Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

Факультет «Радиоэлектроника и лазерная техника»

Кафедра «Технологии приборостроения»

Проект на тему:

«Предиктор ВАХ слоистой квантоворазмерной структуры на основе искусственной нейронной сети»

Выполнил:

Веселенко Н. Ю.

Группа РЛ6-71Б

Проверил: доцент кафедры РЛ6, к. т. н.

Ветрова Н. А.

Москва, 2020

Оглавление

Введение

Цель работы: предсказание вольт-амперных характеристик слоистой квантов размерной структуры на основе искусственной нейронной сети.

Задачи работы:

1. Построение модели искусственной нейронной сети.
2. Обучение искусственной нейронной сети на тренировочных данных и получение ВАХ на тестовых.

Актуальность: почти во всех сферах жизни активно интегрируются нейронные сети и используется машинное обучения, наноэлектроника не исключение.

1. Искусственная нейронная сеть

1.1 Что такое нейронная сеть?

Нейронная сеть (также **искусственная нейронная сеть, ИНС**) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. [1]

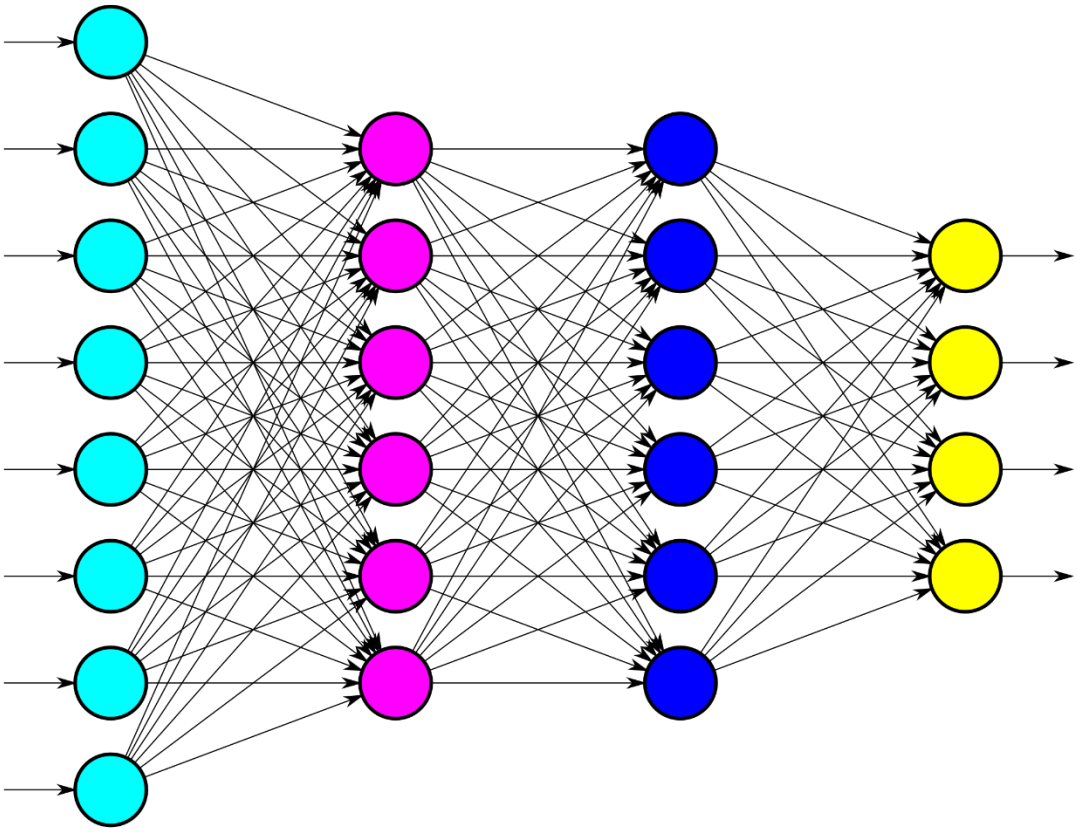

Рисунок 1 – Графическое представление нейронной сети

Так же можно сформулировать другое определение нейронной сети исходя из рисунка 1: нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Нейроны – это разноцветные окружности, синапсы – связи между нейронами.

1.2 В каких задачах используются нейронные сети?

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:

- **Классификация** — распределение данных по параметрам.

Классическим примером является задача по классификации ирисов Фишера, где основными параметрами цветков являлись: 1) длина наружной доли околоцветника; 2) ширина наружной доли околоцветника; 3) длина внутренней доли околоцветника; 4) ширина внутренней доли околоцветника.

