Федеральное государственное учебное предприятие
Политехнический институт Сибирского Федерального Университета

Кафедра Систем Искусственного Интеллекта

Жуков Л.А., Решетникова Н.В.

Учебное пособие по дисциплине «Приложения нейронных сетей»

для студентов укрупненной группы 260000 - Технология и производство продовольственных продуктов и потребительских товаров, 230000 – Вычислительная техника и информационные технологии

Направления 260500 Технология полиграфического и упаковочного производства; 230200 Информационные системы и технологии; 230100 Информатика и вычислительная техника

Красноярск 2007

УДК 004.8.032.26
ББК 32

Жуков Л.А., Решетникова Н.В. Приложения нейронных сетей: Учебное пособие для студентов, учащихся лицея и ЗПШНИ / Л. А. Жуков, Н. В. Решетникова. Красноярск: ИПЦ КГТУ, 2007. 154 с.

Описана абстрактная технология нейросетевой обработки данных для решения прикладных задач. Рассматривается технология на основе нейронных сетей с учителем. Построена формальная модель технологии. Используется иерархическая модель технологии до уровня реализаций операций пользователя, которые предполагаются реализованными в идеальном нейрокомпьютере. Данная работа может быть полезна для исследователей, использующих нейросетевые методы для решения прикладных задач; студентов, учащихся лицея и ЗПШНИ, аспирантов, преподавателей и специалистов. С дополнительной информацией можно ознакомиться на сайте http://zhukov.org.ru.

Оглавление

Введение

Актуальность и причины введения. Специалисты по вычислительной технике должны разбираться в основных принципах, особенностях, архитектурах средств вычислительной техники, в том числе новейших, таких как нейронные сети и нейрокомпьютерные системы. Предполагается, что этот курс будет служить только основой для знаний, получаемых в ходе самостоятельной работы, чтения новых книг и журналов. Необходимость введения курса «Приложения нейронных сетей» обусловлена тем, что он является одним из необходимых современных курсов при подготовке специалистов по программированию, информационным технологиям и обработке данных.

Курс тесно связан и опирается на такие ранее изученные дисциплины, как «Информатика», «Дискретная математика», «Алгоритмические языки и программирование», «Организация ЭВМ и систем», «Теория языков программирования и методы трансляции». В свою очередь, знания, полученные при изучении данного предмета, могут быть использованы при изучении предметов «Параллельные системы», «Вычислительная математика» и основой для последующей профессиональной деятельности.

Программа учебной дисциплины соответствует Государственному образовательному стандарту высшего профессионального образования.

Целью настоящего курса является ознакомление студентов с основными принципами организации программного и аппаратного обеспечения нейроЭВМ и систем. Основные задачи курса заключаются в изучении архитектуры основных типов современных нейроЭВМ, терминологию в данной предметной области, принципов построения и обучения нейрокомпьютеров, умении использовать нейронные сети для решения прикладных задач.

В структуре изучаемого курса выделяются следующие основные разделы и темы: Методы обучения нейронных сетей, Ассоциативные ИНС, Самоорганизующиеся сети или сети естественной классификации, Приложения ИНС для гуманитарных наук и лингвистики, Основы организации параллельной обработки информации, Архитектура и проектирование нейроимитаторов. Однако, большинство перечисленных разделов подробно рассматривается в многочисленных книгах и учебных пособиях, поэтому в данном пособие основное место отведено нейросетевой технологии обработки информации.

Отличительными чертами данного курса является обязательное использование ЭВМ и других средств вычислительной техники, как на аудиторных занятиях, так и при самостоятельной работе. Лабораторные занятия по курсу требуют обязательного использования вычислительной техники.

Программой курса предусмотрено чтение лекций, проведение семинарских и практических занятий, выполнение лабораторных и контрольных работ.

В ходе изучения данного курса студент слушает лекции, посещает практические занятия, занимается индивидуально. Освоение курса предполагает, помимо посещений лекций и семинарских занятий, выполнение домашних заданий. Особое место в овладении данным курсом отводится самостоятельной работе. В связи с особенностями курса, задание на самостоятельную работу не выдается по вариантам, а выбирается студентом из предварительного списка возможных заданий или предлагается новое, с обязательным утверждением преподавателем. Задания могут иметь разную направленность – более аппаратную, более программную, сочетание аппаратной и программной работ, также может включать в себя различные работы поискового и обзорного характера, в том числе по материалам Интернета, на английском языке, другие работы. В связи с большим различием в специфике работ студенты самостоятельно выбирают направление работ и тему, что требует, кроме всего прочего, наличия активного подхода к изучаемой профессии и знание желаемого студентом вида деятельности.

Курс завершается зачетом. Обязательным условием допуска студента к экзамену является выполнение и защита в течение семестра лабораторных работ. Кроме того, способствуют получению максимальной оценки участие в НИРС, выступление на научных семинарах и конференциях, получение сертификатов по темам или специальностям, близким к изучаемому предмету. Экзаменационные требования сводятся к следующему: студент должен самостоятельно ответить на несколько десятков вопросов с закрытыми вариантами ответов.

1. Цели и задачи дисциплины

Цель дисциплины - ознакомление студентов с новой перспективной областью информатики, основными принципами организации и работы нейрокомпьютеров, возможностями использования нейронных сетей для решения прикладных задач.

2. Требования к уровню освоения содержания дисциплины

В результате изучения дисциплины студенты должны:

1) знать базовые модели нейронов и нейронных сетей;

2) знать основные типы моделей нейрокомпьютерных систем и области их применения.

3) владеть основными способами решения прикладных задач распознавания образов, диагностики, управления с помощью нейронных сетей;

4) получить навыки разработки и реализации программных моделей нейрокомпьютерных систем;

5) иметь представление о современных достижениях в разработке и коммерческом использовании нейрокомпьютерных систем и нейрокомпьютеров;

6) иметь основные представления о структуре мозга и биологических нейронных сетях.

Типовые разделы и темы программы по нейронным сетям могут содержать следующее:

1. Введение. Принципы организации и функционирования ИНС.

Идея коннекционизма в истории науки. Основные направления в нейроинформатике. Биологические нейронные сети: нейроны, аксоны, дендриты, синапсы, доли мозга, полушария, мозжечок и его структура, предварительная обработка зрительной информации у лягушки и кошки. Очерк истории нейроинформатики. Основные определения для искусственных нейронных сетей (ИНС). Элементы ИНС. Общая характеристика принципов организации информационных процессов в ИНС. Задачи, решаемые в настоящее время с помощью нейронных сетей.

2. Явные методы формирования ИНС.

Явные дискриминантные правила в задаче распознавания; дискриминантные нейроны. Ассоциативная память и сети Кохонена.

3. Методы обучения ИНС.

Правило Хебба. Персептрон и его обучение. Принцип двойственности, быстрое дифференцирование и методы оптимизации в обучении сетей автоматов. Нейронные сети обратного распространения ошибки. Избыточность ИНС; методы оптимизации структуры сети.

4. Применение ИНС.

Нейросетевые экспертные системы (диагностика, предсказания, консилиумы нейронных экспертов). Примеры из медицины, экономики, финансового анализа и др.

Решение задач оптимизации с помощью ИНС (сети обратного распространения ошибки как универсальные оптимизирующие устройства, решение задач дискретной оптимизации с помощью сетей Хопфилда, оценка показателей чувствительности с помощью сетей Кохонена и сетей обратного распространения ошибки).

Явное формирование ИНС для решения некоторых задач вычислительной математики (линейная алгебра, математическая физика).

Специализированные ИНС. Основные определения и классификация. Сети пространственно-временной обработки информации. Стохастические и иерархические ИНС.

5. Надежные системы из ненадежных элементов.

Методы обучения для формирования устойчивых к флуктуациям параметров навыков ИНС (использование флуктуаций, возрастающих в ходе обучения, метод виртуальных сетей).

Выработка навыков, устойчивых к разрушениям части нейронной сети.

Проблема устойчивости старых навыков по отношению к выработке новых.

6. Архитектура и проектирование нейроимитаторов (Реализация ИНС). Проблемы реализации ИНС. Методы реализации. Программные модели ИНС. Программно-аппаратные модели и аппаратная реализация ИНС. Нейрокомпьютеры. Основные характеристики коммерческих нейрокомпьютеров.

7. Заключение. Перспективы развития и применения ИНС и нейрокомпьютеров.

Нейронные сети — это раздел искусственного интеллекта, в котором для обработки сигналов используются явления, аналогичные происходящим в нейронах живых существ. Важнейшая особенность нейронных сетей, свидетельствующая об их широких возможностях и огромном потенциале, состоит в параллельности обработки данных при аппаратной реализации. При большом количестве межнейронных связей это позволяет значительно ускорить процесс обработки информации. Во многих случаях становится возможным преобразование сигналов в реальном времени. Кроме того, при большом числе межнейронных соединений сеть приобретает устойчивость к ошибкам, возникающим на некоторых линиях. Функции поврежденных связей берут на себя исправные линии, в результате чего деятельность сети не претерпевает существенных возмущений.

Не менее важна способность нейронных сетей к обучению и обобщению накопленных знаний. Нейронная сеть обладает чертами искусственного интеллекта. Обученная на ограниченном множестве данных сеть, способна обобщать полученную информацию и показывать хорошие результаты на данных, не использовавшихся в процессе обучения.

Кроме того, архитектура нейронных сетей позволяет реализовать ее с применением технологий сверхвысокой степени интеграции. Различие элементов сети невелико, а их повторяемость огромна. Это открывает перспективу создания универсального процессора с однородной структурой, способного перерабатывать разнообразную информацию и не требующего обязательного наличия программы обработки, достаточна только постановка задачи.

Использование перечисленных свойств на фоне развития устройств со сверхвысокой степенью интеграции (VLSI) и повсеместного применения вычислительной техники, вызвало в последние годы огромный рост интереса к нейронным сетям и существенный прогресс в их исследовании. Создана база для выработки новых технологических решений, касающихся восприятия, искусственного распознавания и обобщения видеоинформации, управления сложными системами, обработки речевых сигналов и т.п. Искусственные нейронные сети в практических приложениях, как правило, используются в качестве подсистемы управления или выработки решений, передающей исполнительный сигнал другим подсистемам, имеющим иную методологическую основу.

Функции, выполняемые сетями, подразделяются на несколько групп: аппроксимация; классификация и распознавание образов; прогнозирование; идентификация и оценивание; ассоциативное управление.

Аппроксимирующая сеть играет роль универсального аппроксиматора функции нескольких переменных, который реализует нелинейную функцию вида $y = f(x)$, где x — входной вектор, а y — реализованная функция нескольких переменных. Множество задач моделирования, идентификации, обработки сигналов удается сформулировать в аппроксимационной постановке.

Для классификации и распознавания образов сеть накапливает в процессе обучения знания об основных свойствах этих образов, таких, как геометрическое отображение структуры образа, распределение главных компонентов (PCA), или о других характеристиках. При обобщении акцентируются отличия образов друг от друга, которые и составляют основу для выработки классификационных решений.

В области прогнозирования задача сети формулируется как предсказание будущего поведения системы по имеющейся последовательности ее предыдущих состояний. По информации о значениях переменной в моменты времени, предшествующие прогнозированию, сеть вырабатывает решение о том, чему должно быть равно оцениваемое значение исследуемой последовательности в текущий момент времени.

В задачах управления динамическими процессами нейронная сеть выполняет, как правило, несколько функций. Во-первых, она представляет собой нелинейную модель этого процесса и идентифицирует его основные параметры, необходимые для выработки соответствующего управляющего сигнала. Во-вторых, сеть выполняет функции следящей системы, отслеживает изменяющиеся условия окружающей среды и адаптируется к ним. Она также может играть роль нейрорегулятора, заменяющего собой традиционные устройства. Важное значение, особенно при управлении роботами, имеют классификация текущего состояния и выработка решений о дальнейшем развитии процесса.

В задачах ассоциации нейронная сеть выступает в роли *ассоциативного запоминающего устройства*. Здесь можно выделить память *автоассоциативного* типа, в которой взаимозависимости охватывают только конкретные компоненты входного вектора, и память *гетероассоциативного* типа, с помощью которой сеть определяет взаимосвязи различных векторов. Даже если на вход сети подается вектор, искаженный шумом, либо лишенный отдельных фрагментов данных, то сеть способна восстановить полный и очищенный от шумов исходный вектор путем генерации соответствующего ему выходного вектора.

Различные способы объединения нейронов между собой и организации их взаимодействия, привели к созданию сетей разных типов. Каждый тип сети, в свою очередь, тесно связан с соответствующим методом подбора весов межнейронных связей (т.е. обучения).

Интересным представляется объединение различных видов нейронных сетей между собой, особенно сетей с самоорганизацией и обучаемых с учителем. Такие комбинации получили название "гибридные сети". Первый компонент - это сеть с самоорганизацией на основе конкуренции, функционирующая на множестве входных сигналов и группирующая их в кластеры по признакам совпадения свойств. Она играет роль *предобработчика* (препроцессора) данных. Второй компонент - в виде сети, обучаемой с учителем (например, персептронной), сопоставляет входным сигналам, отнесенным к конкретным кластерам, соответствующие им заданные значения. Это можно соотнести с блоком *интерпретатор ответов* или *постпроцесор*. (Впрочем, это не избавляет от необходимости выполнять предобработку и интерпретацию ответов обычного типа в каждом из таких компонентов). Подобная сетевая структура позволяет разделить фазу обучения на две части: вначале обучается компонент с самоорганизацией, а потом — сеть с учителем. Дополнительное достоинство такого подхода заключается в снижении вычислительной сложности процесса обучения, а также в лучшей интерпретации получаемых результатов.

Литература для изучения нейронных сетей многочисленна и представлена в списке литературы в конце работы. Однако, бОльшая часть ее про изучение архитектуры и алгоритмов работы нейронных сетей. Развернутые тексты по технологическим аспектам использования нейронных сетей встречаются намного реже, можно сказать про работы Россиева Д.А. и других авторов. В работе Комарцовой сделана ссылка на методики и технологии в работах школы Галушкина А. И. Поэтому данная работа окажется полезной для большого числа пользователей.

Глава 1

Нейронные сети – удобный и достаточно простой инструмент для создания различных экспертных систем, решения задач классификации и извлечения знаний из данных и, кроме того, дают возможность высокопараллельных реализаций. Рассматривая нейронную сеть как набор элементов, производящих некоторые вычисления над приходящими к ним данными, можно изучать решение главной и вспомогательных задач. К главной относятся функции генерации, обучения и тестирования нейронных сетей. Вспомогательные задачи: определение значимости входных сигналов, контрастирование нейронных сетей, обучение примеров, определение минимального решающего набора входных параметров, получение логически прозрачной нейронной сети и, в конечном счете, знаний из данных. Для решения этих задач разработан и используется математический аппарат искусственных нейронных сетей. Архитектуры и алгоритмы позволяют обучать и тестировать нейронные сети, решать различные задачи обработки данных.

Применение технологии нейронных сетей для решения вычислительных задач позволяет достичь высокого уровня параллелизма и делает возможным для реализации алгоритмов использовать распределённые вычислительные комплексы, системы и вычислительные сети с различными топологиями.

Однако, поскольку большинство пользователей НКС пока не могут получить прямого доступа к параллельным системам, постольку в данной работе идет обсуждение интеллектуальных возможностей и преимуществ нейронных сетей. В большинстве обсуждаемых задач и их решений использованы искусственные нейронные сети прямого распространения, по зарубежным классификациям относящиеся к классу MLP (многослойный персептрон). Также нейронные сетей такого вида называют двойственные нейронные сети из-за часто используемых методов обучения нейронных сетей такого вида, сигмоидные нейронные сети – из-за преобладающего использования сигмоидной (сигмоидальной, сигмовидной) функции в качестве функции активации.

Нейронные сети представляют собой совокупность однотипных базовых элементов – нейронов, соединенных между собой линиями передачи информации или синаптическими связями с определенными коэффициентами веса и, благодаря такой структуре, обладающих очень высокой степенью внутреннего параллелизма. Выделяют группу связей, по которым сеть получает информацию из внешнего мира, и группу выходных связей, с которых снимаются выдаваемые сетью сигналы. Нейронные сети применяются для решения различных задач классификации и прогнозирования. Нейронная сеть обучается решению задачи на основании

некоторой обучающей выборки – "задачника", состоящего из набора пар "вход–требуемый выход", и далее способна решать примеры, не входящие в обучающую выборку.

Принципы обработки информации в современных системах обработки данных и функционирования человеческого мозга существенно различаются. Мозг функционирует на основе параллельной обработки информации в нейронных сетях. И хотя о работе человеческого мозга известно не так много, существует множество моделей, имитирующих внешние проявления этой работы. Системы, построенные на основе принципов параллельной обработки информации в распределенных нейронных сетях, называют *нейрокомпъютерами*.

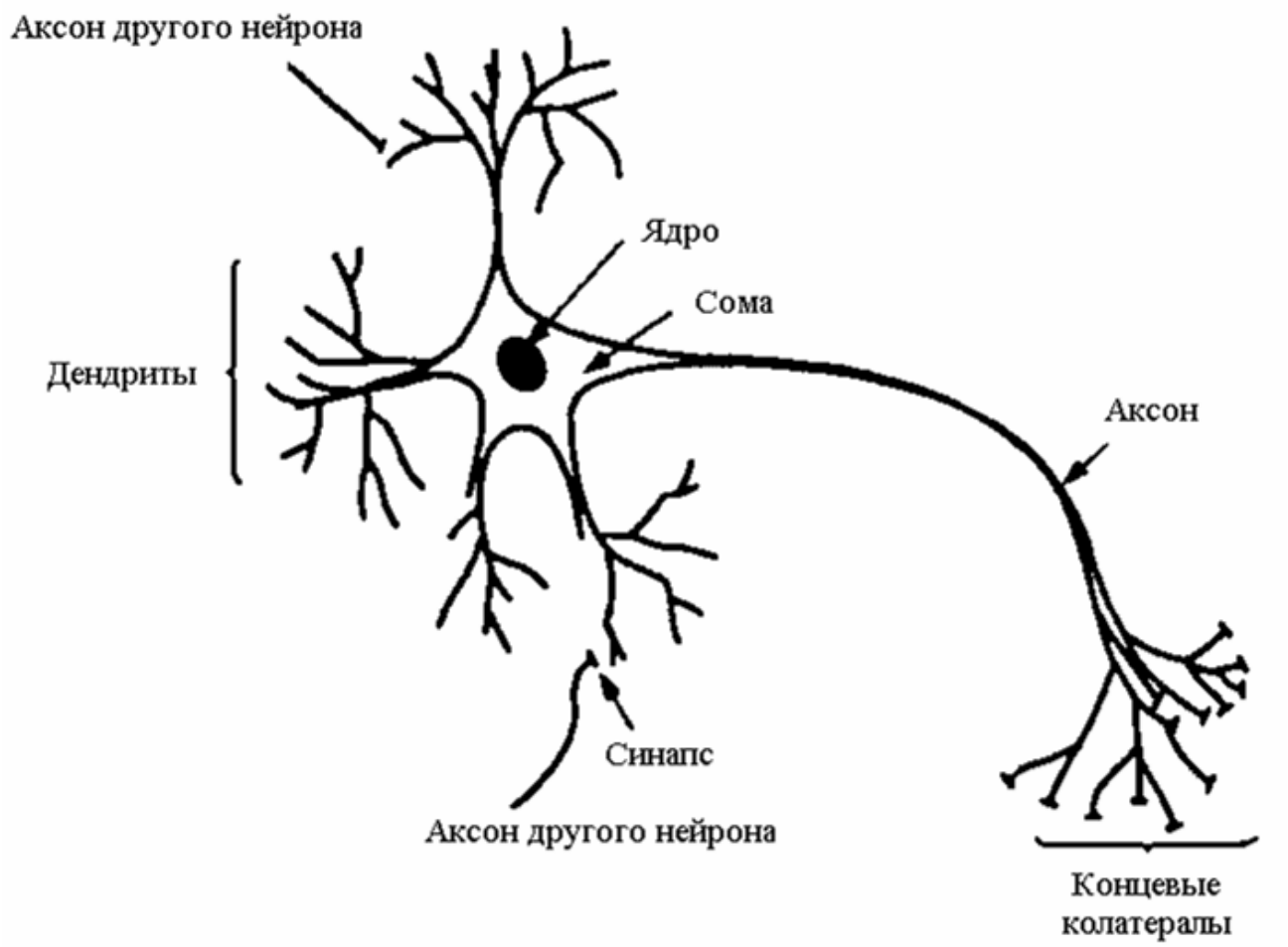
1.1 Биологические нейроны и нейронные сети

Биологические нейроны и нейронные сети служили прототипами при создании искусственных нейронных сетей. Была сделана попытка использования процессов, происходящих в нервных системах, для выработки новых технологических решений. В большинстве современных нейросетевых архитектур былое сходство стало существенно меньше. Тем не менее, знакомство с естественными нейронными сетями полезно, т.к. они успешно решают задачи, к выполнению которых лишь стремятся искусственные сети.

Нервная система человека построена из специальных нервных клеток, называемых нейронами. Мозг имеет около 10^{11} нейронов, которые образуют около 10^{15} передающих связей, имеющих длину до метра и более.

Биологический и искусственный нейроны.

Нервная клетка, сокращенно называемая *нейроном,* является основным элементом нервной системы. Изучение механизмов функционирования отдельных нейронов и их взаимодействия принципиально важно для познания протекающих в нервной системе процессов поиска, передачи и обработки информации. С этой точки зрения представляется необходимым изучить модель биологического нейрона.



Нейрон состоит из ядра и протоплазмы. У нейрона имеется тело со стандартным набором органелл, называемое *сомой,* внутри которого располагается ядро. От окружающей среды нейрон отделен тонкой мембраной. Белковые молекулы, встроенные в мембрану, выполняют функции рецепторов и ионных каналов. Нейроны имеют многочисленные отростки, по которым они получают информацию от рецепторов и других нейронов и передают сигналы нейронным клеткам исполнительных органов. Можно выделить два типа отростков: многочисленные тонкие, густо ветвящиеся древовидные отростки (*дендриты*) и более толстый, расщепляющийся на конце *аксон*. Через *дендриты* нейрон получает информацию через специальные контакты – *синапсы*. Передача информации от одного нейрона к другому осуществляется распространением нервного импульса по нервному волокну – *аксону*. Выходной сигнал отводится аксоном через его многочисленные нервные окончания, называемые *колатералами.* Колатералы контактируют с сомой и дендритами других нейронов, образуя очередные синапсы.

Синапсы отличаются друг от друга размерами и возможностями концентрации нейромедиатора вблизи своей оболочки. По этой причине импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут возбуждать ее в разной степени. Мерой

возбуждения клетки считается уровень поляризации ее мембраны, зависящий от суммарного количества нейромедиатора, выделенного на всех синапсах.

Длительность импульса составляет около 1 мс. При подходе нервного импульса к синаптическому контакту происходит химическая реакция, следствием чего является выброс из аксона биологически активного вещества, которое, диффундируя через синаптическую щель, достигает клетки, реализуя синаптическую передачу. Передача возбуждений и торможений между нейронами происходит через химические и электрические синапсы. Существует множество различных типов нейронов.

Отдельный нейрон не является элементарной единицей обработки информации, а выполняет функции нервного центра. Дендриты и аксоны могут вступать в связи с участками мембран других нейронов, образуя сети. Эти сети и служат системами обработки информации. Тело нейрона в данном случае лиц поддерживает функционирование этих участков мембраны, а сигналы, поступающие на вход по синаптическим связям, могут «записываться» на молекулу ДНК.

Из сказанного следует, что каждому входу клетки можно сопоставить численные коэффициенты (веса), пропорциональные количеству нейромедиатора, однократно выделяемого на соответствующем синапсе. В математической модели нейрона входные сигналы должны умножаться на эти коэффициенты для того, чтобы корректно учитывать влияние каждого сигнала на состояние нервной клетки. Синапсические веса должны быть натуральными числами, принимающими как положительные, так и отрицательные значения. В первом случае синапс оказывает возбуждающее, а во втором - тормозящее действие, препятствующее возбуждению клетки другими сигналами. Таким образом, действие возбуждающего синапса может моделироваться положительным значением синаптического веса, а действие тормозящего синапса - отрицательным значением.

В результате поступления входных импульсов на конкретные синапсы и высвобождения соответствующих количеств нейромедиатора происходит определенное электрическое возбуждение нервной клетки.

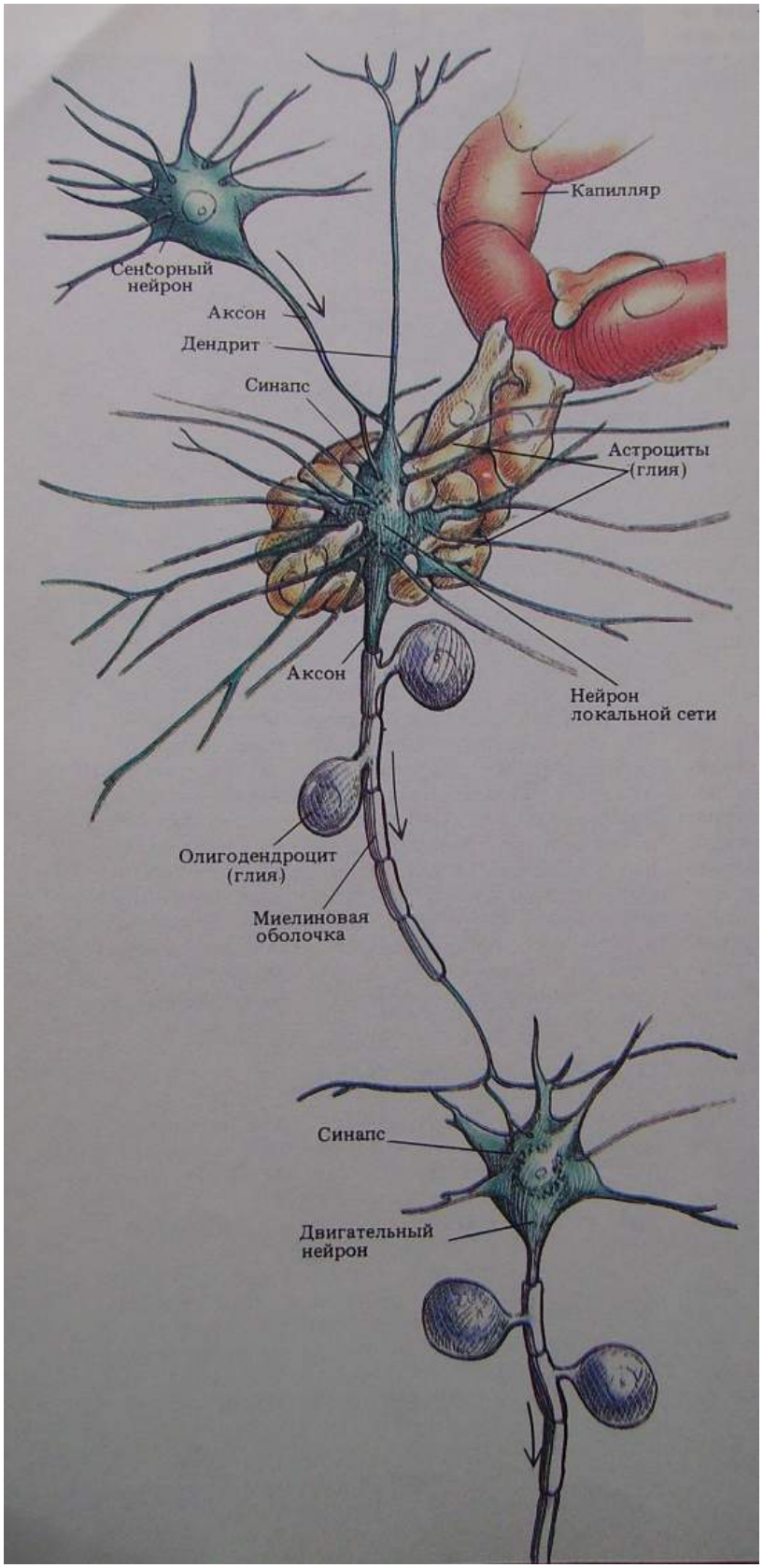
Количество взаимодействующих друг с другом нервных клеток чрезвычайно велико. Считается, что человеческий мозг содержит около 10^{11} нейронов, каждый из которых выполняет относительно примитивные функции суммирования весовых коэффициентов входных сигналов и сравнения полученной суммы с пороговым значением. Каждым нейрон имеет свои веса и свое пороговое значение.

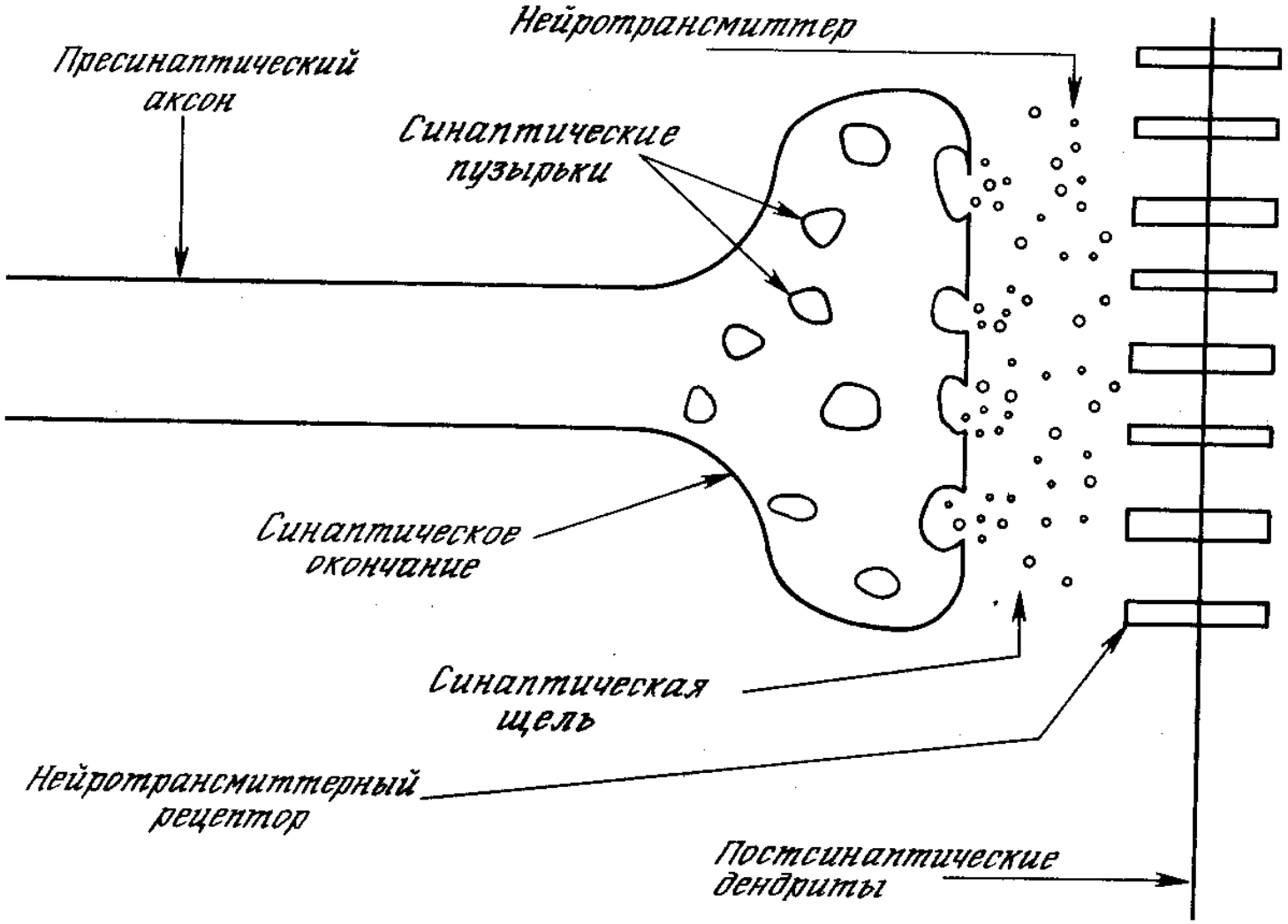
Огромное количество нейронов и межнейронных связей (до 1000 входов в каждый нейрон) приводит к тому, что ошибка в срабатывании отдельного нейрона остается незаметной в общей массе взаимодействующих клеток. Нейронная сеть проявляет высокую устойчивость к помехам - это "стабильная" сеть, в которой отдельные сбои не оказывают существенного

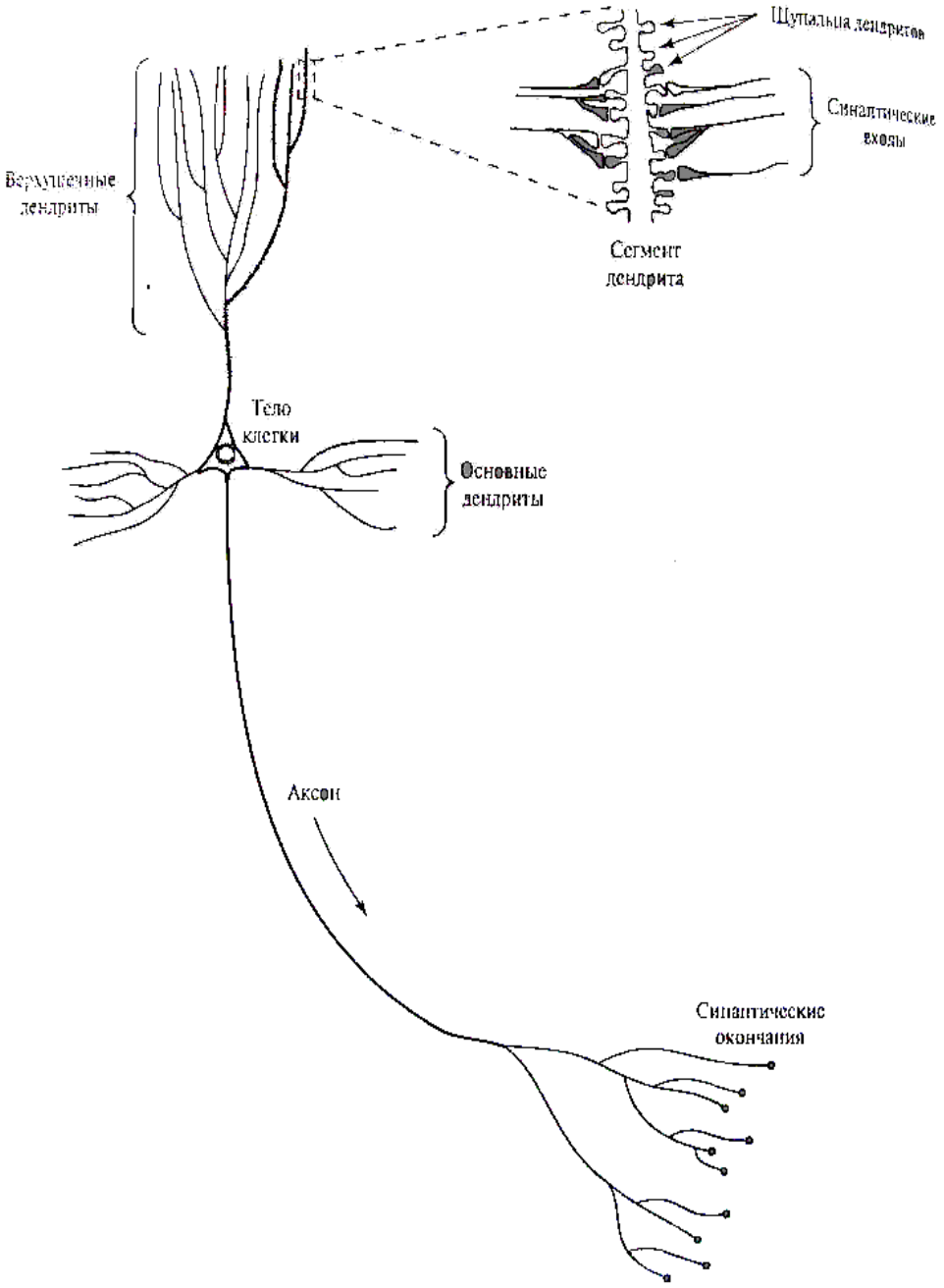
влияния на результаты ее функционирования. Таково главное отличие нейронных систем от обычных электронных систем, созданных человеком. Следует подчеркнуть, что ни одна современная технология не позволяет построить искусственную нейронную сеть, близкую по масштабам к нейронной сети мозга. Однако изучение и копирование биологических нервных систем позволяют надеяться на создание нового поколения электронных устройств, имеющих аналогичные характеристики.

Другая важная особенность нервных систем — высокая скорость их функционирования, несмотря на относительно длительный цикл срабатывания каждой отдельной клетки, измеряемый в миллисекундах. Она достигается благодаря параллельной обработке информации в мозге огромным количеством нейронов, соединенных многочисленными межнейронными связями. Такие операции, как распознавание образов и звуков либо принятие решений, выполняются человеческим мозгом за промежутки времени, измеряемые миллисекундами. Достижение такого результата при использовании полупроводниковой технологии VLSI все еще выходит за границы современных технических возможностей, хотя цикл срабатывания отдельных исполнительных элементов СБИС является достаточно коротким. Если удастся, взяв за образец нервную систему, создать устройство с высокой степенью параллельности выполнения независимых операций, то скорость его функционирования может быть существенно увеличена и приближена к уровню, наблюдаемому в процессах обработки информации биологическими объектами.

Искусственные нейронные сети возникли на основе знаний о функционировании нервной системы живых существ. Они представляют собой попытку использования процессов, происходящих в нервных системах, для выработки новых технологических решений.


Капилляр
Сенсорный нейрон
Аксон
Дендрит
Синапс
Астроциты (глия)
Аксон
Нейрон локальной сети
Олигодендроцит (глия)
Миелиновая оболочка
Синапс
Двигательный нейрон


Нейротрансмиттер
Пресинаптический аксон
Синаптические пузырьки
Синаптическое окончание
Синаптическая щель
Нейротрансмиттерный рецептор
Постсинаптические дендриты



Упрощенная структура биологической нервной клетки

1.2 Теоретические предпосылки

В различных приложениях часто возникает проблема нелинейности и сложных нелинейных систем. Как хорошо известно, прямой анализ этих систем может быть сложным, а решение может быть NP-задачей. По этой причине часто желательно иметь простую аппроксимационную модель, которая может быть использована для синтеза, анализа и идентификации исходной системы. Цель - получить модель, которая может быть использована для экстраполяции и управления. Проблема идентификации нелинейной системы может быть решена удовлетворительно, если имеется достаточно широкий класс аналитических моделей, способных обеспечить хорошую аппроксимацию нелинейной системы.

В работах по функциональному анализу и теории функций сформулированы основные положения теории аппроксимации функций. В частности, хорошие результаты достигнуты в полиномиальной аппроксимации. Ключевое место здесь занимает хорошо известная теорема Вейерштрасса о возможности равномерной аппроксимации с помощью полиномов с любой точностью непрерывной действительной функции f, определенной на диапазоне [-1,1]. Полученные результаты используются для обоснования возможности аппроксимации с помощью нейронных сетей.

В многочисленных работах зарубежных авторов есть ссылки на теорему Колмогорова о точном представлении функций как основание для возможности аппроксимации с помощью нейронных сетей. Однако, известно, что в теореме Колмогорова речь идет о точном представлении функций. Нейронные сети – неточное устройство, решающее задачи аппроксимации, приближения функции. Зачастую попытка наиболее точного обучения нейронной сети приводит к плохим результатам тестирования. Поэтому более правильным представляется ссылаться на следующие теоремы об аппроксимации, позволяющие использовать нейронные сети для решения прикладных задач:

1) теорема Вейерштрасса (1885)

2) теорема Стоуна М. (1947)

3) обобщенная теорема Стоуна (1990-е).

В работах А.Н. Горбаня и его школы доказаны теоремы о полноте класса функций, вычислимых нейронными сетями: для любой непрерывной функции нескольких переменных можно построить нейронную сеть, которая вычисляет эту функцию с любой наперед заданной точностью. Для этого достаточно, чтобы класс нейронных сетей содержал сети с нелинейной характеристической функцией нейронов; функция может быть произвольной непрерывной функцией. Рассматривались линейные пространства непрерывных функций вещественных переменных, замкнутых относительно операции суперпозиции. Доказано, что если такое пространство содержит

константы, линейные функции и хотя бы одну нелинейную функцию, то оно плотно в пространстве всех непрерывных функций в топологии равномерной сходимости на компактах. Доказана Обобщенная теорема Стоуна об аппроксимации функций, которая обобщает и классическую теорему Стоуна и аппроксимацию функций многих переменных суперпозициями и линейными комбинациями функций одной переменной. Она относится к линейным пространствам непрерывных функций, замкнутым относительно любой нелинейной операции. Теорема интерпретируется как утверждение об универсальных аппроксимационных возможностях произвольной нелинейности.

Обобщенная аппроксимационная теорема. Пусть $E \subseteq C(X)$ - замкнутое линейное подпространство в $C(X)$, $1 \in E$, функции из E разделяют точки в X и E замкнуто относительно нелинейной унарной операции $f \in C(R)$. Тогда $E = C(X)$.

Названные выше теоремы являются частью теоретического основания для использования нейронных сетей и позволяют говорить о возможности применения нейронных сетей для решения прикладных задач, в том числе биолого-экологических и эколого-медицинских.

1.3. Элементы искусственных нейронных сетей

Значительная часть аппарата нейронных сетей заимствована в других разделах математики, среди которых методы оптимизации, численные методы, распознавание образов, математическая статистика. Многие методы и способы созданы для нейроинформатики и в нейроинформатике. При заимствовании готовые методы преобразованы так, чтобы их можно было реализовать на высокопараллельных компьютерных системах. Кроме того, как было сказано выше, нейроинформатика предложила некоторые новые подходы к решению вспомогательных задач, оказавшихся важными для пользователей.

Для преобразования старых и разработки новых методов и алгоритмов нейроинформатика использует специальные способы описания архитектуры и методов обучения. Самые распространенные элементы архитектуры нейронных сетей - синапс, сумматор, нелинейный элемент, точка ветвления. Эти элементы не обязательно используются при построении нейрокомпьютеров и нейроимитаторов, но очень удобны для описания нейронных сетей.

Для описания алгоритмов и устройств в нейроинформатике выработана специальная "схемотехника", в которой элементарные устройства – сумматоры, синапсы, нейроны и т.п. объединяются в нейронные сети, предназначенные для решения различных задач. Используемая в

нейроинформатике идеальная схемотехника представляет собой особый язык для представления нейронных сетей и их обсуждения. При программной и аппаратной реализации выполненные на этом языке описания переводятся на языки другого уровня, более пригодные для реализации. При этом элементы нейронных сетей вовсе не обязательно реализуются как отдельные части или блоки.

