**Министерство образования республики Беларусь**

**Учреждение образования**

**Белорусский государственный университет информатики и радиоэлектроники**

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**РЕДАКТОР НЕЙРОННОЙ СЕТИ**

БГУИР КР 1-40 02 01 27\*ПЗ

**Студент**: гр.150501 Белов А. В.

**Руководитель**: Калабухов Е. В.

**Минск 2013**

# ПОСТАНОВКА ЗАДАЧ

## Задание

Необходимо реализовать программу редактор нейронных сетей, обладающую достаточным функционалом для получения пользователем начальных знаний по принципам работы нейросетевых алгоритмов.

Реализовать функционал добавления, изменения, сохранения, загрузки нейронных сетей. В качестве конфигурационных файлов использовать файлы формата XML.

Если это будет необходимо, использовать в программе параллельные вычисления на базе технологии NVIDIA CUDA. Изучить показателю улучшения производительности нейронных сетей при применении технологий параллельной обработки информации.

Обеспечить возможность тренировки сети пользователем вручную, либо с помощью отдельно сформированных файлов с обучающим множеством.

Для предотвращения явления паралича сети реализовать возможность задания констант скорости обучения сети и множителя последней тренировки сети.

Снабдить программу дружественным интерфейсом пользователя, подходящем для первого знакомства с тематикой нейронных сетей.

**ОГЛАВЛЕНИЕ**

[ПОСТАНОВКА ЗАДАЧ 2](#_Toc358185210)

[Задание 2](#_Toc358185211)

[ВВЕДЕНИЕ 4](#_Toc358185212)

[ОБЗОРНАЯ ЧАСТЬ 5](#_Toc358185213)

[ОПИСАНИЕ ДИАГРАММЫ КЛАССОВ И ДАННЫХ ПРОГРАММЫ 9](#_Toc358185214)

[ОПИСАНИЕ АЛГОРИТМОВ 12](#_Toc358185215)

[Алгоритм чтения конфигурации из файла 12](#_Toc358185216)

[Алгоритм прямого прохода по нейронной сети 13](#_Toc358185217)

[Алгоритм обратного распространения ошибки 14](#_Toc358185218)

[ТЕСТИРОВАНИЕ 15](#_Toc358185219)

[Системные требования 15](#_Toc358185220)

[Программные требования 15](#_Toc358185221)

[Работа с программой 15](#_Toc358185222)

[ЗАКЛЮЧЕНИЕ 18](#_Toc358185223)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc358185224)

[ПРИЛОЖЕНИЕ А 20](#_Toc358185225)

[Диаграмма классов 20](#_Toc358185226)

[ПРИЛОЖЕНИЕ Б 21](#_Toc358185227)

[Блок-схема 1 21](#_Toc358185228)

[Блок-схема 2 22](#_Toc358185229)

[Блок-схема 3 23](#_Toc358185230)

# ВВЕДЕНИЕ

Искусственная нейронная сеть – программируемая конструкция, имитирующая свойства биологических нейронов. Нейронные сети могут решать широкий круг задач обработки и анализа данных: распознавание и классификация образов, прогнозирование, управление, автоматическая торговля, анализ естественного языка, медицинских данных и т.д. Конкурентами нейронных сетей являются классические методы анализа данных, однако нейронные сети имеют над ними ряд преимуществ. Используя способность обучаться на множестве примеров, нейронная сеть способна решать задачи, в которых неизвестны закономерности развития ситуации и зависимости между входными и выходными данными. Нейронные сети устойчивы к зашумленности входных данных и способны адаптироваться к изменениям окружающей среды. Также нейронные сети обладают потенциальным сверхвысоким быстродействием и значительной отказоустойчивостью за счет массового параллелизма обработки информации.

Однако есть у нейронных сетей и недостатки. Как следствие отсутствия формализованных алгоритмов настройки сети и ее высокой сложности, возникает необходимость в привлечении высококлассных специалистов, обладающих необходимыми знаниями. Отсутствует строгая теория по выбору архитектуры нейронной сети под специфические задачи, хотя некоторая работа в этом направлении ведется. Из обученной сети практически невозможно извлечь приобретенные знания, то есть, нейронная сеть является своего рода черным ящиком.

Для реализации нейросетевых алгоритмов используются различные языки программирования, в том числе и специализированные диалекты LISP. Но использование языка С++, одного из самых распространенных и востребованных в мире языков программирования, может быть обосновано широким использованием данного языка для организации параллельных вычислений, что в случае нейронных сетей дает очень весомый выигрыш в быстродействии вследствие их нелинейной природы.

