МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

«КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ

УНИВЕРСИТЕТ им. В.П. Астафьева»

(КГПУ им. В.П. Астафьева)

Институт Информатики, математики, физики и информатики

Выпускающая кафедра Базовая кафедра информатики и информационных технологий в образовании

Полковникова Анастасия Викторовна

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Подготовка школьников к разработке и применению искусственных нейронных сетей

Направление подготовки/ специальность 44.03.05 Педагогическое образование

Направленность(профиль) образовательной программы Физика и информатика

ДОПУСКАЮ К ЗАЩИТЕ

Зав.кафедрой: Д-р.пед., профессор кафедры ИИТвО Пак Н.И.

20.06.2019

(дата, подпись)

Руководитель: к.ф.-м.н., доцент кафедры ИИТвО Шикунов С.А.

20.06.2019

(дата, подпись)

Дата защиты 22.06.2019

Обучающийся: Полковникова А.В

20.06.2019

(дата, подпись)

Оценка хорошо

(прописью)

Красноярск 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире искусственный интеллект очень прочно вошел в нашу жизнь и помогает в решении задач абсолютно различных направлений, начиная от государственной безопасности, бизнеса, маркетинговой деятельности и заканчивая возможностями смартфонов обычных людей. Искусственные нейронные сети применяются в различных областях науки: начиная от систем распознавания речи до распознавания вторичной структуры белка, классификации различных видов рака и генной инженерии. Особенно привлекают внимание приложения для смартфонов, трансформирующие изображения. Всё это делает привлекательным и полезным освоение основ разработки искусственных нейронных сетей школьниками.

Практически все передовые компании активно занимаются исследованиями и разработками искусственного интеллекта, что способствует тому, что новые открытия в этой сфере появляются чуть ли не каждый день. Наиболее перспективным направлением искусственного интеллекта являются нейронные сети. Основная идея заключается в том, чтобы создать максимально близкую модель человеческой нервной системы, то есть смоделировать основные ее способности: обучаться и исправлять ошибки.

Безусловно, изучение нейронных сетей сейчас одно из актуальных направлений развития для всех отраслей наук. Учитывая тот факт, что популярность искусственного интеллекта еще не достигла своего пика развития, мы можем предположить что еще многие годы, а может и десятилетия эта сфера будет активно изучаться. Соответственно, всегда будет оставаться потребность в большом количестве квалифицированных специалистов. К сожалению, в современных школах нет конкретного направления в изучении нейронных сетей, и лишь в профильных классах и на олимпиадах можно услышать о существовании такого перспективного направления.

Я считаю, что задача каждого учителя информатики в школе, познакомить детей с азами изучения нейронных сетей, объяснить основные принципы и положения, для того чтобы уже в школьном возрасте они понимали актуальные перспективы развития в сфере программирования и могли пойти по этому пути, еще до того как они поступят в институт.

Моя задача состоит в том, чтобы создать такое методическое пособие, которое поможет любому учителя информатики сначала изучить эту тему самому и лишь после начать учить детей самым первым азам в освоении нейронных сетей.

Цель: разработать рекомендации для школьного учителя, которые помогут ему построить курс первоначального обучения основам искусственных нейронных сетей

Объект исследования: обучение основам информатики в средней школе

Предмет исследования: обучение основам искусственных нейронных сетей

Задачи:

1. Определить объем теоретического минимума знаний для практического освоения школьниками основ искусственных нейронных сетей

2. Проанализировать некоторые существующие онлайн платформы использования искусственных нейронных сетей

3. Проанализировать некоторые существующие программные платформы разработки искусственных нейронных сетей

4. Рекомендовать наиболее понятную и простую программную платформу для первоначального обучения

5. Разработать примерные задания для практических занятий

Глава 1. Разработка ресурсов для построения курса изучения основ искусственных нейронных сетей

1.1. Основные теоретические положения по искусственным нейронным сетям

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 синаптической силе, и все произведения суммируются, определяя уровень активации нейрона[1]. На рис.1 представлена модель, реализующая эту идею.

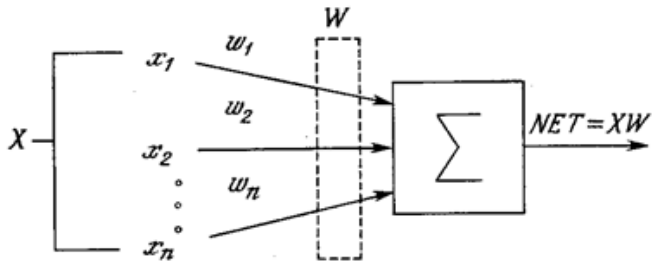

Рис. 1 Искусственный нейрон

Хотя сетевые парадигмы весьма разнообразны, в основе почти всех их лежит эта конфигурация. Здесь множество входных сигналов, обозначенных x1, x2, x3...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, w3...wn, и поступает на суммирующий блок, обозначенный СУМ. Каждый вес соответствует "силе" одной биологической синаптической связи. (Множество весов в совокупности обозначается вектором W). Суммирующий блок, соответствующий телу биологического элемента, складывает взвешенные входы алгебраически, создавая выход, который мы будем называть NET.

Сигнал NET далее, как правило, преобразуется активационной функцией F и дает выходной нейронный сигнал OUT . Активационная функция может быть обычной линейной функцией:

OUT=K(NET)

где К – постоянная, пороговой функцией

OUT=1, если NET>T

OUT=0 в остальных случаях,

где Т – некоторая постоянная пороговая величина, или же функцией более точно моделирующей нелинейную передаточную характеристику биологического нейрона и представляющей нейронной сети большие возможности [2]. На рис 2 блок, обозначенный F, принимает сигнал NET и выдает сигнал OUT.

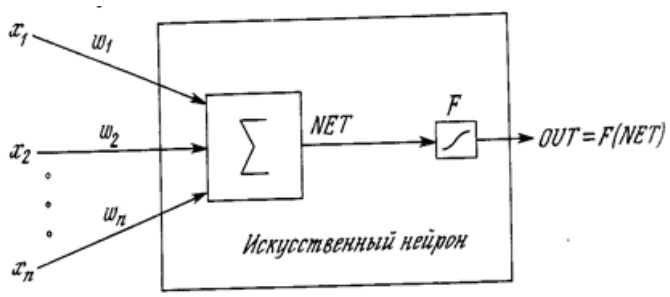

Рис. 2 Нейрон с активационной функцией

Если блок F сужает диапазон изменения величины NET так, что при любых значениях NET значения OUT принадлежат некоторому конечному интервалу, то F называется сжимающей функцией. В качестве сжимающей функции часто используется логистическая или сигмоидальная (S-образная) функции. Эта функция математически выражается как F(x)=1/(1+e-x) . Таким образом, OUT=1/(1+e-NET).

По аналогии с электронными системами активационную функцию можно считать нелинейной усилительной характеристикой искусственного нейрона. Коэффициент усиления вычисляется как отношение приращения величины OUT к вызвавшему его небольшому приращению величины NET.

Он выражается наклоном кривой при определенном уровне возбуждения и изменяется от малых значений при больших отрицательных возбуждениях (кривая почти горизонтальна) до максимального значения при нулевом возбуждении и снова уменьшается, когда возбуждение становится большим положительным. Гроссберг (1973) обнаружил, что подобная нелинейная характеристика решает поставленную им дилемму шумового насыщения. Слабые сигналы нуждаются в большом сетевом усилении, чтобы дать пригодный к использованию выходной сигнал. Однако усилительные каскады с большими коэффициентами усиления могут привести к насыщению выхода шумами усилителей (случайными флуктуациями), которые присутствуют в любой физически реализованной сети. Сильные входные сигналы в свою очередь также будут приводить к насыщению усилительных каскадов, исключая возможность полезного использования выхода. Центральная область логистической функции, имеющая большой коэффициент усиления, решает проблему обработки слабых сигналов, в то время как в области с падающим усилением на положительном и отрицательном концах подходят для больших возбуждений. Таким образом, нейрон функционирует с большим усилением в широком диапазоне уровня входного сигнала [3].

Данная модель искусственного нейрона имеет некоторые отличии от модели биологического нейрона. Искусственная нейронная сеть не учитывает задержки по времени, напрямую влияющие на динамику системы. Выходные сигналы образуются моментально после поступления входных сигналов.Так же она не учитывает воздействий функции частотной модуляции или синхронизирующей функции биологического нейрона, которые ряд исследователей считают решающими. Однако, даже не учитывая эти факты, искусственные нейронные сети обнаруживают свойства, которые напрямую проецируют биологическую систему [4]. В скором будущем мы сможем выяснить, являются ли эти совпадения случайными, или же результатом того, что

в модели искусственного нейрона спроецированы основные черты биологического нейрона.

Классификация

1. по типу входной информации: Аналоговые нейронные сети (используют информацию в форме действительных чисел); Двоичные нейронные сети (оперируют с информацией, представленной в двоичном виде).

2. по характеру обучения: С учителем (выходное пространство решений нейронной сети известно); Без учителя (нейронная сеть формирует выходное пространство решений только на основе входных воздействий). Такие сети называют самоорганизующимися [5]; С критиком (система назначения штрафов и поощрений).

3. по характеру настройки синапсов: Сети с фиксированными связями (весовые коэффициенты нейронной сети выбираются сразу, исходя из условий задачи, при этом: $dW/dt = 0$, где W — весовые коэффициенты сети); сети с динамическими связями (для них в процессе обучения происходит настройка синаптических связей, то есть $dW/dt \neq 0$, где W — весовые коэффициенты сети).

4. по характеру связей: Сети прямого распространения. Все связи направлены строго от входных нейронов к выходным [6]; Рекуррентные сети. Сигнал с выходных нейронов или нейронов скрытого слоя частично передается обратно на входы нейронов входного слоя. Как любая система, имеющая обратную связь, рекуррентная сеть стремится к устойчивому состоянию. Как известно, наиболее устойчивое состояние обеспечивается минимизацией энергии системы. Рекуррентная сеть «фильтрует» входные данные, возвращаясь к устойчивому состоянию и, таким образом, позволяет решать задачи компрессии данных и построения ассоциативной памяти [7];

Все нюансы работы человеческого мозга - все еще остаются загадкой. Однако, частично все же были выяснены основные аспекты данного процесса. Самым главным аспектом человеческого мозга являются особые клетки - нейроны [8]. Биологические нейроны способны запоминать, обрабатывать и

применять в дальнейшем любую полученную информацию. Это кардинально отличает их от любых других клеток. На рисунке 3 нейрон - состоит из тела клетки - сомы (soma), и двух типов внешних древовидных ответвлений: аксона (axon) и дендритов (dendrites). Тело клетки содержит ядро (nucleus), которое содержит информацию о наследственных свойствах нейрона, и плазму, обладающую молекулярными средствами для производства необходимых нейрону материалов. Нейрон получает сигналы (импульсы) от других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль аксона (передатчика), который в конце разветвляется на волокна (strands). На окончаниях волокон находятся синапсы (synapses).

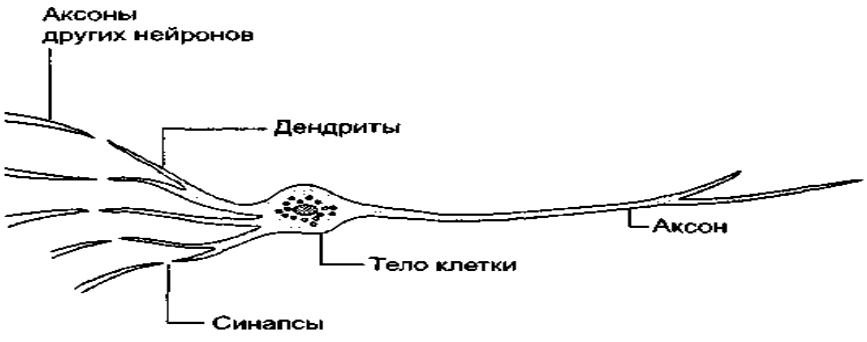

Рис. 3. Биологический нейрон

1.2 Онлайн платформы использования искусственных нейронных сетей

Для первичного понимания структуры и принципа работы искусственных нейронных сетей, изучение следует начать с работы на готовых симуляторах. На данный момент существует большое множество готовых симуляторов, подразумевающих работу как и в онлайн режиме, так и в условиях отсутствия интернета [9]. Из них можно отметить следующие.

NeuroSolutions мощный симулятор искусственных нейронных сетей. Хороший и стабильный конструктор. Поддерживает огромное количество механизмов ввода информации и обработки данных. Но, к сожалению, является довольно дорогим продуктом и существует только на английском языке. так же отсутствует справочная информация.

MemBrain мощный графический редактор и симулятор искусственных нейронных сетей. Поддерживает сети с архитектурой практически любого размера. так же как и NeuroSolutions имеет много механизмов ввода и обработки информации. Не имеет русского интерфейса и справки, в связи со сложным интерфейсом, интуитивное освоение продукта является трудной задачей [10].

SNNS разработанный впервые в университете Штутгарта симулятор искусственных нейронных сетей. В нем реализована поддержка классических нейросетевых парадигм и алгоритмов обучения. Его функциональность также может быть расширена за счет пользовательских модулей. Кроме того, для изучения и модификации доступны исходные коды этой программы.

Deductor 5.2 разработан BaseGroup Labs. Поддержка нейросетевых моделей в нем играет роль одного из многих инструментов анализа данных. Этот продукт имеет бесплатную версию для образовательных целей, русскоязычный интерфейс и справку. В нем реализованы нейросетевая модель многослойного персептрона и два алгоритма обучения, включая алгоритм обратного распространения ошибки.

Primat.org [11] является очень обширным сайтом посвященный прикладной математике и программированию. На нем можно найти самые популярные онлайн компиляторы, множество инструкций по обучению, видео уроки,

решебники, а также огромное количество научных статей по программировании и математике. Симулятор дает возможность познакомится с работой с простыми искусственными нейронными сетями. Доступно три вида демонстрации.