Простой сумматор (рисунок 1) не имеет настраиваемых параметров. Он имеет векторный вход и выдает на выходе сумму координат входного сигнала.

Очень важный элемент нейросетей – это адаптивный сумматор. *Адаптивный сумматор* вычисляет скалярное произведение вектора входного сигнала *x* на вектор параметров α. Другими словами, он вычисляет линейную однородную функцию (x, α), имеет n настраиваемых параметров. На схемах будем обозначать его так, как показано на рис. 2. Адаптивным называем его из-за наличия вектора настраиваемых параметров α.

Для многих задач полезно иметь неоднородную линейную функцию выходных сигналов. Ее вычисление также можно представить с помощью адаптивного сумматора, имеющего n+1 вход и получающего на 0-й вход постоянный *единичный сигнал*. Здесь главное – подавать на один из входов постоянную. Использование единицы чаще всего удобнее, хотя не обязательно. Сумматор с таким дополнительным входом называют *неоднородный адаптивный сумматор* (рис. 3).

В некоторых задачах могут рассматривать и другие типы сумматоров, например, квадратичный. Квадратичный сумматор вычисляет квадратичную форму от вектора входного сигнала *X*. Выходной сигнал равен $\Sigma\Sigma QijXiXj$. При размерности входного сигнала n квадратичный сумматор имеет n(n+1)/2 настраиваемых параметров Qij, а также n+1 обычных параметров W. Сумматор этого типа хорошо исследован в работах. Доказано, что квадратичная зависимость − простейшая нелинейность, требующаяся для полноты класса нейросетевых функций. Эксперименты показали, что нейронные сети с квадратичными сумматорами имеют более высокую (в 2-4 раза) скорость обучения и большую способность к обобщению, однако, сложность реализации нейроимитатора возрастает многократно.

Линейная связь - *синапс* – есть линия передачи сигнала, имеет один настраиваемый параметр, который называется *вес синапса* или *синаптический коэффициент* α. Синапс получает на входе скалярный сигнал *x* - *входной сигнал* - и выдает на выходе произведение αx. Отдельно от сумматоров синапс обычно не встречается, однако для некоторых рассуждений бывает удобно выделить этот элемент (рис. 4). Как правило, набор синапсов рассматривается вместе с сумматором, к которому они подключены, а веса синапсов одновременно служат весами адаптивного сумматора. Веса синапсов сети образуют набор адаптивных параметров,

настраивая которые, нейронная сеть обучается решению задачи. Обычно на диапазон изменения весов синапсов накладываются некоторые ограничения, например, принадлежности веса синапса диапазону [-1, 1] или другой удобный диапазон.

Адаптивный сумматор может быть представлен с использованием *n* линейных связей и простого сумматора (рисунок 5).

Нелинейный преобразователь сигнала изображен на рис. 6. Он получает скалярный входной сигнал x и переводит его в φ(x). Функцию φ принято называть *функция активации* или *характеристическая функция* (activation function). Как правило, она нелинейна, но иногда может быть и линейной функцией. Функция активации может быть ступенчатой (sign) или гладкой. Среди гладких функций чаще всего используются сигмоидные. Подробнее некоторые виды функций активации обсуждаются ниже.

Точка ветвления служит для рассылки одного сигнала по нескольким адресам (рис. 7). Она получает скалярный входной сигнал x и передает его всем своим выходам.

Формальный нейрон чаще всего состоит из входного сумматора, нелинейного преобразователя и точки ветвления на выходе (рис. 8). Существуют архитектуры нейронных сетей, в которых нейрон имеет два сумматора - нейрон Паде. Возможны также и другие типы нейронов и сумматоров, например, нейрон второго слоя в некоторых сетях Кохонена определяет максимальный (или минимальный) из пришедших к нему сигналов, иногда такие блоки даже не считают нейронами.

Иногда полезно объединять синапсы и точки ветвления, образуя элемент, называемый *выходная звезда*. На входе он получает один сигнал (A), на *k* своих выходах выдает сигналы $\alpha_i A$ ($i = 1,\ldots,k$). Выходная звезда – элемент, двойственный адаптивному сумматору, – переходит в него, если обратить распространение сигналов. Аналогично этому точка ветвления двойственна простому сумматору.

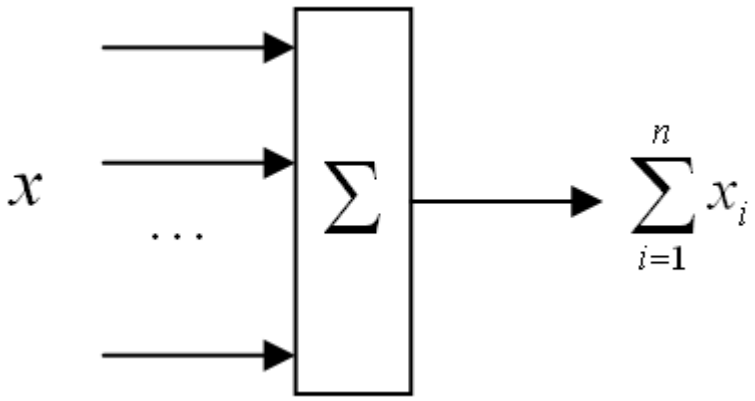


Рис. 1. Простой сумматор

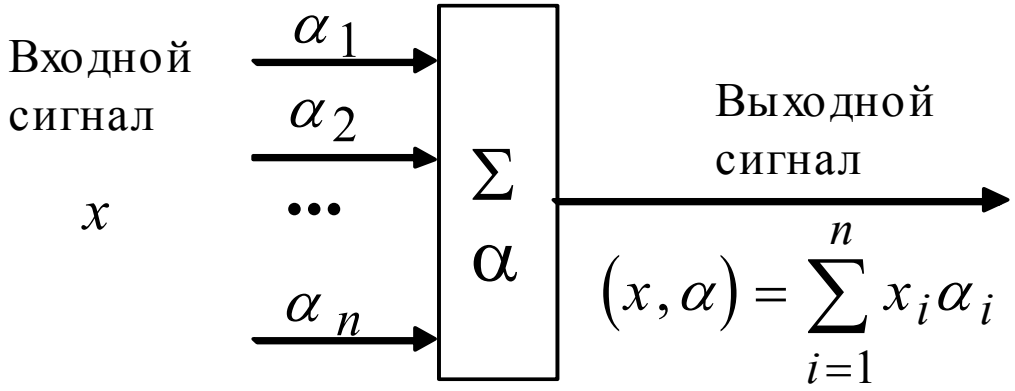


Рис. 2. Адаптивный сумматор

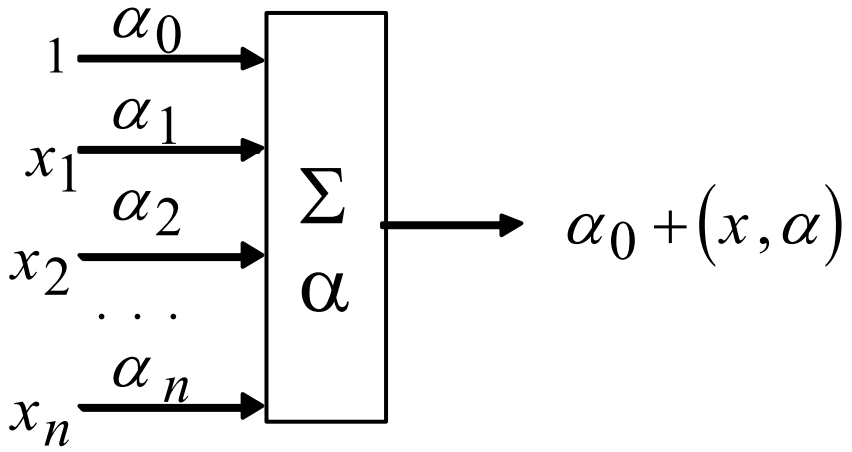


Рис. 3. Неоднородный адаптивный сумматор

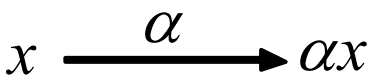


Рис. 4. Синапс или связь

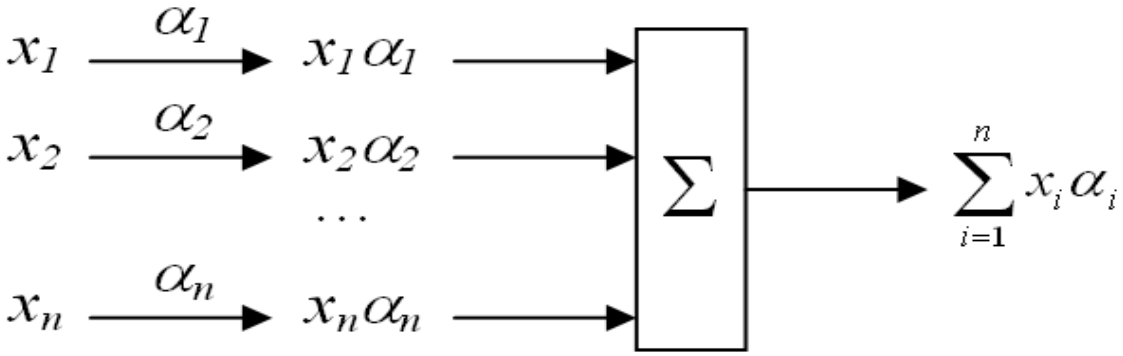


Рис. 5. Адаптивный сумматор, представленный как композиция n линейных связей и простого сумматора

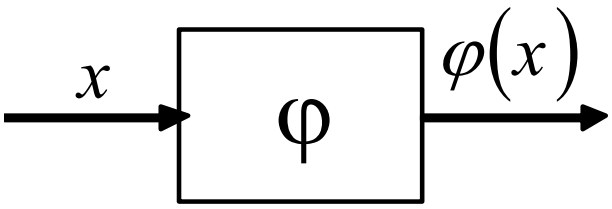


Рис. 6. Нелинейный преобразователь сигнала

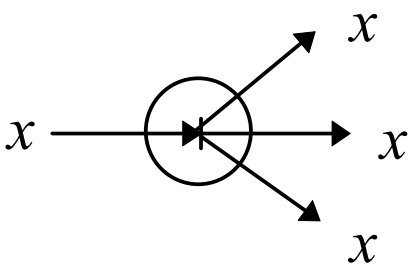


Рис. 7. Точка ветвления

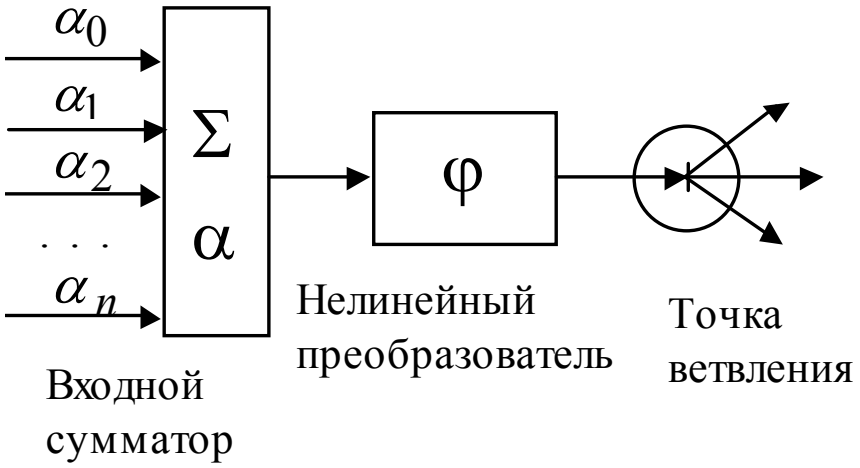


Рис. 8. Формальный нейрон

Перечисленные выше простые элементы нейронных сетей могут быть объединены в более сложные элементы: блок, слой, колонка. Чаще всего используется слой. *Слой* (layer) – набор нейронов или сумматоров, (псеводо)одновременно воспринимающий входную информацию и (псеводо)одновременно генерирующих выходные сигналы.

Выделяют несколько видов слоев – входной, выходной, *скрытый* внутренний рабочий слой (*hidden*), не получающий входных сигналов и не генерирующий выходных. В зарубежных источниках принято особо выделять нейронные сети с двумя скрытыми слоями, по аналогии с теоремой Колмогорова о точном представлении функций, несмотря на отсутствие прямого отношения этой теоремы к нейронным сетям. Чаще всего

используется от 1 до 3 скрытых слоев, хотя встречается использование нейронных сетей с 9 скрытыми слоями.

Из указанного набора элементов строятся нейронные сети большинства различных нейросетевых архитектур, среди которых очень много так называемых *двойственных сетей*. Сети такого вида тоже можно разделить на типы. Чаще всего по связности выделяют особо две базовых архитектуры – *слоистые* и *полносвязные* сети, и еще несколько особенно важных дополнительных. Слоистые сети неполносвязны. Эти типы частично совпадают с делением типов по количеству слоев – однослойные и многослойные.

Следует особо оговорить, что при программной реализации на цифровых вычислительных машинах асинхронные сети и непрерывное время как правило не могут быть реализованы без больших трудностей даже приближенно. Для реализации асинхронных сетей следует использовать аналоговые или гибридные вычислительные машины. Поэтому асинхронные сети для дальнейшего обсуждения не представляют интереса. Далее будет рассматривать нейронные сети, синхронно функционирующие в дискретные моменты времени.

В слоистых сетях нейроны расположены в несколько слоев (рисунок 9). Нейроны первого слоя получают входные сигналы, преобразуют их и через точки ветвления передают нейронам второго слоя. Далее срабатывает второй слой и т.д. до k-го, который выдает выходные сигналы для интерпретатора и пользователя. Если не оговорено обратное, то каждый выходной сигнал i-го слоя подается на вход всех нейронов i+1-го. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. Хотя часто используют одинаковое число нейронов в слоях, это никак не обосновывается алгоритмами или технологией обработки информации. Чаще всего так делают по незнанию – сколько нейронов иметь в каждом слое. Общепринятый способ подачи входных сигналов: все нейроны первого слоя получают каждый входной сигнал. Особое распространение получили трехслойные сети, в которых каждый слой имеет свое наименование: первый – входной (Input), второй – скрытый (Hidden), третий – выходной (Output).

Полносвязные сети имеют один слой нейронов; каждый нейрон передает свой выходной сигнал остальным нейронам, включая самого себя. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети. Все входные сигналы подаются всем нейронам.

Для полносвязной сети входной сумматор нейрона фактически распадается на два: первый вычисляет линейную функцию от входных сигналов сети, второй – линейную функцию от выходных сигналов других нейронов, полученных на предыдущем шаге.

Выбор элементов для слоистых и полносвязных сетей – по необходимости. Для двойственных нейронных сетей обычно выбирают нейрон с адаптивным неоднородным линейным сумматором на входе.

Функция активации нейронов (характеристическая функция) φ – нелинейный преобразователь, преобразующий выходной сигнал сумматора, – может быть одной и той же для всех нейронов сети. В этом случае сеть называют однородной (гомогенной). Если же φ зависит еще от одного или нескольких параметров, значения которых меняются от нейрона к нейрону, то сеть называют неоднородной (гетерогенной).

Если функция активации имеет сигмоидный вид, то сеть может называться сигмоидная сеть.

Выбор в качестве элементов сети нейронов наиболее распространенного (рисунок 8) вида не является обязательным. Слоистая или полносвязная архитектура не налагает существенных ограничений на участвующие в ней элементы. Единственное жесткое требование, предъявляемое архитектурой к элементам сети, – это соответствие размерности вектора входных сигналов элемента (она определяется архитектурой) числу его входов.

Если полносвязная сеть функционирует до получения ответа заданное число тактов k, то ее можно представить как частный случай k-слойной сети, все слои которой одинаковы и каждый из них соответствует такту функционирования полносвязной сети.

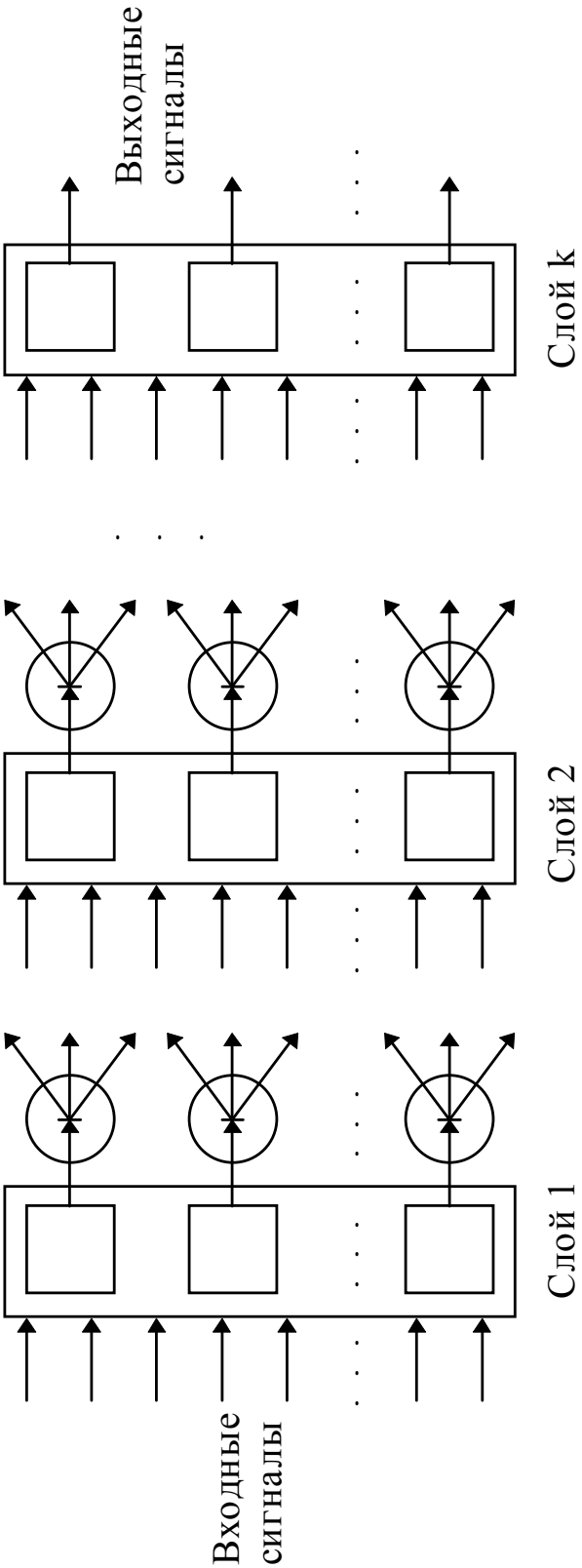


Рис. 9. Слоистая сеть

Существенное различие между полносвязной и слоистой сетями возникает тогда, когда число тактов функционирования заранее не ограничено – слоистая сеть так работать не может.

Слоисто-циклические сети отличаются тем, что слои замкнуты в кольцо – последний (*k*-й) передает свои выходные сигналы первому (рисунок 1.10). Все слои равноправны и могут как получать входные сигналы, так и выдавать выходные. Такие сети могут до получения ответа функционировать неограниченно долго – так же, как и полносвязные.

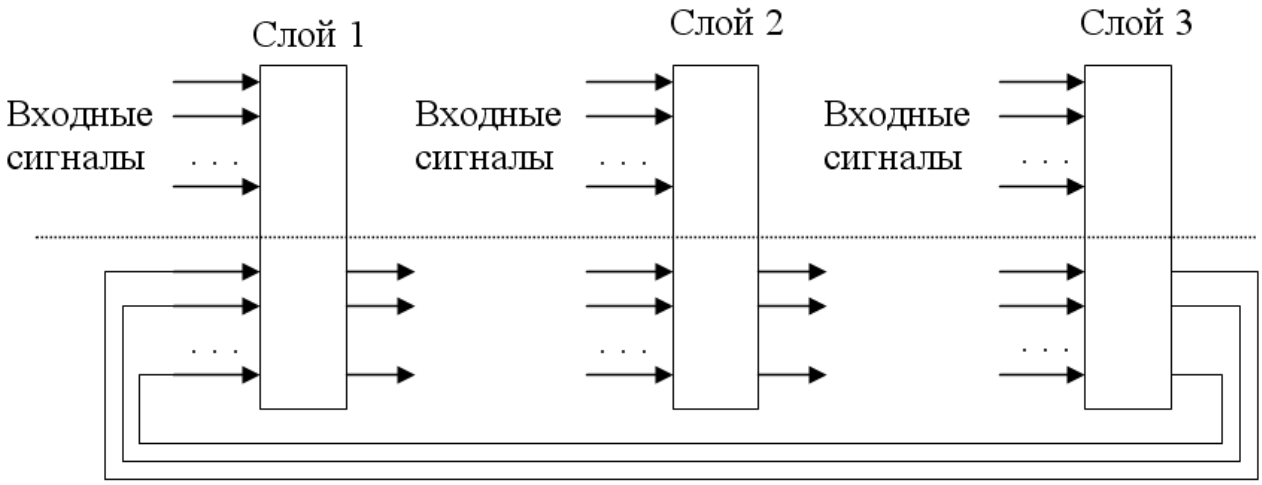


Рис. 10. Слоисто-циклическая сеть

Слоисто-полносвязные сети состоят из слоев, каждый из которых, в свою очередь, представляет собой полносвязную сеть. При функционировании сигналы передаются от слоя к слою и происходит обмен сигналами внутри слоя. В каждом слое этот процесс протекает следующим образом: прием сигналов с предыдущего слоя (и/или входных) → обмен сигналами внутри слоя → передача последующему слою (или на выход).

Подобные сети до получения ответа функционируют определенное число тактов, соответствующее количеству слоев – так же, как и слоистые сети.

Полносвязно-слоистые сети по своей структуре такие же, но функционируют по-другому. В них не разделяются фазы обмена внутри слоя и передачи следующему: на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и предыдущего, после чего передают сигналы как внутри слоя, так и последующему (или на выход). До получения ответа такие сети могут функционировать долго – так же, как и полносвязные.

Входы и выходы

Обычный способ подачи входных сигналов состоит в том, что они подаются на специальные входы элементов (для стандартных нейронов – на специальные входные сумматоры). Существует, однако, и несколько других вариантов подачи входных сигналов сети:

1) *один сигнал – один нейрон* – входные сигналы подаются сразу на нелинейные преобразователи, минуя сумматоры, которые используются только для обмена сигналами между нейронами;
2) *подача на те же сумматоры, которые служат для обмена сигналами*;
3) *параметрическая подача* – выделенному массиву параметров (обычно синаптическим весам – некоторым параметрам выделенных адаптивных сумматоров) присваиваются значения, равные входным сигналам.

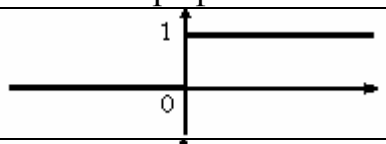
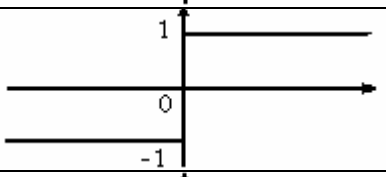
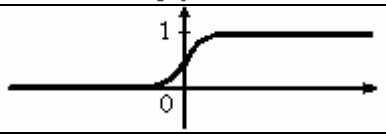
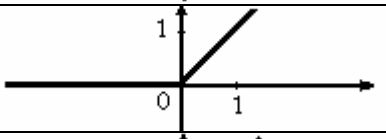
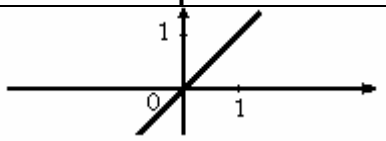
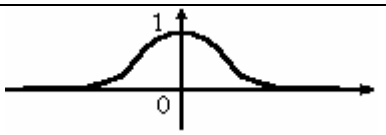
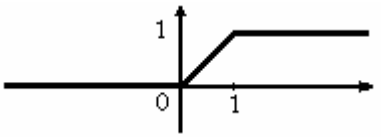
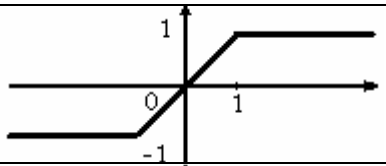
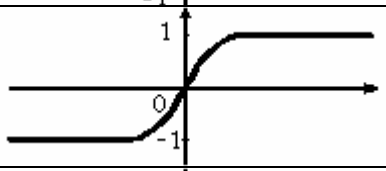
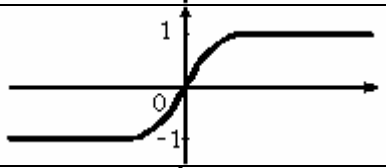
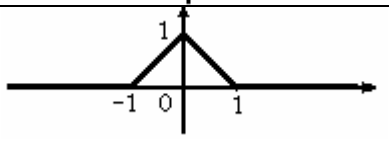
Обычный способ снятия выходных сигналов состоит в том, что некоторые нейроны объявляются выходными и их выходные сигналы в определенные моменты времени считаются выходными сигналами сети. Только это и выделяет их из всего набора нейронов.

Существует также несколько дополнительных вариантов снятия сигналов:

1) *аддитивное снятие с синапсов* – на выход подаются сигналы, проходящие по выделенным связям, уже умноженные на соответствующие веса;
2) *параметрическое снятие выходов* – выходными сигналами считаются установившиеся значения выделенных параметров (например, веса выделенных связей).

В последнем случае процесс функционирования должен включать изменение выходных синапсов. В обычной постановке задачи это соответствует, скорее, обучению. Функционирование с изменением параметров можно рассматривать как обучение со вспомогательной оценкой, параметры которой являются подстраиваемыми в ходе «настоящего» обучения, устанавливающего правильное соответствие вход-выход при данном типе входных сигналов.

Функции активации нейронов.

	Название	Функция	График
1	Пороговая	$f(u)=\begin{cases}0 & u<0\\1 & u\geq 0\end{cases}$	
2	Знаковая или сигнатурная	$f(u)=\begin{cases}1 & u>0\\-1 & u\leq 0\end{cases}$	
3	Сигмоидная или сигмоидальная	$f(u)=\frac{1}{1+e^{-u}}$	
4	Полулинейная	$f(u)=\begin{cases}u & u>0\\0 & u\leq 0\end{cases}$	
5	Линейная	$f(u)=u$	
6	Радиальная базисная (гауссова)	$f(u)=e^{-u^2}$	
	Полулинейная с насыщением	$f(u)=\begin{cases}0 & u\leq 0\\u & 0<u<1\\1 & u\geq 1\end{cases}$	
	Линейная с насыщением	$f(u)=\begin{cases}-1 & u\leq -1\\u & -1<u<1\\1 & u\geq 1\end{cases}$	
	Гиперболический тангенс (сигмоидная)	$f(u)=\frac{e^{u}-e^{-u}}{e^{u}+e^{-u}}$	
	Сигмоидная	$f(u)=\frac{u}{\lvert u\rvert+const}$	
	Треугольная	$f(u)=\begin{cases}1-\lvert s\rvert & \lvert u\rvert\leq 1\\0 & \lvert u\rvert>0\end{cases}$	

В качестве функции активации часто выступает сигмоидальная функция (т.е. функция, график которой похож на букву "S"). На практике используются как униполярные, так и биполярные функции активации.

Униполярная функция, как правило, представляется формулой

$$f(u) = \frac{1}{1+\exp(-b \cdot u)},$$

тогда как биполярная функция может быть задана в виде

$$f(u) = \tanh(-b \cdot u).$$

или

$$f(u) = \frac{u}{|u| + const}$$

Графики униполярных и биполярных сигмоидальных функций

График униполярной сигмоидальной функции

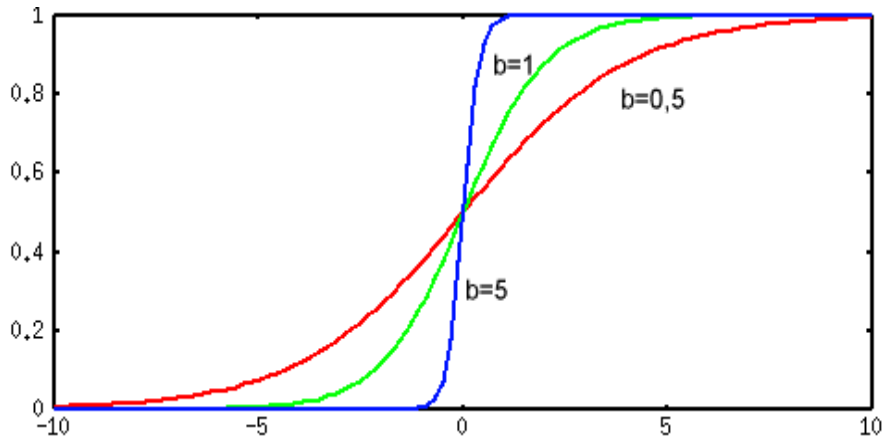
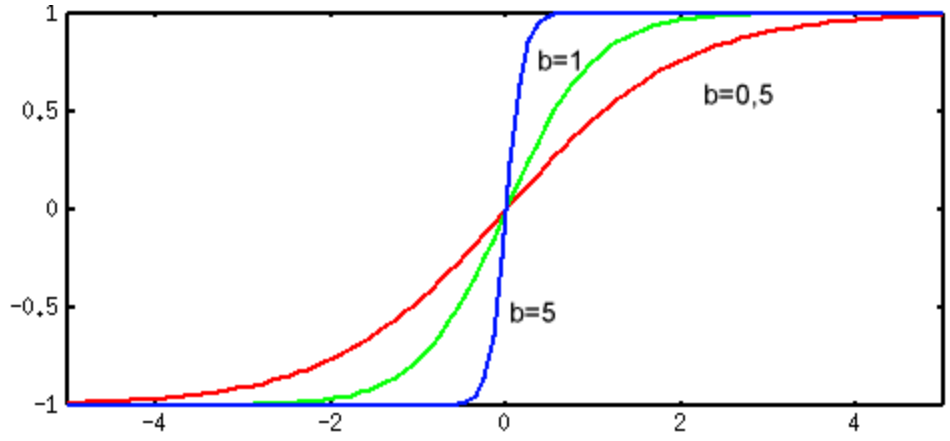


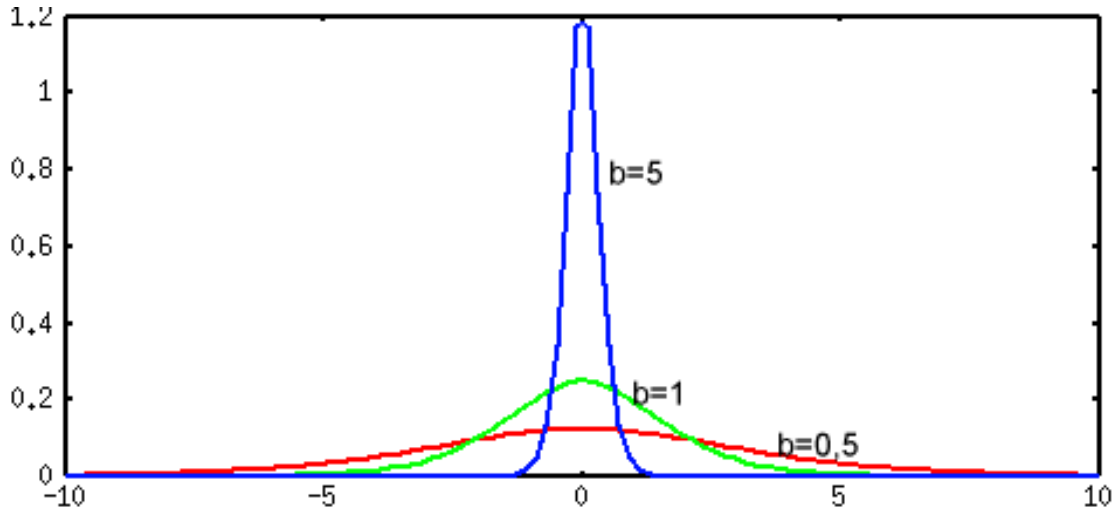
График биполярной сигмоидальной функции



Коэффициент b определяет "крутизну" функций и выбирается разработчиком сети. На практике b для упрощения назначают обычно равным 1.

Производная униполярной функции активации имеет вид -

$$\frac{df(u)}{du} = b \cdot f(u) \cdot (1 - f(u)),$$



По аналогии с электронными системами активационная функция считается нелинейной усилительной характеристикой искусственного нейрона. Коэффициент усиления вычисляется как отношение приращения величины OUT к вызвавшему его небольшому приращению величины NET. Он выражается наклоном кривой при определенном уровне возбуждения и изменяется от малых значений при больших отрицательных возбуждениях (кривая почти горизонтальна) до максимального значения при нулевом возбуждении и снова уменьшается, когда возбуждение становится большим положительным. Подобная нелинейная характеристика решает поставленную им дилемму шумового насыщения, благодаря чему одна и та же сеть может обрабатывать как слабые, так и сильные сигналы.

Другой широко используемой активационной функцией является гиперболического тангенса, часто используется биологами в качестве математической модели активации нервной клетки:

$$F = th(x) \qquad (5)$$

Гиперболический тангенс симметричен относительно начала координат, и в точке NET = 0 значение выходного сигнала OUT равно нулю (см. рисунке 3). В отличие от логистической функции гиперболический тангенс принимает значения различных знаков, что оказывается выгодным для ряда задач и приложений.

Однонаправленные многослойные сети сигмоидного типа

Однонаправленные многослойные нейронные сети сигмоидального типа получили наибольшее распространение на практике в силу относительной простоты их математического описания.

Собственно история ИНС началась с однослойных сетей данного типа. Однако, возможности однослойных сетей крайне скромны, поскольку расположенные на одном уровне нейроны функционируют независимо друг от друга, и свойства всей сети ограничены свойствами отдельных нейронов. Многослойные сети получили свое развитие с конца 70-х годов, когда были предложены эффективные алгоритмы их обучения.

Однонаправленные многослойные сети сети сигмоидального типа имеют устоявшееся название **многослойный персептрон MLP** (MultiLayer Perceptron).

1.5 Нейрокомпьютер

Из рассмотрения конструкции нейросети и основных режимов ее работы следует необходимость существования следующих блоков нейрокомпьютера:

1) основной
2) шеф
3) настройка
4) конструктор NN
5) нейросеть
6) задачник
7) редактор обучающей выборки
8) предобработчик обучающей выборки (нормировщик !)
9) обучающая выборка
10) конструктор оценки
11) оценка, оценщик, модуль оценки
12) конструктор учителя
13) tutor (учитель)
14) интерпретатор ответов нс
15) тестер
16) конструктор алгоритмов обучения
17) контрастер или модуль контрастирования НС
18) конструктор контрастера

Схема НКС

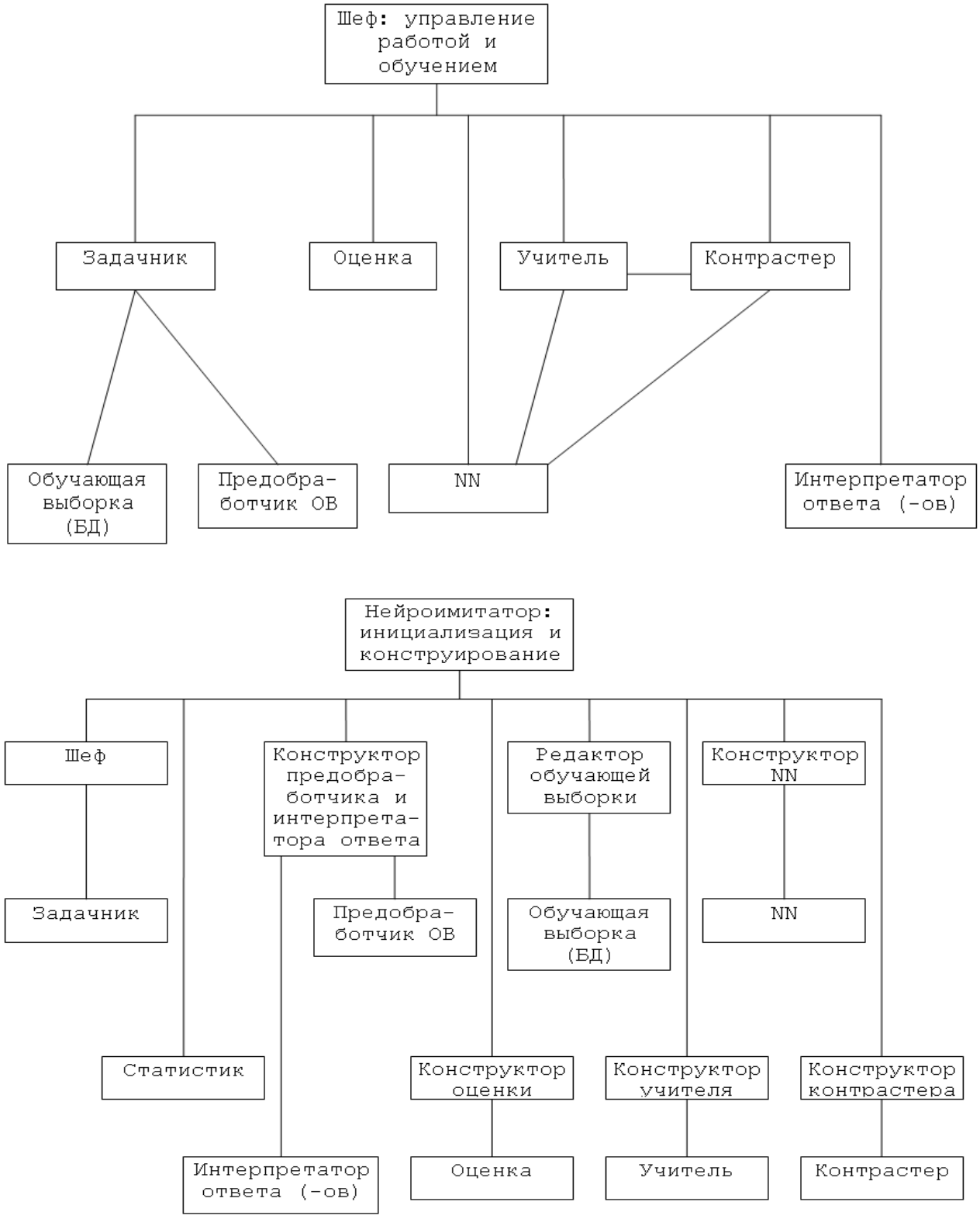


Рис. 11. Двуслойный персептрон

1.6 Решение задач нейронными сетями

Множество задач, решаемых с помощью нейронных сетей, постоянно пополняется, поэтому сложно дать их завершенную классификацию. Однако многие задачи, для решения которых используются нейронные сети, могут рассматриваться как частные случаи следующих основных проблем.

1) Построение функции по конечному набору значений.
2) Оптимизация.
3) Построение отношений на множестве объектов.
4) Распределенный поиск информации и ассоциативная память.
5) Фильтрация.
6) Сжатие информации.
7) Идентификация динамических систем и управление ими.

Нейросетевая реализация классических задач и алгоритмов вычислительной математике: решение систем линейных уравнений, решение задач математической физики сеточными методами и др.

Однозначно построить функцию (обычно многих действительных переменных) по конечному набору значений невозможно без специальных дополнительных условий. В качестве таких условий могут использоваться требования минимизации различных функционалов, например интеграла суммы квадратов вторых производных – требование максимальной гладкости. При этом известные в конечном множестве точек значения функции превращаются в набор ограничений, при которых ищется минимум функционала.

С помощью нейронных сетей строится, естественно, нейросетевая реализация функции: создается нейронная сеть, которая, получая на входе вектор аргументов, выдает на выходе значение функции. Обычно предполагается, что типичные нейросетевые реализации достаточно «хорошие» и любая из них подойдет в качестве решения задачи. При необходимости вместо наложения условий типа максимальной гладкости минимизируют число слоев, количество нейронов и/или число связей, а также требуют, чтобы функция активации нейронов была «максимально пологой».

К построению функции по конечному набору значений приводит одна из самых нужных для пользователей задач: заполнение пропусков в таблицах. Пусть, как обычно, каждая строка таблицы данных соответствует какому-либо объекту, а в столбцах указаны значения признаков (свойства) соответствующих объектов.

В подавляющем случае данные неполны: по крайней мере, для части объектов неизвестны значения некоторых признаков. Необходимо как-то их восстановить. Достоверное статистическое оценивание должно давать для

отсутствующих данных их условное математическое ожидание (условия – известные значения других признаков) и характеристику разброса – доверительный интервал. Это, однако, требует либо непомерно большого объема известных данных, либо очень сильных предположений о виде функции распределения. Приходится вместо статистически достоверных уравнений регрессии использовать правдоподобные нейросетевые реализации.

Можно выделить два класса задач, часто решаемых обучаемыми нейронными сетями. Это задачи предсказания и классификации.

Задачи предсказания или прогнозирования являются, по существу, задачами построения регрессионной зависимости выходных данных от входных. Нейронные сети могут эффективно строить сильно нелинейные регрессионные зависимости. Специфика здесь такова, что, поскольку решаются в основном неформализованные задачи, то пользователя интересует в первую очередь не построение понятной и теоретически обоснованной зависимости, а получение устройства-предсказателя. Прогноз такого устройства непосредственно не пойдет в дело – пользователь будет оценивать выходной сигнал нейросети на основе своих знаний и формировать собственное экспертное заключение. Исключения составляют ситуации, на основе обученной нейронной сети создают устройство управления для технической системы.

При решении задач классификации нейронная сеть строит разделяющую поверхность в признаковом пространстве, а решение о принадлежности ситуации тому или иному классу принимается самостоятельным, не зависящим от сети устройством – интерпретатором ответа сети. Наиболее простой интерпретатор возникает в задаче бинарной классификации (классификации на два класса). В это случае достаточно одного выходного сигнала сети, а интерпретатор относит, например, ситуацию к первому классу, если выходной сигнал меньше нуля, и ко второму, если он больше или равен нулю.

Классификация на несколько классов требует усложнения интерпретатора. Широко используется интерпретатор "победитель забирает все", где число выходных сигналов сети равно числу классов, а номер класса будет соответствовать номеру максимального выходного сигнала.

Одна нейронная сеть может одновременно предсказывать несколько чисел, либо одновременно решать задачи и прогнозирования, и классификации. Потребность в последнем возникает, однако, крайне редко, и лучше решать разнотипные задачи отдельными нейронными сетями.

Среди множества особенностей обучения нейронных сетей можно сопоставить случайные и направленные методы оптимизации. И те и другие методы используются в прикладной математике, в том числе в теории нейронных сетей.

Можно сформулировать два предельно упрощенных утверждения о свойствах этих подходов:

1. Направленные методы гарантируют достижение минимума за конечное число шагов (в конечное время), но минимума локального.

2. Случайные методы гарантируют достижение глобального минимума, но за бесконечное время.

К сожалению, гарантированно получить глобальный минимум за конечное время не получается. Существуют комбинированные методы, одним из известных и часто используемых подходов для комбинирования следует назвать генетические алгоритмы и алгоритмы на их основе. Комбинирование методов улучшает качество работы, но гарантированно получить глобальный минимум за конечное время не получается.