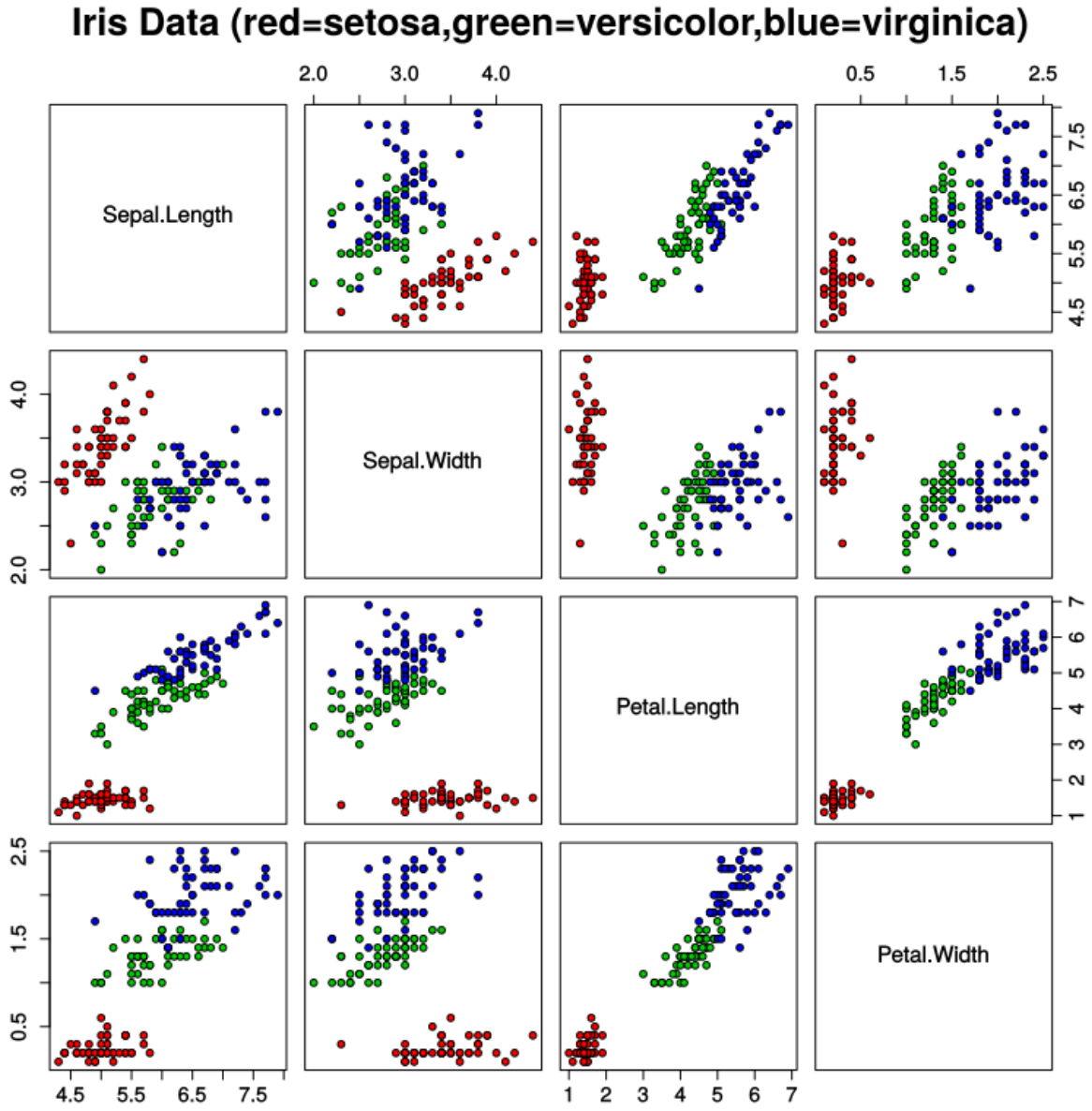

Рисунок 2 – Классификация ирисов Фишера

- **Предсказание** — возможность предсказывать следующий шаг.

Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

- **Распознавание** — в настоящее время, самое широкое применение нейронных сетей.

Классическая задача по распознавания цифр с изображения.

1.3 Структура нейронной сети

В общем виде каждая нейронная сеть представляет из себя нейрон и синапс.

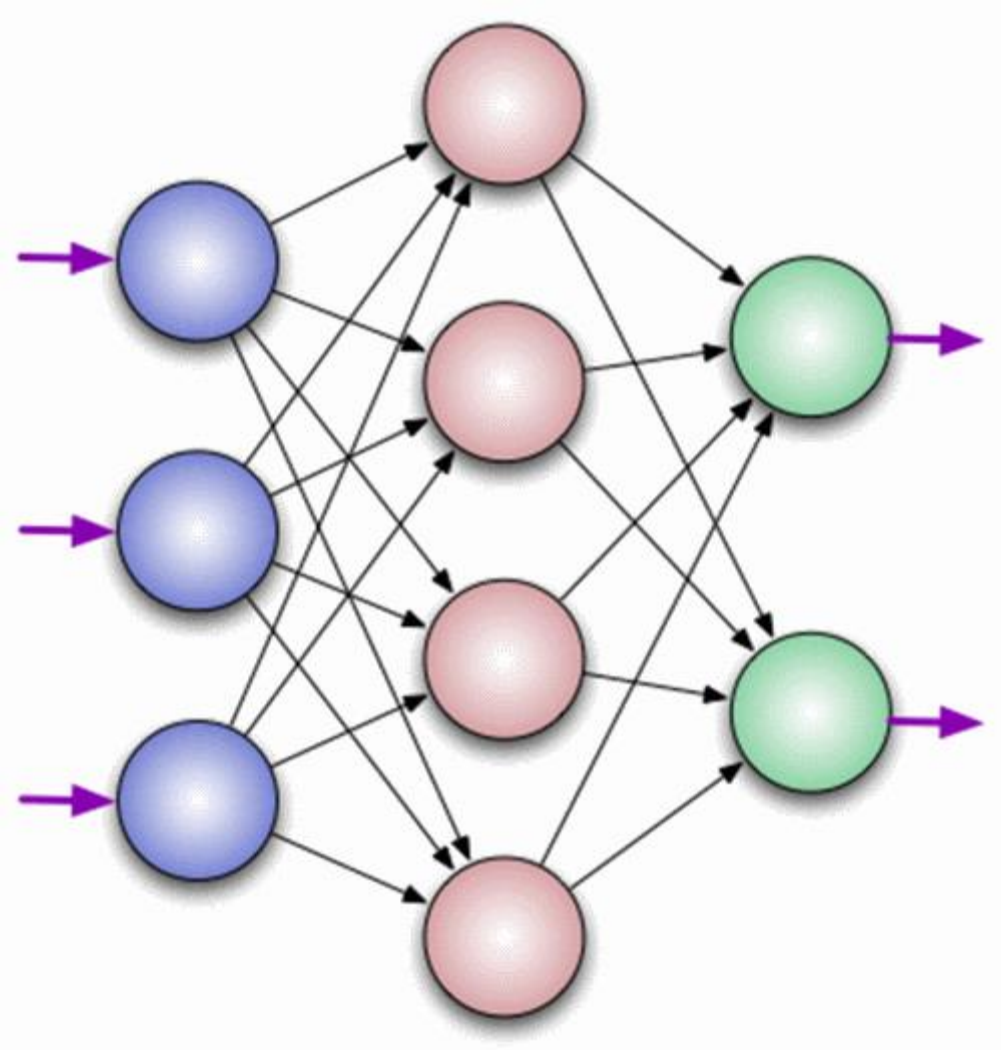

Рисунок 3 – Трехслойная нейронная сеть

- **Нейрон** - это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Все нейроны делятся на 3 типа: 1) входные (синие на рисунке 3); 2) выходные (зеленые на рисунке 3) и 3) скрытые (розовые на рисунке 3). Нейроны одного типа образующие кластер называются – слоем. Количество скрытых слоев может изменяться от 0 до 3 (чаще всего не более 3) в зависимости от задачи. Так же конфигурируется количество нейронов в каждом слое. Каждый нейрон может получить на вход данные(сигналы) и отдавать на выход. Так же нейрон несет в

себе математическую функцию активации, о которой будет рассказано позднее.

- **Синапс** - это связь между двумя нейронами. У синапсов есть 1 параметр — весовой коэффициент или вес. Благодаря ему, входная информация изменяется, когда передается от одного нейрона к другому. Значение весов варьируется от -1 до 1, но чаще от -0.99 до 0.99 и формируется на моменте создания нейронной сети случайным образом. Стоить отметить что синапсы имеются только между входным-скрытым, скрытым-скрытым, скрытым-выходным слоями. Входной слой получает данные извне, выходной отдает сигналы.

