МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА 41

ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | М. В. Савинов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ |
| АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ |
| по дисциплине: ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ, СЕТИ И ТЕЛЕКОММУНИКАЦИИ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М021 |  |  |  | П. А. Носуленко |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

Оглавление

[ВВЕДЕНИЕ 3](#_Toc120816108)

[1. Характеристика и объекты исследования 4](#_Toc120816109)

[1.1 Используемые технологии для исследования 4](#_Toc120816110)

[1.2 Основные понятия нейронных сетей 4](#_Toc120816113)

[1.2.1 Структура искусственного нейрона 4](#_Toc120816114)

[1.2.2 Функция активации 5](#_Toc120816120)

[1.2.3 Слои нейронных сетей 7](#_Toc120816132)

[1.3 Архитектуры нейронных сетей 9](#_Toc120816139)

[1.3.1 Нейронные сети прямого распространения и перцептроны 9](#_Toc120816140)

[1.3.2 Сеть Хоперфилда 10](#_Toc120816147)

[1.3.3 Автокодировщик 12](#_Toc120816160)

[1.3.4 Сверточные нейронные сети 16](#_Toc120816183)

[1.3.5 Генеративно-состязательная нейронная сеть 18](#_Toc120816199)

[1.3.6 Рекуррентные нейронные сети 19](#_Toc120816205)

[1.4 Разработка нейронных сетей 22](#_Toc120816228)

[1.4.1 Описание ИТ и ПО, используемых для создания нейронной сети 22](#_Toc120816229)

[1.4.2 Описание библиотеки Scikit-learn 22](#_Toc120816230)

[2. Разработка нейронной сети 24](#_Toc120816231)

[2.1 Описание разработки 24](#_Toc120816232)

[2.2 Код программы 25](#_Toc120816233)

[2.3 Результат работы программы 28](#_Toc120816234)

[ЗАКЛЮЧЕНИЕ 29](#_Toc120816235)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc120816236)

## ВВЕДЕНИЕ

Тема искусственного интеллекта популярна уже не первое десятилетие, но в последнее время изучение этого вопроса находит широкое распространение. На сегодняшний день существует огромное число разнообразных нейронных сетей, на решение самых разных проблем начиная с предсказывания ситуации на рынке и заканчивая рисованием красивых картин.

Нейронные сети, или, точнее, искусственные нейронные сети, представляют собой технологию, уходящую корнями во множество дисциплин: нейрофизиологию, математику, статистику, физику, компьютерные науки и технику. Они находят свое применение в таких разнородных областях, как моделирование, анализ временных рядов, распознавание образов, обработка сигналов и управление благодаря одному важному свойству - способности обучаться на основе данных при участии учителя или без его вмешательства.

Для создания искусственного интеллекта необходимо разработать модель поведения как у живого организма, способного учиться. На сегодняшний день существует большое количество архитектур нейронных сетей под самые разные задачи.

Целью данной исследовательской работы является изучение существующих архитектур и нейронных сетей и возможности их применения.

Для достижения цели необходимо решить следующие задачи:

* Разбор основных понятий в разработке нейронных сетей
* Изучение архитектур нейронных сетей
* Определение областей применения этих архитектур
* Разработать собственную нейронную сеть

Объектом исследования являются нейронные сети. Предметом исследования являются архитектуры нейронных сетей.

## Характеристика и объекты исследования

## 1.1 Используемые технологии для исследования

В исследовательской работе будут рассмотрены технологии нейронных сетей и их архитектур.

Исследования по искусственным нейронным сетям связаны с тем, что способ обработки информации человеческим мозгом в корне отличается от методов, применяемых обычными цифровыми компьютерами. Мозг представляет собой чрезвычайно сложный, нелинейный, параллельный компьютер. Он обладает способностью организовывать свои структурные компоненты, называемые нейронами, так, чтобы они могли выполнять конкретные задачи (такие как распознавание образов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые быстродействующие современные компьютеры.

## 1.2 Основные понятия нейронных сетей

## 1.2.1 Структура искусственного нейрона

Все искусственные нейронные сети состоят из так называемых нейронов — модели, представляющей из себя пороговую величину, и связей между нейронами — синапсами.

Понятие развития нейронов связано с понятием пластичности мозга - способности настройки нервной системы в соответствии с окружающими условиями. Именно пластичность играет самую важную роль в работе нейронов в качестве единиц обработки информации в человеческом мозге. Аналогично, в искусственных нейронных сетях работа проводится с искусственными нейронами. В общем случае нейронная сеть представляет собой машину, моделирующую способ обработки мозгом конкретной задачи. Эта сеть обычно реализуется с помощью электронных компонентов или моделируется программой, выполняемой на цифровом компьютере.