Функционирование слоистых нейронных сетей

Функционирование слоистых нейронных сетей рассмотрим на примере трехслойной нейросети, изображенной на рисунке 2.1. Сеть состоит из пяти нейронов, имеет два входа x1 и x2 и один выход z. Нейроны обозначены кружками, синапсы – направленными стрелками. f1, f2, …, f5 – функции нелинейного преобразования, в общем случае разные для каждого нейрона. w1, w2, …, w10 – веса синапсов. Черными кружками обозначены точки ветвления для входов.

Сеть считается обученной, если все ее веса w1, w2, …, w10 имеют конкретные значения, подобранные таким образом, что если на входы x1 и x2 подать входные данные, то на выходе z получим некоторый результат, согласующийся с нашей задачей.

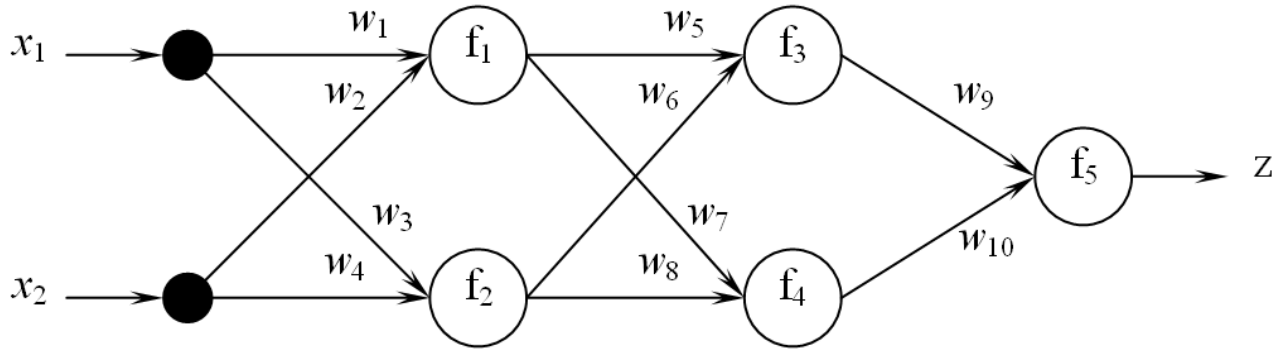


Рис. 12. Модель трехслойной нейросети

Для нашего примера можно написать явное выражение для z как функции двух переменных x_1 и x_2:

$$z = f_5\left(w_9 f_3\left(w_5 f_1\left(w_1 x_1 + w_2 x_2\right) + w_6 f_2\left(w_3 x_1 + w_4 x_2\right)\right) + \right.$$
$$\left. + w_{10} f_4\left(w_7 f_1\left(w_1 x_1 + w_2 x_2\right) + w_8 f_2\left(w_3 x_1 + w_4 x_2\right)\right)\right) \qquad (2.1)$$

Введем дополнительные обозначения: t_i – сумма входов i-го нейрона, умноженных на веса соответствующих синапсов (значение на выходе адаптивного сумматора); y_i – выходное значение i-го нейрона (значение на выходе нелинейного преобразователя), i=1, 2, …, 5. Тогда:

$$\begin{aligned}
t_1 &= w_1 x_1 + w_2 x_2 & y_1 &= f_1\left(t_1\right) \\
t_2 &= w_3 x_1 + w_4 x_2 & y_2 &= f_2\left(t_2\right) \\
t_3 &= w_5 y_1 + w_6 y_2 & y_3 &= f_3\left(t_3\right) \\
t_4 &= w_7 y_1 + w_8 y_2 & y_4 &= f_4\left(t_4\right) \\
t_5 &= w_9 y_3 + w_{10} y_4 & z &= y_5 = f_5\left(t_5\right)
\end{aligned} \qquad (2.2)$$

Таким образом вычисление функции z по нейросети идет слева направо, от слоя к слою. Если сохранять промежуточные результаты в каждом нейроне (значения t_i и y_i), то вычисления в каждом нейроне сводятся к нахождению скалярного произведения вектора входов на соответствующие веса и последующему вычислению функции f_i от полученного результата.

Будем считать, что трудоемкость вычислений (затрачиваемое время, загрузка процессора и т.д.) для каждого нейрона одинакова и равна H. Тогда трудоемкость вычислений по всей сети равна NH, где N – число нейронов.

В случае, если нейросеть имеет m выходов, то для всех выходов вычисления производятся аналогично. При этом трудоемкость вычислений возрастает не в m раз, как это может показаться при анализе формулы (2.1), а на величину, равную (m-1)H, по сравнению с трудоемкостью для той же сети, но с одним выходом.

Для указанной сети трудоемкость вычислений равна 5H.

Подача входных сигналов и снятие выходных сигналов сети

На веса синапсов сети обычно наложены требования принадлежности некоторому диапазону значений. Наиболее часто используемые нелинейные

функции нейронов также обычно принимают значения из некоторого диапазона. Это приводит к тому, что обычно нельзя подавать сети входные сигналы в их истинном диапазоне величин и получать от сети выходные сигналы в требуемом диапазоне.

Поэтому перед подачей сети входных сигналов их необходимо нормировать, например, в диапазон значений [-1,1] или [0,1] в тех случаях, когда существенна положительность, либо делать так, чтобы входные сигналы не слишком сильно выходили за пределы этих отрезков. Нормирование можно выполнить по указанным ниже формулам, в которых каждая i-я компонента входного вектора данных xi заменяется новой величиной. Нормирование на диапазон [0,1]:

$$x_i' = \frac{x_i - min\, x_i}{max\, x_i - min\, x_i},$$

нормирование на диапазон [-1,1]:

$$x_i' = \frac{x_i - (max\, x_i + min\, x_i)/2}{(max\, x_i - min\, x_i)/2},$$

где max xi и min xi – соответственно максимальное и минимальное значение для данной компоненты, вычисленные по всей обучающей выборке.

По этим же формулам пересчитываются и компоненты векторов ответов.

Можно нормировать и по-другому, например, пересчитывать выборку так, чтобы разброс данным был единичным.

Здесь имеется одна сложность. Любое изменение обучающей выборки должно соответственно менять и правило нормирования данных. Обычно в качестве минимальных и максимальных значений берут данные из существующего на настоящий момент задачника и в дальнейшем нормирование не меняют, но если предполагается, что в дальнейшем поступят сильно отличающиеся данные, то значения min и max задаются пользователем по его оценкам. Иными словами, эти величины должны вводиться в момент создания сети и в дальнейшем не зависеть от обучающей выборки.

Выходные сигналы сети должны нормироваться в диапазон истинных значений по обращенным формулам.

Для сетей-классификаторов нормировка выходных сигналов не нужна, поскольку пользователь получает не собственно выходные сигналы сети, а результат отнесения ситуации интерпретатором ответа к тому или иному классу. Здесь каждый выходной вектор задачника преобразуется так, чтобы правильно обрабатываться интерпретатором ответа. Например, при классификации на три класса и интерпретаторе "победитель забирает все"

номер класса будет кодироваться вектором из трех чисел, где компонента вектора, соответствующая номеру класса, имеет значение 1, а остальные две – -1.

Другой проблемой является ситуация, когда в таблице данных имеются пробелы. Вместо таких отсутствующих компонент данных можно подавать нуль, можно исключать некомплектные вектора из обучающей выборки, можно перед обучением сети решать задачу заполнения пробелов в данных некоторыми правдоподобными значениями. Последнее является самостоятельной, часто встречающейся и достаточно сложной проблемой.

$$H = \sum_{i=1}^{N} \left\| Y_i - Y_i' \right\| \longrightarrow min$$

Обучение нейронных сетей

Обучение нейронной сети на "задачнике" означает минимизацию некоторого функционала невязки между выходными сигналами сети и сигналами, которые требуется получить. Минимизация функционала невязки (оценки в дальнейшем) производится путем такой подстройки обучаемых параметров a_i нейронов сети, чтобы сеть на некоторый входной вектор сигналов X выдавала такой ответ Y', который был бы по возможности ближе к требуемому ответу Y. Иными словами, чтобы для задачника с числом примеров N (т.е. для N пар $\{X_i, Y_i\}$, i=1,..,N) достигался минимум суммарной функции оценки.

В качестве функции нормы выступает покомпонентная сумма квадратов элементов вектора Y-Y' (оценка МНК), либо более специализированная.

Минимизация функции оценки выполняется с привлечением градиентных методов оптимизации. Преобразовав по некоторым правилам структуру сети, подавая на выход сети частные производные функции оценки по выходным сигналам и используя так называемое двойственное функционирование, мы можем получить для каждого подстроечного параметра и для каждого входного сигнала сети частные производные функции оценки по значению этого параметра или сигнала. Для вектора параметров сети вектор соответствующих частных производных будет градиентом функции оценки, поэтому возможна градиентная оптимизация функции оценки, в ходе которой нейронная сеть "обучается" давать требуемые ответы на подаваемые ей входные сигналы. Сеть лучше обучать по суммарному градиенту (градиенту по полному задачнику), что ускоряет процесс обучения, и применять специализированные алгоритмы

оптимизации, надстраиваемые над простейшим градиентным спуском. Для вычисления суммарного градиента необходимо просуммировать вектора градиентов, вычисляемые для каждого примера задачника (всего N векторов).

Обучение слоистых нейронных сетей

Обучение нейронной сети представляет собой автоматический поиск закономерности между совокупностью обучающих данных и заранее известным результатом. Обучающие данные образуют обучающую выборку – электронную таблицу или базу данных, каждая строка которых содержит один обучающий пример. Обучающий пример состоит из вектора входных данных размерностью, равной числу входов нейросети, и вектора выходных данных, соответствующего выходам нейросети. Задача обучения состоит в том, чтобы нейросеть в ответ на вектор входных данных выдавала такой вектор выходных данных, который был бы наиболее близок к выходным данным из примера.

Чтобы сравнить, насколько отличается ответ сети от ожидаемого ответа (по примеру), можно использовать различные нормы:

$$H_1 = \sqrt{\sum_{i=1}^{n} (\tilde{z}_i - z_i)^2} \qquad (3.1)$$

$$H_2 = \max_i |\tilde{z}_i - z_i|, \ i = \overline{1,n}, \qquad (3.2)$$

где n – размерность вектора с выходными данными;

$\tilde{z}_i$ - значение i-го выхода по примеру;

z_i - значение i-го выхода, возвращаемое нейросетью.

Таким образом, задача обучения нейросети сводится к минимизации ошибки H как функции весов нейросети w_1, w_2, …, w_n. Это означает, что весь мощный арсенал методов оптимизации может быть испытан для обучения.

Существует, однако, ряд специфических ограничений. Они связаны с огромной размерностью задачи обучения. Число параметров может достигать 10^8 – и даже более. Уже в простейших программных имитаторах на персональных компьютерах подбирается $10^3 – 10^4$ параметров.

Из-за высокой размерности возникает два требования к алгоритму:

1) Ограничение по памяти. Пусть n – число параметров. Если алгоритм требует затрат памяти порядка n^2, то он вряд ли применим для обучения. Вообще говоря, желательно иметь алгоритмы, которые требуют затрат памяти порядка Kn, $K = \text{const}$.

2) Возможность параллельного выполнения наиболее трудоемких этапов алгоритма, и желательно нейронной сетью. Если какой-либо особо привлекательный алгоритм требует памяти порядка n^2, то его все же можно использовать, если с помощью анализа чувствительности сократить число обучаемых параметров до разумных пределов.

Еще два обстоятельства связаны с нейрокомпьютерной спецификой.

1) Обученный нейрокомпьютер должен с приемлемой точностью решать все тестовые задачи (или, быть может, почти все с очень малой частью исключений). Поэтому задача обучения становится по существу многокритериальной задачей оптимизации: надо найти точку общего минимума большого числа функций. Обучение нейрокомпьютера исходит из гипотезы о существовании такой точки. Основания гипотезы – очень большое число переменных и сходство между функциями. Само понятие «сходство» здесь трудно формализовать, но опыт показывает, что предположение о существовании общего минимума или, точнее, точек, где значения все оценок мало отличаются от минимальных, часто оправдывается.

2) Обученный нейрокомпьютер должен иметь возможность приобретать новые навыки без утраты старых. Возможно более слабое требование: новые навыки могут сопровождаться потерей точности в старых, но потеря не должна быть особо существенной, а качественные изменения должны быть исключены. Это означается, что в достаточно большой окрестности найденной точки общего минимума оценок их значения незначительно отличаются от минимальных. Мало того, что должна быть найдена точка общего минимума, так она еще должна лежать в достаточно широкой низменности, где значения всех минимизируемых функций близки к минимуму. Для решения этой задачи нужны специальные средства.

Итак, имеет четыре специфических ограничения, выделяющих обучение нейрокомпьютера из общих задач оптимизации: астрономическое число параметров, необходимость высокого параллелизма при обучении, многокритериальность решаемых задач, необходимость найти достаточно широкую область, в которой значения всех минимизируемых функций близки к минимальным. В остальном, это просто задача оптимизации, и многие классические и современные методы достаточно естественно ложатся на структуру нейронной сети.

3.1 Методы оптимизации

Все пошаговые методы оптимизации состоят из двух важнейших частей: выбора направления и выбора шага в данном направлении. Тогда нахождение минимума функции H сведется к итерационному процессу:

$$w^{k+1} = w^k + hS \tag{3.3}$$

при условии, что

$$H\left(w^{k+1}\right) < H\left(w^k\right), \tag{3.4}$$

где w^k и w^{k+1} - векторы весов на k-й и k+1-й итерации;

h – шаг;

S – направление движения.

Таким образом, направление и шаг нужно выбирать так, чтобы на каждой итерации целевая функция уменьшалась. Когда уменьшение функции прекратится, или расхождения между известными выходными значениями и ответами нейросети будут различаться достаточно мало, процесс обучения можно считать законченным.

Методы одномерной оптимизации дают эффективный способ для выбора шага. Использование оптимального шага в задачах многокритериальной оптимизации невозможно, если критерии рассматриваются «по одному» – сначала улучшается значение одного, потом другого и т.д. Необходим синтез обобщенного (интегрального) критерия. Простейший вариант – суммирование всех. Чуть сложнее – суммирование с весами. При ограничениях по памяти, а также в тех случаях, когда получение значения каждого критерия требует заметных затрат, возможна постраничная организация – выбирается достаточно богатый набор критериев (страница примеров), из них формируется интегральный (оценка страницы) и ведется его оптимизация.

Для обучения по одельным примерам, т.е. без постраничной организации, методы одномерной оптимизации неприменимы – они слишком эффективны, поэтому давая заметное улучшение на одном обучающем примере, они часто разрушают навык решения предыдущих. При постраничном обучении такого обычно не происходит.

3.1.1 Однопараметрическая оптимизация

Пусть нам на каждой итерации известно направление, в котором нужно минимизировать функцию H. Самое простое, что можно сделать – идти по этому направлению с постоянным шагом. Однако, т.к. рельеф функции заранее неизвестен и может иметь горбы и впадины, то более эффективно будет автоматически подбирать шаг на каждой итерации. К примеру, если целевая функция уменьшается достаточно быстро, то шаг на последующих итерациях можно оставить большим. Чем ближе аргумент функции подходит к точке минимума, тем меньше должен быть шаг.

В общем случае шаг можно найти, используя стандартные методы оптимизации. Имеем целевую функцию H(h), зависящую от одной переменной – шага h. Пусть известен отрезок $[a,b]$, на котором находится минимум функции H. Тогда точку минимума можно найти итерационными методами с заранее заданной точностью.

Самый простой метод - половинного деления (дихотомии). В этом методе на каждом шаге отрезок $[a,b]$ делится пополам точкой $c=\frac{a+b}{2}$ и вычисляется значение функции H(c). Это значение сравнивается со значениями функции на краях отрезка H(a) и H(b) и выбирается один из отрезков $[a,c]$ или $[c,b]$, внутри которого находится минимум функции. С новым отрезком повторяются те же операции. Процесс завершает работу, когда длина отрезка $[a,b]$ станет меньше заранее заданной точности ε.

Однако, на входе процедуры однопараметрической оптимизации мы имеем не отрезок $[a,b]$, а всего лишь одну его граничную точку. Вторую точку можно найти, если с некоторым шагом (постоянным или переменным) двигаться вдоль направления минимизации в обе стороны и анализировать поведение функции H. Если, например, функция H сначала уменьшалась, то второй точкой отрезка минимизации будет точка, в которой функция начала рости. Если же функция растет сразу же на первом шаге от начальной точки, то можно попробовать идти в противоположном направлении.

3.1.2 Многопараметрическая оптимизация

Существует несколько методов выбора направления движения при минимизации целевой функции.

Пусть задано начальное значение параметров w^0 и вычислена функция оценки $H=H\left(w^0\right)$. Одномерная оптимизация даст приближенное положение минимума $h(x)=H\left(w^0+xS\right)$ (вообще говоря, локального). Теперь нужно определить направление S и начальный шаг.

Наиболее очевидный выбор S для одномерной оптимизации – направление антиградиента H: $S = -\nabla H$. Выберем на каждом шаге это направление, потом проведем одномерную оптимизацию, потом – снова вычислим градиент H и т.д. Это – метод наискорейшего спуска, примитивнейший из градиентных методов. Иногда работает хорошо.

Другой способ – случайный выбор направления S для одномерной оптимизации. Получаемый метод требует большого числа шагов, но зато предъявляет минимальные требования к сети – ему необходимо только прямое функционирование с вычислением оценки. Кроме того, встречаются ситуации, в которых метод наискорейшего спуска не работает, а поиск в случайном направлении медленно, но верно вытаскивает в нужную область параметров.

Для того, чтобы исправить недостатки наискорейшего спуска, существует огромное число методов. Семейство таких методов – итерационный и модифицированный партан-метод.

Итерационный партан-метод (k-партан) стоится так. В начальной точке w^0 вычисляется градиент H и делается шаг наискорейшего спуска – для этого используется одномерная оптимизация. Далее – снова наискорейший спуск и так k раз. Начальные параметры w^0 при этом хранятся в памяти, промежуточные – нет. После k шагов наискорейшего спуска получаем w^k и проводим одномерную оптимизацию из w^0 в направлении S = $w^k - w^0$ (от w^0 к w^k) с начальным шагом $h = 1$. После этого цикл повторяется.

Модифицированный партан-метод также требует запоминания дополнительной карты параметров. Он строится так. Из w^0 делается два шага наискорейшего спуска. Получаем w^1, w^2. Далее – одномерная оптимизация из w^0 в направлении $w^2 - w^0$. Получаем w^3. После этого w^0 уже не используется. Далее – наискорейший спуск из w^3. Получаем w^4. Потом одномерная оптимизация из w^2 в направлении $w^4 - w^2$, получаем w^5 и т.д.

3.2 Вычисление градиента целевой функции

Целевая функция для нейросети, изображенной на рисунке 2.1, имеет вид:

$$H(w_1, w_2, ..., w_{10}) = (\tilde{z} - z)^2, \qquad (3.5)$$

где $\tilde{z}$ - известное значение выходной переменной для входного вектора (x_1, x_2);

z - значение выходной переменной, возвращаемое нейросетью (2.1).

Найдем частные производные H:

$$\frac{\partial H}{\partial w_i} = 2(\tilde{z} - z) \cdot \frac{\partial z}{\partial w_i} \tag{3.6}$$

Обозначим через p_i сумму произведений весов синапсов, выходящих из i-го нейрона, на соответствующие значения p_j тех нейронов, в которых эти синапсы заканчиваются, умноженную на производную функции f_i по своему аргументу.

Для нашего случая:

$$\begin{aligned} p_5 &= f_5' \\ p_4 &= f_4' \cdot w_{10} p_5 \\ p_3 &= f_3' \cdot w_9 p_5 \\ p_2 &= f_2' \cdot (w_6 p_3 + w_8 p_4) \\ p_1 &= f_1' \cdot (w_5 p_3 + w_7 p_4) \end{aligned} \tag{3.7}$$

Тогда используя (2.1) с учетом (3.7) получим:

$$\frac{\partial z}{\partial w_{10}} = f_5' \cdot y_4 = y_4 p_5$$

$$\frac{\partial z}{\partial w_9} = f_5' \cdot y_3 = y_3 p_5$$

$$\frac{\partial z}{\partial w_8} = f_5' \cdot w_{10} f_4' \cdot y_2 = y_2 p_4$$

$$\frac{\partial z}{\partial w_7} = f_5' \cdot w_{10} f_4' \cdot y_1 = y_1 p_4$$

$$\frac{\partial z}{\partial w_6} = f_5' \cdot w_9 f_3' \cdot y_2 = y_2 p_3$$

$$\frac{\partial z}{\partial w_5} = f_5' \cdot w_9 f_3' \cdot y_1 = y_1 p_3$$

$$\frac{\partial z}{\partial w_4} = f_5' \cdot \left(w_9 f_3' \cdot w_6 + w_{10} f_4' \cdot w_8\right) f_2' \cdot x_2 = x_2 p_2$$

$$\frac{\partial z}{\partial w_3} = f_5' \cdot \left(w_9 f_3' \cdot w_6 + w_{10} f_4' \cdot w_8\right) f_2' \cdot x_1 = x_1 p_2$$

$$\frac{\partial z}{\partial w_2} = f_5' \cdot \left(w_9 f_3' \cdot w_5 + w_{10} f_4' \cdot w_7\right) f_1' \cdot x_2 = x_2 p_1$$

$$\frac{\partial z}{\partial w_1} = f_5' \cdot \left(w_9 f_3' \cdot w_5 + w_{10} f_4' \cdot w_7\right) f_1' \cdot x_1 = x_1 p_1 \qquad (3.8)$$

Таким образом, частная производная $\frac{\partial z}{\partial w_i}$ равна произведению входного значения i-го синапса на p нейрона, в который этот синапс приходит. Вычисления происходят справа налево (от последнего слоя сети к первому). Предварительно должно быть запущено прямое функционирование сети, т.е. по входным значениям рассчитаны выходные и сохранены промежуточные результаты

3.3 Философия оптимизации

Для работы нейросетей можно брать любой алгоритм многомерной оптимизации, если он не требует слишком большой памяти. Для работы по этим алгоритмам учителю предоставляется несколько основных процедур. Среди них вычисление значения целевой функции и ее градиента.

При организации алгоритма желательно использовать естественную структуру сети: параметры, градиенты и другие характеристики разбиты на много локальных массивов, действия в которых можно проводить параллельно. Однако, остается неудовлетворенность подходом в целом. Грубо говоря, теория безусловной оптимизации посвящена минимизации квадратичных форм: $H(x) = (x, Qx) + (b, x) + c$. Постоянно делаются попытки уйти как можно дальше от функций, задаваемых квадратичными формами, но все же и метод Ньютона, и метод сопряженных градиентов, и другие усовершенствованные методы спуска создавались применительно к квадратичным формам и тем функциям, которые с их помощью хорошо аппроксимируются.

Идеализированный модельный объект – путеводная звезда теории и ориентация на него неизбежна, какие бы слова при этом ни произносились. Теория выпуклой оптимизации потому и удалась, что выпуклая функция

обладает многими важными свойствами положительно определенной квадратичной формы.

Адаптивный ландшафт нейронной сети вряд ли похож на чашу – график положительной квадратичной формы. Скорее всего, он напоминает ландшафт горного района без особо острых пиков и пропастей.

То же можно сказать и про многие другие функции. Вывод – нужен новый модельный объект, который бы дал другую идеологическую установку для теории оптимизации.

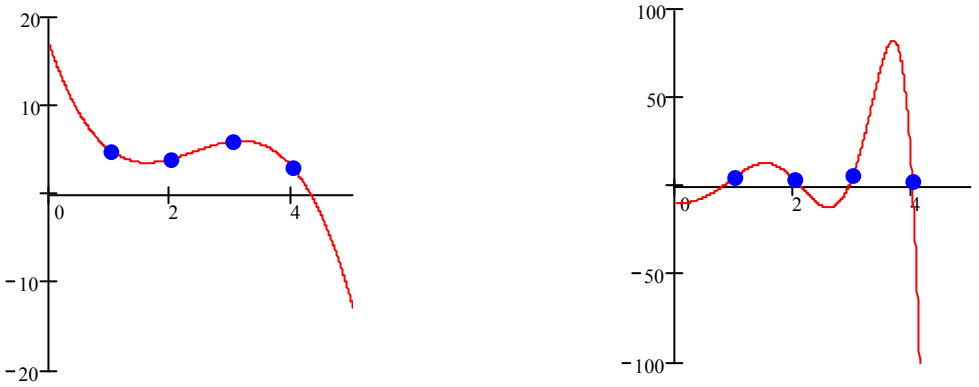
3.4 Определение числа нейронов

Сколько нейронов нужно использовать для конкретной нейросети? По этому вопросу существуют две противоположные точки зрения. Одна из них утверждает, что чем больше нейронов использовать, тем надежнее получится сеть. Сторонники этой позиции ссылаются на пример человеческого мозга. Действительно, чем больше нейронов, тем больше число связей между ними и тем более сложные задачи способна решить нейронная сеть. Кроме того, если использовать заведомо большее число нейронов, чем необходимо для решения задачи, то нейронная сеть точно обучится. Если же начинать с небольшого числа нейронов, то сеть может оказаться неспособной обучиться решению задачи, и весь процесс придется повторять сначала с большим числом нейронов. Эта точка зрения (чем больше – тем лучше) популярна среди разработчиков нейросетевого программного обеспечения. Так, многие из них как одно из основных достоинств своих программ называют возможность использования любого числа нейронов.

Вторая точка зрения опирается на такое «эмпирическое» правило: чем больше подгоночных параметров, тем хуже аппроксимация функции в тех областях, где ее значения были заранее неизвестны. С математической точки зрения задачи обучения нейронных сетей сводятся к продолжению функции, заданной в конечном числе точек на всю область определения. При таком подходе входные данные сети считаются аргументами функции, а ответ сети – значением функции.

Например, если мы имеем значения функции в 4-х различных точках, то при аппроксимации этой функции полиномами 3-й и 8-й степени очевидно, что аппроксимация, полученная с помощью полинома 3-й степени, больше соответствует внутреннему представлению о «правильной аппроксимации» (рисунок 3.1). В этом случае имеем задачу интерполяции, когда число неизвестных параметров полинома равно числу экспериментальных точек. Следовательно, коэффициенты полинома находятся единственным способом. В случае же полинома 8-й степени есть 9

настраиваемых параметров и всего 4 уравнения (по числу экспериментальных точек). Т.е. 5 коэффициентов полинома могут варьироваться произвольным образом, не влияя при этом на ошибку аппроксимации, но заметно искажая вид функции.



а) полином 3-й степени б) полином 8-й степени

Рис. 13. Аппроксимация функции по 4-м точкам

Второй подход определяет нужное число нейронов как минимально необходимое. Основным недостатком является то, что это минимально необходимое число заранее неизвестно, а процедура его определения путем постепенного наращивания числа нейронов весьма трудоемка. Опираясь на опыт группы «Нейрокомп» в области медицинской диагностики, космической навигации и психологии, можно заметить, что во всех подобных задачах ни разу не потребовалось более нескольких десятков нейронов.

Подводя итог анализу двух крайних позиций, можно сказать следующее: сеть с минимальным числом нейронов должна лучше («правильнее», более гладко) аппроксимировать функцию, но выяснение этого минимального числа нейронов требует больших интеллектуальных затрат и экспериментов по обучению сетей. Если число нейронов избыточно, то можно получить результат с первой попытки, но существует риск построить «плохую» аппроксимацию. Истина, как всегда бывает в таких случаях, лежит посередине: нужно выбирать число нейронов большим, чем необходимо, но ненамного. Это можно осуществить путем удвоения числа нейронов в сети после каждой неудачной попытки обучения.

1.6.3 Факторы, влияющие на обучение нейронной сети

Рассмотрим факторы, от которых зависит успешность обучения нейронной сети правильному решению задачи. В первую очередь, сеть должна быть достаточно гибкой, чтобы научиться правильно решать все примеры обучающей выборки. Поэтому в нейронной сети должно быть достаточное количество нейронов и связей.

На основании обучающей выборки достаточно сложно определить, сколько слоев и нейронов сети необходимо. Поэтому поступают обычно так. Обучают сеть со структурой, предлагаемой программой-нейроимитатором по умолчанию, а в дальнейшем, если сеть не может обучиться, пробуют обучить сеть большего размера. На практике при решении разнообразных задач практически не встречается ситуации, когда требуется нейросеть с более чем сотней нейронов – обычно хватает нескольких десятков нейронов и даже меньшего числа.

Однако, даже увеличение размера нейронной сети не поможет, если обучающая выборка противоречива. Иными словами, в обучающей выборке присутствуют задачи с одинаковыми условиями, но разными ответами (одинаковыми входными векторами данных, но разными выходными). Таким задачам нейронная сеть обучиться не может. Здесь возникает проблема разрешения такой противоречивой ситуации. Появление таких конфликтных примеров может, допустим, означать недостаточность набора входных признаков, поскольку при расширении признакового пространства конфликтным примерам могут соответствовать разные значения добавляемого признака и критическая ситуация будет исчерпана. В любом случае пользователь должен решить эту проблему, хотя бы даже простым исключением конфликтных примеров из задачника.

После обучения нейронной сети необходимо провести ее тестирование на тестовой выборке для определения точности решения не входивших в обучающую выборку задач. Точность правильного решения очень сильно зависит от репрезентативности обучающей выборки. Обычно при решении различных неформализованных задач в разных проблемных областях точность в 70-90% правильных ответов на тестовой выборке соответствует проценту правильных ответов при решении этих же задач специалистом-экспертом.

Контрольные вопросы к 1 главе:

1. Нейрон. Понятие, назначение, функционирование. Типы нейронов.
2. Нейросеть. Понятие, назначение, функционирование. Типы нейросетей.
3. Обучение нейросети. Понятие, назначение, функционирование

4. Градиентные и случайные методы. Сходство и различие.

5. Аппаратные и программные реализации нейросетей.

6. Указать тип задач, решаемых с помощью нейросетей.

7. Основные параметры, влияющие на эффективность обучения и работы нейросети.

8. Обучение по задачнику и по отдельному примеру: сравнить достоинства и недостатки.

9. Элементы нейроконструктора

10. Контрастирование: понятие, назначение.

2. Технологии нейросетевой обработки данных

2.1 Описание нейросетевой технологии

Нейросетевые технологии применимы во многих предметных областях, для решения различных прикладных задач. Есть несколько признаков, которыми может или должна обладать задача, чтобы применение нейронных сетей было оправдано:

отсутствует алгоритм или неизвестны принципы решения задач, но накоплено достаточное число примеров;

проблема характеризуется большими объемами входной информации;

данные неполны или избыточны, зашумлены, частично противоречивы.

Нейронные сети для пользователя ПК – инструмент, позволяющий решать задачи на основе накопленного опыта.

Нейроимитаторы – специальные программы для создания, обучения, тестирования и других операций с нейронными сетями. Нейроимитаторы могут быть использованы для обработки результатов различных экспериментов. Основные типы задач, решаемых с помощью нейронных сетей:

1) разделение на классы или классификация, таксономия или кластеризация – если без учителя (сети естественной классификации);
2) предсказание (действительного) числа или предикция (иначе называемая нейросетевой регрессией);
3) распознавание образов;
4) оптимизация;
5) прогнозирование;
6) моделирование.

Дополнительные задачи:

7) прогноз («что будет завтра?»);
8) условный прогноз («что будет завтра, если ..?»);
9) определение значимости входных параметров.

Для решения указанных задач разработан и используется математический аппарат искусственных нейронных сетей, в частности, из сигмоидных нейронов. Архитектуры и алгоритмы позволяют обучать и тестировать нейронные сети, решать различные задачи обработки данных.

2.1.1. Операции предобработки

Выполнение исследования с помощью нейронных сетей можно разделить на несколько групп операций, до некоторой степени независимых друг от друга:

1) подготовительные;

2) обучение нейронных сетей и другие работы с нейронными сетями;

3) поиск смыслов в полученных результатов.

Ниже приведено краткое содержание групп подготовительных операции (методика подготовки данных), название и назначение операций, и комментарии к ним.

1. Сбор данных

В соответствии с постановкой задачи следует составить список параметров (их коррелированность несущественна, в отличие от методов факторного анализа), дать параметрам краткие имена. Для имен лучше всего использовать латинский алфавит, чтобы обеспечить совместимость при обработке разными программами-нейроимитаторами. Определить примерный диапазон значений по каждому параметру, уточнить единицы измерения. Собрать «сырые» данные, если они еще не собраны. Представить данные в виде плоской таблицы, разработать структуру таблицы базы данных (файла данных) (пример – см. табл. 2.1).

Таблица 2.1

Фрагмент структуры базы данных для задачи иммунного плазмафереза

№ п/п	Наименование поля	Расшифровка	Тип	Длина	Кол-во знаков после запятой
1	Ns	Код донора	Numeric	3	
2	Ki	Код манипуляции	Numeric	2	
3	X1m	Месяц	Numeric	4	
4	X1d	День	Numeric	3	
5	X1g	Год	Numeric	4	
6	X2	Гемоглобин	Numeric	3	
7	X3	Эритроциты	Numeric	3	1
8	X4	Ретикулоциты	Numeric	3	1
9	X5	Тромбоциты	Numeric	3	
10	X6	Лейкоциты	Numeric	4	1
11	X7	Эозинофилы	Numeric	2	

12	X8	Палочкоядерные	Numeric	2	
13	X9	Сегментоядерные	Numeric	3	

Архитектура создаваемой базы данных (реляционная, объектно-ориентированная и т. п.) в общем случае не имеет значения, т. к. нейроимитатор работает обычно только с одним входным файлом данных (имеет значение только формат используемого файла данных: большинство нейроимитаторов, разработанных в Красноярске поддерживают форматы Dbase и Paradox). Для реляционных баз данных не требуется даже проводить нормализацию (в терминах теории баз данных). На практике типична обратная ситуация: иногда нужно проводить частичную или полную денормализацию, т. е. сведение информации из разных взаимосвязанных по ключевым полям файлов данных в единый файл (особенно в случае большого числа таблиц). Данная операция относится к предобработке данных и нужна в основном для дальнейшей обработки в нейроимитаторе, так как нейроимитаторы обычно работают только с одним входным файлом данных. Структура конечной таблицы определяется пользователем в зависимости от конкретной задачи.

Разработать информационные классификаторы данных (систему кодирования или рубрикаторы). Для задач классификации используется поле исходной таблицы, содержащее обычно номер класса. Классы представляют собой некоторые совокупности однородных объектов, которые можно сгруппировать по определенным критериям (свойствам). Информационные классификаторы полей данных используют традиционно для задач классификации (в отличие от предикции). Возможно их использование и для задач предикции, что требует дополнительных операций предобработки данных.

На практике некоторым характеристикам объекта, представленным в виде конкретных значений (действительных чисел), можно поставить в соответствие классы (целые числа), определяемые обычно в виде диапазонов изменения значений данного свойства объекта. Например, в медицинских задачах традиционно для практически любого изучаемого параметра существует диапазон его изменения, соответствующий норме, либо уровни его изменения, которые можно представить в виде классов. Для задачи иммунного плазмафереза анализы доноров по значениям титра антител можно сгруппировать в три группы по степени иммунного ответа: низкая (значения титра антител до 6 х103МЕ/л), средняя (значения титра от 6 х103МЕ/л до 12 х103МЕ/л) и высокая (значения титра до 24 х103МЕ/л). Диапазон разбивается на три поддиапазона (как указано). В соответствии с этим разбиением действительные значения титра антител заменяются целыми числами – номерами поддиапазона. Соответственно некоторые

задачи предикции можно представить как задачи классификации (при наличии таких полей). Иногда это может помочь, если нейронные сети не могут обучиться или показывают неудовлетворительные результаты тестирования при использовании нейросетевых методов «в лоб» (без дополнительного преобразования поля, используются значения поля как есть).

При составлении классификаторов желательно монотонное изменение свойств, если это возможно и если известно как. Также важно кодировать одним полем таблицы базы данных один параметр. Создать базу данных, заполнить ее как можно полнее и точнее.

В частности, монотонное изменение свойств предполагает наличие состояний (значений) отдельных параметров объекта, которые изменяются последовательно, например, от меньших значений к большим или наоборот. Приведенный выше пример классификации доноров по степени иммунного ответа полностью соответствует данному определению: низкая, средняя и высокая степени иммунизации. В этом случае числовые наименования классов должны точно отражать порядок следования состояний параметра объекта (числа, определяющие классы, должны увеличиваться или уменьшаться в соответствии с состояниями). Например, 1 – низкая степень иммунизации, 2 – средняя, 3 – высокая. Допустимо использовать обратную последовательность: 1 – высокая, 2 – средняя, 3 – низкая. Для нейронных сетей оба приведенных варианта кодирования признаков эквивалентны и не дают никаких преимуществ (порядок сохраняется). Кроме того, в большинстве случаев можно использовать следующую классификацию: 0 – низкая, 1- средняя, 2 – высокая (или обратный порядок значений: 2 – низкая, 1 – средняя, 0 – высокая). Обычно использование нулевого значения для кодирования никак не мешает, но и не помогает при решении задачи. Данный вариант применяют для удобства предобработки.

Если же не известна степень упорядоченности значений свойства объекта, следует каждое состояние кодировать в отдельном поле (например, 1 – наличие данного состояния, 0 – отсутствие). Например, это характерно для параметров, значения которых содержат цвета (например, цвет глаз или волос, кроме цветов радуги, которые упорядочены по длине волны), иногда дни недели, сезоны года, либо какие-либо профессии (профессиональные группы). В данных примерах не заданы степень и отношения порядка, т. е. нет монотонности изменения свойств. Для задачи иммунного плазмафереза такой подход применен для определения профессиональной принадлежности донора (каждая профессия кодируется отдельным полем), дней недели и сезона года. При этом, естественно, в большинстве случаев существенно увеличивается число полей.

Желательно учесть следующее негласное соглашение: количество записей должно быть больше количества полей (параметров). Обычно при

количестве записей существенно меньшем, чем количество полей наблюдается эффект переобучения: нейронная сеть в этом случае просто запоминает свое обученное состояние. Существует несколько способов избежать этого. Например, можно провести дополнительный сбор данных. Некоторые авторы предлагают добавить в базу данных уже существующие записи. Последний способ не рекомендуется использовать, так как он фактически не решает проблему недостаточности данных. Тем не менее, существуют несколько методов, основанных на данном подходе. Например, для тестирования работы алгоритмов обучения используют выборку, полученную из исходной с добавлением некоторого «шума» – сигнала с известной функцией распределения. Следует заметить, что в некоторых нейроимитаторах при недостаточном количестве данных (примеров выборки) увеличивается вес отдельных классов примеров. Это характерно, например, для нейроимитатора MultiNeuron. Применение данного метода решает скорее проблему несбалансированности количества примеров в разных классах, нежели общего количества примеров. Если нет возможности сбора дополнительных данных, иногда на практике применяют процедуру контрастирования (сокращение числа параметров-полей), которая будет рассмотрена далее (см. п. 2.1.2). В этом случае количество записей остается тем же самым, но удаляются наименее значимые поля, которые можно не учитывать при определении ответа (для нейронных сетей с учителем).

В разных нейроимитаторах существуют различные методы для таких случаев. В нейроимитаторе MultiNeuron, например, удобно использовать «скользящий контроль»: обучение происходит по части выборки, один пример (запись) последовательно выбирается для тестирования, и т. д., пока нейроимитатор не переберет все возможные записи. Строго говоря, «скользящий контроль» можно использовать и в нейроимитаторах NeuroPro v. 0.25 и MDN, но при этом придется предварительно использовать некоторые дополнительные операции по разделению выборок. Использовать данный метод в нейроимитаторе MultiNeuron удобнее.

2. Использование символьных полей

При наличии в исходных данных символьных полей требуется либо отключить их, либо преобразовать в числовые. Обычно используется последовательное кодирование символьных данных целыми числами. При этом имеет значение порядок нумерации. Иногда требуется более сложный подход, например, разбить параметр на 2 или более. Независимо от конкретных вариантов, превращение текстовых данных в числа почти всегда возможно.

Если символьные параметры имеют качественный смысл, такие параметры будем называть качественными, то можно использовать изложенный в работе подход. Качественный параметр может относиться к одному из четырех видов:

1) параметры, обладающие частичным порядком;

2) параметры, обладающие отношением соседства, но не обладающие порядком (для каждого элемента можно указать соседний элемент, но нельзя выделить среди них предшествующий);

3) параметры, не обладающие ни одним из описанных выше отношений (все состояния являются изолированными);

4) параметры, являющиеся комбинацией трех первых видов.

Как сказано ранее, распространена практика присвоения всем состояниям порядковых номеров и подача такого псевдочислового параметра на вход нейронной сети. При этом сети навязывается исходно отсутствующая информация: различным состояниям параметра приписываются отношения соседства и порядка, а также расстояние между всеми состояниями.

Очевидно, что состояния одного и того же параметра могут быть упорядочены различными способами. Так как сигнал кодируется для решения конкретной задачи, то необходимо рассматривать только те отношения порядка и соседства, которые имеют смысл в контексте данной задачи. Чтобы избежать потери информации о параметре и в то же время не наделять параметр не присущими ему свойствами, можно использовать следующий способ подачи качественных признаков.

Качественный параметр кодируется столькими входными сигналами нейронной сети, сколько состояний он может принимать. Для каждого состояния параметра определим понятия верхнего конуса (множество всех состояний предшествующих данному состоянию) и сфер соседства радиуса 1, 2 и т. д. В сферу соседства радиуса один входят все состояния, являющиеся соседом данного и не связанные с ним отношением предшествования. В сферу соседства радиуса R входят все состояния, являющиеся соседями состояний из сферы соседства радиуса R-1, не связанные с данным состоянием отношением предшествования и не принадлежащие сферам соседства меньшего радиуса.

3. Выбор поля ответа

После завершения всех операций кодирования данных следует выбрать поле ответа либо поля ответа для многозадачной таблицы данных или векторного предиктора. Если обучается нейронная сеть с учителем, то нужно выделить поле ответа для непосредственного использования блоком оценки и учителем при обучении. Блок оценки, сравнивая известное заранее значение из поля ответа для текущего примера («истинный» ответ) с вычисленным значением ответа, определяет оценку примера. По оценке примеров определяется оценка выборки и значение обратного сигнала.

Если обучается нейронная сеть без учителя (сети естественной классификации), то нужно выделить поле (поля) ответа для вспомогательных операций либо для отделения этого поля от входных параметров, т. к. поле ответа нельзя использовать для разделения данных в качестве входного

параметра. На начальном этапе исследования поле ответа может отсутствовать, если используются нейронные сети без учителя (естественной классификации). Для нейросетей естественной классификации поле ответа не используется при обучении (если таковое было заранее известно) и в дальнейшем может служить лишь для проверки полученной классификации, известного распределения классов – см. п. 2.1.4.