Программа редактор нейронных сетей так же, как и сами нейронные сети, может использоваться для очень широкого спектра задач. Но наиболее актуальным и адекватным примером использования, на мой взгляд, является академическое применение. Нейронные сети в том или ином виде изучаются на достаточно большом количестве IT-специальностей. И на начальном этапе изучения предмета весьма пригодился бы простой редактор, который может познакомить пользователя с принципами работы нейронных сетей, их преимуществами и недостатками. Сейчас же, в основном, изучение нейронных сетей строится на использовании программ, обладающих огромным избыточным функционалом, что порождает известные трудности в обучении.

# ОБЗОРНАЯ ЧАСТЬ

Программа «Редактор нейронной сети» представляет собой инструмент для создания и редактирования простых нейронных сетей. Единственным видом нейронной сети, реализованным в данной программе, является полносвязный многослойный перцептрон Румельхарта, в котором обучение реализовано с помощью алгоритма обратного распространения ошибки.

Перцептрон Румельхарта (или многослойный перцептрон) – частный случай перцептрона Розенблатта, в котором алгоритм обратного распространения ошибки обучает все слои и используются, в основном, нелинейные функции активации. В многослойном перцептроне допускается произвольная структура связей, в том числе и полносвязные сети. Чаще всего в практических применениях используется не более трех обучаемых слоев, так как необходимость большего их числа, на данный момент, никак не обоснована и ведет к потере скорости обучения без преимуществ в качестве. Теоретически, достаточно использовать один скрытый обучаемый слой, чтобы получить линейную разделимость для выходного представления, то есть, разделить два множества векторов входов одной гиперплоскостью. Однако, существует предположение, что, используя большее число слоев, можно уменьшить число элементов в них, то есть суммарное число нейронов в сети будет меньше, чем если использовать один скрытый слой. (Список используемых источников, ссылка 4)

Также стоит упомянуть про распространенные проблемы алгоритма обратного распространения ошибки – так называемый паралич сети и проблему переобучения. Паралич сети возникает при достижении весовыми коэффициентами связей очень больших значений, что приводит к полной невозможности различить значения на выходе сети, так как они упираются в максимум функции распределения. Переобучение сети предполагает потерю сетью способности к обобщению вследствие обучения на узком множестве значений. То есть, сеть сможет верно трактовать только те данные, распознавать которые ее научили, но на любых других наборах давать абсолютно не имеющий смысла ответ. (Список используемых источников, ссылка 5)

Многослойный перцептрон Румельхарта был выбран для реализации в данной работе именно потому, что основной задачей программы является начальное обучение предмету нейронных сетей. И перцептрон Румельхарта является наиболее простым примером нейронных сетей, который, тем не менее, позволяет получить основные теоретические знания и навыки использования нейронных сетей на практике. В частности, сеть протестирована на стандартной задаче для начального изучения нейросетевых алгоритмов – задаче исключающего или (XOR). Пользователям представляется реальная возможность проверить некоторые утверждения о наиболее оптимальном числе слоев в сети и минимально достаточном числе нейронов в слое.

Как уже было сказано, зачастую при изучении нейронных сетей используются программные пакеты, обладающие огромным избыточным функционалом, что можно рассматривать и как преимущество, в случае достаточно хорошей теоретической подготовки и необходимости в экспериментах, и как недостаток, в случае обладания лишь базовым представлением о работе нейронных сетей и используемых алгоритмов. Зачастую чтобы приступить к углубленному изучению темы, необходимо сначала получить начальное представление о ней, обойдясь сравнительно малыми затратами. Тем не менее, существует ряд аналогов для моей разработки, которые используются уже достаточно широко и предоставляют широкий спектр возможностей для работы с нейронными сетями.

Одним из наиболее мощных пакетов для разработки и редактирования нейронных сетей обладает продукт компании американской The MathWorks – пакет прикладных программ MATLAB. Графический интерфейс для построения нейронных сетей Neural Network Toolbox впервые появился в MATLAB версии 6.0. Преимущество использования именно продукта MATLAB заключается в возможности комбинировать нейронные сети с многочисленными алгоритмами и функциями для инженерных расчетов. Также преимуществом MATLAB является наличие высокоуровневого интерпретируемого языка программирования, включающего основанные на матрицах структуры данных и широкий спектр функций, а также объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках. И, безусловно, немаловажным положительным фактором является наличие очень большого сообщества пользователей, которое позволяет очень быстро получить ответы на возникающие вопросы и квалифицированную помощь. (Список используемых источников, ссылка 6)

Недостатками пакета программ MATLAB является, все же, относительно небольшое число реализованных специфичных структур нейронных сетей, хотя надо заметить, что все основные архитектуры в нем присутствуют. Также реализовано крайне небольшое число функций активации нейронов – целых три штуки. Рассматривать ли это как недостаток данного программного пакета – зависит от поставленной задачи, однако в графическом интерфейсе отсутствует возможность задания произвольной функции активации нейрона, что могло бы быть полезным для исследований конкретных нейросетевых алгоритмов.

