Санкт-Петербургский государственный университет
Факультет прикладной математики - процессов управления
Направление: Процессы управления
Математическое моделирование в задачах естествознания
Кафедра информационных систем

Методика применения нейронных сетей для оптимизации топологии нейронной сети

Выпускная квалификационная работа
студента 2 курса магистратуры
очной формы обучения
Подхватова Никиты Владимировича

Научный руководитель:
Доктор физико-математических наук
Матросов Александр Васильевич

Санкт-Петербург
2019

Введение

Люди часто в обращаются к природе в процессе поиска решения той или иной задачи. Схема строения самолета и принципы его работы были переняты у птиц. Основываясь на опыте природы людям удалось разработать современные средства передвижения без которых трудно представить путешествия. Так же произошло и с самым сложным органом человеческого организма. Мозг человека является универсальным вычислительным устройством, способным приспосабливаться и подстраиваться под окружающую обстановку. Эти качества позволяют ему решать различные по сложности задачи в совершенно не связанных областях.

В середине прошлого века исследования ученых из разных областей биологии легли в основу систем моделирования деятельности головного мозга, на базе которых была построена теория искусственных нейронных сетей. Они дали начало развитию огромному количеству технологий и научных областей [1].

Сначала была разработана модель отдельного нейрона, моделирующего работу нервной сети на низком уровне. Однако такая система могла решать лишь ограниченный спектр задач. В мозгу биологические нейроны объединяются в группы и связываются с огромными количествами себе подобных. Возможности таких систем оказались намного шире и они были приспособлены к решению задач на которые ранее требовалось много усилий. Это распознавание, кластеризация и принятие решений. Кроме того искусственные нейронные сети обладают устойчивостью к зашумлению входных данных. Это означает, что незначительные изменения входных данных приводят к незначительным отклонениям в результатах. В то время как в при применении стандартных алгоритмов, последствия могут быть непредсказуемыми [2-6].

Другой не менее удивительной особенностью нейронных сетей является их способность приспосабливаться к изменениям среды в которой они работают. При изменении входных данных они могут обучиться, подстроится, и продолжить выдавать корректный результат.

Особенно сильно интерес к ИНС проявился в последние годы. Это связано с развитием электронных вычислительных устройств, позволяющим ускорить обработку данных и сократить время необходимое для обучения сети.

Сфера применения нейронных сетей постоянно расширяется. Цифровые ассистенты в больницах помогают врачам при анализе симптомов и определении диагноза, на производстве внедряются системы способные проводить анализ качества продукции, набирают популярность автомобили оснащенные автопилотом для движения без участия водителя. Все эти примеры включают в себя применение нейронных сетей для обработки данных.

На данный момент нет других подходов к решению подобных задач, дающих столь впечатляющие результаты. Вероятнее всего развитие ИНС будет продолжаться. Постоянно появляются новые подходы к обучению и применению сетей. Все больше потребительских устройств поддерживают реализации функций необходимых для работы ИНС на аппаратном уровне, что позволяет повысить их производительность еще больше. Не говоря уже о специализированном оборудовании.

Глава 1. Общие концепции искусственных нейронных сетей.

§1.1 Искусственная нейронная сеть

Иску́сственная нейро́нная се́ть (ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная 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 У. Маккалока и У. Питтса [7].

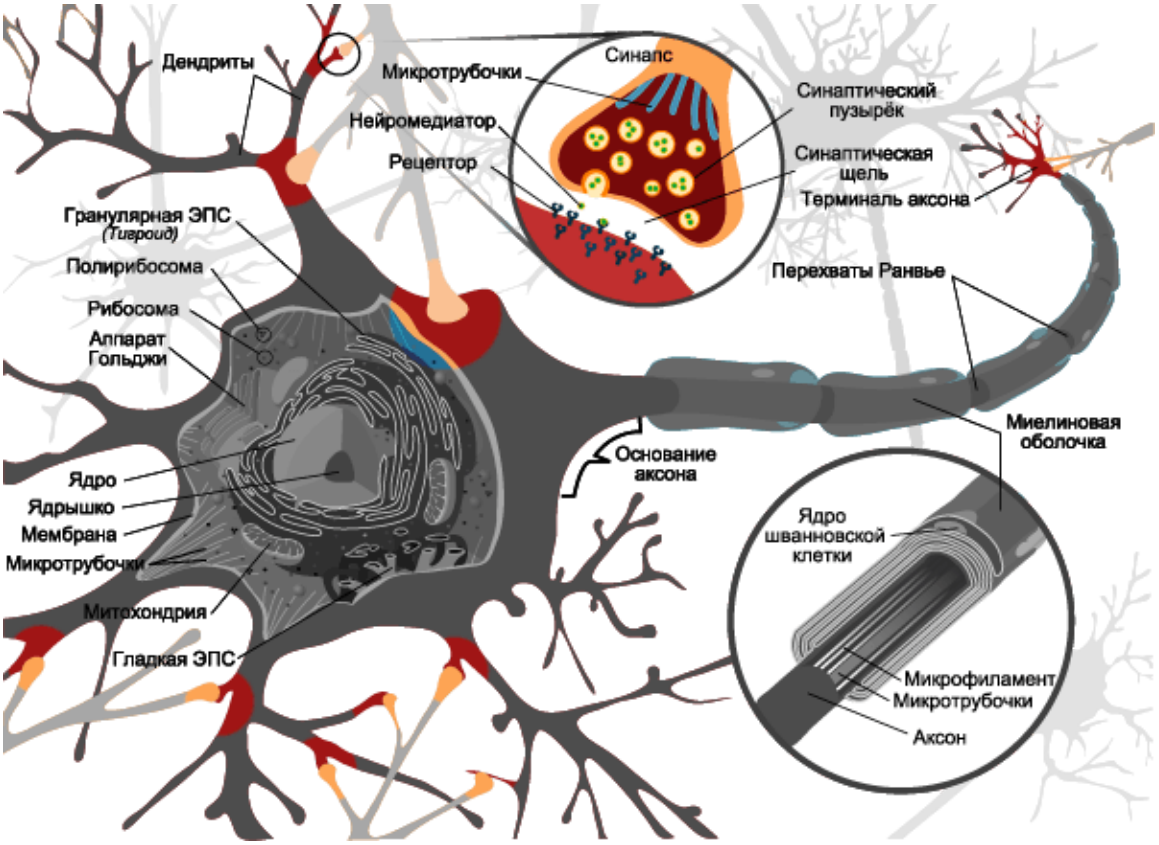
После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

Основная работа сети заключена в обработке огромных массивов данных с целью получения новой информации на их основе. Именно это выделяет ИНС на фоне других алгоритмов, решающих аналогичные задачи. Возможность получать новые данные - “делать” выводы, открывает широкие горизонты для этой области [8].

Различные подходы оптимизируют этот процесс для определенных ситуаций, но в целом все ИНС стремятся решить сходную задачу [9].

Рассмотрим структуру сети более подробно.