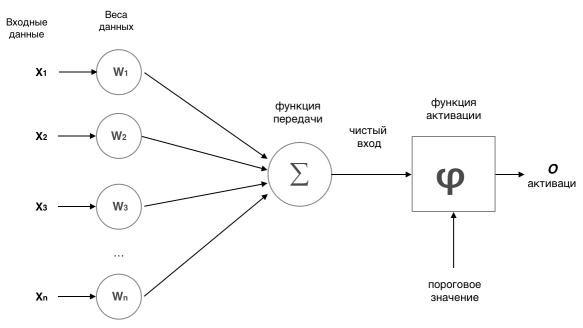


Рисунок 1 – Модель искусственного нейрона

Из Рисунка 1 можно получить общее представление о работе искусственного нейрона. На вход поступает некоторое количество сигналов Х, которое заранее определено архитектором нейронной сети по критериям отбора, связанными с типом решаемой задачи. Эти сигналы умножаются на веса W и суммируются.

## 1.2.2 Функция активации

После, функция активации преобразует полученные данные в своего рода ответ. Функция активации используется как орган принятия решений на выходе нейрона. Нейрон изучает линейные или нелинейные границы принятия решений на основе функции активации. Он также оказывает нормализующее влияние на выход нейронов, что предотвращает выход нейронов после нескольких слоев, чтобы стать очень большим, за счет каскадного эффекта. Функции активации могут быть самыми разными:

* – линейная функция

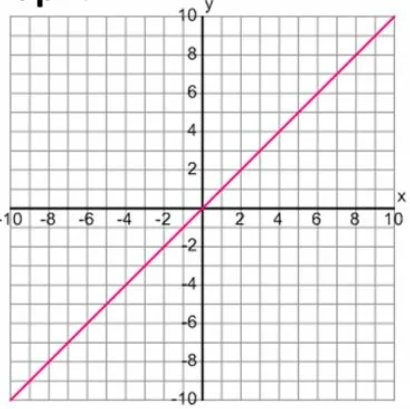


Рисунок 2 – график линейной функции

* – сигмоид

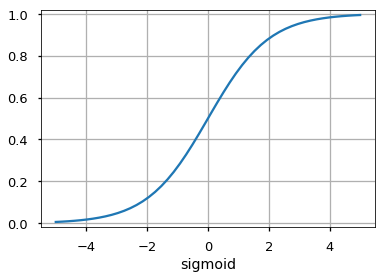


Рисунок 3 – график сигмоида

* – гиперболистический тангенс

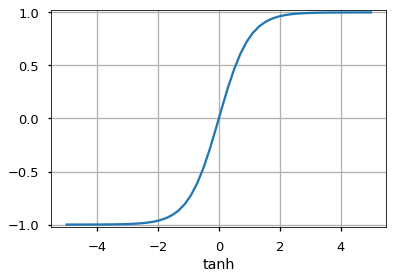


Рисунок 4 - график гиперболистического тангенса

Таким образом, путем объединения нескольких нейронов и создается нейронная сеть.

## 1.2.3 Слои нейронных сетей

Каждая нейронная сеть состоит из нескольких слоев нейронов, где каждый уровень отвечает за свою функцию. Это может быть распознавание цвета, формы, силы звука, объема и т.п. Нейроны одного слоя между собой не связаны, при этом каждый нейрон этого слоя связан с каждым нейроном соседнего слоя.

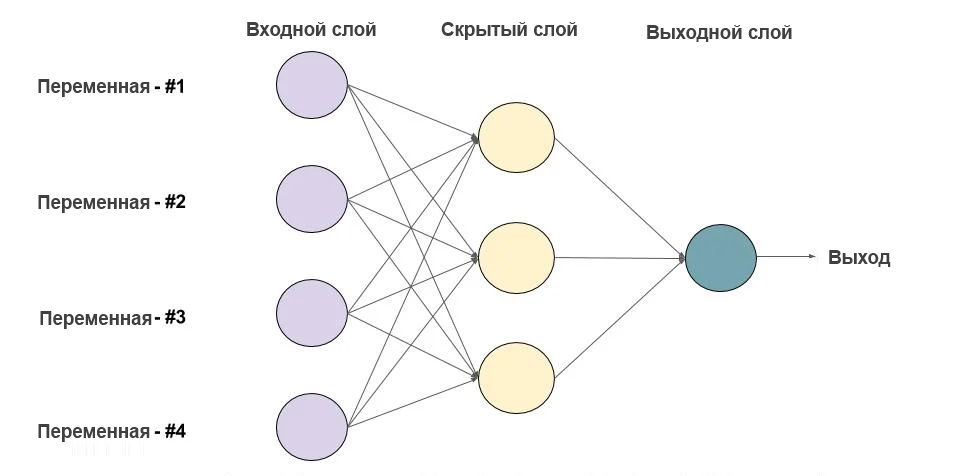


Рисунок 5 – слои нейронных сетей

Входной слой - Это первый слой нейронной сети. Он используется для передачи и приёма входных данных или функций в сеть.

