**ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ,
ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ
ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН**

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

На правах рукописи

УДК 004.0421:004.82

МАЯКУПОВ ЖАМШИД КУРБАНАЛИЕВИЧ

Разработка методов и алгоритмов оценки надежности сетей телекоммуникации на основе нейронных сетей

5А330201 – Компьютерные системы и их программное обеспечение

**Диссертация
на соискание академической степени
магистра**

**Научный руководитель
академик Камилов М. М.**

Ташкент – 2013

АННОТАЦИЯ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

Настоящая диссертационная работа посвящена исследованию современных методов и алгоритмов оценки надежности телекоммуникационных сетей, разработке и созданию методов и алгоритмов оценки надежности с помощью нейронных сетей, а также оптимизации работы нейронной сети путем модификации программного кода и параметров системы.

Актуальность исследования. На сегодняшний день, трудно представить нашу жизнь без средств связи и телекоммуникации. Телекоммуникация стало неотъемлемой частью жизни каждого из нас. И мы пользуемся их возможностями практический везде. Мы знаем что наш век, век информационных технологий (ИТ), время когда ИТ развиваются высокими темпами, и охватывает все сферы жизни. Рост размерности технических и программных средств характеризуется в настоящее время роста телекоммуникационных сетей, включающим от тысячи до миллиона элементов – узлов и каналов взаимодействия. С ростом размерности сети происходит радикальное перераспределение важности решения различных классов задач разработки и эксплуатации подобных объектов. Резко возрастает значимость анализа надежности таких сетей. Таким образом очень важной становится задача, посвященная разработке методов и алгоритмов оценки надежности телекоммуникационных сетей.

Объект и предмет исследования. Объектом исследования являются нейро-информационные методы диагностики систем на основе нейронных сетей, методика и средства вычислительного эксперимента по оценке корректности, достоверности и эффективности построенных моделей и алгоритмов. Предметом исследования является методы оценки надежности систем, модели и алгоритмы.

Цель и задачи исследования. Основной целью исследований является разработка методов и алгоритмов оценки надежности на основе нейронных сетей.

Для достижения поставленной цели в работе решаются следующие задачи:

- Обзор существующих методов и алгоритмов оценки надежности систем.
- Сравнительный анализ существующего программного обеспечения.
- Разработка методов и алгоритмов оценки надежности телекоммуникационных сетей на основе нейронных сетей.

- Анализ существующее состояние проблемы разработки моделей оценки и нейронных методов обработки информации при их оптимальном управлении для решения задач сетей телекоммуникации;
- Решение задач оценки надежности телекоммуникационных сетей с использованием нейронных сетей.

Методы исследования. В работе используются методы математического и структурного анализа, оптимизации программного кода, задействован математический аппарат для моделирования процессов нейронных сетей.

Научная новизна данной работы заключается в том, что в ходе исследований и разработки программного обеспечения были разработаны новые методы анализа и оптимизации работы системы путем внесения изменения в существующие методы и алгоритмы.

Научно-теоретическая и практическая значимость. Разработанная система методик позволяет оптимальным образом построить и настроить нейронные сети для оценки надежности телекоммуникационных сетей. Разработанное программное обеспечение может быть использовано в аналогичных системах для диагностики и прогнозирования, а также для обеспечения стабильности.

Структура и объём магистерской диссертационной работы. Данная диссертационная работа состоит из введения, трех глав, заключения, трех приложений и библиографического списка из 42 наименований. Работа изложена на 86 страницах, включая 1 таблицу и 19 рисунков.

В ходе выполнения диссертационной работы получены следующие результаты:

1. Исследование современное состояние оценки надежности систем позволяет решать задачу оценки системы при необходимом уровне достоверности.
2. При построении нейронных сетей для оценки надежности сети надо предварительно подготовить входные данные и привести их одному виду. Данные должны быть достоверными и обоснованными для получения реальные показатели надежности.
3. Количество нейронов в нейронном слою надо подобрать оптимальным образом. Так как повышения количество нейронов замедляет работы ПО, а понижения увеличивает искажения. При повышении количество нейронов больше 5 на одно измерения ПО обучения производилось

очень долго так как количество нейронов внутреннем слое достигало 390625 шт. Обучения производилась более 15 тыс. данными.

4. Классы было условно разделено на 5 (A, B, C, D, E). Но можно менять количество классов на большее количество или на меньшее.
5. Алгоритм можно усложнить и модифицировать для получения

болеелучших результатов и ускорить обучение сети.

Содержание

Введение

Настоящая диссертационная работа посвящена исследованию современных методов и алгоритмов оценки надежности телекоммуникационных сетей, разработке и созданию методов и алгоритмов оценки надежности с помощью нейронных сетей, а также оптимизации работы нейронной сети путем модификации программного кода и параметров системы.

Актуальность исследований

На сегодняшний день, трудно представить нашу жизнь без средств связи и телекоммуникации. Телекоммуникация стало неотъемлемой частью жизни каждого из нас. И мы пользуемся их возможностями практический везде. Мы знаем что наш век, век информационных технологий (ИТ), время когда ИТ развиваются высокими темпами, и охватывает все сферы жизни. Рост размерности технических и программных средств характеризуется в настоящее время роста телекоммуникационных сетей, включающим от тысячи до миллиона элементов – узлов и каналов взаимодействия. С ростом размерности сети происходит радикальное перераспределение важности решения различных классов задач разработки и эксплуатации подобных объектов. Резко возрастает значимость анализа надежности таких сетей. Таким образом очень важной становится задача, посвященная разработке методов и алгоритмов оценки надежности телекоммуникационных сетей.

В нашей Республике для развития ИТ сферы были приняты ряд законодательных документов, как Указ Президента Республики Узбекистан от 30 мая 2002г. № УП-3080 «О дальнейшем развитии компьютеризации и внедрении информационно-коммуникационных технологий» и в целях обеспечения практических мер по реализации стратегических приоритетов в области информационно-коммуникационных технологий принято Постановление Кабинета Министров Республики Узбекистан № 200 от 6 июня 2006г. «О мерах по

дальнейшему развитию компьютеризации и внедрению информационно-коммуникационных технологий».

Приняты основополагающие Законы Республики Узбекистан «Об информатизации», «Об электронной цифровой подписи» и «Об электронном документообороте». В Законе Республики Узбекистан «Об электронной цифровой подписи» принято понятие электронный документ – информация, зафиксированная в электронной форме, подтвержденная электронной цифровой подписью и имеющая другие реквизиты электронного документа, позволяющая его идентифицировать.

В Законе «Об информатизации» впервые принято понятие о собственнике информационных ресурсов или информационных систем, а также о владельце информационных ресурсов или информационных систем. Информационный ресурс в электронной форме стал основой для развития информатизации в государственных органах управления и власти, а также в секторах экономики страны и в других структурах. Информационные ресурсы и информационные системы должны формироваться на основе использования информации в электронном виде и эта информация при выполнении вышеуказанных требований может называться электронным документом.

АК «Узбектелеком» является крупнейшим оператором телекоммуникаций, который охватывает своей сетью всю территорию Республики Узбекистан. В целях дальнейшего развития телекоммуникационных сетей и внедрения современных услуг в 2012 году она осуществила ряд крупных инвестиционных проектов и продолжит работы в этом направлении. Так, в рамках реализации инвестиционной программы Республики Узбекистан на 2012 год было построено и сдано в эксплуатацию 175 километров оптико-волоконных линий связи в направлениях Бойсун-Денов и Ургут-Шахрисабз. Была увеличена емкость международных коммуникационных центров в 2 раза, а пропускная способность центра Международной пакетной коммуникации - в 4 раза.

Это позволило увеличить скорость соединения с интернетом до 40 Гбит/с и число его пользователей, а также снизить стоимость тарифов на услуги.

Для оказания современных телекоммуникационных услуг населению и повышения качества оказываемых услуг на основе технологий нового поколения сетей (NGN) была осуществлена модернизация коммутационных станций, эксплуатируемых в местных телекоммуникационных сетях. В целях развития услуг мобильной связи, в частности, организации передачи широкополосных данных через мобильные сети стандарта CDMA было установлено 140 базовых станций и построено 5 коммутационных центров. По итогам реализации данного проекта уровень охвата населения сетью достиг 71 процент.

На сегодняшний день со стороны филиала «ТШТТ» проложено 598,8 км волоконно-оптических линий связи до 2 075 зданий. В 873 зданиях города Ташкента создана возможность оказания услуг населению с емкостью 22 344 портов на основе технологии FTTB.

Объектом исследования является телекоммуникационные сети, средства обеспечения надежности работы телекоммуникационных систем и их узлов.

Предметом исследования является методы оценки надежности систем, модели и алгоритмы на основе нейронных сетей.

Основной целью исследований является разработка методов и алгоритмов оценки надежности на основе нейронных сетей.

Постановка задачи:

Основные результаты исследований сводятся к следующему:

- Обзор существующих методов и алгоритмов оценки надежности систем.
- Сравнительный анализ существующего программного обеспечения.
- Разработать методы и алгоритмы оценки надежности телекоммуникационных сетей на основе нейронных сетей.

- Анализировать существующее состояние проблемы разработки моделей оценки и нейронных методов обработки информации при их оптимальном управлении для решения задач сетей телекоммуникации;
- Решить задачи оценки надежности телекоммуникационных сетей с использованием нейронных сетей.

Основная проблема и гипотеза исследования. Основной проблемой в исследовании является задача разработки совокупности методик для оценки надежности телекоммуникационных сетей с использованием нейронных сетей. Гипотеза для решения данной проблемы заключается в том, что для построения такой системы необходим комплексный подход в выборе нейронных сетей, анализе их структуры, быстродействия и стабильности, модификации программного кода и разработке дополнительного программного обеспечения.

Методы исследования. В работе используются методы математического и структурного анализа, оптимизации программного кода, задействован математический аппарат для моделирования процессов.

Научно-теоретическая и практическая значимость исследования. Разработанная система методик позволяет оптимальным образом построить и настроить нейронные сети для оценки надежности телекоммуникационных сетей. Разработанное программное обеспечение может быть использовано в аналогичных системах для диагностики и прогнозирования, а также для обеспечения стабильности.

Научная новизна данной работы заключается в том, что в ходе исследований и разработки программного обеспечения были разработаны новые методы оценки надежности системы путем внесения изменения в существующие методы и алгоритмы.

Структура работы. Данная диссертационная работа состоит из введения, трех глав, заключения, трех приложений и библиографического

списка из 42 наименований. Работа изложена на 86 страницах, включая 1 таблицу и 19 рисунков.

Во введении рассмотрена актуальность данной работы, определена цель, поставлены задачи, описана структура работы.

В первой главе проведен анализ современных технологий по оценке надежности систем, приведена методы и их описания, анализ существующего программного обеспечения для оценки надежности, определены требования к разрабатываемой системе.

Во второй главе проведено исследование математического обеспечения нейронных сетей. Подробно изучены процессы формирования входных данных, обучения, структуры нейронной сети.

В третьей главе разработана и реализована методика оценки надежности телекоммуникационных сетей. Разработаны методы и приемы оптимизации работы и обеспечения стабильности системы. Описана программная реализация разработанных методов и приемов оптимизации работы системы, программная реализация дополнительного программного обеспечения, структура нейронной сети оценки надежности. Произведен анализ полученных результатов. Подведены итоги исследований.

В заключении приведен краткий отчет о проделанной работе.

Апробация работы Основные результаты исследований докладывались на международной научной конференции «Проблемы повышения качества кадров для отраслей связи и информатизации».

Опубликованность результатов. Основные результаты диссертации опубликованы в виде научных статей, материалов конференций. Список публикаций приведен в списке литературы.

Глава I. Анализ сушествующих меотдов для оценки надежности.

В данной главе, в результате обзора информационных источнков, проведены опорные понятия, анализ принципов оценки надежности и сравнительный анализ существующих программных продуктов для реализации системы мультимедийного вещания.

Появление глобальной "сети сетей" Internet и растущее громадными темпами количество ее пользователей (15% ежемесячно; в 1998 году ожидается 100 миллионов; с августа 1997 года по январь 1998-го, по данным компании Relevant Knowledge, число только американских пользователей WWW увеличилось на 10 миллионов) становится планетарным явлением, которое может привести даже к социальным изменениям.

Другими словами, мировое сообщество приближается к такой степени зависимости своего существования от функционирования информационных сетей, которая сравнима с зависимостью от систем обеспечения электроэнергией. Это кроме очевидных достоинств имеет и обратную сторону. Отказ сети связи может иметь последствия, превосходящие последствия аварий энергосистемы. В связи с этим проблема оценки и обеспечения надежности сетей является актуальной.

1. Надежность телекоммуникационных сетей

Термин телекоммуникации состоит из двух слов: теле (в переводе с греческого означает – "далеко") и коммуникация (в переводе с латыни – "сообщение, связь") и означает "дальняя связь" или "связь, сообщение на расстоянии".

Телекоммуникации - это любые формы связи, способы передачи информации на большие расстояния. Телекоммуникации – это также процессы передачи, получения и обработки информации на расстоянии с применением электронных, электромагнитных, сетевых, компьютерных и информационных технологий.

Знания и умения специалиста по телекоммуникационным технологиям примерно наполовину - это информационные технологии (программирование, настройка, конфигурирование, использование телекоммуникационных систем, оборудования, протоколов связи) и наполовину - знание принципов работы, умение проектировать телекоммуникационное оборудование, устройства и системы.

Основными отраслями телекоммуникаций на сегодняшний день являются: Интернет, мобильная связь, сети передачи данных (беспроводные, оптоволоконные и т.д.), спутниковые системы связи, цифровое и аналоговое телевидение, телефонная связь, электронный банкинг (см. рисунок).

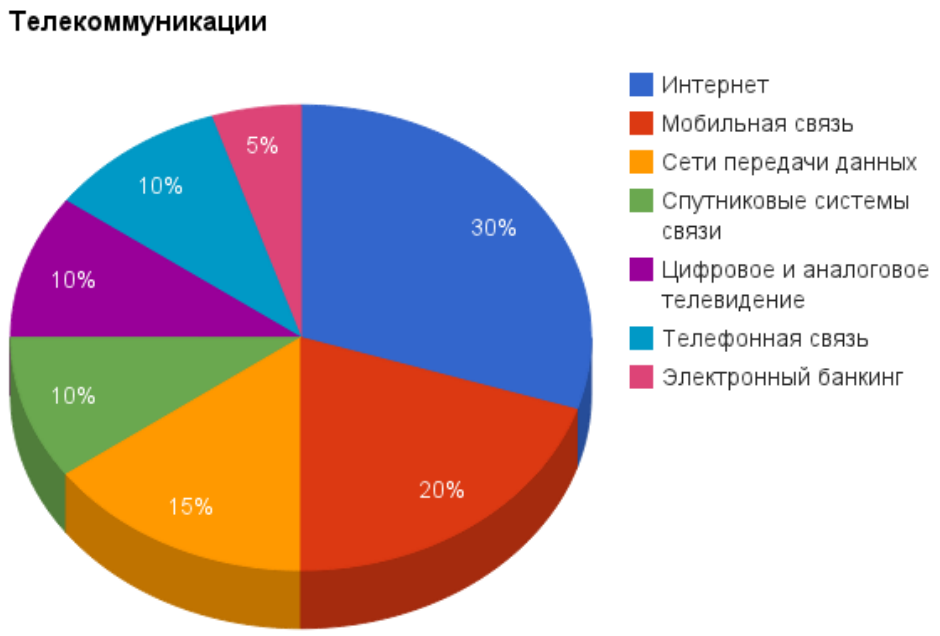


Рис.1. Диаграмма телекоммуникации.

По данным ЮНЕСКО, в настоящее время более половины трудоспособного населения развитых стран прямо или косвенно принимает участие в процессе производства и распределения информации. Три ведущие отрасли информационного сектора общественного производства (вычислительная техника, промышленная электроника и связь) играют сейчас для этих стран ту же роль, которую на этапе их индустриализации играла тяжелая промышленность.

Надежность - это свойство объекта (системы), заключающееся в его способности выполнять заданные функции при определенных условиях эксплуатации. Количественно надежность характеризуется рядом показателей, состав и способ определения которых зависят от типа анализируемой системы.

Теория надежности является основой инженерной практики в области надежности технических изделий. Часто безотказность определяют как вероятность того, что изделие будет выполнять свои функции на определенном периоде времени при заданных условиях. Математически это можно записать следующим образом:

$$R(t) = Pr\{T > t\} = \int_t^\infty f(x)\,dx,$$

где - функция плотности времени наработки до отказа, а t – продолжительность периода времени функционирования изделия, в предположении, что изделие начинает работать в момент времени t=0. Теория надежности предполагает следующие четыре основных допущения:

- Отказ рассматривается как случайное событие. Причины отказов, соотношения между отказами (за исключением того, что вероятность отказа есть функция времени) задаются функцией распределения. Инженерный подход к надежности рассматривает вероятность безотказной работы как оценку на определенном статистическом доверительном уровне.
- Надежность системы тесно связана с понятием «заданная функция системы». В основном, рассматривается режим работы без отказов. Однако, если в отдельных частях системы нет отказов, но система в целом не выполняет заданных функций, то это относится к техническим требованиям к системе, а не к показателям надежности.
- Надежность системы может рассматриваться на определенном отрезке времени. На практике это означает, что система имеет шанс

(вероятность) функционировать это время без отказов. Характеристики (показатели) надежности гарантируют, что компоненты и материалы будут соответствовать требованиям на заданном отрезке времени. В общем случае надежность относится к понятию «наработка», которое в зависимости от назначения системы и условий ее применения определяет продолжительность или объем работы. Наработка может быть как непрерывной величиной (продолжительность работы в часах, километраж пробега в милях или километрах и т.п.), так и целочисленной величиной (число рабочих циклов, запусков, выстрелов оружия и т.п.).