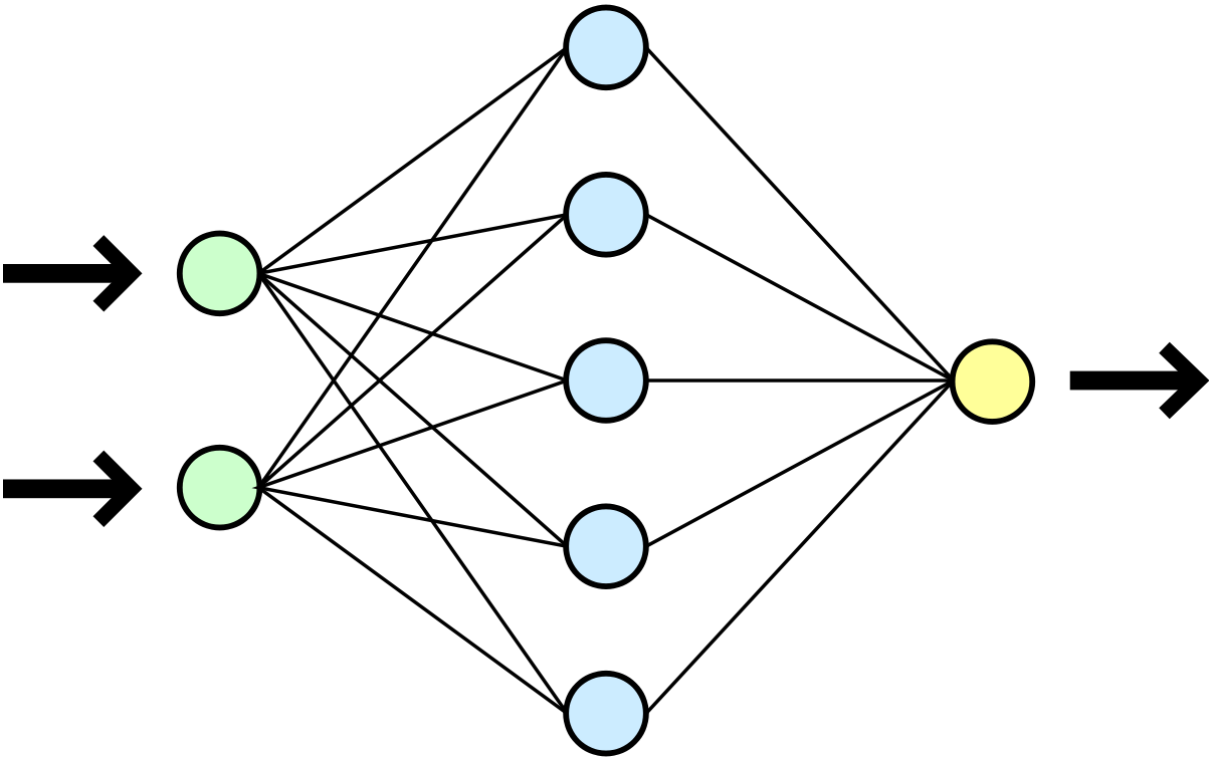
Наш головной мозг состоит из миллиардов элементов называемых нейронами. Приближенно можно сказать, что они занимаются передачей и преобразованием сигналов поступающих извне. Каждый из них представляет собой независимую структуру, связанную с другими посредствам особых соединений (аксонов и дендритов), которые передают сигнал между отдельными нейронами. Попав в очередной нейрон сигнал проходит некоторое преобразование и отправляется дальше [10].



По этому же принципу построены ИНС. Основополагающим элементом в них является искусственный нейрон. Получая информацию от других нейронов он аккумулирует ее и пропускает через так называемую “активационную” функцию. Связанные нейроны могут отличаться по “важности”, поэтому сигналы поступающие от них берутся с некоторым весом. Чем больше вес, тем выше влияние этого нейрона на результирующий сигнал.

Объединяя группу нейронов мы получим слой. Все элементы в нем могут быть связаны лишь логически. Но классическим вариантом является случай, когда нейроны в слое связаны с некоторой другой группой.

Каждый слой выполняет преобразование над входящим в него сигналом и тоже передает его дальше. Таким образом пропустив входные данные через несколько слоев мы получаем некоторое сопоставления входных и выходных сигналов.



Каждый нейрон принимает информацию от предыдущего слоя и преобразует ее в некоторый признак. Так происходит выделение новой информации. На следующем уровне нейрон принимает уже группу таких данных и преобразует ее в признак более высокого уровня. Так при распознавание изображений происходит выделение элементов. Например лицо состоит из глаз, носа и рта, каждый из которых в свою очередь состоит из более мелких деталей, в самом простом представлении состоящих из линий и точек.

Вернемся к вышеупомянутой активационной функции. Каждый нейрон, пропуская через себя входной сигнал, передает его активационной функции для преобразования. Чтобы получить из исходных данных какую-либо новую информацию недостаточно линейной комбинации уже имеющихся данных, поэтому большинство активационных функций имеют нелинейную природу. Это позволяет производить из имеющихся низкоуровневых данных получить более структурно сложную информацию. Например по набору пикселей на картинке определить наличие на ней какого-то объекта [11].

§1.2 Основные задачи, решаемые искусственными нейронными сетями

ИНС можно применять для очень широкого спектра задач. Можно даже обучить их тривиальным вещам вроде сложения или умножения. Однако наибольшую пользу они приносят в следующих областях:

- Распознавание образов
- Принятие решений
- Кластеризация данных
- Прогнозирование
- Аппроксимация
- Сжатие данных
- Анализ данных
- Оптимизация

В этих сферах ИНС показали свою наибольшую эффективность. Подобные задачи решались и традиционными средствами, но получаемые решения были очень сложны, вследствие чего их сложно развивать, применять в практических ситуациях и поддерживать. Не говоря уже о том, что они требовательны к ресурсам и времени, а также тяжело переносили даже незначительные изменения во входных данных. Современные ИНС позволяют избежать этих недостатков [12-17].

§1.3 Возможности повторного использования существующих сетей

Полученную в результате решения некоторой задачи нейронную сеть, можно применять в последствии, не только по ее прямому назначению. Полученную структуру можно использовать при разработке другой ИНС решающей схожую задачу. Если же отличия заключаются только в возможных вариантах ответов, то можно заменить лишь последний слой нейронов. Такой подход позволяет значительно сократить время необходимое для обучения.

Его можно применять, например, для задач распознавания. В группах близких объектов используются схожие характеристики. Например, таким образом нейронную сеть обученную распознавать на изображениях собак, можно быстро заставить искать котов.

Таким образом можно переиспользовать уже существующую сеть для ускорения процесса обучения. Однако вопрос оптимизации обучения первой сети остается открытым.

§1.4 Основные методы проектирования сетей

Для работоспособности нейронной сети ключевым фактором является ее структура. Количество слоев и нейронов в каждом из них могут кардинально повлиять на результат обучения. А внутренние параметры нейрона изменяют поведение сети во время процесса обучения.

Первые нейронные сети появились более 50 лет назад. Тогда их структуру определяли практически на глаз, методом проб и ошибок получая экземпляр сети, который давал приемлемый результат. Так как вычислительные ресурсы были сильно ограничены, то и результаты таких экспериментов были далеки от совершенства. Несмотря на большой промежуток времени прошедший с тех пор и весь технологический и научный прогресс, до сих пор метод проб и ошибок остается наиболее распространенным при проектировании ИНС.

Условно можно разделить процесс на 4 этапа:

1. Поиск схожих задач.
2. Определение наиболее близких по требованиям сетей.
3. Получение эффективной сети на основе рассмотренных ранее.
4. Оптимизация полученной сети и подгонка под реальную задачу.

При первичной разработке ищутся сети решающие сходные задачи или просто имеющие наибольшее количество точек соприкосновения с задачей. Эта выборка и является базой, на основе которой строится сеть.

Пробуются различные варианты структур, их комбинаций, функций активации и внутренних параметров, для получения работоспособной сети. После чего начинается подгонка и оптимизация параметров под конкретную задачу [18].

Почти все существующие сети переиспользуются в дальнейшем, как в явном виде, в других проектах или в качестве базы для дообучения, так и в качестве основы при проектировании новой ИНС. Поэтому вопрос их оптимального построения стоит еще острее. Неоптимальные затраты времени и ресурсов воздействует не только

на текущий экземпляр сети, но и на все последующие основанные на ней.

Глава 2. Постановка задачи

§2.1 Постановка задачи

В данной работе рассматривается возможность построения модели поведения ИНС при обучении при помощи другой ИНС.

Для упрощения общей модели в рассмотрение берутся только следующие входные параметры:

- Структура сети (количество скрытых слоев и нейронов в них)
- Внутренние параметры сети - скорость обучения и мера инертности нейрона
- Целевая ошибка (граничное условие для обучения сети)

В качестве выходных данных рассматриваются:

- Время обучения сети
- Результирующая ошибка (полученная на тестовой выборке)
- Количество проходов необходимое для достижения целевой ошибки

Целью работы является :

Выявление зависимостей между структурой сети и ее поведением в процессе обучения, разработка программного продукта для проведения экспериментов и анализа полученных зависимостей. А также поиск начальных параметров сети оптимизирующих ее работу по заданному критерию.