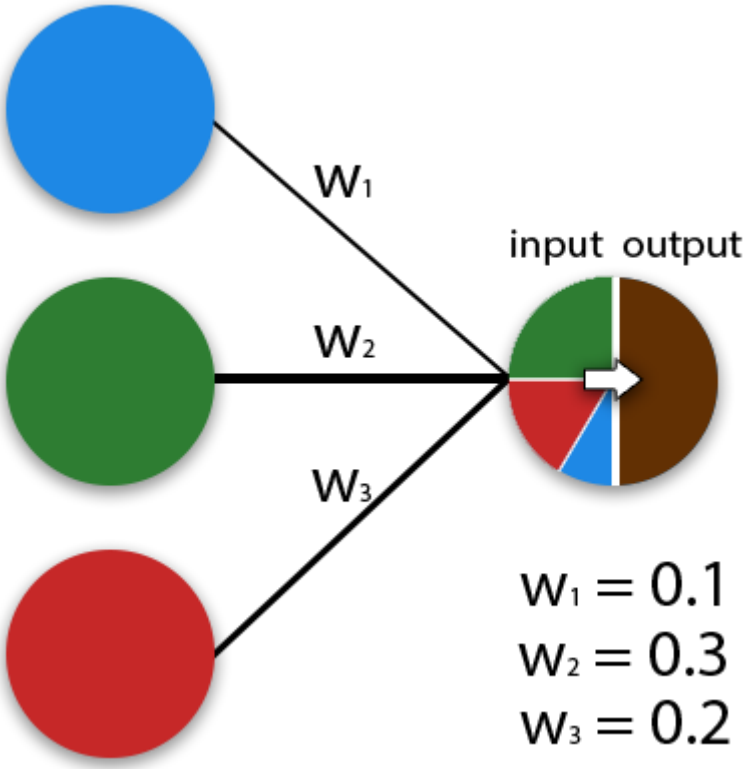

Рисунок 4 – Отображение весовых коэффициентов синапсов

1.4 Принцип работы нейронной сети

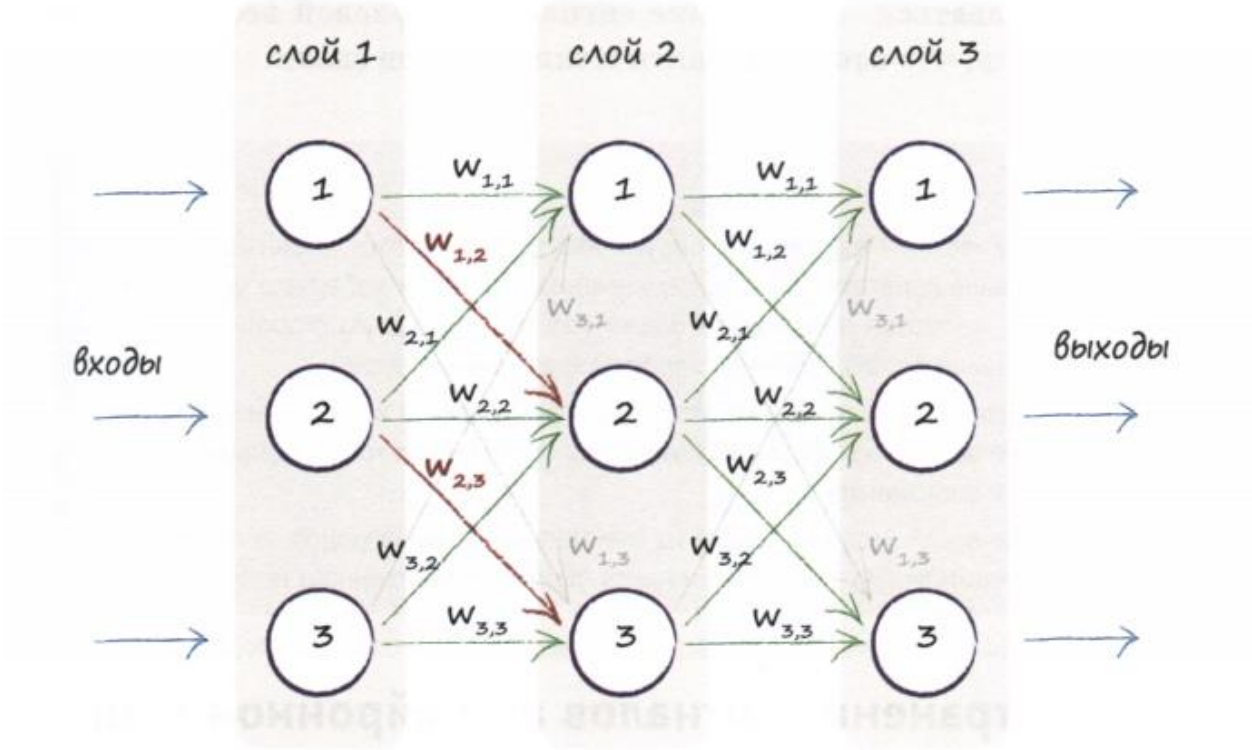

Рисунок 5 – Трехслойная нейронная сеть

В данной части будет рассказано про то, как сигнал проходит через слои нейронной сети, как формируется количество нейронов в каждом из слоев.

Перед тем, как на вход нейронной сети будут подаваться данные, формируются значения весовых коэффициентов случайным образом в диапазоне от -0.99 до 0.99. Каждый нейрон сети с рисунка 5 имеет по 3 синапса, каждый из которых соединяется с нейроном из следующего слоя. Весы для каждого слоя образуют матрицу собственную матрицу W, где w_{ij} – весовой коэффициент i-ого нейрона текущего слоя связанный с j-ым нейроном следующего слоя.

Вспомним, что каждый нейрон несет в себе функцию активации, истоки которой имеют корни биологического аналога нейрона. Биологические нейронные клетки получают сигнал и передают его дальше. Но не каждый входной сигнал возбуждает нейрон (преобразуется в выходной сигнал).

