**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ России**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Инженерно-технологическая Академия**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра Систем Автоматизированного Проектирования**

**Отчет по лабораторной работе № 2**

**на тему: «Проектирование и машинное обучение нейросети»**

по дисциплине «Методы машинного обучения»

Выполнили:

студенты группы КТбо4-4

Батагов С.С.\_\_\_\_\_\_\_\_\_\_\_\_\_

Москаленко М.В.\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

доцент кафедры САПР

Лебедев О.Б. \_\_\_\_\_\_\_\_\_\_\_\_\_

Таганрог 2022

**СОДЕРЖАНИЕ**

[1 ЦЕЛЬ РАБОТЫ 3](#_Toc122197008)

[2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 3](#_Toc122197009)

[2.1 Общая информация 3](#_Toc122197010)

[2.2 Проектирование нейронных сетей 4](#_Toc122197011)

[2.3 Искусственный нейрон 5](#_Toc122197012)

[3 ПРАКТИЧЕСКАЯ ЧАСТЬ 9](#_Toc122197013)

[3.1 Язык программирования Python 9](#_Toc122197014)

[3.3 Разработка нейронной сети 10](#_Toc122197015)

[4 ВЫВОД 14](#_Toc122197016)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 15](#_Toc122197017)

[ПРИЛОЖЕНИЕ А 16](#_Toc122197018)

# 1 ЦЕЛЬ РАБОТЫ

Целью работы данной лабораторной работы заключается в проектировании модели искусственной нейронной сети для решения задач распознавания образов и аппроксимации. Модель будет спроектирована с использованием высокоуровневого языка программирования Python, включая несколько математических и специальных библиотек.

Вариант задания: разработать нейронную сеть, которая будет распознавать написанные от руки цифры. То есть необходимо разработать функцию для представления графической информации в нужном формате для разрабатываемой НС, или воспользоваться сторонними источниками для взятия данных для анализа и обучения нейронной сети.

# 2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 2.1 Общая информация

Нейросеть в машинном обучении — это математическая модель, которая работает по принципу нейронной сети живого организма. В отличие от нейросети животного, которая передаёт сигнал от мозга к другим органам и полностью регулирует жизнедеятельность организма, компьютерная нейросеть учится решать только ту задачу, которую ей ставит человек. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

Архитектура нейронных сетей повторяет структуру человеческого мозга. Клетки человеческого мозга, называемые нейронами, образуют сложную сеть с высокой степенью взаимосвязи и посылают друг другу электрические сигналы, помогая людям обрабатывать информацию. Точно так же искусственная нейронная сеть состоит из искусственных нейронов, которые взаимодействуют для решения проблем. Искусственные нейроны — это программные модули, называемые узлами, а искусственные нейронные сети — это программы или алгоритмы, которые используют вычислительные системы для выполнения математических вычислений.

Нейронные сети не программируются в привычном смысле этого слова, они *обучаются*. Обучение нейронной сети — это процесс обучения нейронной сети выполнению задачи. Нейронные сети обучаются путем первичной обработки нескольких больших наборов размеченных или неразмеченных данных. На основе этих примеров сети могут более точно обрабатывать неизвестные входные данные.

## 2.2 Проектирование нейронных сетей

Этапы разработки нейросетей:

1. Постановка задачи (выполняется также, как и для традиционных экспертных систем).
2. Сбор данных из доступных источников.
3. Предварительная обработка данных, включающая фильтрацию, нормирование, масштабирование и т.п.
4. Отбор наиболее существенных переменных.
5. Разбиение данных на обучающую и тестовую выборки.
6. Разработка архитектуры нейронной сети.
7. Обучение нейросети. Создание интерфейса.
8. Оценка нейросети, отладка и тестирование.
9. Доучивание.

## 2.3 Искусственный нейрон

Далее представлена модель искусственного нейрона (Рисунок 1).

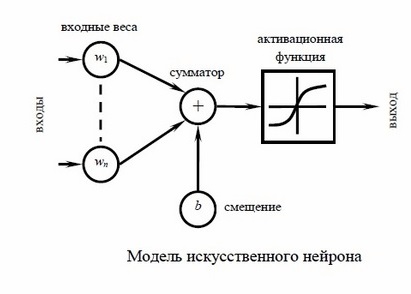


Рисунок — Модель искусственного нейрона

У искусственного нейрона присутствуют входы, на которые поступает сигнал. Принимаемые сигналы (x1) умножаются на веса (w1).