Глава 3. Разработка программной части

§3.1 Разработка программной части

Для упрощения работы с нейронными сетями был разработан комплекс программных объектов. Они решали три типа задач: управление сетями, отображение данных и решение задачи оптимизации. Автоматизация процессов конструирования, конфигурации и обучения сети способствует уменьшению вероятности ошибки и сокращению временных затрат.

Несмотря на простую структуру многослойного персептрона, любая нейронная сеть является объектом с большим количеством внутренних взаимосвязей. Даже небольшое изменение в коде, может оказать существенное, а самое главное неявное, влияние на результат работы. Это требует усиленного внимания ко всем, в том числе, самым маленьким элементам программы.

Для снижения рисков возникновения ошибок и некорректного поведения, архитектура программы была основана на принципах объектно-ориентированного программирования. Все элементы разрабатывались, как отдельные, заменяемые модули. Это позволило проводить независимое тестирование [19].

Так как для проведения исследования выбрана достаточно простая структуру сети, использование современных библиотек было признано избыточным. Их подготовка и настройка требует не только опыта работы с конкретной реализацией, но и дополнительных затрат ресурсов. А большая часть предоставляемого ими функционала, не требовалась.

Кроме того, самостоятельная реализация ИНС помогает лучше понять процессы происходящие внутри сети. А также модернизировать сеть без каких-либо ограничений.

Поэтому в работе использовалась собственная реализация многослойного персептрона. Был создан полный стек классов от нейронов и связей до слоев и нейронных сетей. Каждый из объектов решает свою конкретную задачу.

Нейроны являются основными элементами организующими преобразование внутри сети. В то время как слои необходимы для

логического разграничения групп нейронов и построения связей между ними.

Кроме объектов, непосредственно реализующих работу сети, была разработана группа вспомогательных классов. Для управления генерацией сетей, сбора данных и их преобразования. Небольшая группа модулей, позволяющая производить базовые операции с нейронными сетям: создание, конфигурацию, обучение, а также собирать информацию об их состоянии и процессе обучения.

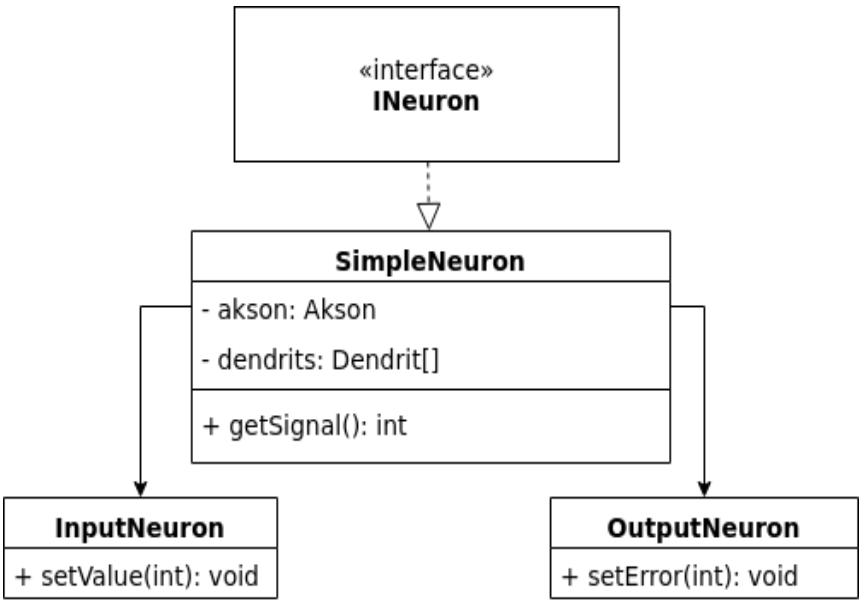
При необходимости можно использовать любую стороннюю реализацию искусственной нейронной сети. Это еще один плюс применения методов объектно-ориентированного программирования. Главное соблюдать формат данных, используемых для анализа.

Основной упор при разработке архитектуры ставился на наглядность процессов распространения сигнала внутри сети для упрощения последующего анализа, и возможность несколькими простыми вызовами сконфигурировать многослойный персептрон заданной структуры.

Для разработки программного продукта использовался язык C++ и библиотека Qt, с помощью которой создавались пользовательский интерфейс для управления программой и наглядного отображения данных.

§3.2 Структура искусственного нейрона

Искусственный нейрон является базовым элементом сети. Все нейроны могут принимать сигнал от других преобразовывать и передавать к следующим. Однако в современных сетях этот функционал может быть гораздо шире, или отличаться по поведению. Даже в самом простом персептроне элементы входного и выходного слоя, имеют особенности поведения, для передачи сигнала в сеть, его извлечения и обучения. Поэтому структура типов нейронов реализована на основе принципа наследования. Все классы реализуют общий интерфейс описывающий базовое взаимодействие.



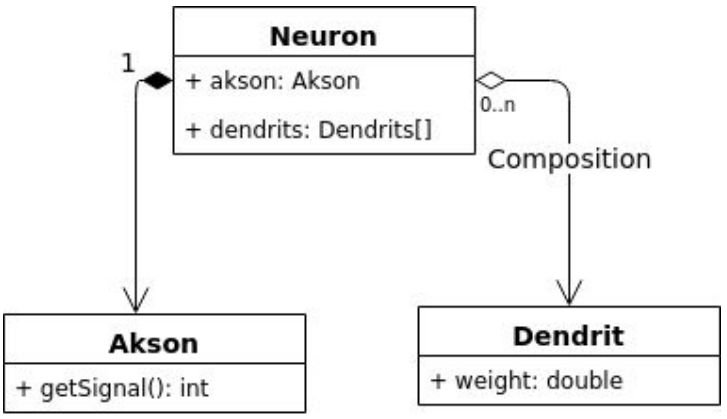
В объектах нейронных сетей, слоев и функциях обращение к нейронам происходит по указателю на базовый класс. Это позволяет расширять структуру нейронов для реализации более сложных сетей.

Кроме стандартных взаимодействия нейроны реализуют логику для изменения весов при обучении.

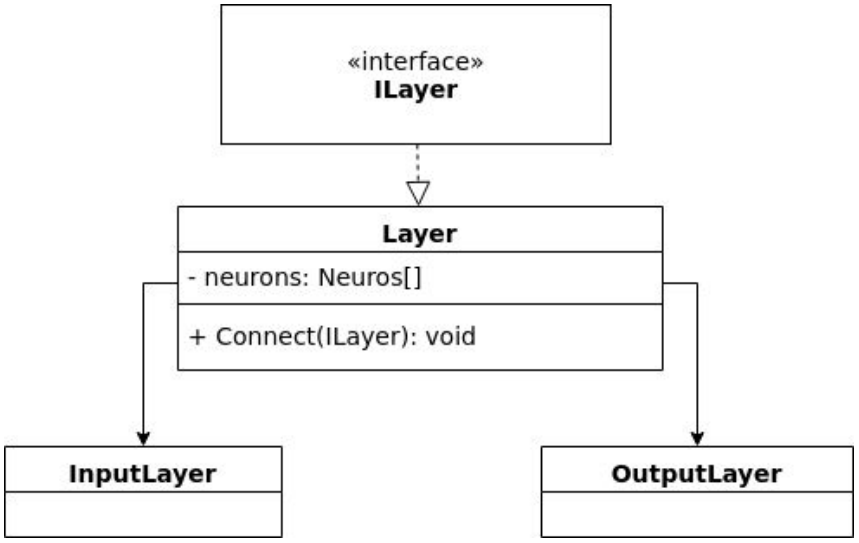
Реализация искусственного нейрона на высоком уровне представляет собой 3 объекта:

1. Аксон - класс, реализующий логику передачи сигнал из нейрона наружу. Он определяет функцию активации и является звеном в связи между нейронами
2. Дендриты - элементы, описывающие логику приема сигнала другим нейроном. В каждом объекте содержится информация о весе и присоединенном Аксоне, а также информация для проведения процесса обучения.
3. Ядро нейрона - отвечает за логику сбора данных с Дендритов, ее суммирования и передачи в Аксон. Оно также содержит управляющие методы для проведения обучения, и установки/получения входных/выходных значений.