Единственным в мире крупным нейросетевым программным продуктом, полностью переведенным на русский язык (по заявлению компании-разработчика), является программный пакет Statistica, разработанный, как это ни странно, компанией Statsoft, тоже американской, а точнее, его модуль Statistica Automated Neural Networks. Как и MATLAB, Statistica тоже представляет большой пакет программ для инженерных расчетов, а значит, нейронные сети можно успешно комбинировать с другими видами алгоритмов в сложных законченных системах. В системе реализовано большое количество нейросетевых архитектур, достаточное для решения многих практических задач и, что немаловажно, мастер решений, который автоматически анализирует задачу и выбирает несколько подходящих для реализации архитектур. Также в пакете программ представлено большое число различных алгоритмов обучения сети, что позволяет иметь большую свободу в выборе оптимальных алгоритмов для решения определенного типа задач. Рекламируются также возможность создания гибридных нейронных сетей, состоящих из нескольких различных архитектур и база готовых решений.

Несмотря на все богатство функциональных возможностей и хорошую русификацию, критике можно подвергнуть и продукт Statistica Automated Neural Networks. Так, благодаря впечатляющим функциональным возможностям, мы имеем достаточно сложный интерфейс самой программы. В итоге задача изучения нейронных сетей и экспериментирования с их различными структурами и типами плавно превращается в задачу изучения интерфейса одной отдельно взятой программы. В оправдание можно сказать, что для программного модуля имеется русский раздел справки и нечто похожее на ликбез по построению стандартных нейросетевых решений. В целом, программа рассчитана на практическое применение в реальных областях специально обученными людьми, что не соответствует задаче академического обучения. Но продукт компании Statsoft можно смело рекомендовать пользователям, достаточно глубоко изучившим нейронные сети и желающим применить полученные знания в какой-либо прикладной задаче. (Список используемых источников, ссылка 7)

Существует достаточно большое количество мелких продуктов для конструирования нейронных сетей. Многие программисты реализуют их в процессе обучения нейросетевым алгоритмам, так как это позволяет лучше отточить необходимые навыки.

Одним из примеров может служить немецкая программа MemBrain. Отличается от вышеперечисленных крупных продуктов она тем, что предлагает бесплатную лицензию для некоммерческого и академического использования. MemBrain также предлагает встроенный объектно-ориентированный скриптовый язык и автоматическое генерирование кода на языке С, а также создание динамических библиотек для использования в сторонних приложениях. Однако данный продукт, все же, не совсем подходит для начальных этапов работы с нейросетями по причине отсутствия поддержки какого-либо языка кроме английского, что, впрочем, не должно стать для студентов проблемой, но дополнительным раздражающим элементом стать вполне может. (Список используемых источников, ссылка 8)

Еще можно назвать конкурентной разработкой фреймворк для создания нейронных сетей Neuroph, реализованный на языке Java и имеющий лаконичный GUI модуль для проектирования нейронных сетей. Несмотря на свою простоту и хорошую поддержку, данный фреймворк имеет смысл рассматривать, скорее, как средство изучение именно программных аспектов реализации нейронных сетей, так как для практического применения язык Java может быть не столь хорошо подходящим для организации параллельных вычислений, как ставший стандартом де-факто в этой области С++. (Список используемых источников, ссылка 9)

Рассмотрим функциональность разрабатываемого редактора нейронных сетей.

* Конструирование простых моделей перцептрона Румельхарта с отображением графического представления получившейся структуры сети.
* Обучение полученной нейронной сети с помощью множества обучающих примеров, подготавливаемых отдельно в виде тренировочного файла, либо в ручном режиме с заданием значений на вхоже сети и желаемых результатов на ее выходе
* Сохранение и загрузка ранее созданных и обученных моделей сетей из файлов формата XML.
* Отображение результата работы сети для задаваемых пользователем значений, так с возможностью загрузки результата из специально созданного файла для большого числа входов и сохранение полученных результатов в отдельный файл.
* Обеспечение возможности редактирования слоев сети, хотя бы в плане задания функций активации различных видов.
* Реализация параллельных вычислений на базе технологии NVIDIA CUDA для ускорения работы нейронных сетей.

Функционал сохранения и загрузки различных файлов, необходимых для работы с программой, обеспечивается c использованием сторонней библиотеки для работы с XML файлами tinyxml. В процессе реализации было решено применять данный формат для абсолютно всех файлов программы, ввиду его изначальной приспособленности для реализации вложенных структур.

# ОПИСАНИЕ ДИАГРАММЫ КЛАССОВ И ДАННЫХ ПРОГРАММЫ

Основными классами программы являются классы NN, Neuron, структура Connection.