Сигнал должен иметь величину, превосходящую порог. И представить его можно в виде **ступенчатой функции.**

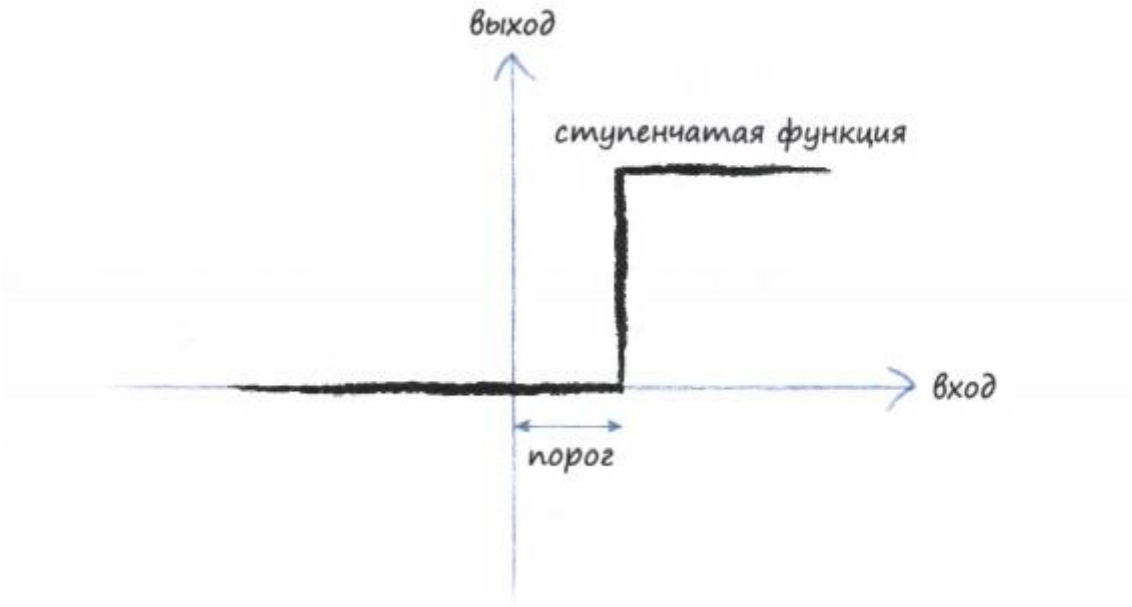

Рисунок 6 – Ступенчатая функция

Можно заметить, что слабые входные сигналы «не пройдут порог», т.е. не смогу преобразоваться в выходной сигнал. Поэтому ступенчатую функцию мы заменим на функцию **сигмоиды.**

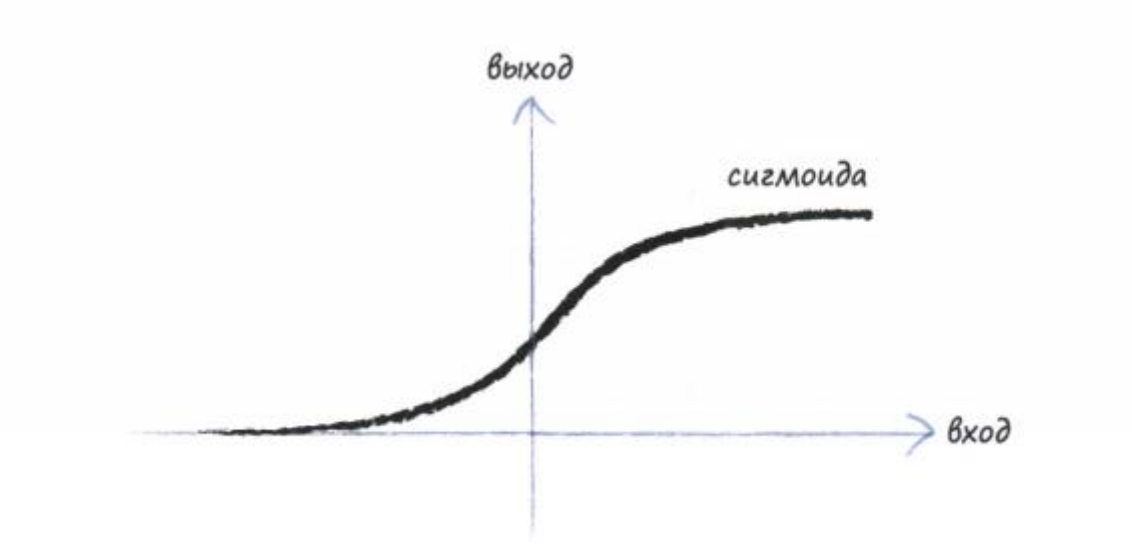

Рисунок 7 – Сигмоидная функция

Математически сигмоида представляется следующим образом:

$$y = \frac{1}{1+ e^{-x}}, \qquad (1)$$

где x-значение входного сигнала, а y-значение выходного сигнала.

Применяются и другие функции в качестве функции активации, но в своей нейронной сети я использовал только сигмоиду.

В итоге, мы имеем 2 матрицы весовых коэффициентов и знаем, что нейроны несут в себе функцию активации – сигмоиду. Стоить отметить, что нейроны входного слоя не имеют функции активации, это важно запомнить. Входной сигнал просто проходит через входные нейроны и по синапсу переходит на нейроны следующего слоя.

Теперь пошагово разберем, что же происходит при подаче входных данных (сигналов) нейронной сети.

Мы подаем на входной слой входной сигнал, который представлен в виде вектора размерностью n, где n – количество нейронов во входном слое. Важно отметить, что входные сигналы должны быть больше 0 и меньше 1. Как уже ранее говорилось, сигнал просто проходит через входной слой на следующий слой нейронов по синапсу. Каждый синапс имеет свой весовой коэффициент, значит сигнал проходя по синапсу должен быть умножен на вес синапса, по которому он проходит. Как видно на рисунке 5, во втором слое в каждый нейрон 2 слоя имеет синапс с нейроном 1 слоя, но в итоге в каждый нейрон 2 слоя должен прийти лишь один сигнал, это значит, что мы должны сложить сигналы, прошедшие по синапсу и тогда мы получим 1 сигнал.

Разберем более подробно на примере 1-го нейрона 2-го слоя. Мы имеем 3 сигнала I_1, I_2 и I_3, которые поступили на 3 нейрона входного слоя. Они проходят по синапсам на 1 нейрон 2-го слоя (т.е. должны быть умножены на вес синапса). Теперь мы имеем: $I_1 * w_{1,1}$, $I_2 * w_{2,1}$ и $I_3 * w_{3,1}$. И для получения одного сигнала мы должны сложить все 3 сигнала: $I_1 * w_{1,1} + I_2 * w_{2,1} + I_3 * w_{3,1}$. И так для всех 3 нейронов 2-го слоя:

$$I_1 * w_{1,1} + I_2 * w_{2,1} + I_3 * w_{3,1}$$
$$I_1 * w_{1,2} + I_2 * w_{2,2} + I_3 * w_{3,2}$$
$$I_1 * w_{1,3} + I_2 * w_{2,3} + I_3 * w_{3,3}$$

Если вспомнить, то похожие действия делаются при скалярном произведении матриц. А ранее я говорил, что мы имеем две матрицы W с весовыми коэффициентами и вектор сигналов. Таким образом, применяя скалярное произведение, мы получим на вход вектор новых сигналов:

$$I_2 = W_1 * I_1$$

Помним про то, что нейроны со 2 слоя и далее несут в себе функцию активации, то есть каждое значение вектора I_2 должно быть пропущено через функцию активации (сигмоиду) и имеем O_2.