Основные элементы были реализованы с применение интерфейсов, таким образом возможно добавление более сложных реализаций без необходимости полной перестройки программы.



Для упрощения взаимодействия группы нейронов объединяются в рамках специальных объектов, называемых слоями. Они реализуют интерфейс для создания связей между собой. Слои повторяют структуру типов нейронов.



§3.3 Реализация сети

Для упрощения создания объектов сетей, а также поддержания взаимозаменяемости, все сети реализуют общий интерфейс, содержащий методы для генерации структуры, ввода и вывода данных, и для обучения.

Конфигурация внутренней структуры передается вспомогательным объектам, представляющим слои. Разработано три типа слоев: входной, выходной и скрытый, в соответствии с общей структурой ИНС. Каждый тип представлен своим классом наследуемым от базового слоя. Они берут на себя ответственность за реализацию конкретных методов, в то время как объект нейронной сети выступает в качестве управляющего узла. Благодаря возможностям

полиморфизма ему нет необходимости знать подробности реализаций этих слоев, он лишь выполняет обязанности дирижера - создает эти объекты и передает им запросы.

Описанная выше структура применялась для создания и обучения сетей в течение всего проведенного исследования. Для повышения надежности и уменьшения ошибок при разработке самые часто используемые элементы проверялись с применение методов Unit-тестирования.

§3.4 Обучение методом обратного распространения ошибки.

Процесс обучения нейронной сети заключается в изменении весов в связях между нейронами. Целью этого процесса является минимизация ошибки - отклонения результата работы сети от эталонного. Для количественной оценки этого значения вводится соответствующая функция, например, сумма квадратов расстояний:

$$E = \frac{1}{2}\sum (Out - Out^*)^2$$

Out - значения нейронов выходного слоя, Out^* - соответствующие эталонные значения.

Основной идеей метода является вычисление градиента, который используется при обновлении весов.

Основным требованием для применения этого метода, является наличие производной у активационной функции. Следствием того, что этот метод является базовым для проведения обучения, активационные функции выбираются таким образом, чтобы удовлетворять этому условию.

В реализации стохастического градиентного спуска к текущему весу добавляется значение:

$$\Delta\omega_{i,j} = -\eta\frac{\partial E}{\partial\omega_{i,j}}$$

Рассмотрим вычисление производной более подробно. Движение начинается с выходного слоя сети, таким образом j - индекс нейрона на нем.

Но $\omega_{i,j}$ не влияет на выходное значение напрямую, а только через $S_j = \sum_i \omega_{i,j} x_i$, поэтому производную распишем как:

$$\frac{\partial E}{\partial \omega_{i,j}} = \frac{\partial E}{\partial S_j}\frac{\partial S_j}{\partial \omega_{i,j}}$$

В свою очередь S_j влияет на общую ошибку, как выход j-го узла, соответственно:

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial S_j}$$

После выполнения подстановки функции ошибки и функции активации:

$$\frac{\partial E}{\partial \omega_{i,j}} = -2\alpha o_j(1-o_j)(t_j - o_j)$$

Если же j-ый элемент не на последнем уровне, то:

$$\frac{\partial E}{\partial S_j} = \sum_k \frac{\partial E}{\partial S_k}\frac{\partial S_k}{\partial S_j},$$

где k - индекс выхода связанного с нейроном на следующем слое.

$$\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j}\frac{\partial o_j}{\partial S_j} = \omega_{j,k}\frac{\partial o_j}{\partial S_j} = 2\alpha o_j(1-o_j)\omega_{j,k}$$

а $\frac{\partial E}{\partial S_k}$ это поправка вычисленная на предыдущем шаге. Обозначим ее как δ_k.

Таким образом для последнего слоя:

$\delta_j = -2\alpha o_j(1-o_j)(t_j - o_j)$

Для внутреннего узла сети:

$$\delta_j = 2\alpha o_j(1-o_j)\sum_{k \in Children(j)} \delta_k \omega_{j,k}$$

Для всех узлов:

$\Delta\omega_{i,j} = -\eta\delta_j o_i$

На основе описанного алгоритма была реализована схема обучения сети с применение алгоритма обратного распространения ошибки. Объект, реализующий логику работы сети, имеет метод автоматизирующий процесс обучения. Применялись два типа подходов: стохастический - обновление весов происходит на каждый тренировочный пример и пакетный - обновление весов происходит по результатам нескольких тренировочных примеров.

При стохастическом обучении сеть вычисляет ошибку для каждого полученного примера и передает ее в последний слой. Каждый нейрон на основе полученного значения вычисляет значение для обновления веса. Затем передает его всем элементам следующего слоя, с

которыми связан. Так происходит распространения до первого слоя. После чего в таком же порядке происходит обновления весов связей, с учетом значения, на которое нужно обновить, и изменения с прошлого прохода.

Глава 4. Производящая сеть

§4.1 Структура

В работе рассматривалась система из двух нейронных сетей. Одна из них представляет собой производящий объект, генерирующий данные. Эта сеть решает некоторую абстрактную задачу, кластеризационную или аппроксимационную, или любую другую из эффективно решаемых при помощи ИНС задач. Тип задачи не имеет критической важности для проведения последующих экспериментов.

В качестве тестовой рассматривается аппроксимация некоторой функции. Суть заключается в сопоставлении вектора входных параметров функции $(x_1, x_2, ..., x_n)$ и вектора значений функции $(y_1, y_2, ..., y_n)$ [20].

Для этой задачи просто генерировать данные для обучения и не требуется специальных подготовительных мероприятий (преобразование форматов входных и выходных данных и другое). Кроме того можно манипулировать сложностью задачи.

Основная цель этого этапа заключается в эмулировании работы некоторой сети и сборе данных.

§4.2 Подготовка тестовых данных

Для аппроксимации были выбраны две конкретные функции. Они отличаются как по поведению, так и по сложности. Первая - логическая функция от n переменных. Работа функции эмулировалась путем построения таблицы соответствия входных и выходных данных.

a_1	a_2	…	a_n	b
0	0	...	0	0
0	0	…	1	0
0	1	...	0	1
...				
1	1		1	1

Выходные значения генерировались случайным образом. Для повышения вероятности сходимости и уменьшения количества шумов в данных, соотношение 0 и 1 было примерно 50/50 [21].

Второй задачей, на основе которой генерировались данные, была аппроксимация синуса на некотором отрезке. Отрезок разбивался на n частей, значения в соответствующих узлах использовались для обучения.

Таким образом подготавливаются данные для обучения “Производящей” сети.

После подготовки данных необходимо произвести обучение сети и записать показания регистрируемых метрик. В список интересующих нас данных включаются количество нейронов в скрытом слое, внутренние параметры нейронов: коэффициент скорости обучения и коэффициент инерции и “целевая” ошибка - значение ошибки на обучаемом множестве, при достижении которой процесс обучения заканчивается.

По окончании процесса обучения фиксируются параметры описывающие поведение сети. Такие как время обучения, количество циклов обучения и ошибка полученная на тестовом множестве.

Для создания обучающего множества необходимо выполнить несколько прогонов обучения с различными начальными параметрами сети. Для автоматизации этого процесса был разработан модуль проведения экспериментов (в дальнейшем объекты соответствующего класса будем называть “экспериментатор” для краткости). При создании экземпляра “экспериментатора”, для каждого параметра

указываются промежуток на котором будет проводится исследование и шаг. После чего в автоматическом режиме создаются экземпляры сетей с соответствующими конфигурациями и производится их обучение. В результате получается множество входных и выходных параметров.

Глава 5. Вторая сеть

§5.1 Структура

После получения данных от “Производящей” сети необходимо обучить на основе них “Аппроксимирующую” сеть. Ее задачей является максимально точно отобразить множество параметров структуры сети в множество метрик описывающих процесс обучения. Это необходимо для проведения последующего анализа зависимостей, и определения возможностей управления.