Далее все произведения передаются в сумматор, который суммирует все входные сигналы, умноженные на соответствующие веса (Рисунок 2).



Рисунок 2 — Формула сумматора для искусственного нейрона

Искусственный нейрон должен обладать активационной функцией. В задачах классификации активационная функция должна иметь характеристику "включателя". Иными словами, если вход больше, чем некоторое значение, то выход должен изменять состояние, например с 0 на 1 или -1 на 1 Это имитирует "включение" биологического нейрона. В качестве активационной функции обычно используют сигмоидальную функцию (Рисунок 3):

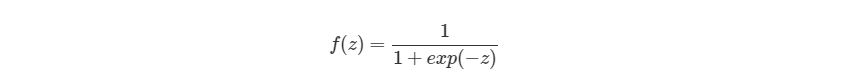


Рисунок 3 — Формула активации искусственного нейрона

Чтобы продемонстрировать, как находить выход, имея уже известный вход, в нейронных сетях, начнем с предыдущего примера с тремя слоями. Ниже такая система представлена в виде системы уравнений (Рисунок 4):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 — Формула входа выхода нейрона

где f(w) — активационная функция узла. В первой строке h1(2) — выход первого узла во втором слое, его входами соответственно являются . Эти входы было сложены, а затем переданы в активационную функцию для расчета выхода первого узла. С двумя следующими узлами аналогично.

Обучение системы происходит за счет расчета значений весов, которые соединяют слои в сети. В контролируемом обучении идея заключается в том, чтобы уменьшить погрешность между входом и нужным выходом Контролируемое обучение заключается в уменьшении погрешности между входом и нужным выходом. Погрешность Если у нас есть нейросеть с одним выходным слоем и некоторой вход xx и мы хотим, чтобы на выходе было число 2, но сеть выдает 5, то нахождение погрешности выглядит как |2–5| = 3.

Смысл контролируемого обучения в том, что предоставляется много пар вход-выход уже известных данных и нужно менять значения весов, основываясь на этих примерах, чтобы значение ошибки стало минимальным. Эти пары входа-выхода обозначаются как (x(1),y(1)),...,(x(m),y(m)), где m является количеством экземпляров для обучения. Каждое значение входа или выхода может представлять собой вектор значений, например x(1) не обязательно только одно значение, оно может содержать N-размерный набор значений. Предположим, что мы обучаем нейронную сеть выявлению спам-сообщений - в таком случае x(1) может представлять собой количество соответствующих слов.

В обучении сети, используя (x,y), целью является улучшение нахождения правильного y при известном x. Это делается через изменение значений весов, чтобы минимизировать погрешность. Для этого нам и понадобится градиентный спуск. Рассмотрим следующий график (Рисунок 5):

