в более крупные системы обработки информации.

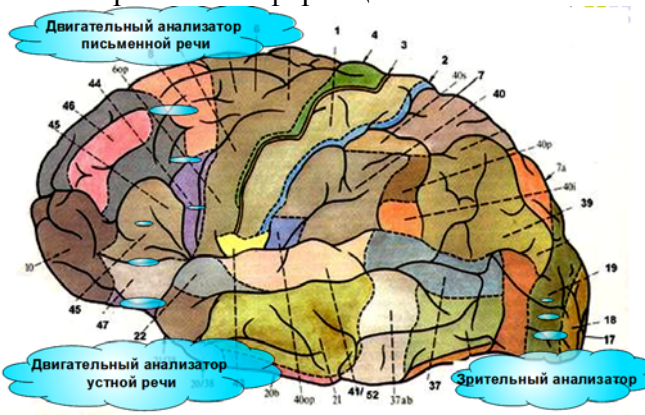


Рис. 1.8. Цитоархитектонические поля коры полушарий

В затылочной доле находится зона зрения, в которой происходит анализ и распознавание зрительных образов.

Часто это происходит без подключения логики сознания. В большинстве случаев сознание не контролирует также работу вегетативной нервной системы – отдела нервной системы, регулирующего деятельность внутренних органов, желёз внутренней и внешней секреции, кровеносных и лимфатических сосудов. Хотя симпатические и парасимпатические

центры вегетативной нервной системы находятся под контролем коры больших полушарий и гипоталамических центров.

Моделирование искусственных нейронных сетей представляет собой попытку копирования процессов прохождения и обработки информационных сигналов в биологических нейронных сетях при решении конкретной задачи или группы задач [86].

Направление «искусственные нейронные сети» развивалось силами специалистов разных областей: математиками, биологами, нейрофизиологами, физиками, инженерами. Поэтому устоявшаяся терминология в этой области представляет собой смешение понятий из различных наук.

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ: основные понятия

Единицей обработки информации в сети выступает искусственный нейрон, в состав которого обычно входят синаптические веса, сумматор и функция активации.

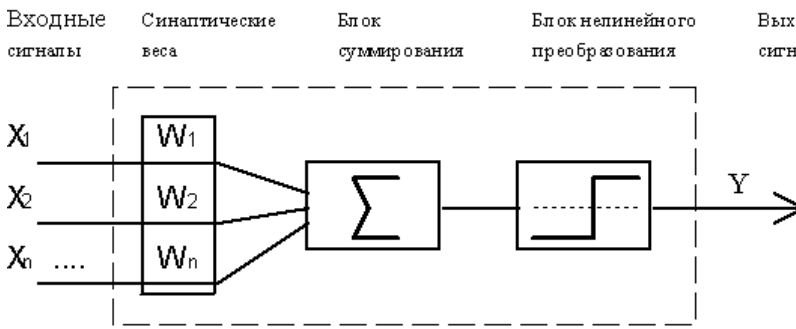


Рис. 1.9. Схема искусственного нейрона

На вход нейрона подается входной вектор $X(x_1,...,x_n)$. Каждая координата входного вектора ослабляется или усили-

вается умножением на синаптический весовой коэффициент ω_i, $i=\overline{1,n}$.

Сумматор складывает входные сигналы, взвешенные относительно соответствующих синапсов нейрона. Выходом блока суммирования является линейная комбинация входных сигналов $\sum_{i=1}^{n}\omega_i x_i$.

Функция активации (сжатия) ограничивает амплитуду выходного сигнала нейрона. Нормализованный диапазон амплитуд выхода блока нелинейного преобразования принадлежит отрезку [0,1] или [-1,1].

Часто используются такие функции активации, как пороговая, кусочно-линейная, сигмоидальная и гиперболический тангенс.

Модель нейрона с пороговой функцией активации называют моделью Мак-Каллока – Питтца, отдавая дань пионерской работе. Выбранный вид функции активации описывает ключевое свойство модели Мак-Каллока – Питтца: «всё или ничего».

$$y=\begin{cases}1, & x\geq T;\\ 0, & x<T.\end{cases}$$

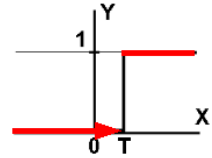


Рис. 1.10. Функция единичного скачка или пороговая функция (функция Хевисайда)

Аргументом x пороговой функции является выходной сигнал блока суммирования, называемый также индуцированным локальным полем нейрона.

Величина T определяет пороговый уровень, начиная с которого нейрон возбуждается, что соответствует появлению единичного значения на выходе нейрона.

При использовании кусочно-линейной функции активации обеспечивается плавный линейный переход выходного значения из нуля в единицу при нарастании возбуждения нейрона.

$$y = \begin{cases} 0, & x \le \xi_1; \\ k(x - \xi_1), & \xi_1 < x \le \xi_2; \\ 1, & x > \xi_2. \end{cases}$$

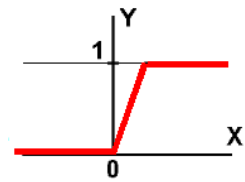


Рис. 1.11. Кусочно-линейная функция активации

Наиболее распространенными функциями активации являются сигмоидальная функция и гиперболический тангенс, поскольку они являются дифференцируемыми и их производные выражаются через значения самих функций.

Сигмоидальная функция активации имеет вид:

$$y = \frac{1}{1 + e^{-a(x-\Theta)}}.$$

Сигмоидальная функция обеспечивает нелинейный, «S» - образный переход выходного значения из нуля в единицу при нарастании возбуждения нейрона. Параметр a управляет уровнем наклона графика активационной функции. Порог Θ определяет точку перегиба графика.

Разновидностью сигмоидальной функции является логистическая функция.

$$y = \frac{1}{1 + e^{-ax}}$$

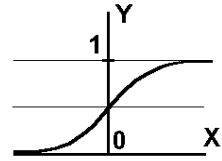


Рис. 1.12. Логистическая функция активации

Сигмоидальная функция поддерживает баланс между линейным и нелинейным поведением нейрона.

Если требуемый диапазон выходного сигнала принадлежит отрезку $[-1;1]$, то используют гиперболический тангенс.

$$\operatorname{th} x = \frac{\operatorname{sh} x}{\operatorname{ch} x} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

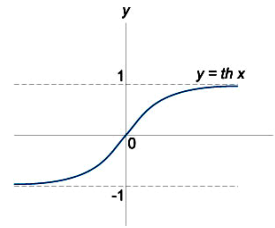


Рис. 1.13. Гиперболический тангенс

Схематически искусственные нейронные сети представляют в виде графов.

Граф передачи сигнала – это сеть направленных связей (или ветвей), соединяющих отдельные точки (узлы).

Элементами графа передачи сигнала являются синаптическая и активационная связи, синаптическая сходимость и дивергенция [65].

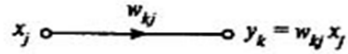


Рис. 1.14. Синаптическая связь

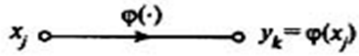


Рис. 1.15. Активационная связь

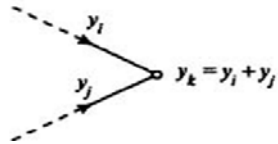


Рис. 1.16. Синаптическая сходимость

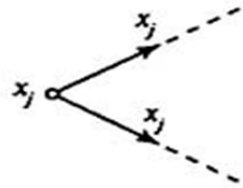


Рис. 1.17. Синаптическая дивергенция

Искусственную нейронную сеть можно представить в виде направленного графа, состоящего из узлов, соединенных синаптическими и активационными связями [65].

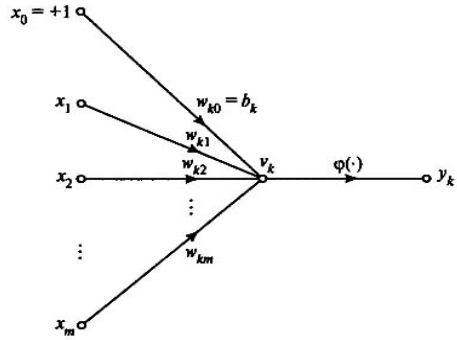


Рис. 1.18. Граф передачи сигнала внутри нейрона

Для улучшения процесса сходимости на вход нейрона подается тождественный сигнал $x_0 = +1$. В этом случае выходной сигнал сумматора имеет вид $\omega_0 + \omega_1 x_1 + ... + \omega_m x_m$. Пусть для пороговой функции активации $T = 0$. Тогда условия $\omega_0 + \omega_1 x_1 + ... + \omega_m x_m \geq 0$ и $\omega_0 + \omega_1 x_1 + ... + \omega_m x_m < 0$ позволят разделить точки двух классов пространства R^m.

При использовании комбинации линейных нейронов в гиперпространстве очерчиваются многогранники, разделяющие входные значения разных классов.

В схематическом изображении взаимодействия различных элементов нейронной сети, подробности прохождения

сигнала внутри отдельного нейрона опускают. Такие схемы называют архитектурными.

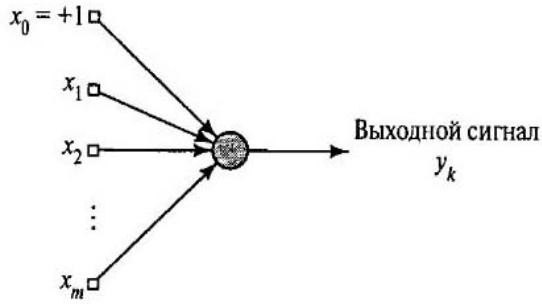


Рис. 1.19. Архитектурный граф нейрона

Основными архитектурами искусственных нейронных сетей являются модели однослойных и многослойных сетей прямого распространения сигнала, а также рекуррентные сети с обратным направлением прохождения сигнала [65].

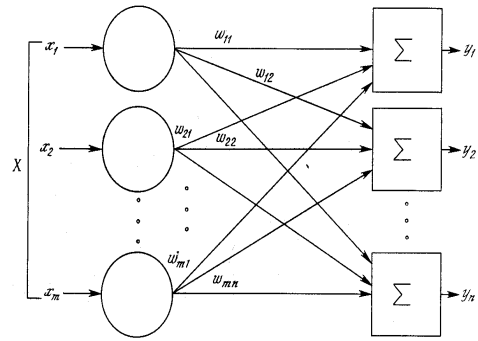


Рис. 1.20. Однослойная сеть прямого распространения

В однослойной сети информация от входного слоя X передается на выходной слой нейронов Y. Сеть называется однослойной, потому что в ней один слой вычислительных узлов Y, служащих элементами обработки информации.

Синаптические коэффициенты в однослойной сети являются элементами матрицы W размера $m \times n$.

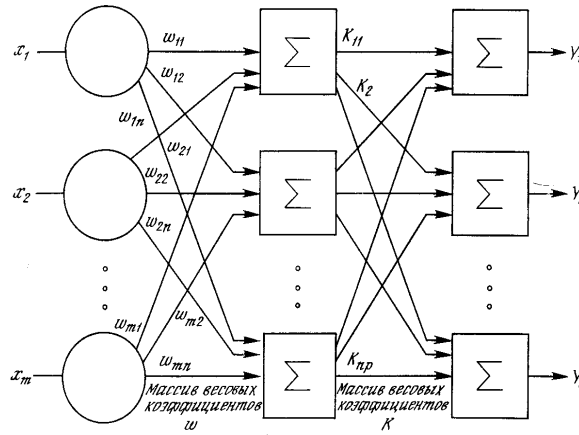


Рис. 1.21. Двухслойная сеть прямого распространения

В двухслойной сети есть скрытый слой нейронов. Сеть на рис. 1.21 называется $m - n - p$ сетью, поскольку в ней m входных элементов, n скрытых нейронов и p выходных нейронов. В такой сети две матрицы синаптических коэффициентов: матрица связей между входными элементами и скрытыми нейронами W размера $m \times n$ и матрица связей между скрытыми и выходными нейронами V размера $n \times p$.

В многослойных сетях имеется несколько слоев скрытых нейронов. Такие сети способны выявлять более сложные статистические закономерности между входными и выходными величинами.

В рекуррентных сетях имеются обратные связи. В таких сетях сигнал многократно передается с выходов нейрона обратно на их входы, вызывая циклические изменения состояния нейронов. На рис. 1.22 показана архитектура сети, состоящей из одного слоя нейронов, каждый из которых направляет свой выходной сигнал на входы остальных нейро-

нов слоя. В приведенной архитектуре отсутствуют обратная связь нейрона с самим собой.

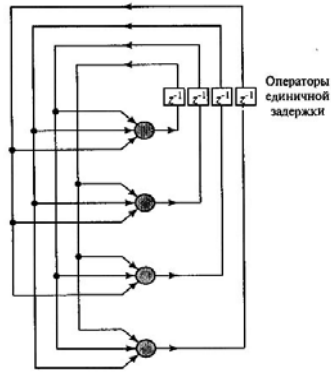


Рис. 1.22. Рекуррентная сеть

Прежде чем использовать искусственную нейронную сеть в режиме решения задачи (функционирования), её требуется обучить. Целью процесса обучения является вычисление значений синаптических (весовых) коэффициентов [13].

МЕТОД ГРАДИЕНТНОГО СПУСКА: как обучить искусственную нейронную сеть?

Концепции (парадигмы) обучения искусственных нейронных сетей бывают разными. Рассмотрим вариант «обучения с учителем», предполагающий наличие обучающей выборки.

Обучающая выборка состоит из пар векторов. Каждая пара включает входной вектор и целевой вектор, представляющий собой желаемый отклик сети при подаче на ее входы координат данного входного вектора, описывающего состояние среды.

В процессе обучения на выходах сети вычисляются текущие сигналы, называемые фактическим откликом сети.

Фактический отклик сравнивается с желаемым откликом сети. По результатам сравнения вычисляется сигнал ошибки, который учитывается при коррекции весовых коэффициентов [65].

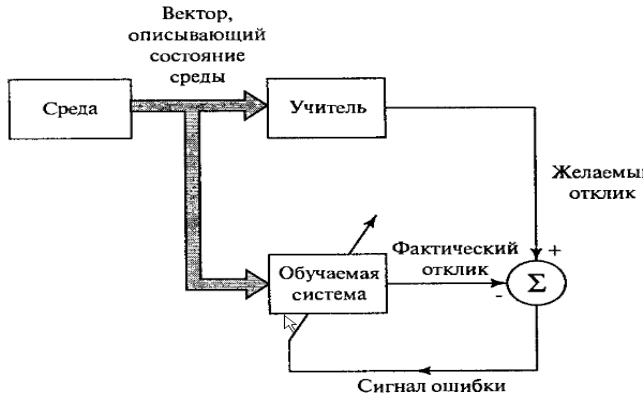


Рис. 1.23. Схема концепции «обучение с учителем»

В роли «учителя» для искусственной нейронной сети выступает эксперт, определяющий для обучающей выборки входных векторов, желаемые отклики сети.

Целью обучения является получение набора весовых коэффициентов, при котором достигается минимум ошибки между фактическим и желаемым откликами сети на множестве входных векторов из обучающей выборки.

Фактически при обучении сети подбираются значения её внутренних параметров, обеспечивающих имитацию принятия сетью решений, наиболее близких к мнению эксперта [27].

Рассмотрим вариант обучения, основанный на коррекции ошибок.

Пусть нейрон k – единственный вычислительный узел сети, $\bar{x}(n)$ – входной сигнал, n – дискретное время (номер шага в процессе настройки синаптических весов).

Выходной сигнал нейрона k – $y_k(n)$ будет сравниваться с желаемым выходом – $d_k(n)$.

Сигнал ошибки вычисляется по формуле:

$$e_k(n) = d_k(n) - y_k(n). \tag{1.1}$$

Цель достигается за счет минимизации функции ошибки, называемой также функцией стоимости или функцией энергии или индексом производительности сети E(n):

$$E(n) = \frac{1}{2} e_k^2(n) \tag{1.2}$$

Пошаговая корректировка синаптических весов нейрона k продолжается до тех пор, пока система не достигнет устойчивого состояния.

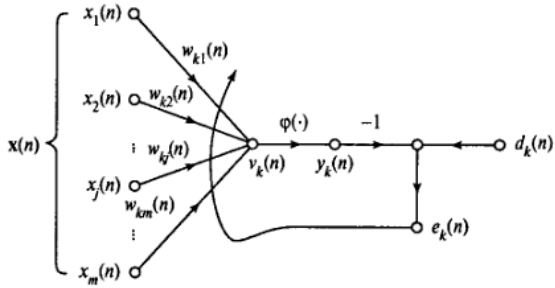


Рис. 1.24. Схема обучения нейрона k по правилу, основанному на коррекции ошибок

Вычисление коррекции весового коэффициента выполняется по дельта-правилу или правилу Видроу-Хоффа [65]:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n). \tag{1.3}$$

В формуле (1.3) положительная константа η регулирует скорость сходимости. Обычно $\eta = 0{,}01$ или $\eta = 0{,}001$.

Поясним, почему коррекция, вычисленная по формуле (1.3), приводит к уменьшению значения функции ошибки $E(n)$.

В методах безусловной оптимизации рассматривается непрерывно дифференцируемая функция стоимости $E(w)$, зависящая на каждом шаге от некоторого неизвестного вектора весов w.

Требуется отыскать такое решение w^*, что $E(w^*) \le E(w)$, т.е. необходимо решить задачу безусловной оптимизации.

Необходимым условием оптимизации является выполнение равенства $\nabla E(w^*) = 0$, где ∇ – оператор градиента.

Вектор градиента функции стоимости имеет вид:

$$\nabla E(w) = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_m} \right). \qquad (1.4)$$

В алгоритмах последовательного спуска, начиная со случайно выбранного значения $w(0)$, происходит последовательная генерация векторов весовых коэффициентов $w(1)$, $w(2)$,…, таких, что значение функции стоимости с каждой итерацией алгоритма уменьшается $E(w(n+1)) < E(w(n))$.

Возможно, такой алгоритм сойдется к оптимальному решению w^*. Причиной, не гарантирующей сходимость алгоритма, является, например, слишком большое значение константы η, приводящее к грубым коррекциям весовых коэффициентов.

В методе наискорейшего спуска корректировка весов выполняется в направлении максимального уменьшения функции стоимости, то есть в направлении, противоположном вектору градиента.

Пусть $g = \nabla E(w)$, тогда ключевая формула алгоритма наискорейшего спуска имеет вид:

$$w(n+1) = w(n) - \eta g(n). \qquad (1.5)$$

Возникает вопрос: как по выборке оценить координаты вектора градиента?

Алгоритм минимизации среднеквадратичной ошибки сети основан на использовании дискретных значений функции стоимости:

$$E(w)=\frac{1}{2}\left(\underbrace{d-x^T w}_{e}\right)^2 \tag{1.6}$$

Дифференцируя по вектору весов, имеем:

$$\frac{\partial E(w)}{\partial w(n)}=-x(n)e(n) \tag{1.7}$$

.

Получим искомую оценку вектора градиента:

$$\hat{g}(n)=-x(n)e(n) \tag{1.8}$$

Подставляя полученную оценку вектора градиента в формулу (1.5), имеем ключевую формулу для алгоритма минимизации среднеквадратичной ошибки сети:

$$\hat{w}(n+1)=\hat{w}(n)+\eta x(n)e(n) \tag{1.9}$$

.

В формуле (1.9) коррекция весового коэффициента имеет тот же вид, что и в правиле Видроу-Хоффа.

НЕЙРОСЕТЕВАЯ РЕАЛИЗАЦИЯ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ: как запрограммировать простую нейронную сеть?

В данном пункте рассматривается задача создания компьютерной реализации нейронной сети, моделирующей работу логического оператора «И».

Для поставленной задачи внешняя среда статична и описывается следующей обучающей выборкой: {(0,0);0}, {(1,0);0}, {(0,1);0}, {(1,1);1}.

Сеть будет состоять из двух входов: x_1, x_2 и одного линейного нейрона y. Схема сети показана на рис. 1.9.

Обучение сети проведем по алгоритму минимизации среднеквадратической ошибки, описанному в предыдущем пункте.

Программирование сети включает кодирование трех элементов: инициализации сети, обучения сети и работы сети в режиме функционирования.

Возникает естественный вопрос о выборе среды программирования. Известным наиболее широкому кругу читателей в нашей стране в настоящее время остается язык программирования Паскаль. Разработка консольных приложений, идейно предназначенных для очень старых версий операционных систем, не отвечает современному уровню развития технологий.

Программы, выполненные в виде оконных приложений со стандартным дружественным интерфейсом, имеют привлекательный вид. Такие программы можно создавать в среде программирования Delphi. Однако Delphi является коммерческим продуктом. Лицензионные версии программы стоят достаточно дорого, а ознакомительные часто оказываются «слегка работающими».

Отечественное инструментальное программное обеспечение, доступное широким слоям населения, пока не разработано. В какой среде программирования можно создавать конкурентоспособные приложения, не нарушая закона об авторских правах?

Несколько лет назад авторы остановили свой выбор на визуальной среде программирования Lazarus, которая разрабатывалась как свободно-распространяемый аналог Delphi. Lazarus базируется на языке программирования FreePascal, который также является диалектом традиционного Паскаля, и практически не отличим от диалекта Object Pascal, используемого в Delphi.

При желании читатель легко может адаптировать описанные ниже алгоритмы к любому другому языку програм-

мирования. В качестве среды разработки нейросетевых приложений можно выбрать Mathcad, Matlab или даже Excel, а также специализированные пакеты для нейросетевого анализа данных, например, NeuroShell2 [1].

При запуске приложения запрашиваются значения параметров, связанных с обучением сети.

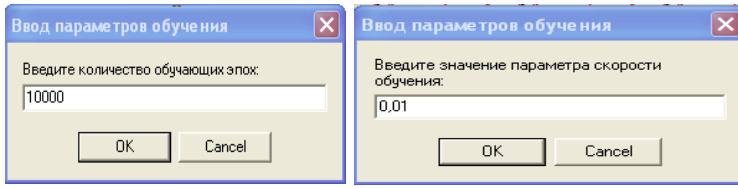


Рис.1.25. Диалоговые окна настройки параметров обучения

Обучающей эпохой называется один цикл коррекции весовых коэффициентов по всем примерам из обучающей выборки.

Описание глобальных данных имеет вид:

```
const
  s : array[1..4,1..2] of byte =
     ((0,0),(1,0),(0,1),(1,1)); // обучающее множество

var
    x : array of integer; // входной вектор
    w : array of real; // вектор весовых коэффициентов
    e : real; // сигнал ошибки
    y : real; // фактический выход сети
    r : array[1..4] of byte = (0,0,0,1); // целевой отклик сети
```

Создание и обучение сети происходит при запуске приложения. Обработчик события **OnCreate** формы имеет вид:

```
var k, i, j  : integer;   eta : real;
begin
```

```
  // настройка параметров обучения
   k := StrToInt ( InputBox ( 'Ввод параметров обучения',
 'Введите количество обучающих эпох:' , '1000' ) ) ;
   eta :=StrToFloat (InputBox ('Ввод параметров обучения',
  'Введите значение параметра скорости обучения:' , '0,01' ) ) ;

//обнуление весов
   setlength(w,3); //нумерация с единицы
   w[1] := 0; w[2] := 0;

  //инициализация входного слоя
   setlength(x,3);

  //обучение
   for i := 1 to k do
    for j := 1 to 4 do
     begin
       y := w[1] * s[j,1] + w[2] * s[j,2];
       e := r[j] - y;
       w[1] := w[1] + eta * e * s[j,1];
       w[2] := w[2] + eta * e * s[j,2];
     end;
end;
```

В результате выполнения описанной выше процедуры в памяти сформирована нейронная сеть, состоящая из одного линейного нейрона, обученного имитировать работу логического оператора «И». Протестировать работу нейрона можно, введя значения операндов в окне созданного приложения.

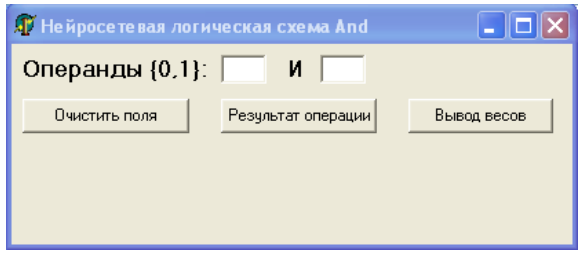


Рис.1.26. Вид приложения в режиме функционирования сети

Для удобства ввода операндов опишем обработчик события OnClick для кнопки «Очистить поля»:

```
begin
    Edit1.Text := '';
    Edit2.Text := '';
    Label3.Visible := false; // метка со знаком равенства
    Edit3.Visible := false; // поле для вывода результата
end;
```

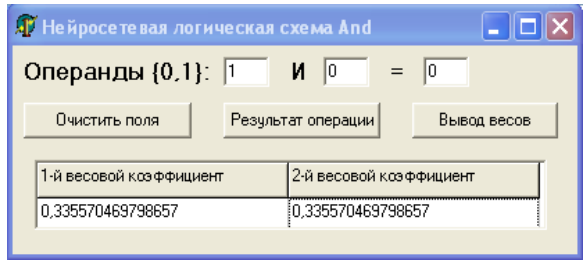


Рис.1.27. Выполнение команд «Результат операции» и «Вывод весов»

Опишем обработчик события OnClick кнопки «Результат операции», нажатием на которую можно получить фактический отклик сети при введенных значениях операндов.

```
begin

 // ввод операндов
```

```
  x[1] := StrToInt(Edit1.Text);
  x[2] := StrToInt(Edit2.Text);
  if (x[1]<0) or (x[1]>1) or (x[2]<0) or (x[2]>1)
     then
      begin
        ShowMessage('Введите правильно операнды {0,1}!');
        Exit
      end;

  //работа сети в режиме функционирования
  y :=  w[1] * x[1] + w[2] * x[2];

//вывод результата
   Label3.Visible := true;
   Edit3.Visible := true;
   Edit3.Text := IntToStr (round(y))
end;
```

Использование операции округления round() тождественно пороговой функции активации, применяемой в классической модели Мак-Каллока – Питтца.

Для отображения вычисленных весовых коэффициентов опишем обработчик события OnClick кнопки «Вывод весов».

```
begin
   StringGrid1.Visible := true;
   StringGrid1.Cells[0,1] := FloatToStr (w[1]);
   StringGrid1.Cells[1,1] := FloatToStr (w[2]);
end;
```

Результат выполнения описанной выше процедуры показан на рис. 1.27. Вычисленные значения весовых коэффициентов оказались равны друг другу. Это означает, что по-

строенная сеть инвариантна относительно двух входных векторов (1,0) и (0,1).

Рассчитанные в результате обучения весовые коэффициенты приблизительно равны $\frac{1}{3}$.

Таким образом, математическая модель работы нейрона при выполнении логической операции «И» имеет вид:

$$y = \begin{cases} 0, & \frac{1}{3}x_1 + \frac{1}{3}x_2 \le \frac{1}{2}; \\ 1, & \frac{1}{3}x_1 + \frac{1}{3}x_2 > \frac{1}{2}. \end{cases} \qquad (1.10)$$

Отложим на плоскости по оси абсцисс значения операнда x_1, по оси ординат - значения операнда x_2. Получим, что прямая, уравнение которой $x_1 + x_2 = \frac{3}{2}$, отделяет единичные выходные значения оператора от нулевых.

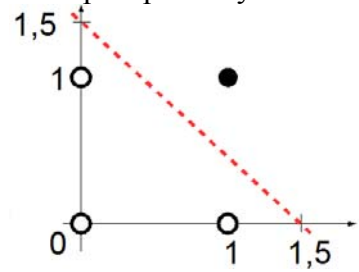


Рис.1.28. Геометрическая иллюстрация результатов обучения сети выполнению операции «И»

Проведя аналогичное исследование для логического элемента «ИЛИ», можно получить следующее уравнение разделяющей прямой: $x_1 + x_2 = \frac{1}{2}$.

Математическая модель работы нейрона при выполнении логической операции «ИЛИ» имеет вид:

$$y = \begin{cases} 0, & x_1 + x_2 \le \dfrac{1}{2}; \\ 1, & x_1 + x_2 > \dfrac{1}{2}. \end{cases} \qquad (1.11)$$

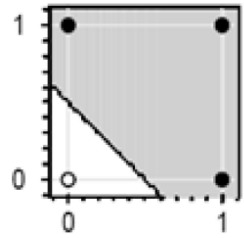


Рис.1.29. Геометрическая иллюстрация результатов обучения сети выполнению операции «ИЛИ»

При нейросетевом моделировании операции отрицания «НЕ» в модель нейрона следует добавить тождественный входной сигнал +1. В этом случае математическая модель работы нейрона примет вид:

$$y = \begin{cases} 0, & -x_1 + \dfrac{1}{2} \le 0; \\ 1, & -x_1 + \dfrac{1}{2} > 0. \end{cases} \qquad (1.12)$$

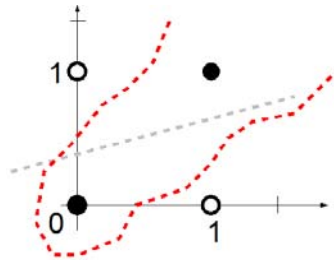


Рис.1.30. Геометрическая иллюстрация разделения единичных и нулевых результатов операции «XOR»

Логическая операция XOR (исключающее ИЛИ) не реализуема одним линейным нейроном. На рис. (1.30) показано,

что невозможно построить прямую, отделяющую единичные результаты (черные точки) выполнения операции XOR от нулевых результатов (белые точки).

Однако проблема решается при использовании нелинейного нейрона со следующей математической моделью:

$$y = \begin{cases} 0, & x_1 + x_2 - 2x_1x_2 \le \dfrac{1}{2}; \\ 1, & x_1 + x_2 - 2x_1x_2 > \dfrac{1}{2}. \end{cases} \qquad (1.13)$$

Другим вариантом реализации операции XOR является использование двухслойной суперпозиции операций И, ИЛИ и НЕ.

$$x_1 \oplus x_2 = \left[(x_1 \vee x_2) - (x_1 \wedge x_2) - \frac{1}{2} > 0 \right]. \qquad (1.14)$$

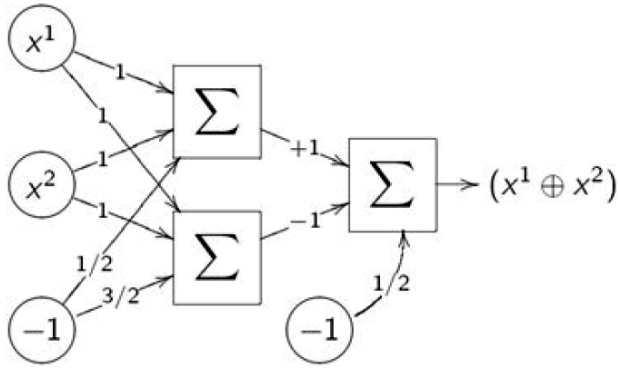


Рис.1.31. Конфигурация нейронной сети, реализующей логическую операцию XOR

Таким образом, нейронные сети позволяют реализовать базовые логические вентили, комбинации которых используются в более сложных цифровых схемах [3].

ГЛАВА II. ОДНОСЛОЙНЫЕ СЕТИ ПРЯМОГО РАСПРОСТРАНЕНИЯ

РАСПОЗНАВАНИЕ ОБРАЗОВ: как закодировать картинку?

Задача распознавания образов относится к классу когнитивных задач, с которыми с большим или меньшим успехом справляется человек. В большинстве случаев, не задумываясь о том, как он это делает, человек узнает знакомых ему людей, читает текст, называет автора известной музыкальной композиции или растение в пору его цветения. Однако с более сложными задачами распознавания, например, определение по запаху сорта манго из имеющихся в мире 300 видов, обычный человек может не справиться.

Многие интеллектуальные системы предназначены для решения задач распознавания образов. Системы машинного зрения, например, получают изображения через камеру, распознают объекты и составляют их описания в символьном виде. Интеллектуальные системы распознавания образов применяются также в медицине, биологии, геологии, дактилоскопии, при распознавании лица, подписи, жестов,… [20, 67, 80, 81].

Системы символьного распознавания позволяют различать буквы и цифры, что используется при оцифровке документов, обработке чеков и почтовой корреспонденции.

В данной главе подробно рассматривается задача распознавания цифр по их изображениям. Изображения цифр представляют собой черно-белые картинки размером 32x32 пикселя.

Рис. 2.1. Фрагмент обучающей выборки для цифры «5»

Подборка изображений состоит из 50 графических файлов .bmp и включает по пять изображений каждой цифры с разными вариантами шрифта и стиля начертания (рис.2.1).

На входы искусственной нейронной сети подается числовой вектор, который формируется в зависимости от расположения черных и белых точек на картинке. Черной точке на изображении соответствует единичный сигнал, белой - нулевой. Входной вектор содержит $32 \times 32 = 1024$ координаты.

Графические файлы в обучающей выборке именуются по шаблону: изображенная цифра + номер варианта начертания. Например, файлы с изображением цифры «5» (рис. 2.1) имеют имена 51, 52,53, 54, 55.

Таким образом, выделяя первую букву в имени файла, программа формирует для каждого вектора из обучающей выборки идеальный (целевой) выходной вектор.

Выходной слой искусственной нейронной сети содержит 10 нейронов, каждый из которых будет учиться распознавать «свою» цифру.

В целевом векторе 10 координат, девять из которых равны нулю (они отвечают за «чужие» цифры) и одна координата равна единице (она стоит на месте «своей» цифры). Например, для всех изображений цифры «5» на рис.2.1 целевые выходные векторы имеют вид $(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)$.

Распознавание цифры сетью осуществляется по максимальному сигналу нейрона выходного слоя, что приводит к выбору соответствующего переключателя в группе (рис. 2.2) .

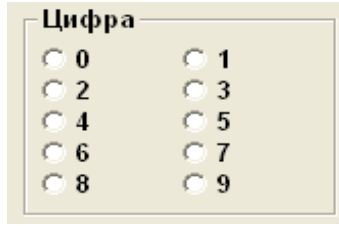


Рис.2.2. Группа зависимых переключателей для отображения результата распознавания цифры на картинке

ОБУЧЕНИЕ ОДНОСЛОЙНОЙ СЕТИ: правило Хебба

Исторически первым правилом обучения искусственной нейронной сети является правило Хебба.

Канадский биолог Дональд Хебб отметил, что если два нейрона возбуждаются одновременно, то сила синаптической связи между ними увеличивается, если возбуждение происходит асинхронно, то уменьшается.

Правило обучения Хебба

Состояние первого нейрона	**Состояние второго нейрона**	**Влияние на связь между нейронами**
Не активен	Не активен	Не влияет
Не активен	Активен	Подавление
Активен	Не активен	Подавление
Активен	Активен	Усиление

В контексте задачи распознавания цифр правило обучения Хебба означает, что связь между черными точками входного изображения и нейроном «своего» класса в процессе обучения усиливается (весовой коэффициент увеличивается).

Ослабляется связь между черными точками входного изображения и нейронами «чужих» классов, а также ослабляется связь между белыми точками входного изображения и нейроном «своего» класса.

Перечислим задачи, решаемые приложением, предназначенным для моделирования сети, умеющей распознавать цифры:

1. Инициализация и обучение сети:
    1.1. Загрузка и масштабирование изображений.
    1.2. Выбор каталога с изображениями.

1.3. Реализация алгоритма обучения Хебба.
1.4. Вывод ошибки сети.

2. Распознавание цифр сетью:
2.1. Распознавание сетью изображений из обучающей выборки.
2.2. Распознавание сетью изображений из контрольной выборки.

3. Цветовая визуализация результатов обучения.

Контрольная выборка состоит из десяти изображений – по одному на каждую цифру (рис. 2.3). Файлы с изображениями цифр называются произвольно.

Рис. 2.3. Изображения цифр из контрольной выборки

На изображениях контрольной выборки сеть не обучается. Контрольные выборки используются для оценки результатов обучения и служат для ответа на вопрос: «Насколько хорошо сеть обобщила имеющиеся экспериментальные знания и научилась распознавать неизвестные ей в процессе обучения образцы?».

Цветовая визуализация результатов обучения сети приведена для графической иллюстрации ответа на следующие вопросы: «Как именно сеть распознает цифры? Чему она научилась в процессе обучения? Какой смысл имеют вычисленные значения весовых коэффициентов?».

РЕШЕНИЕ ЗАДАЧИ РАСПОЗНАВАНИЯ ЦИФР: программирование сети от «А» до «Я»

Компьютерная реализация приложения для моделирования сети, умеющей распознавать цифры, выполнена в свободно-распространяемой среде программирования **Lazarus**,

которая по своей сути является аналогом **Delphi**. Встроенный в эту среду язык программирования **Free Pascal** практически не отличим от диалекта **Object Pascal**, используемого в **Delphi**.

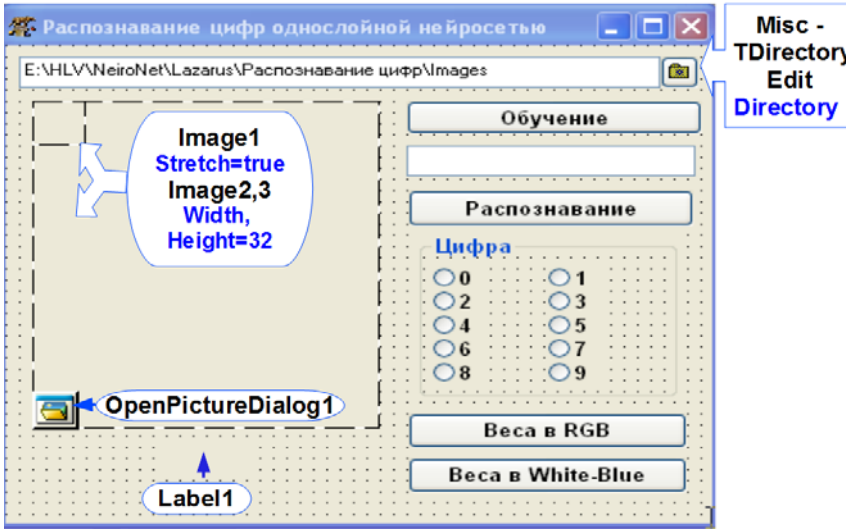


Рис. 2.4. Интерфейс приложения для распознавания цифр

На странице **Misc** можно найти и добавить на форму компонент класса **TDirectoryEdit**, предназначенный для указания пути к каталогу с обучающей выборкой. Значением свойства **Directory** этого компонента является путь, заданный по умолчанию.