Любой процесс можно представить, как модель “Черного ящика”, принимающего на вход некоторые данные и выдающего результат. Будь то биологическая система, какое-либо техническое устройство или некая информационная система. Таким же образом построена и нейронная сеть, следовательно можно использовать ее, в качестве моделируемого объекта [22-23].

Основной принцип заключается в замене процесса полного переобучения сети, использования приближенного значения полученного от обученной сети.

В качестве “Аппроксимирующей” сети также рассматривается многослойный персептрон. Задача проектирования для этой сети решается обычными методами. Однако предполагается, что структура этой сети заметно проще, чем исследуемой (“Производящей”). Таким образом ручная проверка возможных конфигураций, не занимает много времени. А кроме того сеть может быть дообучена и использована для класса подобных “Производящей”. За счет этого можно не только получить более оптимально сконфигурированную сеть, но и сократить затраты ресурсов.

Скрытые слои сети дополнены нейронами сдвига, для улучшения сходимости. Эксперименты проводились для одного и двух скрытых слоев.

В качестве активационной функции использовалась сигмоида.

§5.2 Анализ зависимостей

После создания данных о процессе обучения “Производящей” сети, их необходимо было подготовить для для использования в качестве

обучающей выборки. В первую очередь все данные необходимо было нормировать. Так как разные показатели лежат в совершенно разных пределах, они могут искажать результаты обучения. Для исключения подобного влияния все величины отображаются в промежутки от 0 до 1.

Для проведения такого преобразования был реализован алгоритм определяющий наибольшее и наименьшее значение во всем массиве данных и производящий сдвиг следующим образом:

$$newValue = ((currentValue - minValue) / (maxValue - minValue)$$

Таким образом были преобразованы все входящие и исходящие массивы данных.

Для наглядности рассматриваемые графики зависимости параметров представлены в парах. По одному входящему и выходящему.

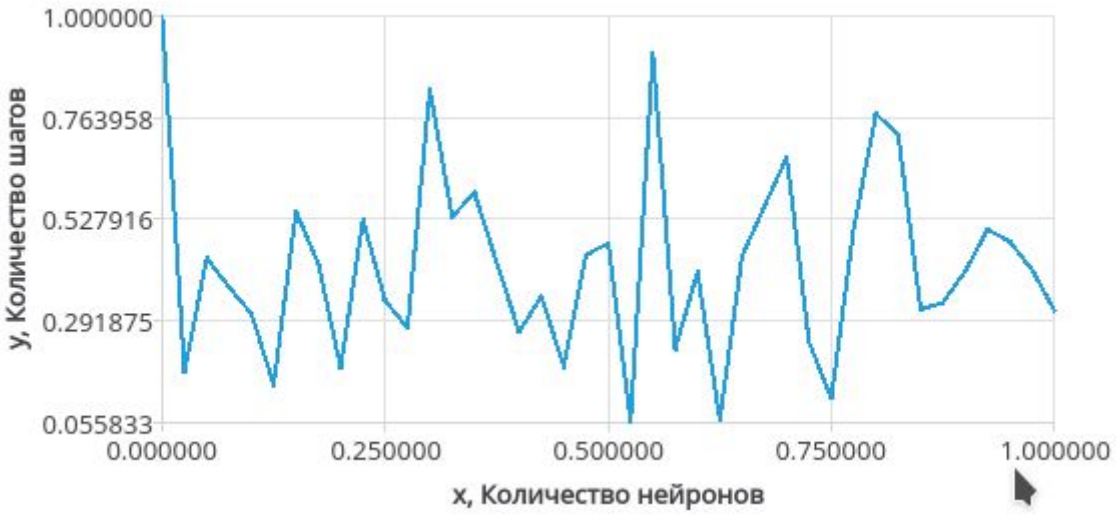
Основной параметр определяющий структуру сети - количество слоев и количество нейронов в них. Именно эти величины определяют будет ли ИНС работоспособна в принципе. Для определения наличия зависимостей между этими параметрами и процессом обучения был проведен ряд экспериментов. “Производящая” сеть обучалась с различным числом нейронов в скрытом слое, и полученные выборки передавались во вторую сеть.

На следующих графиках в наглядной форме отображены зависимости начальных параметров сети и характеристик обучения.

Для пары количество нейронов и время обучения, наблюдается явная прямая зависимость. Это обусловлено тем, что на каждый

дополнительный нейрон приходится выполнять комплекс действий для обновления его весов.

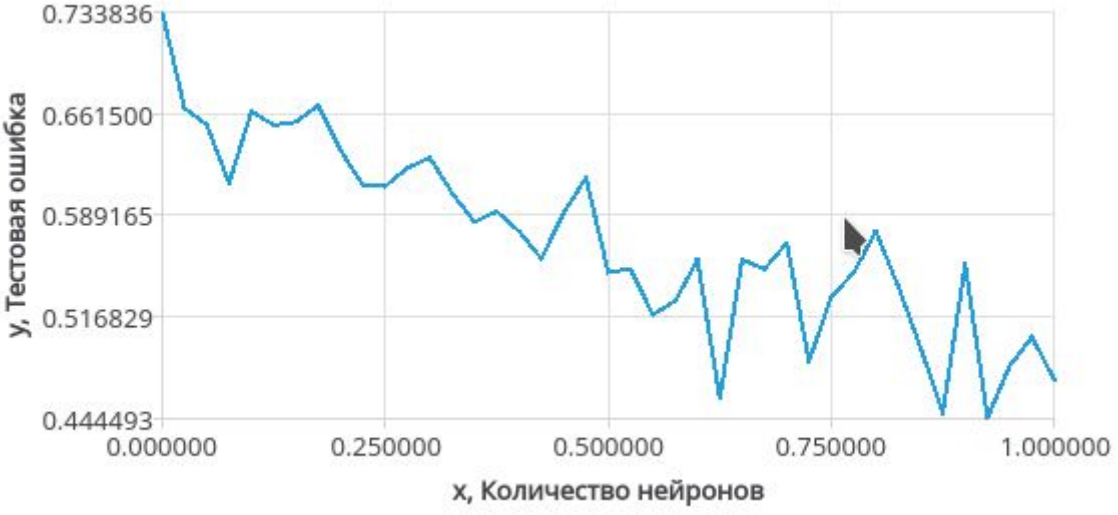
Следующей парой параметров является количество нейронов и количество циклов обучения.



Данные получились слишком зашумленными. Это связано с начальной инициализацией сети. При создании все значения весов устанавливаются случайными значениями, что приводит к сильному разбросу количества циклов необходимых для схождения сети.

Также это указывает на низкую взаимосвязь количества этапов и количества нейронов.

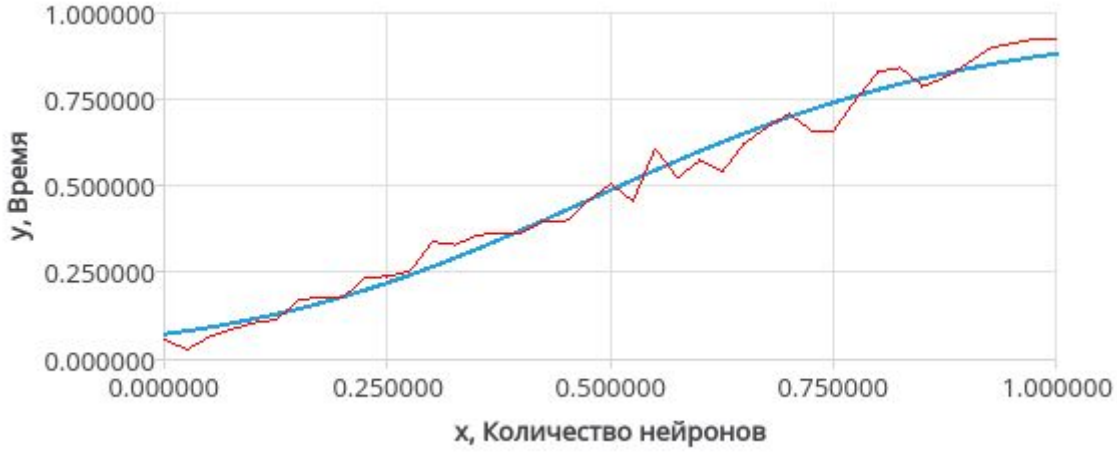
На рисунке ниже отображена зависимость ошибки на тестовом множестве от количества нейронов в скрытом слое.