Скрытый слой – это внутренний слой нейронной сети. Имея несколько скрытых слоев, нейронная сеть способна вычислять сложные функции, каскадируя более простые функции. Наиболее широко используемый скрытый блок — это тот, где функция активации использует выпрямленный линейный блок (ReLU). Выбор скрытых слоёв — очень активная область исследований в машинном обучении. Количество скрытых слоев называется глубиной нейронной сети. Чем больше глубина нейронной сети, тем более сложные функции она способна вычислять.

Выходной слой – это слой, который выдает прогнозы. Функция активации, используемая на этом уровне, различается для разных задач. Так, для задачи двоичной классификации необходимо, чтобы на выходе было либо 0, либо 1. Таким образом, используется сигмовидная функция активации. Для задачи мультиклассовой классификации используется Softmax. Для задачи регрессии, когда результат не является предопределенной категорией, можно просто использовать линейную единицу.

## 1.3 Архитектуры нейронных сетей

## 1.3.1 Нейронные сети прямого распространения и перцептроны

Сети прямого распространения (Feed forward neural networks, FF or FFNN) и перцептроны (perceptrons, P) очень просты — они передают информацию от входа к выходу.



Рисунок 6 – Архитектура FFNN

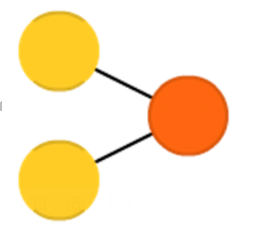


Рисунок 7 – Архитектура прецептрона

Простейшая мало-мальски рабочая сеть состоит из двух входных и одного выходного нейрона и может моделировать логический вентиль — базовый элемент цифровой схемы, выполняющий элементарную логическую операцию. FFNN обычно обучают методом обратного распространения ошибки, подавая модели на вход пары входных и ожидаемых выходных данных. Под ошибкой обычно понимаются различные степени отклонения выходных данных от исходных (например, среднеквадратичное отклонение или сумма модулей разностей). При условии, что сеть обладает достаточным количеством скрытых нейронов, теоретически она всегда сможет установить связь между входными и выходными данными. На практике использование сетей прямого распространения ограничено, и чаще они используются совместно с другими сетями, но даже такая простая модель способна решить задачу классификации и сформировать базовый логический элемент, например, исключающее ИЛИ.

## 1.3.2 Сеть Хоперфилда

Нейронная сеть Хопфилда (Hopfield network, HN) — это полносвязная нейронная сеть с симметричной матрицей связей. Во время получения входных данных каждый узел является входом, в процессе обучения он становится скрытым, а затем становится выходом.

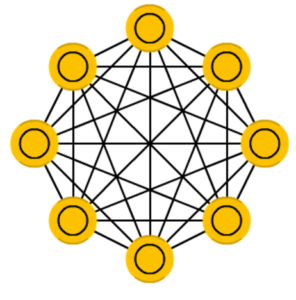


Рисунок 8 – Архитектура сети Хопфилда

Сеть Хопфилда является однослойной, каждый из нейронов связан со всеми остальными, но не воздействует на себя самого. В качестве функции активации нейронов сети Хопфилда используется пороговая функция:

Процедура обучения для одного входного образца (процедура сохранения образца) представляет из себя просто расчет весовых коэффициентов по вполне конкретной формуле:

Где - это матрица весов, - входной образец, а - транспонированный вектор. Данный процесс происходит всего лишь в один этап. После того, как сеть обучилась на одном или нескольких шаблонах, она всегда будет сводиться к одному из них (но не всегда — к желаемому). Она стабилизируется в зависимости от общей «энергии» и «температуры» сети. У каждого нейрона есть свой порог активации, зависящий от температуры, при прохождении которого нейрон принимает одно из двух значений (обычно -1 или 1, иногда 0 или 1).

Для успешной работы сети необходимо, чтобы нейроны перешли в некое устойчивое состояние, собственно, отсюда и вытекает вопрос – а будет ли сеть вообще устойчива? И здесь нельзя не рассмотреть два возможных режима работы:

* + - 1. Синхронный режим. Идея заключается в том, что расчеты для всех нейронов производятся последовательно, то есть по очереди. Но их состояния не меняются, а запоминаются. И лишь после того, как были пройдены все нейроны сети, их состояния одновременно изменяются на новые.
      2. Асинхронный режим. Здесь отличие заключается в том, что состояния нейронов меняются не одновременно (не синхронно). То есть произвели расчеты для первого нейрона, в соответствии с чем установили его новое состояние. Далее производим вычисления для второго нейрона, но уже с учетом нового состояния первого. И также вслед за этим меняем состояние второго. И далее все по цепочке аналогично.