Компоненты класса **TImage** расположены на странице **Additional**. Компоненты **Image2** и **Image3** имеют размеры **32x32** и являются невидимыми на этапе исполнения программы. Визуальный компонент **Image1** имеет произвольный размер и служит для отображения цифр. Для разрешения масштабирования загруженных изображений свойство **Stretch** компонента **Image1** должно иметь значение **True**.

Клик по кнопке **Button1** (Обучение) запускает процесс обучения нейронной сети. Ошибка сети выводится в поле **Edit1**, расположенном под кнопкой **Button1.**

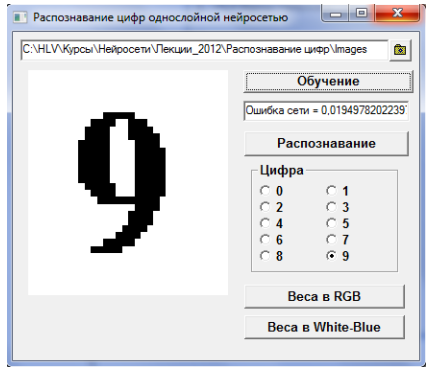


Рис. 2.5. Окно приложения после завершения обучения сети

Клик по кнопке **Button2** (Распознавание) позволяет выбрать на компьютере файл с изображением цифры через компонент **OpenPictureDialog1**. При этом обученная нейросеть распознает цифру и активизирует соответствующую опцию в группе зависимых переключателей **RadioGroup1**.

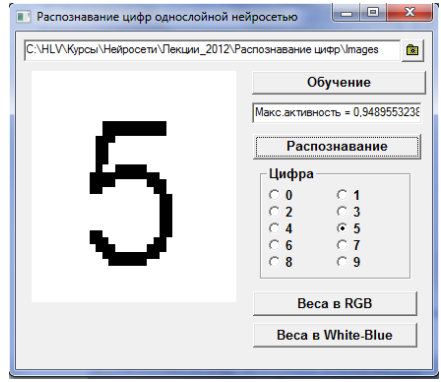


Рис. 2.6. Окно приложения в режиме распознавания цифры 5

На рис. 2.6 в поле **Edit1** отображено максимальное значение, вычисленное на выходных нейронах сети при распо-

знавании цифры «5» в контрольной выборке. Цифра «5» распознана сетью с вероятностью 95%.

В ситуации распознавания похожих объектов, например, цифр «3» и «8» степень уверенности сети в правильности распознавания существенно ниже. Так цифра «3» из контрольной выборки распознается с уверенностью в 46% (рис. 2.7). Цифра «8» с уверенностью всего 6% (рис.2.8).

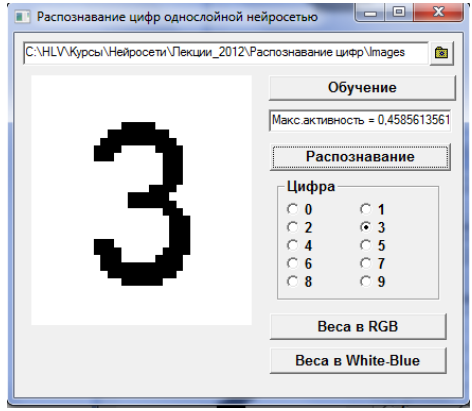


Рис. 2.7. Окно приложения в режиме распознавания цифры 3

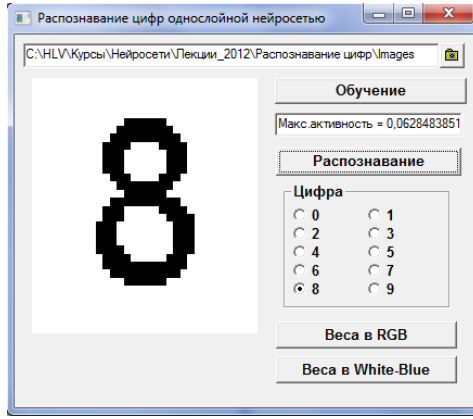


Рис. 2.8. Окно приложения в режиме распознавания цифры 8

С ситуацией неуверенного распознавания цифр может столкнуться и человек, например, когда он пытается рассмот-

реть номер маршрутного транспортного средства в сумерках или в спешке.

В программе используются глобальные данные, описание и назначение которых приведено ниже.

```
const
        eta = 0.001; // скорость обучения
        n = 10; // количество обучающих эпох
        input = 32*32;  // размерность входного вектора

private
   { private declarations }
   x: array [1..input] of integer; // входной вектор
   w: array [0..9,1..input] of real; // матрица весов
   y: array [0..9] of real; // выходной вектор
   t: array [0..9] of byte; // целевой вектор
   e: array [0..9] of real; // вектор ошибки
   err: real; // ошибка сети
   function sigmoid (arg: real) : real;
              // сигмоидальная функция активации

function TForm1.sigmoid (arg:real) : real;
begin
  sigmoid:=1/(1+exp((-1)*(arg-0.5)))
end;
```

Инициализация сети происходит при создании формы. Обработчик события OnCreate формы представлен ниже.

```
procedure TForm1.FormCreate (Sender: TObject);
var i,j : integer;
begin
  (* инициализация весов *)
   Randomize;
```

```
        for i := 0 to 9 do
         for j := 1 to Input do
          w[i,j] := 0.001 * Random;
      end;
```

При инициализации сети весовые коэффициенты принимают малые случайные значения.

В процессе обучения сети из каталога с обучающей выборкой последовательно выбираются все изображения цифр и происходит корректировка синаптических весов. Полный прогон обучающей выборки называется *обучающей эпохой*.

```
procedure Tform1.Button1Click (Sender: TObject);
var dir, filename, name_: string;
      k, d, r, i, j :integer;
      sum : real;
     SearchRec : TSearchRec;
begin
 Label1.Visible := false;
 dir := DirectoryEdit1.Directory;
 dir := dir+'\';   k:=0;

// обучение сети
  for k := 1 to n do
     begin

// выбор изображений из каталога с обучающей выборкой
          FindFirst(dir+'*.bmp', faAnyFile, SearchRec);
          filename := dir+SearchRec.Name;
          name_ := SearchRec.Name;
        while FindNext(SearchRec) = 0  do
          begin
           Image1.Picture.LoadFromFile(filename);
           Image2.Picture.LoadFromFile(filename);
```

```
d := StrToInt(name_[1]);
RadioGroup1.ItemIndex:=d;

(* Определение сигнала на входе*)
r := 0;
for i := 1 to trunc(sqrt(input)) do
  for j := 1 to trunc(sqrt(input)) do
   begin
     r := r+1;
     if Image2.Canvas.Pixels[i,j] = clBlack
        then x[r] := 1
        else x[r] := 0;
   end;

(* Вычисление взвешенных сумм *)
   for i := 0 to 9 do
    begin
      sum := 0;
      for j := 1 to Input do sum := sum + w[i,j] * x[j];
      y[i] := sum;
    end;

(* определение активности нейронов выходного слоя *)
      for i := 0 to 9 do y[i] := sigmoid(y[i]);

(* Вычисление ошибок для каждого класса и ошибки сети *)

      for i := 0 to 9 do
       if i = d then t[i] := 1 else t[i] := 0;
      err := 0;
      for i := 0 to 9 do
       begin  e[i] := abs (t[i] — y[i]);
                 err := err + e[i]
       end;
```

```
(* Модификация весовых коэффициентов *)
    for i := 0 to 9 do
      for j := 1 to Input do
       begin
         if (x[j] = 1) and (t[i] = 1)
            then w[i,j] := w[i,j] + eta * e[i];
         if (x[j] = 1) and (t[i] = 0)
            then w[i,j] := w[i,j] — eta * e[i];
         if (x[j] = 0) and (t[i] = 1)
            then w[i,j] := w[i,j] — eta * e[i];
       end;

(* Переход к следующему изображению *)
filename:=dir + SearchRec.Name;
   name_:=SearchRec.Name;
  end;
end; // Конец обучения сети

(* Вывод усредненной ошибки сети *)
  Edit1.Text:='Ошибка сети = '+FloatToStr(err/10)
end;
```

При повторных запусках обучения сети ошибка уменьшается, т.е. происходит постепенное уточнение весовых коэффициентов и всё лучшее распознавание объектов из обучающей выборки.

В режиме функционирования обученная сеть решает задачу распознавания цифр. При этом сразу вычисляются значения выходных нейронов. Нейрон, который имеет максимальное значение, служит маркером класса, к которому следует отнести цифру на изображении.

```
procedure TForm1.Button2Click(Sender: TObject);
```

```
var d, r, i, j : integer;
     sum, max : real;
begin
 if OpenPictureDialog1.Execute
  then
   begin

// диалог открытия файла с изображением цифры
     Image1.Picture.LoadFromFile
        (OpenPictureDialog1.Filename);
     Image2.Picture.LoadFromFile
        (OpenPictureDialog1.Filename);
   end;

// распознавание цифры

    (* Определение сигнала на входе*)
    r := 0;
    for i := 1 to trunc(sqrt(input)) do
     for j := 1 to trunc(sqrt(input)) do
      begin
        r := r + 1;
        if Image2.Canvas.Pixels[i,j] = clBlack
          then x[r] := 1
          else x[r] := 0;
      end;
(* Вычисление взвешенных сумм *)
   for i := 0 to 9 do
     begin
       sum := 0;
       for j := 1 to Input do sum := sum + w[i,j] * x[j];
       y[i] := sum;
     end;
```

```
(* определение активности нейронов выходного слоя *)
   for i := 0 to 9 do y[i] := sigmoid(y[i]);

(* определение нейрона с максимальной активностью *)
   max := y[0]; d := 0;
   for i := 1 to 9 do
    if y[i]>max then begin max := y[i]; d := i end;
   Edit1.Text:='Макс.активность = '+ FloatToStr(max);
   RadioGroup1.ItemIndex:=d
end;
```

На данном этапе приведено описание основных элементов моделирования искусственной нейронной сети: инициализация, обучение и функционирование.

После того, как программа отлажена и заработала, следует вначале обучить сеть (при этом уточняется матрица весовых коэффициентов). Затем можно переходить к распознаванию изображений из обучающей и контрольной выборок. Если сеть путает цифры, следует повторно запустить процесс обучения. Заменой обучающей выборки можно научить сеть решать аналогичную задачу, например, распознавать буквы.

Проводя учебные курсы по нейросетевому моделированию, автору не раз приходилось слышать вопросы такого рода: «Почему это работает? Как принимает решение нейронная сеть?». В учебных и практических целях актуальной является задача перевода внутреннего состояния обученной искусственной нейронной сети в форму, понятную пользователям.

КАЧЕСТВЕННАЯ ОЦЕНКА РЕЗУЛЬТАТОВ ОБУЧЕНИЯ: визуализация внутреннего состояния обученной искусственной нейронной сети

В искусственных нейронных сетях обработка информации происходит путем распределения вычислений между нейронами. В результате обучения сети формируется набор весовых коэффициентов связи между нейронами. По полученному набору чисел непросто качественно оценить внутреннее состояние обученной нейронной сети, особенно в случае большой размерности входного пространства и использования скрытых слоев нейронов.

Для визуализации результатов обучения сети можно использовать отображение полученного набора весовых коэффициентов в выбранную зону цветового спектра [7].

Десятеричную раскладку цвета ($BBGGRR), соответствующую весовому коэффициенту, вычислим по формуле:

$$RGB_w = RGB_2 - \frac{(w - w_{\min})}{(w_{\max} - w_{\min})} \cdot (RGB_2 - RGB_1), \qquad (2.1)$$

где w - весовой коэффициент связи,

RGB_w - цвет весового коэффициента,

$(w_{\min}, w_{\max})$ - диапазон изменения весовых коэффициентов,

(RGB_2, RGB_1) - диапазон изменения цветов.

Для перевода цветовой раскладки в шестнадцатеричную систему весовые коэффициенты связи удобно увеличить на несколько порядков.

На рис.2.9 хорошо видно, что при обучении сети по правилу Хебба для каждого нейрона формируется «матрица чувствительности» или «портрет» класса.

В такой своеобразной картине восприятия образов наибольшее усиление получают точки, формирующие размытый контур цифры. При этом для каждой цифры формируется

остов из самых темных точек. Точки остова часто встречаются в разных вариантах изображения цифры в обучающей выборке.

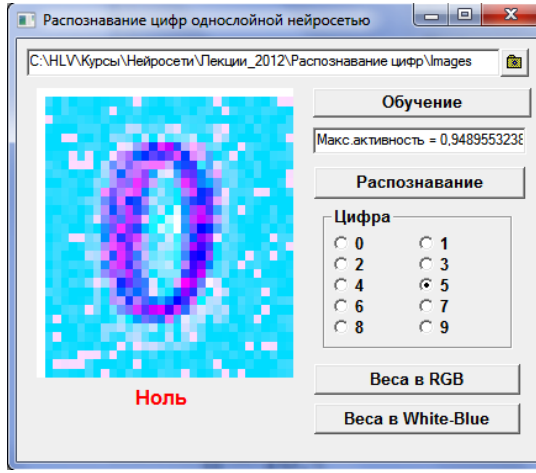


Рис. 2.9. Окно приложения в режиме визуализации весов нулевого нейрона сети

Таким образом, вычисленные весовые коэффициенты нейрона складываются в шаблон цифры с размытыми краями.

Обработчик события OnClick кнопки Button3 (Веса в RGB) представлен ниже.

```
procedure TForm1.Button3Click(Sender: TObject);
const eps = 10000;
var s : string; c : char;
    i, k, j, r, weight_int : integer;
    weight, min, max : real;
begin
    image1.Visible:=false;
    image3.Visible:=true;
   for i := 0 to 9 do
     begin
```

```
  //название класса

    case i of
    0 : label1.Caption := 'Ноль';
    1 : label1.Caption := 'Единица';
    2 : label1.Caption := 'Двойка';
    3 : label1.Caption := 'Тройка';
    4 : label1.Caption := 'Четверка';
    5 : label1.Caption := 'Пятерка';
    6 : label1.Caption := 'Шестерка';
    7 : label1.Caption:= 'Семерка';
    8:label1.Caption := 'Восьмерка';
    9 : label1.Caption := 'Девятка'
    end;

// очистка канвы

   image3.Canvas.Brush.Color := clWhite;
   image3.Canvas.FillRect(0, 0,
                          image3.Width, image3.Height);

 // поиск максимума и минимума среди весов k-го нейрона

   min := w[i,1]; max := min;
   for k := 2 to input do
    begin
      if w[i,k] > max then max := w[i,k];
      if w[i,k] < min then min := w[i,k];
    end;
   min := min * eps;  max := max * eps;

// определение цвета по весу

    for k:=1 to trunc( sqrt ( input ) ) do
```

```
    for j:=1 to trunc( sqrt ( input ) ) do
     begin
      weight := w[i,(k-1) * trunc (sqrt (input) ) + j];
      weight := weight*eps;
      weight := 16777215 — (weight — min) *
                  16777215 / (max — min);
      s := '';
      weight_int := round (weight);

while weight_int <> 0 do
       begin
         r := weight_int mod 16;
         weight_int := weight_int div 16;
         case r  of
         0 : c:='0';
         1 : c:='1';
         2 : c:='2';
         3 : c:='3';
         4 : c:='4';
         5 : c:='5';
         6 : c:='6';
         7 : c:='7';
         8 : c:='8';
         9 :  c:='9';
         10: c:='A';
         11: c:='B';
         12: c:='C';
         13: c:='D';
         14: c:='E';
         15: c:='F'
         end;
        s:=c+s;
       end;
     while length(s) < 6 do s:='0' + s;
```

```
        s:='$'+s;
        Image3.Canvas.Pixels[k,j] := StringToColor(s)
       end;
    Image3.Stretch := true;
     Image3.width := 208;
     Image3.height := 235;
     MessageDlgPos ( 'Смотри!', mtWarning,
                          [mbOK], 1, 500, 600 );
     if i <> 9 then
      begin
       Image3.Stretch := false;
       Image3.width := 32;
       Image3.height := 32;
      end
     end;
end;
```

После того, как программа отлажена и заработала, следует обучить сеть. Убедиться, что сеть успешно распознает цифры контрольной выборки. Затем провести цветовую визуализацию результатов обучения. Повторное обучение сети влияет также и на результаты визуализации.

Наиболее наглядные результаты получаются при отображении весовых коэффициентов в бело-синюю часть спектра. При таком подходе самым маленьким коэффициентам соответствует белый цвет, самым большим коэффициентам – синий.

Обработчик события **OnClick** кнопки **Button4** (Веса в White-Blue) является копией обработчика выбора кнопки **Button3** (Веса в RGB) с заменой номеров цветов в формуле (2.1).

```
procedure TForm1.Button4Click(Sender: TObject);
  ...
  weight := 16777215 — ( weight — min) *
```

(16777215 — 16711680) / (max — min);

...

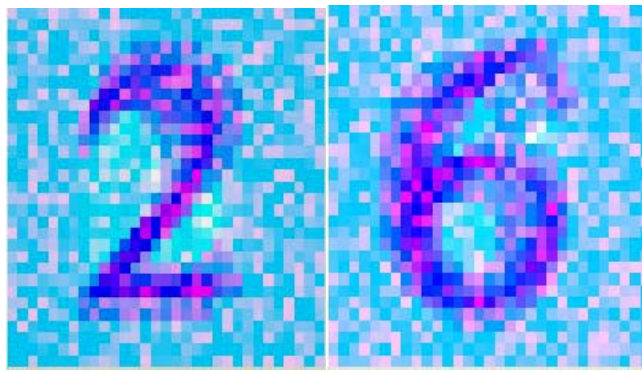


Рис. 2.10. Отображение весов второго и шестого нейронов в бело-синюю часть спектра

На рис. 2.10 приведены цветовые матрицы весовых коэффициентов связи между координатами входного вектора и нейронами, отвечающими за распознавание «своей» цифры.

Вокруг контура цифры формируется «вытравленная» в результате обучения зона пониженной чувствительности с самыми маленькими весовыми коэффициентами, которым соответствуют светлые точки.

При увеличении контрастности изображения (рис. 2.11) заметно, что зона пониженной чувствительности представляет собой результат объединенного изображения других цифр.

Весовые коэффициенты связи между точками холста вокруг изображения цифр «усредняются» для всех выходных нейронов. Процесс усреднения особенно заметен при продолжительном обучении (рис. 2.12).

Рис. 2.11. Визулизация весов первого нейрона со светлой зоной вокруг контура цифры

Используя цветовую визуализацию удобно следить за динамикой изменения весовых коэффициентов при обучении.

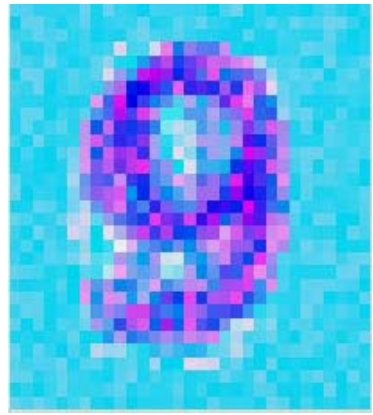


Рис. 2.12. Визулизация весов девятого нейрона с однородным фоном при продолжительном обучении

На рис. 2.10 показаны цветовые матрицы весовых коэффициентов при кратковременном обучении, на рис. 2.12 при продолжительном обучении. Заметно, что при продолжи-

тельном обучении остов цифры становится крупнее (веса от черных точек «своей» цифры увеличиваются), зона пониженной чувствительности – светлее (веса от черных точек «чужих» цифр уменьшаются), общий фон холста – ровнее (веса от точек холста усредняются).

Обобщая представленную идею, можно предположить, что для лучшего восприятия качественного состояния обученной сети следует использовать отображение внутренних характеристик сети в подпространство входного пространства. Например, в картинку при распознавании образов, в звуковой сигнал при распознавании речи, …

Представленная в данном пункте методика позволяет выделить типичного представителя («портрет») для каждого класса.

ЗАДАЧА МЕДИЦИНСКОЙ ДИАГНОСТИКИ: прогнозирование исходов мерцательной аритмии

Рассмотренные в данной главе методики моделирования однослойных нейронных сетей можно использовать при решении практических задач. Для примера рассмотрим одну задачу медицинской диагностики, связанную с распознаванием состояния синусового узла, выполняющего роль водителя ритма сердца, по данным, полученным с помощью датчика межпульсовых интервалов.

Восстановление синусового ритма при мерцательной аритмии, включая вопросы прогноза его возможной нормализации, относятся к серьезной врачебной задаче. Выявление меры влияния синусового узла при мерцательной аритмии позволяет прогнозировать результаты дефибриляции и трепетания предсердий антиаритмическими препаратами [42].

Решение проблемы полезности восстановления синусового ритма, основанное на общеклинических критериях исследования относится к трудоемким и субъективным мето-

дам, обеспечивающим маловоспроизводимые результаты в силу отсутствия разработанных стандартов в критериях оценок [32, 45].

Для целей классификации нарушений синусового ритма известны методы условно-вероятностного анализа, корреляционной ритмографии, скатерографии [41, 59, 93]. Известны способы автоматизации данных методов. В них предварительно обработанную информацию в виде авторегрессионных облаков предъявляют врачу-эксперту, который и дает заключение о конечном прогнозе [42, 46], либо прогноз реализуется в автоматическом режиме [32, 39, 43].

В 1995 г. Ф.А. Пятаковичем на основании нечетких множеств впервые был описан алгоритм дифференциации авторегрессионных облаков у больных с синдромом фибрилляции предсердий [53].

В 2000 г. автором, с помощью разработанного генератора для формирования скатерограмм, были получены адекватные модели и алгоритмы автоматического распознавания авторегрессионных облаков с использованием элементов теории нечетких множеств [73].

В [68, 71] описан геометрический алгоритм распознавания класса авторегрессионного облака, базирующийся на поиске сгущения точек в заданной экспертом зоне. Алгоритмы распознавания авторегрессионных облаков, описанные в [69, 79], используют разделение гиперплоскостью мономодальных и немономодальных классов на основе выделенных информационных признаков оценки микроструктуры и макроструктуры ритма.

Однако эти исследования носили больше экспериментальную и методологическую направленность, нежели клиническую [47].

В задачах диагностики в режиме on-line целесообразно использовать быстрые интеллектуальные системы. К ним, прежде всего, относятся искусственные нейронные сети. В

литературе представлены лишь единичные исследования, касающиеся использования «нейрокомпьютинга» для решения задач диагностики у больных с синдромом фибрилляции предсердий [70, 77].

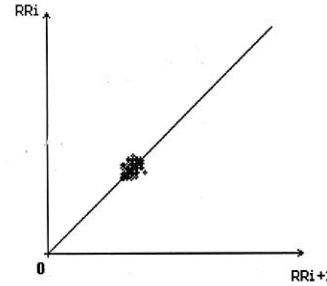


Рис. 2.13. Авторегрессионное облако пациента с синдромом повышенной ритмичности

Временной интервал между наиболее высокоамплитудными зубцами на электрокардиограмме называется RR-интервалом.

На плоскости отмечаются точки (RR_i, RR_{i+1}), где $i = 1,2,...,n-1$, n - количество RR интервалов на электрокардиограмме.

Полученную совокупность точек в зарубежной литературе именуют скаттерграммой или авторегрессионным облаком, а в отечественной литературе используют название – корреляционная ритмография, или сокращенно КРГ.

При наличии правильного ритма сокращения желудочков скопление точек образуется на биссектрисе координатного угла. При стойкой ритмичности – стабильном ритме – основная совокупность превращается в точку на биссектрисе. Авторегрессионное облако пациента с синдромом повышенной ритмичности показано на рис. 2.13.

Умеренная синусовая аритмия приводит к образованию авторегрессионного облака в виде эллипса (рис. 2.14) или круга.

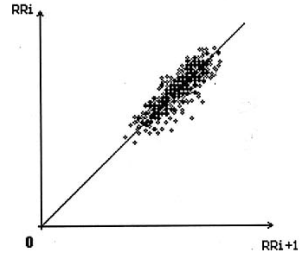


Рис. 2.14. Авторегрессионное облако пациента с умеренной синусовой аритмией

При благоприятном прогнозе на восстановление синусового ритма на авторегрессионном облаке точки группируются на сравнительно ограниченной округлой площади, выраженное сгущение точек наблюдается на биссектрисе.

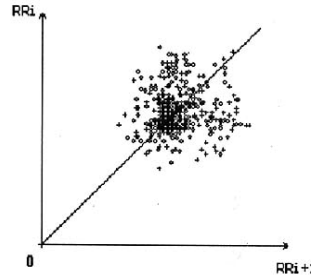


Рис. 2.15. Авторегрессионное облако мономодального симметричного класса

Сгущение может быть в центре (рис. 2.15), в левом нижнем (рис. 2.16) и правом верхнем углу авторегрессионного облака (рис. 2.17).

При сомнительном прогнозе на восстановление синусового ритма на авторегрессионном облаке образуется несколько симметрично расположенных относительно биссектрисы совокупностей точек (рис. 2.18), либо имеет место широкое рассеивание точек на плоскости (рис. 2.19).

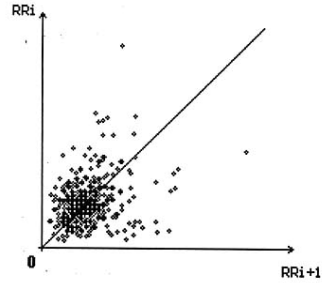


Рис. 2.16. Авторегрессионное облако мономодального асимметричного класса

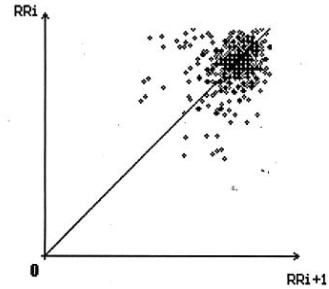


Рис. 2.17. Авторегрессионное облако мономодального инвертированного класса

Таким образом, выраженная аритмичность сокращения желудочков, сопровождающая мерцание предсердий, проявляется на корреляционной ритмограмме широким рассеиванием точек на плоскости. Наблюдения показали, что полная беспорядочность чередования межпульсовых интервалов при мерцании предсердий является кажущейся.

Метод корреляционной ритмографии позволяет разделить больных на несколько функциональных классов в зависимости от типа авторегрессионного облака: 1. Мономодальный симметричный. 2. Мономодальный асимметричный. 3. Мономодальный инвертированный. 4. Полимодальный. 5. Амодальный (рис. 2.15 – 2.19).

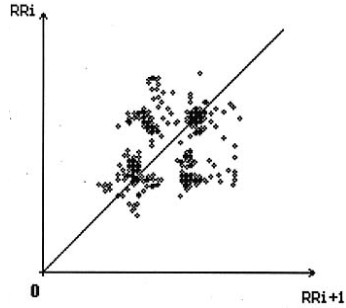


Рис. 2.18. Авторегрессионное облако полимодального класса

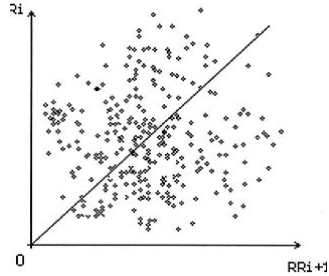


Рис. 2.19. Авторегрессионное облако амодального класса

Если точки группируются на сравнительно ограниченной округлой площади и выраженное сгущение их расположено в центре, на биссектрисе, то это означает, что имеется значительное число сердечных циклов с одинаковой длительностью, т.е. наблюдается правильный ритм сокращений желудочков (рис. 2.15).

Наличие выраженного сгущения точек на биссектрисе координатного угла является прогностически благоприятным и свидетельствует о вполне удовлетворительном функциональном состоянии сердца.

Имеется предположение о возможности влияния синусового узла на ритм желудочков, регулируемый в обход мерцающих предсердий через межузловые тракты.

Иначе говоря, мономодальный тип свидетельствует о высоких функциональных возможностях синусового узла.

Прямым подтверждением справедливости этого утверждения является высокая эффективность восстановления и удержания синусового ритма у больных с таким типом авторегрессионного облака.

При втором типе облака на биссектрисе также имеется сгущение точек, однако, площадь их рассеивания ограничена прямыми линиями, идущими параллельно осям координат (рис. 2.16).

Иначе говоря, на облаке второго типа отсутствуют точки, соответствующие в облаках первого типа самым коротким сердечным циклам. Ограничение территории распределения точек на облаке второго типа отражает фильтрацию наиболее частых импульсов в атриовентрикулярном соединении.

При третьем типе облака скопление точек на биссектрисе находится в отдаленном от начала координат месте и также ограничено линиями, идущими параллельно осям координат (рис. 2.17).

При четвертом типе облака точки группируются на биссектрисе параллельно осям координат в виде отдельных групп (пакетов) скоплений (рис. 2.18). В зависимости от расстояния между центрами скоплений выделяют два подтипа: 1 - трепетание предсердий с меняющимся атриовентрикулярным проведением; 2 - крупноволновая форма мерцания предсердий.

При пятом типе облака область распределения точек также ограничена линиями, идущими параллельно осям координат, однако сгущение точек на биссектрисе отсутствует (рис. 2.19). В данном случае функциональное состояние синусового узла неудовлетворительно и перспективы на восстановление правильного синусового ритма сомнительны.

Считают, что тип авторегрессионного облака отражает степень утраты регуляторных воздействий на ритм сердца при мерцательной аритмии: достаточное сохранение при первом типе, меньшее – при втором, наименьшее – при третьем

типе. У больных с третьим типом авторегрессионного облака, которому сопутствует большой разброс точек, не удается сохранить восстановленный синусовый ритм на срок до 6 месяцев. Переходной формой между трепетанием и мерцанием предсердий является четвертый тип облака.

Как видно из представленных данных границы между дифференцируемыми классами нечетки или размыты, особенно это касается асимметричных и низкомодальных типов авторегрессионных облаков.

РЕШЕНИЕ ЗАДАЧИ МЕДИЦИНСКОЙ ДИАГНОСТИКИ: с помощью искусственной нейронной сети

Метод решения задачи прогнозирования исходов мерцательной аритмии может базироваться на моделировании однослойной нейронной сети прямого распространения с пятью нелинейными нейронами.

Входной слой является моделью сенсорных клеток, принимающих двоичные сигналы от внешнего мира. На вход нейронной сети поступают бинарные прямоугольные изображения авторегрессионных облаков.

Во входной слой включается m нейронов s_j, где $j=1,...,m$ (m - количество точек на изображении).

Если на изображении точка черная, то нейрон считается активным. В этом случае значение элемента одномерного массива, реализующего входной слой, равно 1. Если на изображении точка белая, то нейрон неактивен и значение соответствующего элемента массива равно 0.

Входной слой является моделью сенсорных клеток, принимающих двоичные сигналы от внешнего мира.

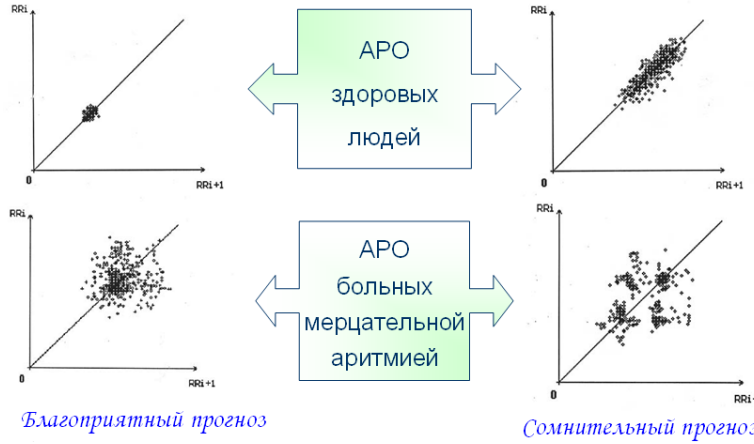


Рис. 2.20. Отличие авторегрессионных облаков здоровых людей и больных мерцательной аритмией

Далее сигналы поступают в слой ассоциативных клеток (выходных нейронов) a_i, где $i=1,..,n$, которые имеют изменяемые веса связей с сенсорными клетками. Ассоциативных клеток в нашем примере пять, каждая из них отвечает за распознавание своего класса авторегрессионных облаков (рис. 2.15 – 2.19).

Активность i-й ассоциативной клетки определяется формулой:

$$a_i = \sum_{j=1} w_{ij} s_j \,, \qquad (2.2)$$

где w_{ij} – коэффициент связи между i-й ассоциативной клеткой и j-й сенсорной клеткой.

Информация от ассоциативных клеток подлежит нелинейной обработке, в результате которой формируется реакция персептрона на входной образ. Для вычисления активности нейрона выходного слоя r_i, где $i=1,..,n$, используется сигмоидальная функция активации:

$$r_i = \frac{1}{1 + e^{-(a_i - \Theta)}}, \qquad (2.3)$$

где константа Θ выполняет роль порога, влияющего на возбуждение нейрона выходного слоя. В нашем исследовании $\Theta = 0{,}5$.

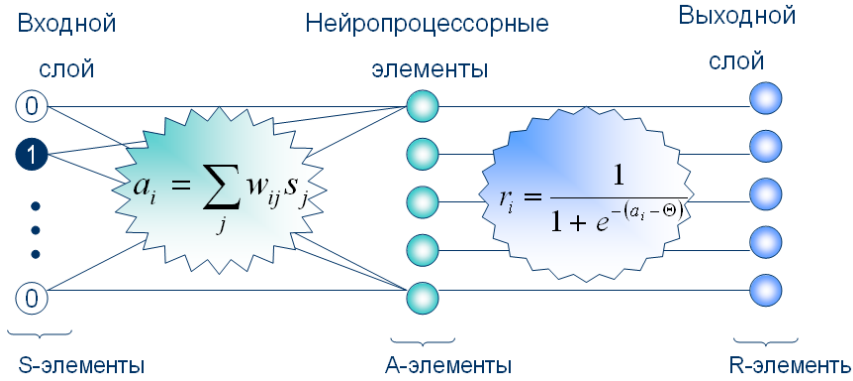


Рис. 2.21. Архитектурная схема сети

Узнавание класса проводится по самому возбужденному нейрону выходного слоя.

Перед обучением персептрона эксперт проводит классификацию авторегрессионных облаков из обучающей выборки, определяя класс, к которому следует отнести данный элемент.

В начале процесса обучения персептрона случайным образом заполняется матрица весовых коэффициентов. В нашем примере начальные значения весовых коэффициентов не превосходили 0,001.

В ходе сеанса обучения случайным образом определяется номер облака из обучающей выборки. По выбранному номеру входного примера восстанавливается номер класса i, к которому было отнесено данное облако экспертом. У целевого вектора $t_i = 1$, а все остальные координаты - нулевые.

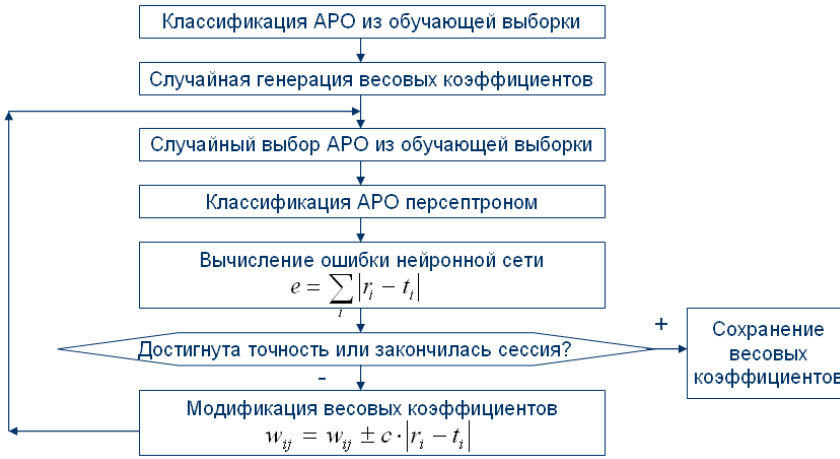


Рис. 2.22. Алгоритм обучения сети

По выбранному элементу обучающей выборки вычисляется активность нейронов входного слоя, ассоциативных клеток и нейронов выходного слоя.

Для каждого класса вычисляется вектор ошибки:

$$e_i = |a_i - t_i|. \qquad (2.4)$$

Суммарная ошибка сети вычисляется по формуле:

$$net = \sum_{i=1}^{n} e_i \ . \qquad (2.5)$$

Весовые коэффициенты модифицируются с учетом ошибок по классам согласно правилу Хебба.

Усиливается связь между черными точками и ассоциативной клеткой, соответствующей номеру “своего” класса: если $s_j = 1$ и $t_i = 1$, то $w_{ij} = w_{ij} + c \cdot e_i$, где c - константа, влияющая на скорость и качество обучения. В нашем примере $c = 0{,}00001$.

Ослабляется связь между черными точками и ассоциативными клетками, отвечающими за “чужой” класс: если $s_j = 1$ и $t_k = 0$, то $w_{ik} = w_{ik} - c \cdot e_k$, где $k \neq i$.

Ослабляется связь между белыми точками и ассоциативной клеткой, отвечающей за “свой” класс: если $s_j = 0$ и $t_i = 1$, то $w_{ij} = w_{ij} - c \cdot e_i$.

Обучающие сеансы повторяют до тех пор, пока суммарная ошибка сети не станет меньше некоторой константы или не закончится обучающая сессия.