Те же самые действия производятся и для 3 слоя нейронов –выходного.

$$I_3 = W_2 * O_2$$

На выходе нейронной сети мы получим O_3.

Это еще не все. В данной части был разобран лишь принцип того, как входной сигнал проходит через нейронную сеть на выход.

1.5 Обучение нейронной сети

Если нейронной сети просто подавать данные на вход, то мы ничего не получим, раз за разом мы будем получать какие-то непонятные случайные данные на выходе.

Обучение нейронной сети заключается в уравновешивании весовых коэффициентов синапсов. В проекте использовалась нейронная сеть прямого распространение, обучение происходило по принципу «обучение с учителем», когда на вход подаются данные и так же есть истинное значение, которое должно быть получено на выходе – целевое значение. Принцип обучения заключается в использовании **обратного распространения ошибки** и **градиентного спуска** для уравновешивания весов.

Разберем, как происходит процесс обратного распространения ошибки. Снова обратимся рисунок 5, для абстрактного разбора. Вспомним, что как ранее мы считали, на выходе нейронной сети мы имеем вектор выходного сигнала O_3. При обучении с учителем, мы будем иметь вектор целевых значений T, имеющий такую же размерность, что и O_3. Ошибка на выходном слое: E_3= T - O_3. В данном случае, мы получаем значение ошибки, которая была получена в результате прохождения сигнала через синапсы 2 и 3 слоев или скрытого и выходного. Чтобы получить ошибку на синапсах 1 и 2 слоев, необходимо транспонировать матрицу весов W_1 и умножить на значение ошибки, полученное ранее: $E_2= W_1^T * E_3$. Транспонирование необходимо для того, чтобы получить вклад каждого весового коэффициента, внесенного в полученную ошибку. В случае, если слоев больше 3, то мы продолжаем передавать ошибку назад по слоям по такому же принципу.

Пришло время разобрать саму сложную тему в обучении нейронных сетей – градиентный спуск. Именно благодаря ему мы можем изменять весовые коэффициенты. **Градиентный спуск** — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента. Идея заключается в том, что мы должны минимизировать величину

ошибки посредством градиентного спуска, т.е. изменением весовых коэффициентов.

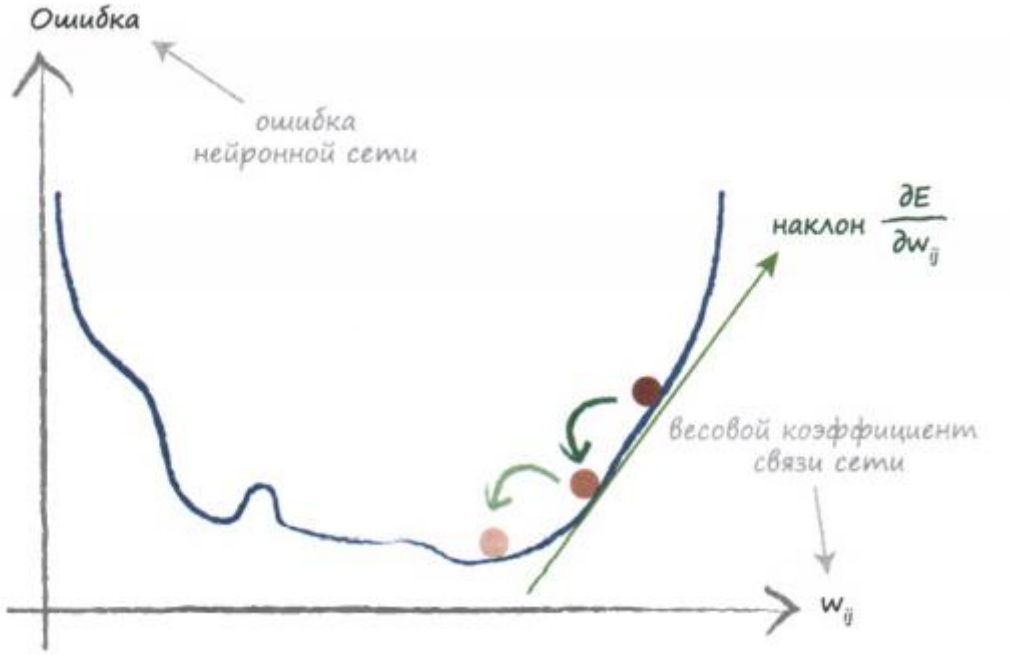

Рисунок 8 – Графическое отображение градиентного спуска

Перед тем как мы разберем то, как вычисляется $\frac{\partial E}{\partial w_{ij}}$, определим то, что функция ошибки имеет вид: (целевое значение – фактическое)^2.

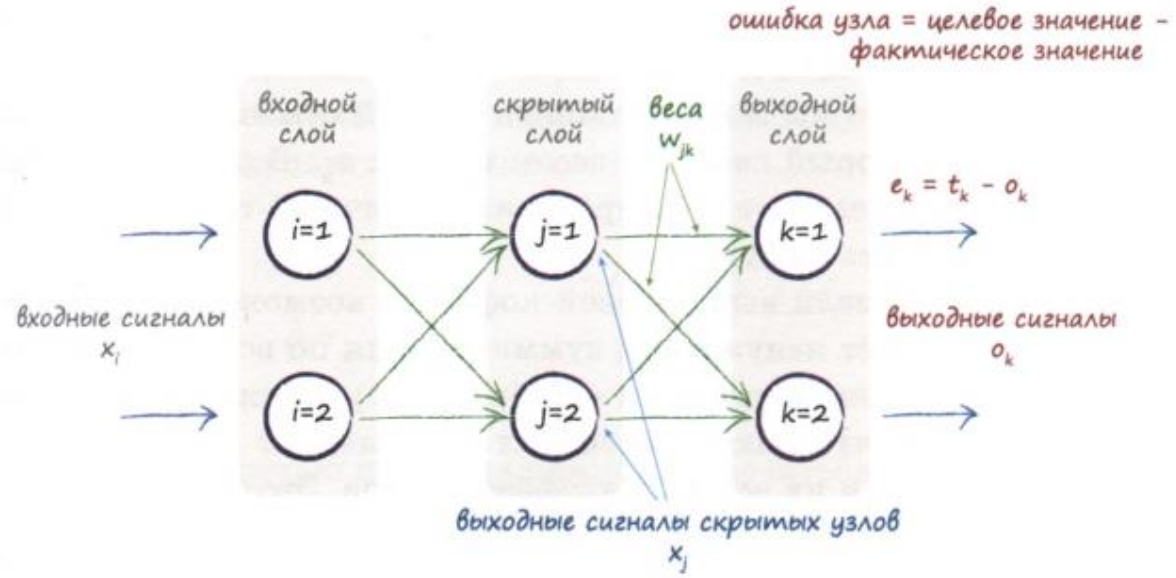

Рисунок 9 – Трехслойная нейронная сеть по 2 нейрона в каждом слое

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \sum_{n} (t_n - o_n)^2$$

Суммирование происходит по всем n выходным узлам (нейронам). Сразу же мы можем упростить, т.к. видно, что выходной сигнал o_n зависит лишь от связей, которые с ним соединены.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} (t_k - o_k)^2$$