- Согласно определению, надежность рассматривается относительно заданных режимов и условий применения. Это ограничение необходимо, так как невозможно создать систему, которая способна работать в любых условиях. Внешние условия функционирования системы должны быть известны на этапе проектирования.

Методы оценки надежности существует не первый день. Рассмотрим некоторые из них:

1. Структурные методы расчета надежности.

Структурные методы являются основными методами расчета показателей надежности в процессе проектирования объектов, поддающихся разукрупнению на элементы, характеристики надежности, которых в момент проведения расчетов известны или могут быть определены другими методами. Расчет показателей надежности структурными методами в общем случае включает:

- представление объекта в виде структурной схемы, описывающей логические соотношения между состояниями элементов и объекта в целом с учетом структурно-функциональных связей и взаимодействия элементов, принятой стратегии обслуживания, видов и способов резервирования и других факторов;

- описание построенной структурной схемы надежности объекта адекватной математической моделью, позволяющей в рамках введенных предположений и допущений вычислить показатели надежности объекта по данным о надежности его элементов в рассматриваемых условиях применения.

В качестве структурных схем надежности могут применяться:

- схемы функциональной целостности;
- структурные блок-схемы надежности;
- деревья отказов;
- графы состояний и переходов.

2. Логико-вероятностный метод.

В логико-вероятностных методах (ЛВМ) исходная постановка задачи и построение модели функционирования исследуемого системного объекта или процесса осуществляется структурными и аналитическими средствами математической логики, а расчет показателей свойств надежности, живучести и безопасности выполняется средствами теории вероятностей.

ЛВМ являются методологией анализа структурно-сложных систем, решения системных задач организованной сложности, оценки и анализа надежности, безопасности и риска технических систем. ЛВМ удобны для исходной формализованной постановки задач в форме структурного описания исследуемых свойств функционирования сложных и высоко-размерных систем. В ЛВМ разработаны процедуры преобразования исходных структурных моделей в искомые расчетные математические модели, что позволяет выполнить их алгоритмизацию и реализацию на ЭВМ.

3. Общий логико-вероятностный метод.

Необходимость распространения ЛВМ на немонотонные процессы привела к созданию общего логико-вероятностного метода (ОЛВМ). В ОЛВМ расчета надежности аппарат математической логики используется для первичного графического и аналитического описания условий реализации функций отдельными и группами элементов в проектируемой системе, а методы теории вероятностей и комбинаторики применяются для количественной оценки безотказности и/или опасности функционирования проектируемой системы в целом. Для использования ОЛВМ должны задаваться специальные структурные схемы функциональной целостности исследуемых систем, логические критерии их функционирования, вероятностные и другие параметры элементов.

В основе постановки и решения всех задач моделирования и расчета надежности систем с помощью ОЛВМ лежит так называемый событийно-логический подход. Этот подход предусматривает последовательное выполнение следующих четырех основных этапов ОЛВМ:

- этап структурно-логической постановки задачи;
- этап логического моделирования;
- этап вероятностного моделирования;
- этап выполнения расчетов показателей надежности.

4. Метод деревьев отказов.
5. Метод Марковского моделирования.

Программные средства, предназначенные для анализа и расчета надежности, готовности и ремонтопригодности (в алфавитном порядке):

- АРБИТР
- АРМ Надежности
- АСОНИКА-К
- AnyGraph
- CRISS
- BlockSim
- ITEM Software

- Reliability Workbench
- Windchill

2. Изучение существующих методов и программных средств для решения данной задачи.

Программный комплекс АРБИТР.

АРБИТР - это программный комплекс «Автоматизированного Расчета Безопасности И Технического Риска». В настоящее время ПК АРБИТР позволяет автоматически строить математические модели и рассчитывать показатели свойств надежности, стойкости, живучести, устойчивости, технического риска, ожидаемого ущерба и эффективности, а также решать задачи оптимизации надежности. Предназначен для инженеров-проектировщиков, работающих в различных отраслях промышленности, для проведения научных исследований и организации учебного процесса.

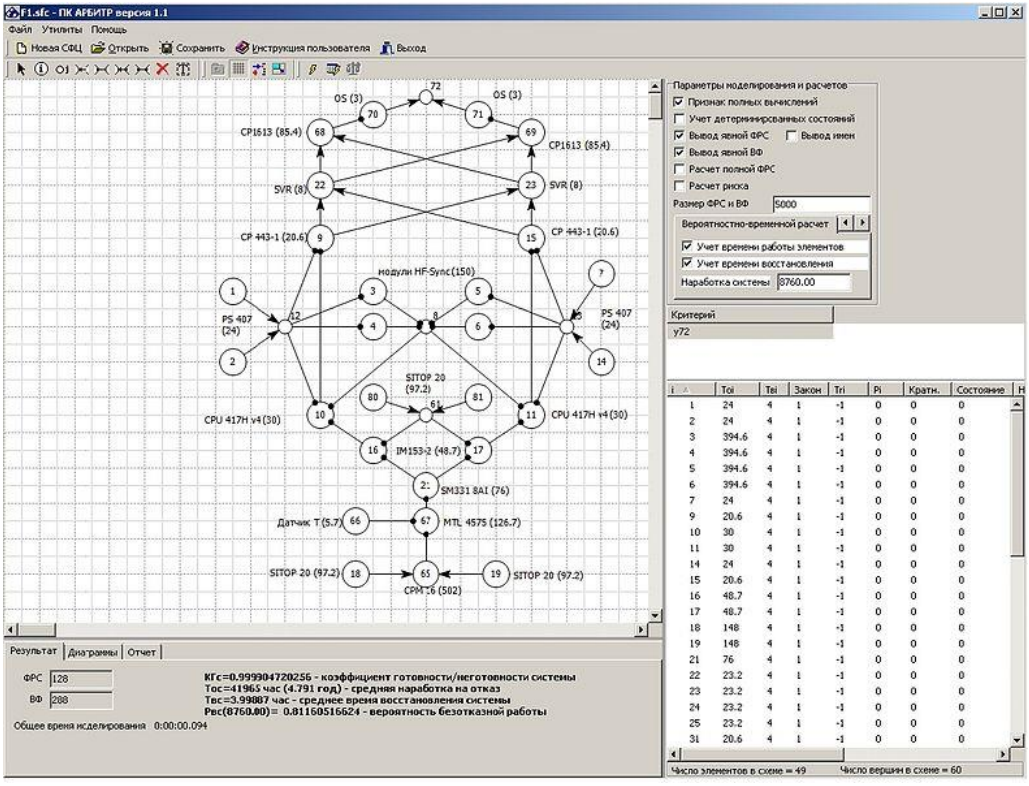


Рис. 2. Интерфейс программы АРБИТР.

Предыдущие названия программного комплекса: ПК "АСМ", ПК "АСМ 2001", ПК "АСМ СЗМА".

Программный комплекс ПК АРБИТР был аттестован в "Совете по аттестации программных средств" Научно-технического центра по ядерной и радиационной безопасности (НТЦ ЯРБ) Федеральной службы по экологическому, технологическому и атомному надзору (Ростехнадзор) РФ. АРБИТР аттестован 21 февраля 2007 г. сроком на 10 лет и разрешен к применению на предприятиях Ростехнадзора РФ.

Теоретической основой программного комплекса является общий логико-вероятностный метод. В качестве графического средства описания функционирования систем используется схема функциональной целостности.

Основные возможности

- представление в исходной СФЦ (в суперграфе СФЦ) до 400 элементов (вершин) и до 100 элементов в каждой декомпозированной вершине (подграфах СФЦ) основного графа исследуемой системы (т.е. возможность ввода до 40 000 вершин);
- автоматическое построение логических функций, представляющих кратчайшие пути успешного функционирования (КПУФ), минимальные сечения отказов (МСО) или их немонотонные комбинации (явные детерминированные модели исследуемых свойств системы);
- автоматическое построение вероятностных функций, обеспечивающих точный расчет показателей устойчивости, эффективности и риска исследуемых систем;
- расчет вероятности реализации заданных критериев, представляющих свойства устойчивости (надежности, стойкости, живучести) и безопасности (технического риска, вероятностей возникновения аварийных ситуаций и аварий) систем;
- расчет вероятности безотказной работы или отказа и средней наработки до отказа невосстанавливаемых систем;

- расчет коэффициента готовности, средней наработки на отказ, среднего времени восстановления и вероятности безотказной работы восстанавливаемых систем;
- расчет вероятности готовности смешанных систем, состоящих из восстанавливаемых и невосстанавливаемых элементов;
- расчет значимостей, положительных и отрицательных вкладов всех элементов исследуемой системы в вероятность реализации исследуемого свойства, используемые для выработки и обоснования управленческих решений по обеспечению устойчивости, живучести, безопасности эффективности и риска функционирования;
- вспомогательный режим приближенных расчетов, которые выполняются по двум методикам: для независимых отказов элементов (аналог методики, используемой в комплексах "Risk Spectrum", Швеция) и "Saphire-7" (США)), и с учётом трех типов отказов элементов – "отказ на требование", "отказ в режиме работы" и "скрытый отказ в режиме ожидания" (методы разработаны специалистами ФГУП ОКБМ им И.И.Африкантова и впервые реализованы в аттестованном комплексе "CRISS 4.0");
- расчет вероятности реализации отдельных КПУФ или МСО системы;
- расчет значимости и суммарной значимости сечений отказов по Fussell-Vesely;
- расчет значимости, уменьшения и увеличения риска элементов по Fussell-Vesely;
- приближенный расчет вероятностных характеристик системы с учётом трех типов отказов элементов: отказ на требование, отказ в режиме работы и скрытый отказ в режиме ожидания (по методике, реализованной в ПК CRISS 4.0);
- структурный и автоматический учёт отказов групп элементов по общей причине (модели альфа-фактора, бета-фактора и множественных греческих букв);

- учёт различных видов зависимостей и множественных состояний элементов, представляемых с помощью групп несовместных событий;
- учёт двухуровневой декомпозиции структурной схемы, дизъюнктивных и конъюнктивных кратностей сложных элементов (подсистем);
- учёт неограниченного числа циклических (мостиковых) связей между элементами и подсистемами;
- учёт различных комбинаторных отношений (К из N) между группами элементов.

Система АСОНИКА.

Система АСОНИКА сориентирована на разработчика РЭА. В состав комплекса АСОНИКА входят 18 подсистем, связанных с моделированием аэродинамических, тепловых, механических воздействий на радиоэлектронную аппартуру.

В подсистемах АСОНИКА-М и АСОНИКА-ТМ разработаны специальные интерфейсы для ввода типовых конструкций радиоэлектронной аппаратуры (РЭА) – шкафов, блоков, печатных узлов, что значительно упрощает анализ физических процессов в РЭА. Если бы пользователь строил модель механических процессов сложного шкафа или блока в обычной конечноэлементной системе, например, ANSYS, ему бы пришлось вначале пройти специальное обучение и набраться опыта, что заняло бы примерно около года, а затем в течение нескольких часов вводить саму модель. В системе АСОНИКА не нужно проходить специального обучения, нужно просто вводить на доступном конструктору языке то, что представлено на чертеже. Ввод того же сложно шкафа может быть осуществлен в течение получаса.

Таким образом, полноценный комплексный анализ шкафа на тепловые и механические воздействия вплоть до каждого

электрорадиоизделия (ЭРИ) (получаем ускорения и температуры на каждом элементе) может быть проведен в течение 1 дня.

Структура справочной базы данных по параметрам ЭРИ и параметрам материалов конструкций РЭА отличается от существующих наличием полных условных записей ЭРИ, наличием моделей вариантов установки ЭРИ, позволяющих значительно сократить время на ввод геометрических, физико-механических, тепловых и др. параметров ЭРИ, возможностью создания новых моделей вариантов установки ЭРИ, наличием необходимых графических параметров, позволяющих придавать реалистичность изображению ЭРИ в пространстве, возможностью создания дополнительных таблиц параметров ЭРИ, содержащих числовые, строковые, функциональные, логические, текстовые и графические данные об ЭРИ.

Система является открытой, так как позволяет включать дополнительные программы, например Pro/ENGINEER, на уровне пользователя без привлечения программистов. На территории РФ данная система аналогов не имеет и разрабатывается впервые. По зарубежным аналогам информация в открытой печати отсутствует.

Система «АСОНИКА» с 2000 г. проходит апробацию на предприятиях, выполняющих ГОЗ и поддерживается 22 ЦНИИ МО РФ, выпустившим в 2000 г . (ред. 2003 г .) РДВ 319.01.05-94 по применению автоматизированной системы «АСОНИКА» с целью повышения надежности проектируемой аппаратуры на основе математического моделирования, электрических, тепловых, аэро(-гидро)динамических и механических (вибраций, ударов, линейных ускорениях, акустических шумов) процессов.

Программное обеспечение AnyGraph.

Программное обеспечение AnyGraph создано с целью упростить разработку системных моделей используемых при расчете надежности сложных технических систем и их анализ.

Теоретические и методологические основы программы.

Теоретической основой ПО AnyGraph являются логико-вероятностные методы (ЛВМ) моделирования, количественной оценки надежности и безопасности, сложных технических систем, разработанные под руководством профессора Рябинина И.А. ПО AnyGraph использует наиболее эффективную разновидность ЛВМ а именно разработанный в начале 90-х годов профессором А.С. Можаевым метод системного анализа сложных технических систем – общий логико-вероятностный метод.

“Логико-вероятностными методами называются методы системного анализа, в которых аппарат математической логики используется для первичного структурного и промежуточного аналитического описания знаний о правилах и условиях функционирования элементов в исследуемой системе, а методы теории вероятностей применяются для количественной оценки различных свойств рассматриваемой системы, на основе заданных вероятностных и других параметров ее элементов.

Базовой концепцией ПО AnyGraph является представление модели как набора взаимодействующих между собой узлов (технических элементов) и логических связей между ними. Логические связи определяют конкретные условия реализации узлами (элементами технической системы) своих функций.

Построенная с помощью графического редактора ПО AnyGraph модель имеет высокую степень наглядности, как правило, структура созданной модели повторяет принципиальные технологические схемы разработанные на различных этапах создания технической документации исследуемых проектов.

Методологическая основа ЛВМ характеризуется следующими положениями:

- состояние узла в моделях надежности и безопасности кодируется двумя несовместными исходами нулем или единицей:
- 1 – состояние работоспособности узла модели (включено устройство, выполнено какое либо действие и т.д.)
- 0 – отказовое состояние узла модели (не включено устройство, не выполнено какое либо действие и т.д.)
- основным способом описания задач является графическое построение схемы надежности или сценария возникновения аварии исследуемой системы.
- с помощью функций алгебры логики записывается условие работоспособности (не работоспособности) модели через состояния ее элементов.
- полученная функция алгебры логики, является основной формой представления модели, преобразуется из логической формы в вероятностную с помощью которой проводится количественная оценка различных свойств исследуемой системы.

Функционал и состав программы.

Функционал представленный ПО AnyGraph представляет универсальную альтернативу программному обеспечению в котором основным аппаратом моделирования и расчета показателей надежности и безопасности сложных технических систем являются деревья отказов, деревья событий, графы связности а также марковские модели.

ПО AnyGraph состоит из двух модулей:

Графический модуль, с помощью которого осуществляется:

- проектирование и документирование модели исследуемой системы

- передача в Математический модуль созданной модели исследуемой системы для выполнения точных аналитических или статистических расчетов

- отображение результатов выполненных расчетов

Математический модуль, с помощью которого проводятся расчеты различных показателей надежности исследуемой системы.

Blocksim: система надежности и ремонтопригодности анализ программных средств

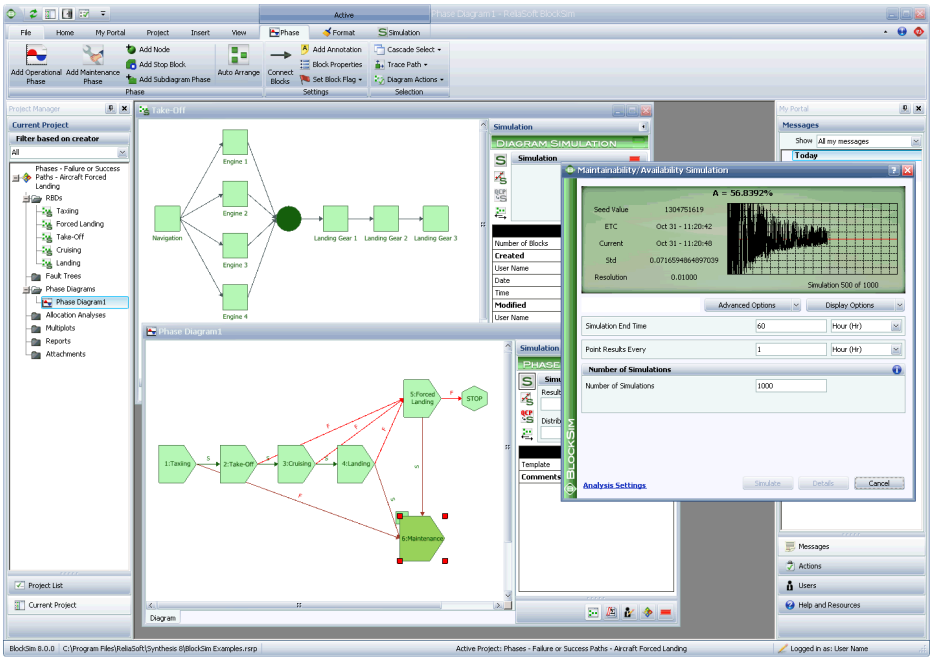


Рис. 3. Интерфейс программы Blocksim.

ReliaSoft инструмент предоставляет комплексную платформу для системы обеспечения надежности, готовности, ремонтопригодности и соответствующий анализ .

Программное обеспечение предлагает сложный графический интерфейс, который позволяет моделировать простейшие или самые сложные системы и процессы с помощью блок-схем надежности (RBDS) или анализ дерева отказов (FTA) - или комбинация обоих подходов!