Для дальнейшего использования нейронных сетей с учителем следует из списка полей выбрать поле ответа. В любом случае поле ответа исключается из списка входных параметров и не подается на вход нейронной сети. Допустимо выбирать несколько полей ответа для обучения векторного предиктора. Кроме того, иногда можно (нужно) выполнить эксперименты с разными полями ответа.

Обучающая выборка должна состоять из примеров всех классов, составляющих классификационную модель. Другими словами, количество примеров любого класса должно быть не меньше 1, хотя количество 1 мало и в некоторых случаях после обработки нейронными сетями без учителя (естественной классификации) данный класс может быть включен в наиболее близкий другой класс (если количество классов больше 2). Количество примеров разных классов может быть неодинаковым, но иметь одинаковый порядок. Желательно, чтобы для любых i, j, где $i \neq j$ (i, j – номера классов), из посылки, что $Ni > Nj$, следует $Ni < kNj$ (где Ni – количество примеров i класса), k может быть около 2. В некоторых нейроимитаторах (например MultiNeuron) можно примерам (классам) назначать вес.

4. Поиск и заполнение пробелов в данных

При наличии пробелов в данных следует заполнить их. Простейшие способы заполнения пробелов в данных по степени предпочтительности их использования:

средним по классу, если класс известен;

средним по полю;

нулем;

минимальным, максимальным.

Выбор лучшего из этих способов - в зависимости от цены ошибки или из других соображений. Для заполнения пробелов можно использовать программы FAMaster, Predmake и другие.

Данная операция обычно желательна, но может отсутствовать при условии заранее известной полноты данных. Можно интерпретировать операцию заполнения пробелов как операцию модификации данных даже при исходной заполненности. В данном случае модифицированный набор данных будет тождественно равен входному.

Заметим, что значения полей равные нулю в общем случае не являются пробелами. Нулевые значения не являются пустыми. Обычно это не вызывает проблем, но иногда при операциях чтения или сохранения пустоты

в числовых полях автоматически заполняются нулевыми (или иными) значениями в некоторых нейроимитаторах. Это следует учитывать, т. к. данная настройка может существенно повлиять на результаты обработки данных. Некоторые нейроимитаторы автоматически заполняют пустоты некоторыми константами (например, MultiNeuron заполняет все пробелы значением «-1») или расчетными значениями по некоторым правилам. Это следует учитывать при работе с данными, содержащими пробелы.

5. Определение простых статистических параметров: среднее, среднее квадратичное отклонение, парные корреляции, корреляционное отношение.

Проверка гипотезы о нормальном распределении данных выборки, т. к. для данных, собранных в реальных условиях, распределение далеко не всегда нормально. Кроме того, данных часто недостаточно для обоснования нормальности распределения.

6. Нормирование данных. Обычно выполняется нейроимитатором перед началом работы. Часто используемые способы нормирования:

на интервал [0,1] линейно;

на интервал [0,1] кусочно линейно [-ks, +ks], где s есть выборочное среднее квадратичное отклонение, k – число сигм (обычно 1, 2 или 3 в соответствии с принятыми в статистике), на левую границу отображается все, что левее, на правую границу – все, что правее;

на интервал [0,1] нестрого, но линейно [-ks, +ks], т. е. результатом нормирования будет диапазон [a, b], где $a < 0$, $b > 1$, прямая отображается в прямую, но интервал [-ks, +ks] отображается в [0, 1];

на интервал [0,1] нелинейно, с использованием, например, сигмоидного нормирования;

на сферу единичного радиуса (каждый вектор делится на свою длину).

Способ нормирования может быть изменен после проведения ряда экспериментов.

В конечном итоге создается несколько нейросетей со сходной структурой, где указываются входные, выходные параметры, значение отклонения (точности) значения выходной величины (для задач предикции) либо значения выходной величины для каждого класса (задачи классификации), число слоев, число нейронов в каждом слое.

2.1.2. Описание технологии использования нейронных сетей с учителем

Данные с известным ответом можно обрабатывать с помощью нейронных сетей с учителем. Учитель – блок, который обучает нейронную сеть давать правильный ответ по обучающей выборке. Классификация нейронными сетями с учителем обычно выполняется эффективней, т. к. при этом для обучения используется дополнительная информация - номер класса.

Однако, если номер класса заранее неизвестен (часто бывает неизвестно и число классов), то такие данные без известного ответа с помощью нейронных сетей с учителем обработать невозможно. Для обработки таких данных используют более простой инструмент – нейронные сети естественной классификации (unsupervised по зарубежной классификации нейронных сетей), также называемые самоорганизующиеся сети. Они работают в случае отсутствия классификационной модели, что часто бывает на начальном этапе любого исследования. Нейронные сети без учителя (естественной классификации) предназначены для построения классификационной модели, а также позволяют определять значимости входных сигналов и минимизировать их количество.

Из всех типов задач, решаемых на основе накопленного ранее опыта, значительную группу составляют задачи классификации. Ответом для них является указание класса – выбор одного из нескольких возможных вариантов решения. Обученная сеть по входным данным, аналогичным по структуре обучающей выборке, должна быть способна вычислять определенный ответ (класс).

В настоящее время основной для нейроимитаторов на ПК называют задачу «извлечение знаний из данных». Явное знание – то, что может быть передано от знающего субъекта к познающему в ходе обычного акта общения (коммуникации) между субъектами. Явное знание зависит от социально-культурной среды, традиций, контекста общения.

Нейронная сеть делится с нами знаниями, только если мы ее этому научили. Основные способы получения явного знания от нейронной сети (с учителем):

1) определение значимости входных параметров;

2) контрастирование нейронной сети или получение логически прозрачной нейронной сети;

3) получение вербального описания сети;

4) тестирование обученной сети;

5) обучение примера.

Рассматривая нейронную сеть как набор элементов, производящих некоторые вычисления над приходящими к ним данными, можно изучать решение главной и вспомогательных задач. В главной задаче выделяют функции или подзадачи: генерации, обучения и тестирования нейронных сетей. Генерация – задание начальных значений всех параметров нейронной сети; обучение – изменение всех или части параметров нейронной сети с учителем так, чтобы нейронная сеть по обучающей выборке давала требуемые ответы; тестирование – сравнение ответов обученной нейронной сети с известными ответами по тестовой выборке – набору примеров, не использовавшихся для обучения. Иными словами, тестирование – проверка качества обучения сети. Вспомогательные задачи: определение значимости

входных сигналов, контрастирование нейронных сетей, обучение примера, определение минимального решающего набора входных параметров, получение логически прозрачной нейронной сети и, в конечном счете, знаний из данных. Обучение примера – такое изменение параметров примера, чтобы обученная нейронная сеть при тестировании давала требуемый ответ. Вспомогательные задачи сформулированы в процессе развития и практического использования нейронных сетей, как новые потребности пользователей и направления развития. Именно вспомогательные задачи чаще всего есть основной элемент технологии решения прикладных задач пользователя.

Обучение нейронных сетей с учителем может быть выполнено с помощью программ Clab [С. Е. Гилев], MultiNeuron [А. А. Россиев, Е. М. Миркес, С. Е. Гилев и др.], NeuroPro [В. Г. Царегородцев], Monoton и многих других. Кроме того, в настоящее время в некоторых пакетах статистики и в электронных таблицах предусмотрены дополнительные модули для работы с нейронными сетями. Например, продукты серии Excel Neural Package (Winnet, Kohonen map), реализованные как набор надстроек над Microsoft Excel.

Следует отметить, что некоторые нейроимитаторы поддерживают собственные внутренние форматы, как входных, так и выходных данных, либо же предполагают их определенную структуру зачастую не стандартизованную и строго уникальную. Это сильно затрудняет работу с подобными нейроимитаторами и не позволяет эффективно использовать другие программные продукты для обработки полученных результатов (проблема совместимости форматов).

Однако справедливо и то, что существуют нейроимитаторы, поддерживающие общепринятые форматы данных, либо позволяющие конвертировать (для выходных данных) свой внутренний формат в один из общепринятых. Например, статистические пакеты, позволяющие реализовать нейронные сети, поддерживают в основном форматы xls и некоторые форматы баз данных (Dbase, Paradox), результаты обработки можно передать офисным пакетам (Microsoft Excel, Word) или сохранить во внутренним формате. Нейроимитаторы, разработанные в г. Красноярске (MultiNeuron, NeuroPro, MDN), в качестве входных данных обычно поддерживают форматы Dbase, Paradox (файлы dbf, db), а результаты чаще всего легко конвертируются в текстовый формат (txt). При использовании Microsoft Excel таблицу данных формата xls можно легко конвертировать в формат Dbase. При этом данные должны быть оформлены как таблица и в первой строке содержать наименования полей (желательно использование латинских букв во избежание проблем совместимости между версиями). Входные и выходные поля во входных файлах данных должны быть числового типа,

примерная последовательность преобразования полей приведена при описании предобработки (см. п. 2.1.1).

Ниже приведено краткое описание групп основных операций для нейросетевой технологии с учителем, их название, назначение и комментарии к ним.

1. Обучение нейронной сети по полной выборке. Нейронная сеть с учителем, обученная по полной выборке, свидетельствует о простейшей непротиворечивости данных, отсутствии одинаковых примеров с разным ответом, а также групп примеров с более сложной структурой противоречия (рис. 2.1).

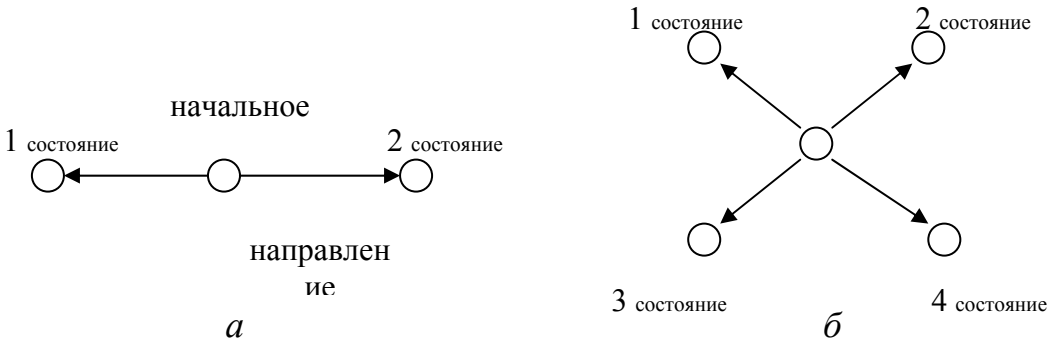


Рис. 14. Графическое представление противоречий в данных:

а – пара одинаковых примеров с разным ответом;
б – группы примеров с противоречием, класс i – состояние, в котором правильно определяется пример i, стрелками показаны направления вектора градиента функции

Рассмотрим структуру противоречия на примере биологической задачи классификации ирисов Фишера. Противоречивость данных может появиться после удаления одного или нескольких входных параметров, даже если исходно она была непротиворечива (табл. 2.2). Можно получить противоречие при дополнительном наборе данных, когда объем выборки возрастает и могут появиться противоречивые примеры. Противоречивость может быть получена в исходной таблице данных, если исследователем не учтен один или несколько факторов (входных полей), существенно влияющих на изучаемый процесс в рассматриваемой области знаний. Иногда признак противоречия данных есть признак неполноты или недостаточности классификационной модели. Явного противоречия в этом случае исходно не наблюдается, но может быть выявлено после детального анализа содержимого каждого из классов (рассматривается каждый пример и сравнивается с соответствующей классификацией: особо изучаются примеры с ошибочно определенной классификацией при тестировании).

Таблица 2.2. – Пример получения противоречивых наборов входных данных при сокращении длины описания (числа полей) на примере ирисов Фишера

CLASS	DL_ CHASH	SH_ CHASH	DL_ LEP	SH_ LEP
1	5,1	2,2	1,4	1,5
1	4,6	3,1	1,5	0,2
2	5,5	2,3	4,0	1,3
2	6,5	2,8	4,6	1,5
2	5,7	2,8	4,5	1,3
2	6,2	2,2	4,5	1,5
2	5,6	2,5	3,9	1,1
2	5,9	3,2	4,8	1,8
2	6,1	2,8	4,0	1,3
3	6,0	2,2	5,0	1,5
3	6,9	3,2	5,7	2,3
3	5,6	2,8	4,9	2,0
3	7,2	3,2	6,0	1,8
3	6,2	2,8	4,8	1,8
3	6,3	2,8	5,1	1,5
3	6,1	2,6	5,6	1,4

CLASS	SH_ CHASH	SH_ LEP
1	**2,2**	**1,5**
1	3,1	0,2
2	2,3	1,3
2	**2,8**	**1,5**
2	2,8	1,3
2	**2,2**	**1,5**
2	2,5	1,1
2	**3,2**	**1,8**
2	2,8	1,3
3	**2,2**	**1,5**
3	3,2	2,3
3	2,8	2,0
3	**3,2**	**1,8**
3	2,8	1,8
3	**2,8**	**1,5**
3	2,6	1,4

Следует помнить, что все алгоритмы обучения сетей методом обратного распространения ошибки (общепринятый метод, используемый при обучении с учителем) опираются на способность сети вычислять градиент функции ошибки по обучающим параметрам. Таким образом, обучение состоит в вычислении градиента и модификации параметров сети. Тем не менее, существует множество не градиентных методов обучения, например, метод покоординатного спуска, случайного поиска и ряд методов

Монте-Карло. Данные методы обычно менее эффективны на «хороших» (непротиворечивых, непустых) данных, чем градиентные. С другой стороны, направленные или градиентные методы в чистом виде иногда вообще не дают решения на данных с множеством экстремумов (на данных с локальными минимумами, «ямами»).

Некоторые часто используемые методы обучения:

1) неградиентные методы:

метод случайной стрельбы (один из методов Монте-Карло);

метод покоординатного спуска (псевдоградиентный);

метод случайного поиска;

метод Нелдера-Мида;

2) градиентные методы:

метод наискорейшего спуска;

kParTan;

квазиньютоновские методы (BFGS).

2. Прогнозирование. Для получения прогноза нужно обучить нейронную сеть по обучающей выборке – подмножеству полной выборки, тестировать сеть по тестовой выборке – подмножеству полной выборки. Количество правильных ответов по тестовой выборке (процент от количества примеров тестовой выборки) показывает, насколько «хороши» данные – полны, непротиворечивы, точны и т. д. Одновременно получаем качество обучения сети. На практике качество обучения сети считается удовлетворительным, если процент правильно определенных примеров составляет более 50 %. Обосновать это значение можно со статистической точки зрения – при наличии обученной нейронной сети или другого устройства, которое будет правильно решать задачу чаще, чем ошибаться, мы можем с определенной достоверностью правильно предсказать все что угодно лучше, чем при использовании случайных методов поиска (типа бросания монетки). Соответственно, чем выше этот показатель, тем лучше обучилась сеть.

Если процент правильно определенных примеров составил 100 %, следует проверить наличие во входных полях параметра (параметров), сильно связанных с полем ответа. Характер этой связи в общем случае может быть любым, вплоть до линейной зависимости. На практике обычно обсуждается избыточность линейной связи (линейно связанные параметры рекомендуется не использовать при статистическом анализе). Обнаружение тесной нелинейной связи – одна из традиционных задач, где применяют нейронные сети. В этом случае обычно не наблюдается 100% правильно определенных примеров. Например, для задачи иммунного плазмафереза между большинством параметров наблюдается нелинейная связь (наиболее тесная между параметрами, составляющими лейкоцитарную формулу), что

подтверждается данными корреляционного анализа и многочисленными экспериментами с использованием нейронных сетей с учителем.

Существует вторая причина наличия 100 % правильно определенных примеров: тестирование по той же выборке, по которой обучали сеть, либо при использовании обучающей и тестовой выборок, полученных на основе общей совокупности, обучение по одной из выборок (обучающей или тестовой) или последовательное дообучение (при обучении используются две выборки), тестирование по общей выборке. Запишем описанные варианты использования выборок в терминах теории множеств. Пусть ОВ – обучающая выборка, ТВ – тестовая выборка, В – общая выборка (ТВ и ОВ получены из В), или ТВ$\subset$В, ОВ$\subset$В, ТВ$\cup$ОВ=В. Эффект 100 % правильно определенных примеров при тестировании наблюдается, например, если ТВ=ОВ (выборки одинаковые, т. е. содержат одинаковые записи) или ТВ $\cap$ ОВ $\neq \varnothing$. В последнем случае тестирование может не дать 100 % правильно определенных примеров, но методологически ошибка здесь присутствует в любом случае.

Эффект переобучения (overfitting) можно предположить, если качество тестирования сети низкое или очень низкое (процент правильно определенных примеров меньше 50 %). При этом наблюдается низкая ошибка обучения и высокая ошибка генерализации (обобщения). Данный эффект обычно связан с избыточной структурой сети по сравнению с используемой выборкой. Сеть скорее «зазубривает» примеры обучающей выборки, чем учится по ним. На практике это иногда случается, т. к. не существует общепринятых априорных методов определения исходной структуры нейронной сети, требуемой для решения конкретной задачи. Иногда переобучение может быть результатом обучения с малым (нулевым) значением допустимого отклонения при оценке правильности примеров. Последнее приводит к продолжительной «подгонке» ответов сети под точные значения ответов в обучающей выборке. Как следствие – снижение правильности ответов по генеральной совокупности.

Во избежание эффекта переобучения для малых выборок применяется «скользящий контроль» – обучение по всем примерам выборки без одного, тестирование по одному, где тестовым примером служит один пример выборки, по очереди. Для выборок среднего размера и больших часто используют обучение по четным примерам, тестирование по нечетным и наоборот. При использовании нейронных сетей выборка считается малой, если количество примеров имеет порядок количества полей или меньше.

Часто тестирование разделяют на 2 задачи: тестирование для данных с известным ответом (test) позволяет оценить качество обученной нейронной сети; тестирование для данных без ответа (validation) позволяет получить прогноз либо заполнить пропуск в данных. Последнее (данные без ответа)

можно назвать решением задачи прогнозирования. Термин «тестирование» используется как жаргонное выражение, характерное для группы NeuroComp.

3. Определение значимости входных параметров. Анализ значимости – определение относительно более информативных параметров, определение параметров, зависящих друг от друга и степени этой зависимости.

Значимость входного сигнала или параметра - относительная оценка важности данного сигнала или параметра сети для получения заданных выходных сигналов сети в сравнении с аналогичными показателями значимости для других сигналов, полученных одновременно или в сходной серии экспериментов.

Для выбора наиболее значимых параметров используем ранжирование значимостей по серии нейросетей. С точки зрения статистики рангом наблюдения называют тот номер, который получит это наблюдение в упорядоченной совокупности всех данных – после их упорядочения по определенному правилу (например, от меньших значений к большим или наоборот). Будем упорядочение чисел (значимостей) производить по величине – от больших к меньшим. Ранги применяют, если имеющие в распоряжении числовые данные (например, значения элементов выборки) носят в той или иной мере условный характер. Анализ таких данных требует особой осторожности, поскольку многие предпосылки классических статистических методов (например, предположение о нормальности закона распределения) для них не выполняются. В этом случае имеет смысл вообще отказаться от анализа конкретных значений данных, а исследовать только информацию об их взаимной упорядоченности.

Можно предложить использовать следующую методику для определения рангов по серии нейросетей: значения значимостей входных параметров упорядочиваем по убыванию для каждой сети. Расставляем ранг для каждого значения, начиная с максимального. Одинаковые значения значимостей (с учетом округления) имеют одинаковые ранги. Затем для каждого параметра определяем суммарный ранг для серии сетей, который рассчитывается как сумма рангов каждой сети. Упорядочиваем параметры в соответствии с порядком возрастания суммарного ранга. Итоговый ранг по серии нейросетей проставляется в соответствии с возрастанием суммарного ранга. При определении значимостей и рангов после контрастирования внешней структуры (входных параметров) можно использовать аналогичную методику.

Следует отметить, что ранги, в отличие от средних значимостей параметров, более адекватно использовать при анализе влияния параметров. В большинстве случаев распределение параметров по рангам соответствует средним значимостям параметров (сохраняется порядок следования параметров по рангам и по значимостям), но иногда отличаются. Среднее значение может быть небольшим (значимости усредняются по всем сетям),

тем не менее, ранг показывает большую значимость данного параметра. Ранг точнее определяет порядок следования параметров по степени их влияния на выходное поле.

Существует несколько способов определения значимостей параметров для нейронных сетей с учителем: по сумме проходящих сигналов, по сумме их модулей, по сумме квадратов модулей и т. п.

Отметим, что данные показатели, как правило, нет смысла определять, если при тестировании сеть показывает менее 50 % правильно решенных примеров, т. к. в этом случае влияние параметров слабое или практически отсутствует. Тем не менее возможно конструирование более тонких настроек сети.

4. Контрастирование нейронных сетей (pruning, contrasting), минимизация числа входных параметров. Проведение серии экспериментов по минимизации входных параметров позволяет определить набор ядерных параметров, которые остаются всегда или почти всегда (в 80 % сетей). Остальные параметры вырезаются в большинстве сетей.

Контрастирование – процесс минимизации числа связей нейронной сети, удаление избыточных связей, без которых решаемая задача продолжает решаться так же верно (с той же точностью). Частный случай контрастирования – минимизация числа входных сигналов. Минимизация числа входных сигналов – это сокращение их числа до минимального набора, по которому можно получить ответ на данной обучающей выборке.

Контрастирование может включать в себя следующие операции:

сокращение числа входных параметров;

сокращение числа нейронов сети;

сокращение числа синапсов сети;

сокращение числа неоднородных (пороговых) входов нейронов сети;

равномерное упрощение сети так, чтобы на каждый нейрон сети приходило не более n сигналов.

Данные операции могут привести к сокращению целого слоя нейронов (в некоторых нейроимитаторах существует ограничение, согласно которого слой не может быть удален). Реализация контрастирования слоев предполагает наличие минимального количества нейронов в слое (слой из одного нейрона), после удаления которого синапсы, приходящие ранее на его вход, направляются на вход каждого из нейронов следующего слоя. Процесс контрастирования после этого продолжается и излишние связи сокращаются.

Решение задачи контрастирования позволяет получить разреженную, логически прозрачную нейронную сеть, с более простой структурой, понятной пользователю. Далее можно перейти к получению знаний из данных – построить полуэмпирическую теорию или набор полуэмпирических теорий изучаемого явления. Однако для решения задачи контрастирования с помощью большинства видов нейронных сетей, в том

числе сигмоидных, требуется выполнить дополнительные вычисления, даже более трудоемкие, чем обучение нейронной сети. Для большинства используемых архитектур и методов обучения нейронных сетей определение значимости и контрастирование – специальные операции, которые выполняют после обучения сети, причем контрастирование обычно требует многократного дообучения и определения значимости входных параметров и синаптических связей.

У контрастированных сетей, кроме простой структуры и логической прозрачности, есть и другие преимущества. Контрастирование ускоряет срабатывание обученной нейронной сети в любых нейронных системах. Кроме того, очень важен вопрос технической реализации обученной нейронной сети, т. к. контрастированная система реализуется существенно проще, что связано с меньшим числом синапсов и нейронов, т. е. может быть меньше размером и дешевле.

Существуют алгоритмы и методы обучения, которые выполняют контрастирование в ходе первоначального обучения. Это означает использование двух целевых функций или специальной целевой функции, которая усложнена требованием контрастирования. Однако трудно соблюсти баланс двух процедур минимизации, поэтому такие алгоритмы и методы существенно более сложны, но менее эффективны. Известно исключение – в монотонных нейронных сетях наблюдается частичное контрастирование в ходе первоначального обучения без применения специальных алгоритмов.

5. Построение логически прозрачной нейронной сети, с использованием контрастирования как метода. Прозрачная сеть может быть построена из слоистой. Причем реализация данного метода может быть существенно различной для разных нейроимитаторов и определяется исключительно разработчиком. Например, в нейроимитаторе NeuroPro v. 0.25 (разработчик В. Г. Царегородцев) при контрастировании слоистая структура сохраняется, т. е. число слоев остается неизменным (слои не удаляются) и каждый слой обязательно содержит хотя бы один нейрон.

В общем случае процесс построения логически прозрачной сети (ЛПС) сводится к упрощению сети, которое обычно включает в себя: операции контрастирования и бинаризации весов синапсов, неоднородных входов сети (бинаризованные синапсы и неоднородные входы в NeuroPro далее не обучаются). Затем получают вербальное описание нейронных сетей, где перечисляются используемые поля файла данных, правила их предобработки для подачи сети, описание нелинейных функций нейронов, функционирование нейронной сети послойно и понейронно, правила нормировки выходных сигналов сети в диапазон истинных значений. Сигналам, генерируемым нейронами сети, присваиваются некоторые имена и в дальнейшем при анализе сети можно именовать эти сигналы в терминах предметной области. Далее, получив вербализованное описание нейронной

сети, можно попытаться восстановить правила, сформированные сетью для решения задачи – записать на естественном языке алгоритм решения неформализованной задачи предсказания или классификации.

Приведем критерии, на основе которых можно строить некоторые оценки приближения к идеальному логически прозрачному состоянию сети:

1. Чем меньше слоев нейронов в сети, тем сеть более логически прозрачна. Однако число слоев зафиксировано при создании сети. Этот критерий важен в тех ситуациях, когда число слоев сети избыточно и после контрастирования возникают отдельные нейроны (или слои нейронов), являющиеся просто передатчиками информации с предыдущего слоя на следующий. Такие нейроны имеют единственный вход (и могут иметь также и единственный выход). Эти нейроны должны по возможности заменяться линиями передачи информации. Однако часто не удается полностью избавиться от излишних слоев и поэтому можно ввести окончательную формулировку критерия логической прозрачности – *чем меньше нейронов в каждом из имеющихся путей прохождения сигналов в сети от входа к выходу, тем лучше*.

2. *Чем меньше число нейронов в каждом слое сети, тем лучше*. Этот факт не требует пояснения – мы минимизируем число сигналов, генерируемых каждым слоем сети, что позволяет оставить только действительно значимые промежуточные сигналы (признаки).

3. *Чем меньше входных сигналов сети, тем лучше*. Этот критерий отсеивает признаковое «шумовое поле», оставляя минимально необходимый для правильного решения задачи набор наиболее значимых входных признаков.

4. *Чем меньше число приходящих на нейрон сигналов, тем лучше.* Это утверждение опирается на тот факт, что человек может одновременно оперировать с достаточно малым числом сущностей. Минимизируя число приходящих на нейрон сигналов, мы облегчаем пользователю задачу содержательного осмысления признака, генерируемого нейроном, и, может быть, помогаем в именовании этого признака. Однако желательна модификация этого критерия и введение критерия равномерной простоты сети – *на каждый нейрон сети должно приходить не более* ***n*** *сигналов, где* ***n*** *достаточно мало* (2 – 3, но может задаваться и пользователем).

5. *Чем меньше общее число синапсов в сети, тем лучше.* Критерий ликвидирует все излишние синапсы нейронной сети.

6. *Синапс, по которому передается сигнал, логически непрозрачнее неоднородного входа нейрона.* Действительно, неоднородный вход нейрона – это просто константа, в отличие от синапса, выполняющего умножение своего веса на величину поступающего на синапс сигнала. Поэтому в первую очередь из сети должны исключаться синапсы, а только потом – неоднородные входы адаптивных сумматоров.

7. *Необходимо приведение значений настраиваемых параметров сети к конечному набору выделенных значений.* Желательна бинаризация параметров сети – приведение весов синапсов к значениям -1 и 1.

При решении прикладных задач логически прозрачные сети получают крайне редко, почти никогда. Поэтому имеет смысл различать виды или степени логической прозрачности:

1) строгая – с соблюдением всех требований (критерии 1 – 7);

2) слабо логически прозрачные сети – соблюдаются только некоторые из критериев со слабой степенью соответствия.

6. Бинаризация нейронной сети – такое обучение сети, что большая часть синаптических весов принимает значения из множества {-1, 0, 1}. Существуют разные виды бинаризации, которые предусматривают различные конечные множества значений весов синапсов. Например, в нейроимитаторе NeuroPro v. 0.25 реализованы следующие виды бинаризации:

приведение весов синапсов к значениям из множества {-1, 1};

приведение весов синапсов к значениям из множества {-1, -0.5, 0.5, 1};

приведение весов синапсов к значениям из множества {-1, -0.75, -0.5, -0.25, 0.25, 0.5, 0.75, 1};

приведение весов синапсов к значениям из множества {-1, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}.

Отметим, что в NeuroPro бинаризация не предусматривает приведение веса синапса к нулевому значению, что фактически означало бы удаление синапса. Бинаризация в данном случае не упрощает внутреннюю структуру сети (контрастирования не происходит).

Изначально бинаризация создавалась для упрощения реализации нейронной сети на цифровых ЭВМ, что подразумевает приведение действительных значений весов синапсов к значениям из множества, которые удобно представить в двоичном коде. Это позволяет ЭВМ быстрее и точнее обрабатывать бинаризованные веса синапсов при расчетах. Если рассматривать виды бинаризации, реализованных в NeuroPro, то первые три вида легко представить в двоичном коде, четвертый же вид (кроме отдельных значений) теоретически не представим в двоичной системе счисления, поэтому округляется до некоторой точности.

7. Обучение по исходной выборке на предсказание одного из входных параметров или решение одной из обратных задач (т. е. выбор другого поля ответа).

8. Построение множества минимальных безизбыточных наборов входных параметров (для устранения дублирования 1-го рода). Минимальный безизбыточный набор входных параметров – набор, позволяющий обучить нейронную сеть для получения ответа. Причем имеется в виду обучение нескольких нейронных сетей для выбранного поля

ответа (поле ответа – одно и то же для всех нейросетей). После чего каждая нейронная сеть последовательно упрощается, но не в смысле упрощения внутренней структуры (удаления синапсов, нейронов, неоднородных входов), а идет только сокращение числа входных параметров (в NeuroPro этот процесс автоматически включает дообучение сети). Для построения минимального безизбыточного набора используется операция контрастирования.

9. По каждому входному параметру строим набор параметров, который его однозначно определяет (для устранения дублирования 2-го рода). Для этого выполняется обучение нейронных сетей с выбором в качестве поля ответа каждого из входных параметров, по очереди. Таким образом, проводятся эксперименты с последовательной сменой поля ответа, начиная с самого первого входного параметра (поля) и заканчивая последним. Затем определяется минимальный безизбыточный набор входных параметров, аналогично предыдущему пункту.

Таким образом, дублирование 1-го рода есть частный случай дублирования 2-го рода. Кроме того, методы выявления и устранения дублирования 2-го рода есть развитие п. 7.

10. Обучение примера с использованием обученной нейронной сети – выяснение минимальных изменений в параметрах примера для получения нужного вычисленного поля ответа.

11. Поиск смыслов. Поиск в результатах устойчивых, повторяющихся сочетаний («сущностей»), возникающих в сходных условиях. Именование устойчивых, повторяющихся сочетаний в данных («сущностей») некоторым уникальным термином. Проверка, что есть эта «сущность» – общее, особенное или единичное (частное) (проверка – знание ли это?). Сравнение смысла термина с близкими ему подобиями (выяснение – новое ли это знание или оно уже существует, т. е. известно ранее). К сожалению, операция поиска смыслов имеет специфический характер и для каждой задачи выполнение этого пункта несколько отличается от других. Соответственно, эта операция является самой сложной и, в некотором смысле, интеллектуальной. Поиск смыслов обычно проводится совместно со специалистами в предметной области.

Используя описанный технологический порядок операций, можно получить практические результаты для любых данных прикладной области. Применяя дополнительные инструменты обработки данных, например, аппарат математической статистики, проверяют адекватность полученных результатов.

2.2 Формальный язык и грамматика описания нейросетевой технологии

В данном разделе предлагается модель нейросетевой технологии использования нейронных сетей с учителем, построенная на основе формальных языков и грамматик. Описывается технология использования нейронных сетей для решения прикладных задач на уровне пользователя, т. е. без детализации используемых операций. Операции предполагаются уже реализованными в идеальном нейроимитаторе.

При построении модели используем четыре уровня детализации технологии нейросетевой обработки данных (нумерация начинается с 0):

0) группы операций – предварительные, основные и вспомогательные операции;

1) подгруппы операций – неэлементарные составляющие (которые можно разложить на более простые операции) групп операций 0 уровня, например, нормирование, предобработка и т.п.;

2) операции – операции, используемые для нейросетевой обработки данных, рассматриваемые в самом общем виде без учета конкретной реализации;

3) реализация операций – рассмотрение конкретной реализации или алгоритма выполнения (на уровне компонент сети) операций. Этот уровень предусматривает участие квалифицированного пользователя или разработчика нейроимитатора (другой подобной программной системы).

В дальнейшем для построения всех приведенных технологических операций будем рассматривать три группы операций, скомпонованные в соответствии с описанием нейросетевой технологии обработки данных для нейронных сетей с учителем (см. п. 2.1.1, 2.1.2):

1. Предварительные операции – считывание файлов данных, предобработка, представление данных.

1.1. Обязательные операции – считывание файлов данных, предобработка.

1.2. Необязательные операции - представление данных.

2. Основные – создание сети, чтение сети, обучение.

3. Вспомогательные – тестирование, упрощение структуры сети, определение значимостей входных параметров, вербализация сети, сохранение проекта (сети). (Название «вспомогательные» дано не по их степени важности, а по месту – порядку выполнения – в технологии по сравнению с основными. В ходе обработки данных обязательным является выполнение основных и не обязательным – любой из вспомогательных операций.)

Следует отметить, что предварительные операции реализуются обычно в специализированных программах – предобработчиках или других

подходящих пакетах. Существует несколько подобных программ: FAMaster (выполняет операции заполнения пропущенных данных, разработчик А. А. Россиев), PredMake (разработчик Л.А. Жуков), предобработчик А. А. Батуро, LingStat (предназначен для предобработки текстов, разработчик Л. А. Жуков), Jointer (разработчик Н. В. Решетникова). Имеются и интегрированные пакеты, реализующие сразу несколько подходов для обработки данных, например, продукты серии Excel Neural Package (Winnet, Kohonen map), реализованные как набор надстроек над Microsoft Excel, дополнительный пакет для работы с нейронными сетями в статистическом продукте Statistica.

Некоторые предварительные операции, в частности считывания файлов данных предусмотрены во всех используемых пакетах (в предобработчиках, нейроимитаторах и интегрированных системах), заполнение пробелов в данных может выполняться как в предобработчиках, так и в некоторых нейроимитаторах. Некоторые нейроимитаторы (не все) автоматически после загрузки данных проводят их нормирование.

Приведем схематично группировку указанных операций по месту их выполнения (идеальный случай, т. е. здесь и далее описана идеальная ситуация реализации операции – желаемое, но не всегда действительное).

Таблица 3.1. – Группы операций по месту выполнения

	Предварительные	Основные	Вспомогательные
Предобработчик	+	-	-
Статистические пакеты	Частично	-	-
Нейроимитатор	Частично	+	+ / -
Интегрированные системы	+ / Частично	+	+ / -

Таблица 3.2. – Описание некоторых нейроимитаторов (выполняемые операции)

	Neuro Office	Neuro Iterator	GA	Neural Plan-ner	MultiN euron	Evolu-tion	Neuro Pro
Чтение файла данных	+	+	+	+	+	+	+
Формат входных данных	Text	Text	Text	Text	Dbase/ Para-dox	Text	Dbase, Para-dox
Редактирование данных	+	+	+	+	+	+	+
Заполнение пробелов	-	-	-	-	- (“-1”)	-	-
Нормирование данных	+	+	+	+	+	+	+
Преобразование полей	-	-	-	-	-	-	-
Обучение	+	+	+	+	+	+	+
Тестирование по выборке	+	+	+	+	+	+	+
Тестирование по примеру	-	-	-	-	+	+	-
Контрастирование внешней структуры	-	-	-	-	+	-	+
Контрастирование внутренней структуры	-	-	-	-	-	-	+
Бинаризация	-	-	-	-	-	-	+
Вербализация	-	-	-	-	+	-	+
Формат выходных данных	Text	Text	Text	Text	Text	Text	Text
Хранение сетей	+	+	+	+	+	+	+
Значимости входных параметров	+	+	+	+	+	+	+

В действительности же группировка операций не является такой очевидной и не все операции из приведенных групп реализуются в полном объеме. В большинстве случаев каждый автор соответствующего программного пакета сам определяет и трактует необходимый набор операций для реализации. Поэтому разумнее в данном контексте привести более детализованную классификацию как по операциям, так и по отдельным программным продуктам (табл. 3.2).

В целом основные и некоторые из предварительных операций являются обязательными для выполнения, а вспомогательные являются необязательными (дополнительными) и могут не выполняться при обработке данных. В данной работе рассматриваются возможные последовательности действий при обработке данных, выполняемые для одной нейросети (одного проекта). В каком-то смысле здесь рассматривается жизненный цикл одной нейронной сети (проекта) и результаты обработки данных с ее помощью. Из этого следует, например, что создание нейронной сети обязательно присутствует в слове рассматриваемого формального языка один и ровно один раз. Аналогичным образом можно рассматривать данное утверждение в терминах использования одного проекта, если операции над проектом подразумевать как операции, реализуемые над каждой нейронной сетью, содержащейся в проекте. При этом указанные операции и их порядок является одинаковым для всех сетей проекта.

Если рассматривать жизненный цикл обработки данных, который является более сложным процессом, описание формальной грамматики в этом случае будет отличаться от разрабатываемой. При расширении поля операций могут присутствовать сразу несколько сетей, в общем случае разной архитектуры и структуры, что будет накладывать определенные ограничения на язык (особенно при описании нижнего уровня детализации). Процесс обработки данных потенциально может быть бесконечен и начинаться не с самого начала и проводиться не до конца с точки зрения описания одной сети. Можно взять сохраненную нейросеть и с некоторой периодичностью проводить различные эксперименты, используя определенные операции или наборы операций. Или же использовать сохраненную сеть на других данных для других задач. Кроме того, возможно создать или прочитать несколько нейронных сетей и проводить отдельные (в общем случае различный набор операций) эксперименты для каждой из сетей. Аналогично процесс обработки для отдельных или всех сетей может проводиться не с начала (сети могут быть созданы ранее) и не до конца (сети могут быть использованы для решения других задач на других данных).

Следует отметить, что формальное описание нейросетевой технологии (а именно алфавит языка описания) может быть представлено для двух различных уровней детализации: первый уровень включает приведенные операции без рассмотрения их алгоритмических реализаций (операции

реализованы в нейроимитаторе), второй – включает реализацию этих операций уже на базе конкретной элементной составляющей сети (нейронов, синапсов, вычисление градиентов и т.п.). Здесь можно провести аналогию с классификацией языков программирования: языки высокого уровня (аналог первого уровня детализации) и низкого уровня (второй уровень детализации).

Таким образом, весь процесс нейросетевой обработки данных рассматривается с некоторой промежуточной точки зрения. Для разработки нейроимитатора необходимо рассматривать процесс работы с одной сетью (проектом). Для обработки данных характерна работа с несколькими сетями (проектами). Соответственно, формальное описание обобщенной обработки данных намного сложнее и едва ли когда-либо будет реализовано в полной мере для всех вариантов и всех особенностей.

Рассмотрим три группы технологических операций при нейросетевой обработке данных. Предварительные операции условно разделим на несколько подгрупп: операции представления данных (сбор данных, структура таблицы базы данных, предварительный выбор поля ответа), предобработка (преобразование символьных полей и полей типа дата, поиск и заполнение пробелов, нормирование), операции считывания файлов данных. Основные операции представляют создание/чтение (загрузка) и обучение нейронной сети. Вспомогательные операции – это тестирование, упрощение нейросетей, определение значимостей входных параметров, вербализация, загрузка сети (открытие файла) и сохранение сети в файл. Загрузка сети может входить как в группу основных операций, так и вспомогательных.

Представим описанную технологию в виде формального описания. Имеем следующий нетерминальный алфавит. Ниже перечислены символы 0 и 1 уровней:

0 уровень

B	Предварительные операции	Before main	Операции сбора данных, создания таблицы, считывания, предобработки данных
M	Основные операции	Main operations	Создание / чтение сети и обучение сети
A	Вспомогательные операции	Auxiliary operations	Тестирование, упрощение нейросетей, определение значимостей входных и внутренних параметров, вербализация, сохранение сети (проекта)

1 уровень

I	Представление	Data	Сбор данных, создание структуры

	данных	introducing	таблицы базы данных, выбор поля ответа (предварительный)
P	Предобработка	Preprocessing	Изменение структуры таблицы, преобразование символьных полей и полей типа дата и других нечисловых полей, поиск и заполнение пробелов, нормирование
L	Обучение сети	Learn	Обучение сети
C	Упрощение структуры сети	Contrasting	Упрощение структуры сети, например, операции контрастирования и бинаризации
N	Нормирование	Normalization	Преобразование данных для приведения значений к определенному диапазону
E	Заполнение пробелов	Empty data	Заполнение пробелов по некоторому алгоритму
T	Тестирование	Test / Validate	Тестирование по примеру и по выборке
D	Предварительное описание данных	Data	Создание структуры таблицы, предварительный выбор поля ответа

Запишем набор допустимых выражений с использованием нетерминальных символов 0 уровня (B, M, A):

{B M A},

{B M},

B M {B M A},

B M A {B M},

B M A {M A},

B M A {B A},

B M {B A},

B M.