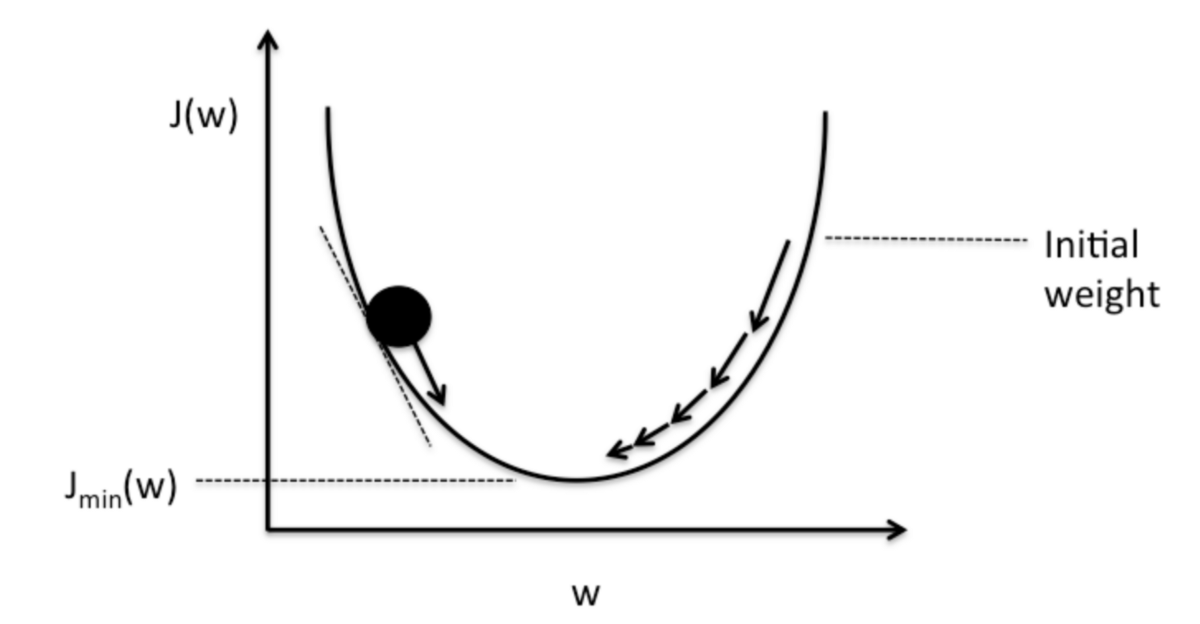


Рисунок 5 — График градиентного спуска

На данном графике изображена погрешность, зависящую от скалярного значения веса, w. Подсчет начинается со случайного значения переменной w, которая дает погрешность. Необходимо изменить w таким образом, чтобы достичь минимальной погрешности. Одним из самых распространенных способов является градиентный спуск.

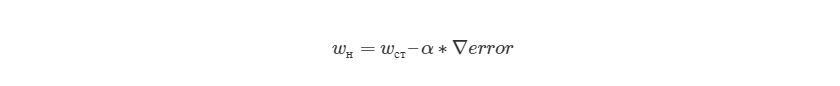
 Метод градиентного спуска использует градиент, чтобы принимать решение о следующей смены в w для того, чтобы достичь минимального значения кривой. Он итеративным методом, каждый раз обновляет значение w через (Рисунок 6):

Рисунок 6 — Формула градиентного спуска

где wн означает новое значение w, wст- текущее или "старое" значение w, ∇error является градиентом погрешности на wст и α является шагом. Шаг α также будет означать, как быстро ответ приближается к минимальной погрешности. При каждой итерации в таком алгоритме градиент должен уменьшаться. Из графика выше можно заметить, что с каждым шагом градиент "стихает". Как только ответ достигнет минимального значения, мы уходим из итеративного процесса. Выход можно реализовать способом условия "если погрешность меньше некоторого числа". Это число называют точностью.

Градиентный спуск для каждого веса w(ij)(l) и смещение bi(l) в нейронной сети выглядит следующим образом:

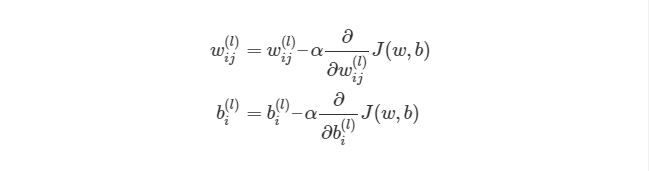


Рисунок 7 — Формула градиентного спуска в нейронных сетях

Выражение выше фактически аналогично представлению градиентного спуска: wnew=wold-α\*∇error. Нет лишь некоторых обозначений, но достаточно понимать, что слева расположены новые значения, а справа - старые. Опять же задействован итерационный метод для расчета весов на каждой итерации, но на этот раз основываясь на функции оценки J(w,b).

Значения ∂/∂wij(l)и ∂/∂bi(l) являются частными производными функции оценки, основываясь на значениях веса. Что это значит? Вспомните простой пример градиентного спуска ранее, каждый шаг зависит от наклона погрешности / оценки по отношению к весу. Производная также имеет значение наклона / градиента. Конечно, производная обозначается как d/dx. x в нашем случае является вектором, а это значит, что наша производная тоже будет вектором, который является градиент каждого измерения x.

# 3 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 3.1 Язык программирования Python

Python — это высокоуровневый язык программирования, который используется в различных сферах IT, таких как: машинное обучение, разработка приложений, web, парсинг и другие.

Помимо данных особенностей для языка программирования Python было написано и разработано большое количество библиотек, приложений и программных инструментов.