Особенности программы

BlockSim поддерживает обширный набор блок-схемы надежности (RBD) конфигураций ианализ дерева отказов (FTA) ворота и событий, в том числе расширенные возможности для моделирования сложных конфигураций, распределение нагрузки, в режиме ожидания избыточности, фазы и рабочих циклов. Использование точных вычислений и / или моделирования дискретных событий, BlockSim облегчает широкий

спектр анализов для обоих ремонту и невосстанавливаемых систем. Это включает в себя:

- Анализ надежности системы
- Идентификация критических компонентов (Меры Надежность значения)
- Оптимальное распределение Надежность
- Поддержку системы анализа (определить оптимальные интервалы профилактического технического обслуживания, запасных частей положения и т.д.)
- Доступность системы анализа (Рассчитать работы, простоя, доступность и т. д.)
- Пропускная Расчет (выявить узкие места, оценка емкости производства и др.)
- Жизненный цикл анализа затрат

Синтез интеграции

С выходом версии 8 BlockSim анализов хранятся в централизованной базе данных, которая поддерживает одновременный доступ нескольких пользователей и акций соответствующей информацией между синтезом надежности с поддержкой программных средств. Для корпоративного уровня хранилищах, как Microsoft SQL Server ® и Oracle ® поддерживаются.

Доступно обучение

ReliaSoft предлагает три учебных курсов, направленных на тему системы надежности, ремонтопригодности и соответствующий анализ с использованием диаграммы надежности блока (RBD) или анализ дерева отказов (FTA) подхода. Объединив прочную теоретическую основу с практическими примерами применения и практическую подготовку по использованию программного средства BlockSim, эти курсы даст вам знания и навыки вам нужно для успешного применения этих важных методов надежности.

Применения и преимущества

Программное обеспечение BlockSim предоставляет широкий спектр инструментов, которые помогут вам моделирования и анализа систем и / или процессов. Некоторые из потенциальных приложений и преимущества использования блок-схем надежности (RBDS), анализ дерева отказов (FTA) и BlockSim ReliaSoft программного обеспечения включают в себя:

- Выявление критических компонентов (или отказов) и определить наиболее эффективные пути для повышения производительности системы путем усовершенствования конструкции и / или планирование технического обслуживания.
- Использование моделирования для получения расчетной метрикам производительности, которые могут способствовать принятию решений в различных областях, таких как планирование планового технического обслуживания, планирование на запчасти, выявления узких мест в производстве и пропускная оценке стоимости жизненного цикла.
- Определение уязвимостей в системе и определить наиболее эффективные способы снижения риска.

Reliability Workbench.

Reliability Workbench является флагманом Люкс Изограф по разработке программного обеспечения надежности, безопасности и ремонтопригодности. Программное обеспечение находится в непрерывном развитии с 1980 года. Это простой в использовании и является отличным инструментом для всех надежности и техническому обслуживанию профессионалов.

Ключевые особенности Reliability Workbench

Reliability Workbench включает в себя все инструменты, которые понадобятся для управления надежностью и безопасностью исследования:

- FMECA и FMEA
- FaultTree + Анализ дерева отказов

– Надежность Блок-схема анализа

– Распределение Надежность и роста

– Событие Дерево и анализа Маркова

– Вейбулла анализа исторических данных недостаточностью

– Комплексная Библиотеки запчастей

– Расширенные средства построения отчетов

– Импорт и экспорт объектов

– Предприятие инструмент классового сотрудничества

Reliability Workbench позволяет разрабатывать проекты с использованием одной или нескольких интегрированных модулей анализа.

Каждый из модулей является мощным приложением в своем собственном праве и может быть использован самостоятельно, но большей мощности достигается за счет интеграции модулей в окружающей среде Reliability Workbench.

Модули могут динамически обмениваться данными для простоты и последовательности. Пользователю нужно только входные данные один раз, но может использовать его несколько раз.

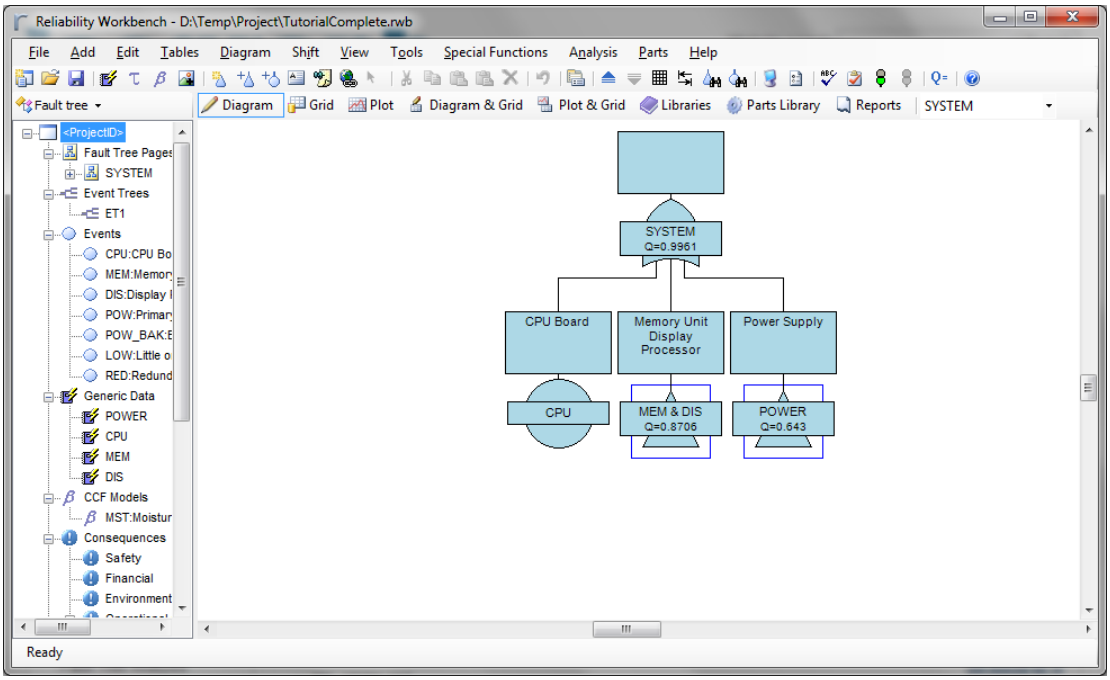


Рис. 4. Интерфейс программы Reliability Workbench.

При анализе параметров системной надежности учитывается структура системы, состав и взаимодействие входящих в нее элементов, возможность перестройки структуры и алгоритмов ее функционирования при отказах отдельных элементов.

Наиболее часто в инженерной практике рассматривают последовательное, параллельное, смешанной (последовательно-параллельное и параллельно-последовательное) соединение элементов, а также схемы типа «K из N», мостиковые соединения.

По возможности восстановления и обслуживания системы подразделяются на восстанавливаемые и невосстанавливаемые, обслуживаемые и необслуживаемые. По режиму применения (функционирования) – на системы непрерывного, многократного (циклического) и однократного применения.

3. Задача классификации в нейронных сетях

Для решения поставленной задачи будет браться за основу задачи типа «Классификация».

Решение задачи классификации является одним из важнейших применений нейронных сетей.

Задача классификации представляет собой задачу отнесения образца к одному из нескольких попарно не пересекающихся множеств. Примером таких задач может быть, например, задача определения кредитоспособности клиента банка, медицинские задачи, в которых необходимо определить, например, исход заболевания, решение задач управления портфелем ценных бумаг (продать купить или "придержать" акции в зависимости от ситуации на рынке), задача определения жизнеспособных и склонных к банкротству фирм.

Сети с прямой связью являются универсальным средством аппроксимации функций, что позволяет их использовать в решении задач классификации. Как правило, нейронные сети оказываются наиболее эффективным способом классификации, потому что генерируют

фактически большое число регрессионных моделей (которые используются в решении задач классификации статистическими методами).

К сожалению, в применении нейронных сетей в практических задачах возникает ряд проблем. Например, заранее не известно, какой сложности (размера) может потребоваться сеть для достаточно точной реализации отображения. Эта сложность может оказаться чрезмерно высокой, что потребует сложной архитектуры сетей. Так Минский в своей работе "Персептроны" доказал, что простейшие однослойные нейронные сети способны решать только линейно разделимые задачи. Это ограничение преодолимо при использовании многослойных нейронных сетей. В общем виде можно сказать, что в сети с одним скрытым слоем вектор, соответствующий входному образцу, преобразуется скрытым слоем в некоторое новое пространство, которое может иметь другую размерность, а затем гиперплоскости, соответствующие нейронам выходного слоя, разделяют его на классы. Таким образом сеть распознает не только характеристики исходных данных, но и "характеристики характеристик", сформированные скрытым слоем.

Для построения классификатора необходимо определить, какие параметры влияют на принятие решения о том, к какому классу принадлежит образец. При этом могут возникнуть проблемы, такие как, если количество параметров мало, то может возникнуть ситуация, при которой один и тот же набор исходных данных соответствует примерам, находящимся в разных классах. Тогда невозможно обучить нейронную сеть, и система не будет корректно работать (невозможно найти минимум, который соответствует такому набору исходных данных). Исходные данные обязательно должны быть непротиворечивы. Для решения этой проблемы необходимо увеличить размерность пространства признаков (количество компонент входного вектора, соответствующего образцу). Но при увеличении размерности пространства признаков может

возникнуть ситуация, когда число примеров может стать недостаточным для обучения сети, и она вместо обобщения просто запомнит примеры из обучающей выборки и не сможет корректно функционировать. Таким образом, при определении признаков необходимо найти компромисс с их количеством.

Далее необходимо определить способ представления входных данных для нейронной сети, т.е. определить способ нормирования. Нормировка необходима, поскольку нейронные сети работают с данными, представленными числами в диапазоне 0..1, а исходные данные могут иметь произвольный диапазон или вообще быть нечисловыми данными. При этом возможны различные способы, начиная от простого линейного преобразования в требуемый диапазон и заканчивая многомерным анализом параметров и нелинейной нормировкой в зависимости от влияния параметров друг на друга.

Для решения подобного рода задач хорошо подходят искусственные нейронные сети, так как для оценки надежности телекоммуникационных сетей нужно учесть огромное количество факторов, а также трудно моделировать зависимость надежности от этих факторов.

Потенциальными областями применения искусственных нейронных сетей являются те, где человеческий интеллект малоэффективен, а традиционные вычисления трудоемки или физически неадекватны (т.е. не отражают или плохо отражают реальные физические процессы и объекты).

Действительно, актуальность применения нейронных сетей многократно возрастает тогда, когда появляется необходимость решения плохо формализованных задач.

Типовые задачи, решаемые с помощью нейронных сетей и нейрокомпьютеров следующие:

- автоматизация процесса классификации;
- автоматизация прогнозирования;
- автоматизация процесса предсказания;

– автоматизация процесса принятия решений;
– управление;
– кодирование и декодирование информации;
– аппроксимация зависимостей и др.

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от нашей кожи, ушей и глаз к мозгу, процессы мышления и управления действиями – все это реализовано в живом организме как передача электрических импульсов между нейронами.

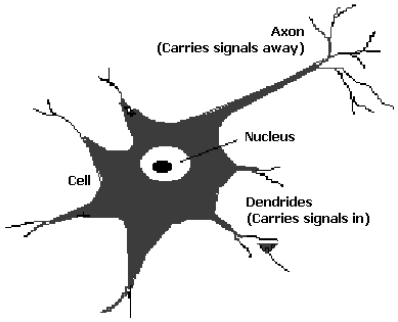


Рис. 5. Биологический (или естественный) нейрон

Биологический нейрон (Cell) имеет ядро (Nucleus), а также отростки нервных волокон двух типов – дендриты (Dendrites), по которым принимаются импульсы (Carries signals in), и единственный аксон (Axon), по которому нейрон может передавать импульс (Carries signals away). Аксон контактирует с дендритами других нейронов через специальные образования – синапсы (Synapses), которые влияют на силу передаваемого импульса.

Структура, состоящая из совокупности большого количества таких нейронов, получила название биологической (или естественной) нейронной сети.

Искусственный нейрон (далее – нейрон) является основой любой искусственной нейронной сети.

Нейроны представляют собой относительно простые, однотипные элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены и заторможены.

Искусственный нейрон, также как и его естественный прототип, имеет группу синапсов (входов), которые соединены с выходами других нейронов, а также аксон – выходную связь данного нейрона – откуда сигнал возбуждения или торможения поступает на синапсы других нейронов.

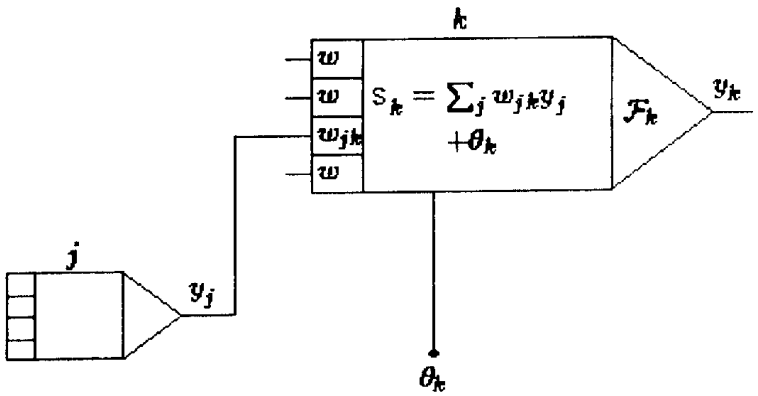


Рис. 6. Искусственный нейрон – простейший элемент искусственной нейронной сети

yj – сигнал, поступающий от нейрона *j;*

sk – скалярное произведение вектора входных сигналов и вектора весов;

fk – функция возбуждения; *yk* – выходной сигнал нейрона

Каждый синапс характеризуется величиной *синаптической связи* или *весом wi*, который по своему физическому смыслу эквивалентен электрической проводимости.

Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} xi\ wi$$

где *x* – вход нейрона, а *w* – соответствующий этому входу вес.

Выводы по главе I

В главе дана анализ методов и существующих программных обеспечений по оценке надежности системы.

В основном, в качестве параметра надежности используется среднее время до отказа (MTTF), которое может быть определено через интенсивность отказов или через число отказов на заданном отрезке времени. Интенсивность отказов математически определяется как условная плотность вероятности возникновения отказа изделия при условии, что до рассматриваемого момента времени отказ не произошел. При увеличении интенсивности отказов среднее время до отказа уменьшается, надежность изделия падает. Обычно среднее время до отказа измеряется в часах, но также может выражаться в таких единицах, как циклы и мили.

В других случаях надежность может выражаться через вероятность выполнения задачи. Например, надежность полетов гражданской авиации может быть безразмерной, или иметь размерность в процентах, как это делается в практике системной безопасности. В отдельных случаях успешным результатом системы может являться едино-разовое срабатывание. Это актуально для систем, которые рассчитаны на срабатывание всего 1 раз: например, подушки безопасности в автомобиле. В этом случае задается вероятность срабатывания или, как, например, для ракет, вероятность попадания в цель. Для таких систем мерой надежности является вероятность срабатывания. Для восстанавливаемых систем может задаваться такой параметр, как среднее время восстановления (ремонта) и время проверки (тестирования). Часто параметры надежности задаются в виде соответствующих статистических доверительных интервалов.

Нейронные сети продемонстрировали свою способность решать сложные задачи. Они имеют уникальные потенциальные возможности, хотя не свободны от ограничений и вопросов, на которые до сих пор не существует ответа. Такая ситуация настраивает на умеренный оптимизм.

Глава II. Оценка надежности сети на основе нейронных сетей

Надежность системы зависит от большого числа факторов, таких как структура, аппаратные узлы, ПО станций, загруженности каналов связи, природный факторы, человеческий фактор и т.д. Так как вид этих зависимостей неизвестен, то стандартные методы анализа неэффективны в задаче оценки надежности. Как правило, эта задача решается экспертами, работающими в данной области. Недостатком такого подхода является субъективность оценщика, а также возможные разногласия между различными экспертами.

Система на основе нейронной сети способна эффективно решать широкий спектр задач объективной оценки. Обычно, для уверенной оценки опытному эксперту необходимо провести достаточно много тестов и учесть факторы, что занимает много ресурсов. Система на основе нейронной сети способна с той же достоверностью определить надежность системы в течение всего нескольких минут, причем без участия квалифицированного персонала.

В данной главе проводится исследование математического обеспечения нейронных сетей, а также методы их обучения.

1. Нейронные сети Кохонена.

Искусственный нейрон имитирует в первом приближении свойства биологического нейрона. На вход искусственного нейрона поступает некоторое множество сигналов, каждый из которых является выходом другого нейрона. Каждый вход умножается на соответствующий вес, аналогичный синоптической силе, и все произведения суммируются, определяя уровень активации нейрона. Множество входных сигналов, обозначенных x1, x2,:, xn, поступает на искусственный нейрон. Эти входные сигналы, в совокупности, обозначаемые вектором **X,** соответствуют сигналам, приходящим в синапсы биологического нейрона. Каждый сигнал умножается на соответствующий вес w1, w2,..., wn, и

поступает на суммирующий блок. Каждый вес соответствует "силе" одной биологической синоптической связи. (Множество весов в совокупности обозначается вектором **W**.) Суммирующий блок, соответствующий телу биологического элемента, складывает взвешенные входы алгебраически, создавая выход.

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

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

Существует много разновидностей нейронных сетей, но наиболее распространенными являются сети Кохонена, Хопфилда, обратного распространения ошибки.