Лучший результат обучения (матрица весовых коэффициентов, с которыми сеть сделала минимальную ошибку в распознавании элементов обучающей выборки) сохраняется в файл. При загрузке весовых коэффициентов из файла можно проводить последовательное обучение сети в режиме off-line.

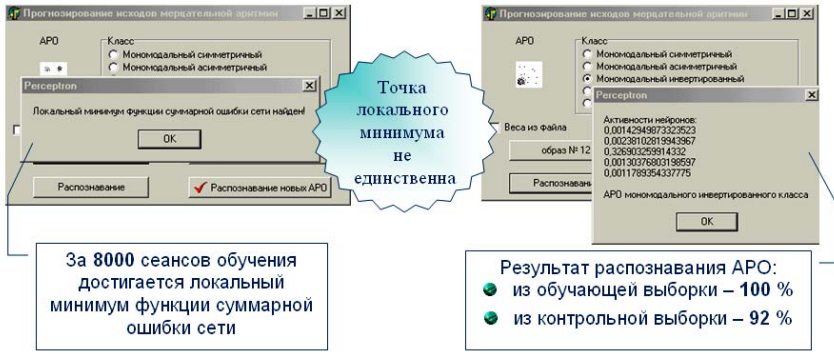


Рис. 2.23. Вид приложения после обучения сети и работы сети в режиме распознавания образов

В ходе проведения эксперимента было установлено, что за 8000 сеансов обучения достигается локальный минимум функции суммарной ошибки. Точка локального минимума не единственна. Суммарная ошибка сети составляет 0,65-0,68.

Авторегрессионные облака из обучающей выборки распознаются все правильно. Авторегрессионные облака мономодальных классов из контрольной выборки все были распознаны правильно. Ошибки сеть допускала при распознавании авторегрессионных облаков немономодальных классов из контрольной выборки.

Проведенные исследования показали, что искусственная нейронная сеть дает гибкий инструментарий в распознавании авторегрессионных облаков, который точно можно настроить под мнение авторитетного эксперта [8].

ИНТЕГРАЦИЯ НЕЙРОСЕТЕВОГО МОДУЛЯ: в информационную систему прогнозирования исходов синдрома фибрилляции предсердий

На кафедре пропедевтики внутренних болезней и клинических информационных технологий в Белгородском государственном национальном исследовательском университете был разработан макетный образец автоматизированной телемедицинской системы прогнозирования исходов синдрома фибрилляции предсердий с передачей данных по каналу модемной связи [49].

Разработанная система может функционировать в режиме on-line, обеспечивая консультативной вычислительной диагностикой неясных больных с синдромом фибрилляции предсердий, госпитализированных в центральные районные больницы.

Целью разработки такой системы является классификация осложняющих патологических состояний, направленная на оптимизацию принятия диагностических и прогностических решений, реализуемая посредством алгоритмов автоматической классификации авторегрессионных облаков.

В систему включен нейросетевой модуль, а также другие модули классификации авторегрессионных облаков, основанные на использовании методов нечеткой логики принятия решений, хаотической динамики и общестатистических подходов с вычислением степени утраты функции концентрации синусового узла.

Для достижения поставленной цели было обследовано 507 больных. Из них – 88 человек с диффузным токсическим

зобом и вегетососудистой дистонией, 93 человека с ишемической болезнью сердца и ревматизмом, 96 пациентов с недостаточностью кровообращения. Из всех у 230 были проведены статистические исследования по верификации правил прогнозирования. У всех больных анализировались от 512 до 1084 межпульсовых интервалов, которые при помощи датчика пульса вводились в ЭВМ в режиме on-line.

Телемедицинская система прогнозирования исходов синдрома фибрилляции предсердий включала модуль ввода электрофизиологической информации, АРМ-врача, АРМ-эксперта, сервер и два модема (рис. 2.24).

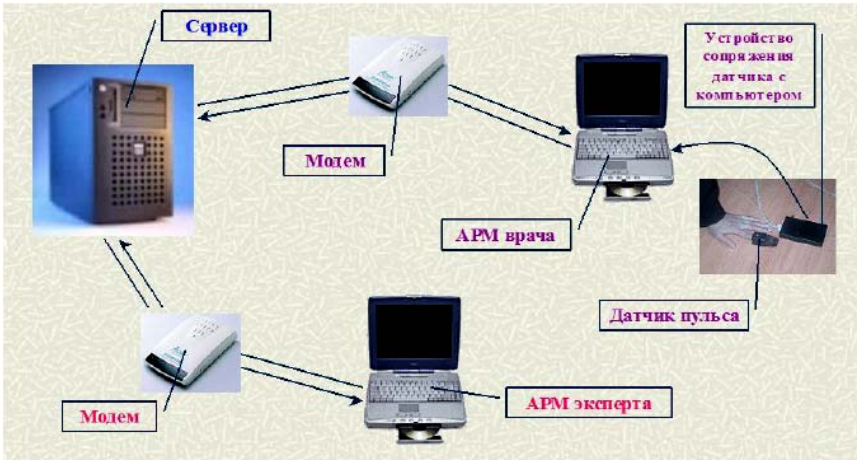


Рис. 2.24. Структура телемедицинской системы «Медэксперт»

Сервер баз данных размещен на компьютере, имеющем доступ в Интернет. Он обеспечивает прием, хранение, обработку полученных данных: списки обследуемых пациентов, результаты их обследования, экспертные заключения, а также доступ к принятым и переработанным данным от АРМ врача и АРМ эксперта по локальным и глобальным сетям.

Автоматизированное рабочее место врача представляет собой программно-аппаратный комплекс, состоящий из дат-

чика приема межпульсовых интервалов, устройства сопряжения датчика и компьютера и программного обеспечения. С его помощью осуществляется сбор первичных данных о пациентах как паспортных и антропометрических, так и полученных с датчика пульса, а также отображение и печать экспертных заключений.

Ввод и редактирование данных о пациенте обеспечивается в окне ввода данных.

Выбор пункта меню «Новое обследование» запускает последовательность окон, в которых вводятся необходимые данные о текущем обследовании. Затем в окне «Обследование» предоставляется возможность проверки датчика пульса, приема данных с датчика пульса, выбора режима записи – укороченный цикл – 512 интервалов или полный – 1084 интервала в двух положениях тела пациента: лежа и стоя, а также во время изменения положения тела (переходный процесс).

По окончании обследования одного или более пациентов полученные данные отправляются для хранения и обработки на сервер.

На сервере реализован автономный исследовательский модуль изучения микроструктуры вариабельности ритма сердца у больных с синдромом фибрилляции предсердий, направленный на решение задач классификации и отличающийся использованием искусственных нейронных сетей, предназначенный для решения консультативных задач выбора оптимальных методов лечения [49].

ГЛАВА III. МНОГОСЛОЙНЫЕ СЕТИ ПРЯМОГО РАСПРОСТРАНЕНИЯ

ОБУЧЕНИЕ МНОГОСЛОЙНЫХ СЕТЕЙ: алгоритм обратного распространения ошибки

Одним из распространенных методов обучения многослойных искусственных нейронных сетей традиционной архитектуры является метод обратного распространения ошибки.

Рассмотрим ключевые формулы метода на примере двухслойной искусственной нейронной сети прямого распространения.

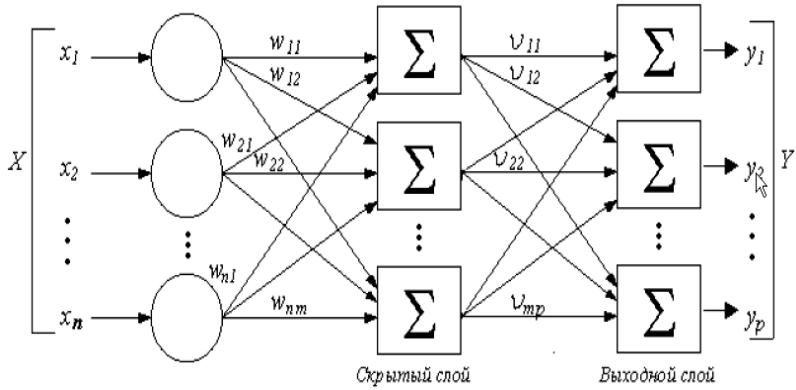


Рис. 3.1. Архитектура двухслойной сети

Введем ряд обозначений:

- x_j, где $j=\overline{1,n}$, — j-я координата входного вектора;
- y_k^c, где $k=\overline{1,m}$, — k-й нейрон скрытого слоя;
- y_i, где $i=\overline{1,p}$, — i-нейрон выходного слоя;
- w_{kj} — весовой коэффициент связи между j-м входом и k-м нейроном скрытого слоя;
- v_{ik} — весовой коэффициент связи между k-м нейроном скрытого слоя и i-м нейроном выходного слоя;

- t_i, $i=\overline{1,p}$, — i-я координата целевого вектора.

Активность нейронов скрытого слоя y_k^c, $k=\overline{1,m}$, вычисляется по формулам:

$$y_k^c = f\left(\sum\nolimits_{j=1}^{n} w_{kj} x_j\right), \tag{3.1}$$

где w_{kj} - весовой коэффициент связи между j-м входом и k-м нейроном скрытого слоя.

В качестве активационной функции $f(\cdot)$ обычно выбирается гиперболический тангенс или сигмоидальная (логистическая) функция.

Активность нейронов выходного слоя y_i, $i=\overline{1,p}$, вычисляется по формулам:

$$y_i = f\left(\sum\nolimits_{k=1}^{p} v_{ik} y_k^c\right), \tag{3.2}$$

где v_{ik} - весовой коэффициент связи между k-м нейроном скрытого слоя и i-м нейроном выходного слоя.

При обучении искусственной нейронной сети решается задача минимизации целевой функции

$$E(v,w) = \frac{1}{2}\sum\nolimits_{i=1}^{n}(y_i - t_i)^2 . \tag{3.3}$$

При инициализации сети все весовые коэффициенты принимают малые случайные значения.

В процессе обучения пошаговая корректировка весов выполняется в направлении антиградиента целевой функции по формулам:

$$v_{ik}^{N+1} = v_{ik}^{N} - \eta \partial E / \partial v_{ik}^{N}, \tag{3.4}$$

$$w_{kj}^{N+1} = w_{kj}^{N} - \eta \partial E / \partial w_{kj}^{N}, \tag{3.5}$$

где N - дискретный момент времени, η - скорость обучения.

Найдем по выборке $\{\bar{x}_l, \bar{y}_l\}_{l=1}^{K}$, где K - объем выборки, статистические оценки для величин $\frac{\partial E}{\partial v_{ik}}$ и $\frac{\partial E}{\partial w_{kj}}$.

Для определенности выберем в качестве активационной функции гиперболический тангенс:

$$f(s) = \left(e^{s} - e^{-s}\right)/\left(e^{s} + e^{-s}\right). \tag{3.6}$$

Тогда статистические оценки величин $\frac{\partial E}{\partial v_{ik}}$ и $\frac{\partial E}{\partial w_{kj}}$ имеют вид

$$\frac{\partial E}{\partial v_{ik}} = \left(y_i - t_i\right)\left(1 - y_i^{2}\right)y_k^{c}, \tag{3.7}$$

$$\frac{\partial E}{\partial w_{kj}} = \left(\sum_{i=1}^{p}\left(y_i - t_i\right)\left(1 - y_i^{2}\right)v_{ik}\right)\left(1 - \left(y_k^{c}\right)^{2}\right)x_j. \tag{3.8}$$

Введем обозначение

$$\delta_i = \left(y_i - t_i\right)\left(1 - y_i^{2}\right) \tag{3.9}$$

и запишем формулы (3.7) и (3.8) в виде:

$$\frac{\partial E}{\partial v_{ik}} = \delta_i y_k^{c}, \tag{3.10}$$

$$\frac{\partial E}{\partial w_{kj}} = \left(\sum_{i=1}^{p}\delta_i v_{ik}\right)\left(1 - \left(y_k^{c}\right)^{2}\right)x_j. \tag{3.11}$$

С учетом (3.10) и (3.11) ключевые формулы модификации весовых коэффициентов связи (3.4) и (3.5) в случае выбора гиперболического тангенса в качестве функции активации будут иметь следующий вид:

$$v_{ik}^{N+1} = v_{ik}^{N} - \eta\left(\delta_i\right)_l^{N}\left(y_k^{c}\right)_l^{N}, \tag{3.12}$$

$$w_{kj}^{N+1} = w_{kj}^{N} - \eta\left(\sum_{i=1}^{p}\left(\delta_i\right)_l^{N} v_{ik}^{N}\right)\left(1 - \left(\left(y_k^{c}\right)_l^{N}\right)^{2}\right)\left(x_j\right)_l^{N}. \tag{3.13}$$

Выберем теперь качестве активационной функции логистическую функцию:

$$f(s)=\frac{1}{1+e^{-s}} \tag{3.14}$$

Оценим величины $\frac{\partial E}{\partial v_{ik}}$ и $\frac{\partial E}{\partial w_{kj}}$:

$$\frac{\partial E}{\partial v_{ik}}=(y_i-t_i)y_i(1-y_i)y_k^c, \tag{3.15}$$

$$\frac{\partial E}{\partial w_{kj}}=\left(\sum_{i=1}^{p}(y_i-t_i)y_i(1-y_i)v_{ik}\right)y_k^c(1-y_k^c)x_j. \tag{3.16}$$

Введем обозначение

$$\delta_i=(y_i-t_i)y_i(1-y_i) \tag{3.17}$$

и запишем формулы (3.15) и (3.16) в виде:

$$\frac{\partial E}{\partial v_{ik}}=\delta_i y_k^c, \tag{3.18}$$

$$\frac{\partial E}{\partial w_{kj}}=\left(\sum_{i=1}^{p}\delta_i v_{ik}\right)y_k^c(1-y_k^c)x_j. \tag{3.19}$$

С учетом (3.18) и (3.19) ключевые формулы модификации весовых коэффициентов связи (3.4) и (3.5) в случае логистической активационной функции будут иметь следующий вид:

$$v_{ik}^{N+1}=v_{ik}^{N}-\eta(\delta_i)_l^N\left(y_k^c\right)_l^N, \tag{3.20}$$

$$w_{kj}^{N+1}=w_{kj}^{N}-\eta\left(\sum_{i=1}^{p}(\delta_i)_l^N v_{ik}^N\right)\left(y_k^c\right)_l^N\left(1-\left(y_k^c\right)_l^N\right)\left(x_j\right)_l^N. \tag{3.21}$$

Приведем полный алгоритм обучения двухслойной искусственной нейронной сети прямого распространения.

Алгоритм обратного распространения ошибки для двухслойной искусственной нейронной сети

Шаг 1. Инициализация сети.

Определяются размеры входного вектора, скрытого и выходного слоя нейронов, а также матриц весовых коэффициентов связи. Весовым коэффициентам присваиваются малые случайные значения из интервала с центром в начале координат.

Шаг 2. Выбор примера из обучающей выборки.

Пример выбирается случайно, либо имеет место простой перебор примеров из обучающей выборки. Для выбранного входного примера вычисляется активность нейронов скрытого слоя по формуле (3.1) и вычисляется выходной сигнал сети по формуле (3.2).

Шаг 3. Вычисление ошибки сети.

Текущая ошибка сети для выбранного входного примера вычисляется по формуле (3.3).

Шаг 4. Модификация весовых коэффициентов.

Происходит уточнения весовых коэффициентов по формулам (3.12) и (3.13) в случае выбора гиперболического тангенса в качестве функции активации, либо по формулам (3.20) и (3.21) в случае логистической активационной функции.

Шаг 5. Проверка выполнения критерия останова алгоритма.

Если достигнут приемлемый уровень ошибок или выполнено определенное число сеансов обучения, то *конец алгоритма*, иначе переход к шагу 2.

ПРОГНОЗИРОВАНИЕ КУРСА ДОЛЛАРА: как запрограммировать двухслойную сеть?

Рассмотрим задачу обучения искусственной нейронной сети прогнозированию курса доллара по отношению к рублю.

Сеть будет иметь 30 входов, на которые подается курс доллара за предшествующие дню прогнозирования 30 дней. Выходной слой содержит один нейрон, выдающий величину курса на 31-й день.

Скрытый слой будет состоять из 10 нейронов. В нейронах скрытого и выходного слоев используются сигмоидальные активационные функции.

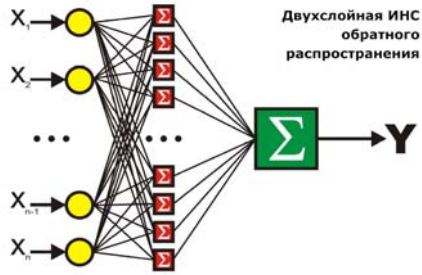


Рис. 3.2. Архитектура сети для прогнозирования курса доллара

Обучающая выборка сформирована по реальным данным, взятым из Интернет с февраля 2007 г. по март 2008 г.

Информация представляется сети в виде текстовых файлов с именами 01.txt, …, 50.txt, в каждом из которых записано по 31 числу в формате **.**. Первые 30 чисел содержали информацию о курсе доллара за тридцать последовательных дней, последним числом был записан фактический курс на 31-й день. Последнее число в каждом файле является целевым откликами нейросети для данного входного примера.

```
26.17 26.16 26.15 26.13 26.17 26.21
26.24 26.23 26.20 26.22 26.18 26.14
26.13 26.04 26.04 26.04 26.03 25.97
26.01 26.07 26.01 25.99 26.02 26.01
25.99 25.98 26.00 25.98 25.92 25.98
25.92
```

Рис. 3.3. Содержимое файла 01.txt из обучающей выборки

Компьютерная реализация модели искусственной нейронной сети выполнена в свободно распространяемой среде программирования **Lazarus**. Встроенный в эту среду язык программирования **Free Pascal** синтаксически почти не отличим от языка **Object Pascal**, используемого в **Delphi**.

Интерфейс приложения показан на рис. 3.3.

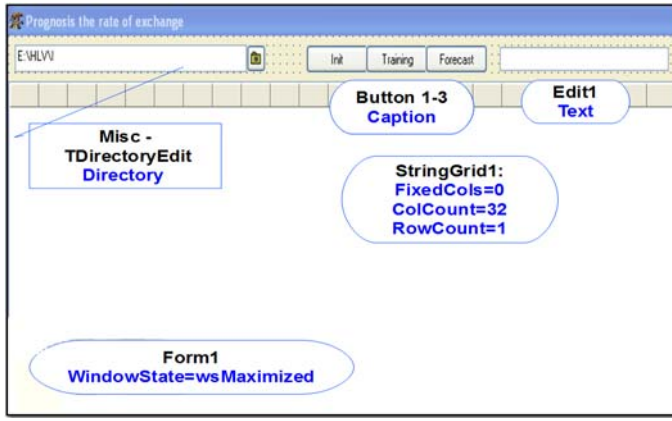


Рис.3.3. Интерфейс приложения для прогнозирования курса $

На странице **Misc** можно найти и добавить на форму компонент класса **TDirectoryEdit**, предназначенный для указания пути к каталогу с обучающей выборкой. Значением свойства **Directory** этого компонента является путь, заданный по умолчанию.

Кнопки **Button1** (Init), **Button2** (Traning), **Button3** (Forecast) предназначены для инициализации, обучения и использования сети в режиме прогнозирования. Названия кнопок указываются через свойство **Caption**.

В текстовое поле **Edit1** выводится ошибка сети через ключевое свойство **Text** компонента **Edit1**.

Основную часть формы занимает таблица **StringGrid1**, в которую выводится информация из обучающей выборки, прогноз сети и реальное значение курса $ на 31-й день.

Окно программы после завершения обучения сети показано на рис. 3.4.

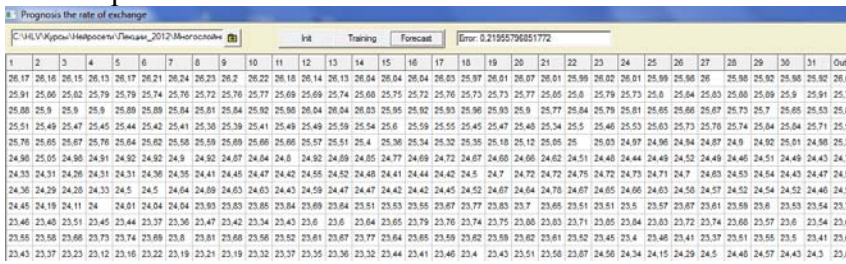


Рис. 3.4. Окно приложения после завершения обучения сети и формирования прогноза для курса $

Количество столбцов в таблице **StringGrid1** равно 32 и определяется значением свойства **ColCount**. Каждая строка таблицы содержит информацию из отдельного файла обучающей выборки (31 число) и прогноз нейросети, который выводится в столбце **Out** по нажатию на кнопку **Button3** (Forecast).

Таким образом, сравнивая информацию в последних двух столбцах таблицы, можно увидеть: насколько отличается фактическое значение курса $ от нейросетевого прогноза (рис. 3.4).

В программе используются следующие глобальные данные:

```
const eta = 0.01;  // скорость обучения
      n = 1000;  // количество эпох обучения
      input = 30; // размер входного вектора
      m = 10; // размер скрытого слоя
private
   x : array [1 .. 50, 0 .. input+2] of real;
// обучающая выборка и прогнозные значения
   w : array [1 .. m,1 .. input] of real; // матрица весов
   v : array [1 .. m] of real; // вектор весов
   y_s : array [1 .. m] of real; // скрытый слой нейронов
```

```
    y :  real; // выходной сигнал
    t :  real; // целевой сигнал
    e :  real; // сигнал ошибки
    d_y : real; // δ_i = (y_i − t_i) y_i (1 − y_i)
    d_y_s : array[1..m] of real; // δ_i^k = δ_i · v_ik
    err, err_av : real; // ошибка и средняя ошибка сети
    my_file : text; // текстовый файл
    function sigmoid (arg : real) : real;
  // сигмоидальная функция активации

  function TForm1.sigmoid (arg : real): real;
  begin
        sigmoid := 1/(1 + exp( - (arg - 0.5)));
  end;
```

При инициализации сети матрицы весовых коэффициентов заполняются малыми случайными числами. Информация из обучающей выборки читается в массив **X** и для визуализации процесса отображается в сетке **StringGrid1**.

```
procedure TForm1.Button1Click(Sender: TObject);
var i, j : integer;
    d, sum, sum_sq: real; dir, filename : String;
begin
 (* инициализация весов *)
   Randomize;

 //  Заполнение матрицы w случайными числами
  //         из полуинтервала [-0,5; 0,5)

   Randomize;
   for i := 1 to m do
    for j := 1 to input do  w[i,j] := -0.5 + Random;
```

```
//  Заполнение вектора v случайными числами
  //     из полуинтервала [-0,5; 0,5)

for j := 1 to m do  v[j] := -0.5 + Random;

// загрузка данных
   // заполнение 0-й строки сетки StringGrid1
     //   числами от 1 до 31 (номера дней)

    for i := 0 to input do stringgrid1.Cells[i,0] := IntToStr(i + 1);
    stringgrid1.Cells[input+1,0] := 'Out';
    dir := DirectoryEdit1.Directory;
    dir := dir + '\';

// загрузка информации из файлов, хранящихся в каталоге
  //  data в сетку

  for i := 1 to 50 do
     begin
      StringGrid1.RowCount := StringGrid1.RowCount+1;
                    filename := inttostr(i);
                    if i<10 then filename := '0'+filename;
                    filename := dir + filename + '.txt';

// чтение информации из текстового файла

                  AssignFile (my_file, filename);
                  reset (my_file);
                   for j:=1 to input+1 do
                          begin
                            read(my_file,d);
                            StringGrid1.Cells[j-1,i]:=FloatToStr(d);
                            x[i,j]:=d;
```

```
                if j=input+1 then x[i,0]:=d;
            end;
```
// Целевое значение курса $ запоминается на месте элемента с индексом «0» !

Для улучшения сходимости входы сети x_j, $j=\overline{1,n}$, подвергаются **процедуре нормализации** вычитанием их выборочного среднего и нормированием на исправленное выборочное среднее квадратичное отклонение.

Для вектора из обучающей выборки вычисляется выборочное среднее по формуле:

$$\overline{x_в} = \frac{1}{30}\sum_{i=1}^{30} x_i \tag{3.22}$$

Далее вычисляется выборочная дисперсия:

$$D_в = \frac{1}{30}\sum_{i=1}^{30} x_i^2 - \overline{x}_в^2 \tag{3.23}$$

Поскольку количество входов сети мало, для нормализации используется исправленная выборочная дисперсия:

$$s^2 = \frac{30}{29} D_в \tag{3.24}$$

Нормализация входных значений выполняется по формуле:

$$x_{норм} = \frac{x - \overline{x_в}}{s}. \tag{3.25}$$

```
{ НОРМАЛИЗАЦИЯ ДАННЫХ! }

// sum — выборочная средняя
// sum_sq — исправленная выборочная дисперсия
    sum:=0;
    for j := 1 to input do  sum := sum + x[i,j];
```

```
            sum:=sum/input;

            sum_sq:=0;
            for j:=1 to input do   sum_sq:=sum_sq+sqr(x[i,j]);
            sum_sq:=sum_sq/input-sqr(sum);
            sum_sq:=input*sum_sq/(input-1);

          // целевое значение тоже подвергается нормализации

           for j:=0 to input do
              begin
               x[i,j] := (x[i,j] - sum) / sum_sq;
              end;

           // выборочное среднее и исправленная дисперсия
           //  сохраняется в 31-м и 32-м столбцах матрицы x

              x[i,input+1]:=sum; x[i,input+2]:=sum_sq;

           // закрываются файл, цикл по эпохам и процедура

             CloseFile(my_file)
          end;
    end;
```

Сети в процессе обучения последовательно предъявляются примеры из обучающей выборки, и выполняется корректировка весовых коэффициентов по формулам (3.20) и (3.21) метода обратного распространения ошибки.

```
procedure TForm1.Button2Click(Sender: TObject);
    var k,  i,  j, f : Integer;
  begin
  for k := 1 to n do begin {цикл по эпохам}
```

```
   err_av := 0;

   {обучение сети}

   for f := 1 to 50 do   begin  {цикл по примерам}

{вычисление взвешенных сумм для нейронов скрытого слоя}

     for i := 1 to m do begin
       sum := 0;
       for j := 1 to Input do sum := sum + w[i,j] * x[f,j];
       y_s[i] := sum;
     end;

{вычисление активности нейронов скрытого слоя }

                for i := 1 to m do y_s[i] := sigmoid(y_s[i]);

{ вычисление активности выходного нейрона }

sum := 0;
for j := 1 to m do sum := sum + v[j]*y_s[j];
y := sigmoid(sum);

                    {Определение целевого значения}
                        t := x [f,0];

{вычисление ошибки для нейрона выходного слоя и ошибки сети}

        e := y - t;
     err :=  sqr(e)/2;
     err_av := err_av + err;
```

```
    {вычисление величин δ_i }

    d_y := e * y * ( 1 — y ) ;

{модификация весов нейрона выходного слоя}

      for i := 1 to m do
        v[i] := v[i] — eta * d_y * y_s[i];

    {вычисление величин δ_i^k }

                    for i := 1 to m do
                          d_y_s[j] := d_y * v[i];

{модификация весов нейронов скрытого слоя}

     for i := 1 to m do
       for j := 1 to input do
w[i,j]: =w[i,j] — eta * d_y_s[i] * y_s[i] * (1-y_s[i]) * x[f,j];

    end; {конец цикла по примерам}

   end; {конец цикла по эпохам}

 // вывод средней ошибки сети

 Edit1.Text:='Error: '+FloatToStr(sqrt(err_av/n/f));

end; // конец процедуры обучения сети
```

В режиме прогнозирования сеть для каждого примера из обучающей выборки определяет активность выходного нейрона, используя полученные в процессе обучения матри-

цы связей. Вычисленные значения выводятся в последний столбец сетки **StringGrid1** с названием **Out** (рис. 3.4).

```
procedure TForm1.Button3Click (Sender: TObject);
var   i, j : integer;
      sum, sum_sq : real;
begin
   for f:=1 to 50 do // цикл по примерам
       begin

{вычисление взвешенных сумм для нейронов скрытого слоя}

     for i := 1 to m do begin
       sum := 0;
       for j := 1 to Input do sum := sum + w[i,j] * x[f,j];
       y_s[i] := sum;
     end;

{вычисление активности нейронов скрытого слоя }

                for i := 1 to m do y_s[i] := sigmoid(y_s[i]);

{ вычисление активности выходного нейрона }

sum := 0;
for j := 1 to m do sum := sum + v[j]*y_s[j];
y := sigmoid(sum);

{ денормализация выхода y и вывод результата в сетку}

StringGrid1.Cells[input+1,f] :=
   FloatToStrF(y*x[f,input+2]+x[f,input+1], ffFixed, 4, 2);

   end; {конец цикла по примерам}
```

end; {конец процедуры прогнозирования}

После того, как программа отлажена и заработала, следует вначале выполнить инициализацию сети. Затем обучить сеть (при этом уточняется матрица весовых коэффициентов). При повторных запусках процедуры обучения ошибка сети должна уменьшаться. После достижения приемлемого уровня ошибки можно переходить к прогнозированию курса доллара. Заменой обучающей выборки можно научить сеть получать краткосрочные прогнозы курсов других валют.

ЗАДАЧА МЕДИЦИНСКОЙ ДИАГНОСТИКИ: классификация степени активности автономной нервной системы

В последние годы широкое развитие получили технологии реабилитационного лечения психосоматических заболеваний, основанные на использовании биологической обратной связи. Среди них успешно применяется и биоуправляемый игровой тренинг для коррекции стрессиндуцированных состояний у лиц опасных профессий, в спортивной практике, для лечения психосоматических заболеваний, а также для реабилитационного лечения детей, страдающих синдромом гиперактивности и дефицита внимания [29, 97].

В процессе проводимого биоуправляемого игрового тренинга постоянно сохраняется необходимость оценки объективного состояния ведущих физиологических систем организма пациента в режиме on-line [30, 51].

В задачах диагностики в режиме on-line целесообразно использовать быстрые интеллектуальные системы.

При реализации такого рода систем необходима непрерывная обработка большого потока электрофизиологической информации. Высокая производительность интеллектуальных систем обработки электрофизиологической информации мо-

жет быть достигнута за счет выполнения интеллектуальных ядер в виде нейросетевых модулей [40, 70, 77].

Обученные нейросетевые системы оказываются более выигрышными в плане временных затрат на обработку эмпирических данных, обеспечивая при этом такие полезные свойства систем, как нелинейность, адаптивность, отказо- и помехоустойчивость

При этом актуальным является проведение исследований, связанных с клинической оценкой эффективности этих алгоритмов с использованием критериев чувствительности и специфичности.

Работа выполнялась при поддержке проекта РНПВШ.2.2.3.3/4307 и в соответствии с планами проблемной комиссии по хронобиологии и хрономедицине РАМН и научным направлениям медицинского факультета БелГУ «Разработка универсальных методологических приемов хронодиагностики и биоуправления на основе биоциклических моделей и алгоритмов с использованием параметров биологической обратной связи» [40, 48].

Известно, что регулирование параметров, избранных для мониторинга при биоуправлении, в обычных условиях реализуется за счет сочетанной деятельности нескольких координирующих и пусковых иерархических систем. На основе информационного анализа было разработано [31, 52] модельное представление об иерархии управления регуляцией частотой сердечных сокращений, включающей шесть режимов:

1) детерминированный; 2) квазидетерминированный; 3) гармонический; 4) квазигармонический; 5) квазистохастический; 6) стохастический.

Указанные математические модели позволяют, во-первых, выделить и прогнозировать динамику того или иного параметра, относящегося к механизмам регуляции частоты сердечных сокращений в условиях перманентного воздействия извне, а во-вторых, определить характер смены динами-

ческих режимов, а следовательно и функциональных состояний, им соответствующих.

Целью исследования, описанного в данном пункте, является оптимизация диагностических исследований распознавания степени активности автономной нервной системы у здоровых лиц, находящихся в различных функциональных состояниях.

Для достижения поставленной цели необходимо решить следующие задачи:

– разработать модель двухслойной прямонаправленной искусственной нейронной сети;

– сформировать обучающий алгоритм с использованием обратного распространения ошибки;

– разработать клинические критерии эффективности алгоритмов распознавания состояния активности автономной нервной системы по показателям чувствительности и специфичности.

Для решения поставленных задач была использована методология системного анализа, теория управления и теория моделирования. Прежде всего, были разработаны составляющие компоненты модели микроструктурных паттернов вариабельности ритма сердца (HRV).

Составляющие компоненты микроструктурной модели паттерна HRV включают вектор повторяющихся значений предыдущего и последующего межпульсовых интервалов, временную составляющую из нулевых, укорачивающих и удлиняющих коррекций межпульсовых интервалов. Алфавит системы всегда постоянен и включает все классы дифференциальной гистограммы распределения паттерна HRV, включающие норму, тахи- и брадиритмию.

Данная модель рассматривается в виде последовательного развертывания цепи событий, имеющих условно-вероятностный характер. В соответствии с алгоритмом строится дифференциальная кривая распределения и вычисляется

энтропия. Вычисления производятся по основной выборке в 500 кардиоинтервалов: всю полученную шкалу длительностей межпульсовых интервалов делят на классовые интервалы по 0,05 с. Каждый интервал временного ряда регистрируемого вектора кодируется номером классового интервала, соответствующего его длительности.

Таким образом, входной информацией модели служит вектор межпульсовых интервалов, получаемый с помощью блока ввода электрофизиологической информации, включающего датчик пульса на оптопаре.

Исходный временной ряд разбивается на участки, каждый из которых содержит два уровня измерения сигнала. Последующий участок сдвигается по отношению к предыдущему на один временной такт. Пара, составленная из предыдущего и последующего межпульсовых интервалов, «скользит» по временному ряду с шагом, равным единице. Каждая пара определяет временную составляющую из нулевых, укорачивающих и удлиняющих коррекций межпульсовых интервалов.

Вводится в рассмотрение непрерывная случайная величина X, значения которой связываются с длиной и видом коррекции. Все наблюдаемые значения признака X попадают в интервал $(-1{,}55;1{,}55)$, который проходится с шагом $0{,}05$.

Таким образом, алфавит системы включает 61 класс дифференциальной гистограммы распределения паттерна вариабельности ритма сердца.

Информационные показатели модели соответствуют параметрам энтропии ритма сердца. Функциональные показатели модели характеризуются формулами вычисления параметров энтропии ритма сердца, каждый из которых характеризует ту или иную меру процесса.

В частности, нормированная энтропия HN отражает степень активности автономной нервной системы [44].

Нормированная энтропия вычисляется в виде отношения фактической энтропии к максимальной энтропии системы по формуле:

$$H_N = \left(-\sum_{i=1}^{61} p_i \log_2 p_i \right) / \log_2 N \,. \qquad (3.26)$$

Заметим, что энтропия системы с конечным множеством состояний достигает максимума, когда все состояния равновероятны.

В результате анализа нормированной энтропии были получены шесть интервальных классов, которые соответствовали известным функциональным состояниям и дифференцированной степени активности автономной нервной системы человека:

– РВП СНС (резко выраженное преобладание симпатической нервной системы);

– ВП СНС (выраженное преобладание симпатической нервной системы);

– УП СНС (умеренное преобладание симпатической нервной системы);

– НОРМА (равновесное состояние между симпатикусом и парасимпатикусом);

– УП ПСНС (умеренное преобладание парасимпатической нервной системы);

– ВП ПСНС (выраженное преобладание парасимпатической нервной системы).

В соответствии с указанными моделями, эксперт при определении степени активности автономной нервной (АНС) анализировал значение показателя нормированной энтропии.

Алгоритм вычисления нормированной энтропии

1. Найти коррекции ритма сердца по формулам:

$$\Delta RR_i = \frac{RR_{i+1} - RR_i}{100},\ i = 1,...,499. \qquad (3.27)$$

2. Провести группировку (ранжирование) коррекций по интервалам: (-1,55;1,5),…, (-0,1;-0,05), (-0,05;0,05), (0,05;0,1), …,(1,5;1,55)

Количество интервалов m (алфавит системы) фиксировано и равно 61. Поэтому максимальная энтропия для микроструктуры ритма сердца всегда равна $H_0 = \log_2 61 \approx 5{,}93$.

3. Вычислить относительные частоты попадания коррекции в интервал по формуле $p_i = \frac{n_i}{499}$, где n_i - количество коррекций, попавших в i-й интервал.

4. Найти общую энтропию по формуле:

$$H = -\sum_{i=1}^{61} p_i \log_2 p_i. \qquad (3.28)$$

5. Найти нормированную энтропию по формуле

$$h_n = \frac{H}{\log_2 499}. \qquad (3.29)$$

РЕШЕНИЕ ЗАДАЧИ КЛАССИФИКАЦИИ: с помощью двухслойной искусственной нейронной сети

В данном пункте рассматриваются основные этапы моделирования двухслойной прямонаправленной искусственной нейронной сети, построенной для решения задачи автоматического распознавания степени активности автономной нервной системы (АНС) у здоровых лиц, находящихся в различных функциональных состояниях [75].

Использованы методы математического анализа, математической статистики, системного анализа, математического и компьютерного моделирования. Компьютерная реализация искусственной нейронной сети выполнена в свободно распространяемой среде программирования Lazarus.

Для решения задачи классификации степени активности автономной нервной системы была построена модель двухслойной прямонаправленной искусственной нейронной сети (61 – 10 – 6).

В нашем исследовании вычисленная нормированная энтропия использовалась для формирования мнения эксперта и целевых выходных векторов $\overline{d}(d_1,...,d_6)$, предназначенных для обучения искусственной нейронной сети. Позиция правильного выходного класса маркировалась в целевом векторе значением 0,5. Остальные координаты целевого вектора принимали значение: - 0,5.