Поэтому для выполнения данной лабораторной работы воспользуемся одной из популярных библиотек для разработки нейронных сетей «scikit-learn».

Scikit-learn - один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он позволяет выполнять множество операций и предоставляет множество алгоритмов. Scikit-learn также предлагает отличную документацию о своих классах, методах и функциях, а также описание используемых алгоритмов.

При выполнении данной лабораторной работы были использованы версии:

* Python — 3.11,
* numpy — 1.23,
* scipy — 1.9.3,
* scikit-learn — 1.1.3.

Полный исходных код представлен в приложении А.

## 3.3 Разработка нейронной сети

Перед созданием нейронной сети, подключим все необходимые инструменты (Рисунок 8).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 — Подключение библиотеки scikit-learn, БД с данными и другие инструменты

Набор данных *digits* состоит из изображений цифр размером 8x8 пикселей. В *images* атрибутов из набора данных магазинов 8x8 массивы значений градаций серого для каждого изображения. Мы будем использовать эти массивы для визуализации первых 4 изображений. Target атрибут из набора данных, хранит цифру каждое изображение представляет собой, и это входит в названии 4 участков ниже (Рисунок 9).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 — Подключение графических данных

Далее представлены сгенерированные и подключенные изображения (Рисунок 10).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 — Сгенерированные изображения

Затем мы можем разделить данные на обучающие и тестовые подмножества и установить классификатор опорных векторов на обучающих выборках. Полученный классификатор может быть использован для прогнозирования значения цифры для образцов в тестовом подмножестве. (Рисунок 11).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 — Разделение данных на два подмножества

Построим матрицу путаницы истинных цифровых значений и предсказанных цифровых значений (Рисунок 12).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 — Матрица путаницы

Далее представлен результат выполнения программы (Рисунок 13-14).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 — Результат обучения нейронной сети в консоли