Класс NN отвечает за реализацию нейронной сети, в нем реализуются методы для тренировки сети и содержится вектор векторов нейронов, что соответствует распределению нейронов по слоям в реальной структуре сети. Для данного класса используется паттерн проектирования Singleton, так как за все время работы программы может быть создана всего одна нейронная сеть.

Класс Neuron обеспечивает реализацию базового элемента любой нейронной сети – нейрона. Данный класс содержит вектор структур Connection, что соответствует наличию у нейронов весовых коэффициентов. Также в классе содержатся методы для получения выходного значения нейрона. Для обработки выходных значений нейрона может быть установлена одна из трех доступных функций.

Структура Connection хранит значение веса связи двух нейронов, а также разницу между ожидаемым значением и реальным значением выхода нейрона, что необходимо для обучения методом обратного распространения ошибки.

Классы WriteXML, ReadXML, TrainingData, InputVector, OutputVector обеспечивают работу программы с файлами формата XML. Данные классы используют стороннюю библиотеку tinyxml, обеспечивающую базовый уровень работы с форматом XML.

Класс WriteXML обеспечивает запись в XML файл полной структуры нейронной сети в виде слоев и нейронов вместе со всеми необходимыми данными, вроде весовых коэффициентов связей отдельного нейрона и общей ошибки всей сети.

Класс ReadXML, соответственно, обеспечивает чтение из файла данных о нейронной сети. Причем, благодаря структуре файла, содержащей все основные данные о нейронной сети и входящих в нее элементах, возможна загрузка заранее обученной сети, что удобно для создания базы работающих примеров.

Класс TrainingData обеспечивает загрузку из XML файла данных для обучения сети, которые представляют собой значения входов нейрона и желаемые значения его выходов. Файл тренировочных данных создается отдельно.

Класс InputVector избавляет пользователя от необходимости вручную вводить данные входов нейронной сети, что весьма полезно при наличии, скажем, нескольких десятков нейронов-рецепторов во входном слое.

Класс OutputVector позволяет записать полученный ответ сети в отдельный файл, для лучшего его сохранения. Другой возможностью сохранить ответ сети на определенные значения входов является копирование данных из текстового поля.

Класс DrawNeurons и структуры NeuroCoordinates и NeuroConnection реализуют графическое отображение структуры сети с помощью средств Windows Forms.

Класс DrawNeurons несет функциональную нагрузку в виде вычисления по заданной топологии сети (вектор целых значений, который кодирует информацию о числе слоев в сети и числе нейронов в каждом слое) координат на декартовой плоскости каждого нейрона в слое и вычисления координат для отображения всех связей между нейронами. Нейроны отображаются в интерфейсе в виде эллипсов красного цвета, связи – в виде черных линий, соединяющих эллипсы друг с другом. Класс содержит вектора структур NeuroCoordinates и NeuroConnection.

Структура NeuroCoordinates используется классом DrawNeurons для хранения для каждого отдельного нейрона позиции его центра и значения радиуса нейрона (которое устанавливается в зависимости от числа нейронов в слое и числа слоев в сети). Также структура содержит вложенную структуру Dimensions, содержащую значения высоты и ширины нейрона, и структуру LeftCorner, хранящую координаты левого угла прямоугольника, который ограничивает круг нейрона.

Структура NeuroConnection пары координат для концов линии связи между двумя нейронами. Координаты краев линии являются координатами центров соответствующих нейронов.

Побочные классы-помощники TopologyHelper, EtaAlphaHelper и VectorAndString содержат данные и методы, необходимые для функционирования других классов.

Класс TopologyHelper реализован с использованием паттерна Singleton и содержит вектор целых значений Topology, представляющий структуру сети. Используется для передачи топологии сети из более поздней вызванной формы в более раннюю. В процессе изучения вопроса передачи параметров между формами выбор был остановлен именно на классе-одиночке.

Класс EtaAlphaHelper содержит константы, необходимые для процесса обучения сети, такие как скорость и множитель последнего изменения весов. Данный класс был добавлен в процессе реализации интерфейса для регулировки скорости обучения сети.

Класс VectorAndString содержит три очень полезных метода и используется во многих формах интерфейса. Метод StringToVector преобразует текст из входного поля в вектор дробных чисел, которые можно отправить на вход нейронной сети. Соответственно, метод VectorToString преобразует выходной вектор результатов работы нейронной сети в строку, которая может отображаться в текстовом поле. Метод SystemStringToStdString реализует преобразование управляемого типа System::String в неуправляемый тип Std::string. Используется при задании имен файлов для открытия и сохранения.

Конфигурация сети представляет собой XML файл, который содержит полную информацию о нейронной сети: ошибку сети, все слои, все нейроны в слоях, функцию активации для каждого нейрона и значения весов связей каждого нейрона. Файл может содержать любое число нейронов и слоев.