NeuroNet2 [12] является конструктором, с помощью которого можно создавать сети прямого распространения, содержащие до 8 скрытых слоев.

В результате анализа перечисленных симуляторов выбор был остановлен на двух из них: Primat.org (рис.4) и NeuroNet2 (рис.5). Эти стимуляторы являются наиболее удобными для начального освоения работы с ИНС и обладают простым, интуитивно понятным, удобным интерфейсом. При задании структуры сеть в реальном времени отображается в графическом виде. Эти стимуляторы не требовательны к аппаратным и программным ресурсам компьютера, и если первый из них является on-line симулятором, то второй загружается в виде архива небольшого размера, и не требует установки.

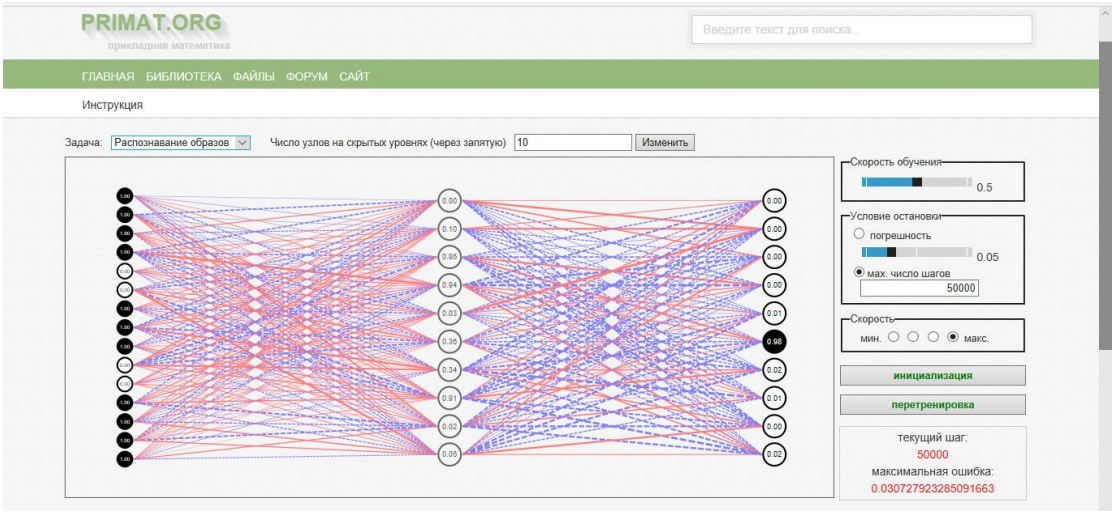

Рис.4. Интерфейс симулятора ИНС Primat.org

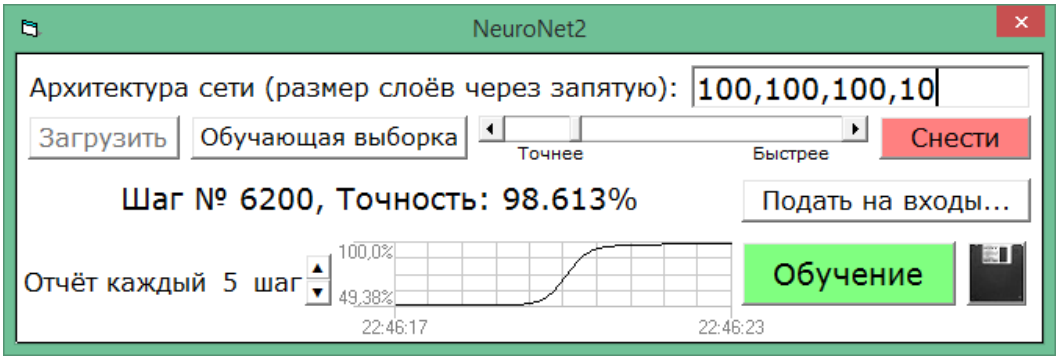

Рис.2. Рис.5. Интерфейс симулятора ИНС NeuroNet2

1.3 Программные платформы разработки искусственных нейронных сетей

Фреймворк — это заготовки, шаблоны для программной платформы, определяющие структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта. «Фреймворк» отличается от понятия библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру программного продукта и не накладывая на нее никаких ограничений. «Фреймворк» же диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять согласно указанным требованиям [13].

Для создания ИНС разработано достаточно много различных фреймворков.

Приведем примеры наиболее известных:

Caffe — среда для глубинного обучения, разработанная Яньцинем Цзя (Yangqing Jia) в процессе подготовки своей диссертации в университете Беркли. Caffe является открытым программным обеспечением, распространяемым под лицензией BSD license. Написано на языке C++, и поддерживает интерфейс на языке Python.

Keras — открытая нейросетевая библиотека, написанная на языке Python. Она представляет собой надстройку над фреймворками Deeplearning4j, TensorFlow и Theano. Нацелена на оперативную работу с сетями глубинного обучения, при этом спроектирована так, чтобы быть компактной, модульной и расширяемой. Она была создана как часть исследовательских усилий проекта ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System), а ее основным автором и поддерживающим является Франсуа Шолле (фр. François Chollet), инженер Google.

TensorFlow — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и

классификации образов, достигая качества человеческого восприятия. Применяется как для исследований, так и для разработки собственных продуктов Google. Основной API для работы с библиотекой реализован для Python, также существуют реализации для C++, Haskell, Java, Go и Swift. Является продолжением закрытого проекта DistBelief. Изначально TensorFlow была разработана командой Google Brain для внутреннего использования в Google, в 2015 году система была переведена в свободный доступ с открытой лицензией Apache 2.0.

Deeplearning4j — библиотека программ на языке Java, используемая как фреймворк для глубокого обучения. Включает реализацию ограниченной машины Больцмана, глубокой сети доверия, глубокого автокодировщика, стекового автокодировщика с фильтрацией шума, рекурсивной тензорной нейронной сети. Эти алгоритмы включены также в версии библиотеки, поддерживающие распределённыевычисления, интегрированные с архитектурами Apache Hadoop и Spark. Является открытым программным обеспечением, распространяется под лицензией Apache 2.0; основные разработчики — группа машинного обучения в Сан-Франциско во главе с Адамом Гибсоном, коммерческие внедрения поддерживают стартап Skymind.

Theano — библиотека численного вычисления в Python. Вычисления в Theano выражаются NumPy-подобным синтаксисом и компилируются для эффективных параллельных вычислений как на обычных CPU, так и на GPU. Theano является проектом с открытым исходным кодом, основным разработчиком которого является группа машинного обучения в Монреальском университете.

Глава 2. Примерное построение первой части курса

2.1. Теоретический минимум

Для использования симуляторов достаточно теоретического минимума по ИНС. Построим этот минимум на примере on-line симуляторов, с web-интерфейсом (рис. 6)

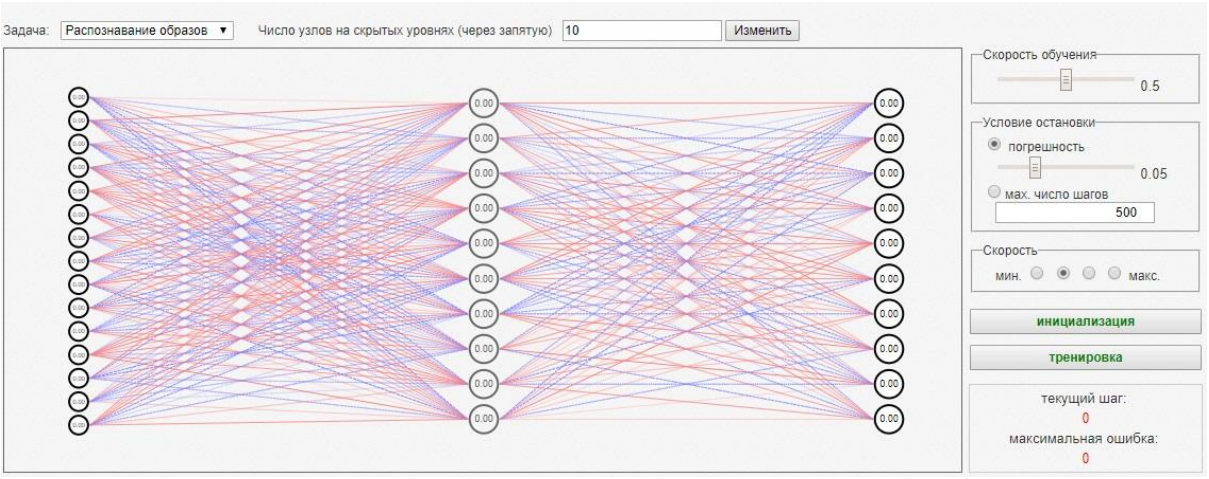

Рис.6. Распознавание образов

Очевидно, что учащийся должен знать следующие понятия.

- Искусственный нейрон

Искусственный нейрон является структурной единицей искусственной нейронной сети и представляет собой аналог биологического нейрона.

С математической точки зрения искусственный нейрон — это сумматор всех входящих сигналов, применяющий к полученной взвешенной сумме некоторую простую, в общем случае, нелинейную функцию, непрерывную на всей области определения. Обычно, данная функция монотонно возрастает. Полученный результат посылается на единственный выход.

Искусственные нейроны объединяются между собой определенным образом, образуя искусственную нейронную сеть. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов – однонаправленных входных связей, соединенных с выходами

других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал поступает на синапсы следующих нейронов.

- Связи

Связи, по которым выходные сигналы одних нейронов поступают на входы других, часто называют синапсами по аналогии со связями между биологическими нейронами. Каждая связь характеризуется своим весом. Связи с положительным весом называются возбуждающими, а с отрицательным — тормозящими.

- Вес связи

Синапс это связь между двумя нейронами. У синапсов есть 1 параметр — вес. Благодаря ему, входная информация изменяется, когда передается от одного нейрона к другому.

- Входной нейрон

Формальный нейрон, выполняющий в конкретной системе нейронов (нейронной сети) функцию входа, т. е. воспринимающий сигналы только от внешней для данной системы среды.

- Входной слой

Система входных нейронов. Такая система получает входные сигналы (исходные данные) через несколько входных каналов(входных нейронов).

- Выходной нейрон

Искусственный нейрон имеет один выход, часто называемый аксоном по аналогии с биологическим прототипом. С единственного выхода нейрона сигнал может поступать на произвольное число входов других нейронов

- Скрытый (или промежуточный) слой

Слои нейронов с первого по предпоследний называются скрытыми слоями (промежуточными). В литературе нет единообразия относительно того, как считать число слоев в многослойных нейронных сетях. Одни предлагают считать число слоев, включая не суммирующий входной слой, другие – считать, только слои, выполняющие суммирование.

- Ошибка обучения

Необходимо назначить такие значения весов и смещений, которые смогут минимизировать ошибку решения. Веса и смещения автоматически настраиваются таким образом, чтобы минимизировать разность между желаемым и полученным на выходе сигналами, которая называется ошибка обучения

- Эпоха

Эпоха - одна итерация в процессе обучения, включающая предъявление всех примеров из обучающего множества и, возможно, проверку качества обучения на контрольном множестве.

- Архитектура сети

Способы соединения нейронных сетей. Включает в себя входной слой, выходной и определенное количество скрытых слоев, зависящих от количества входных нейронов

- Обучающий набор данных

Искусственная нейронная сеть обычно обучается с учителем. Это означает наличие обучающего набора (датасета), который содержит примеры с истинными значениями: тегами, классами, показателями.

Например, если вы хотите создать нейросеть для оценки тональности текста, датасетом будет список предложений с соответствующими каждому эмоциональными оценками [14]. Тональность текста определяют признаки (слова, фразы, структура предложения), которые придают негативную или позитивную окраску. Веса признаков в итоговой оценке тональности текста (позитивный, негативный, нейтральный) зависят от математической функции, которая вычисляется во время обучения нейронной сети.

- Функция активации

Функция активации — это способ нормализации входных данных. То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне.

- Обучение сети

Обучение нейронной сети- это процесс, в котором параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть встроена. Тип обучения определяется способом подстройки параметров. Различают алгоритмы обучения с учителем и без учителя [15].

Процесс обучения с учителем представляет собой предъявление сети выборки обучающих примеров. Каждый образец подается на входы сети, затем проходит обработку внутри структуры НС, вычисляется выходной сигнал сети, который сравнивается с соответствующим значением целевого вектора, представляющего собой требуемый выход сети.

При обучении без учителя обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса сети так, чтобы получались согласованные выходные векторы, т.е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы [16].

Подробные сведения о параметрах симуляторов:

Число узлов скрытых слоев

Вы можете изменить количество нейронов для каждого промежуточного слоя. Пожалуйста, введите значения, разделенные запятыми. Например, "2, 3" указывает, что 1-й слой содержит два нейрона, второй слой - 3. Вы можете увеличивать число слоев, добавляя через запятую число нейронов в новом слое [17].

Замечание. Если вы введете слишком много слоев и нейронов, то ваш браузер может зависнуть. Поэтому не увлекайтесь.

Скорость обучения

Если Вы изучаете нейронные сети, то вы знаете, что это за параметр. Он влияет и на точность получаемого результата, и на скорость обучения.

Условия остановки

Эти условия указывают, когда будет завершено обучение. Тут надо экспериментировать в зависимости от целей обучения. Если Вы зададите конкретное число шагов, то обучение гарантированно остановится после

прохождения указанного числа циклов обучения [18]. Указывая определенную точность, можно надолго запустить процесс обучения и даже не получить требуемой точности никогда для определенного сочетания параметров системы.

Скорость

Вы можете регулировать скорость визуализации от минимальной до максимальной. Минимальная подходит, если вам надо показать, как идет процесс обучения. Но, чтобы реально обучить систему и получать результат с заданной точностью, лучше устанавливать максимальную скорость [19].