На практике в подавляющем числе случаев используется 2-й вариант, а именно асинхронный режим. И, возвращаясь к вопросу устойчивости, нейронная сеть Хопфилда, работающая в асинхронном режиме, всегда будет устойчива, что замечательно и безгранично радует. Сети Хопфилда сеть часто называются сетями с ассоциативной памятью; как человек, видя половину таблицы, может представить вторую половину таблицы, так и эта сеть, получая таблицу, наполовину зашумленную, восстанавливает её до полной.

## 1.3.3 Автокодировщик

Автокодировщики или автоэнкодер (autoencoder, AE) - это особый тип нейронных сетей с прямой связью, где входные данные совпадают с выходными. Они сжимают вход в низкоразмерный код, а затем востанавливают выход из этого представления. Код представляет собой компактную «сводку» или «сжатие» ввода, также называемую представление в скрытом пространстве.

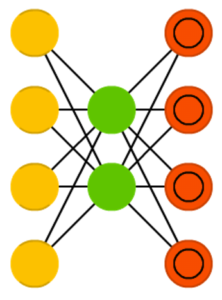


Рисунок 9 – Архитектура автоэнкодера

Автоэнкодер состоит из 3 компонентов: кодер, код и декодер. Кодер сжимает ввод и производит код, затем декодер восстанавливает ввод только с использованием этого кода.

Есть 4 гиперпараметра, которые нам нужно установить перед тренировкой автоэнкодера:

* Размер кода: количество узлов на среднем уровне. Меньший размер приводит к большему сжатию.
* Количество слоев: автоэнкодер может быть настолько глубоким, насколько это необходимо.
* Количество узлов на слой: архитектура автокодера, называется сложенный автоэнкодер, так как слои укладываются один за другим. Обычно сложенные автоэнкодеры выглядят как «сэндвич». Количество узлов на уровень уменьшается с каждым последующим уровнем кодера и увеличивается обратно в декодер. Также декодер обычно симметричен кодеру с точки зрения структуры слоя.
* Функция потери: либо используется среднеквадратическая ошибка или бинарная кроссентропия. Если входные значения находятся в диапазоне [0, 1], то обычно используется кроссцентропия, в противном случае используется среднеквадратичная ошибка.

В зависимости от поставленной задачи, может потребоваться иная архитектура автоэнкодера:

1. Разреженный автокодировщик (sparse autoencoder, SAE) — в каком-то смысле противоположность обычного. Вместо того, чтобы обучать сеть отображать информацию в меньшем «объёме» узлов, мы увеличиваем их количество.

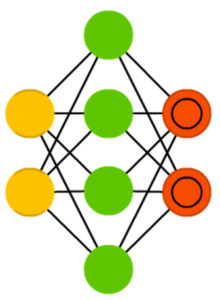


Рисунок 10 – Архитектура разреженного автокодировщика

Вместо того, чтобы сужаться к центру, сеть там раздувается. Сети такого типа полезны для работы с большим количеством мелких свойств набора данных. Если обучать сеть как обычный автокодировщик, ничего полезного не выйдет. Поэтому кроме входных данных подаётся ещё и специальный фильтр разреженности, который пропускает только определённые ошибки.

1. Вариационные автокодировщики (variational autoencoder, VAE) обладают схожей с AE архитектурой, но обучают их иному: приближению вероятностного распределения входных образцов. В этом они берут начало от машин Больцмана.

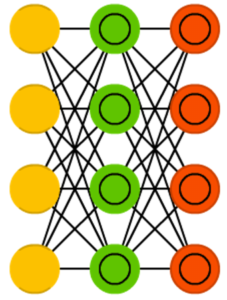


Рисунок 11 – Архитектура вариационного автокодировщика

Тем не менее, они опираются на байесовскую математику, когда речь идёт о вероятностных выводах и независимости, которые интуитивно понятны, но сложны в реализации. Если обобщить, то можно сказать что эта сеть принимает в расчёт влияния нейронов. Если что-то одно происходит в одном месте, а что-то другое — в другом, то эти события не обязательно связаны, и это должно учитываться.

1. Шумоподавляющие автокодировщики (denoising autoencoder, DAE) — это автокодировщики, в которые входные данные подаются в зашумленном состоянии. Ошибка вычисляется так же, и выходные данные сравниваются с зашумленными. Благодаря этому сеть учится обращать внимание на более широкие свойства, поскольку маленькие могут изменяться вместе с шумом.