Структура файла конфигурации:

<Network m\_recentAverageError=" " >

<Layer>

<Neuron func=" " >

<Connection weight=" " deltaweight=" " />

…

</Neuron>

…

</Layer>

…

</Ntework>

Также возможет импорт для дальнейших вычислений значений входов из файла входных данных.

Структура входного файла:

<Input>

<Value val = " " />

…

</Input>

Возможен экспорт результатов работы сети в выходной файл.

Структура выходного файла:

<Output>

<Value val = " " /> …

</Output>

# ОПИСАНИЕ АЛГОРИТМОВ

## Алгоритм чтения конфигурации из файла

1. Открыть XML документ.
2. Если документ не загружен – выход.
3. Получить элемент с тегом “Network”
4. Если элемент не получен – выход.
5. Получить значение средней ошибки сети.
6. Получить элемент с тегом “Layer”
7. Пока существуют элементы с тегом “Layer”
   1. Пока существуют элементы с тегом “Neuron”
   2. Получить элемент с тегом “Neuron”
      1. Получить значение атрибута func (функция активации) нейрона
      2. Получить значение элемента с тегом “Connection”
      3. Пока существуют элементы с тегом “Connection”
         1. Получить значение атрибута weight (весовой коэффициент)
         2. Получить значение атрибута deltaWeight (разница ожидаемого и реального значений на выходе нейрона)
         3. Перейти к следующему элементу с тегом “Connection”
      4. Перейти к следующему элементу с тегом “Neuron”
   3. Перейти к следующему элементу с тегом “Layer”
8. Закрыть XML документ

Алгоритм чтения конфигурации сети из файла позволяет создать экземпляр класса нейронной сети с определенной топологией и установить значения весовых коэффициентов связей нейронов таким образом, что сеть сразу после создания уже будет обучена решать определенную задачу.

Функция чтения конфигурации сети из XML файла реализуется с помощью сторонней подключаемой библиотеки для работы с XML файлами tinyxml. Для файла конфигурации сети специфичным является только иерархическая структура специализированных элементов, а также атрибуты этих элементов. Весть функционал физического уровня реализован в tinyxml.

Блок-схема находится в Приложении Б, схема 1.

## Алгоритм прямого прохода по нейронной сети

1. Пока счетчик меньше числа элементов в векторе, поданном на вход.
   1. Занести значение входного вектора в нейрон рецепторного слоя.
   2. Увеличить значение счетчика.
2. Пока не пройдены все слои.
   1. Пока не пройдены все нейроны слоя.
      1. Подсчитать линейную комбинацию входов нейрона (сумма значений входов, умноженная на весовые коэффициенты связей).
      2. Пропустить сумму, полученную на предыдущем шаге, через функцию активации нейрона.
   2. Перейти к следующему нейрону.
3. Перейти к следующему слою.
4. Получить значения последнего слоя нейронов.

С помощью функции прямого прохода по сети можно получить значения выхода нейронной сети (выходные значения последнего слоя нейронов).

Для вычисления выходных значений сети, алгоритм, для каждого слоя нейронов, вычисляет значения их выходов путем подсчета линейной комбинации выходных значений нейронов предыдущего слоя с весовыми коэффициентами связей, пропускает значения выходов через функцию активации и повторяет это, пока не пройдены все слои нейронной сети.

Так как при вычислении выходов нейронов слоя никак не используются другие нейроны текущего слоя, вся операция прекрасно подается распараллеливанию. Значения все нейронов в слое считаются параллельно, то есть выходные значения всего слоя получаются за время подсчета выходного значения одного нейрона. Что, в случае большого числа нейронов в слое, дает очень большой выигрыш по времени. Но абсолютно не дает выигрыша по времени в случае наличия большого числа слоев с малым количеством нейронов в них.

Имеет смысл программно реализовывать данный алгоритм с использованием технологии Nvidia CUDA.

Блок-схема находится в Приложении Б, схема 2.

## Алгоритм обратного распространения ошибки

1. Пока не пройдены все нейроны выходного слоя за исключением биаса.
   1. Вычисляется разница между ожидаемым значением на нейроне выходного слоя и реально существующим значением.
   2. К ошибке сети прибавляется квадрат разницы между ожидаемым значением на нейроне выходного слоя и реально существующим значением.
2. Ошибка сети делится на число нейронов в выходном слое, не включая биас. Получается среднее значение ошибки.
3. Из ошибки сети извлекается квадратный корень. Так вычисляется среднее квадратичное отклонение от заданных значений.
4. Вычисляется по формуле среднее значение ошибки для сети.
5. Пока не пройдены все нейроны выходного слоя за исключением биаса
   1. Вычислить разницу между ожидаемым значением на нейроне и реально существующим значением.
   2. Подсчитать значение градиента для нейрона выходного слоя.
6. Пока не пройдены все скрытые слои
   1. Пока не пройдены все нейроны скрытого слоя
      1. Подсчитать значение градиента для нейрона скрытого слоя.
7. Пока не пройдены все слои нейронной сети (включая выходной и скрытые)
   1. Пока не пройдены все нейроны в слое.
      1. Обновить значение веса для всех связей данного нейрона с нейронами предыдущего слоя.