На входы искусственной нейронной сети подавались частоты интервального вариационного ряда, соответствующие реализациям случайной величины X в алфавите системы.

Координаты входного вектора равны частотам ранжированной совокупности коррекций ритма сердца, сгруппированных по интервалам: (–1,55; 1,5) ,…, (–0,1; –0,05), (–0,05; 0,05), (0,05; 0,1), …, (1,5; 1,55).

Входы сети x_j, $j=\overline{1,61}$, были подвергнуты процедуре нормализации вычитанием их выборочного среднего и нормированием на квадратный корень из их исправленной выборочной дисперсии.

Активность нейронов скрытого слоя y_k^c, $k=\overline{1,10}$, вычислялась по формулам:

$$y_k^c = f\left(\sum_{j=1}^{61} w_{kj}^{(1)} x_j\right), \qquad (3.30)$$

где $w_{kj}^{(1)}$ - весовой коэффициент связи между j-м входом и k-м нейроном скрытого слоя.

В качестве активационной функции был взят гиперболический тангенс, определяемый формулой (3.6) и называющийся в теории нейросетевого моделирования «*биполярным сигмоидом*».

Активность нейронов выходного слоя y_i, $i=\overline{1,6}$, вычислялась по формулам:

$$y_i = f\left(\sum_{k=1}^{10} w_{ik}^{(2)} y_k^c\right), \qquad (3.31)$$

где $w_{ik}^{(2)}$ - весовой коэффициент связи между k-м нейроном скрытого слоя и i-м нейроном выходного слоя.

Номер нейрона выходного слоя, имеющего максимальную активность, служит маркером класса, к которому сеть относит входной пример.

Искусственная нейронная сеть обучена по алгоритму обратного распространения ошибки.

Алгоритм обучения включает следующие этапы:

1. Инициализация сети проводится со случайными значениями весовых коэффициентов.

   При инициализации сети все весовые коэффициенты принимали случайные значения из отрезка $[-0{,}3;0{,}3]$.

2. Вычисление текущих выходных сигналов для случайно выбранного из обучающей выборки входного вектора.

3. Настройка синаптических весов.

Коррекции весовых коэффициентов связи осуществляются в направлении антиградиента целевой функции:

$$E(w(n)) = \left(\sum_{i=1}^{6}\left(f\left(\sum_{k=1}^{10} w_{ik}^{(2)}(n) f\left(\sum_{j=1}^{61} w_{kj}^{(1)}(n) x_j^{(n)}\right)\right) - d_i^{(n)}\right)^2\right)/2, \tag{3.32}$$

где n – дискретное время;

w – матрица весовых коэффициентов связи;

$x_j^{(n)}$ – j-я координата входного вектора, поданного в момент времени n;

$d_i^{(n)}$ – i-я координата соответствующего целевого вектора, сформированного врачом-экспертом;

$f(\cdot)$ – биполярная сигмоидальная функция активации нейронов скрытого и выходного слоев.

Корректировка синаптических коэффициентов связи выполнялась по формулам (3.12) и (3.13). В нашем исследовании скорость обучения $\eta = 0{,}1$.

4. Шаги 2–3 повторяются до тех пор, пока не будет достигнут приемлемый уровень ошибок I и II-го рода на обучающей выборке.

При работе сети в режиме функционирования отклик сети на входной вектор определяется по формулам:

$$y_i(n) = f\left(\sum_{k=1}^{10} w_{ik}^{(2)}(n) f\left(\sum_{j=1}^{61} w_{kj}^{(1)}(n) x_j^{(n)}\right)\right), \tag{3.33}$$

где $i = \overline{1,6}$.

Распознавание класса производится по максимальному уровню выходного сигнала нейрона, связанного при обучении с одним из шести классов: РВП СНС, ВП СНС, УП СНС, Норма, УП ПСНС, ВП ПСНСП.

Для реализации рассмотренного алгоритма классификации автором было разработано программное средство модульной структуры в среде программирования Lazarus.

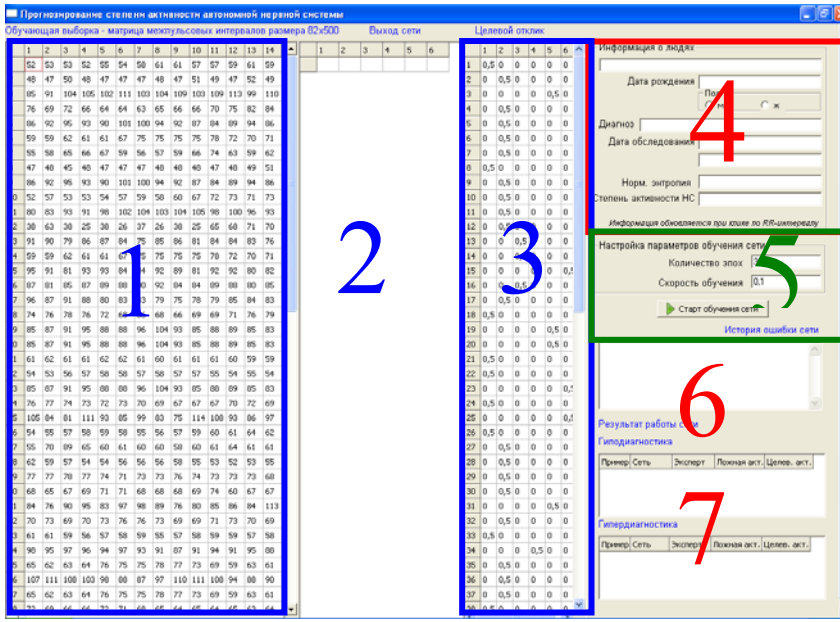


Рис. 3.5. Окно приложения для решения задачи прогнозирования степени активности АНС

Интерфейсную часть можно условно разделить на 7 блоков (рис. 3.5).

В блок 1 загружается обучающее множество, представляющее собой матрицу межпульсовых интервалов. На экране виден только ее фрагмент. Чтобы полностью увидеть данные, нужно воспользоваться полосами прокрутки или клавишами со стрелками.

Во втором блоке отображается фактический отклик сети на конкретную обучающую сессию (рис. 3.6).

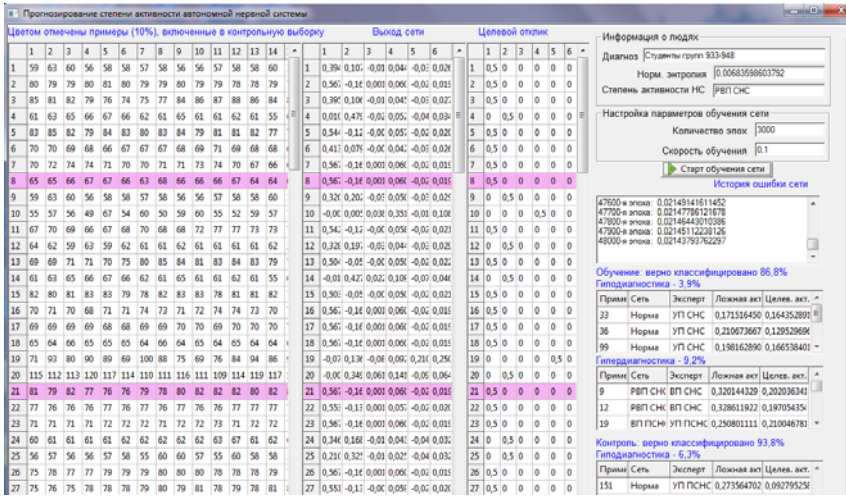


Рис. 3.6. Окно приложения после завершения процессов обучения сети и прогнозирования

В третьем блоке для каждого примера из обучающей выборки выведен целевой отклик нейросети. Значение 0,5 соответствует номеру класса, к которому отнес эксперт данный пример. Сравнивая информацию в блоках 2 и 3, можно оценить близость фактического отклика нейросети и целевого отклика.

Четвертый блок содержит информацию о человеке, которому принадлежит вектор межпульсовых интервалов. Также в данном блоке вычисляется нормированная энтропия и выводится степень активности автономной нервной системы по алгоритму формирования целевого отклика нейросети.

Пятый блок можно рассматривать как функциональный, поскольку в нем можно настраивать параметры, влияющие на результаты обучения сети: скорость и количество сеансов обучения. Значения, установленные по умолчанию, можно менять.

История ошибки сети отображается в шестом блоке. Сеть достаточно быстро справляется с классификацией типичных для своих классов примеров и потом долго подстраивает весовые коэффициенты для правильной классификации пограничных случаев.

В седьмом блоке в табличной форме отображаются текущие результаты обучения с примерами ошибочных решений. При выборе номера примера, подробная информация о нем отображается в блоке 4.

При первом запуске процесса обучения сеть стартует со случайной точки. Поэтому результаты классификации при разных запусках программы могут отличаться.

Как правило, после 30000 обучающих эпох (время обучения несколько минут) сеть устойчиво выходит на 86% верной классификации и ошибается только в граничных случаях. При повторном обучении сети средняя ошибка уменьшается.

ОЦЕНКА ЭФФЕКТИВНОСТИ КЛАССИФИКАЦИИ: проверка адекватности моделирования

В заключительной экспериментальной части работы были проведены исследования на адекватность разработанных моделей реальным электрофизиологическим процессам.

Для этих целей были проанализированы 189 записей межпульсовых интервалов у 94 практически здоровых студентов Белгородского государственного университета. Все обследуемые входили в одну социальную и возрастную группу от 17 до 24 лет [48].

Обучающая выборка включала 139 записей у 69 человек. В экзаменационную выборку входили 25 человек, у которых были проанализированы 50 записей межпульсового интервала.

В табл.3.1 приведены результаты эффективности прогнозирования степени активности АНС, полученные по обу-

чающему множеству примеров при продолжительном обучении.

Таблица 3.1

Анализ эффективности нейросетевого алгорима классификации степени активности автономной нервной системы на обучающей выборке

№ п/п	Степень активности АНС	Общее число	Правильно распознано		Неправильно распознано	
			Ист+	Ист–	Гиподиагно-стика	Гипердиаг-ностика
1	УП СНС	53%	40,5%	12%	0%	0,5%
2	ВП СНС	4%	0,5%	2,0%	0%	1,5%
3	РВП СНС	27%	19%	7,3%	0%	0,7%
4	Норма	6%	4,0%	2,0%	0%	0%
5	УП ПСНС	7%	5,0%	0,7%	0%	1,3%
6	ВП ПСНС	3%	1,0	2,0%	0%	0
7	Итого: человек, %	139–100%	97–70,0%	36–26,0%	–0,0%	6–4,0%

Из представленных в табл. 3.1 данных следует, что нейросетевой алгоритм на обучающей выборке правильно отобрал 94,0 % больных. Неправильно распознано – 4,0 %.

Из них гиподиагностика составила всего 4,0 %, случаев гипердиагностики алгоритм не допустил.

Чувствительность алгоритма распознавания составила 100,0 % (70,0/70,0 + 0,0), специфичность дифференциальной диагностики – 86,7 % (26,0/26,0 + 4,0).

В табл. 3. 2 приведены результаты эффективности распознавания степени активности АНС на примере экзаменационной выборки.

Таблица 3.2

Анализ эффективности нейросетевого алгорима классификации степени активности автономной нервной системы на экзаменационной выборке

№ п/п	Степень активности АНС	Общее число	Правильно распознано		Неправильно распознано	
			Ист+	Ист–	Гиподиагностика	Гипердиагностика
1	УП СНС	6%	4,6%	1%	0,3%	0,1%
2	ВП СНС	48%	32,4%	14%	0,2%	1,4%
3	РВП СНС	20%	20%	0%	0%	0%
4	Норма	2%	2,0%	0%	0%	0%
5	УП ПСНС	16%	4,0%	7,0%	1,5%	3,5%
6	ВП ПСНС	8%	5,0	3,0%	0	0
7	Итого: человек, %	50–100%	34–68%	12–25%	1–2%	3–5%

Из представленных в табл. 3.2 данных следует, что нейросетевой алгоритм на экзаменационной выборке правильно отобрал 93,0 % больных. Неправильно распознано – 7,0 %. Из них гипердиагностика составила 5,0 % и гиподиагностика 2,0 %.

Чувствительность алгоритма распознавания составила 97,1 % (68,0/68,0 + 2,0), специфичность дифференциальной диагностики – 83,3 % (25,0/25,0 + 5,0).

Ошибки классификации составили 7 %, что существенно ниже ошибок распознавания, допускаемых врачом. Нейросетевой алгоритм завысил класс степени активности АНС только в 5 % случаев, а занизил всего лишь в 2 % случаев.

Полученные результаты можно улучшить, устранив недостатки алгоритма обратного распространения ошибки, к которым можно отнести: застревание процесса обучения в локальных минимумах функции ошибки и переобучение сети.

ГЛАВА IV. СТОХАСТИЧЕСКИЕ НЕЙРОННЫЕ СЕТИ

ПРОБЛЕМЫ ОБУЧЕНИЯ МНОГОСЛОЙНЫХ СЕТЕЙ: недостатки алгоритма обратного распространения ошибки

На II-м Международном конгрессе математиков 8 августа 1900 г. Давид Гильберт выделили 23 нерешенные проблемы в математике.

Тринадцатая проблема Гильберта касалась представимости функций нескольких переменных в виде суперпозиции нескольких непрерывных функций двух переменных, в частности, решения уравнения седьмой степени как функции от коэффициентов.

Проблема была решена В.И. Арнольдом совместно с А.Н. Колмогоровым, которые доказали, что любая непрерывная функция любого количества переменных представляется в виде суперпозиции непрерывных функций одной и двух переменных [22].

Приведем теорему Колмогорова, завершившую серию исследований в данном направлении:

«Каждая непрерывная функция n переменных, заданная на единичном кубе n-мерного пространства, представима в виде

$$f(x_1, x_2, ..., x_n) = \sum_{q=1}^{2n+1} h_q \left[\sum_{p=1}^{n} \varphi_q^p (x_p) \right], \qquad (4.1)$$

где функции $h_q(u)$ непрерывны, а функции $\varphi_q^p(x_p)$, кроме того, еще и стандартны, т.е. не зависят от выбора функции f.

Из теоремы Колмогорова следует, что нейронные сети позволяют приблизить произвольную непрерывную функцию $f(x_1, x_2, ..., x_n)$. Следовательно, с их помощью можно аппрок-

симировать функционирование любого непрерывного автомата [14, 92].

Таким образом, с помощью линейных операций и нелинейных функций активации можно приблизить любую непрерывную функцию с желаемой точностью.

Заметим, что нет четкого правила, позволяющего точно определить количество слоёв нейронной сети и количество нейронов в каждом слое, достаточных для приближения неизвестной функциональной зависимости [101, 102].

Поэтому архитектуру нейронной сети для каждой задачи приходится подбирать экспериментально, так же как скорость обучения и диапазон случайного выбора весовых коэффициентов при инициализации сети. Таким образом, подбор комплекса эвристик превращается в искусство [16, 35].

Одним из распространенных методов обучения многослойных искусственных нейронных сетей традиционной архитектуры является метод обратного распространения ошибки, рассмотренный подробно в предыдущей главе.

В основу метода положен алгоритм градиентного спуска по поверхности ошибок. В каждый дискретный момент времени выполняется корректировка весовых коэффициентов в направлении антиградиента функции ошибки.

Корректировка весового коэффициента пропорциональна экспериментально выбранному значению скорости обучения, также она зависит от крутизны склона поверхности ошибок в данной точке.

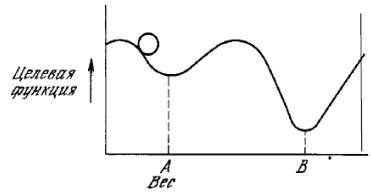


Рис. 4.1. Геометрическая иллюстрация движения текущего значения ошибки сети по поверхности ошибок

Пусть текущее значение функции ошибки, вычисленное в данный дискретный момент времени, представляет собой маленький шарик (рис. 4.1) – точку на графике функции ошибки. В процессе обучения происходит перемещение этой точки по поверхности ошибок, что сопряжено с рядом проблематичных факторов [82, 83].

Если скорость обучения окажется малой для размерности данных задачи, то это приведет к слишком медленной сходимости процесса обучения. Если же скорость обучения будет выбрана слишком большой, то шарик будет все время «прыгать» вокруг точки минимума. При этом большие размеры скачков не позволят приблизиться к желаемому минимуму. В этом случае значение функции ошибки в процессе обучения перестаёт уменьшаться и процесс обучения замирает. Описанная проблема называется «параличом сети».

Улучшению процесса сходимости алгоритма обратного распространения ошибки способствует нормализация входных данных. Нормированные входные значения способствуют более простому выбору значения скорости обучения. Скорость обучения может также уменьшаться по мере приближения к точке минимума [15, 91].

Поверхность функции ошибки может оказаться достаточно сложной и иметь не одну точку минимума. В этом случае при обучении сети исследователя поджидает другая проблема – «застревание» в локальных минимумах.

Шарик будет скатываться в один и тот же овраг (рис. 4.1) в точку А, который может оказаться менее глубоким, чем в точке глобального минимума В.

Решить проблему застревания в локальных минимумах можно, используя стохастические составляющие в коррекции весовых коэффициентов. Подробно этот подход описан в следующем пункте.

Результаты обучения сети методом обратного распространения ошибки связаны с качеством обучающего множества примеров. Если выборка нерепрезентативна, то качественного обобщения информации не произойдет.

В случае разнородной обучающей выборки, которая может иметь место в постоянно меняющейся внешней среде, процесс обучения может никогда не сойтись. При этом ошибка сети не уменьшается, а бесцельно блуждает по множеству примеров.

Таким образом, недостатком метода обратного распространения ошибки является то, что обучающаяся система оказывается чувствительной к размеру коррекции шага, может сходиться к локальным минимумам на поверхности ошибок или вести себя неустойчиво.

СТОХАСТИЧЕСКИЕ МЕТОДЫ ОБУЧЕНИЯ: имитация отжига

В данном пункте описана стратегия коррекции весовых коэффициентов, приводящая процесс обучения сети в точку глобального минимума на поверхности ошибок.

В стохастических методах обучения модифицированные весовые коэффициенты, которые увеличивают текущее значение ошибки, сохраняются с некоторой вероятностью. Такой подход позволяет процессу корректировки весов «выбраться» из точки локального минимума в поисках более глубокого глобального минимума.

Представим поверхность ошибки в виде коробки (рис. 4.1), в которой имеется шарик – точка поверхности, соответствующая текущему значению ошибки. Корректировка весовых коэффициентов приводит к перемещению точки на поверхности ошибок, что можно себе представить как перемещение шарика, происходящее при встряхивании коробки.

Уменьшая с течением времени силу встряхивания, можно добиться того, что скорость движения шарика окажется достаточной для того, чтобы шарик выбрался из точки локального минимума (из А в В), но недостаточной для того, чтобы шарик смог выбраться из окрестности глобального минимума (из В в А).

Подобные процессы происходят при кристаллизации вещества, в том числе при отжиге металлов. При постепенно понижающейся температуре наступает момент, когда атомы уже выстроились в кристаллическую решётку, но ещё возможны переходы отдельных атомов из одной ячейки в другую. Переход атома в другую ячейку происходит с некоторой вероятностью, которая уменьшается с понижением температуры.

В своих работах Л. Эйлер писал: «В мире не происходит ничего, в чём не был бы виден смысл какого-нибудь максимума или минимума».

В устойчивой кристаллической решётке атомы находятся в низкоэнергетическом состоянии. Поэтому переход атома в другую ячейку осуществляется в том случае, если это приводит к состоянию с меньшим уровнем энергии.

Моделирование такого физического процесса называется имитацией отжига. Вариантом имитации отжига является **обучение Больцмана** [65].

Вначале процесса обучения выполняют большие случайные коррекции весовых коэффициентов с сохранением только тех изменений, которые уменьшают целевую функцию ошибки, называемую также *функцией энергии системы*.

Затем средний размер шага постепенно уменьшается. При этом система может сделать случайный шаг в направлении увеличения текущего значения функции ошибки. При этом изменение веса происходит с некоторой вероятностью, зависящей от текущего значения температуры.

Распределение энергетических уровней описывается следующей формулой:

$$p(E) = e^{-\frac{E}{kT}}, \tag{4.2}$$

где $p(E)$ – вероятность того, что система находится в состоянии с энергией E;

k – постоянная Больцмана.

На каждом шаге происходит понижение температуры T (некоторой положительной величины). Алгоритм останавливается при достижении нулевого значения температуры. Говорят, что в конце обучения система «замерзает» (стабилизируется) в низкоэнергетическом состоянии, которое соответствует точке глобального минимума функции энергии E.

Действительно, при высоких температурах $T >> E$ значение $p(E)$ приближается к единице практически для всех энергетических состояний. Высокоэнергетическое состояние почти столь же вероятно, как и низкоэнергетическое. Таким образом, высока вероятность сохранения коррекций весовых коэффициентов, увеличивающих значение функции ошибки.

По мере уменьшения температуры вероятность для высокоэнергетических состояний оказывается меньшей по сравнению с вероятностью для низкоэнергетических состояний. Коррекции, портящие значение ошибки, сохраняются реже.

При приближении температуры к нулю сохранение высокоэнергетического состояния маловероятно.

Алгоритм обучения Больцмана

Шаг 1. Искусственной температуре T присвоить большое начальное значение.

Шаг 2. Выбрать входной вектор из обучающей выборки. Вычислить для него фактический отклик сети и значение функции ошибки сети.

Шаг 3. Придать случайное изменение выбранному весу Δw_{ij} и повторно вычислить фактический отклик и ошибку сети.

Шаг 4. Если значение ошибки уменьшилось, то сохранить изменение веса. В противном случае вычислить вероятность сохранения этого изменения по формуле:

$$p\left(\Delta w_{ij}\right) = e^{\frac{-\Delta w_{ij}}{T}}. \tag{4.3}$$

Выбрать случайное число из равномерного распределения от нуля до единицы.

Если вычисленное значение $p\left(\Delta w_{ij}\right)$ больше выбранного случайного числа, то сохранить изменение веса. В противном случае корректировку весового коэффициента не проводить.

Шаг 5. Повторять шаги 3 и 4 для каждого из весов сети, постепенно уменьшая температуру T, пока не будет достигнуто допустимо низкое значение целевой функции.

Шаг 6. Повторять шаги 2-5 для всех векторов обучающей выборки (возможно неоднократно), пока функция ошибки не станет принимать значения допустимо малые для каждого из них.

Для достижения сходимости к глобальному минимуму скорость уменьшения искусственной температуры должна подчиняться закону:

$$T_N = \frac{T_0}{\ln\left(1+N\right)}. \tag{4.4}$$

Этот результат [24] предсказывает медленную сходимость процесса обучения.

Ускорить процесс сходимости алгоритма можно, изменив формулу (4.3). При этом распределение Больцмана заменяется распределением Коши.

Алгоритм обучения Коши совпадает с алгоритмом обучения Больцмана, в котором формула (4.3) предстаёт в виде:

$$p(\Delta w_{ij}) = \frac{T_N}{T_N^2 + \Delta w_{ij}^2}. \qquad (4.5)$$

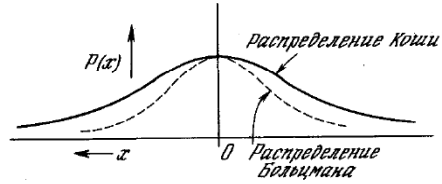


Рис. 4.2. Вид графиков плотностей распределения Больцмана и распределения Коши

Распределение Коши имеет более высокую вероятность больших шагов. При этом максимальная скорость уменьшения температуры становится обратно пропорциональной линейной величине, а не логарифму, что сокращает время обучения [65]:

$$T_N = \frac{T_0}{1+N}. \qquad (4.6)$$

Проблему медленной сходимости стохастических методов обучения можно решить путем их комбинирования с методами градиентного спуска.

КОМБИНИРОВАНИЕ МЕТОДОВ ОБУЧЕНИЯ: объединение метода обратного распространения ошибки с обучением Коши

Основная идея комбинирования алгоритма обратного распространения с обучением Коши заключается в том, что для каждого веса вычисляются две компоненты: случайная, определяемая распределением Коши, и неслучайная, вычисляемая по алгоритму обратного распространения ошибки.

Коррекция весового коэффициента проводится по формуле:

$$w_{ij}^{N+1} = w_{ij}^{N} - \eta\alpha \frac{\partial E}{\partial w_{ij}} + (1-\eta)\Delta w_{ij}^{N} \quad . \tag{4.7}$$

Коэффициент $\eta \in [0;1]$ управляет влиянием каждой из компонент веса на величину коррекции.

После вычисления коррекции весового коэффициента, определяется значение целевой функции. Если имеет место улучшение, то изменение сохраняется. В противном случае изменение сохраняется с вероятностью, определяемой распределением Коши.

Комбинированная сеть обучается быстрее, чем каждый из алгоритмов в отдельности. Сходимость к глобальному минимуму обеспечивается использованием обучения Коши [65].

Рассмотрим подробнее комбинированный способ обучения двухслойной искусственной нейронной сети.

На входы искусственной нейронной сети подаются реализации случайной величины X. Входы сети x_j, $j = \overline{1,n}$, подвергаются процедуре нормализации вычитанием их выборочного среднего и нормированием на выборочное среднее квадратичное отклонение.

Активность нейронов скрытого слоя y_k^c, $k = \overline{1,r}$, вычисляется по формулам:

$$y_k^c = f\left(\sum\nolimits_{j=1}^{n} w_{kj} x_j\right), \tag{4.8}$$

где w_{kj} - весовой коэффициент связи между j-м входом и k-м нейроном скрытого слоя. В качестве активационной функции выбирается гиперболический тангенс.

Активность нейронов выходного слоя y_i, $i = \overline{1,m}$, вычисляется по формулам:

$$y_i = f\left(\sum\nolimits_{k=1}^{r} v_{ik} y_k^c\right), \tag{4.9}$$

где v_{ik} - весовой коэффициент связи между k-м нейроном скрытого слоя и i-м нейроном выходного слоя, $f(s)=\left(e^{s}-e^{-s}\right)/\left(e^{s}+e^{-s}\right)$.

Номер нейрона выходного слоя, имеющего максимальную активность, служит маркером класса, к которому сеть относит входной пример.

При обучении искусственной нейронной сети решается задача минимизации целевой функции

$$E(v,w)=\frac{1}{2}\sum\nolimits_{i=1}^{n}(y_i-t_i)^2 . \qquad (4.10)$$

При инициализации сети все весовые коэффициенты принимают случайные значения из отрезка $[-0{,}3;0{,}3]$.

В процессе обучения пошаговая корректировка весов выполняется в направлении антиградиента целевой функции по формулам:

$$v_{ik}^{N+1}=v_{ik}^{N}-\eta\partial E/\partial v_{ik}^{N}, \qquad (4.11)$$

$$w_{kj}^{N+1}=w_{kj}^{N}-\eta\partial E/\partial w_{kj}^{N}, \qquad (4.12)$$

где N - дискретный момент времени, η - скорость обучения. В нашем исследовании $\eta=0{,}1$.

Найдем по выборке $\{\overline{x}_l,\overline{y}_l\}_{l=1}^{K}$, где K - объем выборки, статистические оценки для величин $\frac{\partial E}{\partial v_{ik}}$ и $\frac{\partial E}{\partial w_{kj}}$.

Получим

$$\frac{\partial E}{\partial v_{ik}}=(y_i-t_i)\left(1-y_i^{\,2}\right)y_k^{c}, \qquad (4.13)$$

$$\frac{\partial E}{\partial w_{kj}}=\left(\sum_{i=1}^{m}(y_i-t_i)\left(1-y_i^{2}\right)v_{ik}\right)\left(1-\left(y_k^{c}\right)^2\right)x_j . \qquad (4.14)$$

Введем обозначение $\delta_i=(y_i-t_i)\left(1-y_i^{\,2}\right)$ и запишем формулы (4.13), (4.14) в виде (4.15), (4.16):

$$\frac{\partial E}{\partial v_{ik}} = \delta_i y_k^c , \tag{4.15}$$

$$\frac{\partial E}{\partial w_{kj}} = \left(\sum_{i=1}^{m} \delta_i v_{ik} \right) \left(1 - \left(y_k^c \right)^2 \right) x_j . \tag{4.16}$$

С учетом (4.15) и (4.16) ключевые формулы метода модификации весовых коэффициентов связи (4.11) и (4.12) принимают следующий вид:

$$v_{ik}^{N+1} = v_{ik}^{N} - \eta \left(\delta_i \right)_l^N \left(y_k^c \right)_l^N , \tag{4.17}$$

$$w_{kj}^{N+1} = w_{kj}^{N} - \eta \left(\sum_{i=1}^{m} \left(\delta_i \right)_l^N v_{ik}^N \right) \left(1 - \left(\left(y_k^c \right)_l^N \right)^2 \right) \left(x_j \right)_l^N . \tag{4.18}$$

В коррекцию синаптического веса следует добавить стохастическую компоненту, используя метод «имитации отжига», базирующийся на распределении Коши.

При этом обучающаяся система может сделать случайный шаг в направлении, увеличивающем текущее значение ошибки, позволяя ей тем самым преодолеть притяжение локального минимума. В конце обучения система «замерзает» и стабилизируется в низкоэнергетическом состоянии.

Перерасчет весов проводиться для целого слоя по формулам:

$$v_{ik}^{N+1} = v_{ik}^{N} - \eta\alpha \left(\delta_i \right)_l^N \left(y_k^c \right)_l^N + \left(1 - \eta \right) v_c , \tag{4.19}$$

$$w_{kj}^{N+1} = w_{kj}^{N} - \eta\alpha \left(\sum_{i=1}^{6} \left(\delta_i \right)_l^N v_{ik}^N \right) \left(1 - \left(\left(y_k^c \right)_l^N \right)^2 \right) \left(x_j \right)_l^N + \left(1 - \eta \right) w_c , \tag{4.20}$$

где η - коэффициент, управляющий долями градиентной и стохастической компонент веса, α - скорость обучения, v_c и w_c - стохастические изменения соответствующих весов.

После корректировки весовых коэффициентов рассчитывается значение целевой функции. Если ошибка уменьшается, то изменения весов сохраняются. В противном случае

изменения сохраняются с "вероятностью", определяемой распределением Коши:

$$p(w) = \frac{T(N)}{T^2(N) + w^2}, \tag{4.21}$$

где $T(N)$ - искусственная температура, рассматриваемая как функция времени:

$$T(N) = \frac{T_0}{1+N}, \tag{4.22}$$

где T_0 - начальная искусственная температура.

Интегрируя $p(w)$ от 0 до w в равенстве (4.20) и разрешая полученное уравнение относительно w, получим значение стохастического изменения веса:

$$w_c = \rho T(N)\ tg(P(w)), \tag{4.23}$$

где ρ - скорость обучения, $P(w)$ - "вероятность" изменения веса на величину w_c.

В качестве $P(w)$ выбиралось случайное число из равномерного распределения на интервале $(-\pi/2; \pi/2)$. Отрицательная часть была введена в рассмотрение для случайного определения знака коррекции.

Апробация элементов представленной методики описана в работах [40, 44]. В среднем после 1000 обучающих эпох сеть устойчиво выходила на 100% верной классификации на обучающей выборке, допуская при этом не более 2% ошибок первого рода на контрольной выборке [48].

Комбинирование обратного распространения с обучением Коши позволило улучшить результаты исследований, проведенных ранее и описанными в [75, 76].

Добавление стохастической составляющей позволяет построить обучающуюся систему, которая сходится быстрее, чем при использовании традиционных алгоритмов градиент-

ного спуска. При этом система находит глобальный минимум и не допускает насыщения сетевых нейронов.

ЗАДАЧА МЕДИЦИНСКОЙ ДИАГНОСТИКИ: комбинированное нейросетевое решение

В рамках развивающегося направления реабилитационной медицины с использованием параметров биологической обратной связи необходима непрерывная обработка большого потока электрофизиологической информации. Высокая производительность интеллектуальных систем обработки такого вида информации может быть достигнута за счет использования интеллектуальных ядер в виде нейросетевых модулей [40, 70].

В данном пункте рассматривается модель стохастической искусственной нейронной сети, построенной для решения задачи автоматического распознавания степени активности автономной нервной системы (АНС) у лиц, находящихся в различных функциональных состояниях.

Решение данной задачи методом обратного распространения ошибки рассмотрено в главе 3.

Входные данные модели представляют собой вектор межпульсовых интервалов, получаемый с помощью блока ввода электрофизиологической информации. Исходный временной ряд разбивается на участки, каждый из которых содержит два измерения сигнала. Последующий участок сдвигается по отношению к предыдущему на один временной такт. Пара, составленная из предыдущего и последующего межпульсовых интервалов, «скользит» по временному ряду с шагом, равным единице. Каждая пара определяет временную составляющую из нулевых, укорачивающих и удлиняющих коррекций межпульсовых интервалов.

Вводится в рассмотрение случайная величина X, значения которой связываются с длиной и знаком коррекции. Все

наблюдаемые значения признака попадают в интервал $(-1{,}55;1{,}55)$, который проходится с шагом $0{,}05$.

Таким образом, алфавит системы включает 61 класс дифференциальной гистограммы распределения паттерна вариабельности ритма сердца, включающий норму, тахи- и брадиаритмию. Вычисления производятся по 500 *RR*-интервалам.

В экспериментальной части работы врачами были проанализированы 189 записей межпульсовых интервалов у 94 практически здоровых студентов Белгородского государственного университета. Все обследуемые входили в одну социальную и возрастную группу от 17 до 24 лет [44, 48].

В результате весь массив экспериментальных данных был разделен на шесть групп, которые соответствовали известным функциональным состояниям и дифференцированной степени активности АНС человека:

1. резко выраженное преобладание симпатической нервной системы (РВП СНС);
2. выраженное преобладание симпатической нервной системы (ВП СНС);
3. умеренное преобладание симпатической нервной системы (УП СНС);
4. равновесное состояние между симпатикусом и парасимпатикусом (Норма);
5. умеренное преобладание парасимпатической нервной системы (УП ПСНС);
6. выраженное преобладание парасимпатической нервной системы (ВП ПСНС).

В нашем исследовании разбиение выборки использовалось для формирования целевых выходных векторов $\bar{t}(t_1,...,t_6)$, предназначенных для обучения искусственной нейронной сети. Позиция правильного выходного класса

маркировалась в целевом векторе значением 0,5. Остальные координаты целевого вектора принимали значение: - 0,5.

На входы искусственной нейронной сети подавались частоты интервального вариационного ряда, соответствующие реализациям случайной величины X в алфавите системы. Входы сети x_j, $j=\overline{1,61}$, были подвергнуты процедуре нормализации вычитанием их выборочного среднего и нормированием на исправленное выборочное среднее квадратичное отклонение.

Активность нейронов скрытого слоя y_k^c, $k=\overline{1,10}$, вычислялась по формулам:

$$y_k^c = f\left(\sum_{j=1}^{61} w_{kj} x_j\right), \tag{4.24}$$

где w_{kj} - весовой коэффициент связи между j-м входом и k-м нейроном скрытого слоя. В качестве активационной функции был взят гиперболический тангенс.

Активность нейронов выходного слоя y_i, $i=\overline{1,6}$, вычислялась по формулам:

$$y_i = f\left(\sum_{k=1}^{10} v_{ik} y_k^c\right), \tag{4.25}$$

где v_{ik} - весовой коэффициент связи между k-м нейроном скрытого слоя и i-м нейроном выходного слоя, $f(s)=\left(e^s - e^{-s}\right)/\left(e^s + e^{-s}\right)$.

Номер нейрона выходного слоя, имеющего максимальную активность, служит маркером класса, к которому сеть относит входной пример.

При обучении искусственной нейронной сети решалась задача минимизации целевой функции ошибки

$$E(v,w)=\frac{1}{2}\sum_{i=1}^{6}\left(y_i - t_i\right)^2 . \tag{4.26}$$

Для обучения сети была использована комбинация градиентного и стохастического методов обучения.

В процессе обучения пошаговая корректировка весов выполнялась не только в направлении оценки антиградиента целевой функции, но и включала стохастическую компоненту.

Перерасчет весов проводился по формулам:

$$v_{ik}^{N+1} = v_{ik}^{N} - \eta\alpha\left(\delta_i\right)_l^N \left(y_k^c\right)_l^N + \left(1-\eta\right)v_c\,, \tag{4.27}$$

$$w_{kj}^{N+1} = w_{kj}^{N} - \eta\alpha\left(\sum_{i=1}^{6}\left(\delta_i\right)_l^N v_{ik}^N\right)\left(1-\left(\left(y_k^c\right)_l^N\right)^2\right)\left(x_j\right)_l^N + \left(1-\eta\right)w_c\,, \tag{4.28}$$

где N - дискретный момент времени, η - коэффициент, управляющий относительными величинами градиентной и стохастической компонент веса, α - скорость обучения, $\delta_i = \left(y_i - t_i\right)\left(1-y_i^{\,2}\right)$, v_c и w_c - стохастические изменения соответствующих весов.

После корректировки весовых коэффициентов рассчитывалось значение целевой функции. Если после корректировки весовых коэффициентов ошибка сети уменьшалась, то изменения весов сохранялись. В противном случае новые веса сохранялись с “вероятностью”, определяемой распределением Коши:

$$p\left(w\right) = \frac{T\left(N\right)}{T^2\left(N\right) + w^2}\,, \tag{4.29}$$

где $T\left(N\right)$ - искусственная температура, рассматриваемая как функция времени:

$$T\left(N\right) = \frac{T_0}{1+N}\,, \tag{4.30}$$

где T_0 - начальная искусственная температура.