2.2. Рекомендуемые онлайн платформы и примерные задачи, решаемые на них

Ранее были приведены различные онлайн платформы для построения искусственных нейронных сетей. Из них были выбраны два наиболее удобных симуляторов: Primat.org и NeuroNet2. Разберем подробнее первый вариант.

Primat.org является очень обширным сайтом посвященный прикладной математике и програмированнию. На нем можно найти самые популярные онлайн компиляторы, множество инструкций по обучению, видео уроки, решебники, а также огромное количество научных статей по программировании и математике.

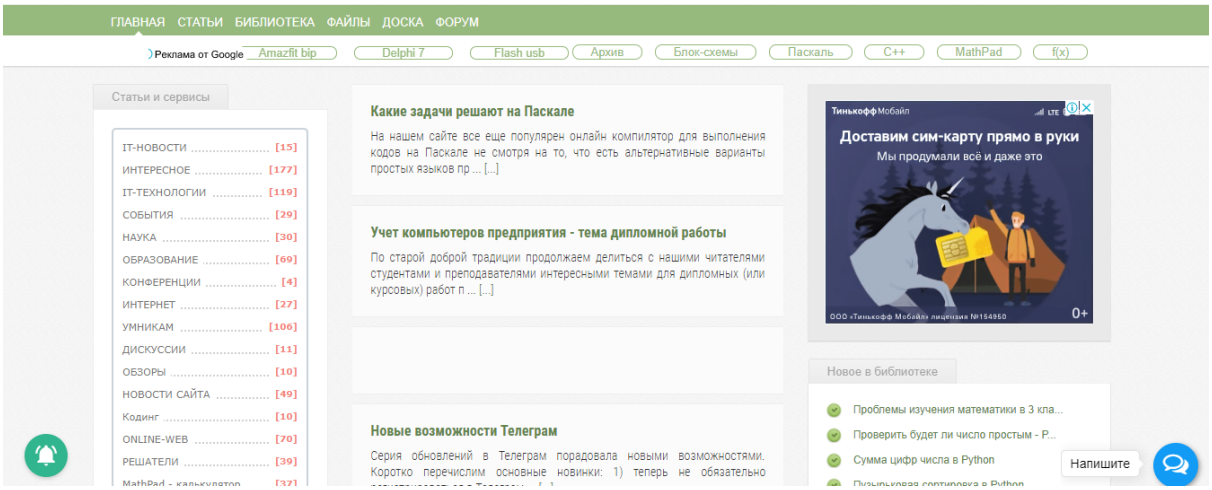

Рис. 6. Интерфейс сайта Primat.org

Для того чтобы найти во всем этом многообразии информацию, касающуюся искусственных нейронных сетей, достаточно запросить в поисковике “нейронные сети” и сайт покажет имеющиеся статьи по этой теме.(рис. 7)

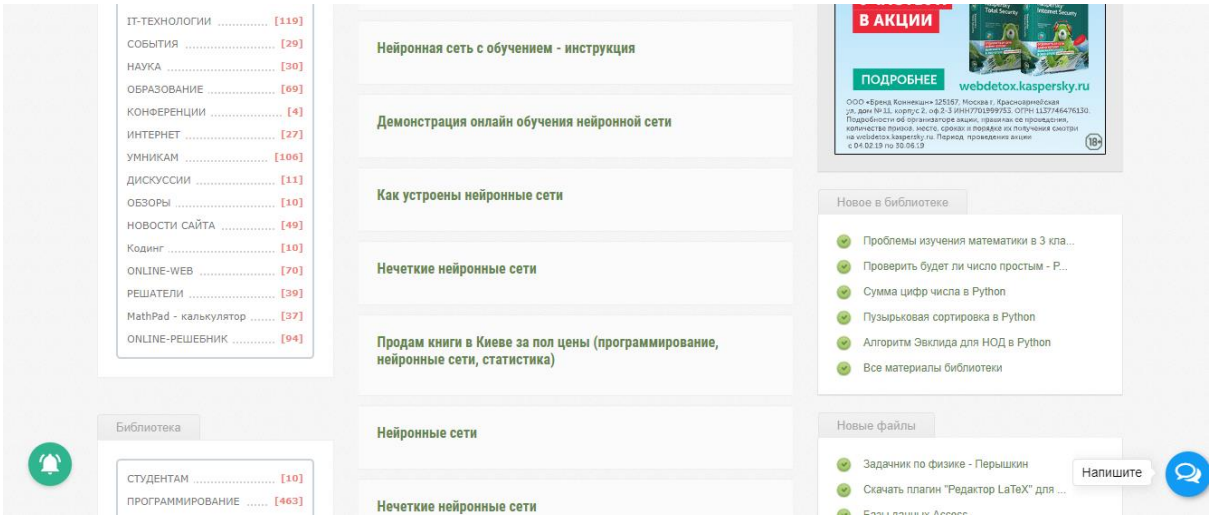

Рис. 7. Результаты поиска по запросу “нейронные сети”

Для тех кто еще не знаком с нейронными сетями будет полезно почитать имеющиеся на сайте статьи посвященные этой теме. Отсюда же можно сразу перейти к демонстрации онлайн обучения нейронной сети. Перейдя по этой ссылке можно увидеть краткие сведения о параметрах данного онлайн компилятора и перейти непосредственно к программе.

Теперь перед нами сам онлайн компилятор. Ранее уже приводились скриншоты данного компилятора, но приведем его еще раз и разберем полностью то, что мы видим на экране. (рис. 8)

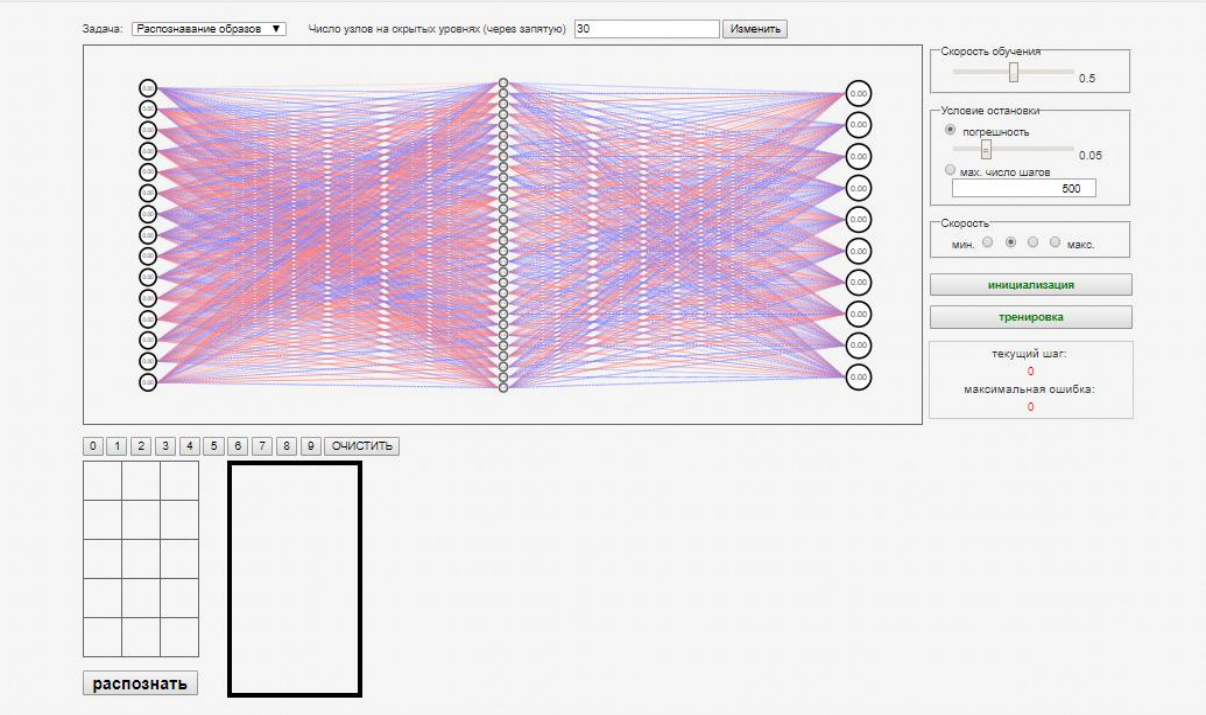

Рис. 8. Онлайн компилятор Primat.org

Онлайн компилятор на Primat.org решает три задачи: распознование образов, аппроксимация функции и логика XOR.

Итак, начнем мы с того, что выберем задачу. Будем разбирать нейронные сети на примере распознавания образов. а конкретно десять чисел от 0 до 9. в самом внизу мы видим эти числа. При нажатии на одно из них оно будет появляться в поле левее в виде пятнадцати пикселей. Эти пиксели и будут соответствовать пятнадцати входных нейронов [20]. После указании “распознать” нейронная сеть постарается это сделать, но с большой вероятностью она укажет неверное число.

Это произошло потому что наша нейронная сеть еще не обучена, и практически рандомно выбирает правильный ответ.

Обратим внимание на меню справа. Первым пунктом стоит скорость Эти условия указывают, когда будет завершено обучение. Далее идет условия остановки. Тут надо экспериментировать в зависимости от целей обучения. Если

Вы зададите конкретное число шагов, то обучение гарантированно остановится после прохождения указанного числа циклов обучения. Указывая определенную точность, можно надолго запустить процесс обучения и даже не получить требуемой точности никогда для определенного сочетания параметров системы. Скорость - вы можете регулировать скорость визуализации от минимальной до максимальной. Минимальная подходит, если вам надо показать, как идет процесс обучения. Но, чтобы реально обучить систему и получать результат с заданной точностью, лучше устанавливать максимальную скорость.обучения. Чем выше скорость, тем точнее и быстрее пройдет обучение. Инициализация сбросит обученную нейронную сеть на изначальное состояние. Тренировка запустит процесс обучения по заданным параметрам.

Теперь наша нейронная сеть будет выдавать более точные результаты. Для достижения минимальной ошибки обучения рекомендуется поэкспериментировать с параметрами тренировки, а также с количеством внутренних слоев.

2.3. Примерное тематическое планирование и содержание первой части курса

Предмет: информатика

Класс: 9

Образовательная область - Информатика

Количество часов: 14

Содержание курса

Первая часть данного курса рассчитана на 14 часов (1 час в неделю в 9 классе), программой предусматривается последовательное изучение разделов:

1. История возникновения искусственного интеллекта
2. Принцип работы искусственных нейронных сетей
3. Сферы применения
4. Архитектура сети
5. Способы обучения нейронной сети
6. Обучающий набор данных
7. Ошибка обучения и эпохи
8. Функция активации
9. Работа с нейронными сетями в онлайн компиляторе Primat.org
10. Решение практических заданий в онлайн компиляторе Primat.org

Цель изучения курса

Основная цель курса — приобретение общих знаний в области Нейронных сетей и приобретение первоначальных навыков в работе с нейронными сетями на готовом онлайн компиляторе.

Образовательные

Получение представления о искусственных нейронных сетях.

Освоение технологии обучения нейронной сети.

Формирование умений использовать искусственные нейронные сети в будущей профессиональной деятельности

Развивающие

Развитие способностей аналитического мышления.

Развитие представлений о новом подходе к обработке информации.

Развитие творческих способностей в процессе моделирования нейронных сетей.

Воспитательные

Воспитание ответственности за результаты своей работы в коллективе, адекватная оценка своего вклада в общее дело.

Воспитание отношения к информационной культуре и информационной безопасности.

Воспитание информационной культуре.

Тематическое планирование

Тематическое планирование первой части курса приведено в табл. 1

Таблица 1

Тематическое планирование первой части курса

№ п/п	**Название темы**	**Количество часов**
1	**Искусственные нейронные сети**	**4**
1.1	История возникновения искусственного интеллекта	1
1.2	Принцип работы искусственных нейронных сетей	2
1.3	Сферы применения	1
2	**Строение и принцип работы с нейронными сетями**	**7**
2.1	Архитектура сети	2
2.2	Способы обучения нейронной сети	2
2.3	Обучающий набор данных	1
2.4	Ошибка обучения и эпохи	1
2.5	Функция активации	1

3	**Практическая работа с нейронными сетями**	**3**
3.1	Работа с нейронными сетями в онлайн компиляторе Primat.org	1
3.2	Решение практических заданий в онлайн компиляторе Primat.org	2
	ВСЕГО:	**14**

Планирование курса

Теоретический минимум

Перед тем как начать изучать искусственные нейронные сети, очень важно понимать что они из себя представляют, и для чего они вообще нужны, основываясь на реалиях современного мира.

Искусственные нейронные сети применяются в различных областях науки: начиная от систем распознавания речи до распознавания вторичной структуры белка, классификации различных видов рака и генной инженерии. Особенно привлекают внимание приложения для смартфонов, трансформирующие изображения. Всё это делает привлекательным и полезным освоение основ разработки искусственных нейронных сетей школьниками.

Практически все передовые компании активно занимаются исследованиями и разработками искусственного интеллекта, что способствует тому, что новые открытия в этой сфере появляются чуть ли не каждый день. Наиболее перспективным направлением искусственного интеллекта являются нейронные сети. Основная идея заключается в том, чтобы создать максимально близкую модель человеческой нервной системы, то есть смоделировать основные ее

способности: обучаться и исправлять ошибки. Безусловно, изучение нейронных сетей сейчас одно из актуальных направлений развития для всех отраслей наук. Учитывая тот факт, что популярность искусственного интеллекта еще не достигла своего пика развития, мы можем предположить что еще многие годы, а может и десятилетия эта сфера будет активно изучаться. Соответственно, всегда будет оставаться потребность в большом количестве квалифицированных специалистов.