Для применения нейронных сетей Кохонена в задачах классификации требуется некоторая формализация. Каждый объект, который требуется классифицировать, представляется в виде некоторого вектора,

подающегося на вход нейронной сети. Количество нейронов во входном слое определяется количеством компонентов этого входного вектора. Количество же выходов определяется количеством классов, т.е. если всего M классов, то количество нейронов в выходном слое тоже будет M. Таким образом, каждый нейрон в выходном слое «отвечает» за свой класс. Значения, которые принимают нейроны в выходном слое, отображают насколько вектор классифицируемого объекта на входе близок, по мнению нейронной сети Кохонена, к тому или иному классу. Чем больше «уверенность», что объект принадлежит к тому или иному классу, тем больше значение принимает нейрон соответствующего класса. Иногда применяют специальную функцию активацию, которая делает сумму выходов со всех нейронов равной единице. В таком случае каждый выход можно трактовать, как вероятность того, что объект принадлежит к данному классу.

Стоит отметить, что существует или более простая реализация нейронной сети Кохонена, которая называется «победитель забирает все». В таком случае каждый нейрон выходного слоя может принимать значение либо ноль, либо единица. При этом для одного входного вектора единице может быть равен один и только один нейрон выходного слоя, т.е. один объект не может относиться сразу к двум классам.

2. Сети векторного квантования, обучаемые с учителем

Нейронная сеть Кохонена - это широкий класс сетей, насчитывающий несколько десятков реализаций, объединяемый общим тезисом "Победитель получает все". Сеть обучается "без учителя": просматривается входная выборка векторов и выявляются законы их распределения. **Сеть** Кохонена используется для решения двух задач: наилучшей квантизации и выделения главных компонент. В этом качестве она может использоваться самостоятельно или же в качестве первого слоя сети более сложной архитектуры (например, сети Кохонена-Гроссберга).

Слой Кохонена состоит из некоторого количества n параллельно действующих линейных элементов. Все они имеют одинаковое число входов m и получают на свои входы один и тот же вектор входных сигналов $x = (x_1, ...x_m)$. На выходе jго линейного элемента получаем сигнал

$$y_j = w_{j0} + \sum_{i=1}^{m} w_{ji} x_i,$$

где w_{ji} — весовой коэффициент iго входа jго нейрона, w_{j0} — пороговый коэффициент.

После прохождения слоя линейных элементов сигналы посылаются на обработку по правилу «победитель забирает всё»: среди выходных сигналов y_j ищется максимальный; его номер $j_{\max} = \arg\max_j \{y_j\}$. Окончательно, на выходе сигнал с номером $j_{\max}$ равен единице, остальные — нулю. Если максимум одновременно достигается для нескольких $j_{\max}$, то либо принимают все соответствующие сигналы равными единице, либо только первый в списке (по соглашению).

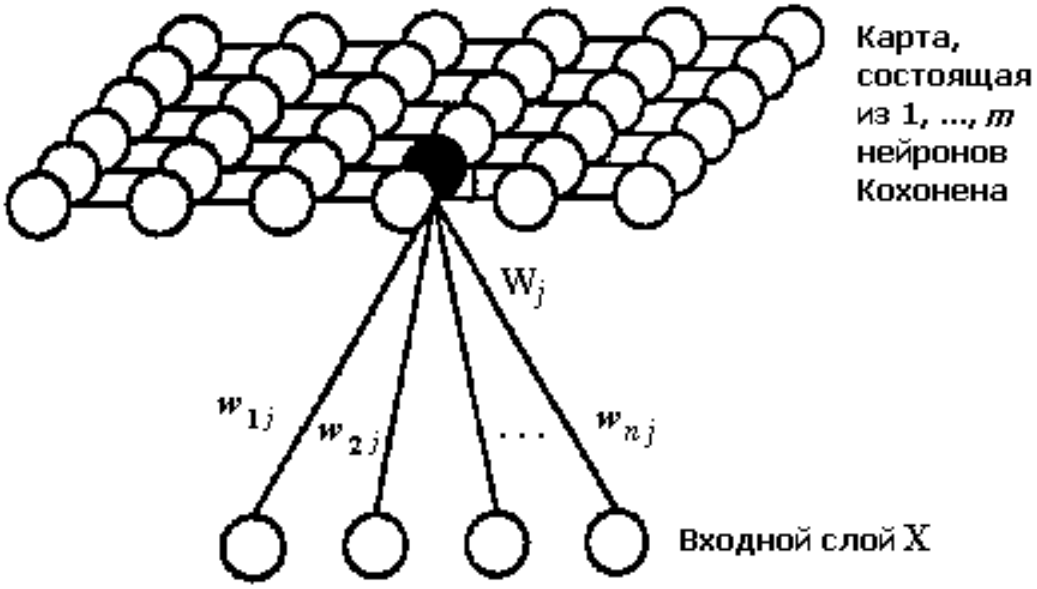


Рис. 7. Нейронная сеть Кохонена

Большое распространение получили слои Кохонена, построенные следующим образом: каждому (jму) нейрону сопоставляется

точка $W_j = (w_{j1}, ...w_{jm})$ в m-мерном пространстве (пространстве сигналов). Для входного вектора $x = (x_1, ...x_m)$вычисляются его евклидовы расстояния $\rho_j(x)$ до точек W_j и «ближайший получает всё» — тот нейрон, для которого это расстояние минимально, выдаёт единицу, остальные — нули. Следует заметить, что для сравнения расстояний достаточно вычислять линейную функцию сигнала:

$$\rho_j(x)^2 = \|x - W_j\|^2 = \|W_j\|^2 - 2\sum_{i=1}^{m} w_{ji}x_i + \|x\|^2$$

здесь $\|y\|$ - Евклидова длина вектора:

$$\|y\|^2 = \sum_i y_i^2$$

Последнее слагаемое $\|x\|^2$ одинаково для всех нейронов, поэтому для нахождения ближайшей точки оно не нужно. Задача сводится к поиску номера наибольшего из значений линейных функций:

$$j_{\max} = \arg\max_j \left\{ \sum_{i=1}^{m} w_{ji}x_i - \frac{1}{2}\|W_j\|^2 \right\}.$$

Таким образом, координаты точки $W_j = (w_{j1}, ...w_{jm})$ совпадают с весами линейного нейрона слоя Кохонена (при этом значение порогового коэффициента $w_{j0} = -\|W_j\|^2/2$).

Если заданы точки $W_j = (w_{j1}, ...w_{jm})$, то m-мерное пространство разбивается на соответствующие многогранники Вороного-Дирихле V_j: многогранник V_j состоит из точек, которые ближе к W_j, чем к другим W_k ($k \neq j$)

Задача векторного квантования с k кодовыми векторами W_j для заданной совокупности входных векторов S ставится как задача минимизации искажения при кодировании, то есть при замещении каждого вектора из S соответствующим кодовым вектором. В базовом варианте сетей Кохонена используется метод наименьших квадратов и искажение D вычисляется по формуле:

$$D = \sum_{j=1}^{k} \sum_{x \in K_j} \|x - W_j\|^2,$$

где K_j состоит из тех точек $x \in S$, которые ближе к W_j, чем к другим W_l ($l \neq j$). Другими словами, K_j состоит из тех точек $x \in S$, которые кодируются кодовым вектором W_j.

Если совокупность S задана и хранится в памяти, то стандартным выбором в обучении соответствующей сети Кохонена является метод K-средних. Это метод расщепления:

при данном выборе кодовых векторов (они же весовые векторы сети) W_j минимизацией D находим множества K_j — они состоит из тех точек $x \in S$, которые ближе к W_j, чем к другим W_l;

при данном разбиении S на множества K_j минимизацией D находим оптимальные позиции кодовых векторов W_j — для оценки по методу наименьших квадратов это просто средние арифметические:

$$W_j = \frac{1}{|K_j|} \sum_{x \in K_j} x,$$

где $|K_j|$ — число элементов в K_j.

Далее итерируем. Этот метод расщепления сходится за конечное число шагов и даёт локальный минимум искажения.

Если же, например, совокупность S заранее не задана, или по каким-либо причинам не хранится в памяти, то широко используется онлайн метод. Векторы входных сигналов x обрабатываются по одному, для каждого из них находится ближайший кодовый вектор («победитель», который «забирает всё») $W_{j(x)}$. После этого данный кодовый вектор пересчитывается по формуле

$$W_{j(x)}^{\text{new}} = W_{j(x)}^{\text{old}}(1 - \theta) + x\theta,$$

где $\theta \in (0, 1)$ — шаг обучения. Остальные кодовые векторы на этом шаге не изменяются.

Для обеспечения стабильности используется онлайн метод с затухающей скоростью обучения: если T — количество шагов обучения, то полагают $\theta = \theta(T)$. Функцию $\theta(T) > 0$ выбирают таким образом, чтобы $\theta(T) \to 0$ монотонно при $T \to \infty$ и чтобы ряд $\sum_{T=1}^{\infty} \theta(T)$ расходился, например, $\theta(T) = \theta_0/T$.

Векторное квантование является намного более общей операцией, чем кластеризация, поскольку кластеры должны быть разделены между собой, тогда как совокупности K_j для разных кодовых векторов W_j не обязательно представляют собой раздельные кластеры. С другой стороны, при наличии разделяющихся кластеров векторное квантование может находить их и по-разному кодировать.

Например: Решается задача классификации. Число классов может быть любым. Изложим алгоритм для двух классов, $\mathbf{A}$ и $\mathbf{B}$. Исходно для обучения системы поступают данные, класс которых известен. Задача: найти для класса $\mathbf{A}$ некоторое количество $k_{\mathbf{A}}$ кодовых векторов $W_j^{\mathbf{A}}$, а для класса $\mathbf{B}$ некоторое (возможно другое) количество $k_{\mathbf{B}}$ кодовых векторов $W_l^{\mathbf{B}}$ таким образом, чтобы итоговая сеть Кохонена с $k_{\mathbf{A}} + k_{\mathbf{B}}$ кодовыми векторами $W_j^{\mathbf{A}}, W_l^{\mathbf{B}}$ (объединяем оба семейства) осуществляла классификацию по следующему решающему правилу:

если для вектора входных сигналов x ближайший кодовый вектор («победитель», который в слое Кохонена «забирает всё») принадлежит семейству $\{W_j^{\mathbf{A}}\}$, то x принадлежит классу $\mathbf{A}$; если же ближайший к x кодовый вектор принадлежит семейству $\{W_l^{\mathbf{B}}\}$, то x принадлежит классу $\mathbf{B}$.

С каждым кодовым вектором объединённого семейства $\{W_j^{\mathbf{A}}\} \cup \{W_l^{\mathbf{B}}\}$ связан многогранник Вороного-Дирихле. Обозначим эти многогранники $V_j^{\mathbf{A}}, V_l^{\mathbf{B}}$ соответственно. Класс $\mathbf{A}$ в пространстве сигналов,

согласно решающему правилу, соответствует объединению $\cup_j V_j^{\mathbf{A}}$, а класс $\mathbf{B}$ соответствует объединению $\cup_l V_l^{\mathbf{B}}$. Геометрия таких объединений многогранников может быть весьма сложной (см. Рис с примером возможного разбиения на классы).

Рис 8. Разделения пространство Вороного-Дирихле на два класса

Правила обучения сети онлайн строится на основе базового правила обучения сети векторного квантования. Пусть на вход системы подаётся вектор сигналов x, класс которого известен. Если он классифицируется системой правильно, то соответствующий x кодовый вектор W слегка сдвигается в сторону вектора сигнала («поощрение»)

$$W^{\text{new}} = W^{\text{old}}(1-\theta) + x\theta,$$

Если же x классифицируется неправильно, то соответствующий x кодовый вектор W слегка сдвигается в противоположную сторону от сигнала («наказание»)

$$W^{\text{new}} = W^{\text{old}}(1+\theta) - x\theta,$$

где $\theta \in (0,1)$ — шаг обучения. Для обеспечения стабильности используется онлайн метод с затухающей скоростью обучения. Возможно также использование разных шагов для «поощрения» правильного решения и для «наказания» неправильного.

3. Задачи для реализации метода

Задачей данной работы является создание методов алгоритмов оценки надежности телекоммуникационных сетей с использованием нейросетей. Для ее выполнения была выбрана нейронная сеть Кохенена. Для решения задачи::

1. Работа с данными

– Составить базу данных из примеров, характерных для данной задачи

– Разбить всю совокупность данных на два множества: обучающее и тестовое (возможно разбиение на 3 множества: обучающее, тестовое и подтверждающее).

2. Предварительная обработка

– Выбрать систему признаков, характерных для данной задачи, и преобразовать данные соответствующим образом для подачи на вход сети (нормировка, стандартизация и т.д.). В результате желательно получить линейно отделяемое пространство множества образцов.

– Выбрать систему кодирования выходных значений .

3. Конструирование, обучение и оценка качества сети

– Выбрать топологию сети: количество слоев, число нейронов в слоях и т.д.

– Выбрать функцию активации нейронов (например "сигмоида")

– Выбрать алгоритм обучения сети

– Оценить качество работы сети на основе подтверждающего множества или другому критерию, оптимизировать архитектуру (уменьшение весов, прореживание пространства признаков)

– Остановится на варианте сети, который обеспечивает наилучшую способность к обобщению и оценить качество работы по тестовому множеству

4. Использование и диагностика

- Выяснить степень влияния различных факторов на принимаемое решение (эвристический подход).
- Убедится, что сеть дает требуемую точность классификации (число неправильно распознанных примеров мало)
- При необходимости вернутся на этап 2, изменив способ представления образцов или изменив базу данных.
- Практически использовать сеть для решения задачи.

Для тестирования созданного класса будет создано Windows-приложение с использованием визуальных компонентов. Тестирующая программа должна осуществлять ввод текстовой информации из файла и с клавиатуры, а также ввод параметров нейросети: количество эпох обучения, количество нейронов скрытого слоя, момент и коэффициент обучения.

4. Формирование входных данных для нейронной сети.

Для понимания методов анализа и синтеза необходимо определить факторы, влияющие на надежность СС. Анализ процессов функционирования сетей связи позволяет отнести к числу таких факторов следующие потоки:

- отказов и восстановлений технических средств;
- заявок на использование СС (загруженность);
- естественных помех;
- искусственных помех;
- разрушающих искусственных воздействий;
- ошибок программного обеспечения СС;
- отказов, вызванных деятельностью человека;
- отказов, вызванных природными явлениями.

Для оценки значимости той или иной группы факторов будем использовать результаты статистического анализа интенсивности и

последствий отказов, вызванных этими факторами (см. табл. 2.1), на примере высокоавтоматизированной широкомасштабной распределенной СС США PSTN (Public Switched Telephone Network).

Таблица 1.

Распределение отказов в сети PSTN.

Группа факторов	Доля от общего количества отказов, %	Потери пользователь-ского времени, %
Отказы технических средств	19	7
Перегрузки сети	6	44
Ошибки ПО	14	2
Ошибки персонала	25	14
Вандализм	1	1
Непреднамеренная разрушительная деятельность людей	24	14
Природные явления	11	18

Характер отказов технических средств СС и обычных радиоэлектронных систем идентичен. Влияние постепенных отказов, вызванных воздействием таких факторов, как нестабильность питания, тепловые воздействия, влажность, перепады давления, запыленность и агрессивность среды, компенсируется конструктивными методами при создании устройств и использованием специальных защитных устройств (стабилизаторы, термостаты, экраны). Таким образом, при оценке показателей надежности СС следует учитывать только потоки внезапных отказов. Среди внезапных отказов выделяют устойчивые отказы и сбои, то есть кратковременные самоустраняющиеся отказы.

Для учета характеристик потока сбоев элемента СС достаточно использовать интенсивность этого потока, а для учета потока устойчивых

отказов и потока восстановлений следует использовать коэффициент готовности и интенсивность отказов либо среднее время наработки на отказ и среднее время восстановления технических средств СС.

Отметим, что пользователю СС безразлично, чем вызван отказ в обслуживании его требования на установление связи - отказом элементов СС или их занятостью обслуживанием других заявок. Современные методы анализа СС позволяют учитывать занятость элементов СС при оценке показателей ее надежности.

Чаще всего нейтрализация влияния потока естественных помех достигается путем введения временной или информационной избыточности, либо применением помехоустойчивого кодирования, либо многократной передачей сообщения (что можно рассматривать так же, как применение своеобразных помехоустойчивых кодов). Влияние этого фактора учитывается путем соответствующего увеличения длительности сеанса связи с последующим использованием этого значения при расчете вероятности поддержания связи.

Характерные особенности воздействия потока искусственных помех позволяют рассматривать искусственную помеху, как сбой элемента, и учитывать влияние этого фактора на надежностные характеристики СС аналогично способу учета сбоев технических средств.

Характерными для воздействия потоков искусственных разрушающих воздействий являются: полная потеря работоспособности элемента СС, подверженного этому воздействию; возможность одновременного повреждения нескольких элементов; устойчивый характер повреждения, то есть отсутствие самовосстановления элемента после окончания воздействия; восстановление элемента после такого воздействия возможно только путем его ремонта или замены. Эти особенности аналогичны условиям возникновения и устранения устойчивых отказов за исключением высокой кратности нарушений

работоспособности элементов, большего времени восстановления и зависимости отказов элементов друг от друга.

Природа ошибок программного обеспечения СС существенно отличается от природы отказов технических средств СС. На количество и качество ошибок программы в основном влияют субъективные факторы. Поэтому отказ ПО - это событие, заключающееся в проявлении, при определенных последовательностях входных данных, тех ошибок, которые были допущены при разработке ПО и не выявлены при его тестировании. В то же время отказ технических средств - это событие, заключающееся в изменении его параметров и приводящее к потере работоспособности. Второе существенное отличие состоит в том, что программа не стареет, то есть с течением времени количество ошибок в программе не увеличивается, а может только уменьшаться в результате их обнаружения и устранения. Другие отличия: если ошибка программы исключена, то она больше в ней не появится; предварительный анализ влияния ошибки практически невозможен из-за большого числа способов влияния ошибки на поведение программы. Количественным показателем корректности программы является число ошибок, оставшихся в программе после завершения отладки и тестирования. По аналогии с количественными показателями безотказности технических устройств используют также вероятность безотказной работы программы на интервале времени и интенсивность потока ошибок.