Наблюдается обратная зависимость с небольшими возмущениями. Для этой зависимости ошибка сети находилась в пределах 9-10%,

из-за значительных колебаний значений. Но даже при таких сильных отклонениях в двух из трех метриках, это не отразилось на первой, что позволяет не зависеть от конкретных типов собираемых данных. Можно сделать вывод, что даже если между какими-то парами нет сильной взаимосвязи, это не помешает обработке других зависимостей.

Первым этапом в применении ИНС было обучение на полученных данных. Основная функция сети после обучения заключается в соотношении входных параметров сети и характеристик обучения.

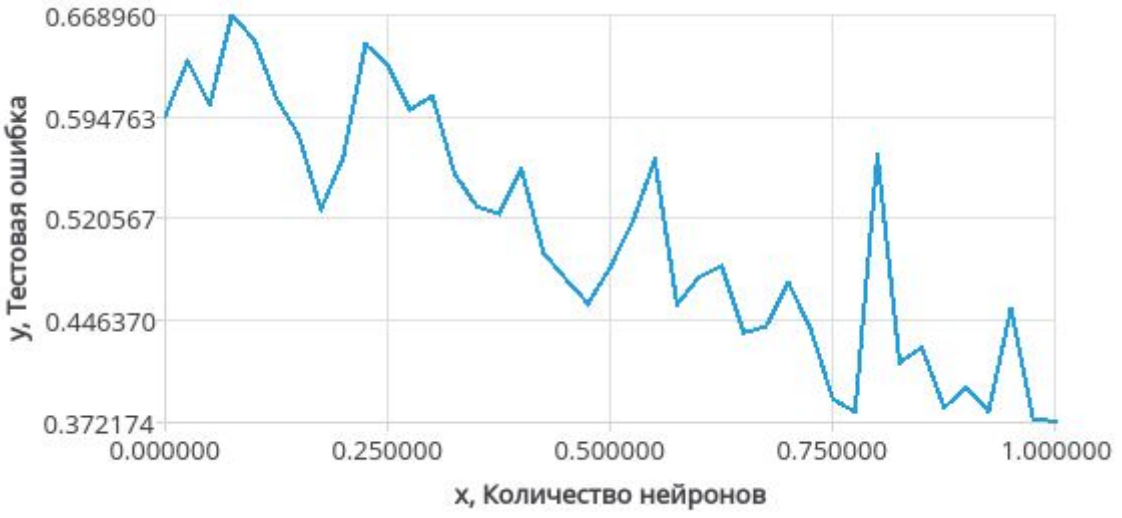


Нейронной сети удалось аппроксимировать зависимость времени обучения от количества нейронов с абсолютной ошибкой в 3%.

Для уменьшения влияния шумов возникающих при генерации сетей было проведено усреднение. Сбор данных был повторен суммарно 5 раз. А в качестве входных и выходных значений были использованы их средние арифметические.

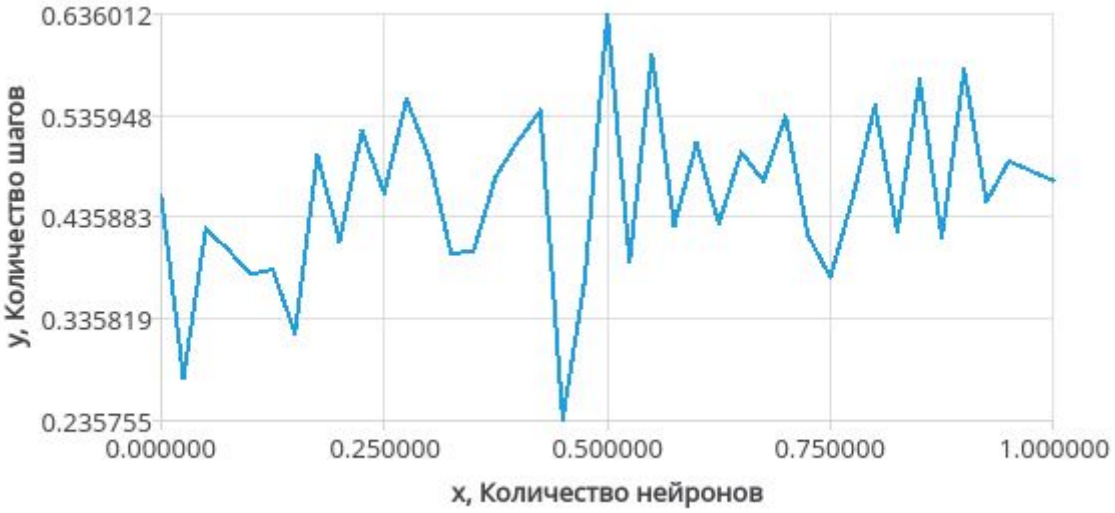
Однако это имело неоднозначный эффект. Например график зависимости времени обучения от количества нейронов стал более гладким.

Что позволило более точно определять данный параметр. А график зависимости ошибки от количества нейронов после усреднения стал более зашумленным.



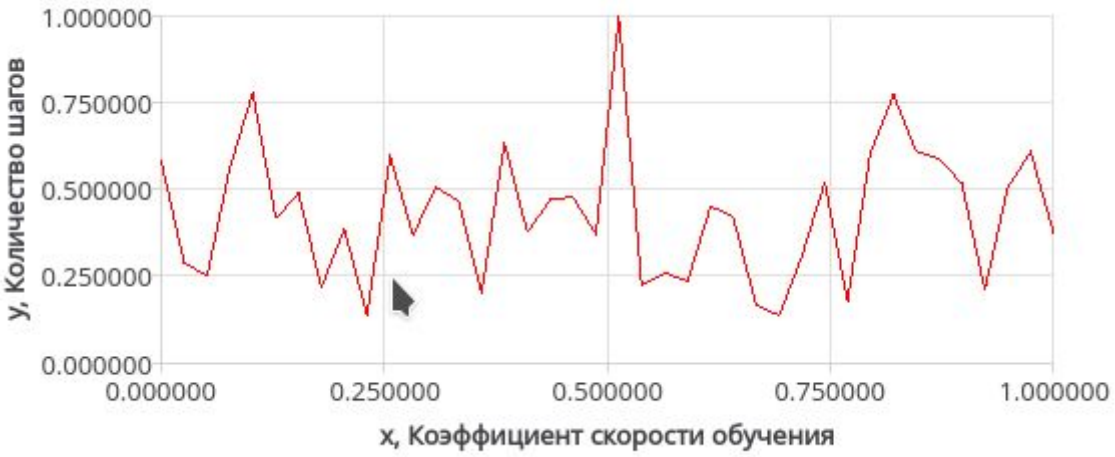
Это говорит о нестабильной зависимости ошибки, и сильном воздействии начальной инициализации сети, приводящем к большим скачкам.

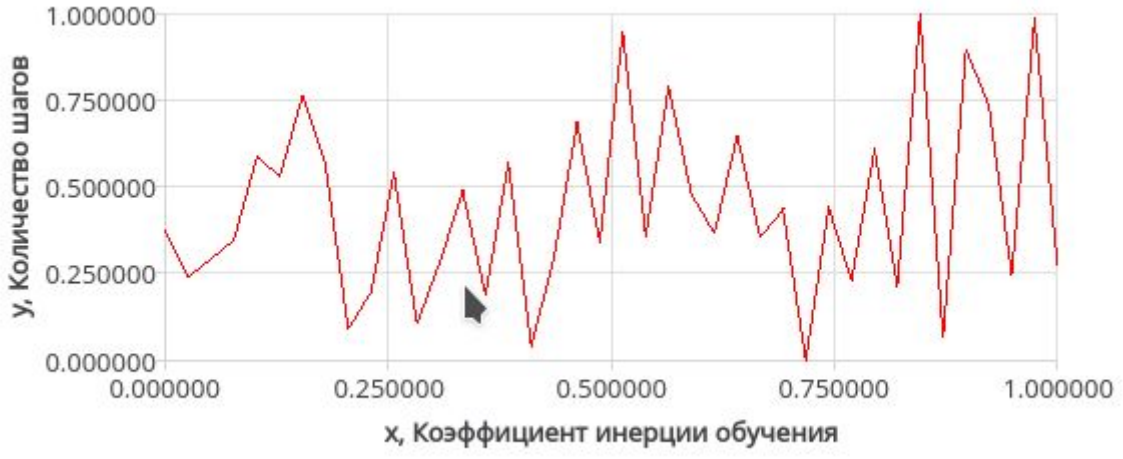
В графике зависимости количества шагов от количества нейронов стала просматриваться тенденция к росту. Но количество всплесков и их амплитуда изменились слабо.