Стохастические компоненты коррекции весового коэффициента определялись формулой:

$$v_c = w_c = \rho T\left(N\right)\ tg\left(P\left(w\right)\right), \tag{4.31}$$

где ρ - скорость обучения, $P\left(w\right)$ - “вероятность” изменения веса на величину w_c. В качестве $P\left(w\right)$ выбиралось слу-

чайное число из равномерного распределения на интервале $(-\pi/2;\pi/2)$. Отрицательная часть введена в рассмотрение для случайного определения знака коррекции.

В среднем после 1000 обучающих эпох сеть устойчиво выходила на 100% верной классификации на обучающей выборке, допуская при этом 1-2% ошибок первого рода на контрольной выборке.

Комбинирование обратного распространения с обучением Коши позволило улучшить результаты исследования, проведенного ранее и описанного в [75].

Добавление стохастической составляющей позволило построить систему, которая сходится быстрее, находя при этом глобальный минимум, и не допускает насыщения сетевых нейронов, приводящего к “параличу” сети.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ: разработка автономного нейросетевого приложения

Компьютерное моделирование искусственной нейронной сети, обученной комбинированием градиентного и стохастического методов, было выполнено автором в среде программирования Lazarus.

Было разработано полнофункциональное приложение «**Forecasting_SNS&PSNS**» [50], предназначенное для оценки текущего состояния автономной нервной системы человека по данным, полученным с датчика его пульса. Интерфейс приложения показан на рис. 4.3.

Вспомогательные файлы хранятся в папке Data. Банк примеров для обучения и тестирования сети накапливается в файле input_patterns.txt. Команды, предназначенные для работы с банком примеров, собраны в меню **Выборка**.

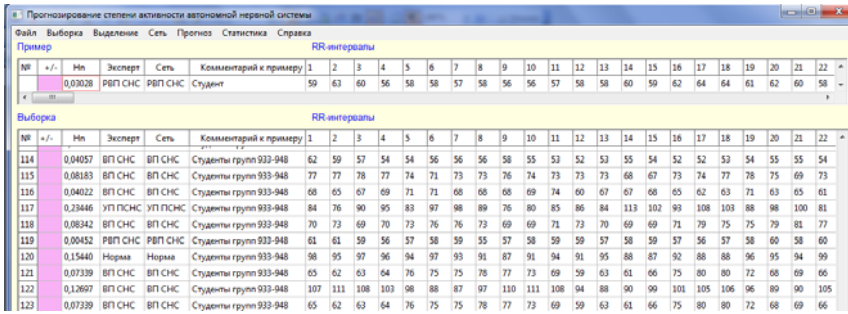


Рис. 4.3. Интерфейс приложения «Forecasting_SNS&PSNS»

По команде **Выборка | Открыть...** появляется диалог открытия файла *.txt, содержащего информацию о группе примеров по шаблону:

Текстовая информация о примере

XX XX XX XX XX ... < RR-интервалы через пробел >

Фрагмент банка примеров показан на рис. 4.4.

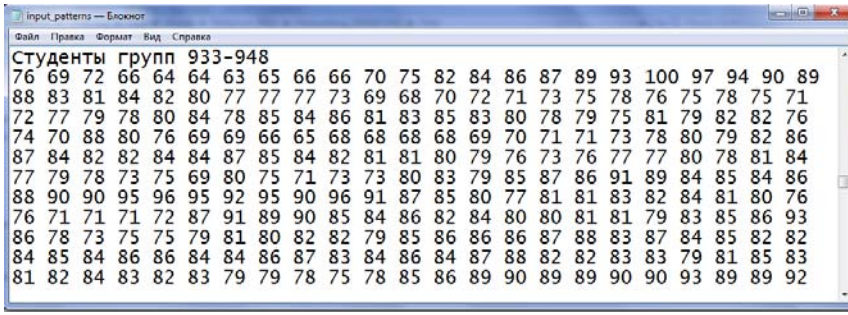


Рис. 4.4. Фрагмент банка примеров для обучения и тестирования сети

Количество *RR*-интервалов в примерах может быть разным. Максимальная длина не должна превышать 600 измерений. После открытия файла input_patterns.txt информация о примерах отображается в основной части формы в группе *Выборка* в табличном виде (рис. 4.3).

Информацию о примерах на форме можно редактировать. Возможно пополнение банка новым примером. Для это-

го нужно открыть файл с информацией о примере по команде **Файл | Открыть...** Информация о примере появиться в верхней части формы в группе *Пример*.

При необходимости можно отредактировать информацию. Сохранить новый пример можно в виде отдельного файла по команде **Файл | Сохранить...** или добавив его в конец имеющегося банка примеров по команде **Выборка | Пример | Добавить.**

Удалить пример из банка можно по команде **Выборка | Пример | Удалить**, предварительно выделив его, кликнув в поле «+/-» в строке примера. При выделении поле «+/-» маркируется цветом. При повторном выборе цветовое выделение исчезает, поле становится светлым.

С помощью команды **Выборка | Пример | Удалить** можно также очистить таблицу *Пример*.

По команде **Выделение | Выделить все** осуществляется выбор всех примеров из группы *Выборка*. Команда **Выделение | Снять выделение** служит для отмены выбора примеров.

Приложение позволяет выполнять автоматическое разбиение имеющегося банка примеров на обучающую и контрольную выборки. Для этой цели используется команда **Выборка | Авторазбиение...** При этом запрашивается требуемый для отбора процент примеров, затем выполняется их случайный выбор и выделение (рис.4.5).

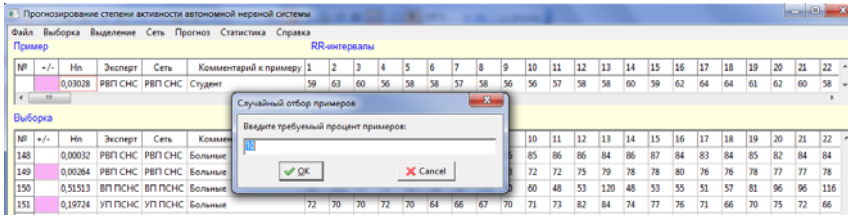


Рис. 4.5. Случайное формирование из банка примеров обучающей и контрольной выборки

По команде **Выборка**|**Сохранить...** выделенные в банке примеры можно сохранить в виде отдельного файла *.txt и использовать его в дальнейшем в виде обучающего или контрольного множества примеров.

Примеры, которые остались невыбранными в результате автоматического разбиения выборки, можно также сохранить в виде отдельного файла, предварительно выделив их по команде **Выделение**|**Инверсия.** Команда позволяет обратить выделение примеров в банке.

Обучающая выборка должна включать не только паттерны RR-интервалов, но и целевые отклики нейросети. По команде **Прогноз**|**Энтропийный** в поле *Hn* выводится значение нормированной энтропии и в поле *Эксперт* под контролем врача уточняется целевой прогноз для обучения сети.

По команде **Выборка**|**Целевое множество...** открывается диалог сохранения в виде файла *.txt целевого набора векторов для выделенных примеров, входящих в обучающую выборку.

Таким образом, разработанное приложение позволяет формировать новые выборки для обучения и тестирования сети, а также пополнять и редактировать старые выборки.

При запуске процесса обучения необходимо выбрать файл, хранящий множество целевых векторов для загруженной в приложение выборки.

По команде **Сеть**|**Создать** выполняется рестарт перед обучением сети со случайным набором весовых коэффициентов. Команда используется для повторного старта обучения сети с новым начальным набором случайных весовых коэффициентов.

При открытии приложения формируется предустановленный случайный набор весовых коэффициентов. Инициализация сети выполняется по умолчанию и сеть оказывается готовой к обучению без выполнения команды **Сеть**|**Создать**.

Обучение сети запускается по команде **Сеть | Обучить...** Перед обучением сети запрашивается информация о настройках процесса обучения (рис. 4.6). В результате уточняется значение скорости обучения и количество обучающих эпох.

Обучение проводится по всей выборке, загруженной в таблицу *Выборка*. При запуске процесса обучения необходимо выбрать файл, хранящий множество целевых векторов для данной выборки.

Кликом по кнопке *Старт обучения сети* запускается процесс обучения. В процессе обучения в информационном окне выводится отчет о текущей средней ошибки сети, полученной по всему множеству обучающих примеров, через каждые сто обучающих эпох. При повторных сеансах обучения информация о предыдущих результатах сохраняется.

При обучении сети с использованием стохастической составляющей текущая ошибка сети может незначительно увеличиваться на локальном этапе (рис. 4.6). Однако в целом в процессе обучения ошибка сети должна уменьшаться.

Программа позволяет проводить неоднократное дообучение сети с сохранением промежуточных наборов весовых коэффициентов в файл. При многократных сеансах обучения для улучшения результатов рекомендуется понижать значение параметра скорости обучения.

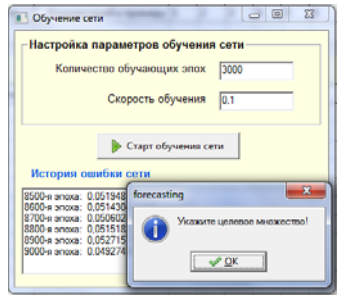


Рис. 4.6. Диалог настроек параметров обучения сети

Для сохранения весовых коэффициентов в файл *.txt предназначена команда **Сеть | Веса | Сохранить...**

После загрузки удачного набора весовых коэффициентов по команде **Сеть | Веса | Загрузить...** сеть сразу готова к прогнозированию степени активности АНС без проведения обучающих сеансов.

Для быстрого получения прогноза по новому примеру достаточно открыть пример, загрузить веса, выделить пример и выполнить прогнозирование.

Прогнозирование выполняется для выделенных примеров в группах *Пример* и *Выборка* по команде **Прогноз | Нейросетевой**. Прогноз выводится в поле *Сеть* (рис.4.5).

После загрузки весовых коэффициентов сеть сразу (без обучения) будет готова к прогнозированию.

Закрытие окна программы и всех дочерних окон выполняется по команде **Файл | Выход**.

По команде **Справка | О программе** выводится информационное окно с руководством пользователя (рис. 4.7).

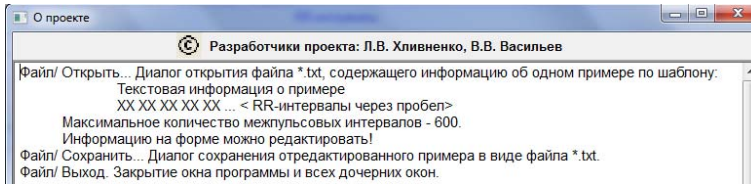


Рис. 4.7. Окно справочного руководства

Тестирование работы обученной сети проводиться с помощью команд меню **Статистика**. По команде **Статистика | Гипердиагностика...** выводится информационное окно, в котором вычисляется процент правильно распознанных примеров среди выделенного множества примеров в таблице *Выборка*, процент гипердиагностики и гиподиагностики. В информационное окно выводится сообщение об ошибочно распознанных примерах из группы *Гипердиагностика*. Сами

примеры, на которых сеть ошибается, выделяются в основном окне приложения в таблице *Выборка*.

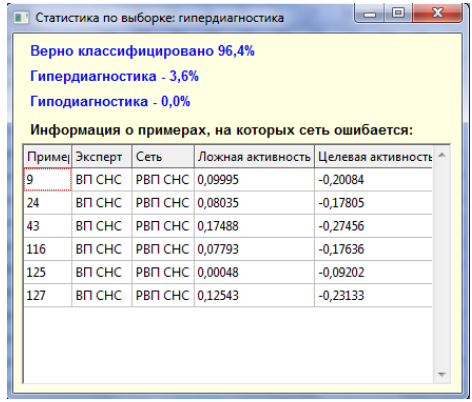


Рис. 4.8. Окно с результатами тестирования сети

По команде **Статистика | Гиподиагностика...** в информационном окне отображается информация об ошибочно распознанных примерах из группы *Гиподиагностика*. Ошибочно распознанные примеры также выделяются в основном окне приложения в таблице *Выборка*.

В заключительной экспериментальной части работы были проведены исследования на адекватность разработанных моделей реальным электрофизиологическим процессам.

Для этих целей были проанализированы 168 записей межпульсовых интервалов у 50 практически здоровых студентов, 47 преподавателей и сотрудников Белгородского государственного университета, а также у 71 пациента МБУЗ "Городская клиническая больница № 1" г. Белгорода. У обследованных пациентов в анамнезе отмечались эндокринные и сердечно-сосудистые нарушения. Все обследуемые входили в разные социальные и возрастные группы.

Таким образом, обучающая выборка включала 168 записей по 500 межпульсовых интервалов.

В результате продолжительного обучения сети была найдена комбинация весов коэффициентов, с помощью кото-

рой выполняется 100% распознавание по всему банку примеров (рис. 4.9).

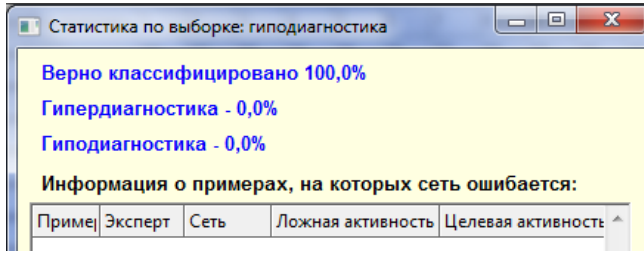


Рис. 4.9. Окно с результатами 100% распознавания примеров

Таким образом, комбинирование обратного распространения с обучением Коши позволило улучшить результаты исследований, описанных в [75, 76].

Разработанное приложение приспособлено для динамического поточного обучения, в котором необходима непрерывная обработка большого потока электрофизиологической информации и высокая производительность интеллектуальных систем ее обработки.

Апробация алгоритма комбинирования градиентных и стохастических методов обучения искусственной нейронной сети показала, что сеть способна обучаться на избыточно больших выборках за счёт того, что использования случайной подвыборки может хватить для вычисления адекватного набора значений весовых коэффициентов [84].

ГЛАВА V. САМООРГАНИЗУЮЩИЕСЯ НЕЙРОННЫЕ СЕТИ

КОНКУРЕНТНОЕ ОБУЧЕНИЕ: «победитель забирает всё»

Конкурентное обучение является вариантом обучения «без учителя». Нейроны сети конкурируют за право представлять близкие точки входного пространства, попавшие в обучающую выборку. После обучения весовые коэффициенты нейронов являются координатами центров групп схожих примеров из обучающей выборки. Такие группы можно считать кластерами или сгустками в пространстве входной информации.

Таким образом, самоорганизующиеся искусственные нейронные сети позволяют ввести своеобразную систему координат во входном пространстве. При этом имеющаяся априорная информация упорядочивается, и выделяются кластеры, которые могут быть объединены в классы [33, 87].

Самоорганизующиеся нейронные сети являются одним из инструментов решения задач категоризации (свободной сортировки) [38].

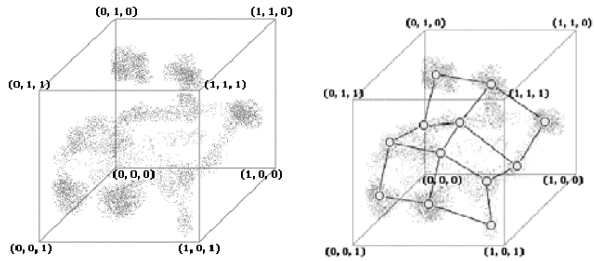


Рис. 5.1. Визуализация трехмерных нормализованных входных данных до (слева) и после (справа) обучения сети. Кружками отмечены центры кластеров

Количество нейронов в самоорганизующейся сети соответствует количеству предполагаемых кластеров. При ини-

циализации сети нейроны получают случайные координаты и произвольно располагаются среди нормализованных данных.

Веса нейрона, оказавшегося наиболее близко к выбранному входному примеру, модифицируются и смещаются в сторону данного примера. Нейрон называют «победителем», так как на этом шаге обучения он выиграл соревнование на представление данного примера в сети. Принцип конкурентного обучения обозначают фразой «победитель забирает всё».

После многократных обучающих сеансов нейроны сети оказываются в центрах близко расположенных друг к другу точек из обучающей выборки. Поэтому сами нейроны служат индикаторами встроенных статистических признаков, содержащихся во входных примерах.

При этом нейроны становятся детекторами признаков различных классов входных образов.

В сети могут существовать обратные связи между нейронами. Обратная связь обеспечивает латеральное торможение, когда каждый нейрон стремится «затормозить» связанные с ним нейроны [60, 62].

Пусть входной пример X имеет координаты $\{x_j\}$, $j=\overline{1,n}$; синаптические веса k–го нейрона обозначаются $\{w_{kj}\}$; скорость обучения равна η.

Тогда корректировка веса нейрона вычисляется так:

$$\Delta w_{kj}=\begin{cases}\eta(x_j-w_{kj}), & \textit{если нейрон } k \textit{ побеждает в соревновании},\\ 0, & \textit{если нейрон } k \textit{ не победитель.}\end{cases}$$

Заметим, что процесс стабилизируется (в том смысле, что $\Delta w_{kj}\to 0$) тогда, когда $w_{kj}\to x_j$ веса нейронов наиболее близко располагаются к координатам примеров из обучающей выборки.

Именно поэтому, в результате конкурентного обучения синаптические веса нейронов оказываются смещенными к центрам тяжести соответствующих кластеров.

Самоорганизующейся сети были предложены финским учёным Тойво Кохоненом как метод проецирования многомерного пространства в пространство с более низкой размерностью, обычно, двумерное.

Идея сети подсказана самой природой. Кора головного мозга человека похожа на плоскость, свернутую складками. Участки, ответственные за близкие части тела, примыкают друг к другу и все изображение человеческого тела словно проецируется на двумерную поверхность.

Архитектура сети Кохонена показана на рис.5.2.

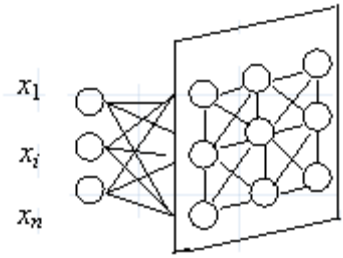


Рис. 5.2. Архитектура сети Кохонена

Каждый из нейронов сети описывается двумя векторами. Один из векторов состоит из весовых коэффициентов связи с координатами входного вектора и имеет такую же размерность, как и входной вектор. Другой вектор состоит из координат самого нейрона на двумерной карте.

Алгоритм конкурентного обучения

Шаг 1. Инициализация сети

Весовым коэффициентам $\{w_{kj}\}$, $k=\overline{1,K}$ $j=\overline{1,n}$ всех нейронов сети присваиваются малые случайные значения из диапазона изменения координат нормализованных входных векторов. Задаются значения α_0 - начальный темп обучения и

D_0 - максимальное расстояние между весовыми векторами (столбцами матрицы W).

Шаг 2. Случайный выбор входного вектора X

Шаг 3. Вычисление квадратов расстояний между вектором X и нейронами сети по формуле:

$$d_k = \sum_{j=1}^{n} \left(x_j - w_{kj}^N \right)^2, \qquad (5.1)$$

где N – дискретный момент времени.

Шаг 4. Выбор нейрона – «победителя»

Номер нейрона k^* определяется наименьшим квадратом расстояния d_k.

Шаг 5. Настройка весов нейрона с номером k^* и всех нейронов, находящихся от него на расстоянии, не превосходящем D_N, по формуле:

$$w_{kj}^{N+1} = w_{kj}^N + \alpha_N \left(x_j - w_{kj}^N \right). \qquad (5.2)$$

Шаг 6. Уменьшение скорости обучения α_N и сужение окрестности нейрона – «победителя» за счет уменьшения значения величины D_N.

Шаг 7. Повторение шагов 2-6 до тех пор, пока суммарное изменение всех весов не станет приемлемо малым.

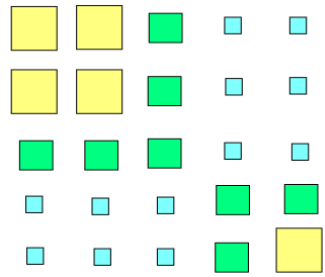


Рис. 5.3. Пример карты самоорганизации Кохонена

Для наглядности карты нейронной активности представляют в виде таблиц, на которых активность нейрона рав-

на площади квадрата, стоящего на месте данного нейрона. Такие карты называют картами самоорганизации Кохонена.

По картам самоорганизации Кохонена или по картам нейронных активностей можно судить о кластеризации пространства входных образов. Одна категория может быть представлена несколькими кластерами.

Обученную сеть можно использовать в режиме классификации. При этом для нового входного вектора потребуется вычислить расстояния до всех нейронов сети. Новый пример относят к группе того нейрона, расстояние до которого оказалось наименьшим [21, 63].

Если выполнить нормировку всех входных векторов, а также если после каждой итерации процесса обучения осуществлять нормировку весов каждого нейрона, то в качестве меры близости входных векторов и весовых векторов нейронов сети можно рассматривать скалярное произведение между ними [65].

Действительно,

$$d_k = \sum_{j=1}^{n}\left(x_j - w_{kj}^N\right)^2 = \sum_{j=1}^{n} x_j^2 - 2\sum_{j=1}^{n} x_j w_{kj}^N + \sum_{j=1}^{n}\left(w_{kj}^N\right)^2 =$$

$$= 2 - 2\sum_{j=1}^{n} x_k w_{kj}^N \,. \qquad (5.3)$$

Наименьшим будет расстояние до того нейрона, скалярное произведение с весами которого у входного вектора максимально.

В этом случае можно считать, что нейрон Кохонена реализует тождественную активационную функцию:

$$f(s) = s \text{, где } s = \sum_{j=1}^{n} w_{kj} x_j \,. \qquad (5.4)$$

МОДЕЛИРОВАНИЕ СЕТИ КОХОНЕНА В EXCEL: решение задачи категоризации объектов жилой недвижимости

Оценка инвестиционной привлекательности проектов на рынке недвижимости является одной из актуальных задач для агентств недвижимости, строительных компаний, а также множества других организаций, деятельность которых связана с инвестициями в объекты недвижимости.

Одной из задач, решаемых при этом, является построение модели ценообразования для жилья. Нейросетевой подход к решению данной задачи предполагает анализ статистических информационных выборок большого объема, в которых влияние разнообразных факторов на целевой параметр (цену) разнопланово и противоречиво.

В данном пункте приведены результаты компьютерного решения задачи категоризации информации о сделках по продажам квартир и выявление на данной основе недооцененных и переоцененных сделок [74].

Для решения задачи категоризации применяется искусственная нейронная сеть с архитектурой Кохонена, обученная по алгоритму self – organizing map. В такой сети нейроны в ходе конкурентного процесса избирательно настраиваются на представление кластеров входной информации.

Для апробации модели по данным газет в 2010 г. была собрана информация о 100 сделках по продажам квартир на территории г. Воронежа.

Требовалось упорядочить информацию, выделив кластеры схожих предложений, сформировав при этом ценовые категории.

На базе решенной задачи категоризации требовалось также решить задачу краткосрочного прогнозирования цены для новых предложений о продажах квартир.

Компьютерное моделирование самоорганизующейся нейронной сети выполнено в Ms Excel с помощью надстройки Kohonen Map из пакета Excel Neural Package [64].

После установки пакета в окне Ms Excel появляется новая панель инструментов, показанная на рис. 5.4.

Кроме компонента Kohonen Map в пакет Excel Neural Package входит инструмент, позволяющий создавать, обучать и использовать многослойные нейронные сети прямого распространения (многослойные персептроны).

Рис.5.4. Инструменты Winnet 3.0 и Kohonen Map

Решение задачи начинается с формирования таблицы исходных данных в документе Ms Excel.

Информация о каждой сделке кодировалась 11-мерным вектором, включающим значения следующих параметров: количество комнат, общая площадь, жилая площадь, тип дома (кирпичный, блочный, панельный), этаж квартиры, этажность дома, площадь кухни, наличие телефона, время ходьбы до остановки, наличие балкона/лоджии, цена (млн.руб).

№	Кол-во ком	Общ. пл.	Жил. пл.	Тип дома	Этаж кв.	Этаж.дома	Пл. кухни	время ход	Телефон	Бал / Лод	Цена
1	1	35	18	3	10	13	11	5	1	1	1,200
2	1	28	10	2	1	9	5	5	1	3	0,800
3	1	30	17,5	2	3	9	6	5	1	3	0,970

Рис.5.5. Фрагмент таблицы исходных данных

Для нечисловых показателей использовались следующие правила представления информации:

- Тип дома: 1 – блочный, 2 – панельный, 3 – кирпичный.
- Наличие балкона/лоджии: 1 – лоджия, 2 – балкон, 3 – отсутствие балкона/лоджии.
- Наличие стационарного телефонного подключения: 1 – наличие телефона, 2 – отсутствие телефона.

Таким образом, входами модели являются 10 микро характеристик квартиры и 1 макро характеристика – цена квартиры.

Выбор инструмента  Kohonen Map приводит к появлению диалогового окна Select data source, в котором требуется указать месторасположение таблицы исходных данных на листе Ms Excel (рис. 5.6).

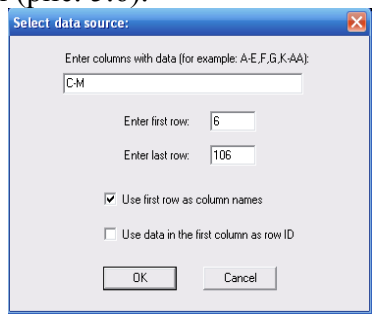


Рис.5.6. Окно выбора месторасположения исходных данных

Основное окно компонента Kohonen Map, показаное на рис. 5.7, содержит две вкладки: Project и Results.

На вкладке Project необходимо указать входные данные модели, выбрав пункт Create patterns…

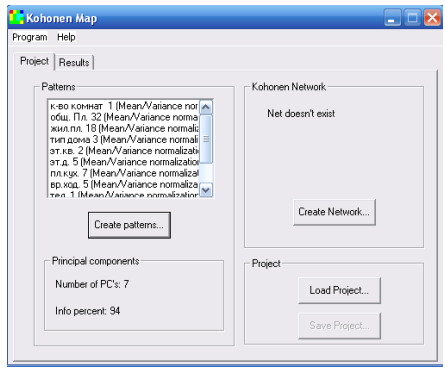


Рис.5.7. Основное окно компонента Kohonen Map

Клик по кнопке Create patterns… приведет к появлению нового диалогового окна Select relevant columns (рис. 5.8), в

котором нужно выбрать все микро характеристики квартиры. По показателю - цена квартиры в дальнейшем будет построена карта Кохонена, элементами которой станут группы квартир близких по стоимости.

В окне Select relevant columns имеется кнопка Normalize…, инициирующая выполнение процедуры нормализации входных данных.

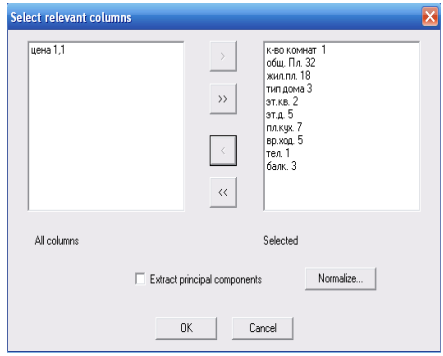


Рис.5.8. Диалоговое окно для определения входов модели

Клик по кнопке Normalize… выводит диалоговое окно Inputs normalization (рис. 5.9), в котором предлагается указать способ проведения нормализации входных данных модели.

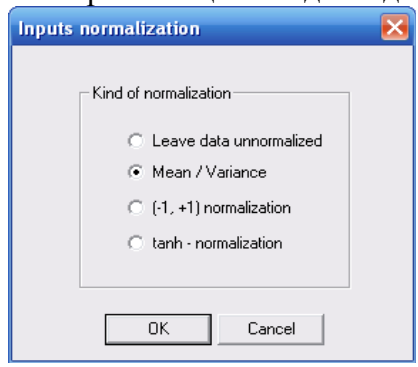


Рис.5.9. Окно выбора способа нормализации входных данных

Нажатие на кнопку OK приводит к автоматическому выполнению процедуры нормализации указанным способом.

В нашем исследовании предобработка входной информации была выполнена методом Mean/Variance, при котором данные о каждой сделке переводятся в безразмерную форму вычитанием выборочного среднего и нормированием на их исправленную выборочную дисперсию.

После подготовки входных данных требуется создать сеть. Для этого в основном окне компонента Kohonen Map (рис. 5.7) необходимо нажать на кнопку Create Network. Появится окно Dialog, показанное на рис. 5.10.

В окне Dialog необходимо указать размеры двумерной решетки, в узлах которой будут располагаться нейроны сети Кохонена.

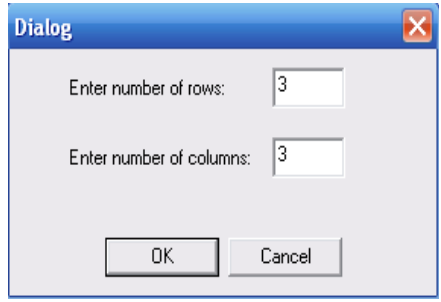


Рис.5.10. Окно инициализации сети Кохонена

В основном окне компонента Kohonen Map (рис. 5.7) на вкладке Results необходимо выбрать ячейку, с которой начнется процесс вывода результатов кластеризации в документ Excel. Первая ячейка выходного столбца должна быть определена в поле First cell (рис. 5.11).

Клик по кнопке Output приведет к выводу на лист результатов кластеризации (рис. 5.12). Каждая строка (ряд) таблицы исходных данных будет отнесена к определенному нейрону в двумерной решетке сети. Местоположение нейрона обозначается буквой столбца и номером строки, на пересечении которых расположен данный нейрон.

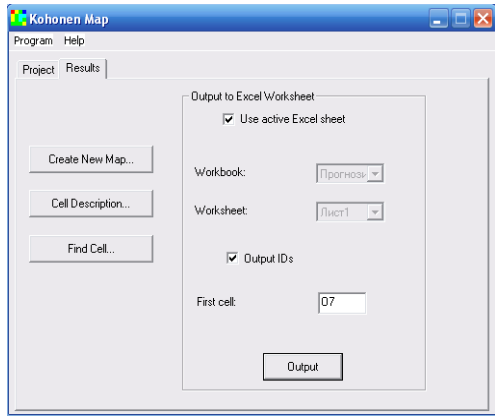


Рис.5.11. Вкладка Results основного окна компонента Kohonen Map

K	L	M	N	O	P	Q
Тел	Бал / Лод	Цена				
1	3	1,100		Row_8	A	3
1	1	1,200		Row_9	A	1
1	3	0,800		Row_10	A	1
1	3	0,970		Row_11	A	2
1	1	1,140		Row_12	A	1
1	3	0,850		Row_13	A	2

Рис.5.12. Фрагмент вывода результатов кластеризации

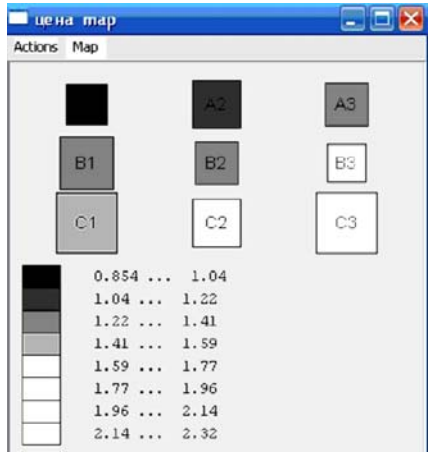


Рис.5.13. Интерактивная карта Кохонена

Нажатие кнопки Create New Map… (рис. 5.11) приведет к визуализации карты Кохонена, показанной на рис. 5.13.

В окне «Цена map» формируется таблица 3x3 из квадратов, площади которых пропорциональны количеству квартир, на представление которых настроились нейроны сети.

В каждый кластер отобраны квартиры из определенной ценовой категории.

Цвет кластера связан с ценовым диапазоном: чем темнее кластер, тем ниже цена квартиры. Построенная карта Кохонена является интерактивной. Клик по квадрату кластера выводит на экран окно с детальной информацией о примерах, попавших в выбранную группу и средних характеристиках по кластеру (рис. 5.14).

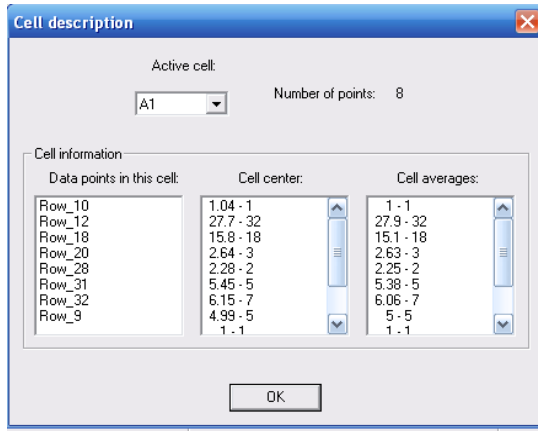


Рис.5.14. Окно характеристик кластера A1

Если построить максимально подробную карту Кохонена, выбрав размеры решетки нейронов 10x10, то такой подход приведет к тому, что каждый из нейронов сети настроится на представление одного примера из обучающей выборки.

После визуализации такой подробной карты вся многомерная информация о квартирах будет спроецирована на двумерную карту и окажется упорядоченной.

На рис. 5.15 приведена такая семантическая карта размером 10х10, на которой выделены цены квартир, отнесенные сетью к различным кластерам при моделировании сети размером 3х3.

Имея информацию о разбиении примеров на кластеры и строя максимально подробную карту, можно убедиться в том, что сеть Кохонена четко отслеживает взаимосвязи между квартирами, выделяя подгруппы квартир с одинаковым количеством комнат, но попадающих при этом в разные ценовые категории.

0,850	0,730	0,800	0,850	0,950	1,100	1,140	1,150	1,200	0,920
0,970	0,820	0,830	0,980	0,945	1,050	1,130	1,300	1,472	1,190
1,200	0,890	1,100	0,900	1,000	1,400	1,545	1,510	1,600	1,740
1,370	1,280	1,350	1,200	1,000	1,200	1,300	1,195	1,370	1,730
1,480	1,340	1,400	1,550	1,180	1,250	1,750	1,950	2,235	1,950
1,040	1,090	1,400	1,500	1,650	1,550	1,650	2,040	1,780	2,450
1,150	1,200	1,750	1,780	1,870	1,500	1,510	2,100	2,100	1,800
1,150	1,250	2,050	1,560	1,500	1,520	2,500	2,200	2,150	1,995
1,250	1,450	2,150	1,620	2,000	1,580	2,500	2,200	2,300	2,930
1,550	1,600	2,200	1,940	1,800	2,200	2,950	2,500	2,550	3,070
1-на ком.кв.				2-х ком.кв.				3-х ком. кв.	

Рис.5.15. Семантическая карта цен на квартиры, состоящая из девяти кластеров и разделенная на три цветные области по количеству комнат в квартирах

Для проверки адекватности построенной компьютерной модели специально были допущены ошибки в цене некоторых квартир с целью выявления сетью переоцененных и недооцененных сделок.

После проведения повторных обучающих сеансов выяснилось, что при обобщении информации во время обучения, сеть корректно определяет кластер квартиры с правильной ценовой группой. «Ошибки» в заявляемых ценах для построенной компьютерной модели играют роль шума во входной информации и при репрезентативной обучающей выборке не влияют на правильность результатов прогнозирования цены квартиры.

Например, для однокомнатной квартиры была завышена стоимость 1,74 млн. руб. Сеть отнесла эту переоцененную квартиру в кластер А3 с ценовым диапазоном 1,22 – 1,41 млн. руб., что значительно ниже запрашиваемой цены.

Инструмент Kohonen Map позволяет «вручную» проводить настройку процесса обучения. Для этого в основном окне (рис. 5.7) необходимо выбрать команду Program | Preferences | Custom. Появится диалоговое окно настроек процесса обучения Program preferences (рис. 5.16).

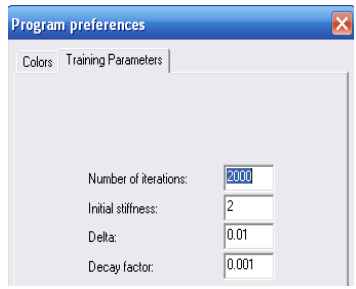


Рис.5.16. Окно настроек параметров процесса обучения

Изменение количества итераций (Number of iterations) влияет на результаты кластеризации. С увеличением значения данного показателя происходит более равномерное распределение образцов по кластерам, что способствует увеличению площадей квадратов, визуализирующих кластер (рис. 5.17).

При малом количестве обучающих сеансов большая часть примеров попадает в две крайние ценовые категории – дешевые и дорогие квартиры. Начиная с 2000 сеансов обучения, увеличение количества итераций перестает влиять на результаты кластеризации. Процесс стабилизируется.

На рис. 5.18 показано влияние значения параметра скорости обучения на результаты кластеризации. Увеличение значения показателя ухудшает результаты кластеризации и способствует появлению «мертвых» нейронов, в группу которых не попадает ни один пример из обучающего множества. При слишком большой скорости обучения все примеры

могут оказаться в одной группе и желаемой кластеризации не произойдет.