2.2.1. Уровень операций пользователя (2-й уровень)

На уровне операций пользователя рассматриваются конкретные операции (предусмотренные в идеальных нейроимитаторах, операции рассматриваются без учета конкретной реализации, в самом общем виде), входящие в состав подгрупп. Предполагается некоторая идеальная система (нейроимитатор или нейросистема), описываются операции, предоставленные конечному пользователю для использования по своему усмотрению. На данном уровне эти операции представляют собой буквы терминального алфавита:

r_d	Считывание файлов данных	reading data	Открытие файла данных
c_i	Контрастирование входных параметров	inputs contrast	Удаление входных параметров с дообучением
c_s	Контрастирование синапсов	synapses contrast	Удаление синапсов с дообучением
c_n	Контрастирование нейронов	neurons contrast	Удаление нейронов с дообучением
c_u	Контрастирование неоднородных входов	non-uniform contrast	Удаление неоднородных входов с дообучением
c_e	Равномерное упрощение сети (контрастирование сигналов так, чтобы на нейрон их приходило не более N)	uniform pruning	Удаление сигналов так, чтобы на нейрон приходило не более N сигналов (с дообучением)
v	Вербализация сети	verbalization	Описание внутренней структуры сети
i	Определение значимостей параметров	importance	Расчет значимостей входных или внутренних параметров, показывающих их значимость при определении выходного поля
b_1	Бинаризация сети ±1	binary	Приведение весов синапсов к значениям из множества {-1, 1}

b_2	Бинаризация сети ±0.5, ±1	binary	Приведение весов синапсов к значениям из множества {-1,-0.5, 0.5, 1}
b_3	Бинаризация сети ±0.25, ±0.5, ±0.75, ±1	binary	Приведение весов синапсов к значениям из множества {-1, -0.75, -0.5, -0.25, 0.25, 0.5, 0.75, 1}
b_4	Бинаризация сети ±0.1, ±0.2, ±0.3, ±0.4, ±0.5, ±0.6, ±0.7, ±0.8, ±0.9, ±1	binary	Приведение весов синапсов к значениям из множества {-1, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}
r_p	Загрузка (чтение файла) сети	read project	Загрузка сохраненной сети (проекта)
s	Сохранение сети в файл	save	Сохранение сети для дальнейшего использования
t_t	Тестирование сети по выборке	test	Тестирование при известном значении поля ответа
t_v	Тестирование сети по примеру	validate	Тестирование при неизвестном поле ответа
t_u	Тестирование по выборке при неизвестном поле ответа		Тестирование по выборке при неизвестном поле ответа
t_p	Тестирование по примеру при известном поле ответа		Тестирование по примеру при известном поле ответа
m_k	Создание сети	make	Создание сети (выбор структуры, поля ответа, характеристик нейронов)
d	Сбор данных	data	Ручная операция ввода данных
s_t	Создание структуры таблицы	table structure	Определение полей, их формата и смыслового содержания

s_y	Выбор поля ответа (предварительный)	select target field	Выбор параметра, для которого хотелось бы получить результаты
m_t	Изменение структуры таблицы	modify table structure	Изменение структуры полей таблицы данных
n_l	Нормирование линейное	linear normalization	Нормирование на интервал [0,1] линейно
n_a	Нормирование кусочно линейное	approximately linear normalization	Нормирование из интервала [-ks,+ks], где s – есть выборочное среднее квадратичное отклонение, k – количество сигм (обычно 1, 2 или 3) на интервал [0,1] кусочно линейно: на левую границу отображается все, что левее, на правую границу – все, что правее
n_d	Нормирование на интервал [a,b]	normalization	Нормирование на интервал [0,1] нестрого, но линейно [-ks, +ks], результатом нормирования будет диапазон [a,b], где a<0, b>1, прямая отображается в прямую, но интервал [-ks, +ks] отображается в [0, 1]
n_n	Нелинейное нормирование	nonlinear normalization	Нелинейное нормирование на интервал [0,1], например с использованием сигмоиды, функций cos или arctg, логарифмического нормированиия
n_s	Нормирование на сферу единичного радиуса	sphere normalization	Нормирование на сферу единичного радиуса (каждый вектор делится на свою длину)
f	Преобразование полей	modify fields	Преобразование полей к числовому формату и введение дополнительных полей (изменение структуры и данных)

e_z	Заполнение нулем	zero fill	Заполнение пробелов нулем
e_a	Заполнение средним	average fill	Заполнение пробелов средним значением по полю
e_c	Заполнение средним по классу	class average fill	Заполнение пробелов средним значением по классу, если класс известен
e_m	Заполнение минимальным	minimal fill	Заполнение пробелов минимальным значением по полю
e_x	Заполнение максимальным	maximal fill	Заполнение пробелов максимальным значением по полю
l_m	Обучение методом Монте-Карло		Обучение методом Монте-Карло
l_p	Обучение методом покоординатного спуска		Обучение методом покоординатного спуска
l_r	Обучение методом случайного поиска		Обучение методом случайного поиска
l_n	Обучение методом Нелдера-Мида		Обучение методом Нелдера-Мида
l_s	Обучение методом наискорейшего спуска		Обучение методом наискорейшего спуска
l_k	Обучение методом kParTan		Обучение методом kParTan
l_c	Обучение квазиньютоновским методом BFGS		Обучение квазиньютоновским методом BFGS
l_b	Удар сети	Bump	Возмущение весов синапсов

Примечание: основой для предложенного списка методов обучения послужили методы, используемые в нейроимитаторах Neurogenesis, STATISTICA Neural Networks, и ряде других подобных программ, в том числе разработанных в г. Красноярске, допустимо дополнение данного перечня (строго говоря, число модификаций методов обучения потенциально

бесконечно, поэтому здесь приведены некоторые основные, чаще используемые методы). Обычно выбор метода обучения реализуется в нейроимитаторах в виде настройки или опций (часть 3 уровня детализации – реализации операций). В этом случае на уровне операций пользователя обучение является одной операцией без указания конкретного метода (соответственно можно использовать только один терминальный символ для обучения сети).

Аналогичным образом можно дополнять операции тестирования, бинаризации и контрастирования (например, полезно ввести операцию контрастирования слоев нейронов). Дополнение операциями не влияет на сложность грамматики, увеличивается только число правил.

Представим некоторые операции (элементы алфавита) в виде форм Бэкуса-Наура, начиная с самого верхнего уровня и далее детализируя элементы нетерминального алфавита, в виде совокупности элементов терминального алфавита:

S::= {B M [A]}| {B M [A] {B A}}| {B M A {M A}}

B::= [I] r_d P

M::= m_k L | r_p L | r_p

A::= i | C | v | s | T

A::= i A | C A | v A | s A | T A

T::= t_t | t_v | t_u | t_p

L::= l_m | l_p | l_r | l_n | l_s | l_k | l_c | l_b

L::= l_m L | l_p L | l_r L | l_n L | l_s L | l_k L | l_c L | l_b L

C::= c_i | c_s | c_n | c_u | c_e | b_1 | b_2 | b_3 | b_4

C::= c_i C | c_s C| c_n C| c_u C| c_e C| b_1 C | b_2 C | b_3 C | b_4 C

I::= [d] s_t [s_y]

P::= [f] [E] N

N::= n_l | n_a | n_d | n_n | n_s | n_s n_l

E::= e_z | e_c | e_a | e_m | e_x

Укажем все правила данной грамматики (на уровне операций пользователя):

S $\rightarrow$ B M A K | B M K (1)

A $\rightarrow$ A M A | A M A K | A M (2)

K $\rightarrow$ B M K | B M A K | B A K | B M | B M A | B A | λ (3)

$C \rightarrow c_i\ C \mid c_s\ C \mid c_n\ C \mid c_u\ C \mid c_e\ C \mid b_1\ C \mid b_2\ C \mid b_3\ C \mid b_4\ C$ (4)

$C \rightarrow c_i \mid c_s \mid c_n \mid c_u \mid c_e \mid b_1 \mid b_2 \mid b_3 \mid b_4$ (5)

$C \rightarrow \lambda$ (6)

$A \rightarrow i\ A \mid C\ A \mid v\ A \mid s\ A \mid T\ A$ (7)

$A \rightarrow i \mid C \mid v \mid s \mid T$ (8)

$L \rightarrow l_m\ L \mid l_p\ L \mid l_r\ L \mid l_n\ L \mid l_s\ L \mid l_k\ L \mid l_c\ L \mid l_b\ L$ (9)

$L \rightarrow l_m \mid l_p \mid l_r \mid l_n \mid l_s \mid l_k \mid l_c \mid l_b$ (10)

$T \rightarrow t_t \mid t_v \mid t_u \mid t_p$ (11)

$B \rightarrow r_d\ P \mid I\ r_d\ P$ (12)

$P \rightarrow m_t\ f\ E\ N \mid m_t\ f\ N \mid E\ N \mid N$ (13)

$I \rightarrow d\ D \mid d$ (14)

$I \rightarrow \lambda$ (15)

$D \rightarrow s_t\ s_y \mid s_t$ (16)

$M \rightarrow m_k\ L \mid r_p\ L \mid r_p$ (17)

$N \rightarrow n_l \mid n_a \mid n_d \mid n_n \mid n_s \mid n_s\ n_l$ (18)

$E \rightarrow e_z \mid e_c \mid e_a \mid e_m \mid e_x$ (19)

Укажем, что правила вывода введены с учетом следующих ограничений:

а) первой операцией должно быть считывание файла данных;

б) операции предобработки нельзя проводить до считывания файла данных (тривиальное ограничение);

в) операции сбора данных нельзя проводить после определения структуры базы данных и предварительного выбора поля ответа (тривиальное ограничение) – рассматривается с точки зрения использования нейроимитатора, который предполагает работу с конкретной структурой и набором данных;

Примечание: сбор дополнительных данных (дополнение существующего файла данных) предполагает обрыв реализации дальнейших технологических операций, т. к. нейроимитатор обычно не предполагает постоянного обращения к файлу данных. В этом случае требуется заново

открыть файл данных с дополненными записями и проводить все операции с самого начала. Данный подход может быть эффективен, так как сбор дополнительных данных или дополнение полей позволяет более точно определить взаимосвязи между параметрами и найти обоснование результатов.

г) операции нормирования нельзя проводить до изменения структуры и преобразования типов полей и заполнения пробелов;

д) все операции первоначально (при первом входе) обязательно должны начинаться с предварительных и основных;

е) основные операции нельзя проводить до считывания файла данных (тривиальное ограничение);

ж) вспомогательные операции нельзя проводить до предварительных (тривиальное ограничение);

з) последняя операция не должна быть предварительной.

Утверждения и ограничения, касающиеся данной грамматики, приведены далее. Первоначально при разработке формального описания нейросетевой технологии отдельные свойства грамматики не рассматривались. Правила вывода строились на основе эмпирических данных о корректной последовательности операций и их более общей группировки. Отметим, что нетерминальный символ K введен для учета ограничений на комбинации операций 0 уровня.

Данная грамматика не предполагает использование циклов при выполнении последовательности операций, что подразумевает наличие условий или критериев возникновения данного события. Реализация входа и выхода из цикла представляет собой организацию соответствующего внутреннего алгоритма, который описывается на уровне реализации операций в самом нейроимитаторе и чаще всего не имеет средств для его корректировки пользователем. Грамматика разработана только до уровня операций пользователя, поэтому реализация цикличности здесь не учтена.

На данном уровне детализации возможно лишь описание критериев наличия таких циклов операций. Перечень таких критериев традиционен и общепринят для данной предметной области (нейроинформатики): обучение происходит по заданной выборке до требуемого уровня точности (задается в настройках при создании сети), сокращение числа элементов – последовательно с дообучением (попытка обучить сеть без отдельного элемента, если требуемая точность не достигнута, элемент не вырезается, переход к следующему и т. д.).

Данная грамматика, рассматриваемая в контексте описания нейросетевой технологии, имеет некоторые особенности и возможные модификации, которые отличают ее от традиционного подхода в теории формальных языков и грамматик. По разделению иерархии операций,

алфавит грамматики можно назвать 3-уровневым: 0 – группы операций, 1 – подгруппы, 2 – операции.

Кроме того, изначально предложенная грамматика описания нейросетевой технологии «разраслась» до семейства грамматик, которые описывают различные модификации первоначальной, полученных не столько эквивалентными преобразованиями, сколько изменениями правил вывода (усечением или расширением) в соответствии с анализом предметной области.

Поэтому в дальнейшем примем первоначальные правила грамматики за базовые и при описании правил грамматики семейства будем указывать только правила, которых касаются изменения или модификации.

2.2.2 Построение атрибутных грамматик для нейросетевой технологии

Для реализации грамматики нейросетевой технологии возникает необходимость дополнительного описания некоторых параметров, связанных с их семантикой. Для этого существуют так называемые атрибутные грамматики (по аналогии атрибутные грамматики используются для описания семантики языков программирования).

Описание атрибутной грамматики состоит из раздела описания атрибутов и раздела правил. Раздел описания атрибутов определяет состав атрибутов для каждого символа грамматики и тип каждого атрибута. Правила состоят из синтаксической и семантической части. Семантическая часть правила состоит из локальных объявлений и семантических действий. В качестве семантических действий допускаются как атрибутные присваивания, так и составные операторы.

При этом аналогично уровням технологии нужно учесть наличие уровней или групп данных параметров по степени «сложности» нейроимитатора в смысле реализации некоторых специфических функций: условно для неспециалистов (реализация основных функций с минимальной настройкой или настройкой по умолчанию) и для профессионалов (возможность тонкой настройки архитектуры сети и ее параметров).

2.2.2.1. Уровень «неспециалиста» (общий уровень)

При описании терминального алфавита введены индексы, что позволяет выделять различные виды отдельной операции (например, несколько видов нормирования, обучения и т.п.). Дополнительно можно ввести параметры, определяющие, например вид выборки, ее имя для операции открытия файла данных. В данном контексте нетерминальные символы (и терминальные символы, если рассматривать уровень реализации операций) можно рассматривать как полу терминальные или частично

терминальные. Например, операции нормирования (N) входят в операции предобработки (P), операции упрощения – во вспомогательные и т.п.

Некоторые операции, например, чтения файла данных или загрузки, сохранения сети требуют наличия параметров, которые однозначно определяют данную операцию (например, указание конкретного файла данных). Использование параметров позволяет точнее определить контекст и общий смысл последовательности операций.

Например, операцию чтения файла данных можно представить следующими терминальными символами:

r_{dl}	считывание обучающей выборки	reading data (learn sample)
r_{dt}	считывание тестовой выборки	reading data (test sample)
r_{dg}	считывание общей совокупности данных	reading data (general sample)

Строго говоря, для данного описания требуется использовать своеобразный «вывод», который аналогичен подходу, применяемому для вывода слов языка из правил грамматики. Только в данном случае с каждым шагом происходит конкретизация рассматриваемой операции (терминального символа на данном уровне детализации описания технологии). Например, для операций чтения файла данных можно представить следующий “вывод” (отдельно для каждой выборки):

r_{dl}	r_{dt}	r_{dg}
↓	↓	↓
r_{dl}(source)	r_{dt}(source)	r_{dg}(source)
↓	↓	↓
r_{dl}(svod1.dbf)	r_{dt}(svod2.dbf)	r_{dg}(svod.dbf)

В данном случае символ, описывающий операцию чтения общей выборки данных из файла svod.dbf формально может быть записан как r_{dg}(svod.dbf).

Следует отметить, что на данном уровне в качестве атрибутов операции чтения данных используется ограниченный набор, относящийся непосредственно к нейросетевой технологии, т.е. не учитываются атрибуты, относящиеся непосредственно к технологии самого процесса чтения: например, чтение заголовка, структуры базы данных (заголовки полей, типы данных) и т.п. Предполагается, что реализация процесса является традиционной и не требует дополнительного описания в данной работе.

Обычно в атрибутных грамматиках считается, что для терминальных символов не существует семантических правил вывода для вычисления атрибутов. Предполагается, что атрибуты терминальных символов – либо предопределенные константы, либо доступны как результат работы лексического анализатора.

В данном случае в связи со сложностью и неоднозначностью трактовки грамматики данное утверждение будем считать неактуальным, т.е. для данной грамматики и для терминальных символов будут существовать правила вывода для вычисления атрибутов.

Данное предположение аналогично опыту разработки трансляторов для атрибутных грамматик, согласно которому использование «чистого» атрибутного формализма вызывает трудности при создании транслятора.

В дальнейшем будем использовать две равнозначные формы записи атрибутных грамматик: 1) атрибут a символа X как a(X) или 2) атрибут a символа X как X(a). В первом случае значение a(X) будет приведено после символа «=» (например, a(X)=’svod.dbf’ – значение атрибута a символа X есть ’svod.dbf’). Аналогично во втором случае: X(a)=’svod.dbf’.

С учетом данных рассуждений построим набор атрибутов, которые необходимы для некоторых операций технологии:

l	learning source	обучающая выборка
t	test source	тестовая выборка
g	general source	общая (генеральная совокупность данных)
r	read network/ read project/ create network	чтение (сети, проекта) или создание сети

Набор необходимых значений атрибутов:

source	источник данных	имя файла данных, который используется для обучения или тестирования (в формате DBF или DB)
network	сеть	имя файла сети (обученной сети)
project	проект	имя проекта (содержит серию обученных сетей)

report	отчет	сводная отчетная информация по результатам обучения, тестирования, вспомогательных операций сети

При описании правил вывода и семантических правил для атрибутов используем следующий вид записи: слева – традиционные правила вывода слов языка (указываются в основном те правила, терминальные или нетерминальные символы в которых имеют атрибуты), от центра справа – семантические правила для атрибутов, по правому краю – номер правила (в соответствии с нумерацией правил первоначальной грамматики).

Построим предварительные правила с использованием атрибутов:

S	→	B M A K | B M K		(1)
A	→	A M A | A M A K | A M		(2)
K	→	B M K | B M A K | B A K | B M | B M A | B A | λ		(3)
C	→	c_i C | c_s C| c_n C| c_u C| c_e C| b_1 C | b_2 C | b_3 C | b_4 C		(4)
C	→	c_i | c_s | c_n | c_u | c_e | b_1 | b_2 | b_3 | b_4		(5)
C	→	λ		(6)
A	→	i A| C A| v A| s A| T A		(7)
A	→	i | C | v | s | T	l(s)=’project’ l(s)=’network’ l(i)=’report’ l(v)=’report’	(8)
L	→	l_m L | l_p L | l_r L | l_n L | l_s L | l_k L | l_c L | l_b L	l(L)=l(l_i)	(9)
L	→	l_m | l_p | l_r | l_n | l_s | l_k | l_c | l_b	l(l_i)=’network’	(10)
T	→	t_t | t_v | t_u | t_p		(11)
B	→	r_d P | I r_d P	l(r_d)=’source’ t(r_d)=’source’	(12)

			$g(r_d)$=’source’	
P	→	m_t f E N | m_t f N | E N | N		(13)
I	→	d D | d		(14)
I	→	λ		(15)
D	→	s_t s_y | s_t		(16)
M	→	m_k L | r_p L | r_p	$r(m_k)$=’network’ $r(r_p)$=’project’ $r(r_p)$=’network’	(17)
N	→	n_l | n_a | n_d | n_n | n_s | n_s n_l		(18)
E	→	e_z | e_c | e_a | e_m | e_x		(19)

Для данной грамматики (A_s – множество синтезируемых атрибутов, A_i – множество наследуемых атрибутов):

$A_s(s)=Ø$	$A_I(s)=\{l\}$
$A_s(i)=Ø$	$A_I(i)=\{l\}$
$A_s(v)=Ø$	$A_I(v)=\{l\}$
$A_s(l_i)=Ø$	$A_I(l_i)=\{l\}$
$A_s(L)=\{l\}$	$A_I(L)=Ø$
$A_s(r_d)=Ø$	$A_I(r_d)=\{l, t, g\}$
$A_s(r_p)=Ø$	$A_I(r_p)=\{r\}$
$A_s(m_k)=Ø$	$A_I(m_k)=\{r\}$

Запись $A_s(L)=\{l\}$ обозначает, что атрибут l символа L входит во множество синтезируемых атрибутов; аналогично $A_I(l_i)=\{l\}$ – атрибут l символа l_i входит во множество наследуемых символов; $A_I(L)=Ø$ – множество наследуемых атрибутов для символа L пустое.

2.2.2.2. Уровень «специалиста» (специальный уровень)

Данный уровень характеризуется более глубоким (по степени вложенности) уровнем детализации терминального алфавита. Здесь сохраняются все атрибуты и правила вывода предложенные ранее (на уровне «неспециалиста»), но дополнительно формируется еще ряд атрибутов.

Представим некоторые дополнительные атрибуты, характерные для данного уровня детализации:

p	network params	параметры сети (число слоев, нейронов и т.п.)

Значения атрибутов, характерные для данного уровня детализации:

const	Константа активации	числовой параметр
layers	число слоев	число слоев созданной сети
neuron	число нейронов	число нейронов в слое созданной сети
func	вид функции	вид функции
ver	уровень точности	числовое значение, использующееся при обучении сети, как допустимое отклонение для поля ответа
lkor	признак обученности сети до конца	показатель того, что сеть обучилась правильно по всем примерам выборки (100% правильности примеров при тестировании по обучающей выборке)

Данный набор значений атрибутов (за исключением последнего) относится к операции создания сети (m_k), для этого введем «общий» атрибут (p), который будет содержать в себе это подмножество. Возможно, предпоследний параметр следует передать не только на этап обучения, но и на этап тестирования, хотя технологически данная константа не должна менять свое значение при тестировании.

Последнее значение атрибута (lkor) относится, скорее, к операции обучения (L) и должен относиться к множеству синтезируемых атрибутов

(A_s). Данный признак формируется при окончании цикла обучения (1 вариант – сравнивается общее количество примеров и примеров, по которым сеть обучилась; 2 вариант – определяются процент правильно определенных примеров на этапе тестирования по выборке, которая использовалась для обучения) и в случае ложности слово языка обрывается (технологические этапы нужно начинать сначала) (b(A)=λ).

Набор правил (выборка):

M	→	m_k L | r_p L | r_p	l(M)= p(m_k) l(L) p(m_k)=’const’ p(m_k)=’layers’ p(m_k)=’neuron’ p(m_k)=’func’ p(m_k)=’ver’	(17)
L	→	l_m L | l_p L | l_r L | l_n L | l_s L | l_k L | l_c L | l_b L		(9)
L	→	l_m | l_p | l_r | l_n | l_s | l_k | l_c | l_b		(10)
A	→	i A| C A| v A| s A| T A		(7)
A	→	i | C | v | s | T		(8)

Для данной грамматики:

$A_s(m_k)=Ø$ $A_I(m_k)=\{p\}$

$A_s(M)=\{l\}$ $A_I(M)=\{p\}$

2.2.2.3. Уровень «разработчика» (уровень реализации)

Данный подход (использование атрибутивной грамматики) расширительно можно применять для разработки специализированных вспомогательных программ (модулей, макросов), подключаемых к нейроимитатору, которые будут обучать требуемое количество однотипных сетей (по запросу пользователя), выполнять другие операции, а затем выдавать пользователю сводные результаты и описания выполненных действий. Использование готовых нейроимитаторов требует значительных временных затрат для проведения исследования, так как полноценные

эксперименты требуют выполнения однотипных операций над множеством сетей (обычно приходится многократно повторять одни и те же операции над каждой сетью, т. к. в нейроимитаторах не предусмотрена возможность пакетного использования каждой операции). Разработка макросов позволит устранить эту особенность, существенно автоматизировать и облегчить процесс проведения многочисленных экспериментов со множеством сетей.

Присутствие в алфавите отдельных терминальных символов для чтения разных выборок предполагает наличие соответствующих отдельных операций, реализованных в нейроимитаторе. В некоторых зарубежных нейроимитаторах данные операции (в частности чтения обучающей и тестовой выборок) представлены отдельно в виде двух кнопок. Таким образом, можно одновременно открыть обучающую и тестовую выборки, а затем проводить дальнейшие операции без повторного их открытия. При разработке данного языка и грамматики использовался другой подход, который предполагает открытие только одного файла данных (либо обучающей, либо тестовой выборки).

Данные особенности определяются прежде всего интерфейсом нейроимитатора, поэтому для систематизации описания приведем рекомендуемый перечень операций и их расположение на форме, в зависимости от степени важности и частоты использования. Отметим, что условно операции можно расположить на следующих элементах экранной формы: 1) меню; 2) кнопки в виде панели инструментов (“горячие” клавиши); 3) элементы настройки. В элементах настройки можно расположить операции наименее используемые: например, задание структуры сети, определение типов полей (входные, выходные), значение отклонения выходного параметра (используется при обучении и тестировании), настройка алгоритмов обучения и методов нормирования. Элементы меню представляют операции, которые используются средне часто: например, операции упрощения сети (контрастирование, бинаризация), вербализация. В виде кнопок на панели можно организовать операции, которые используются очень часто: обучение сети, тестирование, определение значимостей.

Операции нормирования выполняются в некоторых нейроимитаторах автоматически, в иных по запросу пользователя, некоторые нейроимитаторы требуют подавать нормированные значения во входном файле данных (операции нормирования выполняются внешними программами). Из этих соображений данные операции включены в обязательный перечень операций предобработки при построении слов языка.

Вывод

Использование атрибутных грамматик хотя и усложняет исходную КС-грамматику, но позволяет детально рассмотреть особенности реализации и семантику различных технологических операций и существенно уточняет нейросетевую технологию. Данный подход может быть распространен и на все семейство грамматик, полученных на основе первоначальной. Как дальнейшее развитие следует определить класс атрибутных грамматик и построить транслятор по аналогии с языками программирования.

Метод с использованием атрибутных грамматик аналогичен по своей сути методологии IDEF3, реализующий сценарий процессов посредством перекрестков и светофоров. Тем не менее, является более предпочтительным для разработки трансляторов, как исторически теоретическая основа. Преимуществом атрибутного подхода является функциональная природа атрибутного вычислителя и неупорядоченность процесса вычисления атрибутов. С одной стороны это вызывает некоторые трудности при построении трансляторов. Но позволяет существенно расширить грамматику потенциально для любой сложности предметных областей, в том числе и для нейросетевой технологии.

Контрольные вопросы ко 2 главе:

1. Технология
2. Предобработка
3. Обучение и тестирование
4. Контрастирование: виды, способы
5. Основные типы задач, решаемых с помощью нейронных сетей?
6. Генеральная совокупность, обучающая выборка, тестовая выборка.
7. Эффект переобучения сетей
8. Выбор поля ответа для двойственных нейронных сетей
9. Качество обучения сети
10. Способы получения явного знания от нейронной сети с учителем

3 Нейроимитаторы

Описанные выше технологии и методы использования нейронных сетей для обработки данных полезны для использования на начальном этапе работы – обдумывания и постановки задачи. Постановка задачи есть первый и необходимый этап работы, но он не является достаточным. Очень важно практическое использование конкретных инструментов – имитаторов нейронных сетей, степень реализации предложенной технологии в таких имитаторах, качество и удобство работы с программами и т.д.

Среди большого количества готовых пакетов программ можно отметить некоторые, наиболее известные специалистам нашей страны и особенно Сибири: Clab (С.Е. Гилев), MultiNeuron разных версий (А.А. Россиев, Е. М. Миркес, С. Е. Гилев и др.), NeuroPro (В. Г. Царегородцев), Eye, FAMaster (Д.А. Россиев), _ (Зиновьев А.), STATISTICA Neural Networks, MATLAB и многие другие. Есть большое количество менее известных имитаторов MDN, Neurogenesis (А.В. Хомич) и ряде других подобных программ/

Обучение нейронных сетей с учителем может быть выполнено с помощью программ Monoton и многих других. Кроме того, в настоящее время в некоторых пакетах статистики и в электронных таблицах предусмотрены дополнительные модули для работы с нейронными сетями. Например, продукты серии Excel Neural Package (Winnet, Kohonen map), реализованные как набор надстроек над Microsoft Excel.

При выборе программ для данной работы оценивался ряд объективных и субъективных свойств имитаторов. Например, требовался интерфейс на русском языке или хотя бы наличие текстов или учебных пособий на русском языке. Для большинства зарубежных имитаторов или пакетов программ, в которые включены возможности использования нейронных сетей, как правило недостаточно учебной или методической литературы, не говоря уже о том, что интерфейс большинства из них на английском языке. Среди более известных зарубежных пакетов был выбран пакет SNN. Выбор SNN обосновывается еще и тем, что в большинстве имитаторов или пакетов отечественного производства используют нейронные сети одной архитектуры. SNN предлагает на выбор несколько различных архитектур.

Очень хорошие программы MultiNeuron (разных версий), EYE, CLAB разработаны для работы в операционной системе MS DOS, поэтому иногда вызывали проблемы при работе – недостаточна совместимость с современными ЭВМ и операционными системами. Кроме того, CLAB был построен для очень быстрого и эффективного решения задачи бинарной классификации. EYE позволяет решать разные задачи для изучения разных методов и алгоритмов обучения нейронных сетей, но менее удобен для освоения основ технологии в современном понимании. MultiNeuron более

удобен, но к нему предъявляют претензии из-за полносвязной архитектуры используемых нейронных сетей.

Ниже предложено только краткое описание некоторых из указанных нейроимитаторов. Нейроимитатор NeuroPro версии 0.25 в настоящее время не поддерживается и не сопровождается разработчиком. Однако, эта программа была внедрена во многих учебных заведениях и легко доступна в Интернете на многих сайтах, имеет интерфейс на русском языке, хорошо решает многие задачи даже с установкой параметров по умолчанию. Выбора и использование NeuroPro как одного из удобных и доступных представителей невозможно избежать.

Выбор Neurogenesis основан на интересной архитектуре встраиваемой системы. Описание данного имитатора основано в большей мере на текстах разработчика.

3.1 Особенности технологии работы с NeuroPro

Здесь приведено краткое описание методики использования общей технологии обработки данных с помощью нейросетей с учителем. Для этого использовался нейроимитатор NeuroPro v. 0.25 (разработчик В.Г. Царегородцев). Данный порядок не является единственно возможным, в частности, существует авторский «ритуал» обработки данных, включающий в себя более полную и адекватную (с точки зрения математической статистики) рекомендуемую последовательность обработки данных. Автор считает, что нейроимитатор NeuroPro устарел и указывает на необходимость использования более гибких и мощных методов. Тем не менее, данный программный продукт широко распространен, реализует основные операции и позволяет быстро получить предварительные результаты начинающими исследователями. Для получения дистрибутива программы авторы учебного пособия рекомендуют обратиться в Интернет, где программа выложена на десятках сайтов для бесплатного доступа.

В данном случае рассматривается самый упрощенный порядок действий, полезный в первую очередь для студентов или начинающих исследователей, слабо знакомых с аппаратом искусственных нейронных сетей, и позволяющий быстро получить предварительные результаты.

Первоначально выполняются операции, связанные со сбором данных, перевод их в электронный вид (если они представлены на бумажном носителе). Чаще предполагается использование формата DBase (хотя возможно и Paradox). Поля файла данных должны быть числового типа (целые или действительные). Символьные поля не используются при обработке, но могут быть полезны для идентификации записей.

Перед запуском нейроимитатора нужно сформировать на основе общего файла данных две выборки: обучающую и тестовую, которые

помещаются в отдельные файлы. Критерии отбора записей могут быть выбраны пользователем, для начала рекомендуется отбор четных/нечетных записей (четные – в один файл, нечетные – в другой). Далее для удобства описания назовем обучающую выборку – выборка 1, тестовую – выборка 2.

Следует предварительно выбрать поле ответа – одно из полей (параметров) файла данных, которое требуется прогнозировать (обычно поле ответа выбирают с учетом мнения специалиста в предметной области).

Рекомендуется вести краткий протокол всех действий. Например, для чего был создан и использовался проект АВС – тестирования, либо обучения по 1-й выборке. Такой протокол позволит в дальнейшем избежать многих ошибок и уточнить результаты в случае необходимости.

Рассмотрим три простейших варианта методики для неподготовленного пользователя:

А. Обучение по общей выборке.

Б. Простое обучение по обучающей выборке и тестирование по тестовой выборке.

В. Перекрестное обучение и тестирование.

Г. Дополнительные операции (определение значимостей параметров, упрощение сети, вербализация).

При каждом из этих подходов первоначальный порядок действий по созданию и сохранению начального состояния нейронных сетей (проекта) является одинаковым (с. 40, 41).

В данной методике предлагается использовать следующие рекомендации:

а) обучающая и тестовая выборки (отдельные файлы данных) названы в соответствии с общей совокупностью добавлением после названия цифр 1 (обучающая) или 2 (тестовая). Например, если общий файл данных назван svod.dbf, то первично обучающая выборка получит название svod1.dbf, тестовая – svod2.dbf. Это окажется полезным в дальнейшем при сохранении проектов сетей. Буквы t и o в названия файлов-выборок добавлять не рекомендуется, т. к. может потребоваться использовать тестовую выборку в качестве обучающей (например, при перекрестном тестировании). В этом случае метки t и o будут скорее сбивать с толку, чем помогать. Метки 1 и 2 индифферентны (не определяют конкретный вид выборки);

б) для сохранения сетей и результатов при использовании отдельного выходного поля, либо проведения дополнительных экспериментов рекомендуется создавать отдельные каталоги (папки). Причем, для дополнительных экспериментов либо изменении структуры сети в пределах одного выходного поля созданные каталоги должны быть вложенными. Например, для сохранения результатов с использованием выходного поля X6, создан каталог «X6», если нужно провести эксперименты с удалением отдельных параметров (отдельные поля не используются при обучении) при

том же выходном поле X6, следует внутри каталога «X6» завести директорию «VIS_X7-9» (параметры X7, X8, X9 исключаются из числа входных). Если нужно провести аналогичные эксперименты, используя структуру сети, отличную от заданной по умолчанию (3 слоя по 10 нейронов), например, 1 слой по 30 нейронов, создаем каталог «1_30» и т. п.;

в) проект после первоначального создания сетей (сети не обучены) назван n_ish (исходный), здесь «n» – условное обозначение решаемой задачи;

г) проект с обученными по одной из выборок сетями назван в соответствии с номером выборки: n_o1 (сети обучены по выборке svod1), n_o2 (сети обучены по выборке svod2), n_o (сети обучены по общей выборке);

д) для контрастированных сетей в наименование добавлена буква «k», при контрастировании внутренней структуры дополнительно используются первые буквы соответствующих элементов: «s» – синапсов, «n» – нейронов, «u» – неодрородных входов, «r» – равномерное упрощение сети. Например, проект, содержащий обученные по выборке svod1 сети после контрастирования синапсов, назван n_oks1, обученный по общей выборке после сокращения числа входов – n_ok;

е) при бинаризации весов синапсов в наименование проекта добавляется буква «b». Например, проект n_ob2 содержит сети, обученные по выборке svod2 после бинаризации;

ж) возможны более сложные варианты, соответствующие комбинации нескольких операций, например: n_okb2 – проект, содержащий сети, обученные по выборке svod2, после последовательного сокращения числа входов и бинаризации.

Примерный план выполнения работы:

1. Создать несколько нейросетей одинаковой структуры (Файл/Создать).

Количество сетей определяется исследователем, влияет на степень достоверности результатов. Минимальное количество при временных ограничениях рекомендуется не менее 3. Статистически требуемое количество сетей никак не обосновано, но если нет ограничений по времени, рекомендуется использовать не менее 10. Открываем исходный файл данных, создаем новую сеть. При этом для каждой сети определяются входные поля и поле ответа (выходное поле). В NeuroPro по умолчанию назначается полем ответа – последнее. Если нужно использовать другое поле, то последнее поле следует отметить входным, т. к. нейроимитатор может использовать несколько полей в качестве поля ответа. Существует две настройки типа выходного поля: «количественный (непрерывный)» и «качественный». По умолчанию установлен «количественный», этот тип соответствует решению задачи предикции (предсказания). При этом можно дополнительно выбрать точность решения задачи (по умолчанию установлено значение «0,1»). В

общем случае точность выбирается с учетом характера распределения значений данного поля либо расчета основных статистических характеристик (дисперсии, среднеквадратичного отклонения и т. п.). Данная величина влияет на качество тестирования сети. Если нет дополнительных ограничений и особенностей задачи, рекомендуется не менять данное значение точности (оставить значение по умолчанию). Тип «качественный» предполагает дискретность изменения значений параметра в небольшом диапазоне, что на практике соответствует задачам классификации. В этом случае дополнительно появляется вкладка «дискретные состояния поля X», где указаны число дискретных состояний и их конкретные значения, можно указать отношение предшествования значений данных состояний. Для любого поля выбирается одно из возможных состояний: поле не используется сетью (автоматически помечаются символьные поля), поле является входным для сети и поле является выходным для сети (поле ответа). Начинающим исследователям можно начинать с настроек сети и характера выходного поля (количественное), установленных по умолчанию. В общем случае даже задачи классификации можно упрощенно трактовать как задачи предикции.

Для входных полей также существуют аналогичные выходным полям настройки типов поля. При выборе типа «качественный» появляется столько дополнительных полей, сколько состояний поля (классов) указано в настройке. Наименования дополнительных полей образованы из исходного с добавлением номера класса (например, исходное поле X23 имеет 3 состояния, появятся следующие поля: X23_1, X23_2, X23_3). Это следует учитывать при обработке данных, так как данные поля считаются самостоятельными и имеют отдельные значения значимостей.

Отметим, что реализация операций для задач классификации в NeuroPro несколько отличается от предикции: результаты тестирования представлены развернуто – не только средние количество и процент, но и отдельно для каждого класса. Кроме того, помимо традиционных процентов и количества верно и неверно определенных примеров приводятся количество и процент примеров, определенных неуверенно.

Следует отметить, что существуют поля, которые не рекомендуется использовать сетью при обучении:

а) поля, представляющие собой порядковые номера (например, код донора, номер строки, номер плазмафереза, код студента и т. п.), использовать на входе бессмысленно, т. к. они не несут существенной информации; существуют ситуации осмысленности таких полей, например, если они несут информацию о временном порядке, поэтому для каждой конкретной задачи и параметра требуется выполнять дополнительный анализ;

б) поля, вычисляемые на основе других полей и однозначно определяющие поле ответа (например, если поле суммы определенных параметров меньше 5 – поле ответа 0, иначе – 1), не используются сетью, т. к. при этом обычно получается линейная задача, которая решается тривиально и не требует использования нейронных сетей.

Следует учитывать и фактор неполноты данных (в данных имеются пробелы): желательно либо заполнить их (если такое возможно), либо не использовать в качестве входных пустые (полупустые) записи и поля. В NeuroPro вырезание пустых записей происходит в автоматическом режиме. Запись не используется при обучении и тестировании, если не заполнено хотя бы одно значение поля.

Структура нейронной сети (количество слоев и нейронов) определяется пользователем в зависимости от конкретной задачи. По умолчанию в NeuroPro используется 3 слоя по 10 нейронов в каждом слое. Для большинства прикладных исследований больших нейронных сетей не требуется, и обычно выбирают структуру, заданную по умолчанию. Характеристика нейронов влияет на скорость обучения сети, чем больше значение характеристики, тем медленнее может обучаться сеть, но при этом может давать лучшее качество прогноза. Однако для большинства случаев можно использовать по умолчанию установленное в NeuroPro значение, равное 0,1.

2. Сохранить данный проект (например, n_ishx6 – для выходного поля X6). Под проектом понимается файл внутреннего формата NeuroPro (*.npp), в котором хранятся все созданные сети.

Первоначальное сохранение проекта в нескольких копиях (сети не обучены) и его дальнейшее использование обеспечивает идентичность (тождественность) начального состояния пространства сетей в соответствии с порядком расположения сетей в проекте. Идентичность начального состояния проектов не обязательна. Для некоторых конкретных задач или операций идентичность недопустима, сети должны быть разными. Обычно, отдельные проекты содержат сети, сформированные для одних условий (как в данной методике), например, обученные по одной и той же выборке. В общем случае это не влияет на результат и производительность, но иногда дает некоторые преимущества. Разумеется, сети в одном проекте различны, даже по начальному состоянию. В NeuroPro уникальность каждой сети обеспечивается еще при создании (в общем случае один проект может содержать сети, разные по структуре и данным). В дальнейшем при обучении состояние сетей будет изменяться, но исходно «параллельные» сети (из копий одного проекта) начинают обучение с одной и той же точки многомерного пространства. Конечное состояние, достигнутое после обучения, будет не одинаковым, если использовать различные наборы входных данных. Соответственно результаты, полученные при данном

подходе для разных выборок, можно сравнивать и сопоставлять с большим основанием.

Для варианта А далее выполнить следующее:

3. Сохранить проект под другим именем, если далее предполагается использование варианта Б (например, n_o).

4. Открыть общую выборку.

5. Обучить все нейросети.

6. Сохранить проект под тем же именем.

В результате имеем сохраненный проект с обученными нейросетями по общей выборке.

Для варианта Б выполнить следующее:

3. Сохранить проект под другим именем (например, n_o1).

4. Открыть выборку 1 (обучающую).

5. Обучить все нейросети.

6. Сохранить проект под тем же именем.

7. Провести тестирование, для этого:

7.1. Открыть выборку 2 (тестовую).

7.2. Протестировать все нейросети.

7.3. Сохранить все результаты тестирования: предварительные результаты в виде текстовых файлов (сохранение из главного меню), сводные результаты в виде таблицы (примерный вид таблицы – табл. 3.1).

Следует обратить внимание на процент правильно определенных примеров для нейросети. Если этот показатель составляет около 60 %, то результаты тестирования и, соответственно, качество обучения сети удовлетворительные. Если 70–80 % – хорошие, 90 % и более – отличные. Если процент правильно определенных примеров меньше 50 %, то следует более детально исследовать данные, провести дополнительные операции по их преобразованию, и только затем повторно применить данную методику (либо обратиться к более квалифицированному специалисту за помощью).

Для варианта В нужно выполнить следующее:

3. Сохранить проект под другим именем (например, n_o1).

4. Открыть выборку 1.

5. Обучить все нейросети.

6. Сохранить проект под тем же именем – имеем сохраненный проект с обученными сетями по выборке 1.

7. Открыть исходный проект, полученный при выполнении п. 2.

8. Сохранить проект под другим именем (например, n_o2).

9. Открыть выборку 2.

10. Обучить все нейросети.

11. Сохранить проект под тем же именем – имеем сохраненный проект с обученными сетями по выборке 2.

12. Провести перекрестное тестирование, для этого:

12.1. Открыть проект с обученными сетями по выборке 1 (см. п. 3).

12.2. Открыть выборку 2.

12.3. Протестировать по ней все нейросети.

12.4. Сохранить все результаты тестирования: предварительные результаты в виде текстовых файлов (сохранение из главного меню), сводные результаты в виде таблицы (примерный вид таблицы – табл. 3.1).

12.5. Открыть проект с обученными сетями по выборке 2 (см. п. 8).

12.6. Открыть выборку 1.

12.7. Протестировать по ней все нейросети.

12.8. Сохранить все результаты тестирования: предварительные результаты в виде текстовых файлов (сохранение из главного меню), сводные результаты в виде таблицы (примерный вид таблицы – табл. 3.1, можно в той же таблице, но отделив жирной линией). Результаты перекрестного тестирования можно представить в единой таблице: обученные по первой выборке, протестированные по второй – в первой части таблицы (первые N сетей до жирной линии), обученные по второй выборке, протестированные по первой – во второй части (N сетей после жирной линии). Жирная линия используется для удобства идентификации тестирования выборок.

Таблица 3.1. – Результаты тестирования

		Верно решенных примеров		Неверно решенных примеров		Средняя ошибка	Макс. ошибка
		кол-во	%	кол-во	%		
ОВ1 ↓ ТВ1	Сеть 1	194	74,05	68	25,95	1,52	5,09
	Сеть 2	177	67,56	85	32,44	1,65	7,44
	Сеть 3	185	70,61	77	29,39	1,60	5,50
	Сеть 4	191	72,90	71	27,10	1,55	5,47
ОВ2 ↓ ТВ2	Сеть 5	193	73,66	69	26,34	1,50	5,10
	Сеть 6	177	67,56	85	32,44	1,63	5,44
	Сеть 7	169	64,50	93	35,50	1,79	5,30
	Сеть 8	180	68,70	82	31,30	1,58	5,70
	Среднее	183,6	70,08	78,4	29,92	1,61	5,79

Примечание: в данных экспериментах ОВ1=ТВ2, ТВ1=ОВ2 – для обучения первых 4-х сетей используется выборка 1 (ОВ1), которая является тестовой для сетей 5–8 (ТВ2); выборка 2 используется только для обучения сетей 5–8 (ОВ2), соответственно для первых четырех сетей она используется в качестве тестовой (ТВ1).