Можно сделать вывод что небольшое усреднение почти не повлияло на характер зависимостей, но потребовало кратного увеличения времени моделировани. Поэтому в дальнейшем использовались данные одного прохода обучения.

Самые нестабильные показатели связаны с количеством шагов. Ни для скорости обучения ни для инерции не наблюдается явно выраженной зависимости.





За счет нормализации данных графики выглядят сильно возмущенными, однако абсолютные колебания не превосходят 1-2%. Это позволяет аппроксимировать значения невысокой погрешностью.

На ошибку эти показатели, как и ожидалось влияния не оказывают. Это связано с тем, что они изменяют скорость движения вдоль градиента. Однако несмотря на это они могут вытолкнуть нейронную сеть в область локального минимума, где она застрянет. При сборе данных наблюдался ряд подобных случаев при манипулировании именно с параметрами скорости и инертности обучения.

Полученные данные можно применить для локализации областей приводящих к некорректному локальному минимуму. Для последующего исключения их из области допустимых значений.

§5.3 Определение оптимальных параметров

Поиск зависимостей в процессе обучения сети, конечно, важен для понимания процесса, но почти не несет практической пользы. Основной его целью является последующее использование в процессе поиска оптимальных параметров.

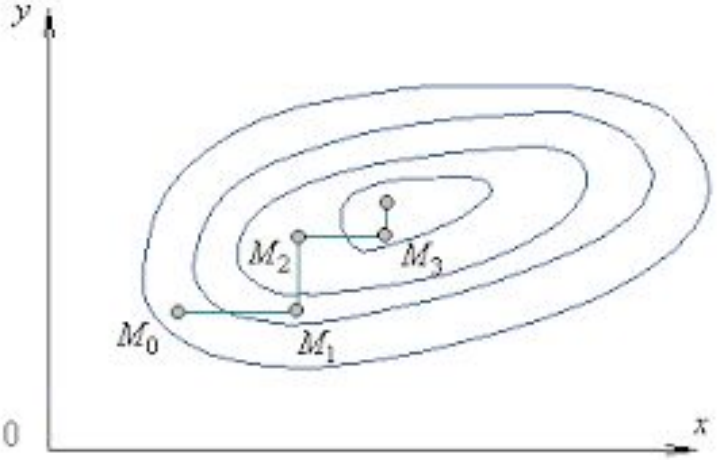
Оптимальность в данном случае подразумевает минимизацию значения некоторой функции. Она задается в зависимости от требований и ограничений накладываемых на решаемую задачу [24].

$$f(x_1, x_2, ..., x_n);$$

Например, если требуется лишь минимизировать время обучения, то достаточно использовать следующую функцию:

$$f(x_1, x_2, ..., x_n) = t;$$

Для определения минимума функции многих переменных использовался метод основанный на покоординатном спуске. Так как значения вычисляются при помощи, нейронной сети, то получить производную в явном виде не получится [25-27].



Вычисление зависимостей происходили с некоторой погрешностью, поэтому для поиска оптимальных значений достаточно локализовать некоторую область для каждого параметра, в рамках которой будет найдено требуемое значение.

Алгоритм состоит из следующих шагов:

1. Выбрать начальное приближение - любая точка внутри области ограниченной исследуемыми отрезками.
2. Зафиксировать все координаты, кроме одной.
3. Разбить отрезок на n точек и провести вычисление значений в каждой из них.
4. Найти минимальное значение и зафиксировать соответствующую координату.

5. Перейти к следующей координате.
6. Продолжать пока пункты 3 - 5 пока выполняется условие остановки.

   В качестве условия остановки рассматривалось классическое условие на расстояние между текущей и предыдущей точкой.

$$|y_n - y_{n-1}| > \varepsilon$$

   Для определения меры разности двух точек использовалось квадратичное расстояние.

$$|a - b| = \sum_k (a_k - b_k)^2$$

Для уменьшения вероятности зацикливания при проверке условия остановки необходимо брать достаточно большое количество точек разбиения. Для каждого случая разбиение подбирается экспериментально, но затраты на вычисление значения функции в точке, малы. На этом основании следует брать заведомо большее число точек разбиения, чем минимально необходимо.

Для получения точного значения оптимальных параметров на основе метода покоординатного спуска выделяется область минимума. Для определения границ можно использовать соседние точки, но при недостаточном качестве аппроксимации нейронной сетью этого может быть недостаточно. Область определялась исходя из временных затрат на цикл обучений сети, необходимый для покрытия этой области.

По результатам уточненных обучений определялись оптимальные параметры, как минимум заданной функции оптимальности.

При необходимости первые два этапа поиска оптимальной области можно повторять несколько раз. На основе полученного нового множества проводилось обучение ИНС и выделялась новая область минимума. Это способствовало повышению точности определения оптимального значения параметров, но влекло за собой нелинейное увеличение времени необходимого для поиска.

Выводы

В данной работе было проведено исследование возможностей применения нейронных сетей для анализа других сетей. Разработан программный продукт, реализующий набор инструментов для построения и управления ИНС, а также позволяющий проводить анализ данных обучения и представлять их в наглядной форме. В результате:

- подтверждено наличие зависимостей между внутренними параметрами и поведением в процессе обучения.
- проанализированы полученные в результате экспериментов данные.
- определен метод поиска параметров для достижения минимума функции оптимальности.

Процесс сбора информации и выявления зависимостей удалось практически полностью автоматизировать. А вот процесс поиска оптимальных значений требует значительного участия человека.

Литература

[1][24] *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия - Телеком, 2006.

[2] *Калацкая Л.В., Новиков В.А., Садков В.С.* Организация и обучение искусственных нейронных сетей Мн.:БГУ, 2003

[3] *Хайкин С.* Нейронные сети полный курс. М., 2006.

[4] *Под ред. В.П. Боровикова.* Нейронные сети. STATISTICA Neural Networks. Методология и технология современного анализа данных. М.: Телеком.

[5] *Горбань А.Н., Россиев Д.А.* Нейронные сети на персональном компьютере. Новосибирск: Наука 1996.

[6] *Горбань А.Н.* Обучение нейронных сетей. М.: ПараГраф, 1990.

[7] *Мак-Каллок У. С., Питтс В.* Логическое исчисление идей, относящихся к нервной активности. Автоматы. Под ред. К. Э. Шеннона и Дж. Маккарти. — М.: Изд-во иностр. лит., 1956.

[8] *Загоруйко Н.Г.* Прикладные методы анализа данных и знаний. - Новосибирск. Изд. Института математики, 1999.

[9] *Максимов Ю.А., Филлиповская Е.А.* Алгоритмы решения задач нелинейного программирования. — М.: МИФИ, 1982.

[10] *Н. Винер.* Кибернетика. 2-е изд., 1961, гл. I.

[11] *Коршунов Ю.М.* Математические основы кибернетики. — М.: Энергоатомиздат, 1972.

[12] *И.В. Заенцев.* Нейронные сети: основные модели. Учебное пособие по курсу "Нейронные сети" для студентов 5 курса магистратуры к. электроники физического факультета Воронежского Государственного университета.

[13] А*.Н.Горбань, В.Л.Дунин-Барковский, А.Н.Кирдин и др.* Нейроинформатика. - Новосибирск: Наука. Сибирское предприятие РАН, 1998.