Искусственный нейрон является упрощенной моделью биологического нейрона. Вся работа мозга основана на непрерывной связи множества нейронов. Каждый из них связан с другими нейронами с помощью нервных волокон, которые передают электрические импульсы. Это означает что все внешние и внутренние раздражители передаются к мозгу с помощью электрических импульсов. Сама же биологическая клетка, нейрон, обрабатывает информацию и передает на следующий нейрон. Когда импульс достигает синаптического окончания, высвобождаются химические вещества отвечающие за нашу реакцию на раздражитель.

По этому принципу и построены искусственные нейронные сети. Искусственный нейрон, так же как и биологический, является структурной единицей искусственной нейронной сети. С математической точки зрения искусственный нейрон — это сумматор всех входящих сигналов, применяющий к полученной взвешенной сумме некоторую простую, в общем случае, нелинейную функцию, непрерывную на всей области определения. Обычно, данная функция монотонно возрастает. Полученный результат посылается на единственный выход. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Связи, по которым проходит информация от одного нейрона к другому называются синапсами, так же как и в биологической нейронной сети. Каждая из таких связей имеет свой вес, который является единственным параметром синапсов. Отличают возбуждающие связи (имеющие положительный

вес) и тормозящие (с отрицательным весом). В биологической сети входные сигналы поступают через через нервные окончания, которые образуют наши органы чувств. В искусственной сети за входные данные отвечают входные нейроны, образующие входной слой. Выходной нейрон - искусственный нейрон, который имеет один выход, является аналогом аксона. с единственного выхода нейрона сигнал может поступать на произвольное число входов других нейронов. Слои нейронов с первого по предпоследний называются скрытыми слоями (промежуточными). В литературе нет единообразия относительно того, как считать число слоев в многослойных нейронных сетях. Одни предлагают считать число слоев, включая не суммирующий входной слой, другие – считать, только слои, выполняющие суммирование.

Итак, что же включает в себя искусственная нейронная сеть.

1. Входной слой, на нейроны которых будет поступать некая информация. В случае с распознаванием картинки, количество слоев будет соответствовать количеству пикселей.

2. Связи соединяющие нейроны. Каждая связь имеет свой вес. Информация каждый раз будет обрабатываться при переходе от одного нейрона к другому.

3. Скрытый слой нейронов

4. Выходной слой. Количество выходных слоев зависит от того какое число результатов у нас возможно. Если мы распознаем цифры от 0 до 9, то количество выходных слоев будет 10. Каждый выходной нейрон будет соответствовать некоторому числу от 0 до 1. Нейрон с числом максимально приближенным к 1 будет соответствовать с правильным распознаванием изображения.

Способы соединения входного слоя, скрытых слоев и выходного слоя называется архитектура сети.

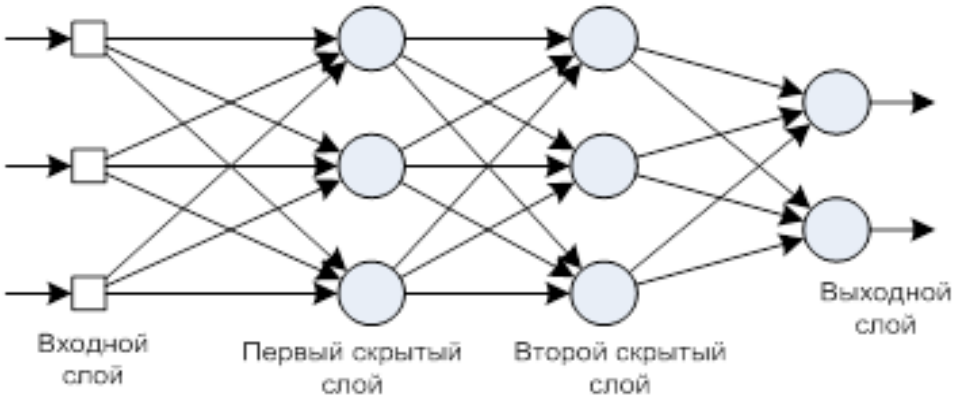

Рис. 9. Архитектура сети прямого распространения

Искусственные нейронные сети способны обучаться. При прохождении всех возможных итоговых вариантов сеть будет выдавать некоторые числовое значение от 0 до 1 обозначающее ошибку обучения. Соответственно, минимальное значение будет обозначать наиболее правильное распознавание. Веса и смещения автоматически настраиваются таким образом, чтобы минимизировать разность между желаемым и полученным на выходе сигналами. На ошибку обучения также влияет количество эпох. Эпоха - одна итерация в процессе обучения, включающая предъявление всех примеров из обучающего множества и, возможно, проверку качества обучения на контрольном множестве.

Обучение искусственной нейронной сети возможно двумя методами: с учителем и без учителя. При обучении без учителя нейронная сеть имеет входные данные, но не знает что должно получится на выходе. Нейронная сеть пытается самостоятельно найти корреляции в данных, извлекая полезные признаки и анализируя их. Обучение с учителем обозначает наличие обучающего набора (dataset). Такой набор включает в себя примеры истинных значений, например различные варианты цифр написанных от руки. С таким набором нейронная сеть быстрее и точнее найдет все истинные значения и минимизирует ошибку обучения.

Онлайн компилятор на Primat.org

Изучив основные теоретические положения, мы наконец то можем приступить к нашему первому знакомству с нейронными сетями на практике. Мы начнем с готового онлайн компилятора на Primat.org. Он достаточно прост в использовании и идеально подходит для первоначального понимания работы нейронных сетей.

Для того чтобы найти нужную информацию на Primat.org., касающуюся искусственных нейронных сетей, достаточно запросить в поисковике “нейронные сети” и сайт покажет имеющиеся статьи по этой теме.(рис. 10)

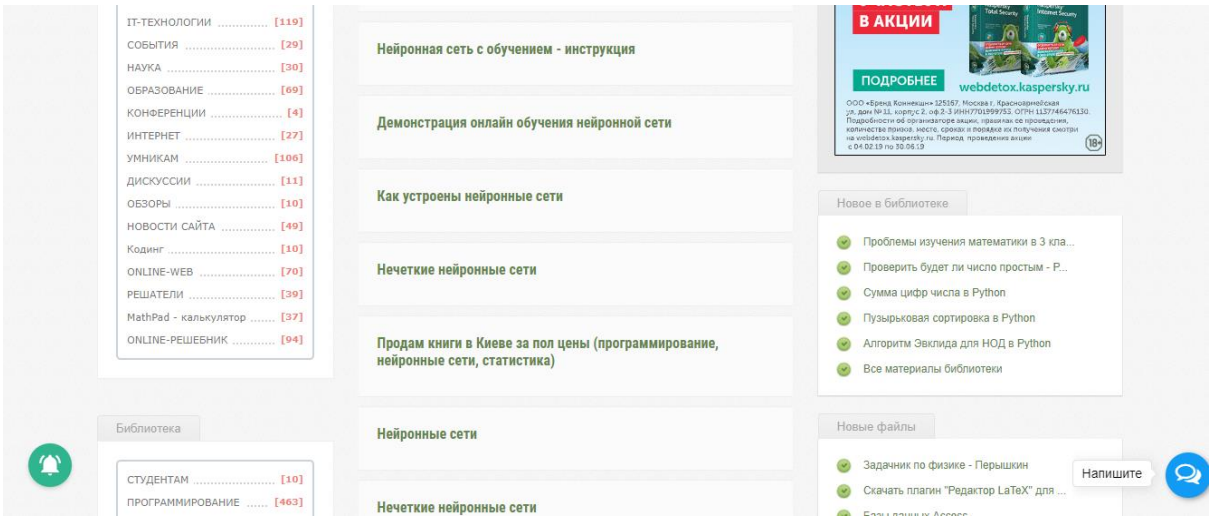

Рис. 10. Результаты поиска по запросу “нейронные сети”

Можно увидеть множество статей касающихся нейронных сетей, а также демонстрацию онлайн обучения нейронным сетям. Откроем онлайн компилятор и подробно разберем его возможности.

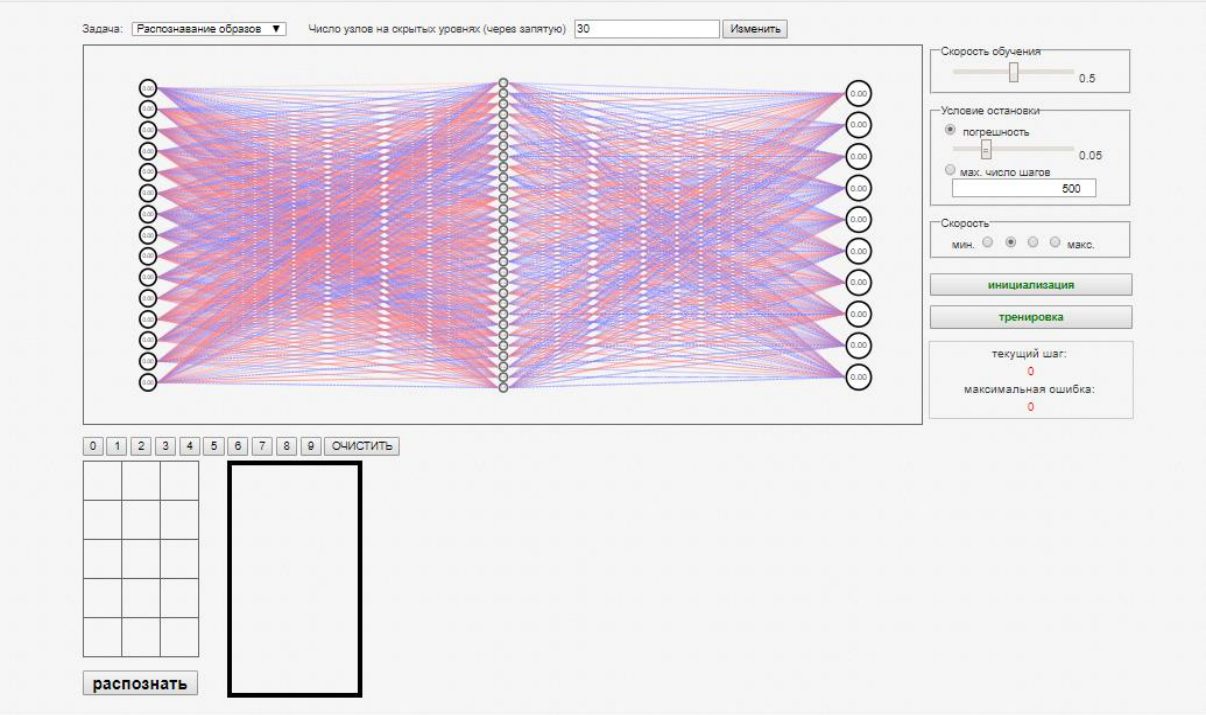

Рис. 11. Онлайн компилятор Primat.org

Итак, начнем мы с того, что выберем задачу. Будем разбирать нейронные сети на примере распознавания образов. а конкретно десять чисел от 0 до 9. в самом внизу мы видим эти числа. При нажатии на одно из них оно будет появляться в поле левее в виде пятнадцати пикселей. Эти пиксели и будут соответствовать пятнадцати входных нейронов. После указании “распознать” нейронная сеть постарается это сделать, но с большой вероятностью она укажет неверное число. (рис. 12)

Рис. 12. Результат распознавания необученной сети

Это произошло потому что наша нейронная сеть еще не обучена, и практически рандомно выбирает правильный ответ.

Обратим внимание на меню справа. Первым пунктом стоит скорость Эти условия указывают, когда будет завершено обучение. Далее идет условия остановки. Тут надо экспериментировать в зависимости от целей обучения. Если Вы зададите конкретное число шагов, то обучение гарантированно остановится после прохождения указанного числа циклов обучения. Указывая определенную точность, можно надолго запустить процесс обучения и даже не получить требуемой точности никогда для определенного сочетания параметров системы. Скорость - вы можете регулировать скорость визуализации от минимальной до максимальной. Минимальная подходит, если вам надо показать, как идет процесс обучения. Но, чтобы реально обучить систему и получать результат с заданной точностью, лучше устанавливать максимальную скорость.обучения. Чем выше скорость, тем точнее и быстрее пройдет обучение. Инициализация сбросит обученную нейронную сеть на изначальное состояние. Тренировка запустит процесс обучения по заданным параметрам.

Теперь наша нейронная сеть будет выдавать более точные результаты. Для достижения минимальной ошибки обучения рекомендуется поэкспериментировать с параметрами тренировки, а также с количеством внутренних слоев.

Лабораторные работы на Primat.org

1 задание: Исследовать влияние количество узлов на обучаемой сети, выбирая задачу “распознавание образов”.

Ход работы:

1. Сделать один узел из “10” нейронов

2. Для быстрой проверки выбираем максимальную скорость
3. Делаем инициализацию и тренировку и получаем невысокую максимальную ошибку
4. Распознаем цифры и видим что все цифры распознаются правильно
5. Затем добавим еще один узел из “10” нейронов
6. Нажимаем “Изменить”
7. Делаем инициализацию и тренировку и получаем максимальную ошибку на чуть больше
8. И так продолжаем добавлять узлы по 10 нейронов
9. С каждым добавлением узлов мы делаем инициализацию и тренировку и получаем максимальную ошибку с каждым раз больше
10. Распознаем цифры и видим что все не цифры распознаются правильно

Вывод: Чем больше выбираем скрытых слоев, тем выше максимальная ошибка, что в итоге не все цифры распознаются правильно.

2 задание: Исследовать влияние количество узлов на обучаемой сети, выбирая задачу “аппроксимация функции”.

Ход работы:

1. Тоже самое что и в первом задании
2. Выбираем функцию sin(x)
3. Делаем инициализацию и тренировку и видим, что функция строится не правильно
4. Затем каждый раз добавляем еще один узел из “10” нейронов
5. Нажимаем “Изменить”
6. С каждым добавлением узлов мы делаем инициализацию и тренировку и видим, что функция все равно строится не правильно

Вывод: Сколько бы мы не брали узлов, функция все равно строится не правильно.