Отказы, вызванные деятельностью человека, можно разделить на отказы, вызванные деятельностью обслуживающего персонала, и отказы, вызванные деятельностью других людей.

Первая группа отказов обусловлена ошибками персонала при эксплуатации СС. Их характер и интенсивность существенно зависят от уровня автоматизации процессов управления СС. Применительно к СС PSTN отказы этой группы вызываются следующими ошибками: эксплуатации и ремонта кабелей, систем питания, технических устройств,

контроля питания, использования не тех версий программного обеспечения; некорректной инсталляции и определения конфигурации программного обеспечения (исключая ошибки, вносимые в тексты программ); неверного ввода даты. Вторая группа отказов обусловлена преднамеренной (вандализм) и непреднамеренной разрушительной по отношению к СС деятельностью людей, не являющихся персоналом СС, которая проявляется в нарушении целостности кабелей и другого оборудования вследствие катастрофических событий (в основном автомобильных аварий).

К отказам, вызванным природными явлениями, относятся повреждения кабелей, систем питания и других устройств животными, а также молнией, ураганами, землетрясениями и наводнениями.

Выходными классами мы можем условно разделить на 5 или более дискретных класса для метода «Победитель забирает все».

Все входные параметры приводятся к интервалу от 0 . . 1 в зависимости от пределов данного фактора.

5. Конструирование, обучение и оценка качества сети.

Искусственный нейрон является элементарным функциональным модулем, из множества которых строятся ИНС. Он представляет собой, в нашем понимании, отдельно взятую личность сообщества, однако, лишь в смысле осуществляемых ею преобразований, а не способа функционирования. Взаимодействие отдельных членов представляет собой сетевое сообщество.

Функционалом отдельного члена сообщества (формального нейрона) можно считать :

1. Накопление данных (адаптивный сумматор).
2. Анализ (функция активации).
3. Передача опыта другим членам сообщества (точка ветвления).

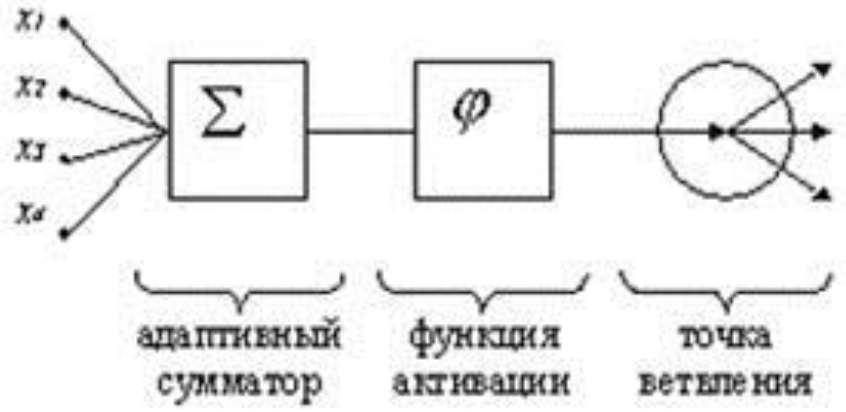


Рис. 9. Формальный нейрон

Большое распространение получили слои Кохонена, построенные следующим образом: каждому (jму) нейрону сопоставляется точка $W_j = (w_{j1}, ...w_{jm})$ в m-мерном пространстве (пространстве сигналов). Для входного вектора $x = (x_1, ...x_m)$вычисляются его евклидовы расстояния $\rho_j(x)$ до точек W_j и «ближайший получает всё» — тот нейрон, для которого это расстояние минимально, выдаёт единицу, остальные — нули. Следует заметить, что для сравнения расстояний достаточно вычислять линейную функцию сигнала:

$$\rho_j(x)^2 = \|x - W_j\|^2 = \|W_j\|^2 - 2\sum_{i=1}^{m} w_{ji}x_i + \|x\|^2$$

здесь $\|y\|$ - Евклидова длина вектора:

$$\|y\|^2 = \sum_i y_i^2$$

Последнее слагаемое $\|x\|^2$ одинаково для всех нейронов, поэтому для нахождения ближайшей точки оно не нужно. Задача сводится к поиску номера наибольшего из значений линейных функций:

$$j_{\max} = \arg\max_j \left\{ \sum_{i=1}^{m} w_{ji}x_i - \frac{1}{2}\|W_j\|^2 \right\}.$$

Таким образом, координаты точки $W_j = (w_{j1}, ...w_{jm})$ совпадают с весами линейного нейрона слоя Кохонена (при этом значение порогового коэффициента $w_{j0} = -\|W_j\|^2/2$).

Если заданы точки $W_j = (w_{j1}, ...w_{jm})$, то m-мерное пространство разбивается на соответствующие многогранники Вороного-Дирихле V_j: многогранник V_j состоит из точек, которые ближе к W_j, чем к другим W_k ($k \neq j$)

Рис 10. Обученные нейроны в 3х мерном пространстве признаков.

Задача векторного квантования с k кодовыми векторами W_j для заданной совокупности входных векторов S ставится как задача минимизации искажения при кодировании, то есть при замещении каждого вектора из S соответствующим кодовым вектором. В базовом варианте сетей Кохонена используется метод наименьших квадратов и искажение D вычисляется по формуле:

$$D = \sum_{j=1}^{k} \sum_{x \in K_j} \|x - W_j\|^2,$$

где K_j состоит из тех точек $x \in S$, которые ближе к W_j, чем к другим W_l ($l \neq j$). Другими словами, K_j состоит из тех точек $x \in S$, которые кодируются кодовым вектором W_j.

Количество внутренних нейронов можно будет задать в ручную, на каждое измерения. Нейронная сетка распределяется равномерно по измерениям и меняется входе обучения. Чем больше и плотнее расположены внутренний слой нейронной сети, тем точнее выводимый результат, но тем медленнее работа сети.

Выводы по главе II

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации.

Глава III. Реализация алгоритма и программы оценки надежности телекоммуникационных сетей с помощью нейронных сетей.

В данной главе описаны разработанные методы и приемы оптимизации работы и обеспечения стабильности системы, а также структура разрабатываемой нейронной сети. Описывается программная реализация разработанных методов и приемов, описывается методология разработки программного обеспечения нейронной сети. Проводится анализ полученных результатов. Подводятся итоги исследований.

В рамках исследования была написана программа реализующая алгоритмы и методы оценки надежности, на основе нейронных сетей.

Для реализации было выполнено:

1. Изучение существующих методов и программных средств для решения данной задачи.
2. Сбор исходных данных для обучения и тестирования.
   2.1. Сбор данных.
   2.2. Нормирование каждого показателя.
   2.3. Разделения на обучающие и тестовые данные.
3. Создание классов и программы нейронной сети.
4. Тестирование и анализ результатов программы.

1. Алгоритм обучения сети.

Суть обучения нейронной сети Кохонена заключается в такой подстройке весов, при которой близкие входные векторы будут активировать один и тот же нейрон Кохонена.

Если совокупность S задана и хранится в памяти, то стандартным выбором в обучении соответствующей сети Кохонена является метод K-средних. Это метод расщепления:

При данном выборе кодовых векторов (они же весовые векторы сети) W_j минимизацией D находим множества K_j — они состоит из тех точек $x \in S$, которые ближе к W_j, чем к другим W_l;

при данном разбиении S на множества K_j минимизацией D находим оптимальные позиции кодовых векторов W_j — для оценки по методу наименьших квадратов это просто средние арифметические:

$$W_j = \frac{1}{|K_j|} \sum_{x \in K_j} x,$$

где $|K_j|$ — число элементов в K_j.

Далее итерируем. Этот метод расщепления сходится за конечное число шагов и даёт локальный минимум искажения.

k-means (метод k-средних) — наиболее популярный метод кластеризации. Был изобретён в 1950-х годах математиком Гуго Штейнгаузом и почти одновременно Стюартом Ллойдом. Особую популярность приобрёл после работы Маккуина.

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

где k — число кластеров, S_i — полученные кластеры, $i = 1, 2, \ldots, k$ и μ_i — центры масс векторов $x_j \in S_i$.

По аналогии с методом главных компонент центры кластеров называются также главными точками, а сам метод называется методом главных точек и включается в общую теорию главных объектов, обеспечивающих наилучшую аппроксимацию данных.

Алгоритм представляет собой версию EM-алгоритма, применяемого также для разделения смеси гауссиан. Он разбивает множество элементов векторного пространствана заранее известное число кластеров k.

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное уклонение V уменьшается, поэтому зацикливание невозможно.

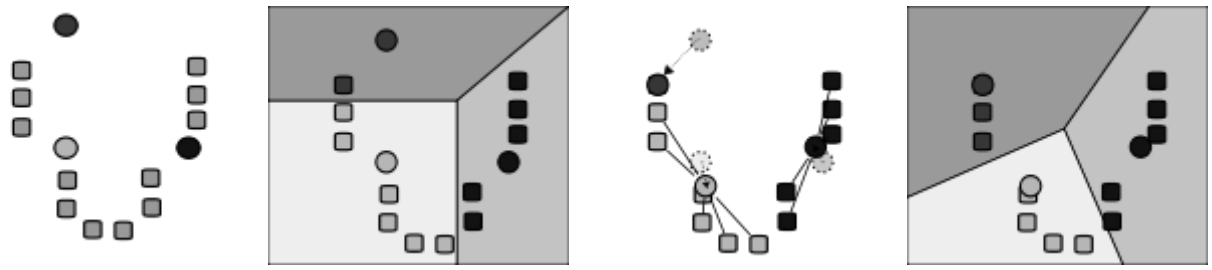


Рис 11. Обучение нейронной сети методом К-средних.

Но этот метод не гарантируется достижение глобального минимума суммарного квадратичного отклонения *V*, а только одного из локальных минимумов.

Результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен.

Число кластеров надо знать заранее.

ОВК (обучающее векторное квантование).

Алгоритм квантования обучающего вектора (LVQ) был придуман тем же Туэво Кохоненом, который изобрел самоорганизующиеся карты признаков (которые в нашем пакете называются сетями Кохонена).

Квантование обучающего вектора представляет собой преобразованный метод обучения Кохонена. Стандартный алгоритм Кохонена с помощью итерационной процедуры подбирает положение

векторов образцов, хранящихся в радиальном слое сети Кохонена, учитывая только положения существующих векторов и обучающих -- данных. Этот алгоритм стремится поместить векторы образцов в центры кластеров обучающих данных. При этом не принимаются во внимание метки классов обучающих наблюдений. Для достижения наилучшего качества классификации, желательно, чтобы векторы образцов формировались на основе имеющихся классов, т.е. чтобы они отражали естественные группы в каждом отдельном классе. Скорее всего, образец, расположенный на границе, на одинаковом расстоянии от наблюдений из двух классов, будет бесполезен для распознавания этих классов. В то же время крайне важную роль может играть образец, расположенный внутри, достаточно далеко от границ.

Существует несколько вариантов метода квантования обучающего вектора, и три из них реализованы в программе STATISTICA Нейронные Сети. Основная версия, LVQ1, очень похожа на алгоритм обучения Кохонена. Выбирается образец, ближайший к обучающему наблюдению, и его положение корректируется. Однако если в алгоритме Кохонена образец просто смещается в сторону обучающего наблюдения, в методе LVQ1 проверяется совпадение меток классов образца и обучающего наблюдения. Если такое совпадение имеет место, образец смещается к обучающему наблюдению, а если совпадения нет - смещение происходит в противоположную сторону. Еще более гибкие алгоритмы, LVQ2.1 и LVQ3, учитывают и дополнительную информацию. Они определяют два ближайших к обучающему наблюдению образца. Если один из этих образцов имеет правильное значение класса, а другой неправильное, то правильный образец смещается к обучающему наблюдению, а неправильный - от него. Метод LVQ3 смещает в сторону обучающего наблюдения оба образца, если они оба имеют правильный класс. Основной смысл обоих методов, LVQ2.1 и LVQ3, заключается в постепенном

перемещении образцов в те места, где существует опасность ошибочной классификации.

Технические замечания. Для корректировки весов используются следующие правила:

$$\omega_t = \omega_{t-1} + \eta_t(x - \omega_{t-1})$$

если образец и обучающее наблюдение принадлежат одному классу;

$$\omega_t = \omega_{t-1} - \eta_t(x + \omega_{t-1})$$

если не принадлежат.

x - обучающее наблюдение, h_t - скорость обучения.

В методе LVQ2.1 два ближайших образца корректируются только в том случае, если один из них принадлежит правильному классу, а другой нет, и оба они находятся на "примерно одинаковом" расстоянии от обучающего наблюдения. Для определения понятия "примерно одинаковый" используется специальный параметр, e, и следующие формулы:

$$\min\left(\frac{d_1}{d}, \frac{d_2}{d_1}\right) > 1 - \varepsilon$$

$$\max\left(\frac{d_1}{d}, \frac{d_2}{d_1}\right) < 1 + \varepsilon$$

В методе LVQ3 используется аналогичная формула для того, чтобы определить, расположены ли два ближайших образца на "примерно одинаковом" расстоянии от обучающего наблюдения:

$$\min\left(\frac{d_1}{d}, \frac{d_2}{d_1}\right) > (1 - \varepsilon)(1 + \varepsilon)$$

Кроме того, в методе LVQ3 оба ближайших образца смещаются в сторону обучающего наблюдения, если они принадлежат одному классу. При этом стандартный коэффициент скорости обучения для данной эпохи умножается на коэффициент b.

2. Архитектура нейронной сети Кохонена.

Структура нейронной сети Кохонена состоит из:

- Входного слоя
- Слоя кодовых векторов (нейронов)
- Слоя классов.

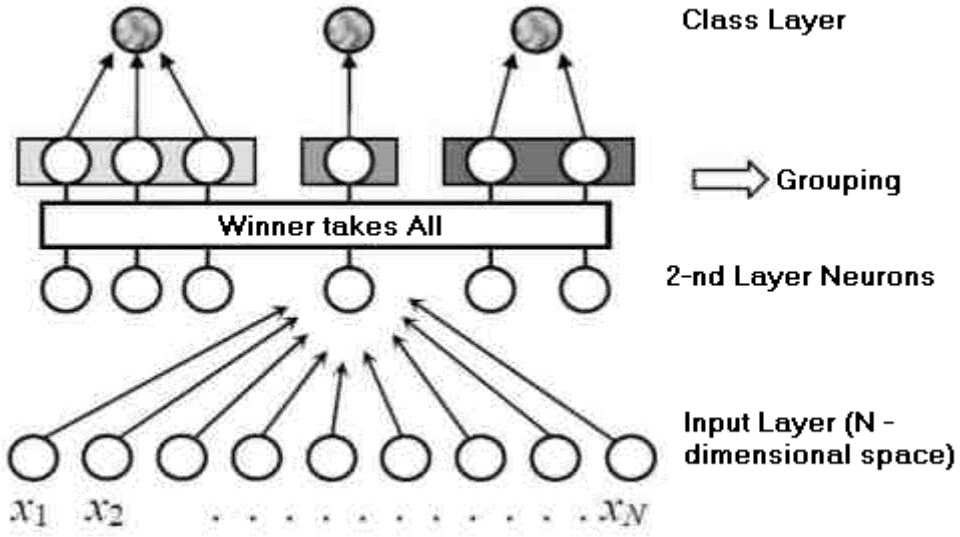


Рис.12. Структура нейронной сети Кохонена.

Входной слой определяется количеством факторов влияющих на конечный результат. Все данные должны быть в интервале 0..1.

В нашем случае входных параметров 8:

- статистика отказов и восстановлений технических средств;
- заявок на использование СС или загруженность сети;
- защищенность от естественных помех;
- защищенность от искусственных помех;
- защищенность от разрушающих искусственных воздействий;
- статистика ошибок программного обеспечения СС или уровень надежности ПО;
- защищенность от отказов, вызванных деятельностью человека или уровень автоматизации;
- защищенность от отказов, вызванных природными явлениями.

Слой кодовых векторов можно задать предварительно, перед обучением и чем больше их тем меньше ошибок, но медленнее работать будет.

Слой классов содержат список кодовых векторов относящиеся к этому классу. В нашем случае мы задаем 5 классов надежности сети: A, B, C, D, E.

Соответственно A класс с самым высоким уровнем надежности, а E класс самым низким уровнем надежности.

3. Сбор исходных данных для обучения и тестирования

Исходными данными могут послужить статистические данные или знания эксперта.

Надежность системы пропорционален входным параметрам нейронной сети. Но некоторых состояниях, с несколькими низкими параметрами, система может сохранять свою работоспособность.

Исходными данными возьмём точки в n-мерном пространстве. Максимальная длина вектора точки может быть $8^{1/2}$. Эту длину разделим на 5 отрезок и определим границы классов:

- E класс – от 0 до $8^{1/2}/5$
- D класс – от 0 до $2*8^{1/2}/5$
- C класс – от 0 до $3*8^{1/2}/5$
- B класс – от 0 до $4*8^{1/2}/5$
- A класс – от 0 до $8^{1/2}$

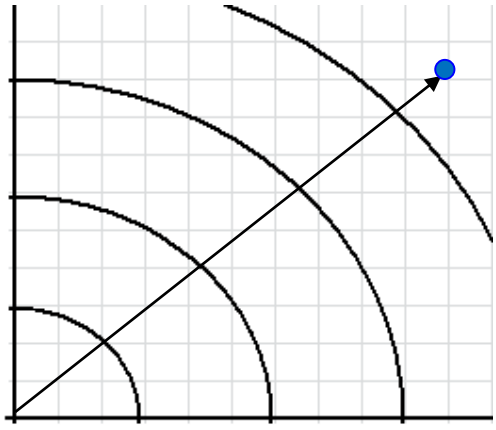


Рис. 13. разделения пространства на 5 классов.