[14] *Миркес Е.М.* Учебное пособие по курсу Нейроинформатика. Красноярск - 2002.

[15] *Сотник С.Л.* Конспект лекций по курсу "Основы проектирования систем искусственного интеллекта".

[16] Комашинский В.И., Смирнов Д.А. Нейронные сети и их применения в системах управления и связи. М.: Горячая линия Телеком. 2003.

[17] *Максимов Ю.А.* Алгоритмы линейного и дискретного программирования. — М.: МИФИ, 1980.

[18] *Данилин С.Н, Макаров М.В., Щаников С.А.* Алгоритм проектирования нейронных сетей с минимальной разрядностью.

[19] *Р. Лафоре.* Объектно-ориентированное программирование в C++.

[20] *Шведов А. С.* Аппроксимация функций с помощью нейронных сетей и нечетких систем. М.: Изд. ВШЭ Проблемы управления. 2018.

[21] *Федоров С.Н.* Аппроксимация функций с помощью нейронных сетей.

[22] *Бодянский Е.В., Руденко О.Г.* Искусственные нейронные сети: архитектуры, обучение, применения. Харьков: Телетех, 2004.

[23] *Данилин С.Н.* Современное представление об информации. Информационные системы и технологии. 2012 № 4.

[24] Данилин С.Н Исследование влияния выбора показателя качества работы на результат оценки отказоустойчивости устройств с нейросетевой архитектурой. Радиотехнические и телекоммуникационные системы. 2011 Вып. 4.

[25] *Акулич И.Л.* Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов. — М.: Высш. шк., 1986.

[26] *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. Пер. с англ. — М.: Мир, 1985.

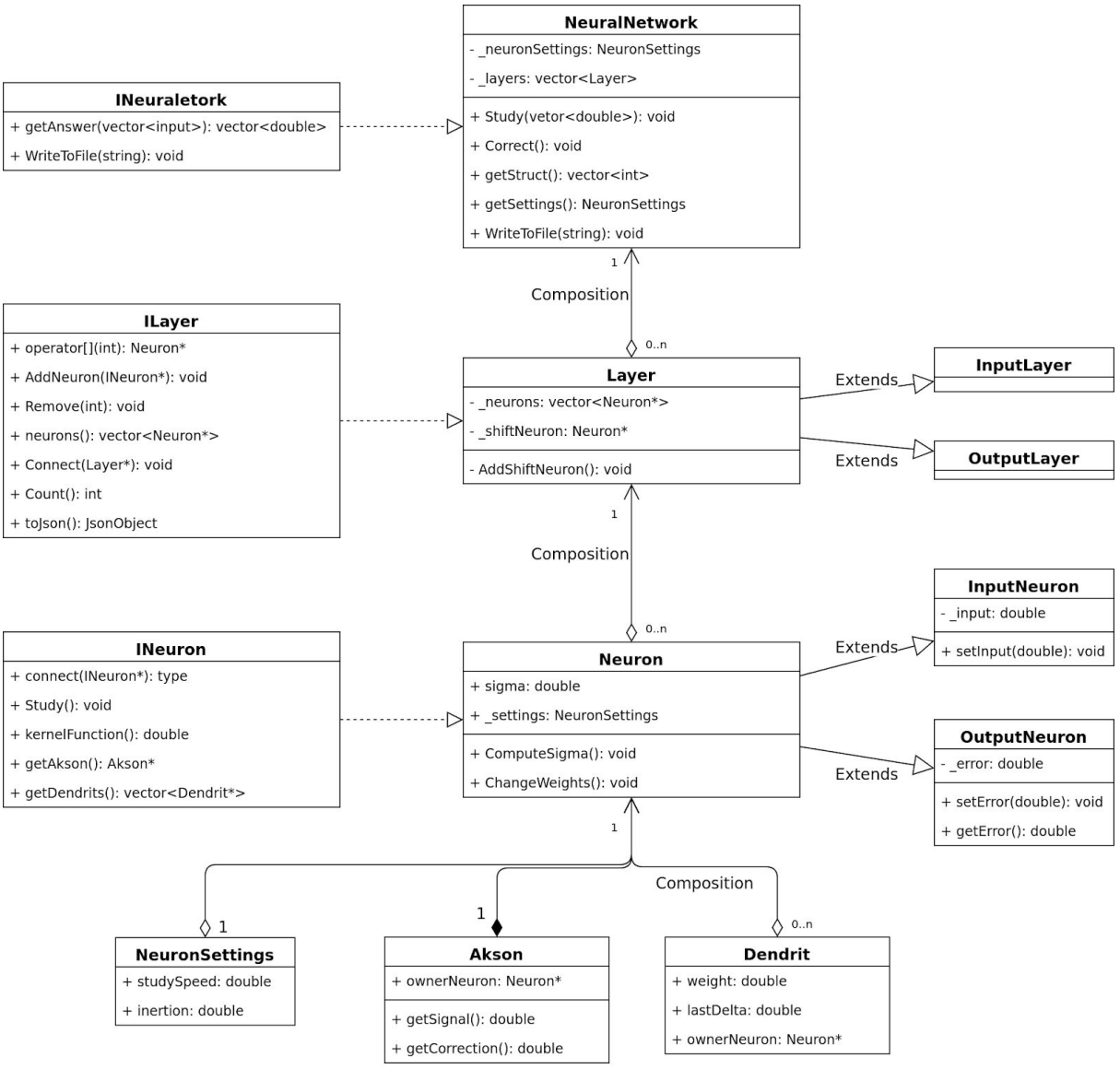
[27] *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. — М.: Наука, 1970. — С. 575-576.

[28] *Б. Уидроу, С. Стирнз.* Адаптивная обработка сигналов. М.: "Радио и связь", 1989.

[29] *Ф. Уоссермен.* Нейрокомпьютерная техника: теория и практика. М. Мир - 1992.

Приложения

Приложение А



Приложение Б

```
#include <QVector>
namespace MNN{
  class Akson;
  class Dendrit;
  class INeuron;
  class Akson{
  public:
    double getSignal();
    double getCorrection();
    INeuron* ownerNeuron;
    QVector<Dendrit*> _conectedDendrits;

  private:
    double f(const double x);
    double df(const double x);
  };
  class Dendrit{
  public:
    Dendrit(){}
    Dendrit(const double& someWeight, Akson* someAkson) : weight(someWeight),
inputAkson(someAkson){}
    double weight = (static_cast<double>(2 * qrand())/(RAND_MAX) - 1) / 10;
    double lastDelta = 0;
    Akson* inputAkson;
    INeuron* ownerNeuron;
  };

  class INeuron
  {
  public:
    INeuron();
    virtual ~INeuron();
    virtual void Connect(INeuron* other) = 0;
    virtual void Study() = 0;
    virtual double kernelFunction() = 0;
    virtual Akson* getAkson() = 0;
    virtual QVector<Dendrit*> getDendrits() = 0;
  };
}
```

Приложение В

```
#include <QVector>

#include "../FirstNeuro/Interfaces/ineuron.h"
#include <QJsonObject>
#include <QJsonArray>

namespace MNN{
  class ILayer
  {
  public:
      ILayer(){}
      virtual INeuron* operator[](const int& index) = 0;
      virtual void AddNeuron(INeuron* newNeuron) = 0;
      virtual void Remove(const int& index) = 0;
      virtual QVector<INeuron*> neurons() = 0;
      //Производит связывание с предыдущим слоем
      virtual void connect(ILayer*) = 0;
      virtual ~ILayer(){}
      virtual uint Count() = 0;
      virtual QJsonObject toJson() = 0;
  };

}
```

Приложение Г

```
#include <QVector>
#include "../FirstNeuro/Interfaces/ineuron.h"
#include "../FirstNeuro/Exceptions/notimplemented.h"
#include "../FirstNeuro/Interfaces/ilayer.h"

namespace MNN{
  class INeuralNetwork
  {
  public:
      INeuralNetwork(){}
      virtual QVector<double> getAnswer(const InputArray& input) = 0;
      virtual void WriteStructToFile(const QString& filename) = 0;
      virtual ~INeuralNetwork(){}
  };
}
```