Для перекрестного тестирования используются две выборки, полученные из общей совокупности данных. Самым простым критерием отбора является выделение четных/ нечетных записей. При этом формируется два подмножества: подмножество 1 (например, четные записи), подмножество 2 (нечетные записи). Использование данных подмножеств для перекрестного тестирования аналогично описанному для выборок (табл. 3.2).

Таблица 3.2. – Представление использования и деления выборок при перекрестном тестировании

Деление	Использование	
	Обучающая выборка	Тестовая выборка
Подмножество 1 (например, четные записи)	1	2
Подмножество 2 (например, нечетные записи)	2	1

Г. Дополнительные операции в свою очередь можно условно разделить на две группы: простейшие (определение значимостей параметров, вербализация) и операции, связанные с упрощением сети (контрастирование, бинаризация).

Г1. Определение значимостей параметров. После обучения можно провести определение показателей значимостей входных параметров (насколько каждый из них значим для прогноза выходного поля).

Показатели значимости определяют только для обученных сетей, поэтому логично определять их значения после обучения и для общей выборки. Для этого следует выполнить следующее:

1. Открыть проект с обученными сетями по общей выборке (вариант А, п. 3).

2. Определить значимости входных параметров для каждой сети (Нейросеть/Значимость входных сигналов сети).

3. Сохранить все значения для каждой сети в текстовых файлах.

Отметим, что данные показатели (значимость входных параметров), как правило, нет смысла определять, если при тестировании сеть показывает

менее 50 % правильно определенных примеров, т. к. в этом случае влияние параметров слабое или практически отсутствует. Тем не менее возможно конструирование более тонких настроек сети.

Рекомендуется дополнительно (помимо значений) определять ранги входных параметров для каждой сети. Это позволяет более адекватно проводить анализ влияния параметров на выходное поле. В большинстве случаев ранги совпадают со средними значениями значимостей параметров, но иногда отличаются. Среднее значение может быть небольшим (значимости усредняются во всем сетям), тем не менее ранг показывает большую значимость данного параметра. Ранг точнее определяет порядок следования параметров по степени их влияния на выходное поле.

Таблица 3.3. – Результаты определения значимостей входных параметров

	Сеть 1	Ранг 1	Сеть 2	Ранг 2	Сеть 3	Ранг 3	Сеть 4	Ранг 4	Среднее	Сумма рангов	Ранг
1	2	3	4	5	6	7	8	9	10	11	12
X9	**1,00**	**1**	**1,00**	**1**	**1,00**	**1**	**1,00**	**1**	**1,00**	4	**1**
X10	**0,43**	**4**	**0,36**	**2**	**0,36**	**2**	**0,38**	**2**	**0,38**	10	**2**
A	**0,53**	**3**	**0,21**	**3**	0,23	5	**0,18**	**3**	**0,29**	14	**3**
SL	0,04	9	0,18	5	0,22	6	0,17	4	**0,15**	24	**4**
E	0,05	8	0,16	6	0,23	5	0,16	5	**0,15**	24	**4**
LIT	0,03	10	0,18	5	0,23	5	0,16	5	**0,15**	25	5
SV	0,03	10	**0,19**	**4**	0,22	6	0,16	5	**0,15**	25	5
SU	0,09	5	0,14	8	**0,28**	**3**	0,09	9	0,15	25	5
WIN	0,08	6	0,15	7	**0,27**	**4**	0,10	8	0,15	25	5
AUT	0,07	7	0,14	8	**0,28**	**3**	0,10	8	0,15	26	6
ST	0,03	10	0,18	5	0,22	6	0,14	6	0,14	27	7

SPR	0,07	7	0,13	9	**0,27**	**4**	0,11	7	0,15	27	7
KR	0,02	11	0,18	5	0,21	7	0,14	6	0,14	29	8
AV	**0,66**	**2**	0,03	12	0,03	9	0,02	13	0,18	36	9
ST	0,03	10	0,05	10	0,04	8	0,05	10	0,04	38	10
X30	0,08	6	0,03	12	0,02	10	0,04	11	0,04	39	11
WN	0,05	8	0,04	11	0,03	9	0,04	11	0,04	39	11
TU	0,04	9	0,03	12	0,03	9	0,05	10	0,04	40	12
X14	0,03	10	0,03	12	0,04	8	0,04	11	0,04	41	13
MN	0,04	9	0,03	12	0,03	9	0,04	11	0,03	41	13
TH	0,04	9	0,04	11	0,03	9	0,03	12	0,03	41	13
ZAR	0,04	12	0,04	11	0,03	9	0,05	10	0,04	42	14
FR	0,04	9	0,04	11	0,02	10	0,03	12	0,03	42	14
X16	0,03	10	0,04	11	0,03	9	0,02	13	0,03	43	15
X2	0,02	11	0,02	13	0,03	9	0,04	11	0,03	44	16
X28	0,02	11	0,02	13	0,03	9	0,03	12	0,03	45	17
X27	0,02	11	0,02	13	0,03	9	0,02	13	0,02	46	18
X17	0,02	11	0,02	13	0,02	10	0,02	13	0,02	47	19
X29	0,01	12	0,03	12	0,02	10	0,02	13	0,02	47	19
X15	0,03	10	0,02	13	0,01	11	0,02	13	0,02	47	19
X12	0,01	12	0,02	13	0,01	11	0,02	13	0,02	49	20

Предлагается следующая методика для определения рангов по серии нейросетей: значения значимостей входных параметров упорядочиваем по убыванию (от больших к меньшим) для каждой сети, предварительно округлив значения до сотых. Расставляем ранг для каждого значения, начиная с максимального: параметру, имеющему наибольшее значение значимости (в NeuroPro это обычно 1) ставим в соответствие ранг 1, менее значимому – 2 и т.п. Одинаковые значения значимостей (с учетом округления) имеют одинаковые ранги. Проставив таким образом ранги для

каждой сети, для каждого параметра определяем суммарный ранг для серии сетей, который рассчитывается как сумма рангов каждой сети. Упорядочиваем параметры в соответствии с порядком возрастания суммарного ранга (от минимального значения суммарного ранга до максимального). Итоговый ранг по серии нейросетей проставляется в соответствии с возрастанием суммарного ранга: минимальная сумма рангов соответствует рангу 1, чуть больше минимального – 2 и т.п. (примерный вид таблицы для определения значимостей по 5 сетям – табл. 3.3)

Следует отметить, что ранги, в отличие от средних значимостей параметров, более адекватно использовать при анализе влияния параметров (по крайней мере, порядок следования значимых параметров по рангам точнее, чем по усредненным значениям значимостей).

Г2. Упрощение сети. Можно отдельно провести эксперименты по контрастированию (сокращению) внутренней либо внешней структуры сети. При этом последовательно удаляется один элемент сети (например, синапс, нейрон, неоднородный вход при контрастировании внутренней структуры) или входной сигнал (сокращение числа входных параметров), затем сеть доучивается. Если при этом качество обучения соответствует требуемому, то процесс сокращения числа элементов или входов повторяется, если же сеть не в состоянии доучиться, то процесс упрощения останавливается и элемент не удаляется. После окончания процесса контрастирования структура сети (при упрощении внутренней структуры) и число оставшихся входных сигналов остается неизменным. При дальнейшем обучении (по другой выборке данных) используется сокращенный набор параметров. Операции контрастирования после начала процесса (пользователь выбрал соответствующий пункт меню) выполняются нейроимитатором автоматически.

Для этого следует:

1. Открыть проект, полученный при обучении по одной из выборок (общей, выборок 1 или 2).
2. Сохранить проект под другим именем (например, n_ok, n_ok1 или n_ok2).
3. Открыть выборку.
4. Провести контрастирование для каждой нейросети.
5. Сохранить количество и процент (от общего количества) удаленных элементов в виде таблицы.
6. Сохранить проект под текущим именем.

После контрастирования можно провести тестирование или перекрестное тестирование аналогично приведенной методике с сохранением результатов (табл. 3.4).

Определение значимостей входных параметров можно проводить после процедуры контрастирования. При определении значимостей и рангов после

контрастирования внешней структуры (входных параметров) используется аналогичная методика (см. Г1). Но для параметров, вырезанных при контрастировании, значимости не вычисляются (в таблице значение 0). Таким параметрам ставится в соответствие, выбранный из рангов всех сетей максимальный ранг, увеличенный на 1. Для эффективного использования временных ресурсов рекомендуется сразу вырезать из таблицы со значимостями после контрастирования входные параметры, имеющие нулевое среднее значение показателей значимости по всем сетям (примерный вид таблицы для 5 сетей после контрастирования – табл. 3.5).

Таблица 3.4. – Результаты перекрестного тестирования после контрастирования

	Удалено входов		Верно решенных примеров		Неверно решенных примеров		Сред-няя ошиб-ка	Макс. ошиб-ка
	кол-во	%	кол-во	%	кол-во	%		
Сеть 1	28	62,22	183	69,85	79	30,15	1,67	5,87
Сеть 2	37	82,22	145	55,34	117	44,66	2,14	9,40
Сеть 3	35	77,78	153	58,40	109	41,60	2,03	9,54
Сеть 4	30	66,67	170	64,89	92	35,11	1,76	6,68
Сеть 5	28	62,22	175	66,79	87	33,21	1,70	5,70
Сеть 6	30	66,67	174	66,41	88	33,59	1,78	6,16
Сеть 7	21	46,67	182	69,47	80	30,53	1,73	9,48
Сеть 8	32	71,11	166	63,36	96	36,64	1,79	7,02
Среднее	30,6	68,00	168	64,12	94	35,88	1,84	7,87

Таблица 3.5. – Результаты определения значимостей входных параметров после контрастирования (фрагмент для 4 из 10 сетей)

	Сеть 1	Ранг 1	Сеть 2	Ранг 2	Сеть 3	Ранг 3	Сеть 4	Ранг 4	Среднее	Сумма рангов	Ранг
1	2	3	4	5	6	7	8	9	10	11	12
X30	**1,00**	**1**	0	17	**0,79**	**3**	0,57	6	**0,59**	27	**1**
X28	**0,75**	**4**	0	17	0,61	7	**0,75**	**3**	**0,53**	31	**2**
FR	**0,97**	**2**	0	17	0,46	14	0,32	12	**0,44**	45	8
ST	**0,79**	**3**	0	17	**0,75**	**4**	0	17	**0,39**	41	5
TU	0	17	**0,30**	**4**	**0,75**	**4**	0,43	9	0,37	34	**3**
WIN	0	17	**0,31**	**3**	**0,88**	**2**	0,28	14	0,37	36	**4**
X11	0	17	0	17	0,45	15	**1,00**	**1**	0,36	50	10
X20	0,62	5	0	17	0,71	5	0	17	0,33	44	7
STU	0,53	6	0,28	6	0,46	14	0	17	0,32	43	6
X4	0	17	0	17	0,58	9	0,63	5	0,30	48	9
LIT	0	17	0	17	0,58	9	0,48	8	0,26	51	11
SL	0	17	0	17	**1,00**	**1**	0	17	0,25	52	12
X6	0	17	**1,00**	**1**	0	17	0	17	0,25	52	12
KR	0	17	0	17	0,60	8	0,34	11	0,23	53	13
X23	0	17	0	17	0	17	**0,82**	**2**	0,21	53	13
WN	0	17	**0,47**	**2**	0,30	16	0	17	0,19	52	12
X22	0	17	0,27	7	0	17	0,49	7	0,19	48	9
MN	0	17	0	17	0	17	**0,67**	**4**	0,17	55	14
AU	0	17	0	17	0,64	6	0	17	0,16	57	15
TH	0	17	0,29	5	0	17	0,30	13	0,15	52	12

SU	0	17	0	17	0,53	10	0	17	0,13	61	17
AN	0	17	0	17	0,52	11	0	17	0,13	62	18
ZAR	0	17	0	17	0,49	12	0	17	0,12	63	19
X18	0	17	0	17	0,48	13	0	17	0,12	64	20
X2	0	17	0	17	0,46	14	0	17	0,11	65	21
SVA	0	17	0	17	0,45	15	0	17	0,11	66	22
X19	0	17	0	17	0	17	0,38	10	0,10	61	17
X16	0,38	7	0	17	0	17	0	17	0,09	58	16
X8	0	17	0	17	0	17	0	17	0,00	68	23

Дополнительно можно провести эксперименты по бинаризации весов синапсов (при сохранении в название проекта вставить букву b). Порядок выполнения аналогичен описанию методики контрастирования. При бинаризации (приведении весов синапсов к заданным значениям) данный процесс происходит сходным контрастированию образом: последовательно бинаризуется отдельный синапс, сеть доучивается. Если качество обучения удовлетворительное – процесс повторяется, если же нет – остановка алгоритма. В ходе дообучения структура сети после бинаризации сохраняется, т. е. бинаризованные синапсы остаются бинаризованными и их веса не меняются при повторном обучении.

Часто в практических исследованиях используют дополнительные операции совместно. Например, определяют минимальный набор входных параметров, необходимый для решения задачи (для этого используют контрастирование входных параметров) и значимости входных параметров. Это полезно для сравнения и анализа результатов, полученных до и после процедур упрощения.

Г3. Вербализация. Можно привести графическое представление структуры нескольких нейросетей (текстовое описание можно получить через пункт «Вербализация» с сохранением всех файлов), полученных после контрастирования и бинаризации.

Вербализация трактуется как процесс восстановления симптом-синдромной структуры понятий предметной области. Входные сигналы сети являются входными симптомами, выходные сигналы нейронов первого слоя сети – синдромами первого уровня и одновременно входными симптомами для нейронов второго слоя, генерирующих синдромы второго уровня, и т. д.

Приведем пример вербального описания для одной сети, полученной после контрастирования.

Поля базы данных (исходные симптомы):

ELEC
KRAN
SVAR
FR
X9
X10
X11
X22

Поля базы данных (конечные синдромы):

X6

Предобработка входных полей БД для подачи сети:

ELEC=(ELEC-0,5)/0,5

KRAN=(KRAN-0,5)/0,5

SVAR=(SVAR-0,5)/0,5

FR=(FR-0,5)/0,5

X9=(X9-10,1)/10,1

X10=(X10-4,55)/4,55

X11=(X11-4,5)/4,5

X22=(X22-36,3)/36,3

Функциональные преобразователи:

Сигмоида1(A)=A/(0,1+|A|)

Сигмоида2(A)=A/(0,1+|A|)

Сигмоида3(A)=A/(0,1+|A|)

Синдромы 1-го уровня:

Синдром1_1=Сигмоида1(ELEC-0,16*KRAN-0,31*SVAR+0,06*X22)

Синдром1_2=Сигмоида1(0,24*X11)

Синдром1_3=Сигмоида1(0,36*FR-0,37*X9-0,10*X10)

Синдромы 2-го уровня:

Синдром2_1=Сигмоида2(0,10*Синдром1_1-0,33*Синдром1_3)

Синдром2_2=Сигмоида2(-0,22*Синдром1_1+0,36*Синдром1_2)

Синдромы 3-го уровня:

Синдром3_1=Сигмоида3(0,50*Синдром2_1+0,42*Синдром2_2)

Синдром3_2=Сигмоида3(0,43*Синдром2_1)

Синдром3_3=Сигмоида3(0,11*Синдром2_1)

Конечные синдромы:

X6=0,45*Синдром3_1+0,07*Синдром3_2-0,52*Синдром3_3-0,42

Постобработка конечных синдромов:

X6=((X6*33,70)+33,70)/2)

Первые пункты вербального описания («поля базы данных») содержат перечень входных и выходных полей файла данных. Предобработка описывает формулы для нормирования значений каждого входного параметра. Далее представлена структура нейронной сети, отдельно для каждого слоя: «синдромы 1-го уровня» описывают связи между входными сигналами (значениями входных полей) и 1-м слоем нейронов (какие входные параметры подаются на вход каждого нейрона в слое); «синдромы 2-го уровня» – связь нейронов 1 слоя и 2 слоя; «синдромы 3-го уровня» – нейронов 2 и 3 слоев.

Пункт «конечные синдромы» представляет собой описание синдромов определенного уровня, которые используются для вычисления выходного поля. Следует учитывать, что при использовании некоторых видов оценки интерпретатор ответа способен оценить уверенность нейронной сети в полученном ответе. Вектор уверенности сети в ответах (для каждого ответа своя уверенность) может оказаться полезным для пользователя. Каждый элемент вектора уверенности в ответе является действительным числом от нуля до единицы. Таким образом, значение «конечных синдромов» представляет собой действительное число в указанном диапазоне. Это не всегда удобно для конечного пользователя, так как в большинстве задач требуется получить значения выходного поля со своим диапазоном изменения. Например, требуется обучить предиктор для вычисления температуры пациента, которая в норме составляет 36,6°. Для вывода конечному пользователю используется «интерпретация ответа» («постобработка»), т. е. перевод «нормированных» значений выходного поля в значения из исходного диапазона (до предобработки данных). Диапазон изменения температуры составил, после нормирования составляет [0, 1] (диапазон может быть другим в зависимости от способа нормирования) – после постобработки температура может изменяться от 36 до 41 (рис. 15).

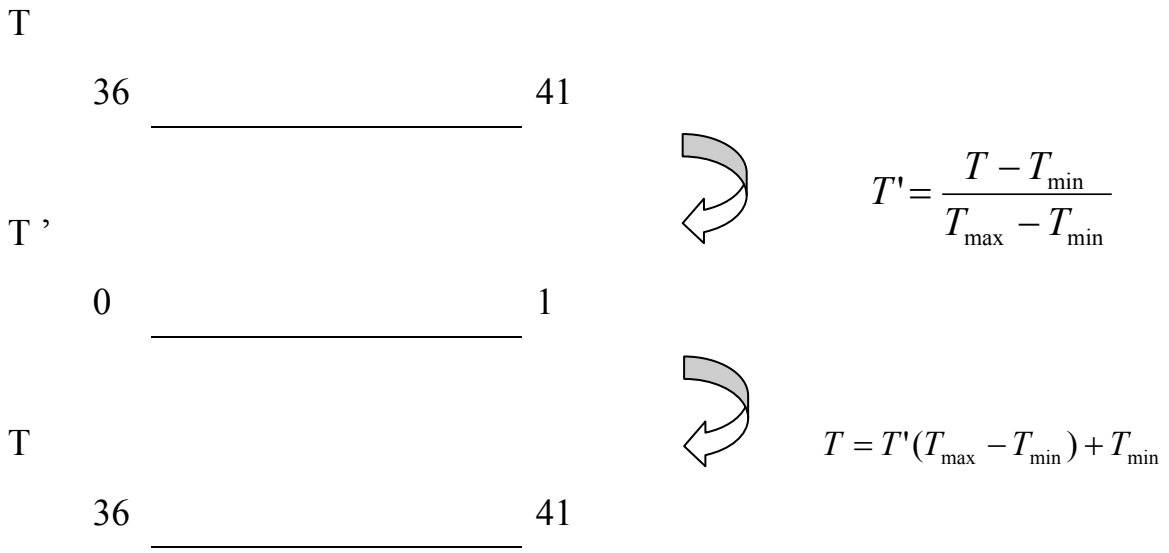


Рис. 15. Схема последовательных преобразований исходных значений

T – температура пациента (исходный диапазон), T’ – после нормирования, T – преобразование ответа нейроимитатора из диапазона [0, 1] (или другого в зависимости от способа нормирования) T’ в исходный диапазон.

Пользователь, последовательно проходя слой за слоем, может на основе описания входных симптомов слоя и их группировки по нейронам давать некоторые осмысленные именования и интерпретации синдромов, генерируемым этим слоем.

Операции контрастирования и бинаризации не являются обязательными для выполнения. Строго говоря, и перекрестное тестирование не является обязательным. Эти операции служат для проверки качества обучения сетей (тестирование) и упрощения структуры сети (контрастирование, бинаризация).

Выводы:

Приведена обобщенная технология использования нейронных сетей с учителем для решения различных прикладных задач. Описана классификация операций и групп операций, традиционно применяемых исследователями. На основе абстрактной технологии предложена частная методика, предназначенная прежде всего для начинающих исследователей. Приводятся основные операции и простейшие задачи, реализация которых позволяет достаточно быстро и легко получить предварительные результаты для конкретных данных при использовании готового нейроимитатора.

3.2. Особенности технологии работы с Neurogenesis

Программный комплекс Neurogenesis 1.0 (свидетельство РОСПАТЕНТ №2005611168) предназначен для выявления функциональных зависимостей в данных представленных в числовой форме в виде таблиц. Программный комплекс работает под управлением MS Windows NT/2000/XP. Функционирование Neurogenesis выполняется в режиме сервера. Поэтому далее в тексте он будет упоминаться как нейросервер. Neurogenesis может применяться для обработки статистических данных в медицине, экологии, психологии, социологии, физике, при разведке месторождений полезных ископаемых, прогнозе поведения бирж и др. Выполняется автоматическая генерация моделей описывающих данные в форме ИНС с учителем. При генерации программа самостоятельно определяет оптимальную структуру и параметры модели. Оптимизация структур ИНС осуществляется эволюционным алгоритмом SIF, описанным в главе 2.2.3. Для ускорения вычислений, можно задействовать несколько компьютеров объединенных в сеть. Реализовано использование возможностей многопроцессорной обработки. Предлагается текст, следующие авторской версии указаний по работе с нейроимитатором.

Программный комплекс состоит из следующих модулей верхнего уровня декомпозиции:

1. Клиент нейросервера;
2. Обработчик запросов;
3. База примеров;
4. База моделей;
5. Планировщик заданий;
6. Исполнитель заданий;
7. Нейронная сеть;
8. Предобработчик.

Модель потоков данных (рис. 16) показывает процесс взаимодействия модулей.

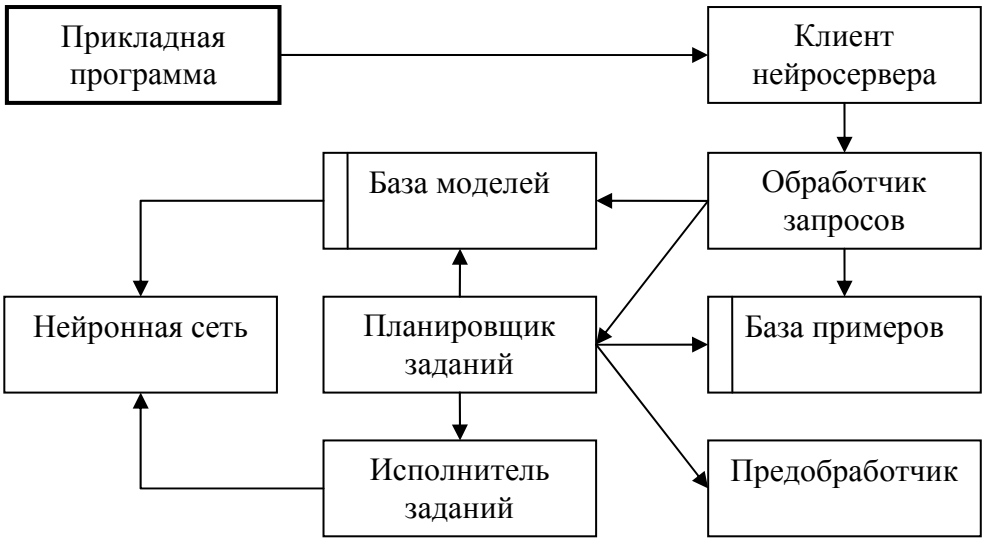


Рис. 16. Модули верхнего уровня декомпозиции и направление потоков данных. Стрелками указаны направления обмена сообщениями между модулями.

Прикладная программа и “Клиент нейросервера”

Нейросервер не имеет пользовательского интерфейса. Взаимодействие с ним осуществляется из прикладных программ через обращение к методам “Клиента нейросервера”. “Клиент нейросервера” реализован в виде ActiveX-объекта и может быть вызван из любой среды поддерживающей технологию ActiveX. В частности все эксперименты проводились с использованием MS Excel. “Клиент нейросервера” транслирует вызовы прикладной программы в XML-документы и отправляет их “Обработчику запросов”. Ответ от “Обработчика запросов” также приходит в виде XML-документа, но от прикладной программы скрываются сложности стандарта XML. При необходимости можно обмениваться сообщениями с “Обработчиком запросов” напрямую. Форматы XML-документов приведены в приложении Б.

“Обработчик запросов”

Модуль служит для первичной обработке запросов, идентификации, аутентификации и вызова методов других модулей. Под первичной обработкой запроса подразумевается выделения из XML-документа идентификатора запрашиваемой процедуры, ее параметров, имени пользователя и пароля.

“База примеров”

Модуль организует хранение и доступ к примерам. Также хранятся сведения о проектах. Примеры сгруппированы по проектам. Модуль

позволяет добавлять, удалять и запрашивать примеры. Обращение к примерам осуществляется либо по коду проекта, либо по коду примера, назначаемого прикладной программой. Код примера имеет строковой формат и может содержать в себе различные сведения. Например, дату и источник поступления примера. Способ форматирования строки кода зависит от прикладной программы.

“База моделей”

Модуль организует хранение и доступ к нейросетевым моделям. Модель состоит из набора ИНС и описания способа предобработки входных значений. Также имеются сведения о надежности модели, полученные в ходе моделирования. С каждым проектом связано не больше одной модели. Проект может не иметь модели, если процесс моделирования еще не выполнялся или он выполняется в первый раз и не завершен. По требованию прикладной программы модель может быть выбрана из базы и вычислен ответ ИНС на основании предъявленных входных данных. При этом вычисление ИНС проводится на стороне нейросервера. Прикладной программе передается только ответ ИНС. Таким образом, прикладной программе не требуется реализовывать процедуры анализа структуры и параметров ИНС.

“Предобработчик”

Модуль реализует функции предобработки примеров. Предобработка включает в себя заполнение пробелов в данных, нормализацию и группировку (в частности, кластеризацию) примеров. Все шаги предобработки выполняются автоматически. Для предобработки применяются методы гарантированно не ухудшающие результаты моделирования. Поэтому, о более специфичной предобработке должна позаботиться прикладная программа.

“Планировщик заданий”

Схема функционирования модуля согласуется со схемой эволюционного алгоритма описанного в главе 2.3.3. Также модуль формирует задания на эволюцию структур, обучение и тестирование. Критерием селекции порождаемых нейросетевых моделей является ошибка тестирования. На структуру ИНС накладываются ограничения начального и максимального количества синапсов в ИНС. Задания распределяются по “Исполнителям заданий”.

“Исполнитель заданий”

Модуль устанавливается на каждом компьютере, задействованном в моделировании. “Исполнитель заданий” выполняет задания, поступившие от “Планировщика заданий”. Задание включает в себя одну (если операция

мутации) или две (если операция скрещивания) ИНС и набор примеров. “Исполнитель заданий” создает отдельный параллельный процесс, в котором производит эволюционную операцию над структурой ИНС, обучение, контрастирование и тестирование. Функционирование на нескольких компьютерах и поддержка многопоточности, реализуют режим параллельных вычислений. Обучение производится методом сопряженных градиентов и стохастическими методами. Метод обучения выбирается автоматически. Тестирование осуществляется методом скользящего контроля.

“Нейронная сеть”

Модуль “Нейронная сеть” реализует базовые операции нейровычислений. Реализованы операции вычисления прямых сигналов ИНС, вычисления обратных сигналов ИНС, эволюционной модификации и контрастирования структуры. Для ускорения нейровычислений используются SSE-инструкции.

Установка Neurogenesis

Для установки Neurogenesis следует запустить программу установки Setup.exe. Пользователю будет предложено ввести следующие параметры:

1. Папку, в которой будут располагаться файлы Neurogenesis. Если папка не существует, она будет создана;
2. TCP-порт обработчика запросов;
3. TCP-порт планировщика заданий;
4. TCP-порт исполнителя задания.

За исключением папки размещения остальные параметры можно будет в последствии изменить. При установке будут зарегистрировано два сервиса операционной системы (NeuroServer и Mutator) и зарегистрированы ActiveX – библиотеки клиента нейросервера. В общем случае доступ к нейросерверу может быть осуществлен с любого компьютера в вычислительной ИНС. Для установки модуля клиента нейросервера на компьютер, работающий под управлением операционной системы MS Windows NT/2000/XP достаточно зарегистрировать ActiveX – библиотеку NeuroServerClient.dll. Это можно сделать из командной строки командой regsvr32 NeuroServerClient.dll. В случае другой операционной системы придется организовать непосредственный обмен XML-сообщениями (приложение А).

Neurogenesis позволяет выполнять параллельную обработку данных. В частности, можно подключать неограниченное количество дополнительных

компьютеров. Для подключения дополнительного компьютера необходимо скопировать на него в одну папку файлы specimen.dll, mutatorservice.exe и addcomp.exe. Затем следует выполнить addcomp.exe и в диалоге указать TCP-порт исполнителя заданий, TCP-порт планировщика заданий и параметры регистрации сервиса операционной системы. При установке будет зарегистрирован сервис операционной системы Mutator и создан настроечный файл mutator.ini. После этого необходимо зарегистрировать компьютер. Для этого необходимо на основном компьютере выполнить программу regcomp.exe указав в диалоге сетевое имя компьютера (или его IP-адрес) и TCP-порт исполнителя заданий выбранный при выполнении addcomp.exe. После этого необходимо перезапустить сервис NeuroServer. Если на момент запуска сеанса обработки данных дополнительный компьютер не будет доступен, то он не будет использоваться. Необходимо сказать, что в случае “зависания“ дополнительного компьютера или отсутствия с ним связи во время сеанса обработки данных, обработка данных будет приостановлена. Для возобновления обработки необходимо либо восстановить связь с компьютером и выполнить перезапуск сервиса NeuroServer, либо просто перезапустить сервис NeuroServer. Во втором случае недоступный компьютер не будет использоваться в текущем сеансе обработки данных.

Управление настройками программного комплекса можно осуществляется через изменение файлов настроек. Всего имеется три файла настроек: neuroserver.ini, settings.ini и mutator.ini.

Файл neuroserver.ini содержит настройки связи с обработчиком запросов, параметры моделирования и размещения базы моделей. Ниже перечислены параметры, содержащиеся в нем:

1. PORT – TCP-порт обработчика запросов;
2. READ_TIMEOUT – предельное время ожидания (в миллисекундах) данных от клиента нейросервера;
3. DIR_SAMLES – папка в которой хранятся сведения о системах классификации, примеры, правила их предобработки и модели;
4. VENDOR – обработчик XML-документов зарегистрированный в операционной системе;
5. MAX_ITER_CLUSTERING – максимальное количество итераций процедуры кластеризации примеров;
6. MIN_SAMPLE_CLUSTER – минимальное количество примеров попадающих в один кластер;
7. MAX_VARIANT – максимальное количество вариантов структур ИНС в популяции эволюционного алгоритма;

8. VALID_PERC – процент примеров от общего числа примеров, используемый на одном шаге тестирования методом скользящего контроля. В нейросервере реализован модифицированный метод скользящего контроля. В классическом варианте на каждом шаге тестируется один пример. В реализованном варианте на каждом шаге тестирования случайно выбираются несколько тестовых примеров. Если 0, то на каждой шаге тестируется один пример, как в классическом варианте;
9. TEST_COUNT – количество шагов теста методом скользящего контроля;
10. MAX_SYNAPSE – максимальное количество синапсов в ИНС;
11. START_SYNAPSE – количество синапсов в начальных вариантах ИНС;
12. PRIORITY – системный приоритет процесса обработки запросов (0 - нормальный; 1 -высокий; 2 - низкий).

Файл settings.ini содержит настройки планировщика заданий и алгоритмов обучения ИНС. Ниже перечислены параметры, содержащиеся в нем и которые могут меняться пользователем:

1. TCP_LOCAL_PORT – TCP-порт планировщика заданий;
2. DB_DATABASE_NAME – папка размещения файлов планировщика заданий;
3. TRAINING_MAX_ITER – максимальное число итераций процедуры обучения ИНС;
4. TRAINING_MIN_H – минимальное значение шага изменения весов синапсов в процедуре обучения ИНС;
5. TRAINING_START_H - стартовое значение шага изменения весов синапсов в процедуре обучения ИНС.

Файл mutator.ini содержит настройки исполнителя заданий. Ниже перечислены параметры, содержащиеся в нем:

1. LocalPort – TCP-порт обработчика заданий;
2. RemotePort – TCP-порт планировщика заданий;
3. RemoteHost – сетевой идентификатор компьютера, на котором исполняется планировщик заданий;
4. Priority – системный приоритет процесса обработки запросов (0 - нормальный; 1 -высокий; 2 - низкий);

5. ServiceName – используемое наименование сервиса операционной системы;
6. DisplayName – отображаемое наименование сервиса операционной системы.

В случае, если на компьютере установлено несколько процессоров, то рекомендуется установить на компьютер столько же исполнителей заданий. Для этого необходимо несколько раз выполнить addcomp.exe всякий раз указывая различные порты и настройки сервиса операционной системы.

Взаимодействие с Neurogenesis

Взаимодействие прикладных программ с нейросервером осуществляется путем обмена XML-сообщениями по TCP/IP протоколу. Для упрощения программирования можно использовать ActiveX- компоненты клиента нейросервера, скрывающие детали сетевого протокола и форматов сообщений. В последующих разделах описаны компоненты и их функции.

Типичный вариант использования нейросервера:

1. Установить и проверить параметры соединения с нейросервером с помощью компонента Connection;
2. Создать систему классификации с помощью компонента Samples;
3. Добавить примеры классификации с помощью компонента Samples;
4. Активизировать процесс моделирования с помощью компонента Model;
5. Дождаться завершения процесса моделирования и выполнить тестирование модели на тестовых примерах с помощью компонента Model.

Connection

Компонент служит для организации связи с нейросервером. Используется остальными компонентами.

Метод Login выполняет установку параметров соединения с нейросервером. В качестве параметров передаются имя пользователя (User), пароль (Password), сетевой идентификатор компьютера, на котором выполняется обработчик запросов (Host), порт обработчика запросов (Port) и максимальное время ожидания ответа от нейросервера в миллисекундах (ReadTimeOut). В текущей реализации нейросервера разграничение прав

доступа не выполнено. По этому в качестве имени пользователя и пароля можно передавать пустые строки. Метод ничего не возвращает.

Check выполняет проверку корректности параметров переданных в метод Login. Результат проверки возвращается в виде логической величины.

GetServerVersion возвращает строку с описанием версии нейросервера.

Samples

Компонент служит для регистрации задач классификации и хранения примеров.

Метод Login устанавливает параметры соединения с нейросервером. На входе метод получает компонент класса Connection.

Нейросервер строит модели в виде нейросетевых классификаторов с учителем. Метод NewSys регистрирует новую систему классификации. На входе метод принимает код системы классификации (SysId), количество классов (ClassCount), количество признаков классификации (PropCount) и параметр, определяющий способ указания веса примера (WhoWeighted). Если WhoWeighted равен 0, то вес примеров указывается на уровне класса, если 1, то вес указывается каждому примеру отдельно. Код системы классификации должен быть уникальным. Классы нумеруются последовательно от 0 до ClassCount -1.

Метод DelSys удаляет систему классификации. При этом удаляются все связанные с системой примеры, правила предобработки примеров и модели. На входе метод принимает код удаляемой системы классификации (SysId).

Метод SetClassWeight устанавливает вес всем примерам указанного класса. На входе метод получает код системы классификации (SysId), номер класса (ClassNum) и вес примеров класса (ClassWeight). Вес должен иметь положительное значение. По умолчанию веса всех примеров равны 1.

GetClassWeight возвращает вес примеров указанного класса. На входе метод получает код системы классификации (SysId) и номер класса (ClassNum).

Примеры добавляются методом AddSample. На входе метод получает код системы классификации (SysId), номер класса примера (ClassNum), метку примера (Mark), вес примера (Weight) и массив значений признаков классификации (Values). Метка примера представляет собой произвольную строку в кодировке ANSI, содержащей до 100 символов. Метка может быть не уникальна. При передаче метки в метод, по требованию технологии ActiveX, строка метки перекодируется в формат UNICODE. Если некоторый признак классификации неизвестен, то соответствующая позиция массива Values должна быть установлена в NULL.

Получить ранее добавленные примеры можно с помощью метода GetSamples. Метод на входе принимает код системы классификации (SysId),

а на выходе массив массивов значений признаков классификации (X), массив номеров классов примеров (Y) и массив меток примеров (Mark).

Можно также получить только массив меток с помощью метода GetMarks. На вход метода необходимо передать код системы классификации (SysId). На выходе получается массив меток примеров (Marks).

Удалять примеры можно с помощью метода DelSample. На входе методу передаются код системы классификации (SysId) и метка примеров. Удаляются все примеры, связанные с системой классификации и меткой.

Для контроля ошибок существуют два метода GetErrorMessage и GetExceptionClassName. GetErrorMessage возвращает сообщение об ошибке, а GetExceptionClassName наименование класса исключения. В случае отсутствия ошибок методы возвращают пустые строки.

Model

Компонент служит для управления процессом моделирования и использования созданных моделей. Модели позволяют по набору признаков предсказывать классы объектов моделирования.

Метод Login устанавливает параметры соединения с нейросервером. На входе метод получает компонент класса Connection.

Активизация процесса моделирования осуществляется вызовом FindModel. На входе метод получает код системы классификации (SysId). Если модель уже существует, ее можно пользоваться во время процесса моделирования. По окончанию процесса моделирования существующая модель будет заменена новой моделью. Если на момент вызова метода уже выполняется другой процесс моделирования, то задача моделирования будет поставлена в очередь на выполнение.

Получить сведения о состоянии процесса моделирования можно с помощью метода GetModelingState. На входе метод получает код системы классификации (SysId). На выходе метод возвращает строку, характеризующую состояние процесса моделирования: ”ANALYSIS” – выполняется анализ задачи моделирования; ”SYNTHESIS” – выполняется синтез нейросетевых моделей; “COMPLETE” – процесс моделирования завершен; “NONE” – моделирование не выполнялось.

Остановить процесс моделирования с потерей всех промежуточных результатов можно с помощью метода StopFindModel. На входе метод получает код системы классификации (SysId).

Приостановить все процессы моделирования с сохранением промежуточных результатов можно с помощью метода PauseFindModel. Возобновить процессы моделирования можно методом ContinueFindModel.

Оба метода не имеет параметров, и воздействуют на все задачи моделирования.

После завершения моделирования можно предсказывать классы моделируемых объектов по набору признаков классификации. Для этого необходимо использовать метод GetClassNum. На входе метод принимает код системы классификации (SysId) и массив значений признаков классификации (Values). Если некоторый признак классификации неизвестен, то соответствующая позиция массива Values должна быть установлена в NULL.

Для контроля ошибок существуют два метода GetErrorMessage и GetExceptionClassName. GetErrorMessage возвращает сообщение об ошибке, а GetExceptionClassName наименование класса исключения. В случае отсутствия ошибок методы возвращают пустые строки.

Computers

Компонент служит для управления списком компьютеров задействованных в вычислениях. Чем больше используется компьютеров, тем больше у нейросервера возможностей для распараллеливания вычислений при создании нейросетевых моделей. На каждом используемом компьютере должен быть установлен хотя бы один модуль ”Исполнитель заданий”. Рекомендуется на один компьютер устанавливать столько модулей ”Исполнитель заданий”, сколько имеется процессоров.

Метод Login устанавливает параметры соединения с нейросервером. На входе метод получает компонент класса Connection.

После выполнения метода Login необходимо выполнить метод Refresh. Только после этого можно обращаться к остальным методам.

GetCount возвращает количество используемых модулей ”Исполнитель заданий”.

AddComputer вносит в список информацию о компьютере и установленном на нем модуле ”Исполнитель заданий”. На входе методу передается сетевой идентификатор компьютера (Host) и TCP-порт (Port), через который модуль ”Исполнитель заданий” получает задания от модуля ”Планировщик заданий”. Также можно использовать утилиту regcomp (см. инструкции по установке). Сведения об используемых компьютерах зачитываются однократно при старте процесса моделирования. Поэтому, если важно немедленно задействовать дополнительные вычислительные ресурсы, необходимо последовательно вызвать методы PauseFindModel и ContinueFindModel компонента Model.

Перед выполнением метода AddComputer можно проверить правильность параметров с помощью метода CheckComputer. На входе метод

получает сетевой идентификатор компьютера (Host) и TCP-порт (Port) модуля ”Планировщик заданий”. Метод возвращает логическое значение, характеризующее корректность параметров. В случае, если метод вернет “ложь”, то значит в данный момент времени связь с модулем ”Исполнитель заданий” с указанными параметрами невозможна, например, по причине отсутствия связи с компьютером.

DelComputer удаляет сведения об используемом компьютере. На входе метод получает сетевой идентификатор компьютера (Host). Удаляются все сведения связанные с этим компьютером.

Метод GetDescription возвращает сведения об используемом компьютере и модуле ”Исполнитель заданий”. На входе метод получает номер в списке (Index). Нумерация в списке ведется от 0 по GetCount – 1. Метод возвращает сетевой идентификатор компьютера (Host) и TCP-порт (Port) модуля ”Исполнитель заданий”.

Для контроля ошибок существуют два метода GetErrorMessage и GetExceptionClassName. GetErrorMessage возвращает сообщение об ошибке, а GetExceptionClassName наименование класса исключения. В случае отсутствия ошибок методы возвращают пустые строки.

3.3 Особенности технологии работы с SNN

Нейроимитатор Statistica Neural Networks обеспечивает возможность работы с несколькими типами нейронных сетей: сигмоидных, Кохонена, вероятностных, RBFN и других.

Предлагается простое описание работы с нейроимитатором STATISTICA Neural Networks в виде описания действий, как правило, иллюстрированных копиями экрана.