У нас 8 входных параметров:

- статистика отказов и восстановлений технических средств;
- заявок на использование СС или загруженность сети;
- защищенность от естественных помех;
- защищенность от искусственных помех;
- защищенность от разрушающих искусственных воздействий;
- статистика ошибок программного обеспечения СС или уровень надежности ПО;
- защищенность от отказов, вызванных деятельностью человека или уровень автоматизации;
- защищенность от отказов, вызванных природными явлениями.

Характер отказов технических средств системы и обычных радиоэлектронных систем идентичен. Влияние постепенных отказов, вызванных воздействием таких факторов, как нестабильность питания, тепловые воздействия, влажность, перепады давления, запыленность и агрессивность среды, компенсируется конструктивными методами при создании устройств и использованием специальных защитных устройств (стабилизаторы, термостаты, экраны). Таким образом, при оценке показателей надежности системы следует учитывать только потоки внезапных отказов. Среди внезапных отказов выделяют устойчивые отказы и сбои, то есть кратковременные самоустраняющиеся отказы.

Для учета характеристик потока сбоев элемента системы достаточно использовать интенсивность этого потока, а для учета потока устойчивых отказов и потока восстановлений следует использовать коэффициент готовности и интенсивность отказов либо среднее время наработки на отказ и среднее время восстановления технических средств системы. Полученный показатель делится на максимальный показатель и отнимается от 1.

Отметим, что пользователю системы безразлично, чем вызван отказ в обслуживании его требования на установление связи - отказом элементов

системы или их занятостью обслуживанием других заявок. Современные методы анализа системы позволяют учитывать занятость элементов системы при оценке показателей ее надежности. Для оценки возможностей сети надо оценить запасную ширину канала и бесперебойность обслуживания. Для этого среднюю загруженность канала умножаем на 1,5 и делим на пропускную способность канала. Результат отнимаем от 1.

Чаще всего нейтрализация влияния потока естественных помех достигается путем введения временной или информационной избыточности, либо применением помехоустойчивого кодирования, либо многократной передачей сообщения (что можно рассматривать так же, как применение своеобразных помехоустойчивых кодов). Влияние этого фактора учитывается путем соответствующего увеличения длительности сеанса связи с последующим использованием этого значения при расчете вероятности поддержания связи.

Характерные особенности воздействия потока искусственных помех позволяют рассматривать искусственную помеху, как сбой элемента, и учитывать влияние этого фактора на надежностные характеристики системы аналогично способу учета сбоев технических средств.

Характерными для воздействия потоков искусственных разрушающих воздействий являются: полная потеря работоспособности элемента системы, подверженного этому воздействию; возможность одновременного повреждения нескольких элементов; устойчивый характер повреждения, то есть отсутствие самовосстановления элемента после окончания воздействия; восстановление элемента после такого воздействия возможно только путем его ремонта или замены. Эти особенности аналогичны условиям возникновения и устранения устойчивых отказов за исключением высокой кратности нарушений работоспособности элементов, большего времени восстановления и зависимости отказов элементов друг от друга.

Природа ошибок программного обеспечения системы существенно отличается от природы отказов технических средств системы. На количество и качество ошибок программы в основном влияют субъективные факторы. Поэтому отказ ПО - это событие, заключающееся в проявлении, при определенных последовательностях входных данных, тех ошибок, которые были допущены при разработке ПО и не выявлены при его тестировании. В то же время отказ технических средств - это событие, заключающееся в изменении его параметров и приводящее к потере работоспособности. Второе существенное отличие состоит в том, что программа не стареет, то есть с течением времени количество ошибок в программе не увеличивается, а может только уменьшаться в результате их обнаружения и устранения. Другие отличия: если ошибка программы исключена, то она больше в ней не появится; предварительный анализ влияния ошибки практически невозможен из-за большого числа способов влияния ошибки на поведение программы. Количественным показателем корректности программы является число ошибок, оставшихся в программе после завершения отладки и тестирования. По аналогии с количественными показателями безотказности технических устройств используют также вероятность безотказной работы программы на интервале времени и интенсивность потока ошибок.

Отказы, вызванные деятельностью человека, можно разделить на отказы, вызванные деятельностью обслуживающего персонала, и отказы, вызванные деятельностью других людей.

Первая группа отказов обусловлена ошибками персонала при эксплуатации системы. Их характер и интенсивность существенно зависят от уровня автоматизации процессов управления системой. Применительно к системе PSTN отказы этой группы вызываются следующими ошибками: эксплуатации и ремонта кабелей, систем питания, технических устройств, контроля питания, использования не тех версий программного обеспечения; некорректной инсталляции и определения конфигурации

программного обеспечения (исключая ошибки, вносимые в тексты программ); неверного ввода даты. Вторая группа отказов обусловлена преднамеренной (вандализм) и непреднамеренной разрушительной по отношению к системе с деятельностью людей, не являющихся персоналом СС, которая проявляется в нарушении целостности кабелей и другого оборудования вследствие катастрофических событий (в основном автомобильных аварий).

К отказам, вызванным природными явлениями, относятся повреждения кабелей, систем питания и других устройств животными, а также молнией, ураганами, землетрясениями и наводнениями.

Все входные параметры приводятся к интервалу от 0 . . 1 в зависимости от пределов данного фактора. Вероятностные факторы вычисляются и учитывается вероятность работоспособности, то есть вероятность отказа отнимаем от 1.

4. Реализация алгоритма нейронной сети для оценки надежности телекоммуникационных сетей.

Алгоритм был реализован в виде двух классов:

- класса Neuron.
- класса Leyer.

Класс Neuron представляет модель нейрона Кохонена. Класс имеет следующие поля:

- private String clName;
- private int count;
- private double[] weights;

Также имеет следущие методы:

- public void addPoint(double[] point)
- public void setNeuron(double[] point)
- public void setClName(String clName)
- public void getClName()
- public double test(double[] point)

Класс Layer представляет модель нойронного слоя Кохонена и содержит в себе нейроны слоя и имеет поле:

- public Neuron[] layer;

Также имеет следующие методы:

- public int searchWithName(double[] point, String clName)
- public int search(double[] point)
- public void study()
- public ArrayList<String> test()
- public void setLayer(int points)

Ввод данных осушествляется с базы данных MySQL, из таблиц STUDY и TEST.

Структура базы данных выглядит следующим образом:

```
CREATE DATABASE `NEURON`;
USE `NEURON`;
CREATE TABLE
  `STUDY` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
  `par1` DOUBLE NOT NULL,
  `par2` DOUBLE NOT NULL,
  `par3` DOUBLE NOT NULL,
  `par4` DOUBLE NOT NULL,
  `par5` DOUBLE NOT NULL,
  `par6` DOUBLE NOT NULL,
  `par7` DOUBLE NOT NULL,
  `par8` DOUBLE NOT NULL,
        `class` CHAR(30) NOT NULL,
    PRIMARY KEY(`id`)
  )
CREATE TABLE
  `TEST` (
```

```
        `id` INT(11) NOT NULL AUTO_INCREMENT,
    `par1` DOUBLE NOT NULL,
    `par2` DOUBLE NOT NULL,
    `par3` DOUBLE NOT NULL,
    `par4` DOUBLE NOT NULL,
    `par5` DOUBLE NOT NULL,
    `par6` DOUBLE NOT NULL,
    `par7` DOUBLE NOT NULL,
    `par8` DOUBLE NOT NULL,
        `class` CHAR(30) NOT NULL,
      PRIMARY KEY(`id`)
    )
```

Описание класса Neuron

Класс Neuron имеет следующие поля:

- private String clName;
- private int count;
- private double[] weights;

Поле “clName” – представляет наименования класса. Используется для идентификации классов.

Поле “count” – представляет количество объектов обучения нейрона.

Поле “weights” – представляет весы нейрона для входного вектора. Длина массива соответсвует длине входного векора.

Класс Neuron имеет следущие методы:

- public void addPoint(double[] point)
- public void setNeuron(double[] point)
- public void setClName(String clName)
- public String getClName()
- public double test(double[] point)

Метод “addPoint” принимает в качестве параметра входной вектор для обучения и не возвращает никакое значение. Этому методу обрашается

для обучения нейрона. Принятый вектор смешает весы нейрона свою сторону по следуюшей формуле.

$$W = \frac{(W * count + X)}{count + 1}$$

Таким образом чем больше обучаются, тем меньше сдвигаются нейронные веса. Этот метод относится динамическому обучению нейрона.

Метод “setNeuron” принимает в качестве взодного параметра начальные веса нейрона и не возвращает никакое значение.

Метод “setClName” принимает в качестве входного параметра строку и никакой параметр не возвращает. Метод устанавливает название класса которому относится нейрон.

Метод “getClName” не принимает никакой параметр и возвращает строку. Метод возвращает название класса которому относится нейрон.

Метод “test” в качестве входного параметра принимает вектор входных параметров и возвращает число с плавующей запятой. Этот метод вычисляет растояние от весов нейрона до входного вектора.растояние вычисляется по следующей формуле:

$$\lambda = \sqrt{\sum_{i=1}^{n} (w_i - x_i)^2}$$

Описание класса Layer.

Класс Layer имеет следующие поля:

- public Neuron[] layer;

Поле “layer” представляет слой нейронов и состоит из множество нейронов относящиеся нескольким классам. Количесво нейронов в слою формируется во время работы программы параметрами пользователя.

Класс Layer имеет следующие методы:

- public int searchWithName(double[] point, String clName)
- public int search(double[] point)

- public void study()
- public ArrayList<String> test()
- public void setLayer(int points)

Метод “searchWithName” принимает в качестве параметра входной вектор и строку и возвращает целочисленное значение. Этот метод ишет самый близкий нейрон входному вектору относящийся к определенному классу и возвращает индекс нейрона в слою.

Метод “search” принимает в качестве параметра вектор объекта и возвращает целочисленное значение. Этот метод ишет самый близкий нейрон входному вектору и возвращает индекс нейрона в слою.

Метод “study” непринимает и невозвращает никакие параметры. Метод производит обучение нейронного слоя. Обучение производится поиском близкого нейрона и обучением его. Обучения производится из таблицы “STUDY” базы данных.

Метод “test” непринимает никакие параметры и возвращает лист строк. Метод проверяет обученнсть нейронного слоя на основе тестов хранящиеся в таблице базы данных “TEST”. Метод возвращает положительный или отрицательный результат каждого теста в листе.

Метод “setLayer” принимает в качестве параметра целое значение и не возвращает параметры. Метод на основе принятого параметра вычисляет количество нейронов и равномерно распределяет в пространстве. Принятый параметр определяет сколько нейронов приходится на каждое измерение. Например: Если ведем цыфру 3 то количество нейронов на каждое измерение приходится по 3, у нас 8 измерений получается $N = n^m = 3^8 = 6561$, то есть 6 561 нейронов в слое.

5. Результаты программы

Программа для реализации метода была написана на языке Java в среде NetBeans как консольное приложение.

NetBeans IDE — свободная интегрированная среда __разработки приложений (IDE) на языках программирования Java, JavaFX, Python, PHP, JavaScript, C, C++, Ада и ряда других.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведется независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

По качеству и возможностям последние версии NetBeans IDE не уступают лучшим коммерческим (платным) интегрированным средам разработки для языка Java, таким, как IntelliJ IDEA, поддерживая рефакторинг, профилирование, выделение синтаксических конструкций цветом, авто-дополнение набираемых конструкций на лету, множество предопределённых шаблонов кода и др.

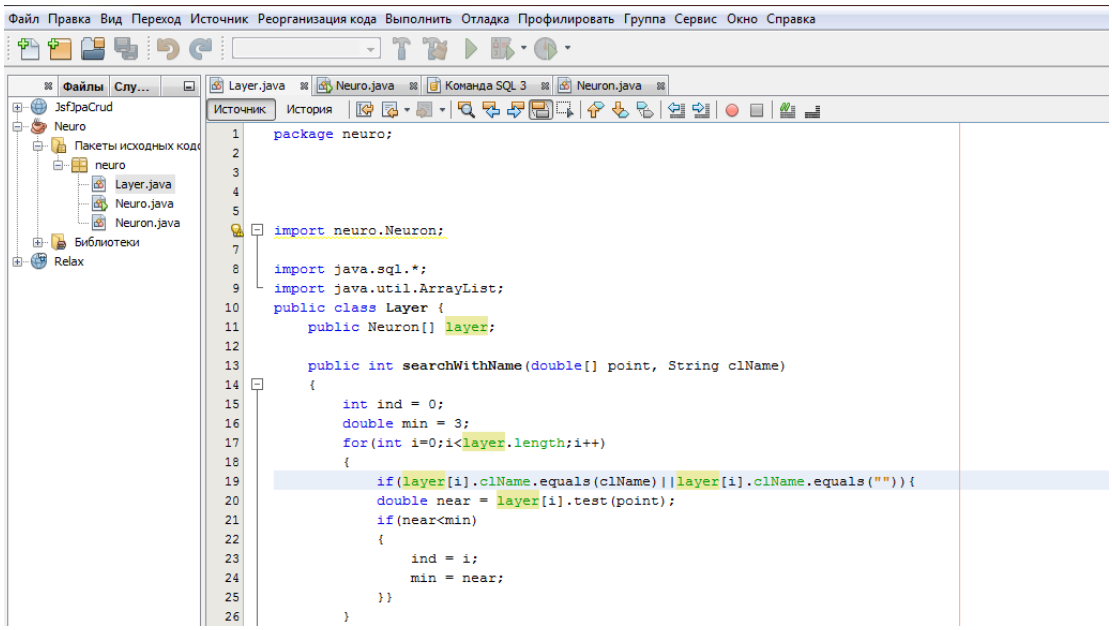
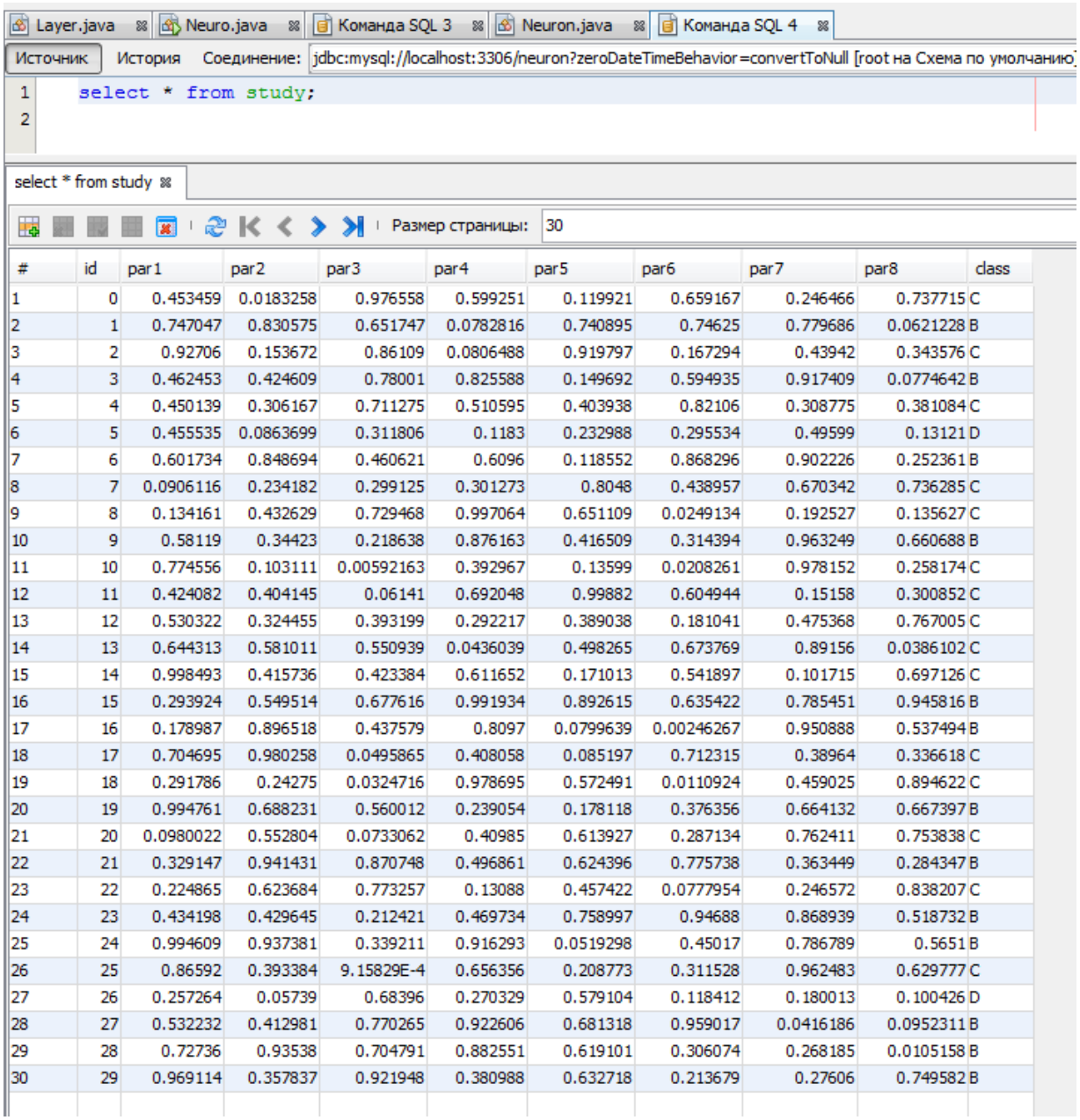


Рис. 14. Среда разработки NetBeans.