3 задание: Исследовать влияние количество эпох обучения на обучаемой сети выбирая задачу “распознавание образов”

Ход работы:

1. Опять тоже самое что и в первом задании
2. Но только на данном этапе у нас 3 узла по 10 нейронов
3. Увеличим количество шагов до 50 000
4. Делаем инициализацию и тренировку и в этот раз получаем невысокую максимальную ошибку
5. Распознаем цифры и видим что все цифры распознаются правильно

Вывод: Чем больше выбираем число шагов, тем ,возможно, ниже максимальная ошибка, что в итоге все цифры распознаются правильно.

4 задание: Исследовать влияние количество эпох обучения на обучаемой сети выбирая задачу “аппроксимация функции”.

Ход работы:

1. Тоже самое что и во втором задании
2. Но только не добавляя узлов
3. Выбираем функцию sin(x)
4. Увеличиваем количество шагов до 5 млн.
5. Делаем инициализацию и тренировку и видим, что функция строится более правильно

Вывод: Чем больше количество шагов, тем функция строится более правильно.

5 задание: Исследовать влияние количество нейронов на обучаемой сети выбирая задачу “распознавание образов”.

Ход работы:

1. Сделать один узел из “30” нейронов
2. Для быстрой проверки выбираем максимальную скорость
3. Делаем инициализацию и тренировку и получаем невысокую максимальную ошибку
4. Распознаем цифры и видим что все цифры распознаются правильно
5. Затем уменьшим количество нейрон до “2”
6. Нажимаем “Изменить”
7. Делаем инициализацию и тренировку и получаем высокую максимальную ошибку
8. Распознаем цифры и видим что все цифры распознаются неправильно

Вывод: Чем меньше выбираем количество нейронов , тем выше максимальная ошибка, что в итоге не все цифры распознаются правильно

6 задание: Исследовать влияние количество двух узлов, меняя количество нейронов в обоих синхронно на обучаемой сети выбирая задачу “распознавание образов”.

Ход работы:

1. Сделать два узла по “20” нейронов в каждом
2. Для быстрой проверки выбираем максимальную скорость
3. Делаем инициализацию и тренировку и получаем невысокую максимальную ошибку
4. Распознаем цифры и видим что все цифры распознаются правильно
5. Затем в первом узле мы уменьшаем количество нейронов до “5”, а во втором увеличиваем до “22”

6. Нажимаем “Изменить”

7. Делаем инициализацию и тренировку и видим, что цифры опять распознаются правильно

8. Затем в первом узле мы увеличиваем количество нейронов до “23”, а во втором уменьшаем до “3”

9. Распознаем цифры и видим что все цифры распознаются неправильно

Вывод: Правильность распознавания цифр зависит от того как вы синхронно будете менять количество нейронов в каждом узле.

Глава 3. Примерное построение второй части курса

3.1. Выбор подходящей программной платформы

Для более глубокой работы с искусственными нейронными сетями, используя программирование, не достаточно просто скачать и установить программу. Нам потребуется множество заготовок и подпрограмм, которые помогут нам программировать намного быстрее и грамотнее. Ранее были приведены примеры разнообразных фреймворков, но, так как фреймворки являются мощными профессиональными продуктами, они подходят для тех, кто лишь начал изучение искусственных нейронных сетей.

PyBrain - одна из лучших Python библиотек предназначенная для первичного изучения искусственных нейронных сетей, способная реализовать огромное количество всевозможных алгоритмов на основе нейронных сетей.

PyBrain представляет собой модульную библиотеку предназначенную для реализации различных алгоритмов машинного обучения на языке Python [21]. Основной целью библиотеки PyBrain остается возможность предоставить пользователю максимально пластичных и простых в работе инструментов, которые в свою очередь способны реализовать множество задач из области машинного обучения. Название PyBrain являетсяаббревиатуройотанглийского: Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library.

Замечательной возможностью PyBrain является то, что как библиотека Python, он позволяет создавать и использовать ИНС в интерактивном режиме при помощи командной строки интерпретатора Python. Это позволяет обеспечить наглядность и прозрачность создания и использования ИНС.

Различные языка программирования предоставляют возможность работать с искусственными нейронными сетями, но мы все же сделаем акцент именно на Python.

Python - интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. Встроенные высокоуровневые структуры данных в сочетании с динамическими типизацией и связыванием делают язык привлекательным для быстрой разработки приложений (RAD, Rapid Application Development) [22]. Кроме того, его можно использовать в качестве сценарного языка для связи программных компонентов. Также стоит отметить синтаксис Python, не являясь особо сложным, он отдает особое предпочтение читаемости кода. Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода. Интерпретатор Python и большая стандартная библиотека доступны бесплатно в виде исходных и исполняемых кодов для всех основных платформ и могут свободно распространяться.

Профессиональные пользователи изучающие искусственный интеллект предпочитают использовать именно Python. И большинство библиотек предназначены именно под Python [23].

3.2. Необходимый минимум знаний по языку Python для построения простейших сетей на Pybrain.

Python это интерпретируемый язык программирования и включает в себя особую командную строку, благодаря которой можно последовательно выполнять несколько команд по порядку в интерактивном режиме. При использовании библиотеки PyBrain это позволяет прослеживать каждое действие при работе с искусственными нейронными сетями [24].

К примеру, для создания нейронной сети с шестью входными нейронами, семью нейронами в скрытом слое, и двумя выходными нейронами достаточно создать команду [25].

```
>>> net = buildNetwork(2, 3, 1)
```

Python мощный и многофункциональный язык. Его возможности обеспечиваются, например, следующими конструкциями и свойствами языка:

- Основы (Комментарии, Литеральные константы, Числа, Строки, Переменные, Имена идентификаторов, Типы данных, Объекты, Логические и физические строки, Отступы)
- Операторы и выражения (Операторы, Порядок вычисления, Изменение порядка вычисления, Ассоциативность, Выражения)
- Поток команд (Оператор if, Оператор while, Цикл for, Оператор break, Оператор continue)
- Функции (Параметры функций, Локальные переменные, Зарезервированное слово “global”, Зарезервированное слово “nonlocal”, Значения аргументов по умолчанию, Ключевые аргументы, Переменное число параметров, Только ключевые параметры, Оператор “return”, Строки документации, Аннотации)

- Модули (Файлы байткода .pyc, Оператор from ... import ..., Имя модуля – __name__, Создание собственных модулей, Функция dir, Пакеты)
- Структуры данных (Список, Краткое введение в объекты и классы, Кортеж, Словарь, Последовательности, Множество, Ссылки)
- Объектно-ориентированное программирование (self, Классы, Методы объектов, Метод __init__, Переменные класса и объекта, Наследование, Метаклассы)
- Ввод-вывод (Ввод от пользователя, Файлы, Pickle)
- Исключения (Ошибки, Исключения, Обработка исключений, Вызов исключения, Try .. Finally, Оператор with)
- Стандартная библиотека (Модуль sys, Модуль logging)
- А так же: Передача кортежей, Специальные методы, Блоки в одно выражение, Lambda-формы, Генераторы списков, Передача кортежей и словарей в функции, exec и eval, Оператор assert, Функция repr, Управляющие последовательности, Необрабатываемые строки [26].

Знакомство с созданием и работой с простейшими ИНС показало, что из всего этого списка достаточно следующих конструкций и возможностей [27]:

- Литеральные константы, числа, строки,
- Переменные, имена идентификаторов, типы данных, объекты
- Операторы и выражения
- Функции и параметры функций
- Локальные переменные
- Значения аргументов по умолчанию
- Оператор “return”
- Модули
- Оператор from ... import ...
- Создание собственных модулей
- Пакеты

- Списки
- Краткое введение в объекты и классы
- Классы
- Методы объектов
- Метод __init__
- Переменные класса и объекта
- Ввод от пользователя
- Файлы

3.3. Установка и использование программной платформы

Изначально компьютер не умеет работать с программным кодом, написанным на языке Python. Для этого необходимо установить программу, которая будет выполнять созданные файлы — это интерпретатор вместе со стандартной библиотекой.

Переходим на сайт https://www.python.org/downloads/ и скачиваем последнюю версию Python (на рисунке 13 выделено красным). Желтая кнопка справа предлагает скачать старую версию Python для Windows XP и других версий Windows, ниже приведенной версии. Программа Python для старых версий Windows не нужна, поэтому нажимаем на левую кнопку.

Запускаем скачанный .exe файл.

В первом открывшемся окне (рис. 14) вас просят выбрать способ установки: простой или сложный. Сначала включите нижний флажок «Add Python 3.6 to PATH». Эта функция позволит использовать Python в любом месте через консоль.

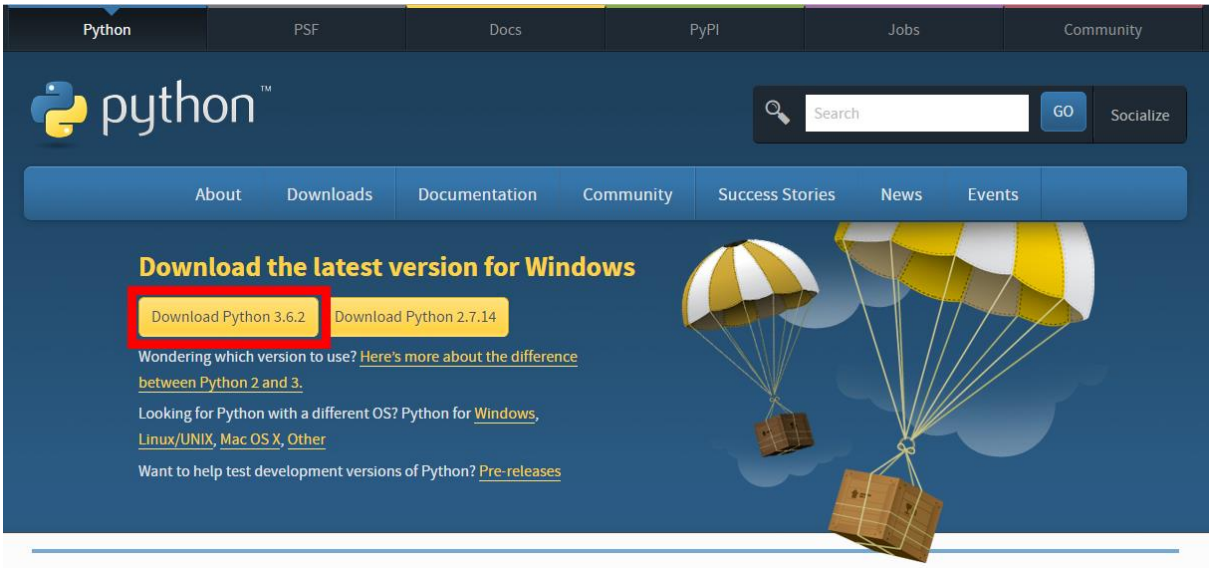

Рис. 13. Окно выбора версии для скачивания

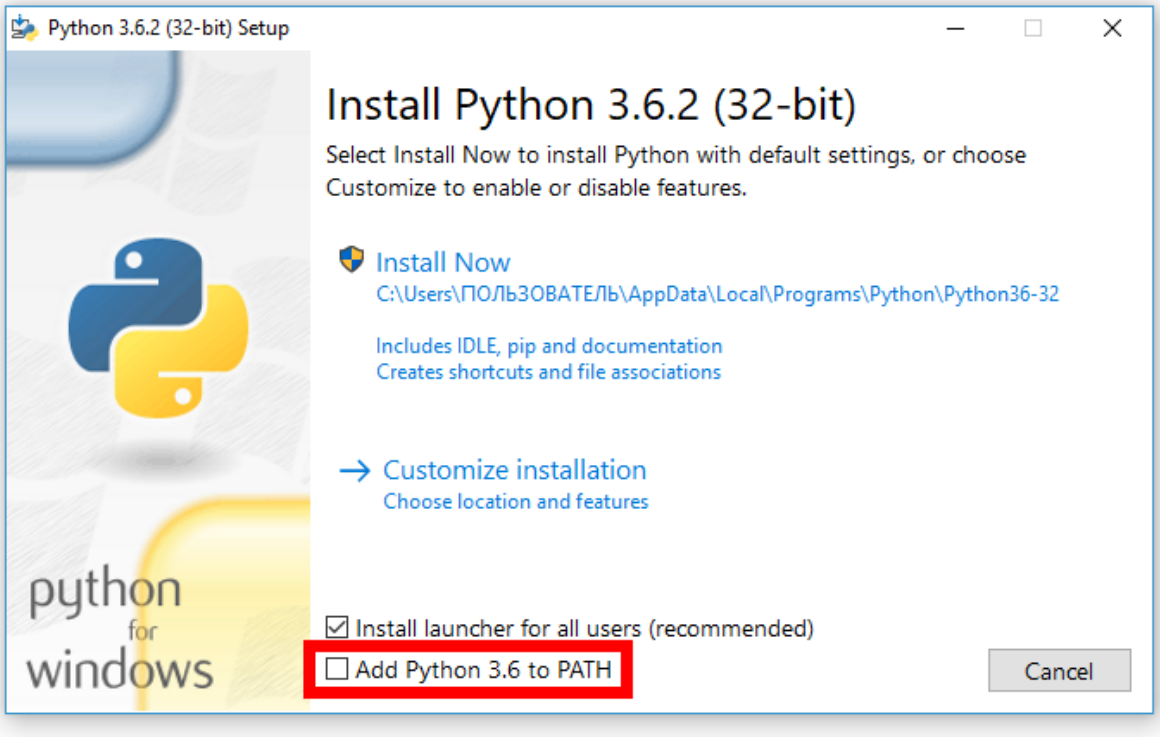

Рис. 14. Окно выбора способа установки

Далее выбираете верхний (простой) способ установки.

После этого начнется процесс установки. По окончанию установки закройте окно установщика.