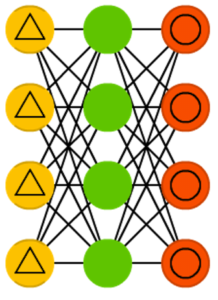


Рисунок 12 – Архитектура шумоподавляющих автокодировщиков

Автоэнкодеры используют для:

* Выявление скрытых зависимостей в исходных данных (автокодировщик используется в качестве первого скрытого слоя другой нейронной сети)
* Уменьшение шума (помех) в исходных данных
* Понижение размерности для визуализации многомерных данных
* Рекомендационные сервисы (используются глубокие автокодировщики, чтобы понять пользовательские предпочтения, порекомендовать фильмы, книги или предметы)

## 1.3.4 Сверточные нейронные сети

Сверточная нейронная сеть (convolutional neural network, CNN) — специальная архитектура нейронных сетей, изначально нацеленная на эффективное распознавание изображений.

Свертка — операция над парой матриц (размера ×) и (размера ×), результатом которой является матрица размера (. Каждый элемент результата вычисляется как скалярное произведение матрицы и некоторой подматрицы такого же размера (подматрица определяется положением элемента в результате). То есть, .

В сверточной нейронной сети выходы промежуточных слоев образуют матрицу (изображение) или набор матриц (несколько слоёв изображения). Так, например, на вход сверточной нейронной сети можно подавать три слоя изображения (R-, G-, B-каналы изображения). Основными видами слоев в сверточной нейронной сети являются сверточные слои, пулинговые слои и полносвязные слои.

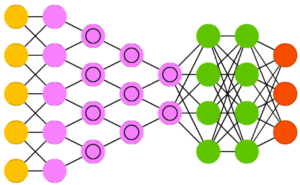


Рисунок 13 – Архитектура сверточной нейронной сети

Сверточный слой нейронной сети представляет из себя применение операции свертки к выходам с предыдущего слоя, где веса ядра свертки являются обучаемыми параметрами. Еще один обучаемый вес используется в качестве константного сдвига (англ. bias). При этом есть несколько важных деталей:

* В одном сверточном слое может быть несколько сверток. В этом случае для каждой свертки на выходе получится своё изображение. Например, если вход имел размерность , а в слое было n сверток с ядром размерности , то выход будет иметь размерность ;
* Ядра свертки могут быть трёхмерными. Свертка трехмерного входа с трехмерным ядром происходит аналогично, просто скалярное произведение считается еще и по всем слоям изображения. Например, для усреднения информации о цветах исходного изображения, на первом слое можно использовать свертку размерности . На выходе такого слоя будет уже одно изображение (вместо трёх);
* Можно заметить, что применение операции свертки уменьшает изображение. Также пиксели, которые находятся на границе изображения участвуют в меньшем количестве сверток, чем внутренние. В связи с этим в сверточных слоях используется дополнение изображения (англ. padding). Выходы с предыдущего слоя дополняются пикселями так, чтобы после свертки сохранился размер изображения. Такие свертки называют одинаковыми (англ. same convolution), а свертки без дополнения изображения называются правильными (англ. valid convolution).
* Еще одним параметром сверточного слоя является сдвиг (англ. stride). Хоть обычно свертка применяется подряд для каждого пикселя, иногда используется сдвиг, отличный от единицы — скалярное произведение считается не со всеми возможными положениями ядра, а только с положениями, кратными некоторому сдвигу s. Тогда, если если вход имел размерность , а ядро свертки имело размерность и использовался сдвиг s, то выход будет иметь размерность:

Пулинговый слой призван снижать размерность изображения. Исходное изображение делится на блоки и для каждого блока вычисляется некоторая функция. Чаще всего используется функция максимума или (взвешенного) среднего. Обучаемых параметров у этого слоя нет. Основные цели пулингового слоя:

* уменьшение изображения, чтобы последующие свертки оперировали над большей областью исходного изображения;
* увеличение инвариантности выхода сети по отношению к малому переносу входа;
* ускорение вычислений.

Обычно сверточные нейронные сети используются для обработки изображений, реже для аудио. Типичным способом применения CNN является классификация изображений: если на изображении есть кошка, сеть выдаст «кошка», если есть собака — «собака». Такие сети обычно используют «сканер», не парсящий все данные за один раз.

## 1.3.5 Генеративно-состязательная нейронная сеть

Генеративно-состязательная нейросеть (Generative adversarial network, GAN) — архитектура, состоящая из генератора и дискриминатора, настроенных на работу друг против друга. Отсюда GAN и получила название генеративно-созтязательная. По сути они состоит их двух нейронных сетей, одна из которых создает, а другая оценивает.

Дискриминационные алгоритмы пытаются классифицировать входные данные. Учитывая особенности полученных данных, они стараются определить категорию, к которой они относятся.