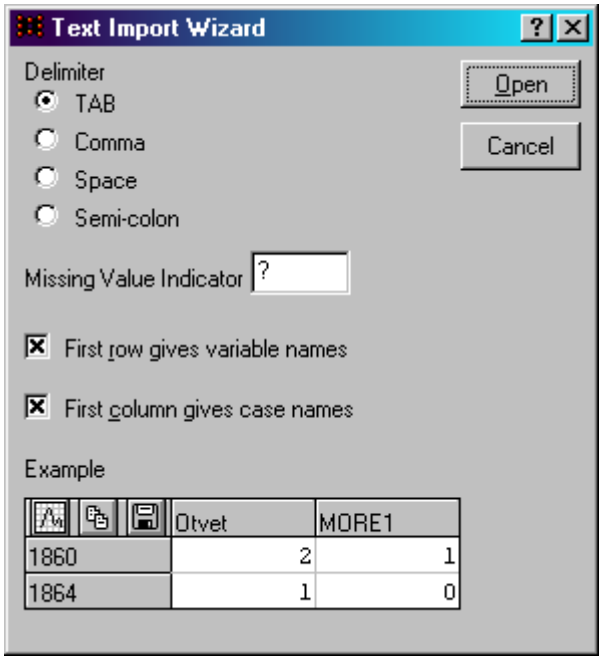
3.3.1 Пример 1

1. Конвертация файла Election.db:

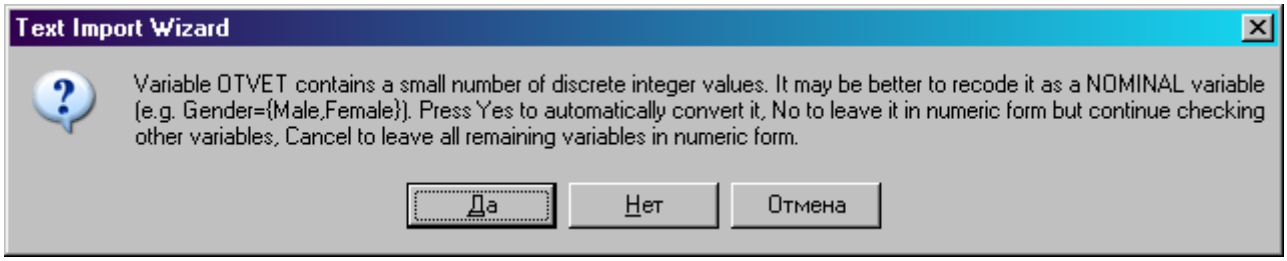
а) Откроем в программе Excel базу данных

б) Сохраним ее как текстовый файл с разделителем табуляцией

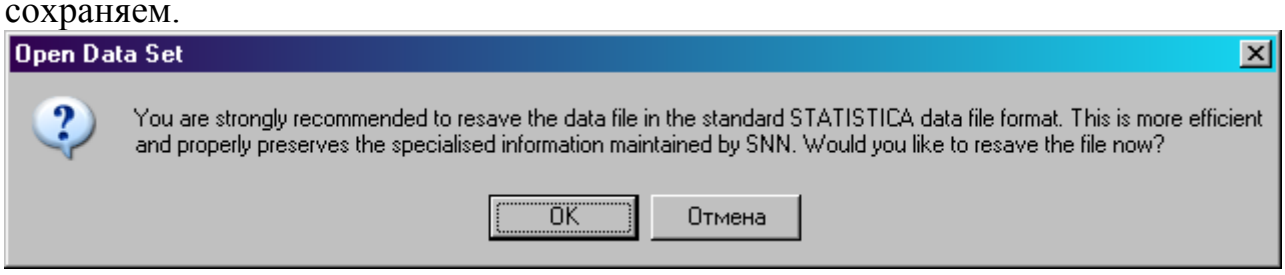
2. Запускаем нейроимитатор STATISTICA Neural Networks и открываем базу данных в текстовом файле. В появившемся после нажатия кнопки «Открыть» окне указываем вид разделителя – табуляция, и отметим что первую строку считать заголовком.



Запрос о переводе данных из текстового в числовой формат., нажимаем «нет»

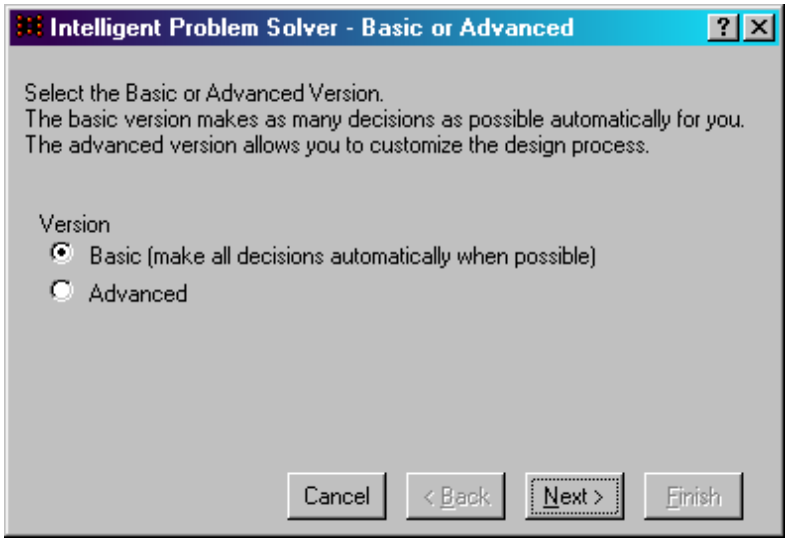


Запрос о пересохранении данных во внутренний формат статистики, сохраняем.

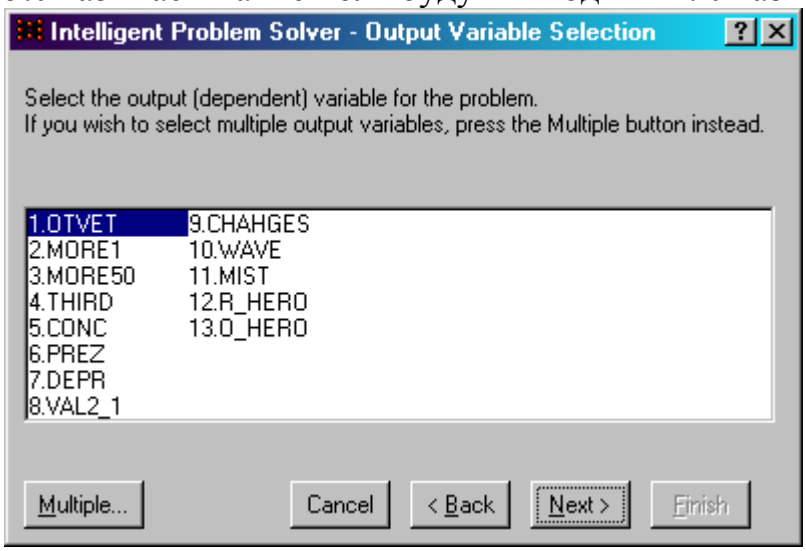


3. Данные были разделены на три выборки: тестовую (в окне “Data set editor” эти строки выделены синим цветом и не используются при обучении, необходимы для независимого тестирования), данные для проверки (Строки выделены красным, необходимые для корректировки ошибки в процессе обучения) и обучающую выборку (выделены черным).

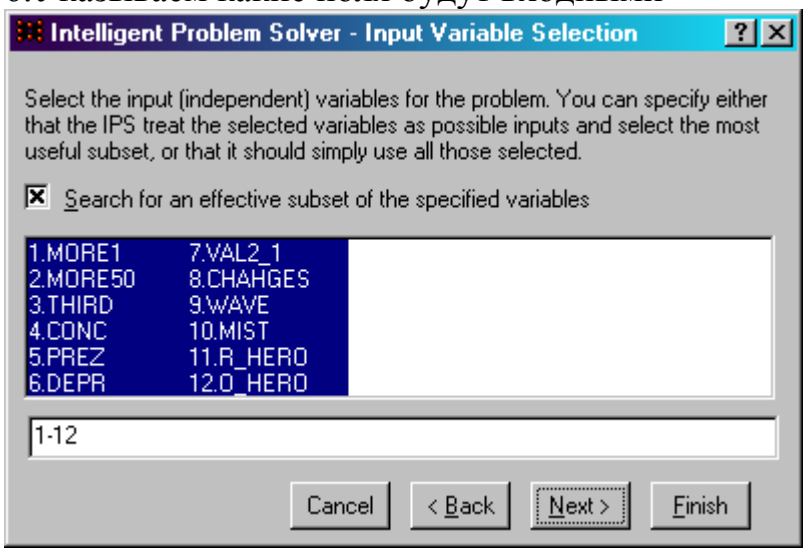
4. Запускаем мастера решения задач. Способ решения задачи - базовый



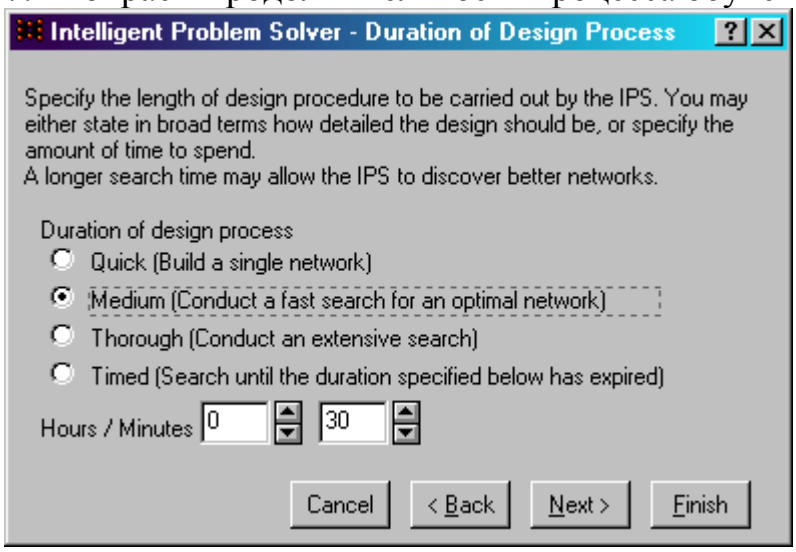
5.Указываем какие поля будут выходными. Указываем поле “Otvet”.



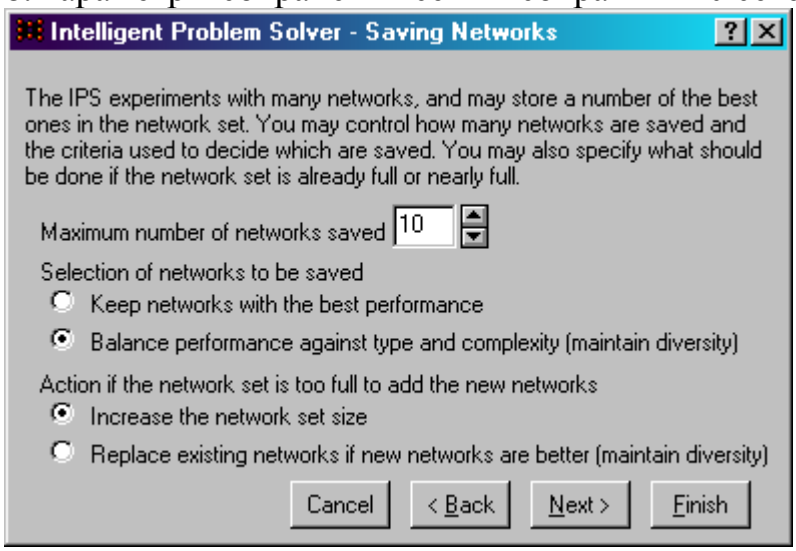
6.Указываем какие поля будут входными



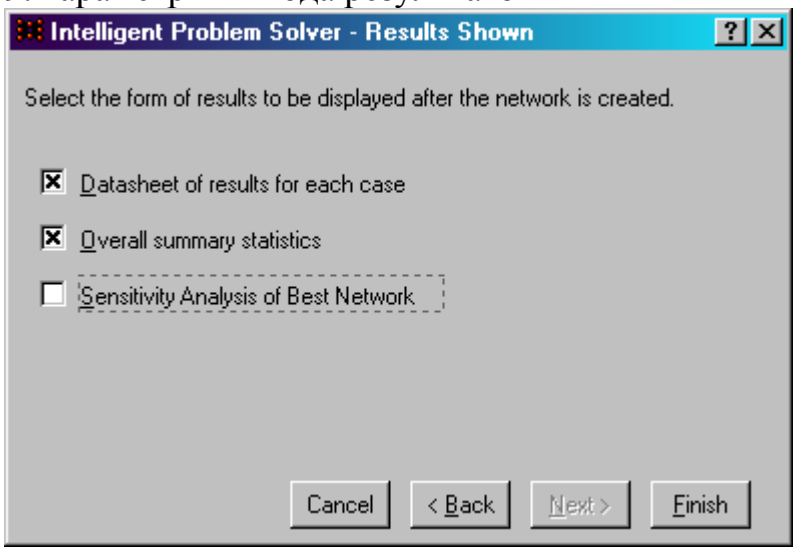
7.Выбираем продолжительность процесса обучения – средняя.



8.Параметры сохранения сети – сохранять 10 сетей.



9.Параметры вывода результатов

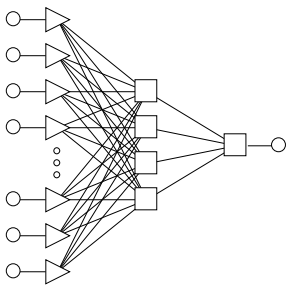


Нажимаем «Finish».

10.В окне network Set Editor смотрим информацию о типах полученных сетей, кол-ве входных сигналов, максимальной ошибке и тд.

	Type	Error	Inputs	Hidden	Performance
01	RBF	0.5316963	10	1	1.009487
02	RBF	0.4725896	10	1	0.8559816
03	MLP	0.3540201	3	4	0.7215794
04	Linear	0.286078	6	-	0.4827206
05	Linear	0.2569745	7	-	0.4302589
06	MLP	0.2457205	6	1	0.4319629
07	Linear	0.2377385	8	-	0.4368305
08	Linear	0.2348149	9	-	0.4301544
09	MLP	0.09045	12	7	0.1840289
10 *	MLP	0.06388	11	4	0.09797

иллюстрация 10-ой сети:

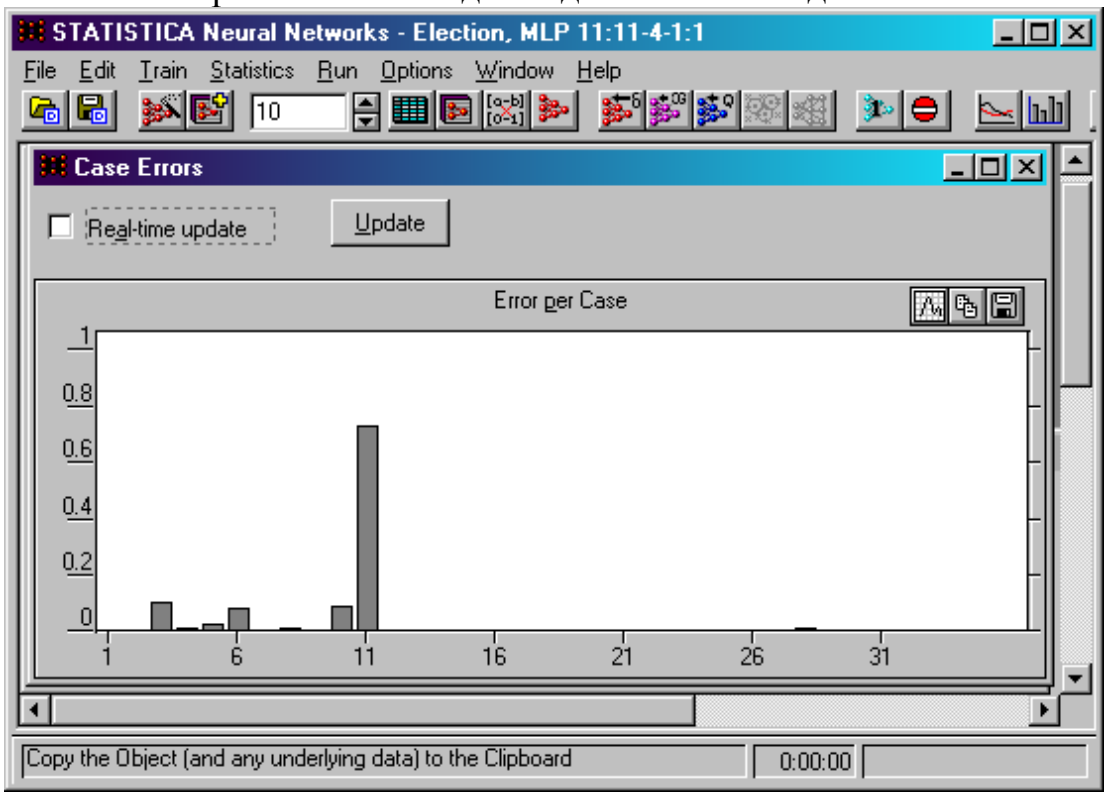


11.Итак, протестируем 10-ую сеть в окне Run data Set:

	OTVET	T. OTVET	E. OTVET	Error
1860	1.991224	2	-0.008776	0.008776
1864	0.9932992	1	-0.006701	0.006701
1868	1.10825	1	0.1082497	0.1082497
1872	0.9836447	1	-0.01636	0.01636
1876	2.034984	2	0.03498	0.03498
1880	1.0848	1	0.0848	0.0848

1884	1.999075	2	-0.0009246	0.0009246
1888	1.018653	1	0.0186534	0.0186534
1892	1.992122	2	-0.007878	0.007878
1896	1.907946	2	-0.09205	0.09205
1900	1.736035	1	0.736035	0.736035
1904	1.001185	1	0.001185	0.001185
1908	1.00043	1	0.0004299	0.0004299
1912	1.988685	2	-0.01131	0.01131
1916	1.000262	1	0.0002617	0.0002617
1920	1.993634	2	-0.006366	0.006366
1924	0.9940131	1	-0.005987	0.005987
1928	1.003802	1	0.003802	0.003802
1932	2.008841	2	0.008841	0.008841
1936	1.002335	1	0.002335	0.002335
1940	0.9973356	1	-0.002664	0.002664
1944	1.00039	1	0.0003897	0.0003897
1948	1.004923	1	0.004923	0.004923
1952	2.005192	2	0.005192	0.005192
1956	1.00259	1	0.00259	0.00259
1960	1.993736	2	-0.006264	0.006264
1964	0.9977177	1	-0.002282	0.002282
1968	2.016099	2	0.0161	0.0161
1972	0.9945278	1	-0.005472	0.005472
1976	1.999936	2	-6.41e-05	6.41e-05
1980	1.998216	2	-0.001784	0.001784
1992	2.002793	2	0.002793	0.002793

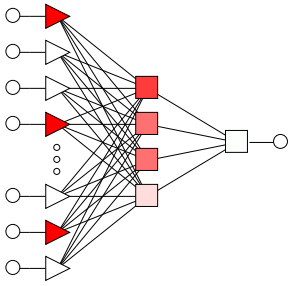
12. Процесс обучения сети в системе STATISTICA Neural Networks сопровождается автоматическим показом текущей ошибки обучения и вычисляемой независимо от нее ошибкой на контрольном множестве, при этом можно гистограммы ошибок для отдельных наблюдений.

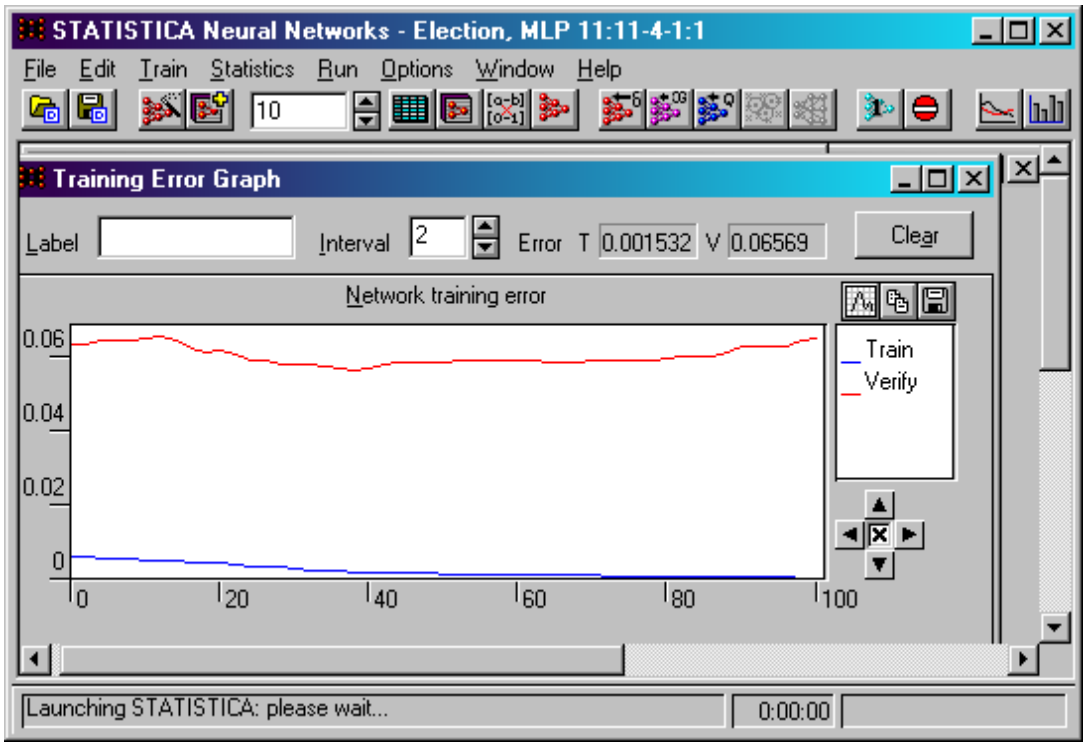


Regression statistics

	Tr. OTVET	Ve. OTVET	Te. OTVET
Data Mean	1.44	1.4	1.5
Data S.D.	0.5066228	0.5477226	0.7071068
Error Mean	7.76e-05	0.04215	0.3219904
Error S.D.	0.007204	0.05366	0.5855474
Abs E. Mean	0.005322	0.0490628	0.4140445
S.D. Ratio	0.01422	0.09797	0.8280891
Correlation	0.9999	0.9957543	1

Run single case => =>Training error graph=>





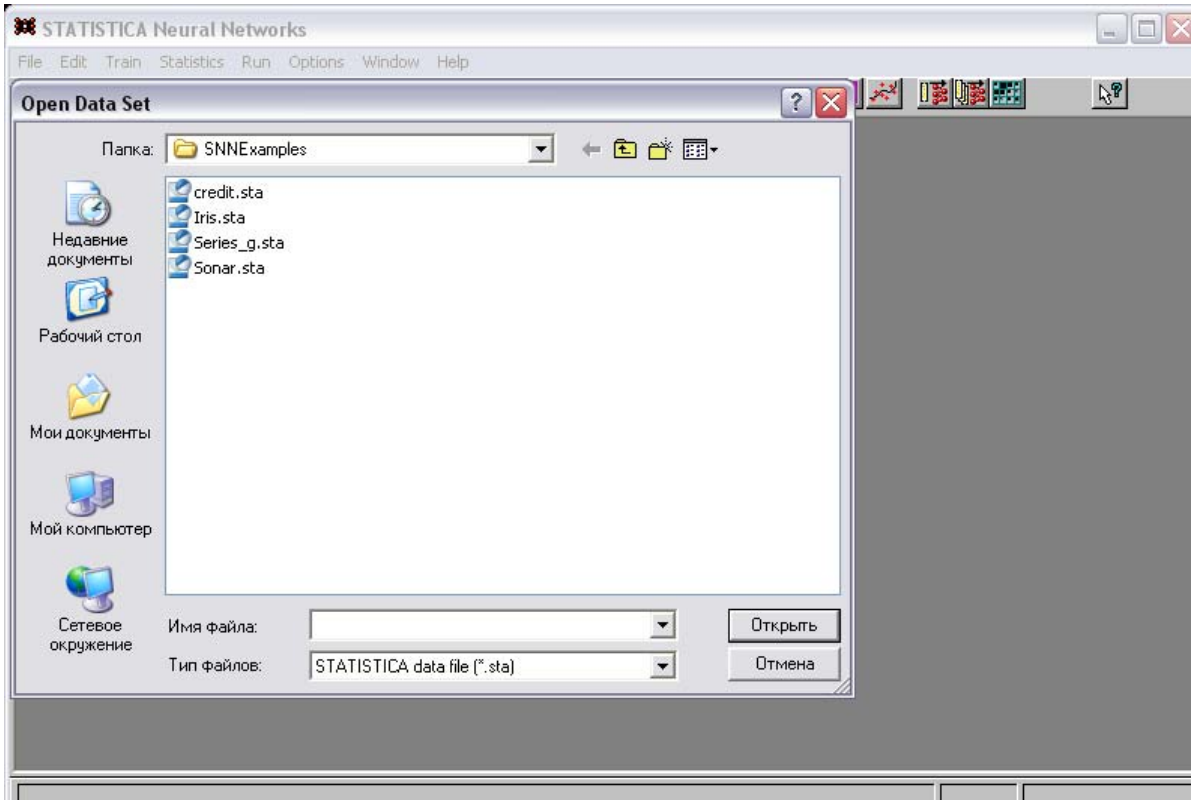
Вывод:

В процессе работы с STATISTICA Neural Networks были использованы различные способы тестирования данных, Заметим, что сети MLP намного лучше справляются с заданием, хотя по сравнению с другими работают медленнее.

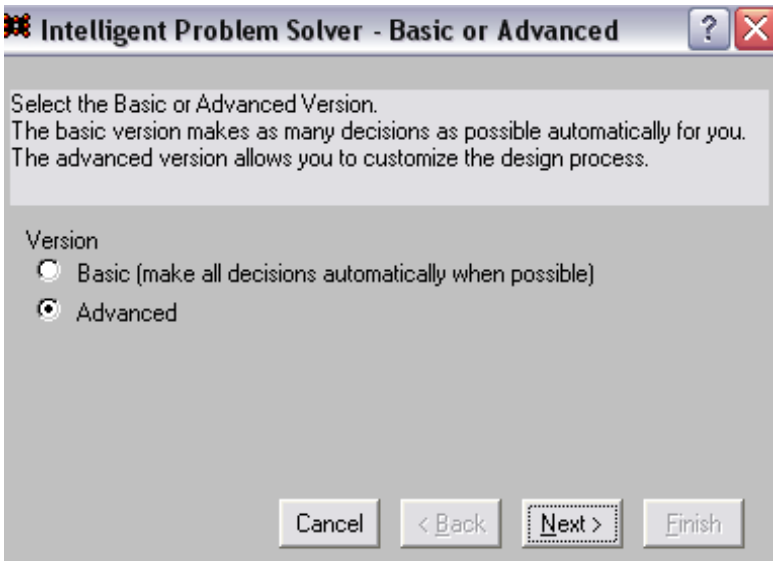
3.3.2 Пример 2

Начало работы с нейроимитатором Statistica Neural Networks, как обычно, начинается с использования файла данных. Ниже показаны несколько экранов при использовании нейронных сетей с ответами, файл данных Credit.sta Этот файл данных имеет текстовый формат, приложен к дистрибутиву SNN.

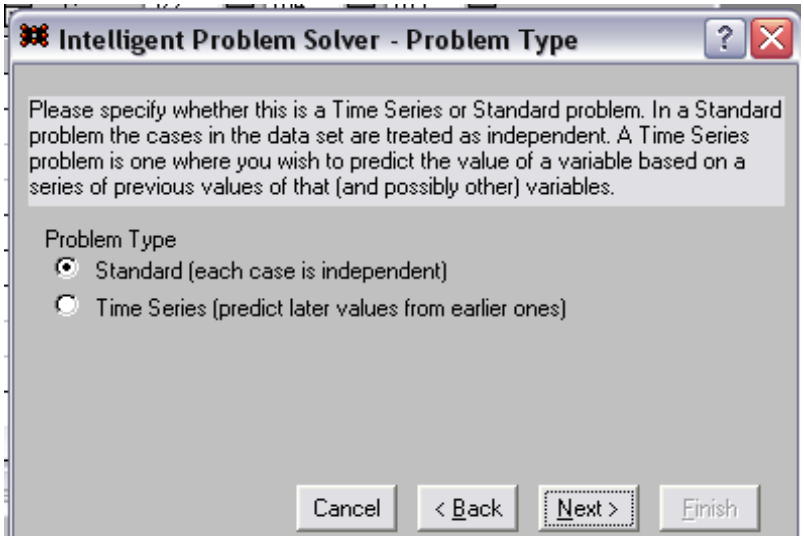
1) Открываем базу данных Credit.sta



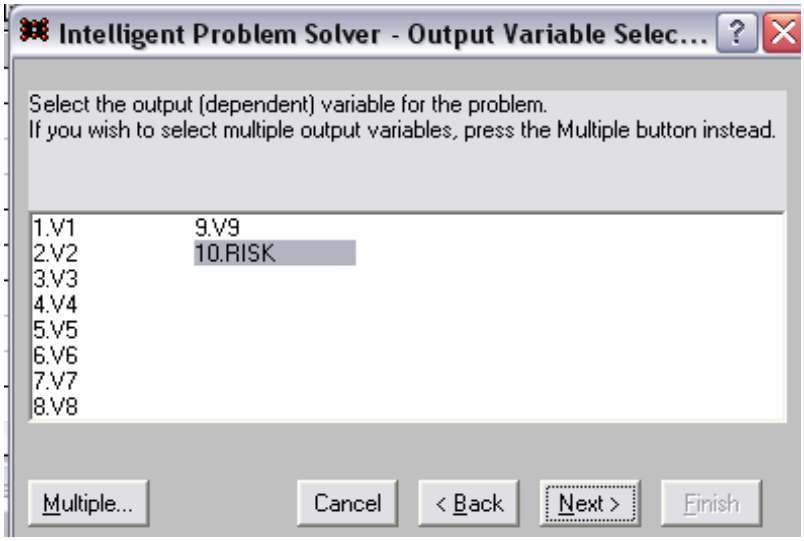
2) Устанавливаем режим решения “Advanced”(Настраиваемый)



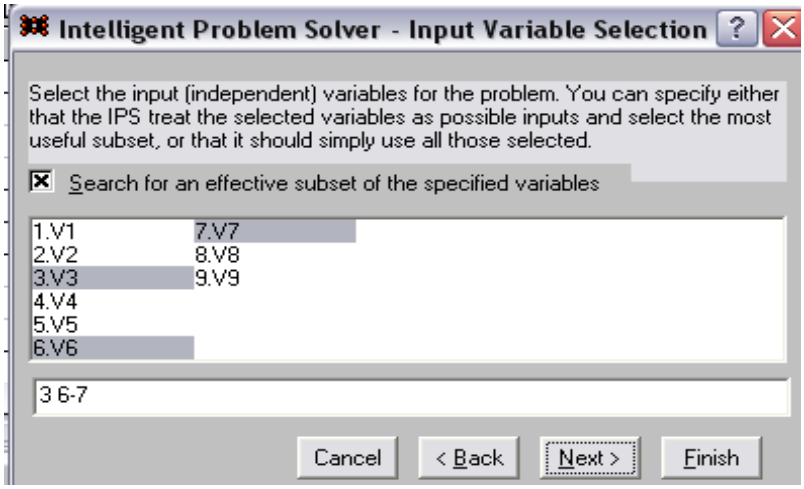
3) Выбираем настройку: Тип решение –>”Стандартный” (каждый выбор независимый).



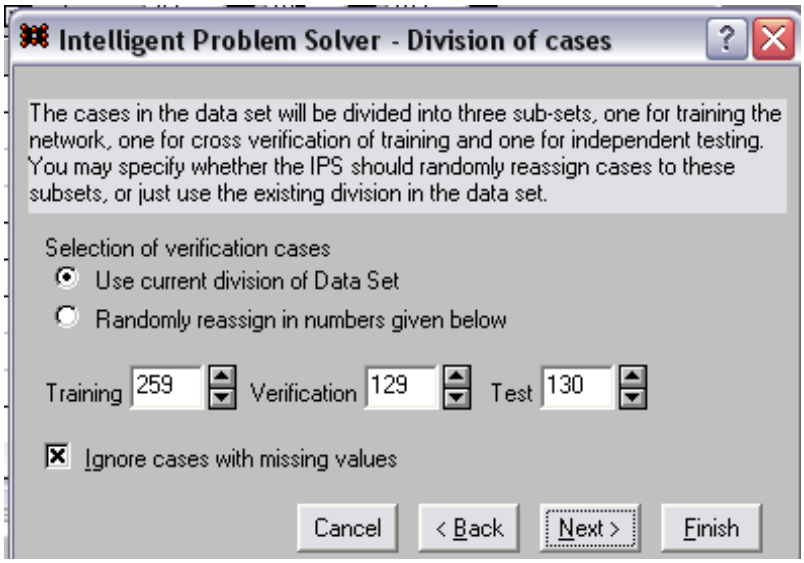
4) В качестве выходных значений выбираем значения, записанные в поле RISK



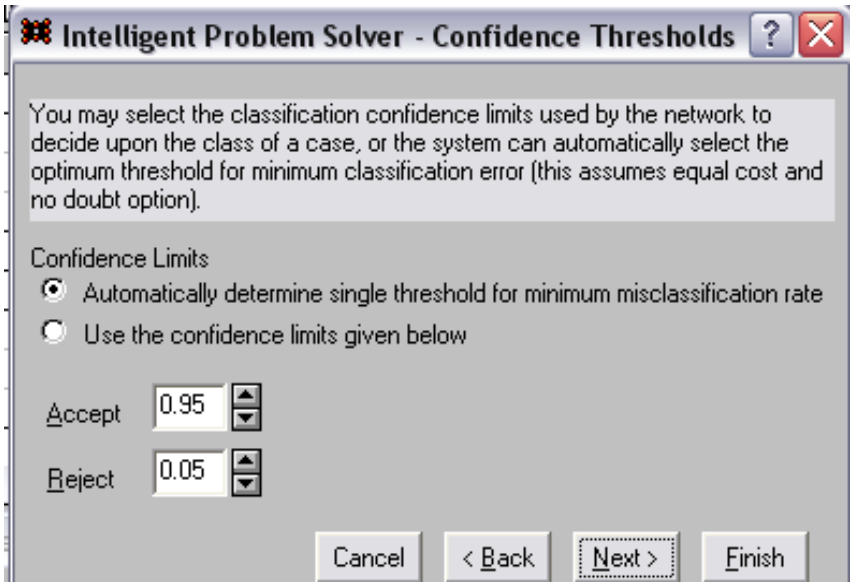
5) В качестве входных значений выбираем значения, записанные в поля V3, V6, V7

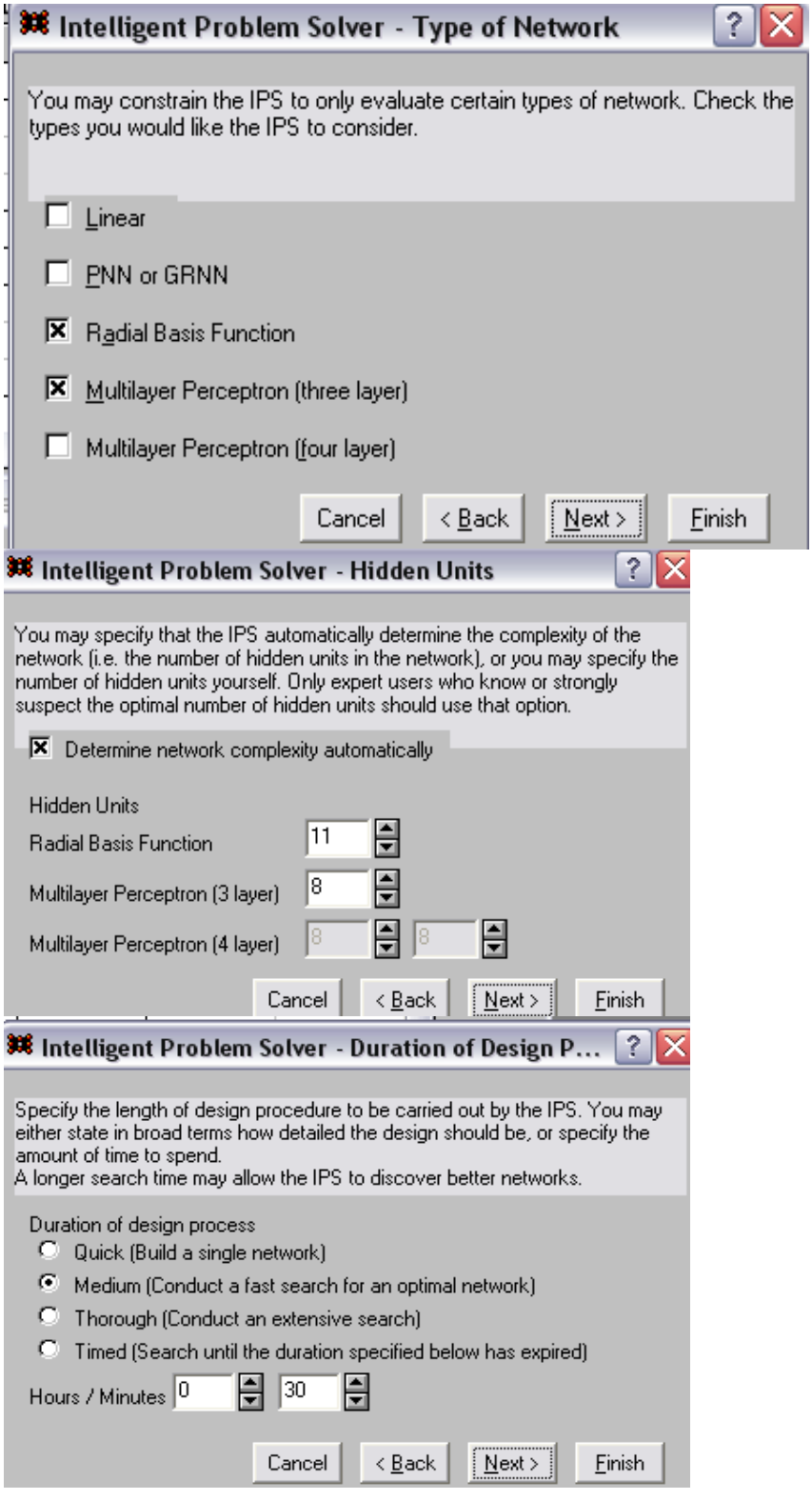


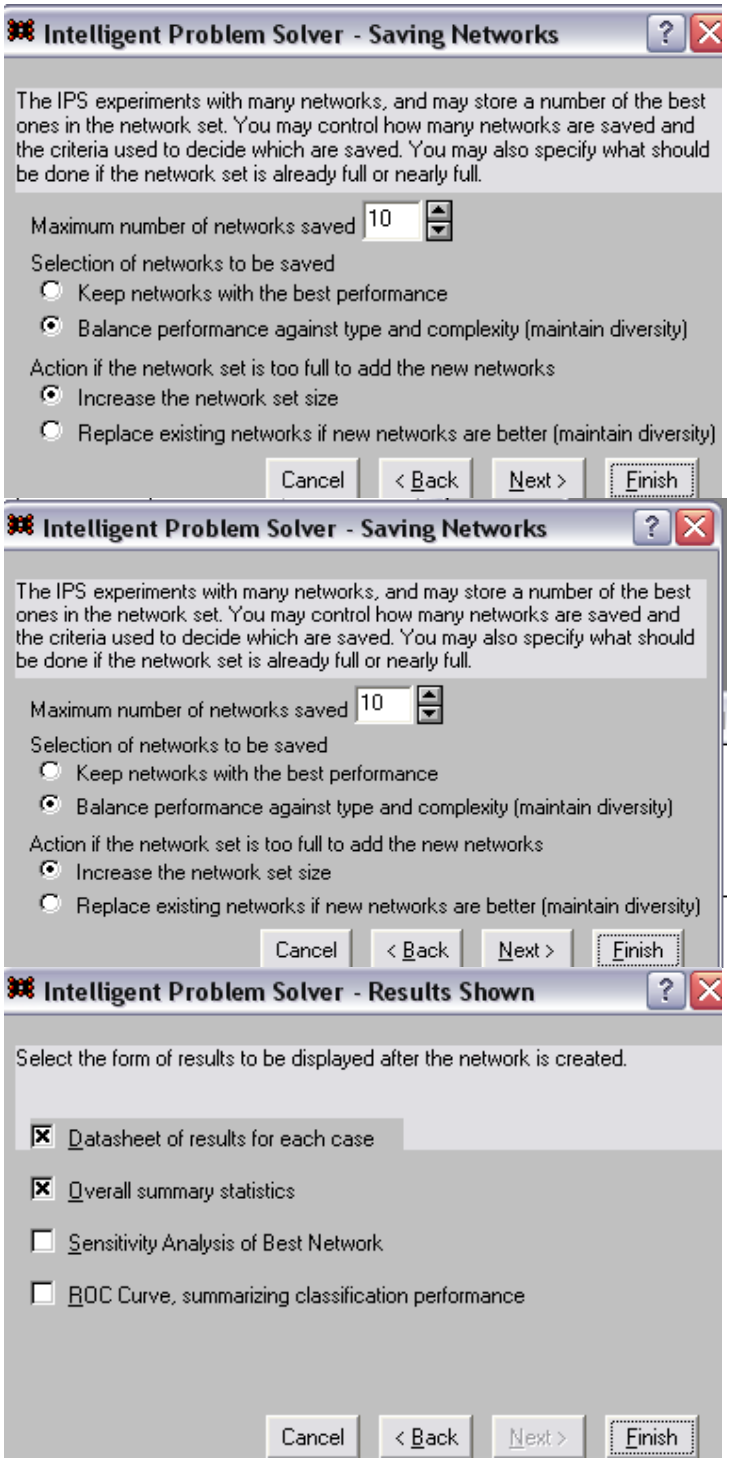
6) Выбираем тип проверки “Разделение набора данных”



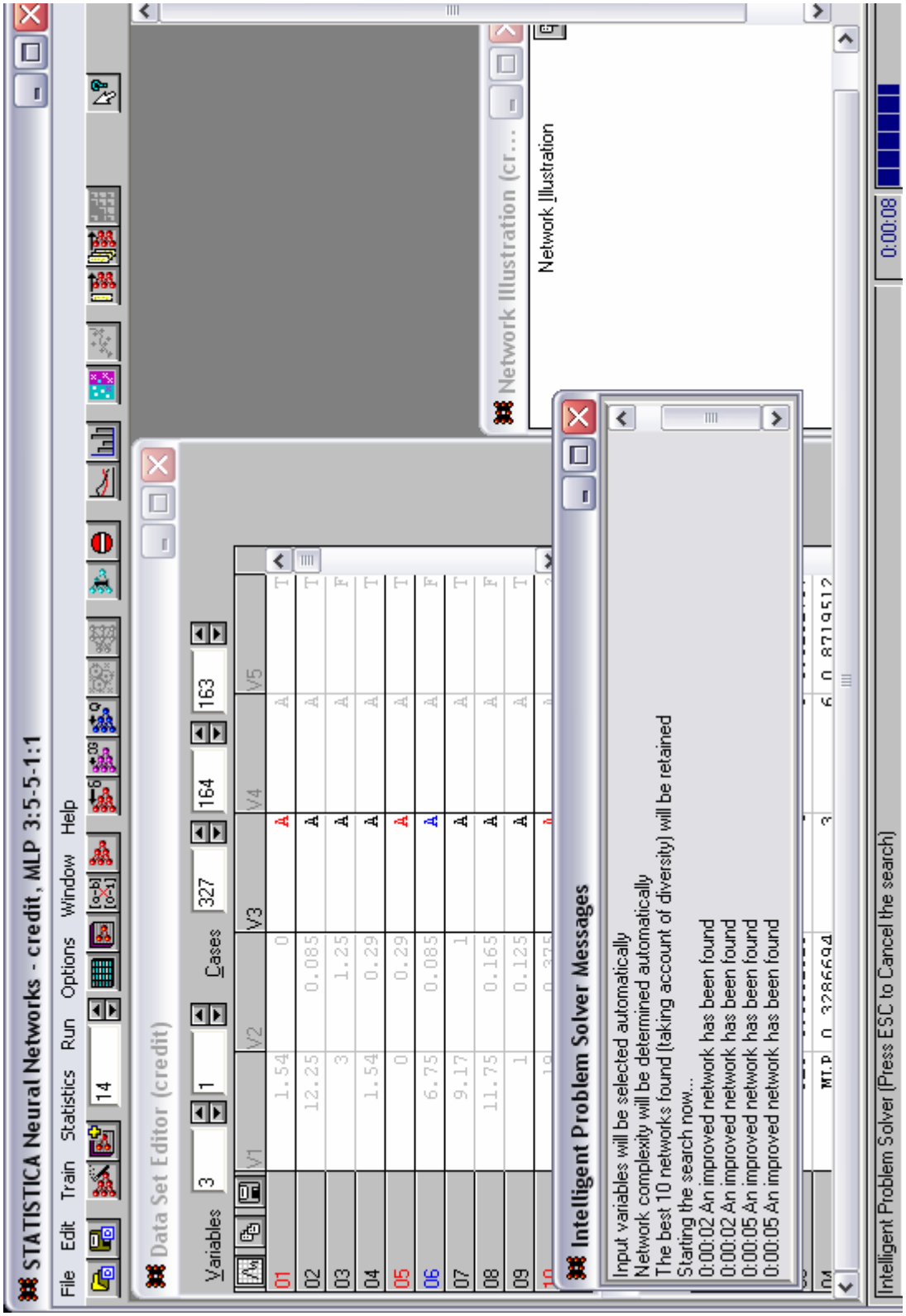
7) Выбираем “Автоматическое определение минимального порога разрядности”, 3 слоя нейронов, число нейронов.