Алгоритм обратного распространения ошибки используется для обучения нейронной сети. Основная идея состоит в том, что ошибка распространяется по сети в обратном направлении, от выходов к входам и для всех связей вычисляются новые весовые коэффициенты.

У алгоритма обратного распространения ошибки существуют определенные недостатки, в том числе ранее описанный паралич сети, при котором значения весов могут стать очень большими величинами и процесс обучения замирает. Существуют методы предохранения от паралича и восстановления после него, но пока они могут рассматриваться лишь как экспериментальные.

Для данного алгоритма также имеет смысл реализация параллельных вычислений.

Блок-схема находится в Приложении Б, схема 3.

# ТЕСТИРОВАНИЕ

## Системные требования

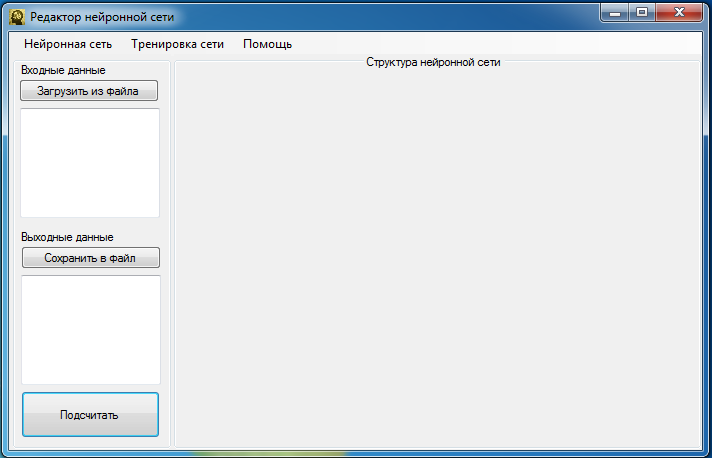
* Операционная система: Microsoft Windows XP (SP2) и выше.
* Процессор: Intel Pentium или AMD Athlon (Sempron, Duron) 1000 МГц или лучше.
* Оперативная память: 512 Мб для Windows XP, 1024 Мб для Windows Vista\7\8.
* Устройство ввода: клавиатура, мышь.

## Программные требования

* Библиотека .Net Framework версии 4.0.3019 или выше.

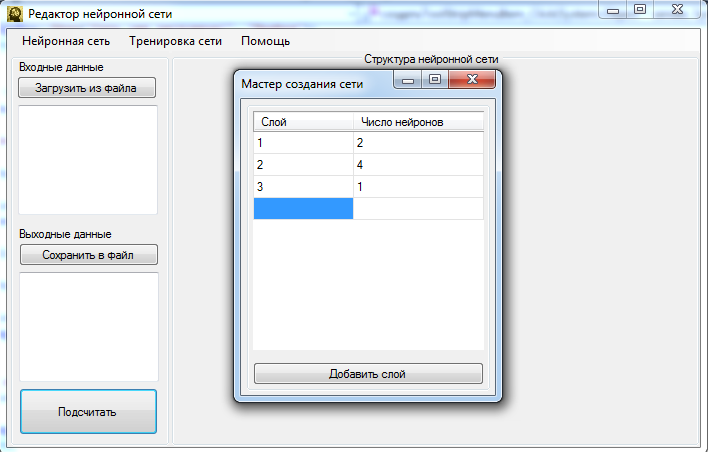
## Работа с программой

В начале работы с программой пользователь видит перед собой окно приложения с пустым полем структуры нейронной сети.

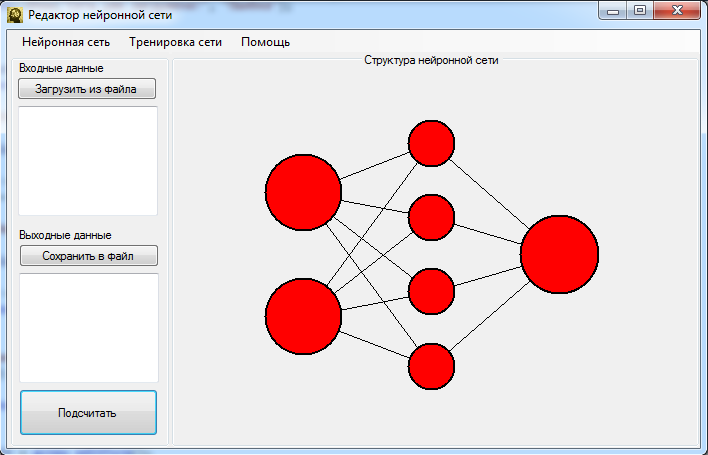


Есть несколько вариантов начала работы с системой.