К примеру, пробегая все слова в письме дискриминационный алгоритм может предсказать, является сообщение спамом или не спамом. Спам — это категория, а пакет слов, собранный из электронной почты — образы, которые составляют входные данные. Математически категории обозначают y, а образы обозначают x. Запись p(y|x) используется для обозначения «вероятности y при заданном x», которая обозначает «вероятность того, что электронное письмо является спамом при имеющемся наборе слов». Таким образом, дискриминационные функции сопоставляют образы с категорией. Они заняты только этой корреляцией.

Таким образом, одна нейронная сеть, называемая генератором, генерирует новые экземпляры данных, а другая — дискриминатор, оценивает их на подлинность; т.е. дискриминатор решает, относится ли каждый экземпляр данных, который он рассматривает, к набору тренировочных данных или нет.

GAN имитируют любое распределение данных. Их обучают создавать структуры, устрашающе похожие на сущности из нашего мира в области изображений, музыки, речи, прозы. Генеративно-состязательные сети, в некотором смысле, роботы-художники, и результат их работы впечатляет.

## 1.3.6 Рекуррентные нейронные сети

Рекуррентные нейронные сети (Recurrent Neural Networks, RNNs) — популярные модели, используемые в обработке естественного языка. Во-первых, они оценивают произвольные предложения на основе того, насколько часто они встречались в текстах. Это дает меру грамматической и семантической корректности. Такие модели используются в машинном переводе. Во-вторых, языковые модели генерируют новый текст. Обучение модели на поэмах Шекспира позволит генерировать новый текст, похожий на Шекспира. Порядок, в котором подаются данные и обучается сеть, в рекуррентных сетях важен. Большой сложностью сетей RNN является проблема исчезающего (или взрывного) градиента, которая заключается в быстрой потере информации с течением времени.

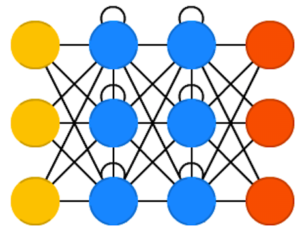


Рисунок 14 – Архитектура рекуррентных нейронных сетей

Идея RNN заключается в последовательном использовании информации. В традиционных нейронных сетях подразумевается, что все входы и выходы независимы. Но для многих задач это не подходит. Если необходимо предсказать следующее слово в предложении, лучше учитывать предшествующие ему слова. RNN называются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности, причем выход зависит от предыдущих вычислений. Еще одна интерпретация RNN: это сети, у которых есть «память», которая учитывает предшествующую информацию. Теоретически RNN могут использовать информацию в произвольно длинных последовательностях, но на практике они ограничены лишь несколькими шагами.



Рисунок 15 – Рекуррентная нейронная сеть и ее развертка

На диаграмме выше показано, что RNN разворачивается в полную сеть. С помощью развертки программа просто выписывает сеть для полной последовательности. Например, если последовательность представляет собой предложение из 5 слов, развертка будет состоять из 5 слоев, по слою на каждое слово. Формулы, задающие вычисления в RNN следующие:

* xt — вход на временном шаге t. Например, x\_1 может быть вектором с одним горячим состоянием (one-hot vector), соответствующим второму слову предложения.
* st — это скрытое состояние на шаге t. Это «память» сети. st зависит, как функция, от предыдущих состояний и текущего входа x\_t: st=f(Uxt+Ws(t-1)). Функция f обычно нелинейная, например, tanh или ReLU. st-1, которое требуется для вычисление первого скрытого состояния, обычно инициализируется нулем (нулевым вектором).
* ot — выход на шаге t.

Рекуррентные нейронные сети используются, когда важно соблюдать последовательность, когда важен порядок поступающих объектов.

1. Обработка текста на естественном языке:

* Анализ текста;
* Автоматический перевод;

1. Обработка аудио:

* Автоматическое распознавание речи;

1. Обработка видео:

* Прогнозирование следующего кадра на основе предыдущих;
* Распознавание эмоций;

1. Обработка изображений:

* Прогнозирование следующего пикселя на основе окружения;
* Генерация описания изображений.

## 1.4 Разработка нейронных сетей

## 1.4.1 Описание ИТ и ПО, используемых для создания нейронной сети

PyCharm – интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA [].

Разработка нейронной сети велась на языке Python.

Python - Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

## 1.4.2 Описание библиотеки Scikit-learn

Scikit-learn (или sklearn) – это бесплатная библиотека программного обеспечения для машинного обучения для языка программирования Python. Она включает различные алгоритмы классификации, регрессии и кластеризации, включая методы опорных векторов, случайные леса, повышение градиента, k-средние и DBSCAN, и предназначена для взаимодействия с числовыми и научными библиотеками Python numpy и scipy.