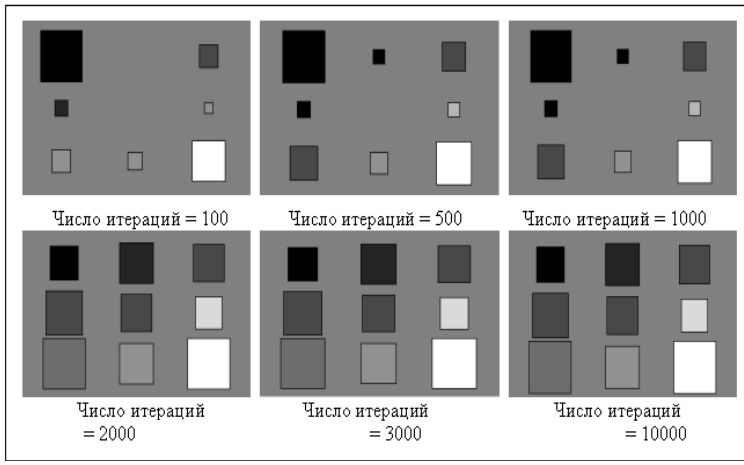


Рис.5.17. Карты Кохонена, построенные при разных значениях количества обучающих итераций

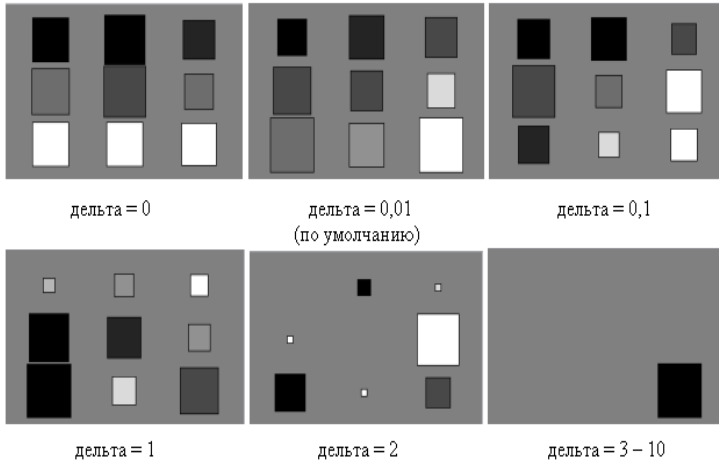


Рис.5.18. Карты Кохонена, построенные при разных значениях показателя скорости обучения

Обученную сеть Кохонена можно сохранить и использовать в дальнейшем в режиме прогнозирования.

Для сохранения сети в основном окне компонента (рис. 5.7) на вкладке Project необходимо выбрать кнопку Create Network… Загрузить сеть можно выбором в том же окне кнопки Load Project…

Для прогнозирования стоимости новых предложений о продажах квартир достаточно пополнить таблицу исходных данных новыми строками, загрузить проект, указать новый размер таблицы исходных данных, обучить сеть и вывести результат в документ Excel.

Например, имеется информация о двух квартирах, которые оценены продавцами в 1,85 и 1,3 млн. руб. Сеть определит первую квартиру в кластер B3 с ценовым диапазоном 1,96 – 2,14 млн. руб., а вторую квартиру - в кластер B1 с ценовым диапазоном 1,22 – 1,41 млн. руб.

Инвестиционная привлекательность первой сделки выше, поскольку квартира является недооцененной, и риелтор может повысить цену за квартиру примерно до средней цены по кластеру – 2 млн. руб.

Проведенное исследование позволяет сделать вывод о том, что метод самоорганизующихся карт является подходящим инструментом для изучения динамики цен на жилье и может применяться при оценке недвижимости.

Обученная нейронная сеть Кохонена позволяет обобщать накопленный опыт проведения сделок в конкретных условиях, сокращать время работы риэлтора и избежать потери прибыли.

КЛАСТЕРИЗАЦИЯ ТОЧЕК ПЛОСКОСТИ: создание генератора обучающей выборки

Рассмотрим учебную задачу, связанную с кластеризацией точек двумерного пространства. Входными данными задачи являются точки плоскости, координаты которых гене-

рируются случайным образом и которые образуют четыре скопления (кластера).

Пусть генерируемые точки пространства R^2 принадлежат квадрату, диагональ которого образована точками (0,0) и (20,20). Мысленно разобьем этот квадрат на четыре равных квадрата, проведя горизонтальную и вертикальную линии по центру фигуры. Пусть сгущения точек имеют место вблизи центров образованных при разбиении квадратов (рис. 5.19).

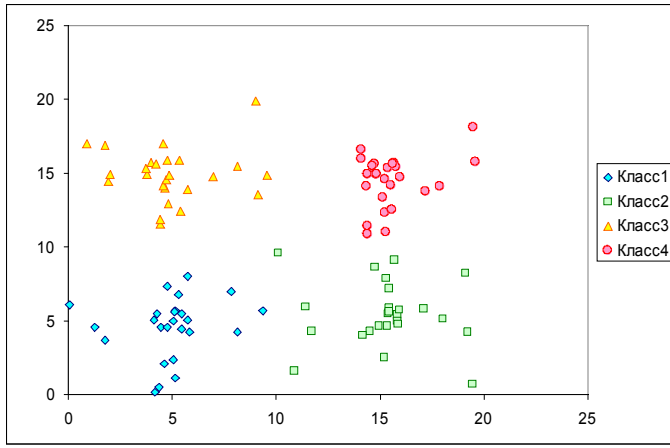


Рис.5.19. Кластеризация точек входного пространства

Рассмотрим способ генерации точек класса 1. Сгущение в центре квадрата с диагональю в точках (0,0) и (10,10) можно получить с помощью преобразования случайной величины $X \sim R(0{,}10)$ в случайную величину Y, график которой показан на рис. 5.20.

Зависимость $Y = f(X)$ *определяется формулой:*

$$y = \begin{cases} 2x, & при \quad x \le 2; \\ \dfrac{x+10}{3}, & при \quad 2 < x < 8; \\ 2x - 10, & при \quad x \ge 8. \end{cases} \qquad (5.5)$$

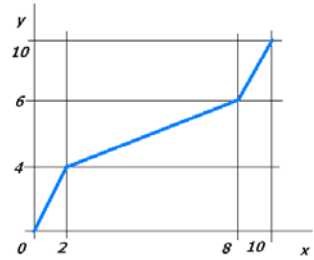


Рис.5.20. График зависимости $Y = f(X)$

Для автоматической генерации выборки исходных данных было разработано приложение, интерфейс которого показан на рис. 5.21.

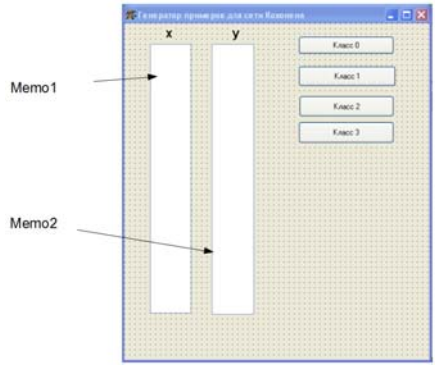


Рис.5.21. Интерфейс генератора обучающей выборки

В программе происходит автоматическая генерация 100 точек из квадрата [0..20, 0..20], разбитых на 4 группы, в каждой из которых имеется одно сгущение точек. Координаты точек сохраняются в текстовые файлы с названиями class1_x.txt , class1_y.txt, class2_x.txt , class2_y.txt, class3_x.txt, class3_y.txt, class4_x.txt , class4_y.txt.

К кнопке «Класс 0» (рис. 5.21) привязана процедура генерации координат 25 точек первого класса, программный код которой приведен ниже.

```
procedure TForm1.Button1Click(Sender: TObject);
 var x,y:integer; i:integer;
begin
  randomize;
  for i:=0 to 24 do
   begin
     x:=random(11);
     y:=random(11);
     if x<=2
       then x := 2*x
       else
         if x>=8 then x := 2*x-10
                 else x := round((x+10)/3);
     if y<=2
       then y:=2*y
       else
         if y>=8 then y:=2*y-10
                 else y:=round((y+10)/3);
     Memo1.Lines[i] := FloatToStr(x);
     Memo2.Lines[i] := FloatToStr(y);
   end;
end;
```

Выбор кнопки «Класс 1» (рис. 5.21) приводит к запуску процедуры генерации 25 точек второго класса.

```
procedure TForm1.Button2Click(Sender: TObject);
var x,y:real; i:integer;
begin
  randomize;
  for i:=0 to 24 do
   begin
     x := random*10 + 10; y := random*10;
     if x<=12
```

```
      then x := 2*x-10
      else
        if x>=18 then x := 2*x-20
              else x := x/3+10;
      if y<=2
       then y:=2*y
       else
        if y>=8 then y:=2*y-10
                 else y:=(y+10)/3;
     Memo1.Lines[i]:=FloatToStr(x);
     Memo2.Lines[i]:=FloatToStr(y);
     Memo1.Lines[25]:=''; Memo2.Lines[25]:='';
  end;
end;
```

Выбор кнопки «Класс 2» (рис. 5.21) приводит к запуску процедуры генерации 25 точек третьего класса.

```
procedure TForm1.Button3Click(Sender: TObject);
var x,y:real; i:integer;
begin
    randomize;
  for i:=0 to 24 do
   begin
     x := random*10+10; y := random*10;
     if x<=12
       then x := 2*x-10
       else
         if x>=18 then x := 2*x-20
                  else x := x/3+10;
      if y<=2
       then y := 2*y
       else
         if y>=8 then y := 2*y-10
```

```
            else y := (y+10)/3;
    Memo2.Lines[i]:=FloatToStr(x);
    Memo1.Lines[i]:=FloatToStr(y);
    Memo1.Lines[25]:=''; Memo2.Lines[25]:='';
   end;
end;
```

Выбор кнопки «Класс 3» (рис. 5.21) приводит к запуску процедуры генерации 25 точек четвертого класса.

```
procedure TForm1.Button4Click(Sender: TObject);
var x,y:real; i:integer;
begin
    randomize;
  for i := 0 to 24 do
   begin
    x := random*10 + 10; y := random*10 + 10;
    if x<=12
      then x := 2*x-10
      else
        if x>=18 then x:=2*x-20
                 else x := x/3+10;
      if y<=12
      then y := 2*y-10
      else
        if y>=18 then y := 2*y-20
                 else y := y/3+10;
    Memo1.Lines[i]:=FloatToStr(x);
    Memo2.Lines[i]:=FloatToStr(y);
    Memo1.Lines[25]:=''; Memo2.Lines[25]:='';
  end;
end;
```

После генерации точек рекомендуется построить диаграмму распределения точек на плоскости (рис. 5.19) и убедиться в том, что в каждой группе имеется одно сгущение

точек. Если точки группы распределены равномерно, то требуется повторно сгенерировать другую серию примеров.

ПРОГРАММИРОВАНИЕ СЕТИ КОХОНЕНА: решение задачи кластеризации

Во время обучения сети Кохонена каждый вектор из входного набора данных связывается с ближайшим к нему нейроном. По завершению обучения весовые векторы нейронов становятся центрами тяжести сгустков близких входных сигналов.

На рис. 5.22 точка из обучающей выборки отмечена знаком «+», узлы карты Кохонена - знаками «о». Иллюстрация показывает смещение узлов карты после модификации весовых коэффициентов в сторону точки из обучающей выборки.

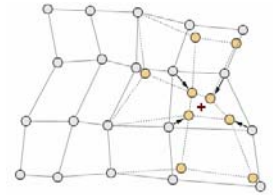


Рис.5.22. Смещение нейронов в сторону примера из обучающей выборки

Модель сети Кохонена для решения задачи кластеризации точек плоскости представляет собой двумерную решетку размером 2x2. Местоположение нейронов в сети обозначается номером строки и столбца, на пересечении которых расположен нейрон.

Интерфейс приложения для компьютерной реализации сети Кохонена показан на рис. 5.23.

Текстовые поля Neuron1-1, Neuron1-2, Neuron2-1, Neuron2-2 служат для вывода значений координат нейронов в

процессе обучения сети. При инициализации сети нейроны приобретают случайные значения координат из квадрата [0..20, 0..20].

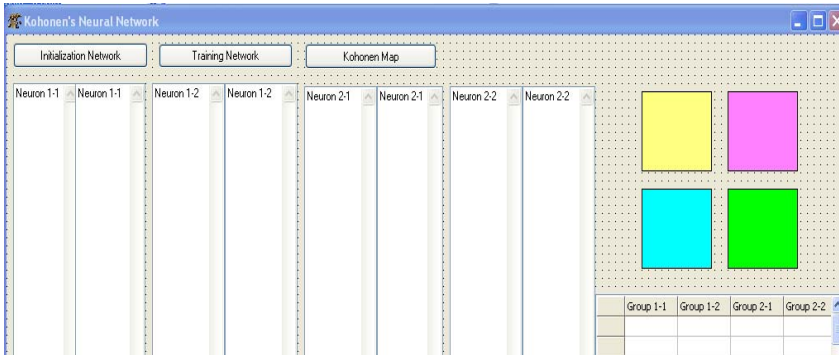


Рис.5.23. Интерфейс приложения для решения задачи кластеризации точек двумерного пространства

Сетка строк с заголовками столбцов Group 1-1, Group 1-2, Group 2-1, Group 2-2 предназначена для вывода информации о сформированных группах примеров, на представление которых настроился каждый из нейронов сети.

Цветные квадраты символизируют группу каждого нейрона, поскольку их площади связаны с численностью группы.

В программе использованы глобальные данные, описанные ниже.

```
const n = 2; // размерность двумерной нейронной решетки
      maxit = 5000; // максимальное число итераций обучения

type primer = record cl, num : integer end; // код примера
...
private
  { private declarations }
  x : array [1..4] of array [1..2] of array [1..25] of integer;
     // обучающее множество примеров
```

```
   w : array [1..n, 1..n, 1..2] of real; // веса нейронов
   d : array [1..n, 1..n] of real; // матрица расстояний
   g  : array [1..n, 1..n, 1..100] of primer; // группы примеров
var
  Form1 : TForm1;
  i, j, k, p, it : integer; // целочисленные переменные
  posi, posj, z : integer; // позиция нейрона-победителя
  count : array [1..n, 1..n] of integer; // численность групп
  d, alpha, min :  real; // вещественные переменные
  t : text; // файловая переменная
  filename, s, test : string; // строковые переменные
```

Создание формы сопровождается загрузкой данных для обучения сети из файлов class1_x.txt , class1_y.txt, class2_x.txt, class2_y.txt, class3_x.txt, class3_y.txt, class4_x.txt, class4_y.txt, формирование которых описано в предыдущем параграфе.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 try
  for k:=1 to 4 do
   begin
    filename := 'class'+IntToStr(k) + '_';
    for i:=1 to 2 do
     begin
       if i = 1
        then filename := filename+'x.txt'
        else
          begin
            delete (filename,8,5); filename := filename+'y.txt'
          end;
       AssignFile(t,filename); reset(t);
       for j:=1 to 25 do
         begin
```

```
        readln(t,s);
        x[k][i][j] := StrToInt(s);
       end;
     CloseFile(t)
   end; {for i}
  end; {for k}
Button2.Enabled:=false;
   Button3.Enabled:=false;
   Shape1.Visible:=false; Shape2.Visible:=false;
   Shape3.Visible:=false; Shape4.Visible:=false;
 except
  ShowMessage('Data file not found!')
 end;
end;
```

Инициализация сети выполняется при нажатии на кнопку «Initialization Network» (рис. 5.23). При этом матрица весовых коэффициентов нейронов заполняется случайными целыми числами из диапазона от 0 до 20.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  randomize;
  for i := 1 to n do
   for j := 1 to n do
    for k := 1 to 2 do
      w[i,j,k] := random(21);
// кнопка для обучения сети становится доступной

  Button2.Enabled:=true;
end;
```

Обучение сети выполняется при нажатии на кнопку «Training Network» (рис. 5.23). В процедуре реализован алгоритм конкурентного обучения сети Кохонена.

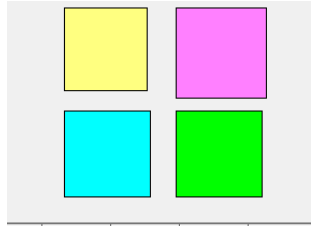
```
procedure TForm1.Button2Click(Sender: TObject);
begin
  alpha:=0.01;
  it := 0;  randomize;
  while it <= maxit do
   begin
     inc(it);
     k := random(4) + 1; p := random(25) + 1;

    // Поиск позиции нейрона-победителя

     min := maxint;
     for i := 1 to n  do
       for j := 1 to n do
        begin
 d [i,j]:= sqr(x[k][1][p]-w[i][j][1]) + sqr(x[k][2][p]-w[i][j][2]);
 if  d [i,j] < min then
          begin min :=  d [i,j]; posi:=i; posj:=j end;
       end;

     // Модификация весов нейрона-победителя

w[posi][posj][1] := w[posi][posj][1] + alpha * (x[k][1][p] - w[posi][posj][1]);
w[posi][posj][2] := w[posi][posj][2] + alpha * (x[k][2][p] - w[posi][posj][2]);

   // вывод значений весов в компоненты Memo через каждые
   // 100 сеансов обучения

   if it mod 100 = 0 then
    begin
      z:= it div 100 +1;
    Memo1.Lines[z] := FloatTostrF(w[1,1,1],fffixed,3,1);
    Memo5.Lines[z] := FloatTostrF(w[1,1,2],fffixed,3,1);
```

```
    Memo2.Lines[z] := FloatTostrF(w[1,2,1],fffixed,3,1);
    Memo6.Lines[z] := FloatTostrF(w[1,2,2],fffixed,3,1);
    Memo3.Lines[z] := FloatTostrF(w[2,1,1],fffixed,3,1);
    Memo7.Lines[z] := FloatTostrF(w[2,1,2],fffixed,3,1);
    Memo4.Lines[z] := FloatTostrF(w[2,2,1],fffixed,3,1);
    Memo8.Lines[z] := FloatTostrF(w[2,2,2],fffixed,3,1);
  end;
 end; {while}
   Memo1.Lines [z+1] := '';  Memo2.Lines[z+1]:='';
   Memo3.Lines[z+1]:=''; Memo4.Lines[z+1]:='';
   Memo5.Lines[z+1]:=''; Memo6.Lines[z+1]:='';
   Memo7.Lines[z+1]:=''; Memo8.Lines[z+1]:='';

// кнопка для вывода результатов становится доступной

   Button3.Enabled:=true;
end;
```

Построение карты Кохонена и вывод информации о составе групп каждого нейрона происходит по нажатию на кнопку «Kohonen Map» (рис. 5.23).

В сетку строк с заголовками столбцов Group 1-1, Group 1-2, Group 2-1, Group 2-2 выводится информация о примерах, попавших в группу соответствующего нейрона.

На рис. 5.24 видно, что нейрон 1-1 настроился на представление примеров второго класса. Нейрон 1-2 собрал в одну группу примеры третьего класса и ошибочно присоединил к ней 22-ю точку второго класса, оказавшуюся на границе между двумя классами точек. Нейрон 2-1 настроился на представление примеров четвертого класса, а нейрон 2-2 представляет первую группу точек.



	Group 1-1	Group 1-2	Group 2-1	Group 2-2
1	2:1	2:22	4:1	1:1
2	2:2	3:1	4:2	1:2
3	2:3	3:2	4:3	1:3
4	2:4	3:3	4:4	1:4
5	2:5	3:4	4:5	1:5

Рис.5.24. Карта Кохонена и фрагменты состава групп

Площади квадратов на рис. 5.24 близки друг к другу, что свидетельствует о приблизительно равной численности нейронных групп.

Заметим, что сеть оказывается чувствительной к выбору стартовых координат нейронов. Поэтому при повторных запусках процедуры инициализации и обучения сети численность и состав нейронных групп могут варьироваться, что влияет и на изменение внешнего вида карты Кохонена.

Процедура анализа результатов обучения сети приведена ниже.

```
procedure TForm1.Button3Click(Sender: TObject);
var h : integer;
begin

  // обнуление счетчиков численности групп в массиве count

   for i := 1 to n do
   for j := 1 to n do count[i,j]:=0;

  for i:=1 to 99 do stringgrid1.Cells[0,i] := inttostr(i);

  // распределение примеров по нейронным группам
```

```
 for k:=1 to 4 do
   for p:=1 to 25 do
    begin

    // нахождение ближайшего нейрона для входного примера
    // из класса k под номером p

     min := maxint;
     for i := 1 to n  do
      for j := 1 to n do
       begin
        d[i,j]:= sqr(x[k][1][p]-w[i][j][1]) + sqr(x[k][2][p]-w[i][j][2]);
        if  d [i,j] < min then
          begin min :=  d [i,j]; posi:=i; posj:=j end;
       end;

   // сохранение информации о примере в нейронной группе

      inc(count[posi,posj]);
     g[posi,posj,count[posi,posj]].cl := k;
     g[posi,posj,count[posi,posj]].num := p;
   end; {for p}

// вывод информации о составе групп в сетку строк
  for i:=1 to n do
   for j:=1 to n do
    begin
      if (i=1) and (j=1) then p:=1; if (i=1) and (j=2) then p:=2;
      if (i=2) and (j=1) then p:=3; if (i=2) and (j=2) then p:=4;
      for k:=1 to count[i,j] do
StringGrid1.Cells[p,k]:=IntToStr(g[i,j,k].cl)+' : '+IntToStr(g[i,j,k].num);
    end;
```

```
// квадраты Shape1-4 становятся видимыми

  Shape1.Visible:=true; Shape2.Visible:=true;
  Shape3.Visible:=true; Shape4.Visible:=true;

// визуализация карты Кохонена

  for i := 1 to 2 do
   for j := 1 to 2 do
   begin
     if (i=1) and (j=1) then p:=1; if (i=1) and (j=2) then p:=2;
     if (i=2) and (j=1) then p:=3; if (i=2) and (j=2) then p:=4;
     h := round (count[i,j] * 3.3);

     with FindComponent('Shape'+IntToStr(p)) as TShape do
      begin
        width:=h; height:=h;
      end; { for j }
   end; {for i}
end;
```

По данным об изменении координат нейронов в процессе обучения сети, которые выводились в текстовые поля Neuron1-1, Neuron1-2, Neuron2-1, Neuron2-2 (рис. 5.23), построены линии тренда значений весовых коэффициентов (рис. 5.25). На диаграмме видны пути от стартовых случайных положений нейронов к центрам своих кластеров.

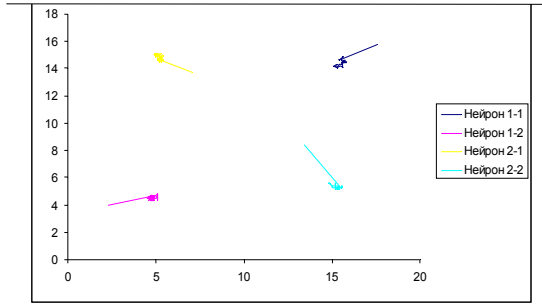


Рис.5.25. Линии тренда весов нейронов к центрам кластеров

На рис. 5.26 приведена диаграмма разброса точек обучающей выборки с указанием дрейфа нейронов решетки при обучении сети.

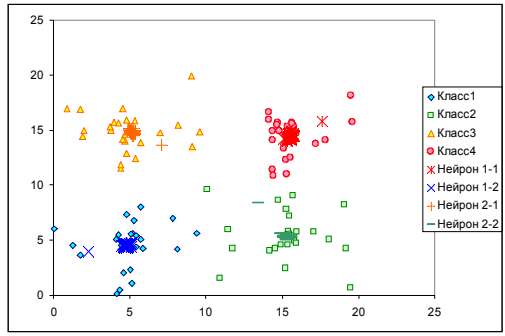


Рис.5.26. Диаграмма исходных данных и изменения местоположения нейронов в процессе обучения

Отдельно стоящие символы «+»,«×»,«-»,«*» соответствуют начальным случайным местоположениям нейронов. При модификации весовых коэффициентов происходит смещение координат нейронов к центрам групп точек в обучающей выборке. На диаграммах 5.25 и 5.26 видно, что нейроны смещены в точки (5;5), (5;15), (15;5), (15;15), что соответствует центрам тяжести соответствующих групп входных примеров.

ГЛАВА VI. РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

АРХИТЕКТУРА СЕТИ ХОПФИЛДА: как функционирует рекуррентная сеть?

Искусственные нейронные сети, в которых имеются обратные связи, называются рекуррентными. Исторически первой архитектурой такой сети была сеть Хопфилда.

Джон Хопфилд описал работу нейронной сети с энергетической точки зрения, введя в рассмотрение аналог функции ошибки сети под названием функции энергии сети и иную модель искусственного нейрона.

Рассмотрим вариант сети Хопфилда в виде однослойной искусственной нейронной сети с обратными связями, состоящей из n входов и n нейронов. Каждый вход такой сети определяет стартовое значение нейронной активности. Нейроны сети связаны друг с другом обратными связями. Выходы сети многократно подаются на входы и изменяют их.

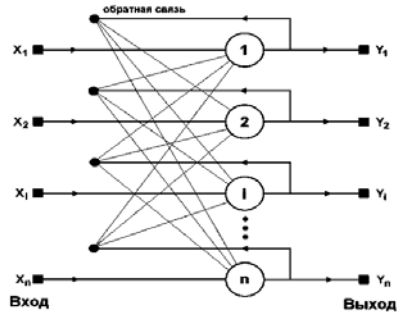


Рис.6.1. Архитектура сети Хопфилда

Выходные значения сети (вектора состояния сети) вычисляются по формулам:

$$y_j^{N+1} = f\left(s_j\right) = \begin{cases} -1, s_j < \Theta_j; \\ 1, s_j > \Theta_j; \\ y_j^N, s_j = \Theta_j. \end{cases} \qquad (6.1)$$

В формулах (6.1) $j = \overline{1,n}$ – номер нейрона в сети; N – дискретный момент времени; $s_j = \sum_{i=1}^{n} w_{ij} \cdot y_i^N$; y_j^N – значение выхода j-го нейрона на шаге N; Θ_j – пороговое значение активации j-го нейрона.

Заметим, что согласно формулам (6.1) выходные значения нейронов бинарные и могут быть равны 1 (нейрон активен) или -1 (нейрон не активен).

В классическом варианте сети Хопфилда входы модели определяют начальные значения активностей нейронов. Поэтому входы модели также бинарные.

Алгоритм функционирования сети Хопфилда

Шаг 1. Определение по входному вектору начальной активности нейронов: $y_j^0 = x_j$.

Шаг 2. Случайный выбор номера нейрона j и вычисление его нового выходного значения по формуле (6.1).

Шаг 3. Повторение шага 2 до тех пор, пока нейроны, выбранные для обновления, не перестанут изменять свое состояние.

Процесс функционирования сети оказывается устойчивым, если последовательные итерации обновления состояний нейронов перестают изменять их выходные значения, и в результате выход сети становится постоянным.

Неустойчивые рекуррентные сети служат примерами хаотических систем. Сети Хопфилда применялись в исследованиях энергетического хаоса и возникновении аттракторов.

В многослойной рекуррентной нейронной сети (сети Элмана) сигнал с внутреннего слоя поступает на дополнительные входы, называемые контекстом, что благоприятно сказывается на устойчивом поведении сети.

Условием устойчивого поведения сети Хопфилда [65] является симметричность связей в матрице весовых коэффициентов и равенство нулю диагональных элементов (у нейрона нет связи с самим собой):

$$\forall i, j:\ w_{ij} = w_{ji},\ w_{ii} = 0,\ i, j = \overline{1,n}. \tag{6.2}$$

Функция энергии сети определяется формулой:

$$E(Y) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j + \sum_{j=1}^{n} \Theta_j y_j. \tag{6.3}$$

Поясним, как выполнение условий (6.2) влияет на устойчивое функционирование сети Хопфилда.

Оценим изменение функции энергии ΔE, вызванное изменением состояния j-го нейрона Δy_j:

$$E(Y) + \Delta E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i \left(y_j + \Delta y_j\right) + \sum_{j=1}^{n} \Theta_j \left(y_j + \Delta y_j\right). \tag{6.4}$$

Пусть выполнены условия (6.2). Тогда связи симметричны и элементы на главной диагонали в матрице связей равны нулю. Имеем:

$$\Delta E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i \Delta y_j + \sum_{j=1}^{n} \Theta_j \Delta y_j. \tag{6.5}$$

$$\Delta E = -\frac{1}{2}\cdot 2\sum_{j=1}^{n} w_{ij} y_i \Delta y_j + \sum_{j=1}^{n} \Theta_j \Delta y_j. \tag{6.6}$$

$$\Delta E = \left(-\sum_{i\neq j} w_{ij} y_i + \Theta_j\right)\Delta y_j = -\left(s_j - \Theta_j\right)\Delta y_j. \tag{6.7}$$

Проверим, что любое изменение состояния нейрона уменьшает функцию энергии, либо оставляет ее без изменения.

Пусть $s_j > \Theta_j$, тогда $\left(s_j - \Theta_j\right) > 0$. По формуле (6.1) новый выход j-го нейрона должен быть равен 1, то есть изменится в положительную сторону или останется без изменения: $\Delta y_j \geq 0$. Следовательно, по формуле (6.7) энергия сети уменьшится или не изменится: $\Delta E \leq 0$.

Пусть $s_j < \Theta_j$, тогда $\left(s_j - \Theta_j\right) < 0$. По формуле (6.1) новый выход j-го нейрона должен быть равен -1, то есть изменится в отрицательную сторону или останется без изменения: $\Delta y_j \leq 0$. Следовательно, по формуле (6.7) энергия сети уменьшится или не изменится: $\Delta E \leq 0$.

Пусть $s_j = \Theta_j$, тогда $\left(s_j - \Theta_j\right) = 0$. Следовательно, по формуле (6.7) энергия сети не изменится: $\Delta E = 0$.

Поскольку функция энергии сети задана на конечном множестве $\left(\forall y_j \in \{-1,1\}\right)$, то функция ограничена снизу и при этом непрерывно стремится к уменьшению. Значит, функция энергии (6.3) должна достигнуть минимума и прекратить изменение.

По смыслу функционирования такая сеть ведет себя устойчиво, и итерационный процесс завершается в соответствии с 3-м шагом алгоритма.

ЗАПОМИНАНИЕ ЭТАЛОННЫХ ОБРАЗОВ: как обучается сеть Хопфилда?

Поверхность функции энергии $E(Y)$, определяемая формулой (6.3), имеет множество локальных минимумов. Вектор активностей нейронов сети $Y = \{y_j\}$, $j = \overline{1,n}$, соответствующий такому локальному минимуму энергии, может интерпретироваться как стационарное состояние или образ памяти нейронной сети.

В процессе устойчивого функционирования рекуррентной нейронной сети происходит сходимость к одному из локальных минимумов функции энергии, что соответствует процессу извлечения некоторого хранящегося в памяти образа [36].

Метод обучения сети Хопфилда запоминанию эталонных образов опирается на правило Хебба, которое подробно рассмотрено в главе II.

Пусть задана обучающая выборка образов X^k, $k = \overline{1,K}$.

Требуется построить матрицу связей W такую, что сеть Хопфилда будет иметь в качестве стационарных состояний образы обучающей выборки.

Пусть в формулах (6.1) и (6.3) пороговые значения равны нулю, т.е. $\forall j : \Theta_j = 0$.

Рассмотрим процесс запоминания одного обучающего образа: $X = (x_1, ..., x_n)$, $x_j \in \{-1;1\}$.

Правило Хебба гласит, что если два нейрона возбуждаются одновременно, то сила связи между ними увеличивается, если возбуждение происходит асинхронно, то уменьшается.

Данное правило обучения сети и условия устойчивости (6.2) приводят к следующим формулам расчета весовых коэффициентов связи между нейронами сети Хопфилда:

$$\forall i \neq j : \; w_{ij} = w_{ji} = x_i \cdot x_j ; \; w_{ii} = 0 ; \; i, j = \overline{1,n} . \qquad (6.8)$$

Проверим, что состояние $Y = X$ является стационарным для сети Хопфилда с матрицей связей $W = \{w_{ij}\}$, определяемой формулами (6.8).

Действительно, значение функции энергии, определяемой формулой (6.3), в состоянии $Y = X$ является глобальным минимумом.

Имеем

$$E(X) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j =$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j x_i x_j = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} (x_i)^2 (x_j)^2 = -\frac{1}{2} n^2. \qquad (6.9)$$

Полученное значение является наименьшим возможным значением функции энергии.

Для запоминания K образов применяется итерационный процесс, при котором формулы расчета весовых коэффициентов принимают вид:

$$\forall i \neq j:\ w_{ij}^k = w_{ij}^{k-1} + x_i^k \cdot x_j^k;\ w_{ii}^k = 0;\ w_{ij}^0 = 0, \qquad (6.10)$$

где $k = \overline{1,K}$ – номер запоминаемого образа; $i, j = \overline{1,n}$.

Таким образом, $\forall i \neq j$ весовые коэффициенты связи между нейронами вычисляются по формулам:

$$w_{ij} = \sum_{k=1}^{K} x_i^k \cdot x_j^k. \qquad (6.11)$$

Заметим, что в сети Хопфилда обучение сети сводится к непосредственному вычислению весовых коэффициентов по имеющимся данным в обучающей выборке.

Итерационный процесс сопровождает функционирование сети. При этом на вход сети можно подать искаженный образ, отличающийся от эталонного образа, хранящегося в памяти. В процессе изменения нейронной активности по формуле (6.1) сеть начнет «исправлять» искажения и будет стремиться свести исходный образ к наиболее похожему эталонному образу. Если искаженный образ окажется «похож» на несколько эталонов, то сеть может вести себя неустойчиво в попытках «подстроить» образ под разные эталоны.

ВОССТАНОВЛЕНИЕ ЗАШУМЛЕННОЙ ИНФОРМАЦИИ: распознавание цифр по неполным изображениям

Рассмотрим компьютерную модель сети Хопфилда для решения задачи распознавания цифр с искажениями и с выполнением восстановления неполного изображения цифры.

На рис. 6.2 показан интерфейс компьютерного приложения для решения поставленной задачи.

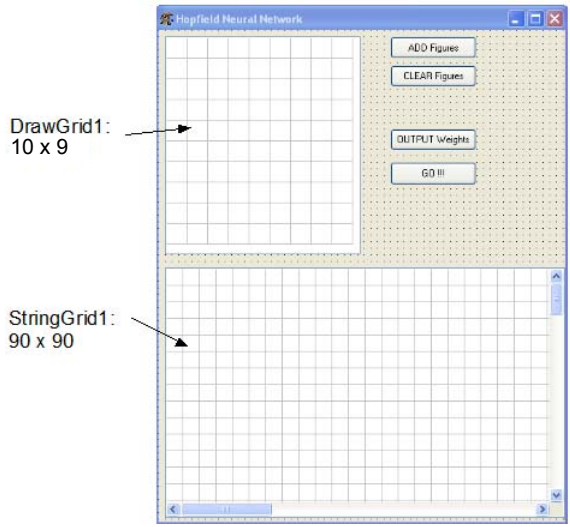


Рис.6.2. Интерфейс приложения для компьютерного моделирования сети Хопфилда

Компонент DrawGrid1 представляет собой клетчатое поле, на котором можно «рисовать» с помощью мыши. Клик по клетке окрашивает поле в красный цвет, повторный клик по окрашенной клетке восстанавливает ее первоначальный белый цвет.

Компонент DrawGrid1 используется для «рисования» эталонных цифр без искажений, совокупность которых определяет обучающую выборку (рис. 6.3).

Размеры поля для рисования 10x9. Таким образом, количество входов сети и нейронов в ней $n = 90$.

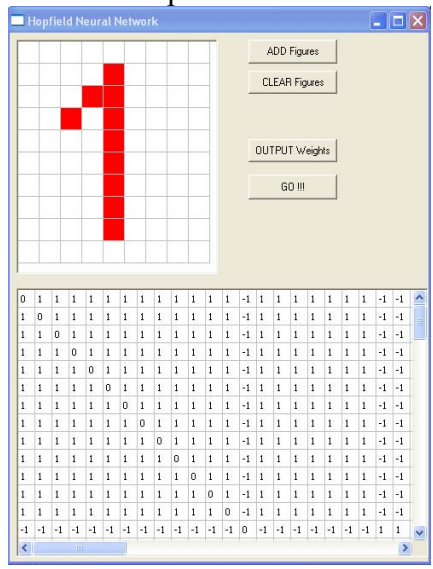


Рис.6.3. Эталонное изображение цифры «1» и фрагмент матрицы весовых коэффициентов после запоминания цифры

Запоминание нового эталонного изображения происходит при нажатии на кнопку *ADD Figures*. При этом пересчитывается матрица весовых коэффициентов по формулам (6.10) и по нажатию на кнопку *OUTPUT Weights* выводится в сетку строк StringGrid1 (рис.6.1).

Заметим, что матрица связей W имеет размер $n \times n$.

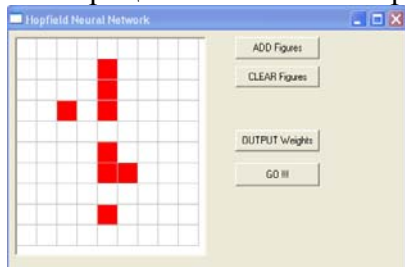


Рис.6.4. Неполное (зашумленное) изображение цифры «1»

Нажатием на кнопку *CLEAR Figures* можно автоматически очистить клетчатое поле и использовать его для подачи на входы сети неполного изображения цифры с искажением (рис. 6.4).

Процесс функционирования сети и пересчета активностей нейронов по формулам (6.1) запускается кликом по кнопке *GO!!!*

При этом через каждые 10000 итераций в клетчатом поле DrawGrid1 выполняется визуализация нейронной активности. Если $y_j = 1$, то нейрон активен и соответствующая ему точка на поле окрашивается в красный цвет. Если $y_j = -1$, то нейрон неактивен и соответствующая точка остается белой.

На рис. 6.5 отображена карта нейронной активности после 10000 шагов по восстановлению зашумленного изображения цифры «1», приведенного на рис. 6.4.

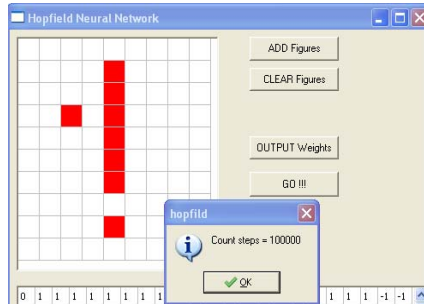


Рис.6.5. Частично восстановленное изображение цифры «1»

При сравнении рис. 6.4 и 6.5 можно отметить, что сеть убрала лишнюю красную точку вне изображения цифры и восстановила одну пропущенную точку на остове цифры.