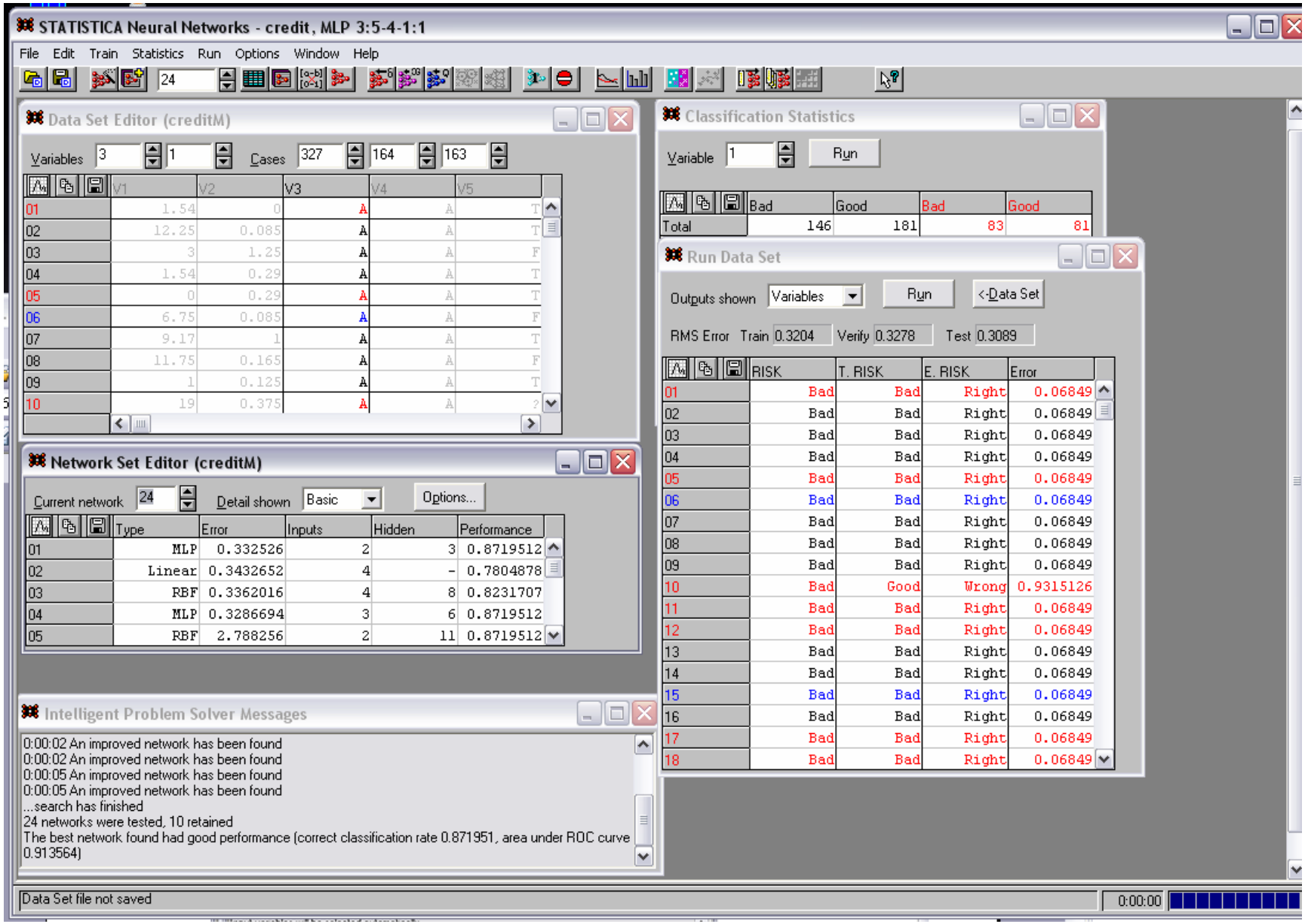

Intelligent Problem Solver - Type of Network
You may constrain the IPS to only evaluate certain types of network. Check the types you would like the IPS to consider.
Linear
PNN or GRNN
Radial Basis Function
Multilayer Perceptron (three layer)
Multilayer Perceptron (four layer)
Cancel
< Back
Next >
Finish
Intelligent Problem Solver - Hidden Units
You may specify that the IPS automatically determine the complexity of the network (i.e. the number of hidden units in the network), or you may specify the number of hidden units yourself. Only expert users who know or strongly suspect the optimal number of hidden units should use that option.
Determine network complexity automatically
Hidden Units
Radial Basis Function
11
Multilayer Perceptron (3 layer)
8
Multilayer Perceptron (4 layer)
8
8
Cancel
< Back
Next >
Finish
Intelligent Problem Solver - Duration of Design P...
Specify the length of design procedure to be carried out by the IPS. You may either state in broad terms how detailed the design should be, or specify the amount of time to spend.
A longer search time may allow the IPS to discover better networks.
Duration of design process
Quick (Build a single network)
Medium (Conduct a fast search for an optimal network)
Thorough (Conduct an extensive search)
Timed (Search until the duration specified below has expired)
Hours / Minutes
0
30
Cancel
< Back
Next >
Finish



Процесс обработки



9) Результат работы


STATISTICA Neural Networks - credit, MLP 3:5-4-1:1
File Edit Train Statistics Run Options Window Help
Data Set Editor (creditM)
Variables 3 1 Cases 327 164 163
Classification Statistics
Variable 1 Run
Bad Good Bad Good
Total 146 181 83 81
Run Data Set
Outputs shown Variables Run <-Data Set
RMS Error Train 0.3204 Verify 0.3278 Test 0.3089
RISK T. RISK E. RISK Error
Network Set Editor (creditM)
Current network 24 Detail shown Basic Options...
Type Error Inputs Hidden Performance
01 MLP 0.332526 2 3 0.8719512
02 Linear 0.3432652 4 - 0.7804878
03 RBF 0.3362016 4 8 0.8231707
04 MLP 0.3286694 3 6 0.8719512
05 RBF 2.788256 2 11 0.8719512
Intelligent Problem Solver Messages
0:00:02 An improved network has been found
0:00:02 An improved network has been found
0:00:05 An improved network has been found
0:00:05 An improved network has been found
...search has finished
24 networks were tested, 10 retained
The best network found had good performance (correct classification rate 0.871951, area under ROC curve 0.913564)
Data Set file not saved
0:00:00

Вывод: В целом правильно решенных примеров (учитывая, что ошибка считалась тогда, когда значение полученное отличалось от исходного более 0,3) около 55%

Примечание: в файлах “Data_1.sta”…”Data_5.sta”, “ creditM.bnt ”- там находятся таблицы подтверждающие результаты тестирования, а в файле ”Credit.bnt” находится сам проект.

Контрольные вопросы к 3 главе:

1. Нейроимитатор и нейроЭВМ
2. Свойства хорошего нейроимитатора
3. Интерфейс с данными

4 Интеллектуальные задачи. Примеры решений

Для практического использования нейронных сетей существует большое число предметных областей. Среди них техника и технические науки характерны наличием большого количества явных расчетных формул, по которым можно получать требуемые результаты, а также большим количеством известных приложений нейронных сетей [Кохонен Т., Хехт-Нильсен Р., Дунин-Барковский В.Л., Фролов А.А., Куссуль Э.М., Оныкий Б.Н., Горбань А.Н., Абовский Н.П. и другие]. Гуманитарные науки, напротив, в настоящее время почти не имеют готовых расчетных формул в явном виде и число приложений нейронных сетей и других математических методов невелико, хотя количество приложений в медицине растет [Россиев Д.А., Baxt W.G., Shufflebarger C.M. и другие]. В силу вышесказанного, в данной работе основными областями приложения выбраны гуманитарные науки - экология, биология, медицина.

В качестве примеров выполнения приведены две работы, выполненные по химии и по политологии.

4.1 Нейросетевое определение валентности химического элемента

Аннотация

Рассматривается вопрос определения валентности химического элемента. Показано, что можно обучить нейросеть для определения валентности химического элемента по сравнительно небольшому числу параметров. В данной работе используется 19 параметров: номер периода элемента, порядковый номер элемента (заряд ядра), номер группы элемента, принадлежность к главной подгруппе своего периода, атомная масса элемента, принадлежность элемента к лантаноидам, принадлежность элемента к актиноидам, количество электронных слоёв, количество внешних электронов, агрегатное состояние в нормальных условиях, принадлежность элемента к металлам, принадлежность элемента к амфотерам, тип элемента по внешнему подслою электронов S- , P- , D- или F, принадлежность элемента к инертным газам, потенциал ряда напряжения, температура плавления, температура кипения, радиоактивность элемента, электроотрицательность.

Работа выполнена с использованием нейроимитатора MultiNeuron версии 1.

Введение

В химии существует большое количество различных математических закономерностей. Для выявления этих закономерностей уже использовался нейросетевой подход, например, для определения закономерностей между химическими элементами - по периодической таблице Менделеева, представляющая одновременно многие из них в табличной форме.

В этой сделана попытка обучения нейросети для определения валентности, с использованием большого количества разных параметров, от которых может или должна зависеть валентность химического элемента.

Данная работа важна не столько для предсказания валентности химического элемента, сколько в определении других параметров, которые статистически связаны с валентностью.

Постановка задачи

Для определения валентности был выбран набор параметров, от которых, предположительно, зависит валентность. Среди этих параметров - порядковый номер элемента или заряд ядра (в данной работе рассматривались только электрически нейтральные атомы, поэтому два последних параметра эквивалентны), атомная масса и некоторые другие.

Описание параметров

1. PER - N периода элемента в диапазоне от 1 до 8.
2. NOM - порядковый номер элемента или число электронов внешнего слоя (1 - 107).
3. GLA - принадлежит ли элемент к главной группе
    1) главная
    0) не принадлежит.
4. AR - атомная масса элемента (1 - 230).
5. LA - принадлежность к лантаноидам (1 да, 2 нет)
6. AC - принадлежность к актиноидам (1 да, 2 нет).
7. ES - количество электронных слоев (1 - 7).
8. EV - количество электронов на внешнем уровне (1 - 8).
9. AGR - агрегатное состояние
    1) твердое
    2) газообразное
    3) жидкое.
10. ML - принадлежность химического элемента к металлам
    1) да
    0) нет.

11. AMF - принадлежность к амфотерам (1 да, 0 нет).
12. SPD - S, P, D или F-элементы
    1) S-элемент
    2) P-элемент
    3) D-элемент
    4) F-элемент.
13. Q - заряд ядра (1 - 107). Примечание: в данной работе не рассматривались ионизированные атомы, поэтому данный параметр дублирует параметр 2.NOM и поэтому не нужен.
14. IG - принадлежность к инертным газам (1 да, 0 нет).
15. PO - потенциал ряда напряжения (-3..3).
16. TPL - температура плавления в Кельвинах (-273..1500).
17. TK - температура кипения в Кельвинах (-273..1500).
18. RAD - принадлежность к радиоактивным (1 да, 0 нет).
19. ELC - электроотрицательность (0 - 40 .
20. GRU - к какой группе принадлежит химический элемент (1..8).
21. OTV - поле ответов.

Обучающая выборка

После формализации параметров были выбраны 20 химических элементов, из них 7 элементов с валентностью 1 или 1-го класса, 7 элементов с валентностью 2, 3 элемента с валентностью 4,2 элемента с валентностью 5 и 1 элемент с валентностью 6. Данные об этих элементах собраны в обучающей выборке и хранятся в таблице структуры dBase - файле VALENT.DBF.

Вывод

Процесс обучения нейросети происходил без изменения надежности и без исключения примеров, параметров. В результате нейросеть обучилась, хотя не настолько быстро, чтобы можно было сказать, что сеть можно быстро обучить вычислению валентности химического элемента. Хотя, мне кажется, что валентность не напрямую связана с какой-либо математической последовательностью. Во всяком случае при обучении сети с установленной опцией "минимизация параметров", в ходе обучения были отброшены 15 параметров. Оставлены были такие параметры, как GRU - номер группы; ML- принадлежность к металлам; TPL- температура кипения и ELC- электроотрицательность (способность элемента присоединять или отдавать электроны). На самом деле валентность и зависит от пяти вышеперечисленных параметров; в особенности от GRU, ML и ELC.

В процессе обучения можно менять степень надежности: от 0 до 100. Чем больше надежность при обучении тем меньше шансов на то, что в будущем обученная нейросеть ошибаться будет реже, чем нейросеть обученная с меньшей надежностью. Также в опциях можно установить настройку на случай тупика во время обучения, как то: остановка, снижение надежности, или исключение в чем-то неугодных примеров, а также такое действие как автоудар. Естественно незаменимо создания файла отчета в котором можно прочесть всю нужную информацию.

4.2 Прогноз выборов в Сосновоборске

Прогнозированием занимаются как люди, состоящие на службе в государственных и коммерческих структурах, так и независимые специалисты. Прогнозирование способствует пониманию прошлого и настоящего. Прогнозирование позволяет смотреть вперед.

Выборы, охватывающие все население страны и подверженное влиянию множества факторов, оказываются вполне прогнозируемым. Речь идет не об угадывании с точностью до балла результаты победителей и проигравших. Так точно не определяют аналитики ни в одной стране мира. Например - итоги недавних выборов в США, где никто, похоже, не предвидел победы Гора в общенародном голосовании. Но общая картины выборов, примерной оценки результатов победителей в соревновании по партийным спискам (на выборах в Думу) или главных претендентов на пост президента все достаточно понятно. Относительно легко прогнозируются стратегии основных участников выборов, их поведение по отношению к соперникам (например, возможные союзы перед вторым туром на президентских выборах) и многие другие параметры.

Перечень достижений и неудач в прогнозировании выборов позволяет сформулировать некоторые выводы общего характера, которые вполне применимы для анализа политических процессов в российском обществе. Долгосрочные политические процессы более прогнозируемы, чем краткосрочные. Иначе говоря, общие тенденции развития на десятилетие вперед нарисовать легче, чем ответить на вопрос, что случится завтра или через месяц. Чем протяженнее период прогноза, тем сильнее влияние объективных факторов - законов экономического развития, экономики страны, эволюции системы ценностей и культуры общества и т.п., и тем обоснованнее эффекты колебания экономической и политической конъюнктуры. Поведение крупных и устойчивых политических институтов (президент, партии, верховный совет, палаты парламента, кабинет министров, местные советы) более понятно для аналитика, поскольку они действуют в рамках множества ограничителей - законодательства,

формальных и неформальных правил, позиций относительно других институтов, наконец, своих корпоративных интересов. Действия же отдельных политиков, обладающих относительной свободой действий, таят в себе гораздо больше неопределенности. В силу этого гораздо легче прогнозировать, что будет делать президент, премьер-министр, лидер думской фракции или губернатор, и труднее просчитать поведение фигур из администрации президента и политических олигархов, например, Борис Березовский и другие.

Была поставлена задача получить прогноз результатов выборов, которые состоятся 23.06.2002 на основе данных собранных по результатам выборов 14.01.2001, проводимых в городе Сосновоборске. Данные собраны и сгруппированы по биографиям кандидатов, статьям и результатам голосования, изложенным в местной газете «Рабочий». Основная информация получена в комиссии по организации промежуточных выборов на нескольких избирательных участках.

Параметры:

Birthday - Год рождения
Sex - Пол: (мужской - 1, женский - 0)
Place - Место рождения: (Красноярский край - 1, другое - 0)
Образование (уровень) среднее, среднее техническое, высшее или послевузовское
Livesosn - С какого года проживает в Сосновоборке
Deputy - Был ли депутатом (да -1 нет - 0)
Familu - Семейное положение (не женат (замужем) - 0, женат (замужем) - 1)
Kids - Количество детей
Категории работы
Jobleess - безработный
Worker -рабочий
Employee - служащий
Leader - руководитель
Bisnes - предприниматель
Pensioner - пенсионер
Место работы(сфера)
Medichine - медицина
Militia -органы внутренних дел
KZAP - КЗАП
GKX - ЖКХ
CHP -частные предприятия
Shool - образование
Krasn - работа в Красноярске
Nowhere - нигде
TEC – ТЭЦ

Политические партии
KPRF - КПРФ
EDIN - Единство
CHEST - Честь и Родина
Nadeg - Надежда и опора
LDPR - ЛДПР
Publwork - Общественные организации:(Да- 1, нет - 0)
Bonus - Награды: (нет - 0,профессиональные -1, боевые - 2)
District - № избирательного участка
Amount - число кандидатов на участке
Publicp - Положительные публикации в прессе в период предвыборной компании: (нет-0, да - 1)
Publicm -Отрицательные публикации в прессе в период предвыборной компании: (нет-0, да - 1)
Quant1 - Количество голосов в 1 туре (14.01.01
Against1 - Число голосовавших против всех в первом туре
Quant2 - Количество голосов во 2 туре (14.01.01)
Against1 -Число проголосовавших против всех во втором туре
Предвыборные обещания - тезисы в прессе:
Municp - Благоустройство города (Да-1, нет-0)
Commynal - Решение коммунальных проблем
Factoru - Оздоровление завода
Bisiness - Поддержка малого бизнеса
Narcotism - Проблемы наркомании
Syn - Привлечение инвестиций
Reconstr -Реконструкция нежилых помещений
Pension - Проблемы пенсионеров
Newjob - Создание новых рабочих мест
Power - Вопросы распределения власти в городе
Order - Общественный порядок
Welfare - Улучшение благосостояния населения
Livregion - Проживает ли кандидат на избирательном участке на котором баллотируется (да-1, нет- 0)
Leaflet - Распространялись ли листовки(да-1, нет- 0)
Ether -Телевизионные эфиры(да-1, нет- 0)
Roller - Телевизионные ролики с выступлениями авторитетных лиц города(да-1, нет- 0)
Meeting - Встречи с избирателями(да-1, нет- 0)
Renown - Известность в городе (не известен -0, положительная известность - 1, отрицательная -2)
Effect -Коммуникабельность(умение произвести впечатление)(да-1, нет- 0)
Grammat - Осведомленность, грамотность в вопросах политики (Да-1, нет- 0)

Для анализа данных был выбран программный продукт представляющий собой менеджер обучаемых искусственных нейронных сетей (NewroPro0.25), работающий в среде MS Windows 95 или MS Windows NT 4.0 и позволяющий производить следующие базовые операции:

1. Создание нейропроекта;
2. Подключение к нейропроекту файла (базы) данных в формате dfb (dBase, FoxBase, FoxPro, Clipper) или db (Paradox);
3. Редактирование файла данных – изменение существующих значений и добавление новых записей в базу данных; сохранение файла данных в другом формате;
4. Добавление в проект нейронной сети слоистой архитектуры с числом слоев нейронов от 1 до 10, числом нейронов в слое – до 100;
5. Обучение нейронной сети решению задачи прогнозирования или классификации. Нейронная сеть может одновременно решать как несколько задач прогнозирования (прогнозирование нескольких чисел), так и несколько задач классификации, а также одновременно задач и прогнозирования, и классификации.
6. Тестирование нейронной сети на файле данных, получение статистической информации о точности решения задачи;
7. Вычисление показателей значимости входных сигналов сети, сохранение значений показателей значимости в текстовом файле на диске;
8. Упрощение нейронной сети;
9. Генерация и визуализация вербального описания нейронной сети, сохранение вербального описания в текстовом файле на диске;
10. Выбор алгоритма обучения, назначение требуемой точности прогноза, настройка нейронной сети.

NewroPro0.25 был выбран потому применение данного программного продукта возможно для решения любой задачи классификации или прогноза, которая решается при наличии выборки данных и для решения которой ранее использовались традиционные математические методы (регрессионный анализ, непараметрическая статистика и другие), однако не была достигнута требуемая точность прогноза. Так же, замечу, что от имеющихся в настоящее время нейросетевых программных продуктов данный продукт отличает наличие возможностей целенаправленного упрощения нейронной сети для последующей генерации вербального описания.

А так же наличие развитых возможностей по упрощению сети в совокупности с построением ее вербального описания придает предлагаемому продукту новые потребительские свойства – возможность порождения знаний из таблицы данных. Под знаниями здесь понимается текст, объясняющий процесс решения нейронной сетью задачи. А поскольку одним из преимуществ нейронных сетей является возможность решения

неформализованных задач классификации и прогноза (тех задач, явный алгоритм решения которых не известен), то данный текст предложит один из алгоритмов решения такой задачи.

Выполнение курсового проекта было разбито на 2 этапа:

1. Данные о выборах 14.01.02 были разделены на 2 части (обучающую и тестовую).

Причем было применено 2 способа разбивки данных (четные и нечетные строки; пропорционально по участкам). Результаты имеют некоторые различия (таблица 4.1)

Характеристика сети для обоих способов была одинакова.

После тестирования было проведено упрощение нейронной сети. (Таблица 2 и 3)

Это сделано потому, что не все входные сигналы сети и синапы необходимы для правильного решения задачи сетью.

Можно достаточно сильно упростить сеть без ухудшения точности решения задачи. Основными результатами проведения процесса упрощения сети являются следующие:

- Сокращается число входных сигналов сети. Если правильно решить задачу можно на основе меньшего набора входных данных, то это может в дальнейшем сократить временные и материальные затраты на сбор информации.

- Нейронную сеть более просто можно будет реализовать на аппаратной платформе.

- Сеть может приобрести логически прозрачную структуру. Известно, что почти невозможно понять, как обученная нейронная сеть решает задачу. После упрощения нейронная сеть становится достаточно обозримой и можно попытаться построить алгоритм решения сетью задачи на основе графического представления или вербального описания структуры сети.

Для упрощения нейронной сети проделывались следующие операции:

1. Сокращение числа входных сигналов – удаление наименее значимых входных сигналов (оказало сильное влияние на улучшение результата);

2. Сокращение числа нейронов – удаление наименее значимых нейронов сети.

3. Сокращение числа синапсов – удаление наименее значимых синапсов сети.

4. Сокращение числа неоднородных входов – удаление наименее значимых неоднородных входов нейронов сети.

5. Равномерное упрощение сети – сокращение числа приходящих на нейроны сети сигналов до задаваемого пользователем.

6. Бинаризация синапсов сети – приведение значений весов синапсов и неоднородных входов нейронов к выделенным значениям.

Операции 2-6 не повлияли на результаты тестирования.

Была определена значимость параметров (таблица 4 и 5). В зависимости от способа разбивки данных значимость параметров получилась различна.

При разбивке «четные-нечетные» самым значимым параметром было место работы – ЖКХ, а так же проведение телевизионных эфиров и показ рекламных роликов.

При разбивке «по участкам» самым важным параметром был показ роликов, место работы не имело большой значимости.

На втором этапе исследования в качестве обучающей выборки были взяты все данные выборов 14.01.01, а в качестве тестовой выборки данные по кандидатам на выборы 23.06.02.

Также было проведено упрощение сети, определение значимости параметров и дан анализ результатов.

На первом этапе исследования, данные собранные по результатам выборов (Приложение 1), проводимых 14.01.01 в г. Сосновоборске, были разбиты на две выборки тестовую и обучающую (причем разбивка проводилась по двум принципам: данные делились на четные и нечетные строк; данные делились пропорционально по участкам).

Было обучено 20 сетей для обоих вариантов разбиения данных.

Описание сети в обоих случаях было одинаковым, а именно:

Число входных полей - 57
Число входов сети - 57
Число выходных полей – 1
Число выходов сети - 1
Слой 1 - 3 нейрона
Слой 2 - 5 нейронов
Слой 3 - 7 нейронов

При обучении по выборке «четные-нечетные»:

Цикл обучения - 11
Шаг - 0,03028921
Средняя оценка - 0
Правильные решения - 32 из 32

При обучении при выборке «пропорционально по участкам»:

Цикл обучения - 9
Шаг - 0,1568877
Средняя оценка - 0
Правильные решения - 32 из 32

Результаты тестирования даны в таблице 1.

Диапазон ошибок при разбивке по четным и нечетным строкам несколько больше (от 5 до11) чем при разбивке по участкам (от 4 до 10)

Количество циклов обучения не влияет на число ошибок при тестировании

При сокращении числа входных сигналов число ошибок при разбивке «чет-нечет» практически всегда уменьшается в 2 раза, крайне редко увеличивается, при разбивке по участкам число ошибок незначительно уменьшается (таблица 2, 3 соответственно)

Значимость параметров обозначена в таблице 4 и в таблице 5.

На втором этапе исследования в качестве обучающей выборки были взяты все данные выборов 14.01.01, а в качестве тестовой выборки данные по кандидатам на выборы 23.06.02. Было обучено 20 сетей. Применены те же принципы, что и на первом этапе.

Результаты тестирования и результаты тестирования после упрощения приведены в Таблице 6, 7. Также приведена значимость параметров.

Выводы

С учетом теоретических знаний изложенных в пункте «Документы в социологии» были собраны и проанализированы данные по двум избирательным кампаниям. На основании проделанной работы можно сделать следующие прогнозы.

По первой части исследования прогнозирование поражения производится с большей точностью, чем победа. Это и понятно – таких данных больше.

По результатам второй части анализа можно предположить, что большее число голосов наберет кандидат под номером 5 на 7 избирательном участке, на 8 избирательном участке кандидат под номером 4, и с небольшой вероятностью на 5 избирательном участке больше голосов наберет кандидат под номером 8.

Анализируя значимость параметров и тот факт, что избирательная деятельность заканчивается 22.07.02 можно дать рекомендации кандидатам.

Сильно поправить положение кандидата могут телевизионные эфиры, личные встречи с избирателями, телевизионные ролики. Правда на результатах кандидатов под номером 1 и 3 даже усиление этих позиций положительно не сказалось.

С учетом того, что местная телевизионная компания реконструируется вероятность таких шагов маловероятна.

Неожиданно большая значимость на втором этапе исследования у параметра «семейное положение».

Контрольные вопросы по курсу

1. Основные архитектуры и виды нейронный сетей: слоистые, полносвязные, сигмоидные, монотонные; нейросети с учителем и без учителя, Хопфилда, Кохонена
2. Элементы нейросетей: синапс или линейная связь, нелинейный элемент или функция активации, точка ветвления, сумматоры - простой, адаптивный, неоднородный, квадратичный
3. Биологический нейрон
4. Режимы работы нейросетей (операции с нейросетями)
5. Типы нелинейных функций
6. Входные и выходные сигналы, функционирование, обучение, тестирование, оценивание
7. Обучение и оптимизация. Методы обучения: градиентный, случайный, партан и др. квазиньютоновский и сопряженных градиентов; одномерная оптимизация
8. Обучаемые нейросети. Обучение по примерам, страницам, по всему задачнику (обучающей выборке); преимущества, проблемы и особенности обучения по страницам
9. Значимость параметров и сигналов
10. Контрастирование
11. Предобработка, ее виды: Перемасштабирование, Нормализация, Стандартизация
12. Задачи для нейросетей: задачи математические и прикладные
13. Оценка работы сети
14. Архитектуры нейроимитаторов: элементы нейрокомпьютера или нейроимитатора
15. Постановка задачи для обучения НС; методика сбора и организации данных
16. Аппроксимация и основные теоремы: Вейерштрасса, Стоуна, обобщенная

КЛЮЧЕВЫЕ СЛОВА

аппроксимация
архитектуры нейроимитаторов
архитектуры нейронный сетей
биологический нейрон
входные и выходные сигналы
градиент
значимость параметров и сигналов
контрастирование
нейроимитатор
нейрон
нейронные сети с учителем и без учителя
обучение
обучение по всему задачнику (обучающей выборке)
обучение по примерам
оптимизация
оценка работы сети
предобработка
сети Кохонена
сети Хопфилда
синапс
слоистые, полносвязные, сигмоидные сети
случайные и направленные методы обучения
сумматоры - простой, адаптивный, неоднородный, квадратичный
тестирование
технология исследования с помощью нейронных сетей
функционирование нейронных сетей
функция активации или нелинейный элемент
элементы нейронных сетей

Заключение

В ходе изучения курса студент слушает лекции, посещает практические занятия, занимается индивидуально. Освоение курса предполагает, помимо посещений лекций и семинарских занятий, выполнение домашних заданий. Особое место в овладении данным курсом отводится самостоятельной работе. Для успешного освоения материала курса студенту предложено данное пособие. В нем кратко, но содержательно изложены основные понятия и определения по дисциплине. Достоинством конспекта лекций также служит возможность быстро и просто найти ответ на интересующий вопрос, так как текст разбит на тематические части. Недостатком можно считать то, что теоретический материал сильно сокращен. Для более полного охвата материала по курсу студент обязан посещать лекции, а не ограничиваться прочтением конспекта лекций.

Литература

1. Галушкин, А.И. Нейрокомпьютеры: Учеб. пособие (Нейрокомпьютеры и их применение. Кн.3) / А.И. Галушкин. М.: ИПРЖР, 2000.- 528 с.
2. Нейронные сети: история развития теории: Учеб. пособие для вузов / ред.Галушкин А.И., ред. Цыпкин А.З. - М.: Радиотехника, 2001.- 840 с. (Нейрокомпьютеры и их применение; кн.5)
3. Галушкин, А.И. Теория нейронных сетей: Учеб. пособие / А.И. Галушкин. (Нейрокомпьютеры и их применение. Кн.1). М.: ИПРЖР, 2000.- 416 с.
4. Галушкин, А.И. Нейрокомпьютеры и их применение на рубеже тысячелетий в Китае. Т.1 и 2 / А.И. Галушкин. М., 2004.- 367+464 с.
5. Нейропрограммы: Учебное пособие: В 2 ч. Ч. 1/ Л.В.Гилева, С.Е.Гилев, А.Н.Горбань и др.; Красноярск: КГТУ, 1994.- 123 с.
6. Нейропрограммы: Учебное пособие: В 2 ч. Ч. 2/ Л.В.Гилева, С.Е.Гилев, А.Н.Горбань и др.; Красноярск: КГТУ, 1994.- 123 с.
7. Горбань, А.Н. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев. Новосибирск: Наука, 1996.- 276 с.
8. Нейроинформатика / А.Н. Горбань, В.Л. Дунин-Барковский, А.Н. Кирдин и др. Новосибирск: Наука, 1998.- 296 с.
9. Горбань, А.Н. Обучение нейронных сетей / А.Н. Горбань. М.: СП "Paragraph", 1990.- 160 с.
10. Миркес, Е.М. Нейрокомпьютер. Проект стандарта / Е.М. Миркес. Новосибирск: Наука, 1999.- 337 с.
11. Осовский, С. Нейронные сети для обработки информации / Осовский С. - М.: Финансы и статистика, 2002. – 343 с.
12. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Рутковская Д. и др. М., 2004.- 452 с.
13. Хайкин, С. Нейронные сети: Полный курс. 2-е издание / Хайкин С.. М.: "Вильямс", 2006.- 1104 с.
14. Дорогов А.Ю. Быстрые нейронные сети / Дорогов А.Ю. М.: Изд-во Санкт-Петербургского университета, 2002.- 77 с.
15. Легалов А.И. Нейроинформатика: Уч. пособие / Легалов А.И., Миркес Е.М., Сиротинина Н.Ю. - Красноярск, 2006. – 172 с.
16. Методы нейроинформатики: Сб. научн. трудов / Под ред. А.Н. Горбаня. - Красноярск: КГТУ, 1998. - 204 с.
17. Терехов, С.А. Вводные лекции по теории и приложениям искусственных нейронных сетей (рукопись) / Терехов С.А. Красноярск, 2000. 69 с.
18. Минский М. Персептроны / Минский М., Пейперт С. М.: Мир, 1971.
19. Розенблатт Ф. Принципы нейродинамики. Перцептрон и теория механизмов мозга / Розенблатт Ф. М.: Мир, 1965.- 480 с.

20. Уоссермен Ф. Нейрокомпьютерная техника / Уоссермен Ф.- М.: Мир, 1992.
21. Жуков, Л.А. Параллельное программирование: Учебное пособие для студентов специальностей 220300 и 220400, направлений 654600, 552800 всех форм обучения / Л. А. Жуков. - Красноярск: СибГТУ, 2004. - 116 с.
22. Жуков, Л.А. Формализация технологии применения нейронных сетей с учителем / Л.А. Жуков, Н.В. Решетникова. Красноярск, 2005.- 168 с.

Дополнительная литература

23. Гилев, С. Е. Обучение нейронных сетей: Методы, алгоритмы, тестовые испытания, примеры приложения: Дис. канд. физ.-мат. наук / С. Е. Гилев. Красноярск, 1997. 187с.
24. Горбань А.Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. - Новосибирск, 1998. - Т.1, №1. - с.11-24.
25. Горбань А.Н., Миркес Е.М. Логически прозрачные нейронные сети для производства знаний из данных. Вычислительный центр СО РАН в г. Красноярске. Красноярск, 1997. 12 с. (Рукопись деп. в ВИНИТИ 17.07.97, № 2434-В97)
26. Каллан Р. Основные концепции нейронных сетей: Пер.с англ. М.-Л.: Вильямс, 2001.- 287 с.
27. Калинин А.В., Подвальный С.Л. Технология нейросетевых распределённых вычислений: Монография. Воронеж: Воронеж. гос. техн. ун-т, 2004. 122 с.
28. Россиев, Д. А. Самообучающиеся нейросетевые экспертные системы в медицине: теория, методология, инструментарий внедрение: Дис. докт. мед. наук / Д. А. Россиев. Красноярск, 1995. 379 с.
29. Рубцов, Д. В. Разработка технологии применения искусственных нейронных сетей в прикладных информационных системах: Автореф. канд. техн. наук / Д. В. Рубцов. Барнаул, 2000.
30. Терехов С.А. Вводные лекции по теории и приложениям искусственных нейронных сетей - Красноярск.: ИВМ СО РАН, 2000. - 69 с.
31. Терехов С.А. Технологические аспекты обучения нейросетевых машин // В сб. "Лекции по нейроинформатике". М.: МИФИ, 2006.- С.13-73
32. Ахо А., Ульман Д. Теория синтаксического анализа, перевода и компиляции, в 2-х т. М.: Мир, 1978.
33. Серебряков В.А., Галочкин М.П. Основы конструирования компиляторов. 2001.- 192 с.
34. Опалева Э.А., Самойленко В.П. Языки программирования и методы трансляции. Издательство: БХВ-Петербург, 2005.

35. Кнут Д. Семантика контекстно-свободных языков. В сб.: Семантика языков программирования. М.: Мир, 1980. С.137-161
36. Хомич А.В. Метод эволюционной оптимизации и его приложение к задаче синтеза искусственных нейронных сетей / А. В. Хомич, Л. А. Жуков // Нейрокомпьютеры: разработка, применение. — 2004. — № 12. — С. 3—15.
37. Хомич А.В. Декомпозиция задачи обучения нейронных сетей с учителем для понижения вычислительной сложности обучения / А. В. Хомич, Л. А. Жуков // Доклады АН ВШ РФ. — 2005. — № 1(4). — С. 59—67.
38. Хомич А.В., Жуков Л.А. Эффективность метода группового учета аргументов при синтезе сигмоидных нейронных сетей / Сборник материалов 14-й международной конференции по нейрокибернетике. Ростов-на-Дону: 2005. (принято к печати)
39. Хомич А.В. Оптимизация топологии рекуррентных и многослойных нейронных сетей с применением генетических алгоритмов / А. В. Хомич, Л. А. Жуков // Нейроинформатика—2004: сб. науч. статей. ч.2. / М.: МИФИ, 2004. — С. 68—74.
40. Хомич, А. В. Анализ и оптимизация операций мутации и кроссовера в генетических алгоритмах / А. В. Хомич, Л. А. Жуков // Новые информационные технологии в исследовании сложных структур: сб. науч. статей / Вестник ТГУ. — 2004. — С. 111—114.
41. Хомич, А. В. Организация параллельных вычислений в нейросервере Neurogenesis / А. В. Хомич // Распределенные и кластерные вычисления: сб. науч. статей / Под ред. В.В.Шайдурова. — Красноярск: ИВМ СО РАН, 2005. — С. 121—129. — ISBN 57636-0759-7.
42. Хомич А.В., Жуков Л.А Эволюционный метод оптимизации структуры нейронной сети с учителем // Нейроинформатика-2005. Сборник научных трудов. Ч.1. М.: МИФИ, 2005. - С.11-18.
43. Хомич А.В. Neurogenesis / Свидетельство Роспатента об официальной регистрации программы для ЭВМ №2005611168
44. Жуков Л.А. Технология нейросетевого решения прикладных классификационных задач для использования в экологии, биологии и медицине: Научное издание / Красноярск. госуд. техн. ун-т. Красноярск, 2004. 148 с.: ил. 26. - Библиогр.: 399 назв. - Рус. - Рукопись деп. в ВИНИТИ
45. Жуков Л.А., Решетникова Н.В. Формализация технологии применения нейронных сетей с учителем и особенности их использования для решения прикладных задач: Монография. Красноярск, КГТУ, 2005.- 168 с.
46. Воскобойникова В., Жуков Л.А. Определение авторства с помощью нейросетей // Нейроинформатика и ее приложения: 5 Всероссийский семинар, Красноярск, 1997. С.43

47. Щукина А.В., Забусова Н.В., Кудрявцева И.П., Раймер Т.П., Богданова Н.А., Емельяненко Г.П., Брюханова Л.С., Матси Л.В., Савина Н.В., Жуков Л.А. Использование нейроимитатора MultiNeuron для прогнозирования температуры воды в Енисее в районе Красноярска // 3-я Межвузовская конференция по экологическому образованию. СибГТУ, 1998. С.162-164
48. Жуков Л.А., Гарманов Д.В., Бегизардов Р.Н. Нейронные сети для прогноза результатов выборов в Законодательное собрание края // Проблемы нейрокибернетики: Материалы XII Международной конференции по нейрокибернетике: Ростов-на-Дону, 1999.- 323с. С.211-212
49. Бумаженко О.А., Калинкин П.В., Жуков Л.А. Нейросетевая классификация элементов // “Проблемы теоретической и экспериментальной химии”: 9 всерос. студ. науч. конф. Екатеринбург: УрГУ, 1999.- С.61
50. Жижаева В.Н., Лукьянов А.А., Григорьев Р.В., Жуков Л.А. Нейросетевое определение уровня экономического развития государства по географическим факторам // Информатика и информационные технологии: Тез. докл. межвузовской научной конференции. Красноярск: КГТУ, 1998. С.87-88
51. Фефелов И.В., Чегуров С.В., Ольшанникова Ю.С., Курохтин В.В., Жуков Л.А. Классификация луней при помощи нейронных сетей без учителя // Нейроинформатика и ее приложения: 8 Всероссийский семинар, Красноярск, 2000. С.174-175
52. Щукина А.В., Забусова Н.В., Кудрявцева И.П., Раймер Т.П., Богданова Н.А., Брюханова Л.С., Матси Л.В., Савина Н.В., Решетникова Н.В., Оводова А.А., Жуков Л.А. Использование нейроимитатора MultiNeuron для прогнозирования температуры воды в Енисее в районе Красноярска // Нейроинформатика и ее приложения: 6 Всероссийский семинар, Красноярск, 1998. С.195
53. Щукина А.В., Забусова Н.В., Кудрявцева И.П., Раймер Т.П., Богданова Н.А., Брюханова Л.С., Матси Л.В., Савина Н.В., Решетникова Н.В., Оводова А.А., Жуков Л.А. Нейросетевое прогнозирование гидрологической обстановки // Нейроинформатика и ее приложения: 6 Всероссийский семинар, Красноярск, 1998. С.196
54. Смешко И.В., Жуков Л.А. Нейросетевая классификация правителей России // Нейроинформатика и ее приложения: 7 Всероссийский семинар, Красноярск, 1999.-167с., С.134
55.
56. Жуков Л.А. Технология классификации с помощью нейронных сетей без учителя // Теоретические и прикладные вопросы современных информационных технологий. Материалы всеросс.конф. Улан-Уде: ВСГТУ, 2001. С.40-47

57. Жуков Л.А., Якимова Л.Д., Эверт Н.А. Нейросетевая обработка критериев педагогического мастерства // Теоретические и прикладные вопросы современных информационных технологий. Материалы всеросс.конф. Улан-Уде: ВСГТУ, 2001. С.58-61
58. Жуков, Л. А. Технология нейросетевого решения прикладных классификационных задач для использования в экологии, биологии и медицине / Л. А. Жуков; КГТУ. Красноярск, 2004. 148 с. Деп. в ВИНИТИ 13.05.2004, № 800-В2004.
59. Жуков, Л. А. Технология нейросетевого решения прикладных классификационных задач в экологии, биологии и медицине: Дис. канд. техн. наук / Л. А. Жуков. Красноярск, 2000. 187 с.
60. Жуков, Л. А. Технология классификации с помощью нейронных сетей без учителя / Л. А. Жуков // Теоретические и прикладные вопросы современных информационных технологий. Материалы всеросс.конф. Улан-Уде: ВСГТУ, 2001. С. 40–47.