Умножим на $\frac{\partial o_k}{\partial o_k}$ и получим:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} * \frac{\partial o_k}{\partial w_{jk}}$$

После чего возьмем производную первого множителя (дифференцирование сложных функций) (t_k −константа):

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) * \frac{\partial o_k}{\partial w_{jk}}$$

o_k- это выходной сигнал на выходном слое, для каждого k-го нейрона (рисунок 9). Как мы помним, он у нас получается прохождением через функцию сигмоиды, в то время как перед попаданием в нейрон, у нас сигнал приходит со скрытого слоя, проходя по синапсам. $o_k = \text{сигмоида}(\sum_j w_{jk} * o_j)$, где o_j- выходной сигнал скрытого слоя. Теперь мы имеем:

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) * \frac{\partial}{\partial w_{jk}} \text{сигмоида}(\sum_j w_{jk} * o_j)$$

Нам необходимо продифференцировать функцию сигмоиды:

$$\frac{\partial}{\partial x} \text{сигмоида}(x) = \text{сигмоида}(x) * (1 - \text{сигмоида}(x))$$

Производная сигмоиды давно известна, и нет смысла разбирать то, как она получается.

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) * \text{сигмоида}\left(\sum_j w_{jk} * o_j\right) * \left(1 - \text{сигмоида}\left(\sum_j w_{jk} * o_j\right)\right) * \frac{\partial}{\partial w_{jk}}\left(\sum_j w_{jk} * o_j\right)$$

Не забываем про то, что нам надо продифференцировать функцию внутри сигмоиды:

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) * \text{сигмоида}\left(\sum_j w_{jk} * o_j\right) * \left(1 - \text{сигмоида}\left(\sum_j w_{jk} * o_j\right)\right) * o_j$$

Мы вправе избавить от множителя 2, т.к. нас интересует лишь направление функции градиента ошибки. И в итоге мы получаем функцию градиента ошибки выходного слоя:

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) * \text{сигмоида}\left(\sum_j w_{jk} * o_j\right) * \left(1 - \text{сигмоида}\left(\sum_j w_{jk} * o_j\right)\right) * o_j$$

Запишем выражение для вычисления градиента ошибки скрытого слоя:

$$\frac{\partial E}{\partial w_{ij}} = -e_j * \text{сигмоида}\left(\sum_i w_{ij} * o_i\right) * \left(1 - \text{сигмоида}\left(\sum_i w_{ij} * o_i\right)\right) * o_i$$

В мы заменили ошибку выходного слоя на ошибку скрытого слоя, которая рассчитывается посредством обратного распространения ошибки.

И так, мы получили две функции градиента ошибки, но стоит вспомнить, что изменение весовых коэффициентов обратно пропорционально

направлению градиента, т.е. мы избавимся от знака минус. И добавим коэффициент обучения α, который определяет шаг градиентного спуска.

Запишем теперь все в матричном виде:

$$\Delta w_{ij} = \alpha * \mathrm{E}_j * O_j * (1 - O_j) * O_i^T,$$

где Δw_{ij} −величина изменения весовых коэффициентов синпса, α −коэффициент обучения, E_j- вектор величины ошибки текущего слоя, O_j − вектор выходного сигнала текущего слоя, O_i − векторы выходного сигнала предыдущего слоя.

На каждой итерации обучения нейронной сети мы будем приближаться к наилучшему значению весовых коэффициентов.

Важно понимать, что для качественного обучения необходимо иметь сформированный дата сет большого объема.

1.6 Программная реализация нейронной сети на языке программирования Python

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой набор полезных функций.

Для реализации нейронной сети мы воспользуемся принципом «объектно-ориентированного программирования» (ООП) Питона. Он заключается в том, что мы можем создать класс, который имеет поля (характеристики класса) и методы (функции, которые может выполнять класс).

Класс нейронной сети лежит в файле *class_neural.py*

Листинг 1 – Инициализация класса нейронной сети

```
# определение класса нейронной сети
class neuralNetwork:
    # инициализирование нейронной сети
    def __init__(self, input_nodes: int, hidden_nodes: List[int],
output_nodes: int, learning_rate: float) -> NoReturn:
        # задаем количество узлов во входном, скрытом и выходном слоях
        self.inodes = input_nodes
        self.hnodes = hidden_nodes
        self.onodes = output_nodes
        # коэффициент обучения
        self.lr = learning_rate
        # список матриц весовых коэфициентов
        self.w = self.create_w()
        # использование сигмоиды в качестве функции активации
        self.activation_function = lambda x: sc.expit(x)

# создание списка матриц весовых коэфициентов
def create_w(self) -> List[Union[np.ndarray, float]]:
    list_w = []
    # если длина скрытых слов 0, то мы создаем одну матрицу весов с размерам
колличество выходных нейронов и входных
    if len(self.hnodes) == 0:
        # записываем в список весов сгенерированные случайным образом весовые
коэффициенты с применением
        # нормального распределения и сдвигом -0.5
        list_w.append(np.random.normal(0.0, pow(self.onodes, -0.5),
(self.onodes, self.inodes)))
    else:
        # иначе проходим по списку колличества нейронов скрытом слое и
генерируем весовые коэффициенты
        for i in range(len(self.hnodes)):
            if i == 0:
                list_w.append(np.random.normal(0.0, pow(self.hnodes[i], -
```

```
0.5), (self.hnodes[i], self.inodes)))
                if len(self.hnodes) > 1:
                    list_w.append(np.random.normal(0.0, pow(self.hnodes[i +
1], -0.5), (self.hnodes[i + 1],

self.hnodes[i])))
                else:
                    list_w.append(np.random.normal(0.0, pow(self.onodes, -
0.5), (self.onodes, self.hnodes[i])))
            elif i == len(self.hnodes) - 1:
                list_w.append(np.random.normal(0.0, pow(self.onodes, -0.5),
(self.onodes, self.hnodes[i])))
            else:
                list_w.append(np.random.normal(0.0, pow(self.hnodes[i + 1], -
0.5), (self.hnodes[i + 1],

self.hnodes[i])))
    return list_w
```