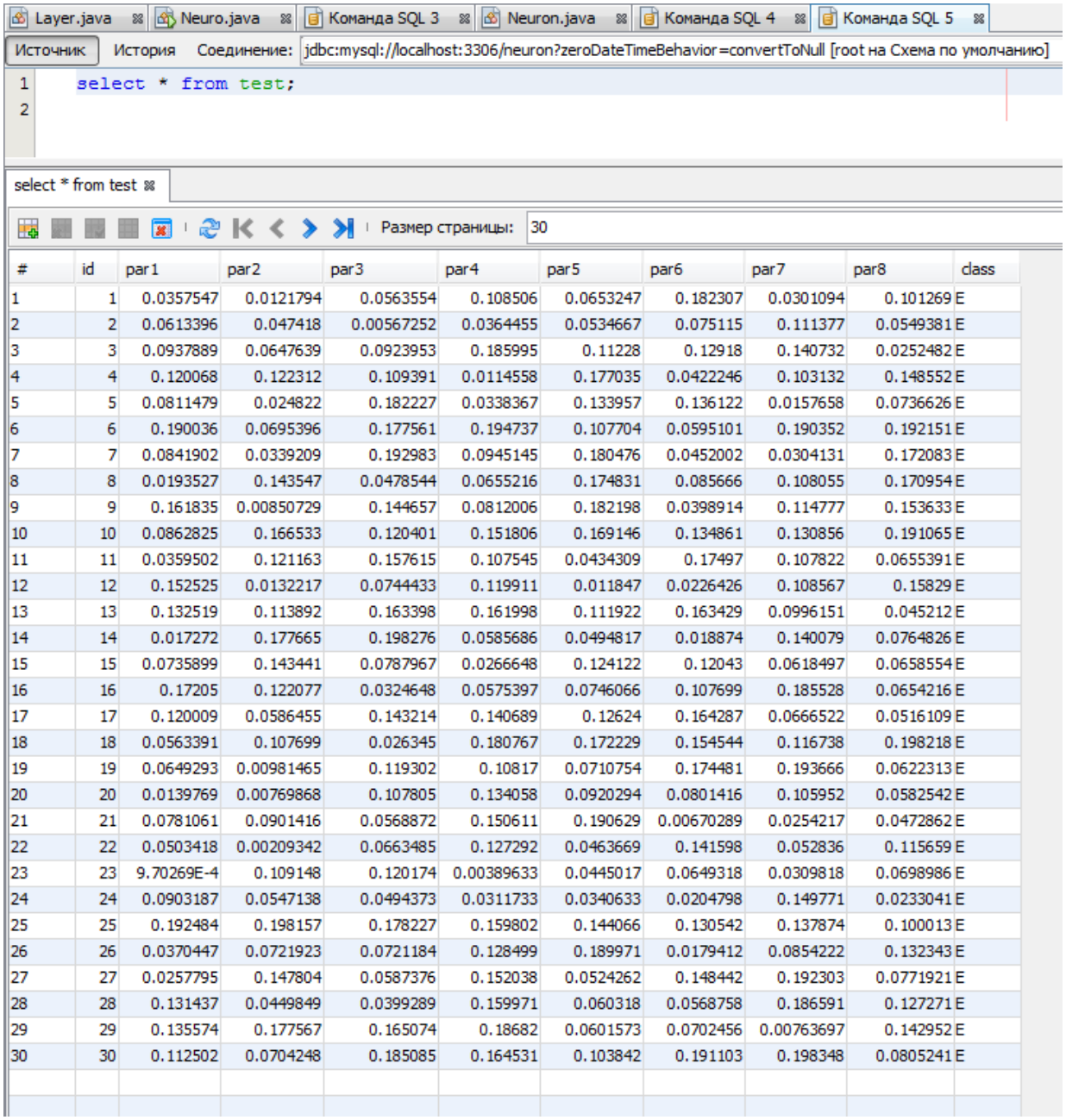
База данных для обучения и тестирования было представлено в база данных MySQL.



#	id	par1	par2	par3	par4	par5	par6	par7	par8	class
1	0	0.453459	0.0183258	0.976558	0.599251	0.119921	0.659167	0.246466	0.737715	C
2	1	0.747047	0.830575	0.651747	0.0782816	0.740895	0.74625	0.779686	0.0621228	B
3	2	0.92706	0.153672	0.86109	0.0806488	0.919797	0.167294	0.43942	0.343576	C
4	3	0.462453	0.424609	0.78001	0.825588	0.149692	0.594935	0.917409	0.0774642	B
5	4	0.450139	0.306167	0.711275	0.510595	0.403938	0.82106	0.308775	0.381084	C
6	5	0.455535	0.0863699	0.311806	0.1183	0.232988	0.295534	0.49599	0.13121	D
7	6	0.601734	0.848694	0.460621	0.6096	0.118552	0.868296	0.902226	0.252361	B
8	7	0.0906116	0.234182	0.299125	0.301273	0.8048	0.438957	0.670342	0.736285	C
9	8	0.134161	0.432629	0.729468	0.997064	0.651109	0.0249134	0.192527	0.135627	C
10	9	0.58119	0.34423	0.218638	0.876163	0.416509	0.314394	0.963249	0.660688	B
11	10	0.774556	0.103111	0.00592163	0.392967	0.13599	0.0208261	0.978152	0.258174	C
12	11	0.424082	0.404145	0.06141	0.692048	0.99882	0.604944	0.15158	0.300852	C
13	12	0.530322	0.324455	0.393199	0.292217	0.389038	0.181041	0.475368	0.767005	C
14	13	0.644313	0.581011	0.550939	0.0436039	0.498265	0.673769	0.89156	0.0386102	C
15	14	0.998493	0.415736	0.423384	0.611652	0.171013	0.541897	0.101715	0.697126	C
16	15	0.293924	0.549514	0.677616	0.991934	0.892615	0.635422	0.785451	0.945816	B
17	16	0.178987	0.896518	0.437579	0.8097	0.0799639	0.00246267	0.950888	0.537494	B
18	17	0.704695	0.980258	0.0495865	0.408058	0.085197	0.712315	0.38964	0.336618	C
19	18	0.291786	0.24275	0.0324716	0.978695	0.572491	0.0110924	0.459025	0.894622	C
20	19	0.994761	0.688231	0.560012	0.239054	0.178118	0.376356	0.664132	0.667397	B
21	20	0.0980022	0.552804	0.0733062	0.40985	0.613927	0.287134	0.762411	0.753838	C
22	21	0.329147	0.941431	0.870748	0.496861	0.624396	0.775738	0.363449	0.284347	B
23	22	0.224865	0.623684	0.773257	0.13088	0.457422	0.0777954	0.246572	0.838207	C
24	23	0.434198	0.429645	0.212421	0.469734	0.758997	0.94688	0.868939	0.518732	B
25	24	0.994609	0.937381	0.339211	0.916293	0.0519298	0.45017	0.786789	0.5651	B
26	25	0.86592	0.393384	9.15829E-4	0.656356	0.208773	0.311528	0.962483	0.629777	C
27	26	0.257264	0.05739	0.68396	0.270329	0.579104	0.118412	0.180013	0.100426	D
28	27	0.532232	0.412981	0.770265	0.922606	0.681318	0.959017	0.0416186	0.0952311	B
29	28	0.72736	0.93538	0.704791	0.882551	0.619101	0.306074	0.268185	0.0105158	B
30	29	0.969114	0.357837	0.921948	0.380988	0.632718	0.213679	0.27606	0.749582	B

Рис. 15. Таблица STUDY базы данных NEURON.

Таблица STUDY имеет более 15000 записей для обучения нейронной сети и содержит входные параметры и классы которым они относятся. Для тестирования нейронный сеть создана таблица TEST. Который содержит более 730 тестов для тестирования системы.



Layer.java | Neuro.java | Команда SQL 3 | Neuron.java | Команда SQL 4 | Команда SQL 5

Источник | История | Соединение: jdbc:mysql://localhost:3306/neuron?zeroDateTimeBehavior=convertToNull [root на Схема по умолчанию]

```
1  select * from test;
2
```

select * from test

Размер страницы: 30

#	id	par1	par2	par3	par4	par5	par6	par7	par8	class
1	1	0.0357547	0.0121794	0.0563554	0.108506	0.0653247	0.182307	0.0301094	0.101269	E
2	2	0.0613396	0.047418	0.00567252	0.0364455	0.0534667	0.075115	0.111377	0.0549381	E
3	3	0.0937889	0.0647639	0.0923953	0.185995	0.11228	0.12918	0.140732	0.0252482	E
4	4	0.120068	0.122312	0.109391	0.0114558	0.177035	0.0422246	0.103132	0.148552	E
5	5	0.0811479	0.024822	0.182227	0.0338367	0.133957	0.136122	0.0157658	0.0736626	E
6	6	0.190036	0.0695396	0.177561	0.194737	0.107704	0.0595101	0.190352	0.192151	E
7	7	0.0841902	0.0339209	0.192983	0.0945145	0.180476	0.0452002	0.0304131	0.172083	E
8	8	0.0193527	0.143547	0.0478544	0.0655216	0.174831	0.085666	0.108055	0.170954	E
9	9	0.161835	0.00850729	0.144657	0.0812006	0.182198	0.0398914	0.114777	0.153633	E
10	10	0.0862825	0.166533	0.120401	0.151806	0.169146	0.134861	0.130856	0.191065	E
11	11	0.0359502	0.121163	0.157615	0.107545	0.0434309	0.17497	0.107822	0.0655391	E
12	12	0.152525	0.0132217	0.0744433	0.119911	0.011847	0.0226426	0.108567	0.15829	E
13	13	0.132519	0.113892	0.163398	0.161998	0.111922	0.163429	0.0996151	0.045212	E
14	14	0.017272	0.177665	0.198276	0.0585686	0.0494817	0.018874	0.140079	0.0764826	E
15	15	0.0735899	0.143441	0.0787967	0.0266648	0.124122	0.12043	0.0618497	0.0658554	E
16	16	0.17205	0.122077	0.0324648	0.0575397	0.0746066	0.107699	0.185528	0.0654216	E
17	17	0.120009	0.0586455	0.143214	0.140689	0.12624	0.164287	0.0666522	0.0516109	E
18	18	0.0563391	0.107699	0.026345	0.180767	0.172229	0.154544	0.116738	0.198218	E
19	19	0.0649293	0.00981465	0.119302	0.10817	0.0710754	0.174481	0.193666	0.0622313	E
20	20	0.0139769	0.00769868	0.107805	0.134058	0.0920294	0.0801416	0.105952	0.0582542	E
21	21	0.0781061	0.0901416	0.0568872	0.150611	0.190629	0.00670289	0.0254217	0.0472862	E
22	22	0.0503418	0.00209342	0.0663485	0.127292	0.0463669	0.141598	0.052836	0.115659	E
23	23	9.70269E-4	0.109148	0.120174	0.00389633	0.0445017	0.0649318	0.0309818	0.0698986	E
24	24	0.0903187	0.0547138	0.0494373	0.0311733	0.0340633	0.0204798	0.149771	0.0233041	E
25	25	0.192484	0.198157	0.178227	0.159802	0.144066	0.130542	0.137874	0.100013	E
26	26	0.0370447	0.0721923	0.0721184	0.128499	0.189971	0.0179412	0.0854222	0.132343	E
27	27	0.0257795	0.147804	0.0587376	0.152038	0.0524262	0.148442	0.192303	0.0771921	E
28	28	0.131437	0.0449849	0.0399289	0.159971	0.060318	0.0568758	0.186591	0.127271	E
29	29	0.135574	0.177567	0.165074	0.18682	0.0601573	0.0702456	0.00763697	0.142952	E
30	30	0.112502	0.0704248	0.185085	0.164531	0.103842	0.191103	0.198348	0.0805241	E

Рис 16. Таблица TEST базы данных NEURON.

Данные таблиц базы данных отличны друг от друга и на повторяются.

Программа тестировалась разными входными параметрами, при 2, 3, 4 разделений на измерения.

При 2х разделениях на измерения:

- Количество нейронов 256.
- Обучающие данных – 15007.
- Тестов – 731.

– Время работы – 1 с.

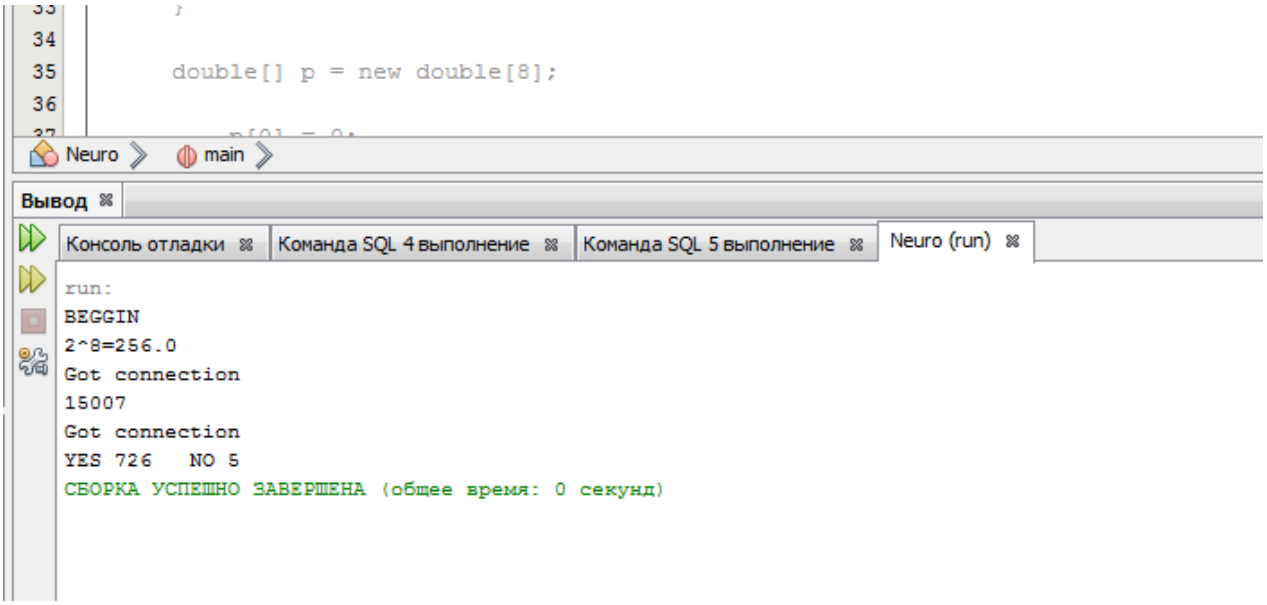


Рис. 17. Результаты работы программы при 2х разделениях на измерения

При 2х разделениях на измерения результат 726 из 731, то есть 99,31%.

При 3х разделениях на измерения:

– Количество нейронов 6561.

– Обучающие данных – 15007.

– Тестов – 731.

– Время работы – 4 с.

Рис. 18. Результаты работы программы при 3х разделениях на измерения

При 3х разделениях на измерения результат 730 из 731, то есть 99,86%.

При 4х разделениях на измерения:

– Количество нейронов 65536.
– Обучающие данных – 15007.
– Тестов – 731.
– Время работы – 44 с.

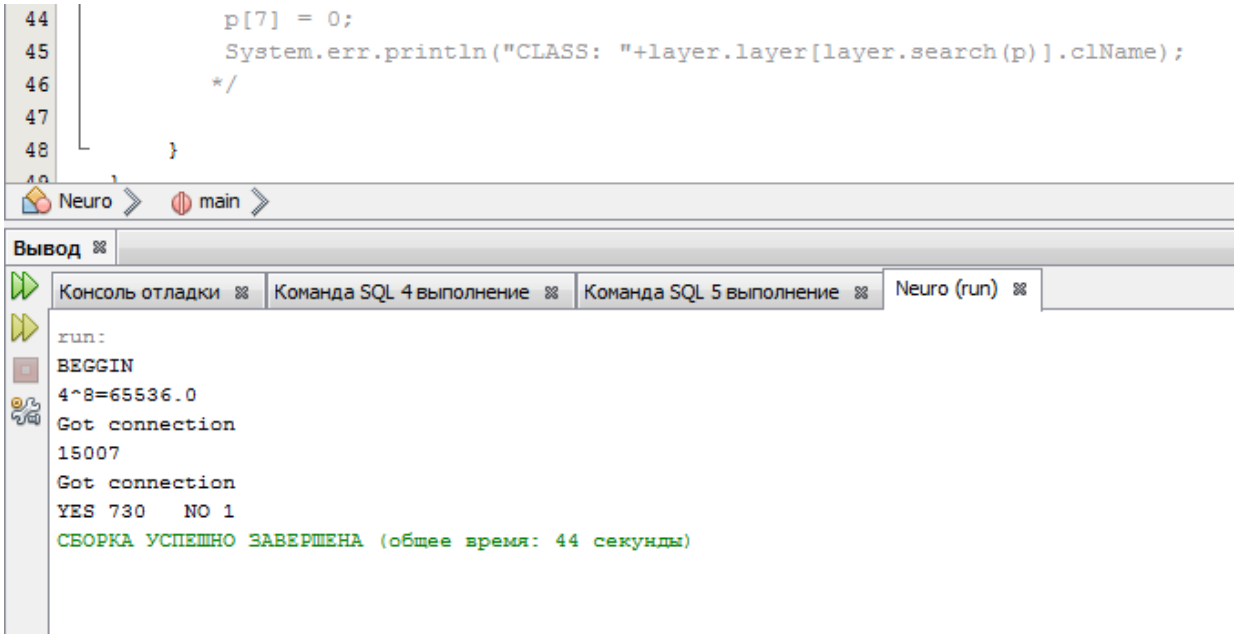


Рис. 19. Результаты работы программы при 4х разделениях на измерения
При 4х разделениях на измерения результат 730 из 731, то есть 99,86%.

Выводы по главе III

Оптимальным параметром является при 3х разделениях на измерения. При этом скорость работы и результативвность оптимальный.

Алгоритм можно усложнить и модифициоровать для получения более лучших результатов и ускорить обучение сети.