## 2. Разработка нейронной сети

## 2.1 Описание разработки

Задача нейронной сети заключается в определении классификации вида цветка ириса как virginica, setosa или versicolor на основе длины педали, высоты педали, длины чашелистника и высоты чашелистника. Для этого необходимо будет использовать модель логистической регрессии.

Логистическая регрессия - это модель, которая использует логистическую функцию для моделирования зависимой переменной. Как и любой регрессионный анализ, логистическая регрессия является прогнозным. Логистическая регрессия используется для описания данных и объяснения взаимосвязи между одной зависимой переменной и одной или несколькими номинальными, порядковыми, интервальными или пропорциональными независимыми переменными.

Функцией активации будет следующая формула:

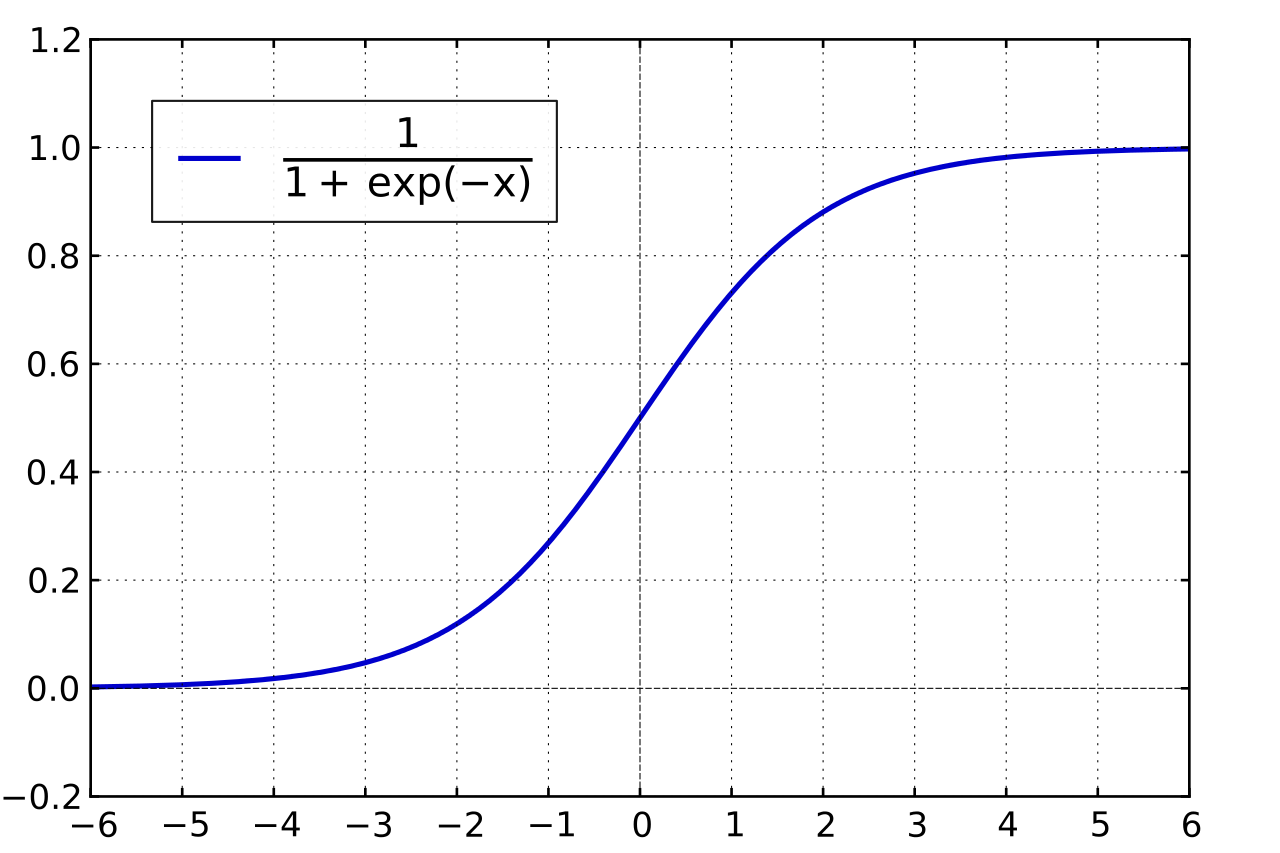


Рисунок 16 – график функции активации

## 2.2 Код программы

Листинг файла main.py:

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from iris import show\_data, train\_model, test\_model

data = sns.load\_dataset("iris")

# X = feature values, all the columns except the last column

X = data.iloc[:, :-1]

# y = target values, last column of the data frame

y = data.iloc[:, -1]

show\_data(data)

#Split the data into 80% training and 20% testing

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#Training model

model = train\_model(x\_train, y\_train)

#Testing model

test\_model(model, x\_test, y\_test)