Листинг 2 – Метод опроса нейронной сети для пропускания входных данных через нейронную сеть и получения выходных

```
# опрос нейронной сети
def query(self, inputs_list: List[float]) -> List[float]:
    # преобразовать список входных значений в двухмерный массив
    inputs = np.array(inputs_list, ndmin=2).T
    # получение выходного сигнала
    for hn in self.w:
        inputs = self.activation_function(np.dot(hn, inputs))
    return inputs
```

Листинг 3 – Метод тренировки нейронной сети

```
# метод для тренировки нейронной сети
def train(self, inputs_list: List[float], targets_list: List[float]) ->
NoReturn:
    # преобразовать список входных значений в двухмерный массив
    inputs = np.array(inputs_list, ndmin=2).T
    targets = np.array(targets_list, ndmin=2).T
    outputs_list = []
    outputs = 0
    i = 0
    # получение выходных сигналов на каждом слое
    for hn in self.w:
        if i == 0:
            outputs = self.activation_function(np.dot(hn, inputs))
        else:
            outputs = self.activation_function(np.dot(hn, outputs))
        outputs_list.append(outputs)
        i += 1
    # получение ошибки в зависимости от целевого значения на выходе
    output_errors = targets - outputs
    # процесс градиентного спуска в зависимости от колличества скртых слоев
    if len(self.hnodes) == 0:
        self.w[0] += self.lr * np.dot((output_errors * outputs_list[0] * (1.0
- outputs_list[0])),
                                      np.transpose(inputs))
    else:
        i = 0
        outputs_list = outputs_list[::-1]
        for out in outputs_list:
            if i == 0:
                self.w[-(i + 1)] += self.lr * np.dot((output_errors * out *
```

```
(1.0 - out)),
                                                np.transpose(outputs_list[i +
1]))
            elif i == len(outputs_list) - 1:
                if len(self.w) > 2:
                    errors = np.dot(self.w[1].T, errors)
                    self.w[0] += self.lr * np.dot((errors * out * (1.0 -
out)), np.transpose(inputs))
                elif len(self.w) == 2:
                    errors = np.dot(self.w[-1].T, output_errors)
                    self.w[-(i + 1)] += self.lr * np.dot((errors * out * (1.0
- out)), np.transpose(inputs))
            elif i == 1:
                errors = np.dot(self.w[-1].T, output_errors)
                self.w[-(i + 1)] += self.lr * np.dot((errors * out * (1.0 -
out)),

np.transpose(outputs_list[i + 1]))
            else:
                errors = np.dot(self.w[-i].T, errors)
                self.w[-(i + 1)] += self.lr * np.dot((errors * out * (1.0 -
out)),

np.transpose(outputs_list[i + 1]))
            i += 1
```

2. Квантоворазмерная структура

Квантоворазмерные структуры – это структуры, в которых свободные носители заряда локализованы в одном, двух или трех координатных направлениях в области с размерами порядка дебройлевской длины волны электрона.

Резонансно туннельная структура — это двойной потенциальный барьер с квантовой ямой. Толщины потенциальных барьеров и квантовой ямы таковы, что возможно эффективное туннелирование через каждый из барьеров, и движение электрона поперек ямы квантуется, чему соответствуют дискретные уровни энергии в яме. Эти явления наблюдаются в полупроводниках при толщинах барьеров и ямы порядка десятков и сотен ангстрем, т. е. сравнимых с длиной волны де Бройля.

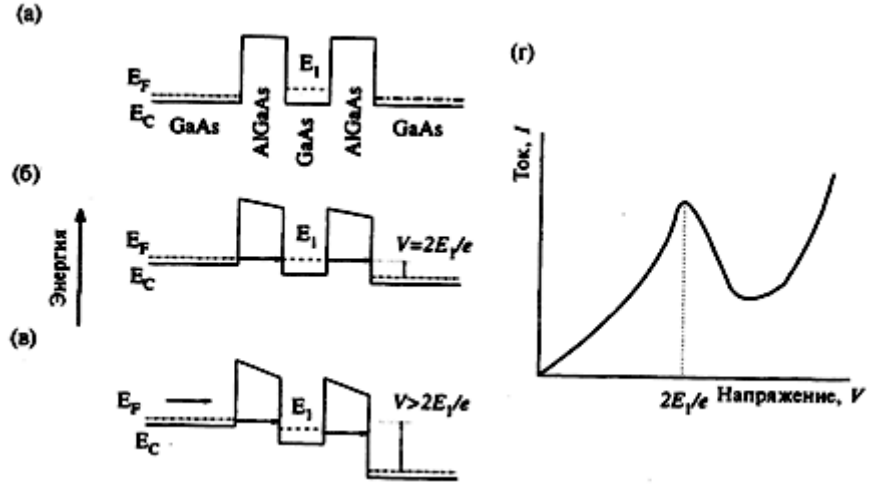

Рисунок 10 – Общий вид резонансно туннельной структуры и ВАХ

За протекание тока через резонансно-туннельную структуру отвечает последовательное туннелирование: при приложении потенциала к структуре электроны, энергия которых совпадает с разрешенным уровнем в квантовой яме, туннелируют на этот уровень сквозь левый барьер, а затем происходит еще одно туннелирование через правый барьер

3. Получение ВАХ резонансно туннельной структуры при помощи искусственной нейронной сети

Как уже говорилось ранее необходимо создать дата сет для тренировки нейронной сети. Дата сет создавался на основе готовой программы из книги «Quantum transport: Atom to Transistor».

Аналитическим методом были вычислены граничные значения изменяющихся параметров: a (ширина потенциальной ямы) = [2.60e-10… 3.50e-10] с шагом 0.01e-10, m (эффективная масса) = [0.15*9.1e-31…0.34*9.1e-31] с шагом 0.01*9.1e-31 и Ef (энергия Ферми) = [0.01…0.17] с шагом 0.01.

В итоге был создан дата сет входных данных состоящий из 3 изменяющихся параметров: ширина потенциально ямы, энергия Ферми и эффективная масса. И был создан дата сет выходных данных значений тока в зависимости от входных данных. Общий объем дата сета примерно 30000 данных.