Нажатие сочетания клавиш Ctrl+N либо выбор в пункте меню «Нейронная сеть» раздела «Создать» приведет пользователя в Мастер создания сети, в котором можно создать сеть путем добавления в нее слоев с определенным числом нейронов – от входного слоя к последнему, выходному.



В случае правильного создания сети в области отображения структуры сети пользователь увидит построенную структурную диаграмму сети.



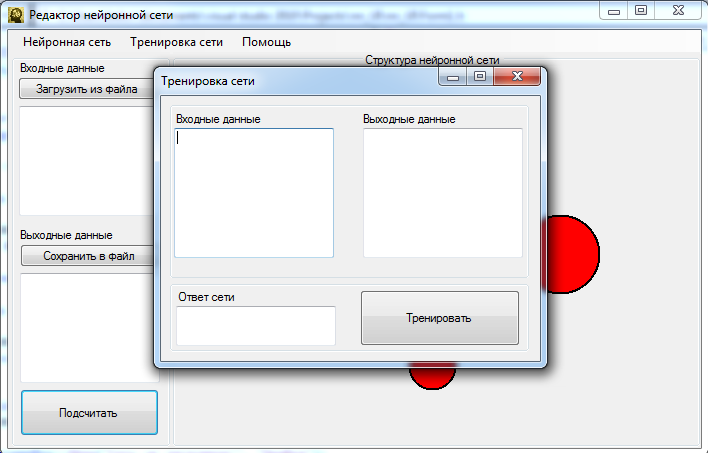
На скриншоте выше изображена диаграмма сети для задачи XOR.

О неправильном задании сети программа сообщит в диалоговых окнах.

Другой вариант начала работы с программой – загрузка ранее сохраненной конфигурации уже обученной сети через пункт меню «Нейронная сеть» – «Загрузить» либо сочетанием клавиш Ctrl+O.

В пункте меню «Нейронная сеть» – «Редактировать» (сочетание клавиш Ctrl+E) пользователям предложено изменить для любого слоя сети функцию активации (доступно три функции: экспоненциальная, логистическая и гиперболический тангенс).

Обучение сети реализуется либо в автоматическом режиме с помощью заранее созданного файла с обучающими примерами через пункт меню «Тренировка сети» – «Загрузить данные» (сочетание клавиш Ctrl+T), либо в ручном режиме вызовом «Тренировка сети» – «Ручной ввод».



Пункт меню «Тренировка сети» – «Установка констант» позволяет задать константы режима обучения сети.

После обучения сети может возникнуть желание ее сохранить – к услугам пользователей пункт «Нейронная сеть» – «Сохранить» (сочетание клавиш Ctrl+S).

Наконец, в главном окне можно ввести в текстовое поле или загрузить из файла данные для тестирования сети в формате «float;float;float…» и, нажав на кнопку «Подсчитать», получить ответ сети, который можно сохранить в XML файл.

# ЗАКЛЮЧЕНИЕ

Прежде всего, стоит отметить, что в программе так и не были реализованы параллельные вычисления. Но опыт показал, что графическое отображение структуры сети рисуется дольше, чем нейронная сеть обучается на двух тысячах тестовых наборах. А значит, применение CUDA саму по себе программу не ускорит. Параллельные вычисления стоит использовать там, где это дает существенный выигрыш в производительности и, тем не менее, о добавлении такой возможности стоит подумать в дальнейшем.

Также стоит заметить, что, несмотря на название программы «Редактор нейронной сети» редактирование происходит только в одном случае – установке функции активации для слоя. В основном же программа служит целям конструирования и обучения нейронной сети. В будущем хотелось бы добавить больше функций редактирования, к примеру, добавление и удаление связей между нейронами.

Получены некоторые начальные знания в области нейронных сетей, что, несомненно, лишним не будет. Эта тема еще долго будет оставаться актуальной и даже больше, она будет активно развиваться. В связи с этим возникает желание попробовать реализацию нейронных сетей в железе (на ПЛИС).