Полностью цифра восстанавливается после 30000 итераций пересчета активностей нейронов. Результат распознавания показан на рис. 6.6.

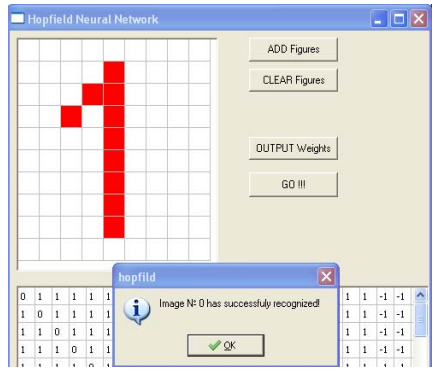


Рис.6.6. Полностью восстановленная цифра «1»

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЕТИ ХОПФИЛДА: как запрограммировать простую рекуррентную сеть?

В заголовке модуля формы требуется описать пользовательские типы, новые процедуры и функции, а также объявить глобальные переменные.

```
type
  amatrix = array [0..9, 0..8] of byte; // матрица входов
  obr = array [0..89] of integer;
                                  // вектор из 90 чисел ∈ {–1;1}

private
  procedure draw_step (ob: obr);
            // визуализация карты нейронной активности
  procedure step (var ob:obr);
   // шаг пересчета активности случайно выбранного нейрона
  function obr_cmp (ob:obr; var res_num:integer):boolean;
      // флаг совпадения текущего состояния с эталоном
  procedure recognize (var ob:obr);
     // процесс функционирования сети
```

```
procedure teach(new_ob: obr);
                                  // расчет весовых коэффициентов

var
  imgmatrix : amatrix; // матрица кодов цветов точек
  i,j,k : integer; // целочисленные переменные
  etalons : array[0..9] of obr; // обучающая выборка
  count_etal : byte; // количество эталонных образов
  W : array[0..89,0..89] of integer; // матрица весов
  x : obr;  // образ для распознавания
```

При создании формы обнуляются матрицы кодов цветов точек на поле для рисования, эталонных образов и весовых коэффициентов.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
   for i:=0 to 9 do
    for j:=0 to 8 do imgmatrix[i,j]:=0;

   for i:=0 to 9 do
    for j:=0 to 89 do etalons[i][j]:=0;
   count_etal := 0;

   for i:=0 to 89 do
    for j:=0 to 89 do W[i][j]:=0;
end;
```

Создадим обработчик события SelectCell для компонента DrawGrid1, связав выбор ячейки с изменением кода цвета ячейки: 0 – белый цвет, 1 – красный цвет.

```
procedure TForm1.DrawGrid1SelectCell (Sender: TObject;
aCol, aRow: Integer;   var CanSelect: Boolean);
```

```
begin
 imgmatrix[arow,acol]:=abs(1-imgmatrix[arow,acol]);
end;
```

Создадим обработчик события DrawCell для компонента DrawGrid1, в котором в зависимости от кодов цветов, хранящихся в матрице imgmatrix, выполняется окрашивание ячеек сетки.

```
procedure TForm1.DrawGrid1DrawCell(Sender: TObject; aCol, aRow: Integer;   aRect: TRect; aState: TGridDrawState);
begin
 with (Sender as TDrawGrid).Canvas do
   begin
      if imgmatrix[arow,acol]=1 then Brush.Color:=clred;
      FillRect(aRect);
   end;
end;
```

Если в данный момент запустить программу на исполнение, то можно убедиться в том, что на клетчатом поле DrawGrid1 (рис. 6.2) теперь можно рисовать.

Напишем код процедуры для выполнения автоматической очистки поля для рисования и обнуления матрицы кодов цветов ячеек сетки.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  DrawGrid1.Enabled:=false;
  for i:=0 to 9 do
    for j:=0 to 8 do imgmatrix[i,j]:=0;
   DrawGrid1.Enabled:=true;
   DrawGrid1.Repaint;
end;
```

Составим обработчик события onClick для кнопки *ADD Figures (рис. 6.2)*, в котором происходит изменение матрицы весовых коэффициентов, связанное с запоминанием эталонного образа, нарисованного в поле DrawGrid1.

```
procedure TForm1.Button1Click(Sender: TObject);
begin

// защита от ввода лишних эталонных образов

 if (count_etal > 9)
   then begin ShowMessage('Memory overflow...'); exit end;

// запоминание эталона с номером count_etal в матрице etalons

  k:=0;
  for i:=0 to 9 do
   for j :=0 to 8 do
    begin
      if imgmatrix[i][j]=0
       then etalons[count_etal][k]:=-1
       else etalons[count_etal][k]:=1;
      inc(k)
    end;

// перерасчет весов по формуле (6.10)

  teach(etalons[count_etal]);
  inc(count_etal);
end;
```

Изменение весовых коэффициентов при запоминании нового образа оформлено в пользовательской процедуре teach.

```
procedure tForm1.teach (new_ob : obr);
var i : integer;
    j : integer;
begin
   for i := 0 to 89 do
    begin
     for j := 0 to 89 do
       if i = j
         then W[i, j] := 0
         else W[i, j] := W[i, j] + new_ob[i] * new_ob[j];
     end;
end;
```

Составим обработчик события onClick для кнопки *OUTPUT Weights (рис. 6.2)*, в котором происходит вывод весовых коэффициентов в таблицу StringGrid1.

```
procedure  TForm1.Button3Click(Sender: TObject);
begin

 //  вывод элементов матрицы W в сетку строк StringGrid1

for i := 0 to 89 do  for j := 0 to 89 do
   StringGrid1.Cells[i,j] := IntToStr(W[i,j]);

end;
```

Заметим, что после запоминания нового эталонного образа, матрица весов изменяется.

Нажатие кнопки *GO!!! (рис. 6.2)* запускает процесс распознавания и восстановления образа цифры, изображенной в поле для рисования.

```
procedure TForm1.Button4Click(Sender: TObject);
begin

  // определение входов сети x по состоянию imgmatrix

k :=0;
  for i :=0 to 9 do
   for j :=0 to 8 do
    begin
      if imgmatrix[i][j]=0
        then x[k]:=-1 else x[k]:=1;
      inc(k)
    end;

// запуск процедуры распознавания и восстановления образа

     recognize(x);
end;
```

Функционирование сети и связанное с ним изменение нейронной активности оформлено в пользовательской процедуре recognize.

```
procedure TForm1.recognize(var ob:obr);
var it, max_it, res_num : integer;
      x : boolean;
      ob_m : obr;
begin
   max_it := 100000;
   it := -1;
```

```
  ob_m := ob;
while not(obr_cmp(ob_m,res_num)) AND (it < max_it) do
   begin
        inc(it);
        step(ob_m); // выполнение шага функционирования сети
   end;

// Вывод количества выполненных итераций

showmessage('Count steps = '+inttostr(it));

// Вывод номера распознанного эталона

  if (res_num=-1)
     then showmessage('Image has not recognized!')
     else showmessage('Image № ' + IntToStr(res_num) +
                          ' has successfuly recognized!')
end;
```

Шаг функционирования сети и однократное изменение нейронной активности оформлено в пользовательской процедуре step.

```
procedure TForm1.step (var ob : obr);
var i : integer;  r : integer;  sum : integer;
begin
   randomize;
   r := random(90);
   sum := 0;
   for i := 0 to 89 do
      sum := sum + ob[i] * W[i, r];
   if sum > 0 then sum := 1 else sum := -1;
   if sum <> ob[r] then
     begin
```

```
        ob[r] := sum;

// Изменение цвета точки при изменении активности нейрона

        draw_step(ob);
      end
end;
```

Рисование шага функционирования сети и визуализация изменения нейронной активности оформлено в пользовательской процедуре draw_step.

```
procedure TForm1.draw_step (ob: obr);
begin
DrawGrid1.Enabled:=false;

// заполнение imgmatrix по текущему состоянию вектора ob

for k := 0 to 89 do
  begin
   if ob[k] = -1
    then imgmatrix[k div 9][k mod 9] := 0
    else imgmatrix[k div 9][k mod 9] := 1
 end;

 DrawGrid1.Enabled:=true;
 DrawGrid1.Repaint;

end;
```

В процедуре recognize используется пользовательская функция obr_cmp(), возвращающая значение True, если текущее состояние нейронной активности полностью совпадает с одним из эталонных образов, хранящихся в памяти сети. В противном случае функция obr_cmp() возвращает значение

False. В заголовке функции описан параметр-переменная res_num, через который в основную программу передается номер распознанного образа.

```
function TForm1.obr_cmp(ob:obr;
                var res_num:integer) : boolean;
var flag,x:boolean;   i,j: integer;
begin
   i := 0; res_num := -1;
   flag := false;
   while (i<=9) and not(flag) do
    begin
     x := true;  j := 0;
     while x and (j<=89)  do
      begin
       x := etalons[i][j]=ob[j];
       inc(j)
      end;
if x then
       begin
         res_num:=i;
         flag:=true
       end;
      inc(i)
    end;
   obr_cmp:=x;
end;
```

При увеличении количества эталонных образов, качество распознавания снижается. Нейронная сеть Хопфилда имеет ограниченный объем памяти. Максимальное количество запоминаемых образов

$$M \approx \frac{n}{\log_2 n}, \qquad (6.12)$$

где n - количество нейронов в сети.

При запоминании слишком большого количества образов, нейронная сеть перестает их распознавать.

Количество итераций при распознавании одного и того же образа может быть разным. Это связано со случайным выбором нейрона, меняющего свое состояние на текущем шаге.

Если запоминаемые образы «похожи» и являются коррелированными, то возможно зацикливание сети на этапе функционирования.

Даже если сеть достигает устойчивого состояния и нейроны перестают изменять свою активность, то это не гарантирует схождение к одному из эталонных образов. Сеть может сойтись к ложному аттрактору, иногда называемому «химерой», состоящему из фрагментов эталонных образов.

Сеть Хопфилда является простейшей и первой в истории архитектурой рекуррентной нейронной сети. Именно с появления сети Хопфилда началось возрождение интереса к искусственным нейронным сетям в середине 80-х годов двадцатого века.

В настоящее время существуют модификации сети Хопфилда. Первым идеи Хопфилда развил Коско в модели гетероассоциативной памяти – нейронной сети Коско.

Нейронная сеть Джордана и ее модификация нейронная сеть Элмана базируются на многослойном персептроне. В таких сетях используются дополнительные входы, называемые «контекстом», на которые поступает обратный сигнал. Наличие «контекста» положительно влияет на устойчивость.

ГЛАВА VII. ЭВОЛЮЦИОНИРУЮЩИЕ НЕЙРОННЫЕ СЕТИ

ЭВОЛЮЦИОННОЕ МОДЕЛИРОВАНИЕ: основные понятия и методы

Под термином «эволюционное моделирование» понимается оптимизационная методология, базирующаяся на аналогии с естественными процессами селекции и генетическими преобразованиями, протекающими в природе. При этом процесс моделирования сложного объекта заменяется моделированием его эволюции [61].

В биологических системах ничего не происходит без кооперации их частей на высоком уровне. Синергетические процессы позволяют биологическим системам трансформировать энергию, предварительно преобразованную на молекулярном уровне, в ее макроформы.

Методы эволюционного моделирования применяются при решении оптимизационных задач в науке и технике [34].

Рассмотрим основные понятия эволюционного моделирования [18, 26].

Эволюция – это синергетический процесс образования новых макроструктур (видов).

Популяция – совокупностью всех «особей».

Особь – это код (строка, запись), состоящий из одной или нескольких хромосом, представляет собой одно из возможных значений целевой функции.

Генотип – это набор хромосом данной особи.

Фенотип – это набор значений данного генотипа.

Каждый бит хромосомы называется *ген*.

Значение каждого гена (0 или 1) называется *аллель*.

Положение гена в стоке – *локус*.

Эволюционный алгоритм – это оптимизационный метод, основанный на эволюции популяции «особей».

Каждая особь связывается со значением *функции приспособленности*, максимизация которой является задачей оптимизации.

Эволюционный процесс представляется как способность “лучших” хромосом оказывать большее влияние на состав новой популяции на основе длительного выживания и более многочисленного потомства [25].

Особенности алгоритма эволюции:

1) Каждая новая популяция состоит из жизнеспособных особей (хромосом).

2) Каждая новая популяция лучше (в смысле целевой функции) предыдущей.

3) В процессе эволюции последующая популяция зависит только от предыдущей популяции.

Рис.7.1. Общая схема эволюционного алгоритма

На рис. 7.1 показана общая схема эволюционного алгоритма, на 1-м этапе которого генерируется множество способов решения – создается *начальная популяция*.

Затем из этих способов отбираются лучшие решения в плане наибольшего (наименьшего) значения функции приспособленности. Происходит *селекция* (отбор).

Отобранные особи могут смешиваться друг с другом. Происходит *скрещивание*. Потомство подвергается случайным небольшим изменениям. Выполняется *мутация*.

Рассмотрим два способа реализации пропорционального отбора особей как одного из методов выполнения селекции [58].

В одном из способов всех особей располагают на колесе рулетки таким образом, что размер кластера пропорционален значению функции приспособленности каждой особи. Случайный запуск рулетки N раз позволяет отобрать N особей для дальнейшего скрещивания. Чем выше приспособленность особи, тем больше у нее шансов быть выбранной. Выбранные особи не снимаются с колеса рулетки и продолжают принимать участие в последующих сеансах отбора.

Рис.7.2. Расположение особей на колесе рулетки

В другом способе реализации пропорционального отбора считается, что у рулетки не одна стрелка, а N. В этом случае за один случайный поворот рулетки отбирается сразу N различных особей.

Из отобранных особей формируется *промежуточная популяция*. Особи промежуточной популяции случайным образом разбиваются на пары и с некоторой вероятностью скрещиваются, в результате образуются два потомка, которые и попадают в новую популяцию. Если особи пары не скрещиваются, то они переписываются в новую популяцию.

В классическом варианте *генетического алгоритма* применяется *одноточечный кроссовер* – один из методов формирования потомков пары особей [12].

Родительские строки делятся на части в точке, выбранной случайным образом. Потомки получаются путем обмена отсеченными частями.

011010.01010001101 => 111100.01010001101
111100.10011101001 => 011010.10011101001

К особям новой популяции применяется оператор мутации, необходимый для решения проблемы «застревания» популяции в локальном минимуме.

Вариантом оператора мутации является инвертирование каждого бита особи с некоторой малой вероятностью.

1110001**0**10110 -> 1110001**1**10110

Итерационный процесс формирования новых поколений останавливается через заданное предельное число итераций или в том случае, когда имеет место схождение популяции.

Схождение популяции означает состояние, при котором все особи «похожи» друг на друга, т.е. все особи находятся в области глобального экстремума функции приспособленности и почти одинаковы (рис. 7.3).

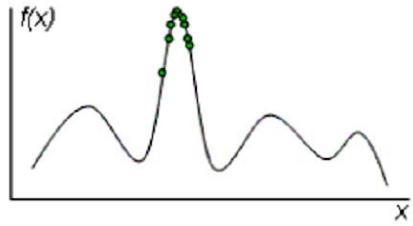


Рис.7.3. Графическая иллюстрация сходимости популяции

Схождение популяции означает, что достигнуто решение, близкое к оптимальному. Итоговым решением задачи является наиболее приспособленная особь последнего поколения.

В островной модели генетического алгоритма начальная популяция разбивается на несколько подпопуляций и процессы формирования нового поколения происходят отдельно

внутри каждой подпопуляции. Можно сказать, что популяции эволюционируют обособленно на изолированных «островах».

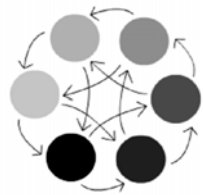


Рис.7.4. Схема миграции в островной модели генетического алгоритма

Через несколько поколений «острова» обмениваются несколькими хорошими особями. Происходит *миграция* (рис. 7.4). Миграция способствует решению проблемы преждевременной сходимости внутри подпопуляций.

Островная схема является вариантом параллельного генетического алгоритма. Одновременно алгоритм запускается несколько раз и происходит совмещение разных хороших решений, что сокращает время поиска глобального экстремума.

МНОГОАГЕНТНЫЕ СИСТЕМЫ: как создать виртуальный мир?

Многоагентной называется система, образованная несколькими взаимодействующими интеллектуальными искусственными организмами, называемыми агентами.

Многоагентаная система характеризуется:

1. автономностью интеллектуальных агентов,
2. ограниченностью представления агентов о «виртуальном мире»,
3. децентрализованной стратегией управления.

Многоагентные системы применяются для моделирования популяций, динамику которых сложно описать дифференциальными уравнениями из-за разнообразия поведенческих стратегий отдельных индивидуумов.

Многоагентные системы используются для исследования сложных систем: биологических, социальных, политических, экономических, исторических.

В прикладном аспекте многоагентные системы находят свое применение в фильмах и компьютерных играх, в области сетевых и мобильных технологий, в проектировании разведывательных и боевых устройств.

При изучении сложных систем (биологических, социальных, политических, экономических, исторических и др.) исследователя часто интересуют общие закономерности и тенденции развития самой системы, а не отдельных ее элементов [48, 74]. Выделение эмерджентных свойств системы сопряжено с рядом сложностей, которые могут быть связаны с излишней детализацией и большим количеством связей между элементами системы [66, 78].

Воздействие на одну часть системы приводит к изменениям в других ее частях и почти всегда сопровождается побочными эффектами. Возникающие при этом петли обратной связи сопряжены с задержками во времени. Результат управляющего воздействия не проявляется мгновенно и часто не соответствует цели управления. Чем сложнее организована система, тем более она инертна и устойчива, тем сложнее найти ее точки бифуркации [67, 76, 80].

Решение парадокса моделирования надежной системы из большого количества ненадежных элементов подсказано самой природой. Биологические нейронные сети при некритическом поражении способны самостоятельно восстанавливаться. При этом сохранившие жизнеспособность нейроны берут на себя функции потерянной части системы [55].

Способность к самовозрождению сложной системы обусловлена распределенным хранением знаний, децентрализованной стратегией управления и коллективным принятием решений.

Всеми перечисленными свойствами обладают математические модели, в основу которых положена технология искусственных нейронных сетей [54, 56, 57].

В данной главе описана методика построения модели многоагентной системы, каждым элементом которой управляет отдельная искусственная нейронная сеть прямого распространения, обучение которой происходит в процессе функционирования агента в виртуальном мире.

Апробация методики проводилась при моделировании виртуального мира, который представляет собой двумерное ячеистое пространство. В ячейках находятся агенты и пища, которая появляется с малой вероятностью при фиксированных размерах виртуального мира.

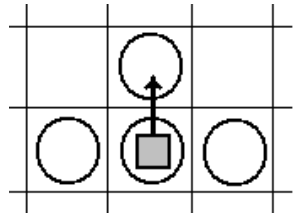


Рис. 7.5. Поле зрения агента

Агенты ориентированы в пространстве. В поле зрения агента попадает содержимое четырех ячеек: той, в которой находится агент; впереди, слева и справа от направления взгляда агента. Окружностями отмечены клетки, из которых агент получает информацию, квадрат – это сам агент. Стрелкой показано направление взгляда агента [4].

При инициализации мира направление взгляда и местоположение агента выбирается случайно. В поле зрения агента попадает пища и другие агенты.

Агенты автономны, поскольку каждый из них принимает решение о совершении некоторого желаемого действия самостоятельно, по результатам контроля своего внутреннего

состояния. Для этого у агентов есть входные сигналы, позволяющие анализировать собственный энергетический ресурс.

Сигналы «много и мало энергии» формируются при сравнении с максимальным значением энергии. Сигнал «радости» появляется при пополнении энергетического запаса, например, при питании или удачной атаке. Все агенты имеют постоянный входной сигнал, позволяющий идентифицировать ситуацию, когда «нечего делать». Подобная ситуация возникает, когда у агента достаточно энергии и нет внутренних побудительных мотивов для совершения действия, и при этом вокруг ничего не видно, то есть нет и внешних побудительных стимулов.

Таким образом, из окружающей среды агент получает информацию в виде бинарных сигналов:

- *сигналы* x_1, x_2, x_3, x_4 – информация о наличии пищи в клетке с агентом, впереди, слева и справа от направления его взгляда;
- *сигналы* x_5, x_6, x_7 – информация о наличии другого агента впереди, слева или справа от направления взгляда данного агента.

У агентов с мотивациями по умолчанию доступны входные сигналы, позволяющие анализировать собственный энергетический ресурс:

- *сигнал* x_8 – мало энергии; *сигнал* x_9 – много энергии;
- *сигнал* x_{10} – увеличение энергии.

Сигналы x_8 и x_9 формируются при сравнении с максимальным значением энергии $E_{\max} = 1500$, одной из характеристик «виртуального» мира. Сигнал «радости» x_{10} появляется при питании или удачной атаке. Все агенты имеют постоянный *сигнал* x_{11}, позволяющий идентифицировать ситуацию, когда энергии достаточно, а побудительные сигналы из внешней среды не поступают.

Действиями агента управляет искусственная нейронная сеть прямого распространения. В простейшем варианте моделирования используется однослойная сеть с линейными нейронами.

Фактически геном агента определяется матрицей синаптических связей, управляющей его действиями искусственной нейронной сети.

Каждому возможному действию соответствует выходной нейрон. В однослойной сети с линейными нейронами значения выходных сигналов равны скалярному произведению входного вектора на вектор весовых коэффициентов, определяющих силу связи между всеми входами сети и конкретным выходным нейроном.

В каждый момент времени агент выполняет действие, соответствующее максимальному выходному сигналу – взвешенной сумме входов:

- y_1 – питаться; y_2 – прыгать вперед;
- y_3 – повернуться налево; y_4 – повернуться направо;
- y_5 – отдыхать; y_6 – делиться.

В популяции с конкурирующими агентами возможны действия:

- y_7 – нападать и y_8 – защищаться.

При атаке энергия нападающего агента увеличивается в том случае, если атакуемый не защищается и уменьшается в противном случае. Если атакуемый защищается, то его энергия увеличивается, если нет – уменьшается.

Начальная популяция агентов имеет «врожденные инстинкты»: при попадании на клетку с пищей – питаться, при виде пищи – прыгать, если много энергии или «нечего делать» – делиться (рис. 7.6).

	Еда тут	Еда впер.	Еда слев	Еда прав	Аг. впер.	Аг. слев.	Аг. прав.	Мало эне	Много эн.	Эн. увел.	Постоянн
Питание	10	0	0	0	0	0	0	0	-5	0	0
Движение	0	2	0	0	0	0	0	0	0	0	0
Повор. нал	0	0	0	0	0	0	0	0	0	0	0
Повор. нап	0	0	0	0	0	0	0	0	0	0	0
Отдых	0	0	0	0	0	0	0	0	0	0	0
Размножен	0	0	0	0	0	0	0	0	10	0	2
Нападение	0	0	0	0	0	0	0	0	0	0	0
Защита	0	0	0	0	0	0	0	0	0	0	0

Рис. 7.6. Врожденная матрица синаптических весов агентов при инициализации виртуального мира

Совершение всех действий, кроме питания и удачно проведенного нападения или защиты, сопровождается уменьшением энергии. Возможность совершения действия определяется достаточным для этого запасом энергии у агента.

Коэффициенты получения и затрат энергии указываются при инициализации мира: $k_{y_1} = 100$, $k_{y_2} = 15$, $k_{y_3} = k_{y4} = 10$, $k_{y_5} = 5$, $k_{y_6} = 75$, $k_{y_7} = 10$, $k_{y_8} = 8$.

При делении родитель передает потомку половину своего энергетического ресурса. Потомок остается в той же ячейке, где находится родитель. Гены родителя подвергаются случайной мутации. Амплитуда варьирования мутационных изменений $p_m = 0{,}02$ задается при инициализации мира.

Благодаря введению таких параметров мира, как репродуктивный и предельный возраст агента, удалось добиться более эффективной эволюции многоагентной системы.

Делиться разрешено только тем агентам, которые сами смогли «выжить» в мире 20 временных тактов. Агент погибает при нулевом энергетическом ресурсе, при достижении предельного возраста (80 тактов) и при случайной замене «новорожденным» агентом в случае перенаселения.

В процессе выживания делятся агенты, наиболее приспособленные к особенностям функционирования виртуального мира. Обучение популяции базируется на эволюционном подходе. В простейшем варианте моделирования агенты не эволюционируют в течение жизни, получая более или менее удачный набор генов (весов) при делении и случайной мутации генов родителя.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ: как разработать приложение для исследования?

В приведенной методике моделирования многоагентной системы каждый ее элемент является интеллектуальным агентом и обладает свойствами автономности, социального поведения, реактивности и активности, имеет базовые знания, цели, желания и намерения.

Методика допускает моделирование в одном виртуальном мире разных популяций агентов. Например, с разными комбинациями сенсорных сигналов и вариантов действий. Для сравнения качества жизни популяций используется объективный количественный критерий - численность популяции.

Результаты моделирования популяций примитивных, миролюбивых и агрессивных агентов описаны в [9], методика построения обучающейся многоагентной системы описана в [72].

При запуске приложения для исследования многоагентной системы, описанной в предыдущем пункте, появляется окно, показанное на рис. 7.7. В данном окне задаются характеристики виртуального мира, значение которых «по умолчанию» показано на рис. 7.7.

Приложение позволяет реализовать процессы эволюции трех популяций агентов. «Примитивные» агенты получают информацию только о наличии еды в данной клетке и впере-

ди и имеют небольшое количество возможных действий: они могут питаться, двигаться, поворачиваться и делиться.

Развитые «миролюбивые» агенты получают информацию о наличии еды в своем поле зрения и о своем энергетическом ресурсе. Данная популяция агентов относится к классу агентов с мотивациями. При совершении действия учитывают не только внешние, но и внутренние стимулы. Могут выполнять действия питания, движения, повороты, отдых и деление. Действия нападения и защиты у них заблокированы.

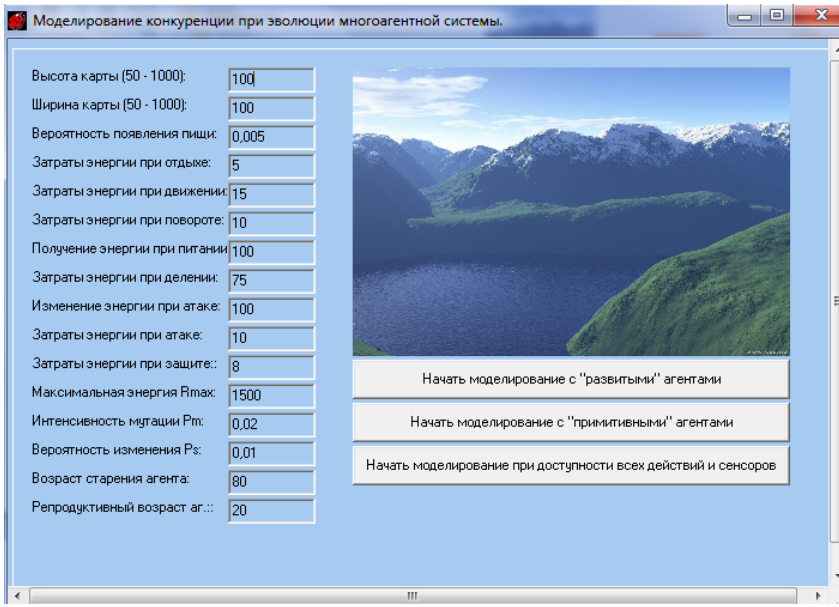


Рис. 7.7. Настройки виртуального мира

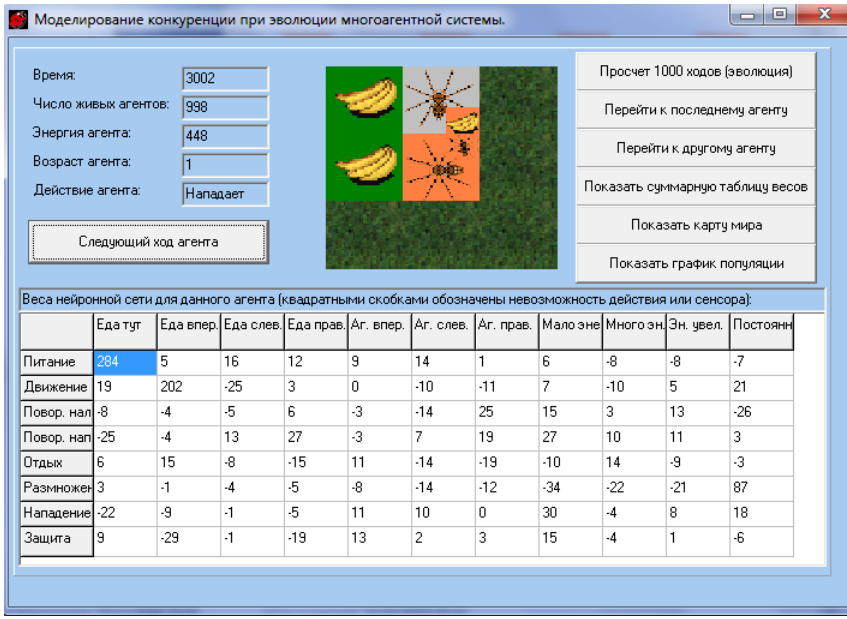
Развитые «агрессивные» агенты с мотивациями получают информацию о наличии еды и других агентов в своем поле зрения, а также о своем энергетическом ресурсе. Кроме таких действий, как питаться, делиться, отдыхать, поворачиваться, двигаться, они умеют нападать и защищаться. Таким образом, развитые агрессивные агенты имеют весь спектр имеющихся сенсоров и эффекторов.

Для примера в окне на рис. 7.7 была выбрана популяция «агрессивных» агентов нажатием кнопки «Начать моделирование при доступности всех действий и сенсоров».

При выборе вида популяции, появляется окно, показанное на рис. 7.8, в котором можно просчитать развитие эволюции «виртуального» мира через 1000 временных тактов. Для этой цели используется кнопка «Просчет 1000 ходов (эволюция)».

При этом в окне можно вести наблюдение за случайно выбранным агентом, информация о котором отображается выше кнопки «Следующий ход агента» (рис. 7.8).

Выбором кнопок «Перейти к другому агенту» и «Перейти к последнему агенту» возможна смена агента, за которым ведется наблюдение.



	Еда тут	Еда впер.	Еда слев.	Еда прав.	Аг. впер.	Аг. слев.	Аг. прав.	Мало эне	Много эн.	Эн. увел.	Постоянн
Питание	284	5	16	12	9	14	1	6	-8	-8	-7
Движение	19	202	-25	3	0	-10	-11	7	-10	5	21
Повор. нал	-8	-4	-5	6	-3	-14	25	15	3	13	-26
Повор. нап	-25	-4	13	27	-3	7	19	27	10	11	3
Отдых	6	15	-8	-15	11	-14	-19	-10	14	-9	-3
Размножен	3	-1	-4	-5	-8	-14	-12	-34	-22	-21	87
Нападение	-22	-9	-1	-5	11	10	0	30	-4	8	18
Защита	9	-29	-1	-19	13	2	3	15	-4	1	-6

Рис. 7.8. Окно наблюдения за агентами

В нижней части формы (рис. 7.8) отображаются весовые коэффициенты искусственной нейронной сети, управляющей

действиями агента, за которым ведется наблюдение. Сравнивая приобретенные в процессе выживания и наследования весовые коэффициенты с врожденной матрицей для агентов начальной популяции (рис. 7.6), можно заметить, что весовые коэффициенты меняются и сеть эволюционирует, приспосабливаясь к условиям виртуального мира.

При нажатии на кнопку «Показать суммарную таблицу весов» в нижнюю часть формы выводится усредненный по всей популяции геном агента.

Понаблюдать за картой всего виртуального мира можно с помощью кнопки «Показать карту мира» в окне на рис. 7.8. Появится клетчатое поле, показанное на рис. 7.9, на котором отображаются агенты и пища.

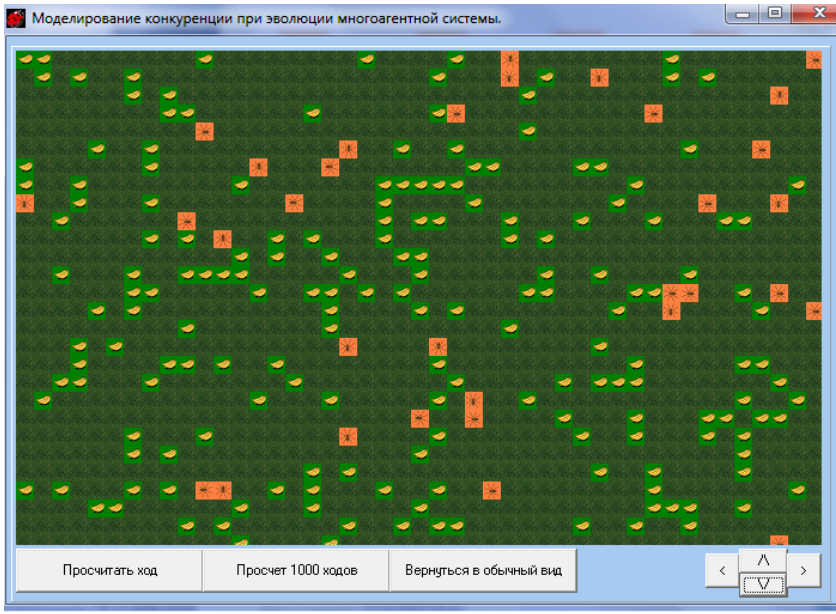


Рис. 7.9. Карта двумерного виртуального мира

На карте можно увидеть изменения через один временной такт и через 100 временных тактов. Для этого имеются кнопки «Просчитать ход» и «Просчет 100 ходов». По нажа-

тию кнопки «Вернуться в обычный вид» происходит возврат к основному окну приложения, показанному на рис. 7.8.

Качество жизни популяций оценивалось по численности агентов. График зависимости количества агентов от момента времени можно вывести в окно приложения, выбрав в основном окне кнопку «Показать график популяции». Появится окно, показанное на рис. 7.10.

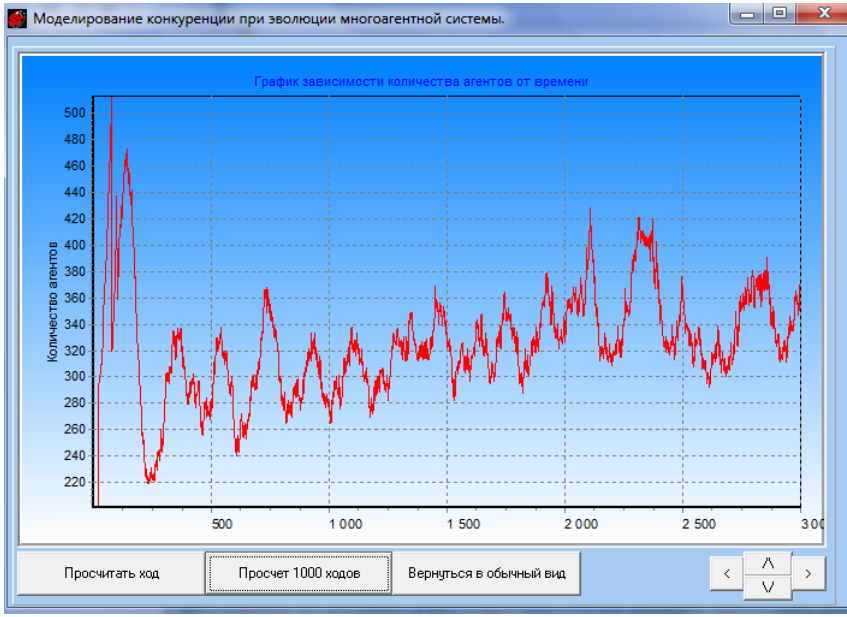


Рис. 7.10. График численности популяции в зависимости от времени

Можно наблюдать за изменениями графика через один временной такт и через 100 временных тактов, используя кнопки «Просчитать ход» и «Просчет 100 ходов».

АНАЛИЗ РЕЗУЛЬТАТОВ ЭВОЛЮЦИИ: чему научились агенты в процессе выживания?

В данном пункте приведены результаты исследования эволюции многоагентной системы и динамики численности

популяций при разных настройках рецепторов и эффекторов агентов.

На рис.7.11 приведен усредненный геном примитивных агентов после 2000 шагов эволюции.

	Еда тут	Еда впер.	Постоянн
Питание	15	-1	8
Движение	-5	4	8
Повор. нал	0	-4	1
Повор. нап	-6	-4	-2
Размножен	6	-4	14

Рис.7.11. Эволюционная матрица синаптических весов для агентов с сенсорами x_1, x_2, x_{11} *и действиями* y_1, y_2, y_3, y_4, y_6

Наибольшие синаптические веса (*побудительные мотивы в принятии решения*) соответствуют действиям питания и деления. В результате естественного отбора сформировался «рефлекс» к делению при нахождении в клетке с пищей. В этом случае также «нецелесообразно» прыгать и отворачиваться от еды, о чем свидетельствует приобретение отрицательных весовых коэффициентов.

При сравнении врожденных и приобретенных геномов (рис. 7.6 и 7.11) можно заметить, что в результате эволюции формируется целенаправленное поведение агентов. На уровне популяции доминирует цель – выживание вида, которая на уровне агента преобразуется в две подцели – получение энергии и деление [98, 99, 100].

На рис. 7.12 показан усредненный геном развитых «миролюбивых» агентов после 3000 шагов эволюции. Можно увидеть приобретение «рефлексов» поворота налево и направо при наличии там еды, желания «есть» и нежелания «отдыхать», когда мало энергии. Сигнал «радости» инициирует действия деления и нападения. Когда энергии достаточно и «вокруг ничего не видно» возникает желание двигаться.