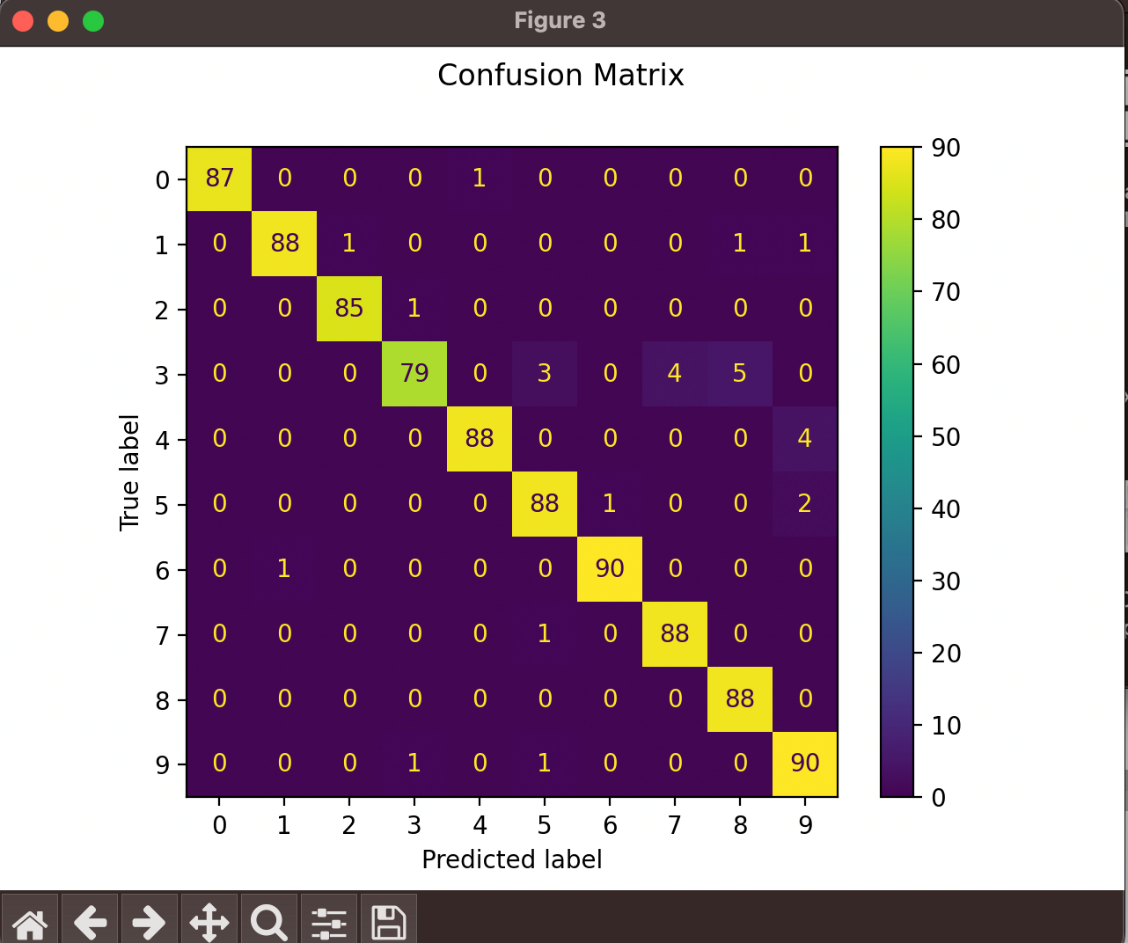


Рисунок 14 — Результат обучения НС представленная в графическом виде

# 4 ВЫВОД

Процесс создания и проверки работоспособности нейронной сети, цель которой распознавать рукописные цифры, можно считать успешным. При выполнении работы были рассмотрены несколько библиотек для разработки нейронных сетей, однако по технических причинам пришлось от них отказаться. Причиной являются устаревшие версии модулей, которые используются в данных библиотеках, или стабильная работоспособность только настроенной программной оболочки.

Данная нейронная сеть предназначена только для считывания графических данных, а именно вручную написанных цифр, для добавления большего функционала необходимо углубляться в теоретическую часть проектирования НС и их разработки, следовательно увеличивать время на изучение и более высокие аппаратные возможности для обучения нейронной сети.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что такое Python и для чего он используется // Всё о python URL: https://all-python.ru/osnovy/yazyk-programmirovaniya.html (дата обращения: 5.11.2022).
2. Пишем свою нейросеть: пошаговое руководство // proglib URL: https://proglib.io/p/neural-nets-guide (дата обращения: 5.11.2022).
3. Что такое нейронные сети и как они работают // sky.pro URL: https://sky.pro/media/neyronnye-seti (дата обращения: 5.11.2022).
4. Основы искусственного интеллекта в примерах на Python. Самоучитель. -СПб.: БХВ-Петербург, 2021. -448 с.: ил. -(Самоучитель)
5. Создаем нейронную сеть. : Пер. с англ. — СПб. : ООО “Альфа-книга”, 2017. — 272 с. : ил. — Парал. тит. англ.

# ПРИЛОЖЕНИЕ А

**Исходный код нейронной сети**

import matplotlib.pyplot as plt

from sklearn import datasets, svm, metrics

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

digits = datasets.load\_digits()

\_, axes = plt.subplots(nrows=1, ncols=7, figsize=(10, 3))

for ax, image, label in zip(axes, digits.images, digits.target):

ax.set\_axis\_off()

ax.imshow(image, cmap=plt.cm.gray\_r, interpolation='nearest')

ax.set\_title('Training: %i' % label)

n\_samples = len(digits.images)

data = digits.images.reshape((n\_samples, -1))

clf = svm.SVC(gamma=0.001)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

data, digits.target, test\_size=0.5, shuffle=False)

clf.fit(X\_train, y\_train)

predicted = clf.predict(X\_test)

\_, axes = plt.subplots(nrows=1, ncols=7, figsize=(10, 3))

for ax, image, prediction in zip(axes, X\_test, predicted):

ax.set\_axis\_off()

image = image.reshape(8, 8)

ax.imshow(image, cmap=plt.cm.gray\_r, interpolation='nearest')

ax.set\_title(f'Prediction: {prediction}')

print(f"Classification report for classifier {clf}:\n"

f"{metrics.classification\_report(y\_test, predicted)}\n")

disp = metrics.plot\_confusion\_matrix(clf, X\_test, y\_test)

disp.figure\_.suptitle("Confusion Matrix")

print(f"Confusion matrix:\n{disp.confusion\_matrix}")

plt.show()