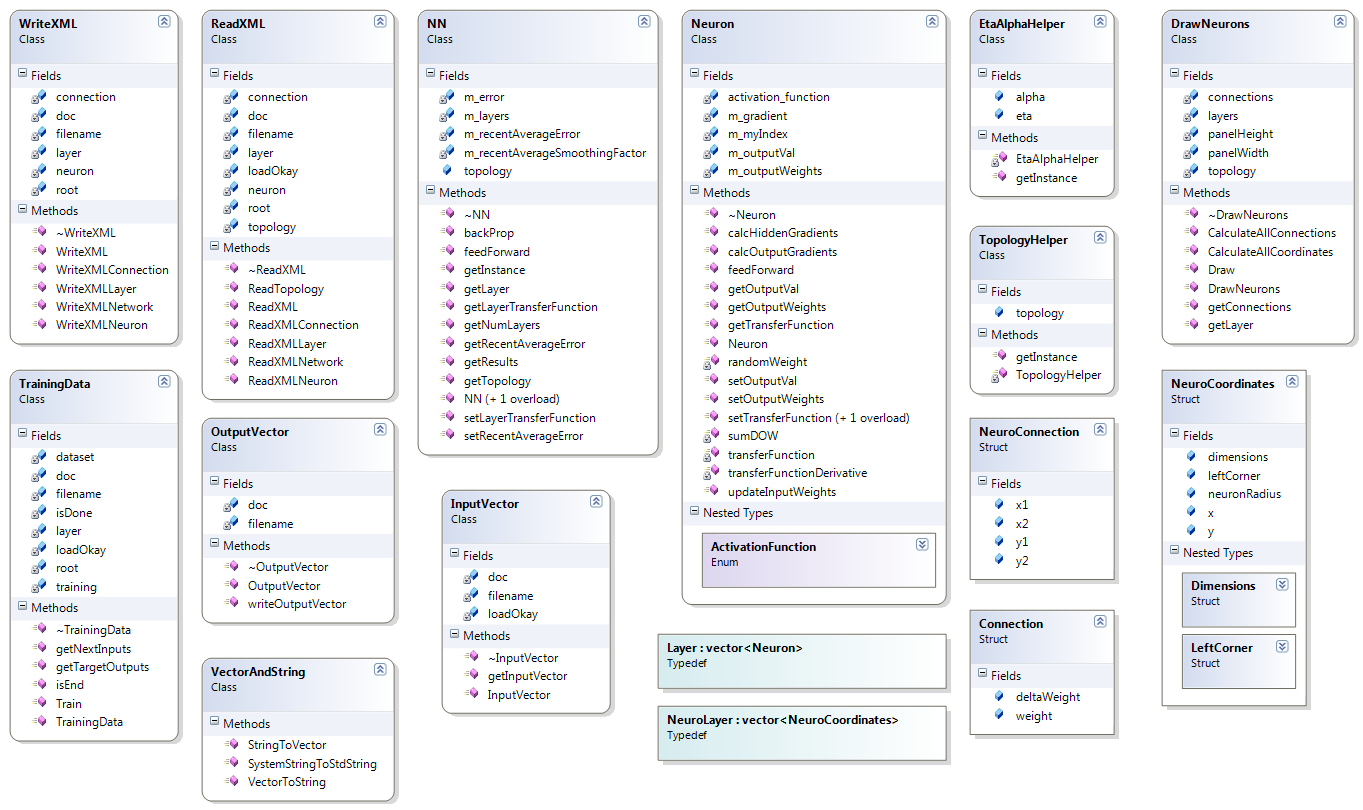
В целом, программу «Редактор нейронных сетей» можно использовать по своему основному назначению – использованию для первичного знакомства с тематикой нейронных сетей. Программа позволяет сконструировать сеть, обучить ее, сохранить для позднейшей работы, получить результаты вычислений в виде выходных файлов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес «Приемы объектно-ориентированного проектирования. Паттерны проектирования» – Санкт-Петербург : Питер, 2012.
2. А. А. Ежов, С. А. Шумский «Нейрокомпьютинг и его применения в экономике и бизнесе» – Москва, 1998. – 216 с.
3. С. Хайкин «Нейронные сети. Полный курс» – Москва : Вильямс, 2006. – 1105 с.
4. Многослойный перцептрон Румельхарта [Электронный ресурс]. – Электронные данные. – Режим доступа : http://ru.wikipedia.org/wiki/ Многослойный\_перцептрон\_Румельхарта
5. Метод обратного распространения ошибки [Электронный ресурс]. – Электронные данные. – Режим доступа : http://ru.wikipedia.org/wiki/ Метод обратного\_ распространения\_ошибки
6. Описание продукта MATLAB [Электронный ресурс]. – Электронные данные. – Режим доступа : http://mathworks.com/products/matlab/
7. Описание продукта Statistica Automated Neural Networks [Электронный ресурс]. – Электронные данные. – Режим доступа : http://statsoft.com/products/statistica-automated-neural-networks/
8. Описание продукта Membrain [Электронный ресурс]. – Электронные данные. – Режим доступа : http://membrain-nn.de /english/details\_en.htm
9. Описание фреймфорка Neuroph [Электронный ресурс]. – Электронные данные. – Режим доступа : http://neuroph.sourceforge.net

# ПРИЛОЖЕНИЕ А

## Диаграмма классов



# ПРИЛОЖЕНИЕ Б

## Блок-схема 1



## Блок-схема 2



## Блок-схема 3