	Еда тут	Еда впер.	Еда слев.	Еда прав.	Мало эне	Много эн.	Эн. увел.	Постоянн
Питание	17	10	0	1	12	-11	-2	4
Движение	-10	8	-3	-5	1	-3	-7	11
Повор. нал	0	-7	16	-4	-4	10	-11	2
Повор. нап	-1	-10	2	13	0	-1	-10	0
Отдых	-1	-16	-1	-1	-14	-2	-5	-8
Размножен	6	3	6	5	8	22	4	16
[Нападение	-9	-2	-9	5	-3	-2	3	6
[Защита]	-10	6	-2	-8	-4	-1	-7	-7

Рис.7.12. Эволюционная матрица синаптических весов для агентов с сенсорами x_1, x_2, x_3, x_4, x_8, x_9, x_{10}, x_{11} и действиями $y_1,...,y_6$

В результате моделирования эволюции популяции агентов со всеми доступными сенсорами и действиями наибольший вес формируется для действия нападать при наличии агента впереди.

	Еда тут	Еда впер.	Еда слев.	Еда прав.	Аг. впер.	Аг. слев.	Аг. прав.	Мало эне	Много эн.	Эн. увел.	Постоянн
Питание	12	11	1	0	2	0	-2	9	-11	5	4
Движение	-15	15	-5	-10	-3	-4	6	9	-5	7	8
Повор. нал	-4	-10	16	-1	-6	-2	0	7	-6	0	-5
Повор. нап	1	-9	-5	16	8	-4	-2	5	-7	5	-8
Отдых	3	-2	-4	-4	8	-2	-2	-15	-6	7	-7
Размножен	7	7	-4	-4	-2	6	9	-4	10	11	15
Нападение	-1	-4	1	0	24	7	-6	-5	8	-7	6
Защита	0	4	-12	7	0	4	1	3	5	-3	4

Рис.7.13. Эволюционная матрица синаптических весов для агентов с сенсорами $x_1,...,x_{11}$ и действиями $y_1,...,y_8$

Сигнал радости инициировал желание делиться и нежелание нападать. При малости энергии формируется желание действовать – питаться, двигаться, поворачиваться, защищаться. Действия отдыхать, делиться и нападать гасятся при

недостатке энергии отрицательными весовыми коэффициентами.

Уже через 2000 временных тактов отмечаются поведенческие тенденции, приобретенные в ходе эволюционного отбора.

1. Если агенты видят пищу слева, то они стремятся повернуть налево. Если агенты видят пищу справа – то поворачивают направо.
2. Если у агента мало энергии, то он, стремясь выжить, будет начинать движение вперед или нападение на других агентов. Напротив, если энергии у агента много, то он будет защищаться, отдыхать или делиться.
3. Если агенты стоят на клетке с пищей или рядом с пищей, то они начинают делиться. Такое поведение объясняется тем, что деление резко уменьшает энергию агента и ему необходимо пополнить ее в ближайшее время, чтобы выжить самому.
4. Если агент видит агентов справа или слева от себя, то он имеет тенденцию обороняться. Если другой агент находится впереди, то текущий агент предпочитает нападение.

Важно отметить, что все эти разумные поведенческие стратегии агентов не были запрограммированы, а развились непосредственно в процессе эволюционного отбора.

Выработанные глобальные стратегии обусловлены врожденными и сформированными рефлексами. Однако заметим, что индивидуальные поведенческие стратегии агентов отличались друг от друга.

При наблюдении за отдельными агентами можно было зафиксировать, что при виде другого агента они предпочитали двигаться - убегать, или отворачиваться от него. Когда два подобных миролюбивых агента видели друг друга, каждый предпочитал развернуться таким образом, чтобы выпасть из поля зрения другого агента.

С построенной компьютерной моделью была проведена серия экспериментов по сравнению действий, выполняемых агентами разных популяций.

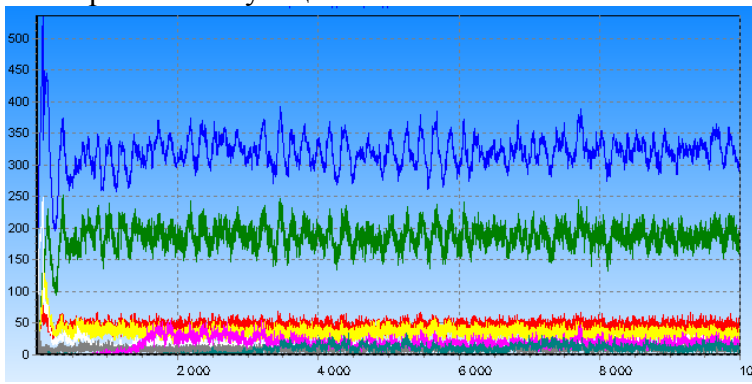


Рис. 7.14. Диаграмма действий для «миролюбивых» агентов

На диаграмме (рис. 7.14) видно, что развитые «миролюбивые» агенты в основном находятся в движении (средняя линия). Общая численность агентов колеблется в пределах от 200 до 400 особей (верхняя линия). Остальные действия примерно равновероятны.

«Агрессивные» агенты также всю жизнь в основном проводят в движении (рис. 7.14). Общая численность агентов колеблется в пределах от 200 до 500 особей.

Первоначально «агрессивные» агенты не умеют бороться, но с течением времени они начинают более активно использовать действия защиты и нападения (нижний всплеск на рис.7.15).

Отметим, что «агрессивные» агенты создают более многочисленную популяцию, что свидетельствует о том, что они более приспособлены к жизни в данном виртуальном мире.

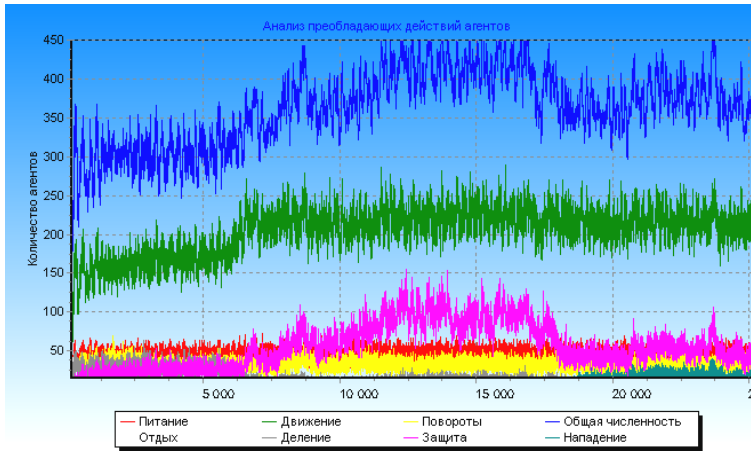


Рис. 7.15. Диаграмма действий для «агрессивных» агентов

График зависимости среднего энергетического ресурса агентов, выполняющих действие «деление», от времени, для «миролюбивой» популяции агентов приведен на рис. 7.16.

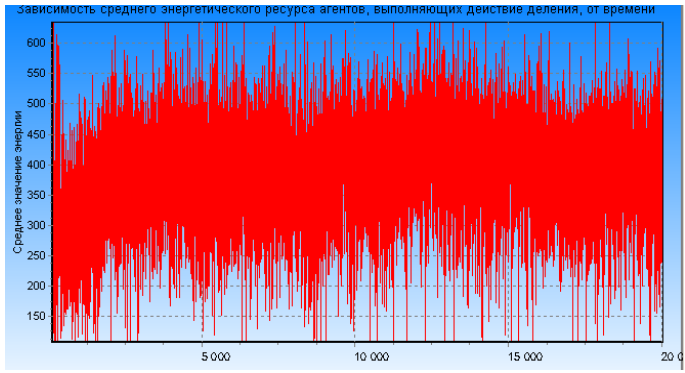


Рис. 7.16. График зависимости среднего энергетического ресурса делящихся «миролюбивых» агентов

График зависимости среднего энергетического ресурса делящихся «примитивных» агентов от времени приведен на рис. 7.17.

Сравнивая графики на рис. 7.16 и 7.17, заметим, что развитые агенты начинают деление в среднем при значении

энергии в 400 единиц, а примитивные – в 200 единиц. Таким образом, примитивные агенты более склонны к делению, а развитые агенты предпочитают сначала накопить необходимую энергию.

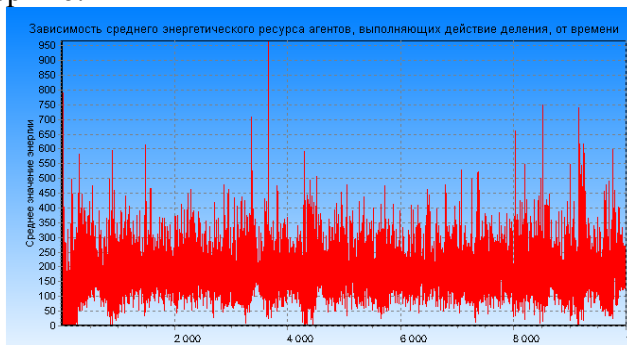


Рис. 7.17. График зависимости среднего энергетического ресурса делящихся «примитивных» агентов

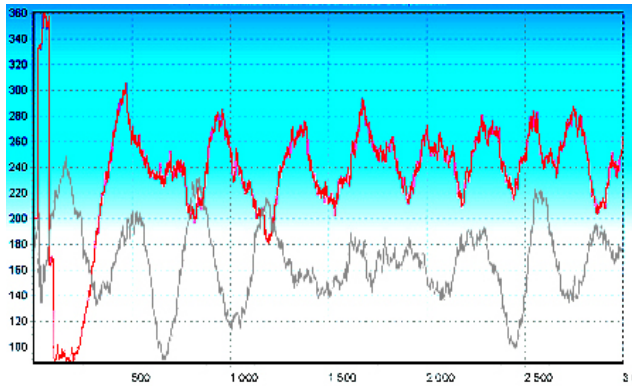


Рис.7.18. Графики зависимости численности популяций «агрессивных» и «примитивных» агентов от времени

Графики зависимости численности популяций «агрессивных» и «примитивных» агентов от времени показаны на рис. 7.18.

Светлый график соответствует популяции примитивных агентов, темный – популяции развитых агентов, конкурирующих друг с другом.

Примитивная многоагентная система выживает при более неблагоприятных настройках виртуального мира по сравнению с конкурирующей популяцией развитых агентов. Однако при выживании конкурирующая развитая многоагентная система обеспечивает более устойчивое и качественное развитие, чем примитивная.

Анализ приобретенных геномов позволяет сделать вывод о том, что в процессе эволюции формируется целенаправленное поведение агентов. При этом особи популяции в ходе эволюционного отбора обучается конкретным методикам достижения целей.

В рамках построенной модели возникает парадоксальное поведение агентов: они не только обучаются действовать в соответствии с ситуациями во внешней среде, но и находят стратегию поведения – часто агенты ничего не делают, выжидая удобного момента, когда ситуация во внешней среде изменится и нужно будет совершить действие, приводящее к полезному результату.

Сравнительный анализ популяций развитых агентов с мотивациями, умеющими анализировать состояние собственного энергетического ресурса, с популяцией примитивных агентов, у которых эти входы искусственно "подавлены" (отключены) показал, что популяция с мотивациями имеет значительные эволюционные преимущества по сравнению с популяцией без мотиваций.

Таким образом, мотивации играют важную роль в формировании адаптивного поведения [5, 6].

ЗАКЛЮЧЕНИЕ

В монографии приведены теоретические основы моделирования однослойных и многослойных искусственных нейронных сетей прямого распространения, сетей с обратными связями, стохастических, самоорганизующихся и эволюционирующих нейронных сетей. Описаны алгоритмы обучения искусственных нейронных сетей различной архитектуры. Приведены методики комбинирования градиентных и стохастических алгоритмов обучения для повышения эффективности решения практических задач.

Описаны оригинальные методики визуализации внутреннего состояния обученной нейронной сети и решения задач распознавания образов, классификации, категоризации, прогнозирования, восстановления зашумленной информации. Рассмотрено моделирование многоагентных систем на основе эволюционирующих нейронных сетей.

В монографии обсуждаются основные этапы компьютерной реализации нейросетевых алгоритмов: инициализация сети, обучение сети, работа сети в режиме функционирования, оценка эффективности. Даны методологические основы проектирования нейросетевых модулей решения задач. Приведены описания структур, интерфейсов и компьютерные коды основных блоков компьютерных приложений.

Описаны методики решения задач прогнозирования курсов валют, автоматического распознавания классов авторегрессионных облаков у больных мерцательной аритмией и степени активности автономной нервной системы в биоуправляемом игровом тренинге, категоризации объектов жилой недвижимости и др. Рассмотрены методы и способы оценки эффективности нейросетевых моделей.

Практической значимостью обладают результаты, в которых содержатся: методики проектирования автономных и интегрированных нейросетевых модулей решения практиче-

ских задач; механизмы разработки нейросетевых приложений в компьютерных кодах; модели многоагентных систем на основе эволюционирующих искусственных нейронных сетей; алгоритмы решения задач медицинской диагностики; методы нейросетевого поиска центров информационных кластеров в пространствах входной информации; способы восстановления утраченных данных по эталонным образам, хранящимся в памяти искусственной нейронной сети; методики оценки эффективности нейросетевых моделей.

В монографию вошли результаты работы над дисциплинами: “Компьютерные технологии в науке и образовании”, “Нейросетевое моделирование”, “Моделирование искусственных нейронных сетей в Delphi”, “Математическое моделирование в научных исследованиях”, которые читались автором студентам старших курсов математического факультета и факультета компьютерных наук в Воронежском государственном университете, а также аспирантам института машиностроения и аэрокосмической техники в Воронежском государственном техническом университете.

Часть исследований выполнялась совместно с ГОУ ВПО «БелГУ» в соответствии с проектом: № 2.2.3.3/4307: «Разработка структур трехуровневых биотехнических систем, предназначенных для виртуального игрового тренинга, включающих видимое фоновое и фиксирующее изображения, а также субсенсорные дискретные световые сигналы» аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы (2009-2010 годы)».

Материал монографии может быть полезен при разработке отечественных компьютерных приложений в свободно распространяемых средах программирования на основе нейросетевых алгоритмов при исследовании математических моделей слабо структурированных и плохо формализуемых процессов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Аналитические технологии для прогнозирования и анализа данных. [Электронный ресурс]. - URL: http://www.neuroproject.ru (дата обращения: 11.02.2015).

2. Барский А.Б. Введение в нейронные сети (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies/courses/607/463/info (дата обращения: 19.05.2014).

3. Барский А.Б. Логические нейронные сети (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies/courses/1061/185/info (дата обращения: 20.05.2014).

4. Бурцев М.С. Исследование новых типов самоорганизации и возникновения поведенческих стратегий : диссер. на соискание степени к.ф.-м.н. – М. : ИПМ РАН, 2005.

5. Бурцев М.С. Модель эволюционного возникновения целенаправленного адаптивного поведения. Исследование развития иерархии целей // Препринт ИПМ РАН,2002, N. 69.

6. Бурцев М.С., Гусарев Р.В., Редько В.Г. Исследование механизмов целенаправленного адаптивного управления // Изв. РАН. Теория и системы управления. 2002. No 6. С. 55-62.

7. Васильев В.В. Визуализация внутреннего состояния обученной искусственной нейронной сети / В.В. Васильев, Л.В. Хливненко // Информатика: проблемы, методология, технологии: материалы 9 межд. науч.-метод. конф., 12-13 февраля 2009 г.– Воронеж: ИПЦ ВГУ, 2009. – Т.1. – С. 142-144.

8. Васильев В.В. Исследование единственности результатов обучения нейронной сети классификации авторегрессионных облаков при мерцательной аритмии/ В.В. Васильев, Л.В. Хливненко // Математические модели и операторные уравнения.– Воронеж: ВГУ, 2007. – Т. 4. – С.31-37.

9. Васильев В.В. Моделирование эволюции многоагентной системы / В.В. Васильев, Л.В. Хливненко, С.Н. Демиденкова // Информатика: проблемы, методология, технологии: материалы X межд. науч.-метод. конф., 11-12 февраля 2010 г. – Воронеж: ИПЦ ВГУ, 2010. – Т.1. – С. 148-152.

10. Васильев В.В. Нейросетевое решение задачи локализации автомобильного номера / В.В. Васильев, Л.В. Хливненко, Л.Л. Погодина // Информатика: проблемы, методология, технологии: материалы 8 межд. науч.-метод. конф., 7-8 февраля 2008 г. – Воронеж: ВГУ, 2008. – Т1. – С.93-95.

11. Васильев В.В. Распознавание автомобильных номеров с помощью нейросетевого моделирования / В.В. Васильев, Л.В. Хливненко, Л.Л. Погодина // Математические модели и операторные уравнения: сборник науч. тр. - Воронеж: ВГУ, 2008. – Т. 5, Ч. 2. – С. 47-54.

12. Вороновский Г.К. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновский, К.В. Махотило, С.Н. Петрашев, С.А. Сргеев. – Х: ОСНОВА , 1997.

13. Воронцов К.В. Машинное обучение (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies /courses/13844/1241/info (дата обращения: 16.06.2015).

14. Горбань А.Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей / Сибирский журнал вычислительной математики, №1, Т.1, 1998. – С.12-24.

15. Дубровин В.И. Методы повышения эффективности процедур нейросетевой диагностики / В.И. Дубровин, С.А. Субботин // Нейрокомпьютеры: разработка, применение. 2002. - № 3.- С.3-9.

16. Дубровин В.И. Оценка значимости признаков на основе многослойных нейронных сетей в задачах диагностики и распознавания / В.И. Дубровин, С.А. Субботин // Информатика и системы управления, № 1(3), 2002. - С. 66-72.

17. Ежов А.А., Шумский С.А. Нейрокомпьютинг и его применения в экономике и бизнесе (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies/courses/2255/139/info (дата обращения: 23.01.2015).

18. Емельянов В.В. Теория и практика эволюционного моделирования / В.В. Емельянов, В.В. Курейчик, В.М. Курейчик. – М.: Физматлит, 2003.

19. Заенцев И.В. Нейронные сети. Основные модели :

учеб. пособие. – Воронеж : ИПЦ ВГУ , 1999.

20. Искусственные иммунные системы и их применение / под ред. Д. Дасгупты; пер. с англ. А.А. Романюхи, С.Г. Руднева под ред. А.А. Романюхи.— М. : Физматлит, 2006 .

21. Каширина И.Л. Искусственные нейронные сети : учеб. пособие. – Воронеж: ИПЦ ВГУ, 2005.

22. Колмогоров А. Н. О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного // Докл. АН СССР. 1957. Т. 114, № 5. С. 953–956.

23. Комашинский В. И., Смирнов Д. А. Нейронные сети и их применение в системах управления и связи. М.:Горячая линия–Телеком, 2003.

24. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. - М.: Горячая линия-Телеком, 2002.

25. Курейчик В.В. О правилах представления решений в эволюционных алгоритмах / В.В. Курейчик, С.И. Родзин // Известия ЮФУ. Технические науки, № 7, 2010. - С. 13-21.

26. Курейчик В.В. Теория эволюционных вычислений / В.В.Курейчик, В.М. Курейчик, С.И. Родзин. - М.: Физматлит, 2012.

27. Ларичев О . И. Теория и методы принятия решений / О. И. Ларичев. – М.: Логос, 2000.

28. Макаров И.М. Искусственный интеллект и интеллектуальные системы управления / И.М. Макаров, В.М. Лохин, С.В. Манько, М.П.Романов. -М.: Наука, 2006.

29. Макконен К.Ф. Игровой модуль с реализацией стратегии, направленной на избегание неудачи / К.Ф. Макконен, Ф.А. Пятакович, А.С. Новоченко // Фундаментальные исследования. – 2007. – №1. – С. 70–72.

30. Макконен К.Ф. Модели и алгоритмы биоуправления в информационной системе игрового автомобильного тренинга / К.Ф. Макконен, Ф.А. Пятакович // Системный анализ и управление в биомедицинских системах: журнал практической и теоретической биологии и медицины. – М., 2008. –Т. 7, № 1. – С. 177–181.

31. Макконен К.Ф. Разработка иерархической системы классификации режимов управления нейродинамической активностью мозга и ритмом сердца, основанной на информационном анализе для диагностического модуля сетевой интегрированной системы БОС-терапии // Прикладные задачи моделирования и оптимизации: межвуз. сб. науч. тр. - Воронеж: ВГТУ, 2008. - С. 75–79.

32. Мандрикова Ю.А. Системный анализ применительно к разработке автоматизированной системы выбора оптимальных методов терапии у больных с синдромом мерцательной аритмии / Ю.А. Мандрикова, Ф.А. Пятакович // Актуальные проблемы современной науки. Ч.10. Медицинские науки. Труды межд. конф. молодых ученых и студентов, 30 сентября – 2 октября 2002 г. – Самара, 2002. – С.37-39.

33. Миркес Е.М. Нейроинформатика : учеб. пособие. – Красноярск: КГТУ, 2002.

34. Мосалов О.П., Прохоров Д.В., Редько В.Г. Сравнение эволюции и обучения как методов адаптации агентов // Научная сессия МИФИ - 2006. VIII Всерос. науч.-техн. конф. «Нейроинформатика-2006»: сборник науч. тр. - М.: МИФИ, 2006. - Ч.1. - С. 201-208.

35. Назаров А.В. Нейросетевые алгоритмы прогнозирования и оптимизации систем / А.В. Назаров - СПб.: «Наука и техника», 2003.

36. Нейроинформатика / А.Н.Горбань, В.Л.Дунин-Барковский, А.Н.Кирдин и др. - Новосибирск: Наука. Сибирское предприятие РАН, 1998.

37. Омату С., Халид М., Юсоф Р. Нейроуправление и его приложения. Нейрокомпьютеры и их применение. - М.: Радиотехника, 2000.

38. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. — М.: Финанасы и статистика, 2002.

39. Пятакович Ф.А. Автоматическое прогнозирование восстановления номотопного ритма при синдроме фибрилляции предсердий / Ф.А. Пятакович, Ю.А. Мандрикова // Материалы 1-го Российского научного форума «МедКомТех 2003».Москва, ЦДХ, 25-28 февраля 2003. РАМН «Мораг Экспо». – М.: «Авиаиздат», 2003. – С. 79-80.

40. Пятакович Ф.А. Алгоритмы классификации степени активности автономной нервной системы на базе нейрокомпьютинга / Ф.А.Пятакович, Л.В. Хливненко, Т.И. Якунченко // Международный журнал фундаментальных и прикладных исследований, № 5, 2010. - С. 115-119.

41. Пятакович Ф.А. Информационный и условно-вероятностный анализ HRV / Ф.А. Пятакович // Научные ведомости БегГУ, №4 (13). – Белгород, 2000. – С. 82-88.

42. Пятакович Ф.А. Компьютерное прогнозирование исходов мерцательной аритмии / Ф.А. Пятакович // Сб.трудов «Ученые вузов Курска – народному хозяйству». – Курск, 1988. – С.129.

43. Пятакович Ф.А. Методологические аспекты авторегрессионного анализа при решении задачи прогнозирования исходов мерцательной аритмии / Ф.А.Пятакович, Ю.А. Мандрикова // Здоровье в XXI веке – 2000. Материалы докладов междунар. науч.-практ. конф. 25-28 сентября. – М. ; Тула, 2002. – С.183-185.

44. Пятакович Ф.А. Модели и алгоритмы нейросетевой дифференциации классов функционального состояния вегетативной нервной системы / Ф.А.Пятакович, Л.В. Хливненко, В.В. Васильев и др. // Системный анализ и управление в био-медицинских системах, Т.9. № 4, 2010. - С. 870 — 874.

45. Пятакович Ф.А. Особенности разработки биотехнических систем хронодиагностики и хронофизиотерапии / Ф.А.Пятакович, Т.И. Якунченко, Л.В. Хливненко и др. // Научные ведомости БелГУ № 4 (13) 2000. Серия Медицина. - Белгород, 2000. - С. 88-93.

46. Пятакович Ф.А. Прогнозирование и динамика восстановления синусового ритма у больных мерцательной аритмией методом авторегрессионного анализа / Ф.А. Пятакович, Г.С. Мезенцева // В кн. «Актуальные вопросы общей терапии и кардиологии». – Курск, 1984. – С.74-76.

47. Пятакович Ф.А. Разделение полимодального и амодального классов авторегрессионных облаков (АРО) при обучении на основе нечетких характеристик эксперта / Ф.А. Пятакович, В.В. Васильев, Л.В. Хливненко // Информатика как педагогическая задача. Материалы региональной конференции 14-15 февраля 2001 г. – Воронеж, 2001. – С.30-33.

48. Пятакович Ф.А. Разработка моделей и алгоритмов нейросетевой классификации степени активности автономной нервной системы и оценка их адекватности на обучающей и экзаменационной выборках / Ф.А. Пятакович, Т.И. Якунченко, Л.В. Хливненко и др. // Фундаментальные исследования, № 2, 2011. – М.: Академия Естествознания, 2011. - С. 136-141.

49. Пятакович Ф.А. Роль авторегрессионных, нечетких, нелинейных моделей и алгоритмов «нейрокомпьютинга» в разработке телемедицинской системы прогнозирования исходов мерцательной аритмии / Ф.А.Пятакович, К.Ф. Макконен, Л.В. Хливненко и др. // Научные ведомости БелГУ. Серия: Медицина. Фармация, Т12, № 22, 2010. - С. 149-156.

50. Пятакович Ф.А. Структура нейросетевого модуля для исследования математической модели автоматической классификации функциональных состояний человека/ Ф.А. Пятакович, Л.В. Хливненко, Т.И. Якунченко, В.В. Васильев // Современные проблемы науки и образования, № 2, 2015. [Электронный ресурс]. - URL: www.science-education.ru/129-21845

51. Пятакович Ф.А., Макконен К.Ф., Новоченко А.С. Патент №2349156. Биоуправляемый игровой тренажер и способ коррекции функционального состояния человека. Заявка №2007117796, приоритет 14 мая 2007 г. Зарегистрированный в государственном реестре Российской Федерации 20 марта 2009 г.

52. Пятакович Ф.А., Т.И. Якунченко. Иерархия режимов управления ритмом сердца на основе анализа энтропийной функции // Проблемы ритмов в естествознании. Материалы второго межд. симпозиума, 1–3 марта 2004 г. – М.: 2004. — С.341– 344.

53. Пятакович Ф.А. Нечеткий алгоритм в системе прогнозирования исходов мерцательной аритмии / Ф.А. Пятакович // Сборник материалов 2-й межд. конф. «Распознавание». – Курск, 1995. – С.159-161.

54. Редько В.Г. Модели адаптивного поведения и проблема происхождения интеллекта // Математическая биология и биоинформатика, 2007. – Т. 2, №1. – С.160-180.

55. Редько В.Г. Перспективы исследований на стыке информатики и биологии // Нейроинформатика, Т. 2, № 1, 2007.

56. Редько В.Г. Эволюционная кибернетика. – М.: Наука, 2001.

57. Редько В.Г. Эволюция, нейронные сети, интеллект. Модели и концепции эволюционной кибернетики, 5-е изд. — М. : КомКнига, 2007.

58. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М : Горячая Линия - Телеком, 2006.

59. Сидоренко Г.И. Анализ сердечного ритма и его нарушений с помощью попарного распределения интервалов RR ЭКГ / Г.И. Сидоренко, Г.К. Афанасьев, Я.Г. Никитин //Здравоохранение Белоруссии, №12, 1974. - С.7-11.

60. Сороколетов П.В. Построение интеллектуальных систем поддержки принятия решений / П.В. Сороколетов // Известия ЮФУ. Технические науки, 2009. - № 4 (93). - С. 117-124.

61. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. - М.: Эдиториал УРСС, 2002.

62. Тарков М.С. Нейрокомпьютерные системы (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies/courses/61/61/info (дата обращения: 10.05.2015).

63. Телков А.Ю. Экспертные системы : учеб. пособие. –

Воронеж : ИПЦ ВГУ, 2007.

64. Федотов В.Х. Нейронные сети в MS Excel: метод. указ. - Чебоксары: Чувашский гос. ун-т, 2004.

65. Хайкин С. Нейронные сети: полный курс, 2-е изд. - М.: Издательский дом «Вильямс», 2006.

66. Хливненко Л.В. Автоматизация документооборота в органах опеки и попечительства/ Л.В. Хливненко, В.В. Васильев, А.Е. Васильев// Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, № 1, 2011. – Воронеж: ИПЦ ВГУ, 2011. – С. 100-104.

67. Хливненко Л.В. Автоматическая локализация ядрышек в клетках растений / Л.В. Хливненко, В.В. Васильев, В.Н. Калаев // Информатика: проблемы, методология, технологии: материалы XIV межд. науч.-метод. конф., 6-8 февраля 2014 г. : в 4 т./ Воронеж. гос. ун-т. – Воронеж: Издательский дом ВГУ, 2014. – Т.1, С. 547-549.

68. Хливненко Л.В. Алгоритмы хронопронозирования исходов мерцательной аритмии / Л.В. Хливненко // Измерительные информационные технологии и приборы в охране здоровья: межд. науч.-практ. конф. Метромед-99, 29 июня - 1 июля 1999 г. - С-Пб: С-Пб ГТУ, 1999. - С. 94-95.

69. Хливненко Л.В. Биотехническая система хронопрогнозирования исходов мерцательной аритмии / Л.В. Хливненко // Микроэлектроника и информатика-2000: VII всерос. межвуз. науч.-техн. конф. студентов и аспирантов "Микроэлектроника и информатика-2000", 17-18 апреля, 2000. - М., 2000. - С. 91.

70. Хливненко Л.В. Возможности решения медицинских диагностических задач с помощью проектирования обучающихся искусственных нейронных сетей / Л.В. Хливненко, В.В. Васильев, Ф.А. Пятакович // Успехи современного естествознания, № 12, 2010. - С. 75-79.

71. Хливненко Л.В. Геометрическое распознавание авторегрессионных облаков в биотехнической системе хроно-

прогнозирования мерцательной аритмии / Л.В. Хливненко // Оптико-электронные приборы и устройства в системе распознавания образов, обработки символьной информации: IV межд. конф. “Распознавание-99”, 20-22 октября, 1999. - Курск, 1999. - С. 167-169.

72. Хливненко Л.В. Методика построения обучающейся многоагентной системы/ Л.В. Хливненко, В.В. Васильев // Научный вестник Воронежского государственного архитектурно-строительного университета. Серия: Информационные технологии в строительных, социальных и экономических системах, № 1, 2015. – Воронеж: ООП ВГАСУ, 2015. – С. 31-35.

73. Хливненко Л.В. Модели и алгоритмы хронопронозирования исходов мерцательной аритмии / Л.В. Хливненко // Диссертация канд.техн.наук. – Белгород, 2000. – 112 с.

74. Хливненко Л.В. Нейросетевое оценивание инвестиционной привлекательности сделок по продаже жилья / Л.В. Хливненко, В.В. Васильев, А.А. Сафонова// Информатика: проблемы, методология, технологии: материалы XI межд. науч.-метод. конф., 10-11 февраля 2011 г. – Воронеж: ИПЦ ВГУ, 2011. – Т.2, С. 422-426.

75. Хливненко Л.В. Нейросетевое решение задачи классификации степени активности автономной нервной системы/ Л.В. Хливненко, В.В. Васильев, Ф.А. Пятакович // Materialy VII Miedzynarodowej naukowi-praktycznej konferencji „Dynamika naukowych badan - 2011”, Volume 18. Matematyka. Nowoszesne informacyine technologie. – Przemysl, Nauka I Studia. - P. 12-16.

76. Хливненко Л.В. Применение стохастической искусственной нейронной сети к задаче медицинской диагностики / Л.В. Хливненко, В.В. Васильев // Информатика: проблемы, методология, технологии: материалы XII межд. науч.-метод. конф., 9-10 февраля 2012 г. – Воронеж: ИПЦ ВГУ, 2012. – Т.1, С. 428-430.

77. Хливненко Л.В. Прогнозирование исходов мерцательной аритмии с помощью искусственной нейронной сети / Л.В. Хливненко // Информатика: проблемы, методология, технологии: материалы 7-й междунар. науч.-метод. конф., 8-9 февр. 2007 г. – Воронеж, 2007. – С. 467- 471.

78. Хливненко Л.В. Разработка интеллектуальной информационной системы для категоризации детей, оставшихся без попечения родителей/ Л.В. Хливненко, В.В. Васильев, А.Е. Васильев// Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, № 2, 2011. – Воронеж: ИПЦ ВГУ, 2011. – С. 170-174.

79. Хливненко Л.В. Структура биотехнической системы хронопрогнозирования мерцательной аритмии / Л.В. Хливненко // Медико-экологические информационные технологии-99: II межд. науч.-техн. конф. "Медико-экологические информационные технологии-99", 19-21 мая, 1999. - Курск, 1999. - с. 51-52.

80. Хливненко Л.В. Уточнение зоны локализации ядрышек в клетках растений / Л.В. Хливненко, В.В. Васильев, В.Н. Калаев // Бъдещите изследвания – 2014 : материалы X межд. науч.-практ. конф., 17-25 февраля 2014 г. – София: «Бял ГРАД-БГ » ООД, 2014. – Т.43. Математика, С. 94-96.

81. Царегородцев В.Г. Нейросетевые анализ и моделирование современных связей климата и растительности // Современные подходы к интеграции информационных технологий : труды V Всерос. семинара "Информационные технологии в энергетике" (Иркутск, 2000). – Иркутск: ИСЭМ СО РАН, 2001. – С.157-165.

82. Царегородцев В.Г. Общая неэффективность использования суммарного градиента выборки при обучении нейронной сети // Нейроинформатика и её приложения : материалы XIII Всерос. семинара. – Красноярск, 2004. – С.145-151.

83. Царегородцев В.Г. Определение оптимального размера нейросети обратного распространения через сопоставление средних значений модулей весов синапсов // Материалы XIV межд. конф. по нейрокибернетике. – Ростов-на-Дону, 2005. – Т.2. - С.60-64.

84. Царегородцев В.Г. Редукция размеров нейросети не приводит к повышению обобщающих способностей // Нейроинформатика и её приложения : материалы XIII Всерос. семинара. – Красноярск, 2004. – С.163-165.

85. Чубукова И.А. Data Mining (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies/courses/6/6/info (дата обращения: 10.03.2015).

86. Ясницкий Л.Н. Введение в искусственный интеллект : учеб. пособие. – М: Издательский центр «Академия» , 2005.

87. Яхъяева Г.Э. Основы теории нейронных сетей (курс лекций) [Электронный ресурс]. - URL: http://www.intuit.ru/studies/courses/88/88/info (дата обращения: 18.01.2015).

88. Benne M., Grondin-Perez B., Chabriat J.-P., Herv'e P. Artificial neural networks for modelling and predictive control of an industrial evaporation process // Journal of food engineering. 2000. no. 46. P. 227–234.

89. Bottou L., LeCun Y. Large scale online learning // Advances in Neural Information Processing Systems 16 (2003). MIT Press. 2004. P. 217-224.

90. Bouchard M. New recursive-least-squares algorithms for non-linear active control of sound and vibration using neural networks // IEEE Trans. on Neural Networks. 2001. Vol. 12, no. 1. P. 135–147.

91. Caruana R.A., de Sa V.R. Benefitting from the variables that variable selection discards // Journal of Machine Learning Research. 2003. Vol.3. P.1245-1264.

92. Gorban A.N. Approximation of continuous functions of several variables by an arbitrary nonlinear continuous function of one variable, linear functions, and their superpositions / Appl. Math. Lett., 1998. Vol.11, №3. - pp.45-49.

93. Hoopen M. Probabalistic characterization of RR intervals/ M Hoopen, I.P.M. Bongaurts //«Cardiovasc.res.». – 1969. – v.3, №2. – P.218-226.

94. Hopfield J.J. Neural Networks and Physical systems with emergent collective computational abilities // Proc. Nat. Sci. USA. 1982. V.79. P. 2554-2558.

95. Lawrence S., Giles C.L. Overfitting and neural networks: conjugate gradient and backpropagation // Proc. Int. Joint Conf. Neural Networks (IJCNN'2000). Como. Italy. 2000. P.114-119.

96. Lin F.-J., Wai R.-J., Hong C.-M. Hybrid supervisory control using recurrent fuzzy neural network for tracking periodic inputs // IEEE Trans. on Neural Networks. 2001. Vol. 12, no. 1. P. 69–90.

97. Pyatakovich F.A. Biotechnical system of car game training based on use of a multiparametrical feedback and subsensitivity light signals of control/ F.A. Pyatakovich, T.I. Yakunchenko// European journal of natural history, № 6, 2009. - P. 38-40.

98. Red'ko V.G., Mosalov O.P., Prokhorov D.V. A model of Baldwin effect in populations of self-learning agents // International Joint Conference on Neural Networks, Montreal. 2005.

99. Red'ko V.G., Prokhorov D.V., Burtsev M.S. Theory of functional systems, adaptive critics and neural networks // International Joint Conference on Neural Networks, Budapest. 2004. P. 1787-1792.

100. Red'ko V.G., Mosalov O.P., Prokhorov D.V. A model of evolution and learning // Neural Networks, 2005. Vol. 18, No 5-6. P. 738-745.

101. Wilson D.R., Martinez T.R. The general inefficiency of batch training for gradient descent learning // Neural Networks. 2003. Vol.16. Issue 10. P.1429-1451.

102. Wilson D.R., Martinez T.R. The inefficiency of batch training for large training sets // Proc. Int. Joint Conf. Neural Networks (IJCNN'2000). Como. Italy. 2000.Vol.2. P.113-117.

ОГЛАВЛЕНИЕ

Научное издание

Хливненко Любовь Владимировна

ПРАКТИКА НЕЙРОСЕТЕВОГО МОДЕЛИРОВАНИЯ

В авторской редакции

Подписано в печать 30.11.2015.
Формат 60×84/16. Бумага для множительных аппаратов.
Усл. печ. л. 13,4. Уч.-изд. л. 10,0. Тираж 500 экз.
Зак. № 197.

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14