#Enter Iris data here:

iris = [[7.9, 3.1, 7.5, 1.8],

[3.4, 5.1, 1.6, 6.2],

[5.1, 2.2, 9.8, 3.5]]

x = []

for i in range(4):

x.append(float(input()))

iris.append(x)

ans = model.predict(iris)

print(ans)

Листинг файла iris.py

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

def show\_data(data):

pltX = data.loc[:, 'sepal\_length']

pltY = data.loc[:, 'species']

plt.scatter(pltX, pltY, color='blue', label='sepal\_length')

pltX = data.loc[:, 'sepal\_width']

pltY = data.loc[:, 'species']

plt.scatter(pltX, pltY, color='green', label='sepal\_width')

pltX = data.loc[:, 'petal\_length']

pltY = data.loc[:, 'species']

plt.scatter(pltX, pltY, color='red', label='petal\_length')

pltX = data.loc[:, 'petal\_width']

pltY = data.loc[:, 'species']

plt.scatter(pltX, pltY, color='black', label='petal\_width')

plt.legend(loc=4, prop={'size': 8})

plt.show()

def train\_model(x\_train, y\_train):

# Train the model

model = LogisticRegression(max\_iter=150)

model.fit(x\_train.values, y\_train.values)# Training the model

return model

def test\_model(model, x\_test, y\_test):

# Test the model

predictions = model.predict(x\_test)

# Check precision, recall, f1-score

print(classification\_report(y\_test, predictions))

print(accuracy\_score(y\_test, predictions))

## 2.3 Результат работы программы

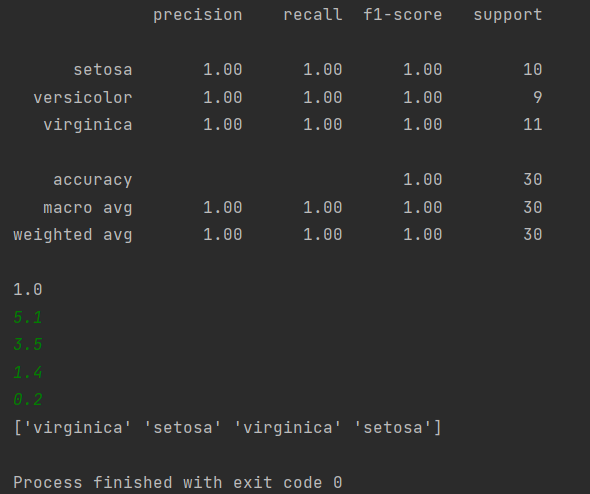


Рисунок 17 – Скрин работы программы

Как можно увидеть на рисунке 17, программа работает исправно, а точность предсказаний очень высокая.

## ЗАКЛЮЧЕНИЕ

Таким образом, в ходе исследовательской работы были выполнены следующие задачи:

* разобраны основные понятия в разработке нейронных сетей
* изучены архитектуры нейронных сетей
* определены области применения этих архитектур
* разработана собственная нейронная сеть

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. THE NEURAL NETWORK ZOO – [статья] – URL: https://www.asimovinstitute.org/neural-network-zoo/ – Текст: электронный.
2. Н. С. Москалев. Виды архитектур нейронных сетей – Молодой учёный №29, 2016 г.
3. А.В. Чижков, С.В. Сеитова. КЛАССИФИКАЦИЯ НЕЙРОСЕТЕВЫХ АРХИТЕКТУР – Известия ЮФУ. Технические науки
4. Vikas Gupta. Understanding Feedforward Neural Networks – [статья] – URL: https://learnopencv.com/understanding-feedforward-neural-networks/ – Текст: электронный.
5. Нейронная сеть Хопфилда – [статья] – URL: https://microtechnics.ru/nejronnaya-set-hopfilda-obzor-obuchenie-i-primer-raboty/ – Текст: электронный.
6. Applied Deep Learning - Part 3: Autoencoders – [статья] – URL: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798 – Текст: электронный.
7. Сверточные нейронные сети – [статья] – URL: https://neerc.ifmo.ru/wiki/index.php?title=Сверточные\_нейронные\_сети – Текст: электронный.
8. Генеративно-состязательная нейросеть (GAN). Руководство для новичков – [статья] – URL: https://neurohive.io/ru/osnovy-data-science/gan-rukovodstvo-dlja-novichkov/ – Текст: электронный.
9. Рекуррентные нейронные сети: типы, обучение, примеры и применение – [статья] – URL: https://neurohive.io/ru/osnovy-data-science/rekurrentnye-nejronnye-seti/ – Текст: электронный.
10. Welcome to Python.org – [сайт] – URL: https://www.python.org/?page=2
11. scikit-learn Машинное обучение в Python – [сайт] – URL:https://scikit-learn.ru/