Программа Python установлена на вашем компьютере, и теперь вы можете программировать на языке Python.

Среда для написания программ на Python

Ваш компьютер уже умеет работать с языком программирования Python. Сам программный код можно писать хоть в Блокноте, но это очень неудобно. Нам нужна умная среда, которая будет подсвечивать набираемый код и автоматически указывать на различные ошибки в нем.

Одна из таких программ— PyCharm от JetBrains. Эта компания специализируется на создании профессиональных сред для программирования. Программы от JetBrains занимают лидирующее положение среди программистов.

Переходим на сайт https://www.jetbrains.com/pycharm/ для скачивания PyCharm. Для скачивания доступно две версии: профессиональная и версия для

сообщества. (на рис. 15 выделено красным). Версия для сообщества бесплатная. Ее и скачаем.

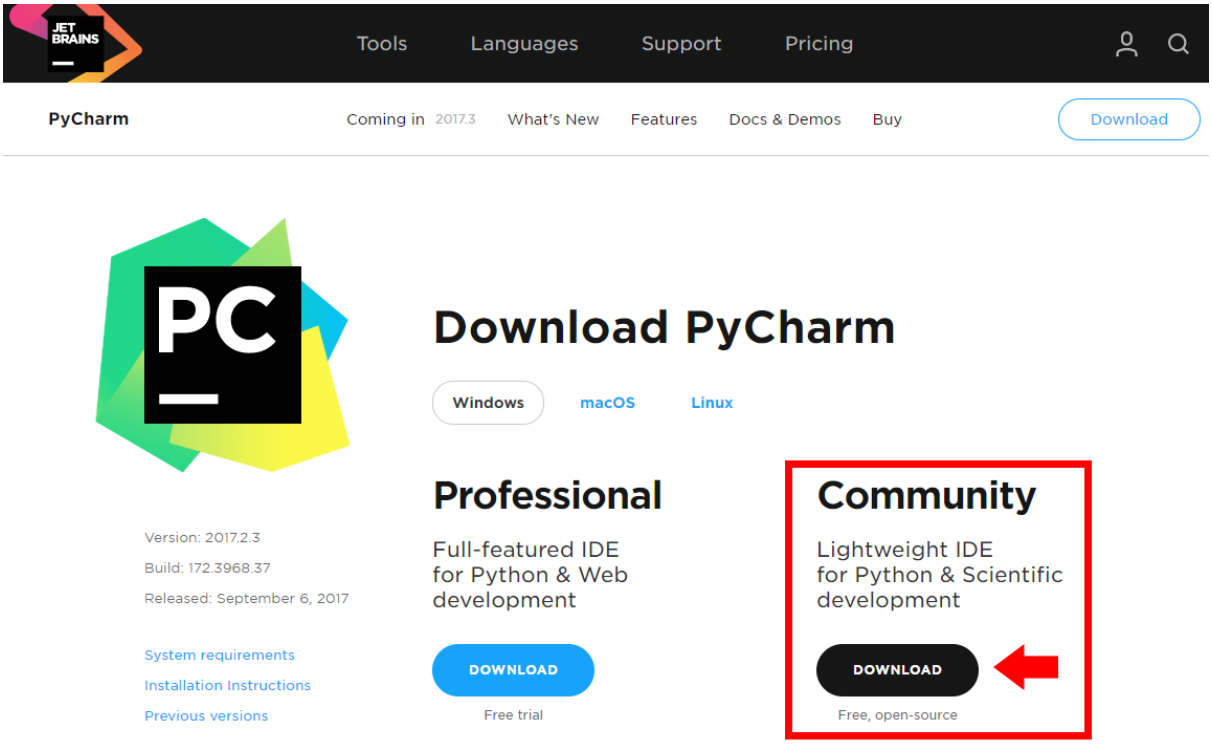

Рис. 15. Окно выбора версии продукта

Запускаете скачанный .exe файл. В первом окне вас приветствует сам установщик. Нажимаете на клавишу «Next». Далее следует указать место установки среды. Выбираем по умолчанию, как предлагает вам компьютер. Обычно это диск C:\.

Важный момент! В следующем окне (рис. 16) обязательно ставьте галочку «Download and install JRE x86 by JetBrains», если на компьютере не установлена Java. Также следует поставить галочку «.py» для того, чтобы все файлы с программным кодом по умолчанию открывались в PyCharm. Это удобно.

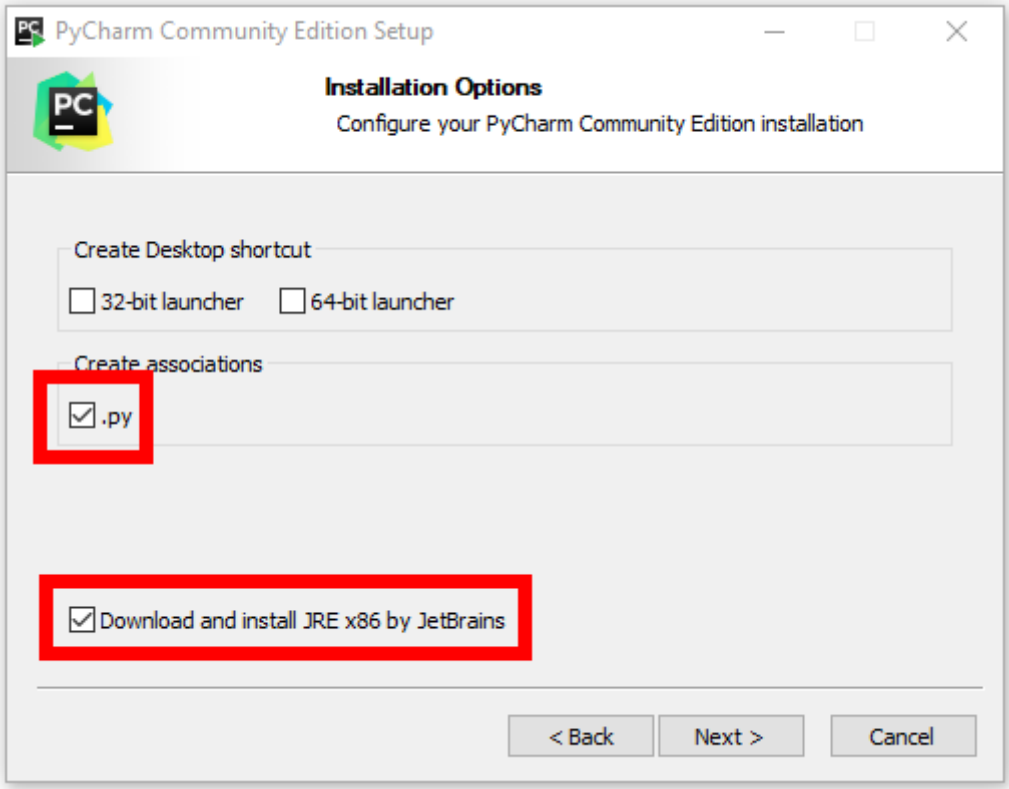

Рис.16. Окно настройки компонентов установки

После окончания установки запускайте PyCharm. После окна загрузки перед вами появится стартовое окно программы. Если поверх стартового окна у вас выскочило второе маленькое окошко, то просто нажмите «OK». Создайте новый проект, кликнув на «Create New Project».

В следующем окне будет два поля. В первом поле следует указать, где будет располагаться новый проект. Второе поле должно быть заполненным по умолчанию. В нем содержится путь к установленной ранее программе Python. Если там пусто, то укажите путь вручную.

Далее откроется окно самого редактора.

Теперь вы готовы начать писать программы на Python!

Ранее, при описании искусственных нейронных сетей мы проводили параллели с человеческим мозгом. Сейчас же мы будем рассматривать нейронную сеть как математическую функцию, которая отображает заданный вход в желаемый результат, не вникая в подробности.

Для начала, нам нужно инициализировать все действующие компоненты нашей сети

Первым делом импортируем Pybrain. Как говорилось ранее, Pybrain это библиотека языка Python, которая позволит нам использовать большие многомерные массивы и матрицы.

import pybrain

Импортируем scipy.special, -scipy содержит модули для оптимизации, интегрирования, специальных функций, обработки изображений и многих других задач, нам же здесь нужна наша функция активации, имя которой - "сигмоида " [26].

import scipy.special

Вероятно, нам понадобится визуализировать наши данные

import matplotlib.pyplot

Нейронные сети состоят из следующих компонентов:

1) входной слой, x
2) произвольное количество скрытых слоев
3) выходной слой, ŷ
4) набор весов и смещений между каждым слоем W и b
5) выбор функции активации для каждого скрытого слоя σ;

Ниже приведена архитектура двухслойной нейронной сети (рис. 17). (обычно, входной слой не входит в подсчет слоев нейронной сети).

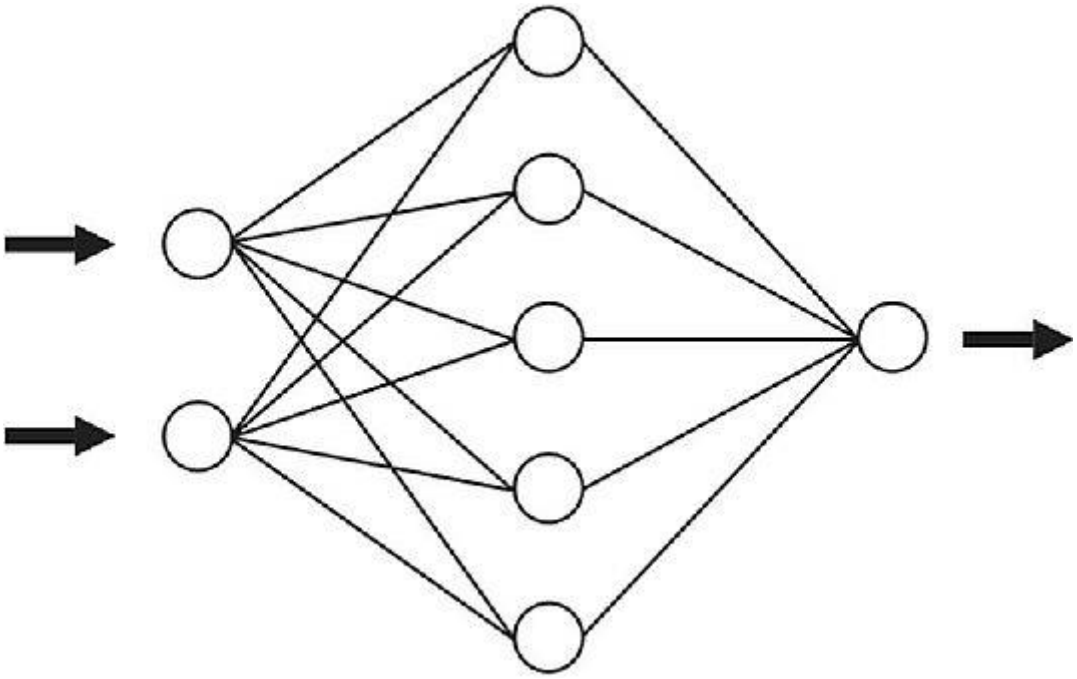

Рис. 17. Архитектура двухслойной нейронной сети

Создание нейронной сети с двумя входами, тремя скрытыми слоями и одним выходом:

```
>>> from pybrain.tools.shortcuts import buildNetwork
>>> net = buildNetwork(2, 3, 1)
```

В результате, в объекте net находится созданная нейронная цепь, инициализированная случайными значениями весов.

Функция активации

Функция активации задаётся следующим образом: net.activate([2, 1])

Количество элементов передаваемых в сеть должно быть равно количеству входов. Метод возвращает ответ в виде единственного числа, если текущая цепь имеет один выход, и массив, в случае большего количества выходов.

Получение данных о сети

Получение данных о сети

Для того, чтобы получить информацию о текущей структуре сети, каждый её элемент имеет имя. Данное имя может быть дано автоматически, либо по иным критериям при создании сети [27].

К примеру, для сети net имена даны автоматически:

```
>>> net['in']
<LinearLayer 'in'>
>>> net['hidden0']
<SigmoidLayer 'hidden0'>
>>> net['out']
<LinearLayer 'out'>
```

Скрытые слои поименованы с номером слоя добавленным к имени.

Оперирование данными (Building a DataSet)

Созданная сеть должна обрабатывать данные. Типичным набором данных является набор входных и выходных значений. Для работы с ними PyBrain использует модуль pybrain.dataset, также далее используется класс SupervisedDataSet.

Настройка данных

Класс SupervisedDataSet используется для типичного обучения с учителем. Он поддерживает массивы входных и выходных данных [28]. Их размеры задаются при создании экземпляра класса:

Записьвида:

```
>>> from pybrain.datasets import SupervisedDataSet
>>> ds = SupervisedDataSet(2, 1)
```

означает, что создаётся структура данных для хранения двухмерных входных данных и одномерных выходных.

Добавление образцов.

Классической задачей при обучении нейронной сети является обучение функции XOR, далее показан набор данных используемый для создания такой сети.

```
>>> ds.addSample((0, 0), (0,))

>>> ds.addSample((0, 1), (1,))

>>> ds.addSample((1, 0), (1,))

>>> ds.addSample((1, 1), (0,))
```

Исследование структуры образца

Для получения массивов данных в текущем их наборе возможно использовать стандартные функции Python для работы с массивами.

```
>>> len(ds)
```

выведет 4, так-как это количество элементов.