Даже в случае успешного, на первый взгляд, обучения сеть не всегда обучается именно тому, чего от неё хотел создатель. Известен случай, когда сеть обучалась распознаванию изображений танков по фотографиям, однако позднее выяснилось, что все танки были сфотографированы на одном и том же фоне. В результате сеть «научилась» распознавать этот тип ландшафта, вместо того, чтобы «научиться» распознавать танки. Таким образом, сеть «понимает» не то, что от неё требовалось, а то, что проще всего обобщить.

Тестирование качества обучения нейросети необходимо проводить на примерах, которые не участвовали в ее обучении. При этом число тестовых примеров должно быть тем больше, чем выше качество обучения. Если ошибки нейронной сети имеют вероятность близкую к одной миллиардной, то и для подтверждения этой вероятности нужен миллиард тестовых примеров. Получается, что тестирование хорошо обученных нейронных сетей становится очень трудной задачей.

Заключение

Нейронные сети продемонстрировали свою способность решать сложные задачи. Они имеют уникальные потенциальные возможности, хотя не свободны от ограничений и вопросов, на которые до сих пор не существует ответа. Такая ситуация настраивает на умеренный оптимизм.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации.

Исследование методов и алгоритмов оценки надежности телекоммуникационных сетей позволяет сформулировать следующие выводы:

1. Современное состояние оценки надежности систем позволяет решать задачу оценки системы при необходимом уровне достоверности.

2. При построении нейронных сетей для оценки надежности сети надо предварительно подготовить входные данные и привести их одному виду. Данные должны быть достоверными и обоснованными для получения реальные показатели надежности.

3. Количество нейронов в нейронном слою надо подобрать оптимальным образом. Так как повышения количество нейронов замедляет работы ПО, а понижения увеличивает искажения. При повышении количество нейронов больше 5 на одно измерения ПО обучения производилось очень долго так как количество нейронов внутреннем слое достигало 390625 шт. Обучения производилась более 15 тыс. данными.

4. Классы было условно разделено на 5 (A, B, C, D, E). Но можно менять количество классов на большее количество или на меньшее.

ПО разработанной в ходе выполнения исследований было создано для исследования и проверки работоспособности алгоритма. Программа отражает принцип работы одной из видов нейронной сети Кохонена.

Применение этого метода уместно в тех случаях, когда Легко получить входные параметры для нейронной сети. Нейронную сеть лучше обучать из знаний экспертов для достоверной работоспособности сети. Обученный сеть может выдавать быстрые и точные результаты. При таких условиях результаты будут иметь достаточную достоверность оценки надёжности.

Оптимальным параметром является при 3х разделениях на измерения. При этом скорость работы и результативвность оптимальный.

Алгоритм можно усложнить и модифициоровать для получения более лучших результатов и ускорить обучение сети.

Тестирование качества обучения нейросети необходимо проводить на примерах, которые не участвовали в ее обучении. При этом число тестовых примеров должно быть тем больше, чем выше качество обучения. Если ошибки нейронной сети имеют вероятность близкую к одной миллиардной, то и для подтверждения этой вероятности нужен миллиард тестовых примеров. Получается, что тестирование хорошо обученных нейронных сетей становится очень трудной задачей.

Библиографический список

1. Закон Республики Узбекистан №560-II «Об информатизации» от 11.12.2003.
2. Постановление Президента Республики Узбекистан №ПП-1730 «О мерах по дальнейшему внедрению и развитию современных информационно-коммуникационных технологий» от 21.03.2012.
3. Постановление Президента Республики Узбекистан №ПП-1920 «О государственной программе «Год благополучия и процветания» от 14.02.2013.
4. Постановление Президента Республики Узбекистан №ПП-1957 «О дополнительных мерах по ускоренному развитию сферы услуг и сервиса в сельской местности в 2013 - 2016 годах» от 17.04.2013.
5. Каримов И.А. Последовательное продолжение курса на модернизацию страны – решающий фактор нашего развития / Доклад Президента Ислама Каримова на торжественном собрании, посвященном 18-летию Конституции Республики Узбекистан. – Ташкент, 07.12.2010.
6. Каримов И.А. Наша главная задача – дальнейшее развитие страны и повышение благосостояния народа / Доклад Президента Республики Узбекистан Ислама Каримова на заседании Кабинета Министров, посвященном итогам социально-экономического развития страны в 2009 году и важнейшим приоритетам экономической программы на 2010 год. – Ташкент, 29.01.2010.
7. Каримов И.А. Все наши устремления и программы – во имя дальнейшего развития родины и повышения благосостояния народа / Доклад Президента Республики Узбекистан Ислама Каримова на заседании правительства по итогам социально-экономического развития страны в 2010 году и важнейшим приоритетам на 2011 год. – Ташкент, 21.01.2011.

8. Каримов И.А. 2012 год станет годом поднятия на новый уровень развития нашей родины / Доклад Президента Республики Узбекистан Ислама Каримова на заседании Кабинета Министров, посвященном основным итогам 2011 года и приоритетам социально-экономического развития на 2012 год. – Ташкент, 19.01.2012.
9. Круг П. Г. «Нейронные сети и нейрокомпьютеры» Москва Издательство МЭИ 2002 г.
10.. Сергей Кабыш статья «Надежность - прежде всего» · СЕТИ И ТЕЛЕКОММУНИКАЦИИ 2004 г.
11. А. А. Зацаринный, А.И. Гаранин, С. В. Козлов. статья «Некоторые Методические Подходы к Оценке Надежности Элементов Информационно-Телекоммуникационных Сетей» Системы и средства информатики 2011 г.
12. Конспект лекций для бакалавров направлений образования «Телекоммуникационные сети и системы» ТУИТ 2004г.
13. M.O. Ball, C.J. Colbourn, J.S. Provan “Network Reliability” Авторизованный перевод Семенова Ю.А. и Гончарова А.А. (ИТЭФ/ЦНТК) 2004 г.
14. Нейман Дж. Теория самовоспроизводящихся автоматов. Дж. Нейман.М.: Мир, 1971.
15. Wolfram S. A New Kind of Science. Wolfram Media. Inc., 2002.
16. Колесников С. Распознавание образов. Общие сведения /Газета"Компьютер-Информ". Программное обеспечение. http://www.ci.ru/
17. Хайкин С. Нейронные сети: полный курс. М. : Вильямс, 2006.
18. Терехов С. А. Лекции по теории и приложениям искусственных нейронных сетей. Лабораторотория Искусственных Нейронных Сетей НТО-2. Снежинск. ВНИИТФ.

19. Smith R.A. Real-Time Language Recognition by One-Dimensional Cellular Automata // J. of Computer and System Sciences, v. 6 (1972), pp. 233–253.
20. Buchholz T., Klein A., Kutrib M. Real-Time Language Recognition by Alternating Cellular Automata /IFIG Research Report 9904. 1999. March.
21. Тоффоли Т., Марголус Н. Машины клеточных автоматов. М.: Мир,1991.
22. Астафьев Г.Б., Короновский А.А., Храмов А.Е.. Клеточные автоматы. Саратов: Колледж. 2003. http://cas.ssu.runnet.ru/sgnp/data/papers/Train/CellAutomat.pdf
23. Ulam S. Random Processes and Transformations / Procedings Int. Congr. Mathem. 1952. №2, pp. 264–275.
24. Shackleford B., Tanaka M., Carter R., Snider G. Cellular and Cryptographic Applications: FPGA implementation of neighborhood-of-four cellular automata random number generators. ACM Press. 2002.
25. Wolfram S. Cellular automation Fluids // J. Stat. Phys. 1986. Vol. 45, PP. 471–526.
26. Frish U. et al. Lattice gas hydrodynamics in two and three dimensions // Complex Systems. 1987. Vol. 1, PP. 649–707.
27. Chopard B., Droz M. Cellular automata model for heat conduction in a fluid // Physics Letters A. 1988. Vol. 126. N 8/9, PP. 476–480.
28. Наумов Л.А. Разработка среды и библиотеки CAME&L для решения задач с использованием клеточных автоматов. СПбГУ ИТМО, 2003. http://is.ifmo.ru/papers/camel/
29. Gardner M. The Fantastic Combinations of John Conway’s New Solitaire Game “Life”// Scientific American. 1970. №223, pp. 120–123.
30. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. М.: Вильямс. 2002.
31. Малинин В.В. Распознавание образов на ЭВМ. ЦИТ СГГА, 2005.

32. Orovas C. Cellular Associative Neural Networks for Pattern Recognition. University of York, 1999.

33. http://uzdtv.uz (UZDIGITAL TV)
34. http://press-service.uz (Пресс-служба Президента Республики
35. http://book.itep.ru/ (Telecommunication technologies book)
36. http://www.ict.edu.ru/ (Портал "Информационно-коммуникационные технологии в образовании")
37. http://habrahabr.ru/ (электронный журнал «Хабрахабр»)
38. http://www.basegroup.ru/ (BaseGroup Labs)
39. http://inter-vuz.tuit.uz/Elib_baza//INTUIT.ru/ (INTUIT)
40. http://library.tuit.uz/(Электронная библиотека ТУИТ)
41. http://www.lessons-tva.info/ (сайт дистанционного обучения)
42. http://k504.org/ (ХАИ, Кафедра 504 "Приема, передачи и обработки сигналов")

Приложение.

Листинг класса Neuron.

```
package neuro;

public class Neuron {
   public String clName = "";
   private int count = 0;
   public double[] weights = new double[8];

   public void addPoint(double[] point){
      for(int i=0; i<8;i++)
      {
         weights[i]=(point[i]+count*weights[i])/(count+1);

      }
      count++;
   }


   public void setNeuron(double[] point){
      for(int i=0;i<8;i++)weights[i] = point[i];

   }

   public void setClName(String cl){
      clName = cl;
   }

   public double test(double[] point)
   {
      double near = 0;
      for(int i=0; i<8;i++)
      {
         near+= (weights[i]-point[i])*(weights[i]-point[i]);
      }
      return Math.sqrt(near);
   }

}
```

Листинг класса Layer.

```
package neuro;

import neuro.Neuron;

import java.sql.*;
import java.util.ArrayList;
public class Layer {
   public Neuron[] layer;
```

```
public int searchWithName(double[] point, String clName)
{
   int ind = 0;
   double min = 3;
   for(int i=0;i<layer.length;i++)
   {
      if(layer[i].clName.equals(clName)||layer[i].clName.equals("")){
      double near = layer[i].test(point);
      if(near<min)
      {
         ind = i;
         min = near;
      }}
   }
   return ind;
}

public int search(double[] point)
{
   int ind = 0;
   double min = 8;
   for(int i=0;i<layer.length;i++)
   {
      if(!layer[i].clName.equals("")){
         double near = layer[i].test(point);
         if(near<min)
         {
            ind = i;
            min = near;
         }
      }
   }
   return ind;
}

public void study()
{
   Connection con = null;
   int po = 0;
   String url = "jdbc:mysql://localhost:3306/neuron?user=root&password=1";
   //String url = "jdbc:mysql://localhost:1527/neuro?user=root&password=123";
   try {
      con = DriverManager.getConnection(url);
      //Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      System.out.println("Got connection");

   } catch (Exception e) {
      System.out.println(e);
   }

try {
```

```
        Statement select1 = con.createStatement();
        String ss1 = "select * from study;";
        ResultSet result1 = select1.executeQuery(ss1);

        while (result1.next()) {
           po++;
           double[] p = new double[8];
           p[0] = Double.parseDouble(result1.getString(2));
           p[1] = Double.parseDouble(result1.getString(3));
           p[2] = Double.parseDouble(result1.getString(4));
           p[3] = Double.parseDouble(result1.getString(5));
           p[4] = Double.parseDouble(result1.getString(6));
           p[5] = Double.parseDouble(result1.getString(7));
           p[6] = Double.parseDouble(result1.getString(8));
           p[7] = Double.parseDouble(result1.getString(9));
           String clName = result1.getString(10);
           int neuronIndex = searchWithName(p, clName);
           //System.out.println(layer[neuronIndex].clName+" / "+clName);


              layer[neuronIndex].addPoint(p);
              layer[neuronIndex].setClName(clName);
        }
     } catch (SQLException ex) {
        ex.printStackTrace();
     }
System.out.println(po);
}

     public ArrayList<String> test()
{
     int yes = 0;
     int no = 0;
     Connection con = null;
     ArrayList<String> ar=new ArrayList<String>();
     String url = "jdbc:mysql://localhost:3306/neuron?user=root&password=1";
     //String url = "jdbc:mysql://localhost:1527/neuro?user=root&password=123";
     try {
        con = DriverManager.getConnection(url);
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        System.out.println("Got connection");

     } catch (Exception e) {
        System.out.println(e);
     }

try {
        Statement select1 = con.createStatement();
        String ss1 = "select * from test;";
        ResultSet result1 = select1.executeQuery(ss1);

        while (result1.next()) {
```

```
            double[] p = new double[8];
            p[0] = Double.parseDouble(result1.getString(2));
            p[1] = Double.parseDouble(result1.getString(3));
            p[2] = Double.parseDouble(result1.getString(4));
            p[3] = Double.parseDouble(result1.getString(5));
            p[4] = Double.parseDouble(result1.getString(6));
            p[5] = Double.parseDouble(result1.getString(7));
            p[6] = Double.parseDouble(result1.getString(8));
            p[7] = Double.parseDouble(result1.getString(9));
            String clName = result1.getString(10);
            //System.out.println(p[0]+" / "+p[1]+" / "+p[2]+" / "+p[3]+" / "+p[4]+" / "+p[5]+" /
"+p[6]+" / "+p[7]);
            int neuronIndex = search(p);
            //System.out.println(layer[neuronIndex].clName+" / "+clName);
            if(layer[neuronIndex].clName.equals(clName))
            {
               ar.add("YES");
               yes++;
            } else{
               ar.add("NO");
               no++;
            }

         }
      } catch (SQLException ex) {
         ex.printStackTrace();
      }
   System.out.println("YES "+yes+"   NO "+no);
   return ar;
   }


   public void setLayer(int points)
   {
      int ind = (Integer)Math.round((float) Math.pow(points, 8));
      layer = new Neuron[ind];
      //System.out.println(ind);
      //layer[0].tes();
      for(int i=0;i<ind;i++) {layer[i] = new Neuron();}
      int index =0;
      double[] w = new double[8];
      for(int i=0; i<points;i++)
      {
         w[0] = (i+0.5)/points;
         for(int j=0; j<points;j++)
            {
               w[1] = (j+0.5)/points;
               for(int k=0; k<points;k++)
                  {
                     w[2] = (k+0.5)/points;
                     for(int l=0; l<points;l++)
                        {
```

```
                        w[3] = (l+0.5)/points;
                        for(int m=0; m<points;m++)
                            {
                                w[4] = (m+0.5)/points;
                                for(int n=0; n<points;n++)
                                    {
                                        w[5] = (n+0.5)/points;
                                        for(int o=0; o<points;o++)
                                            {
                                                w[6] = (o+0.5)/points;
                                                for(int p=0; p<points;p++)
                                                    {
                                                        w[7] = (p+0.5)/points;
                                                        layer[index].setNeuron(w);
                                                        index++;
                                                    }
                                            }
                                    }
                            }
                    }
            }
        }
    }

    public void look()
    {
        for(int i = 0; i<layer.length; i++){
            double[] d = layer[i].weights;
            System.out.println(layer[i].clName+" / "+d[0]+" / "+d[1]+" / "+d[2]+" / "+d[3]+" /
"+d[4]+" / "+d[5]+" / "+d[6]+" / "+d[7]);}
    }
    public void ins()
    {
        Connection con = null;
        String url = "jdbc:mysql://localhost:3306/neuron?user=root&password=1";
        try {
            con = DriverManager.getConnection(url);System.out.println(1);
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            System.out.println("Got connection");

        } catch (Exception e) {
            System.out.println(e);
        }
        for(int j=586; j<100000; j++){
            try {
                double delta = 0;
                String clName = "";
                double[] p = new double[8];
                for(int i=0; i<8; i++)
                {
                    p[i] = (4+Math.random())/5;
```

```
                delta+=Math.pow(p[i], 2);
              }
              delta = Math.sqrt(delta);
              if(delta<Math.sqrt(8)/5) clName = "E";
              else if(delta<2*Math.sqrt(8)/5) clName = "D";
              else if(delta<3*Math.sqrt(8)/5) clName = "C";
              else if(delta<4*Math.sqrt(8)/5) clName = "B";
              else if(delta<Math.sqrt(8)) clName = "A";

              String ss1 = "insert into test (id, par1, par2, par3, par4, par5, par6, par7, par8, class)
" +
             "value("+j+","+p[0]+","+
p[1]+","+p[2]+","+p[3]+","+p[4]+","+p[5]+","+p[6]+","+p[7]+","'+clName+'");";
              System.out.println(ss1);
              PreparedStatement preparedStmt = con.prepareStatement(ss1);
              preparedStmt.execute();

          } catch (SQLException ex) {
                ex.printStackTrace();
          }
    }
    }
}
```

Листинг выполняющего класса Neuro.

```
package neuro;

import neuro.Layer;
import java.util.ArrayList;
import java.util.Iterator;

public class Neuro {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("BEGGIN");

          Layer layer = new Layer();
//          layer.ins();

          layer.setLayer(4);

          System.out.println("4^8="+Math.pow(4, 8));
    ArrayList<String> ar=new ArrayList<String>();
    layer.study();
    ar = layer.test();
    //layer.look();
    for (Iterator iterator = ar.iterator(); iterator.hasNext();)
    {
```

```
            System.out.println((String) iterator.next());
        }

        double[] p = new double[8];

            p[0] = 0;
            p[1] = 0;
            p[2] = 0;
            p[3] = 0;
            p[4] = 0;
            p[5] = 0;
            p[6] = 0;
            p[7] = 0;
            System.err.println("CLASS: "+layer.layer[layer.search(p)].clName);


        }
    }
```