Итерация по множеству также может быть организована обычным для массивов способом:

```
>>> for inpt, target in ds:

print inpt, target

[ 0.  0.] [ 0.]

[ 0.  1.] [ 1.]

[ 1.  0.] [ 1.]

[ 1.  1.] [ 0.]
```

Также к каждому набору полей можно получить прямой доступ с использованием его имени:

```
>>> ds['input']
array([[ 0.,  0.],
[ 0.,  1.],
[ 1.,  0.],
[ 1.,  1.]])
>>> ds['target']
array([[ 0.],
[ 1.],
[ 1.],
[ 0.]])
```

Также можно вручную освободить занимаемую образцом память полностью его удалив:

```
>>> ds.clear()
>>> ds['input']
array([], shape=(0, 2), dtype=float64)
>>> ds['target']
array([], shape=(0, 1), dtype=float64)
```

Тренировка сети на образцах

В PyBrain использована концепция тренеров (trainers) для обучения сетей с учителем. Тренер получает экземпляр сети и экземпляр набора образцов и затем обучает сеть по полученному набору.

Классический пример это обратное распространение ошибки (backpropagation). Для упрощения реализации этого подход в PyBrain существует класс BackpropTrainer.

```
>>> from pybrain.supervised.trainers import BackpropTrainer
```

Обучающий набор образцов (ds) и целевая сеть (net) уже созданы в примерах выше, теперь они будут объединены.

```
>>> net = buildNetwork(2, 3, 1, bias=True, hiddenclass=TanhLayer)
```

```
>>> trainer = BackpropTrainer(net, ds)
```

Тренер получил ссылку на структуру сети и может её тренировать.

```
>>> trainer.train()
```

```
0.31516384514375834
```

Вызов метода train() производит одну итерацию (эпоху) обучения и возвращает значение квадратичной ошибки (double proportional to the error).

Если организовывать цикл по каждой эпохи нет надобности, то существует метод обучающий сеть до сходимости:

```
>>> trainer.trainUntilConvergence()
```

Данный метод возвращает массив ошибок для каждой эпохи.

3.4. Примерное тематическое планирование и содержание второй части курса

Предмет: информатика

Класс: 9

Образовательная область - Информатика

Количество часов: 22

Содержание курса

Вторая часть данного курса рассчитана на 22 часа (1 час в неделю в 9 классе), программой предусматривается последовательное изучение разделов:

1. Установка программы Python.
2. Литеральные константы, числа, строки,
3. Переменные, имена идентификаторов, типы данных, объекты
4. Операторы и выражения
5. Функции и параметры функций
6. Локальные переменные
7. Значения аргументов по умолчанию
8. Оператор “return”
9. Оператор from ... import ...
10. Модули
11. Создание собственных модулей
12. Пакеты
13. Списки

14. Краткое введение в объекты и классы

15. Классы

16. Методы объектов

17. Метод __init__

18. Переменные класса и объекта

19. Файлы

20. Ввод от пользователя

21. Функция активации.

22. Настройка данных.

23. Добавление образцов.

24. Тренировка сети на образцах.

25. Задача на распознавание рукописных цифр.

Цель изучения курса

Основная цель курса — приобретение знаний и умений в работе с нейронными сетями в программной среде Python

Образовательные

Получение представления о работе с нейронными сетями в программной среде программирования.

Совершенствование технологии обучения нейронной сети.

Формирование умений использовать искусственные нейронные сети в будущей профессиональной деятельности

Развивающие

Развитие способностей аналитического мышления.

Развитие представлений о новом подходе к обработке информации.

Развитие творческих способностей в процессе моделирования нейронных сетей.

Воспитательные

Воспитание ответственности за результаты своей работы в коллективе, адекватная оценка своего вклада в общее дело.

Воспитание отношения к информационной культуре и информационной безопасности.

Воспитание информационной культуре.

Тематическое планирование

Тематическое планирование второй части курса приведено в табл. 2

Таблица 2

Тематическое планирование второй части курса

№ п/п	**Название темы**	**Количество часов**
4	**Знакомство с языком программирования Python**	**12**
4.1	Установка программы Python.	1
4.2	Теоретический минимум знаний языка Python.	1

4.3	Литеральные константы, числа, строки, Переменные, имена идентификаторов, типы данных, объекты	1
4.4	Операторы и выражения	1
4.5	Функции и параметры функций	1
4.6	Локальные переменные Значения аргументов по умолчанию	1
4.7	Оператор “return” Оператор from ... import ...	1
4.8	Модули Создание собственных модулей	1
4.9	Пакеты	1
4.10	Списки	1
4.11	Краткое введение в объекты и классы Классы	1
4.12	Ввод от пользователя	1
5	**Основы создания искусственных нейронных сетей при помощи библиотеки PyBrain**	**8**

5.1	Создание искусственной нейронной сети.	2
5.2	Функция активации.	1
5.3	Настройка данных.	1
5.4	Добавление образцов.	2
5.5	Тренировка сети на образцах.	2
6	**Практическая работа с нейронными сетями**	**2**
6.1	Задача на распознавание рукописных цифр.	2
	ВСЕГО:	**22**

Планирование курса

Установка программы

Для того, чтобы начать работать с нейронными сетями на Python, необходимо установить его на наш компьютер.

1. Переходим на сайт https://www.python.org/downloads/ и скачиваем последнюю версию Python (левая кнопка). Кнопка справа предлагает скачать старую версию Python для Windows XP и других версий Windows, ниже приведенной версии. Программа Python для старых версий Windows не нужна, поэтому нажимаем на левую кнопку.
2. Запускаем скачанный .exe файл.

3. В первом открывшемся окне вас попросят выбрать способ установки: простой или сложный. Сначала включите нижний флажок «Add Python 3.6 to PATH». Эта функция позволит использовать Python в любом месте через консоль. Далее выбираете верхний (простой) способ установки.

4. По окончанию установки закройте окно установщика.

Теперь, когда компьютер умеет работать с кодами, написанными на языке Python, мы можем писать код даже в обычном блокноте, но к сожалению это очень неудобно, ведь мы уже привыкли к тому, чтобы определенные функции в коде подсвечивались цветом, а также к автоматическому указанию ошибок. Нам нужна умная среда.

Одна из таких программ— PyCharm от JetBrains. Эта компания специализируется на создании профессиональных сред для программирования. Программы от JetBrains занимают лидирующее положение среди программистов.

1. Переходим на сайт https://www.jetbrains.com/pycharm/ для скачивания PyCharm. Для скачивания доступно две версии: профессиональная и версия для сообщества. Версия для сообщества бесплатная. Ее и скачаем.

2. Запускаете скачанный .exe файл.

3. В первом окне вас приветствует сам установщик. Нажимаете на клавишу «Next». Далее следует указать место установки среды. Выбираем по умолчанию, как предлагает вам компьютер. Обычно это диск C:\.

4. В следующем окне обязательно ставьте галочку «Download and install JRE x86 by JetBrains», если на компьютере не установлена Java. Также следует поставить галочку «.py» для того, чтобы все файлы с программным кодом по умолчанию открывались в PyCharm. Это удобно.

5. После окончания установки запускайте PyCharm

6. После окна загрузки перед вами появится стартовое окно программы. Если поверх стартового окна у вас выскочило второе маленькое окошко, то просто нажмите «OK». Создайте новый проект, кликнув на «Create New Project».

7. В следующем окне будет два поля. В первом поле следует указать, где будет располагаться новый проект. Второе поле должно быть заполненным по умолчанию. В нем содержится путь к установленной ранее программе Python. Если там пусто, то укажите путь вручную.

8. Далее откроется окно самого редактора.

9. Теперь вы готовы начать писать программы на Python.

Программная платформа Python

Разобравшись в принципах работы с нейронными сетями на уже готовых онлайн платформах, мы можем начать создавать нейронные сети в более сложной программной среде, что даст нам намного больше возможностей. Теперь нейронные сети мы будем создавать в программе Python.

Python - интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой.

Именно он и поможет нам начать программировать нейронные сети. Обладая довольно не сложным синтаксисом и отдающий предпочтение читаемости кода, Python идеально подходит как и начинающим пользователям, так и людям, которые уже профессионально изучают нейронные сети. Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода. Интерпретатор Python и большая стандартная библиотека доступны бесплатно в виде исходных и исполняемых кодов для всех основных платформ и могут свободно распространяться.

Безусловно, для работы в Python требуется иметь базовое представление о программировании.

Знакомство с созданием и работой с простейшими ИНС показало, что для работы достаточно следующих конструкций и возможностей Python:

1. Литеральные константы, числа, строки,
2. Переменные, имена идентификаторов, типы данных, объекты
3. Операторы и выражения

4. Функции и параметры функций
5. Локальные переменные
6. Значения аргументов по умолчанию
7. Оператор “return”
8. Модули
9. Оператор from ... import ...
10. Создание собственных модулей
11. Пакеты
12. Списки
13. Краткое введение в объекты и классы
14. Классы
15. Методы объектов
16. Метод __init__
17. Переменные класса и объекта
18. Ввод от пользователя

Работа с нейронными сетями на Python

Ранее, при описании искусственных нейронных сетей мы проводили параллели с человеческим мозгом. Сейчас же мы будем рассматривать нейронную сеть как математическую функцию, которая отображает заданный вход в желаемый результат, не вникая в подробности.

Для начала, нам нужно инициализировать все действующие компоненты нашей сети

Первым делом импортируем Pybrain. Как говорилось ранее, Pybrain это библиотека языка Python, которая позволит нам использовать большие многомерные массивы и матрицы.

import pybrain

Импортируем scipy.special, -scipy содержит модули для оптимизации, интегрирования, специальных функций, обработки изображений и многих других задач, нам же здесь нужна наша функция активации, имя которой - "сигмоида "

import scipy.special

Вероятно, нам понадобится визуализировать наши данные

import matplotlib.pyplot

Нейронные сети состоят из следующих компонентов:

6) входной слой, x

7) произвольное количество скрытых слоев

8) выходной слой, ŷ

9) набор весов и смещений между каждым слоем W и b

10) выбор функции активации для каждого скрытого слоя σ;

Ниже приведена архитектура двухслойной нейронной сети (рис. 18). (обычно, входной слой не входит в подсчет слоев нейронной сети).

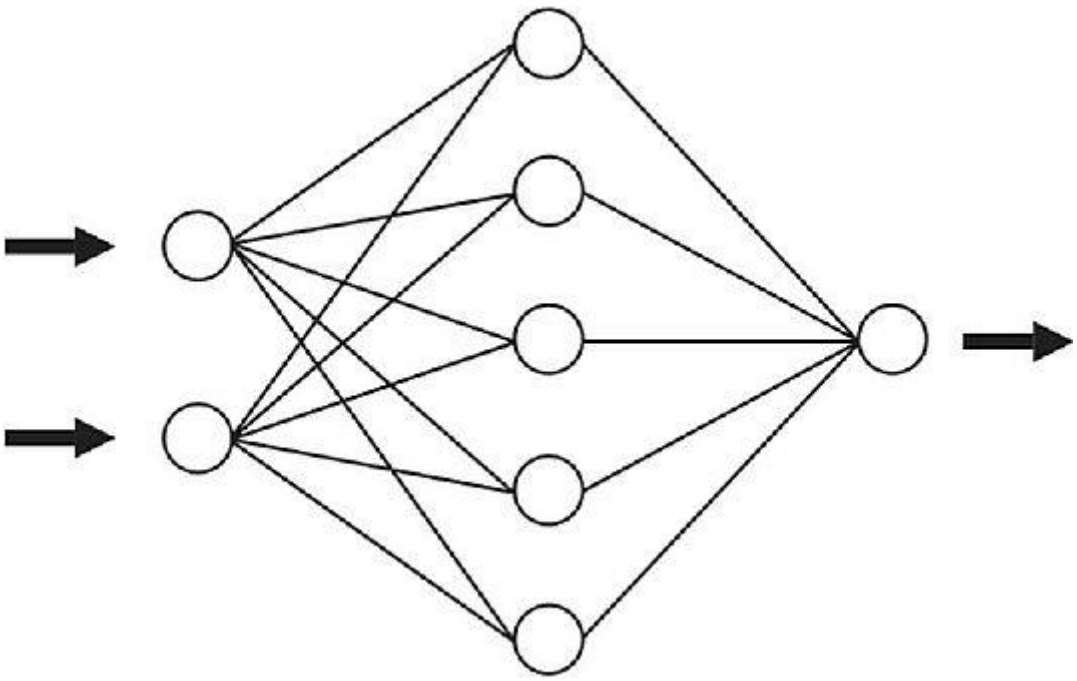

Рис. 18. Архитектура двухслойной нейронной сети

Создание нейронной сети с двумя входами, тремя скрытыми слоями и одним выходом:

```
>>> from pybrain.tools.shortcuts import buildNetwork
>>> net = buildNetwork(2, 3, 1)
```

В результате, в объекте net находится созданная нейронная цепь, инициализированная случайными значениями весов.

Функция активации

Функция активации задаётся следующим образом: net.activate([2, 1])

Количество элементов передаваемых в сеть должно быть равно количеству входов. Метод возвращает ответ в виде единственного числа, если текущая цепь имеет один выход, и массив, в случае большего количества выходов.

Получение данных о сети

Получение данных о сети

Для того, чтобы получить информацию о текущей структуре сети, каждый её элемент имеет имя. Данное имя может быть дано автоматически, либо по иным критериям при создании сети.

К примеру, для сети net имена даны автоматически:

```
>>> net['in']
<LinearLayer 'in'>
>>> net['hidden0']
<SigmoidLayer 'hidden0'>
>>> net['out']
<LinearLayer 'out'>
```

Скрытые слои поименованы с номером слоя добавленным к имени.

Оперирование данными (Building a DataSet)

Созданная сеть должна обрабатывать данные. Типичным набором данных является набор входных и выходных значений. Для работы с ними PyBrain использует модуль pybrain.dataset, также далее используется класс SupervisedDataSet.

Настройка данных

Класс SupervisedDataSet используется для типичного обучения с учителем. Он поддерживает массивы входных и выходных данных. Их размеры задаются при создании экземпляра класса:

Записьвида:

```
>>> from pybrain.datasets import SupervisedDataSet
>>> ds = SupervisedDataSet(2, 1)
```

означает, что создаётся структура данных для хранения двухмерных входных данных и одномерных выходных.

Добавление образцов.

Классической задачей при обучении нейронной сети является обучение функции XOR, далее показан набор данных используемый для создания такой сети.

```
>>> ds.addSample((0, 0), (0,))
>>> ds.addSample((0, 1), (1,))
>>> ds.addSample((1, 0), (1,))
>>> ds.addSample((1, 1), (0,))
```

Исследование структуры образца

Для получения массивов данных в текущем их наборе возможно использовать стандартные функции Python для работы с массивами.

```
>>> len(ds)
```

выведет 4, так-как это количество элементов.

Итерация по множеству также может быть организована обычным для массивов способом:

```
>>> for inpt, target in ds:
print inpt, target
[ 0.  0.] [ 0.]
[ 0.  1.] [ 1.]
[ 1.  0.] [ 1.]
[ 1.  1.] [ 0.]
```

Также к каждому набору полей можно получить прямой доступ с использованием его имени:

```
>>> ds['input']
array([[ 0.,  0.],
[ 0.,  1.],
[ 1.,  0.],
[ 1.,  1.]])
>>> ds['target']
array([[ 0.],
```

```
[ 1.],

[ 1.],

[ 0.]])
```

Также можно вручную освободить занимаемую образцом память полностью его удалив:

```
>>> ds.clear()

>>> ds['input']

array([], shape=(0, 2), dtype=float64)

>>> ds['target']

array([], shape=(0, 1), dtype=float64)
```

Тренировка сети на образцах

В PyBrain использована концепция тренеров (trainers) для обучения сетей с учителем. Тренер получает экземпляр сети и экземпляр набора образцов и затем обучает сеть по полученному набору.

Классический пример это обратное распространение ошибки (backpropagation). Для упрощения реализации этого подход в PyBrain существует класс BackpropTrainer.

```
>>> from pybrain.supervised.trainers import BackpropTrainer
```

Обучающий набор образцов (ds) и целевая сеть (net) уже созданы в примерах выше, теперь они будут объединены.

```
>>> net = buildNetwork(2, 3, 1, bias=True, hiddenclass=TanhLayer)

>>> trainer = BackpropTrainer(net, ds)
```

Тренер получил ссылку на структуру сети и может её тренировать.

>>> trainer.train()

0.31516384514375834

Вызов метода train() производит одну итерацию (эпоху) обучения и возвращает значение квадратичной ошибки (double proportional to the error).

Если организовывать цикл по каждой эпохи нет надобности, то существует метод обучающий сеть до сходимости:

>>> trainer.trainUntilConvergence()

Данный метод возвращает массив ошибок для каждой эпохи.

Пример решения задачи

Рассмотрим нейронную сеть, которая будет распознавать числа на входном изображении. Все очень просто: всего один слой и функция активации. Это не позволит нам распознать абсолютно все тестовые изображения, но мы справимся с подавляющим большинством. В качестве данных будем использовать известную в мире распознавания чисел подборку данных MNIST.

Для работы с ней в Python есть библиотека python-mnist. Чтобыустановить:

pip install python-mnist

Теперь можем загрузить данные

```
from mnist import MNIST
mndata = MNIST("/path_to_mnist_data_folder/")
tr_images, tr_labels = mndata.load_training()
test_images, test_labels = mndata.load_testing()
```

Архивы с данными нужно загрузить самостоятельно, а программе указать путь к каталогу с ними. Теперь переменные tr_images и test_images содержат изображения для тренировки сети и тестирования соответственно. А переменные

tr_labels и test_labels - метки с правильной классификацией (т.е. цифры с изображений). Все изображения имеют размер 28x28. Зададим переменную с размером.

```
img_shape = (28, 28)
```

Преобразуем все данные в массивы numpy и нормализуем их (приведем к размеру от -1 до 1). Этоувеличитточностьвычислений.

```
import numpy as np
for i in range(0, len(test_images)):
test_images[i] = np.array(test_images[i]) / 255
for i in range(0, len(tr_images)):
tr_images[i] = np.array(tr_images[i]) / 255
```

Следует отметить, что хоть и изображения принято представлять в виде двумерного массива мы будем использовать одномерный, это проще для вычислений.

Мы имеем на входе массив из 28*28=784 элементов и еще по 784 веса для определения каждой цифры. В процессе работы нейронной сети нужно перемножить значения входов на веса. Сложить полученные данные и добавить смещение. Полученный результат подать на функцию активации. В нашем случае это будет Relu. Эта функция равна нулю для всех отрицательных аргументов и аргументу для всех положительных.

Есть еще много функций активации! Но это же самая простая нейронная сеть! Определим эту функцию при помощи numpy

```
def relu(x):
return np.maximum(x, 0)
```

Теперь чтобы вычислить изображение на картинке нужно просчитать результат для 10 наборов коэффициентов.

```
def nn_calculate(img):
resp = list(range(0, 10))
for i in range(0,10):
```

```
r = w[:, i] * img
r = relu(np.sum(r) + b[i])
resp[i] = r
return np.argmax(resp)
```

Для каждого набора мы получим выходной результат. Выход с наибольшим результатом вероятнее всего и есть наше число.

В данном случае 7. Вот и все! Но нет... Ведь нужно эти самые коэффициенты где-то взять. Нужно обучить нашу нейронную сеть. Для этого применяют метод обратного распространения ошибки. Его суть в том чтобы рассчитать выходы сети, сравнить их с правильными, а затем отнять от коэффициентов числа необходимые, чтобы результат был правильным.

Нужно помнить, что для того чтобы вычислить эти значения нужна производная функции активации. В нашем случае она равна нулю для всех отрицательных чисел и 1 для всех положительных. Определим коэффициенты случайным образом.

```
w = (2*np.random.rand(10, 784) - 1) / 10
b = (2*np.random.rand(10) - 1) / 10
for n in range(len(tr_images)):
img = tr_images[n]
cls = tr_labels[n]
forward propagation
resp = np.zeros(10, dtype=np.float32)
for i in range(0,10):
r = w[i] * img
r = relu(np.sum(r) + b[i])
resp[i] = r
resp_cls = np.argmax(resp)
resp = np.zeros(10, dtype=np.float32)
resp[resp_cls] = 1.0
```

```
back propagation
true_resp = np.zeros(10, dtype=np.float32)
true_resp[cls] = 1.0
error = resp - true_resp
delta = error * ((resp >= 0) * np.ones(10))
for i in range(0,10):
w[i] -= np.dot(img, delta[i])
b[i] -= delta[i]
```

В процессе обучения коэффициенты станут слегка похожи на числа:

Проверим точность работы:

```
def nn_calculate(img):
resp = list(range(0, 10))
for i in range(0,10):
r = w[i] * img
r = np.maximum(np.sum(r) + b[i], 0) #relu
resp[i] = r
return np.argmax(resp)
total = len(test_images)
valid = 0
invalid = []
for i in range(0, total):
img = test_images[i]
predicted = nn_calculate(img)
true = test_labels[i]
if predicted == true:
valid = valid + 1
else:
invalid.append({"image":img, "predicted":predicted, "true":true})
print("accuracy {}".format(valid/total))
```

Получилось 88%. Не так уж много, но очень интересно.

ЗАКЛЮЧЕНИЕ

Таким образом, на сегодняшний день, обучение нейронным сетям является важной педагогической задачей. Стремительно развивающиеся технологии уже сейчас приводят к тому, что ручной труд повсеместно заменяют новейшими техническими разработками. Ручной труд становится все менее актуальным, как минимум в компьютерной среде развития. Что вызывает большой спрос на квалифицированных, шагающих в ногу со временем специалистов. Учитывая тот факт, что в школьной программе практически не затрагивается тема искусственных нейронных сетей, необходимо устранить отсутствие данной темы.

В рамках своей выпускной квалификационной работы я установила минимум знаний, как практических, так и теоретических, для первоначального освоения основ искусственных нейронных сетей.

Были выполнены задачи:

1. Собран теоретический минимум понятий, терминов и принципов работы искусственных нейронных сетей;
2. Проанализированы готовые онлайн платформы для первоначального освоения принципов работы нейронных сетей, выбран наиболее удобный онлайн компилятор, а также разработаны практические задания, позволяющие понять взаимосвязь основных компонентов построения нейронной сети;
3. Проанализированы программные платформы для продолжения изучения нейронных сетей, выбран наиболее подходящий и удобный программный продукт с инструкцией к установке;
4. Приведены основные принципы работы с нейронными сетями в Python, а также разобран пример решения задачи на определения изображения цифр.

Таким образом, в результате выполнения поставленных задач цель выполнена: рекомендации для школьного учителя, которые помогут ему построить курс первоначального обучения основам искусственных нейронных сетей разработаны.

ИСТОЧНИКИ И ЛИТЕРАТУРА

1. Искусственные нейронные сети (ИНС) [Электронный ресурс] URL: https://ru.wikipedia.org/wiki (дата обращения: 10.02.2019)

2. Простыми словами о сложном: что такое нейронные сети [Электронный ресурс]. URL: https://gagadget.com/another/27575-prostyimi-slovami-o-slozhnom-chto-takoe-nejronnyie-seti/(дата обращения: 24.02.2019)

3. Хайкин С. Нейронные сети: Полный курс. 2-ое-е изд. Издательский дом Вильямс, 2006. 1104 с.

4. Редозубов А.Д. Мозг - это не нейронные сети [Электронный ресурс]. URL: https://www.youtube.com/watch?v=7c6YUJ0JuqI&list=WL&t=1s&index=27 (дата обращения: 06.03.2019).

5. Основы ИНС [Электронный ресурс]. URL: https://neuralnet.info/chapter/(дата обращения:24.02.2019)

6. Многослойные нейронные сети прямого распространения [Электронный ресурс] URL: https://studfiles.net (дата обращения: 06.03.2019).

7. Нейронные сети. Часть 1. Основы искусственных нейронных сетей [Электронный ресурс]. URL: https://habr.com/ru/post/40137/(дата обращения:24.02.2019)

8. История возникновения ИНС [Электронный ресурс] URL: https://studfiles.net (дата обращения: 14.02.2019)

9. Краткий обзор популярных нейронных сетей [Электронный ресурс] URL: https://geektimes.ru/post/83781/ (дата обращения: 06.03.2019).

10. Кузубов А.О. Моделирование нейросетевой системы управления динамическим обьектом // Междунар. конф. «НАУКА НАСТОЯЩЕГО И БУДУЩЕГО». — Санкт-Петербург: ЛЭТИ СПб., 2017. — С. 100–101.

11. Демонстрация НС [Электронный ресурс] URL: http://primat.org/news/demo_onlajn_nejronnaja_set/2016-02-05-1118 (дата обращения: 24.02.2019)

12. Пощупать нейросети или конструктор нейронных сетей.[Электронный ресурс]. URL: https://habr.com/ru/post/351922/ (дата обращения 19.03.2019)

13. Фреймворк [Электронный ресурс] URL: https://ru.wikipedia.org/wiki (дата обращения: 15.02.2019)

14. Терехов С.А.. Лекции по теории и приложениям искусственных нейронных сетей. — 1999. [Электронный ресурс]. URL: http://alife.narod.ru/lectures/neural/Neu_ch08. (дата обращения: 06.03.2019).

15. Шпаргалка по разновидностям нейронных сетей. Часть первая. Элементарные конфигурации [Электронный ресурс] [2016]. URL: https://tproger.ru/translations/neural-network-zoo-1/ (дата обращения: 06.03.2019).

16. Нейронные сети: практическое применение.[Электронный ресурс]. URL: https://habr.com/ru/post/322392/ (дата обращения: 19.04.2019)

17. Сиротенко М.Ю. Применение нейросетей в распознавании изображений [Электронный ресурс]. URL: https://geektimes.ru/post/74326/ (дата обращения: 06.03.2019).

18. Нейронные сети для начинающих [Электронный ресурс]. URL: https://habr.com/ru/post/312450/(дата обращения: 24.02.2019)

19. Нейронные сети за 1 день.[Электронный ресурс]. URL: https://habrahabr.ru/post/320742/. (дата обращения: 19.04.2019)

20. Ефремова Н. Нейронные сети: практическое применение [Электронный ресурс] URL: https://www.youtube.com/watch?v=dCf9qyDHzeI (дата обращения: 06.03.2019).

21. Бум нейросетей: Кто делает нейронные сети, зачем они нужны и сколько денег могут приносить. [Электронный ресурс] URL: https://vc.ru/future/16843-neural-networks (дата обращения: 19.02.2019)

22. Заенцев И. В. Нейронные сети: основные модели / И. В. Заенцев // Воронеж: Изд-во Воронежского госуд. ун-та, 1999. — 76 с.

23. Голованов Вячеслав. Нейросеть в 11 строчек на Python. — 2015.[Электронный ресурс]. URL: https: //habrahabr.ru/post/271563/. (дата обращения: 19.02.2019)

24. Созыкин А. Введение | Глубокие нейронные сети на Python [Электронный ресурс]. URL: https://www.youtube.com/watch?v=GX7qxV5nh5o&index=1&list=PLtPJ9lKvJ4oiz9aaL_xcZd-x0qd8G0VN_ (дата обращения: 06.03.2019).

25. Работаем с нейронными сетями на Python [Электронный ресурс] URL: https://www.pvsm.ru/python/11986#begin (дата обращения: 24.02.2019)

26. Методы обучения искусственных нейронных сетей [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/metody-obucheniya-iskusstvennyh-neyronnyh-setey/ (дата обращения:24.02.2019)

27. Осипов Павел. PyBrain работаем с нейронными сетями на Python. — 2015.[Электронный ресурс]. URL: https://habrahabr.ru/post/148407/. (дата обращения 19.03.2019)

28. Россум Г, Дрейк ФЛ Дж, Откидач ДС. Язык программирования Python. — М.: Вильяме, 2001. — 454 с.