Разберем конфигурации нейронной сети для данной задачи. Мы имеем 3 параметра структуры, которые являются входными данными, следовательно, во входном слое мы имеем 3 нейрона. Мы будем находить 30 значений тока для напряжения от 0 до 0.5 В. Но т.к. нейронная сеть не выдает значение 0, то мы исключим его, следовательно, в выходном слое мы будем иметь 29 нейронов. Коэффициент обучения мы возьмем равный 0.1, для наименьшего шага градиентного спуска. Количество скрытых слоев 1, и в этом скрытом слое 10 нейронов, подобранные аналитическим путем.

В качестве тестовых данных, для проверки возьмем следующие данные: a = 3e-10, m = 0.25*9.1e-31 и Ef = 0.1, выходными данными будут значения тока.

В итоге мы получаем следующий график, получаемый в программе *present.py*:

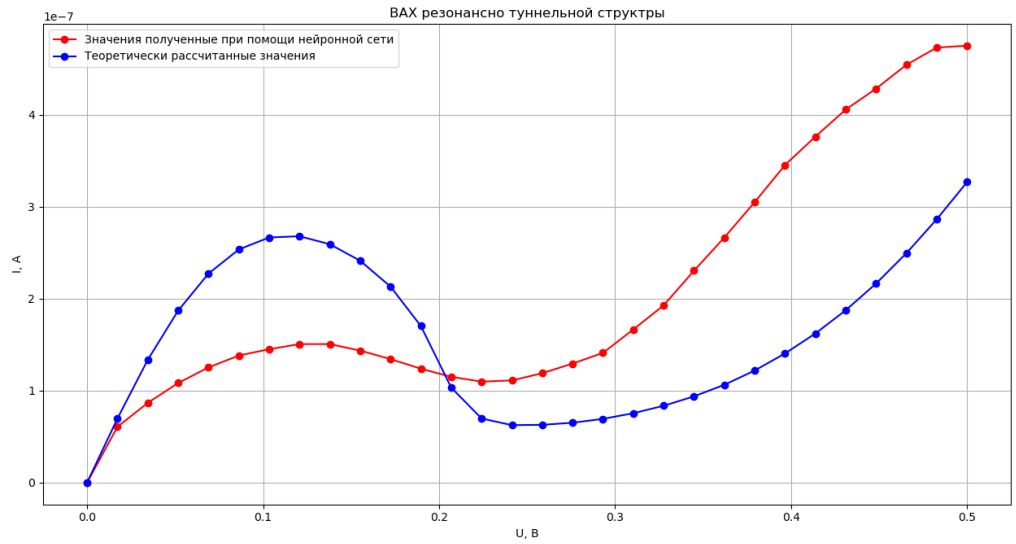

Рисунок 11 – ВАХ полученная теоретически и при помощи искусственной нейронной сети

Как видим на графике, значения тока, полученные при помощи нейронной сети далеки от идеала. Но в общем случае, по данному графику можно сказать, что это ВАХ резонансной туннельной структуры.

Литература

- Тарик Рашид – Создаем Нейронную сеть: Пер. с англ. – СПб.: ООО «Альфа-книга», 2017г.
- Supriyo Datta – Quantum Transport: Atom to Transistor – 2005г.

Приложение А

Листниг 4 – Реализация обучения нейронной сети на дата сете

```
from class_neural import neuralNetwork  # импортирование класса нейронной
сети
import numpy as np  # импортирование библиотеки для работы с матрицами,
векторами
import matplotlib.pyplot as plt  # импортирование библиотеки для построения
графиков

# задаем конфигурации нейронной сети
input_nodes = 3  # колличество нейронов во входном слое
hidden_node = [10]   # колличество нейронов и скрытые слоя
output_nodes = 29  # колличество нейронов во выходном слое
learning_rate = 0.1  # коэффициент обучения
# инициализируем нейронную сеть (создаем объект нейронной сети)
n = neuralNetwork(input_nodes, hidden_node, output_nodes, learning_rate)
input_file = 'input_1.csv'
output_file = 'output_1.csv'
# считываем входные данные из файла
with open(input_file, 'r') as f_o:
    input_data = f_o.readlines()
# считываем входные данные из файла
with open(output_file, 'r') as f_o:
    output_data = f_o.readlines()
# создаем массив значений напряжений
V = np.linspace(0, 0.5, 30)
# записываем истинные значения тока
II = '0 6.96890221646430e-08 1.33585241650928e-07 1.87045491236855e-07
2.27307627131308e-07 2.53586664609999e-07 ' \
     '2.66622768323193e-07 2.67934110493272e-07 2.59238309802126e-07
2.41249942826908e-07 2.13100010815177e-07 ' \
     '1.70220091358203e-07 1.03220030322234e-07 6.96591649626146e-08
6.24181217952826e-08 6.27779536483396e-08 ' \
     '6.50440413691550e-08 6.92566091720526e-08 7.53962762161739e-08
8.35041023302298e-08 9.37313492868583e-08 ' \
     '1.06308849123870e-07 1.21849171025836e-07 1.40420564318235e-07
1.62138307554969e-07 1.87412776774711e-07 ' \
     '2.16502158326872e-07 2.49530239138902e-07 2.86437326558888e-07
3.26923320783867e-07'
# превращаем строку в массив данных
II = II.split(' ')
II = np.asfarray(II)
# цикл треннировки нейронной сети
for i in range(len(input_data)):
    input_l = np.asfarray(input_data[i].split(';'))  # превращаем строку в
массив данных
    output_l = np.asfarray(output_data[i].split(';'))  # превращаем строку в
массив данных
    input_l = input_l[:3]  # обрезаем массив
    output_l = output_l[1:]  # обрезаем массив
    n.train(input_l, output_l)  # вызываем метод треннировки нейронной сети
    """for j, v in enumerate(V):
        if j == 0:
            input_l = np.append(input_l, v)
        elif j != 0:
            input_l[-1] = v
            n.train(input_l, output_l[j])"""

#final_input = [0.2275, 0.1,   0.3, 0.15, 0.16, 0.15, 0.46]
# задаем входные данные для теста
final_input = [0.2275, 0.1,    0.3]
```

```
I = np.array([0])  # создаем массив со значением тока
"""for j, v in enumerate(V):
    if j != 0:
        final_input[-1] = v
        I.append(n.query(final_input)[-1] * 1e-5)"""
I = np.append(I, n.query(final_input) * 1e-5)  # получаем данные при подаче
тестовых данных в нейронную сеть
print(I)
plt.title("ВАХ резонансно туннельной структры")
# построение графика ВАХ с полученными значениями Тока из нейронной сети
plt.plot(V, I, 'r-o', color='r', label='Значения полученные при помощи
нейронной сети')
plt.plot(V, II, 'r-o', color='b', label='Теоретически рассчитанные значения')
# построение графика реальных значений ВАХ
# ось x
plt.xlabel('U, В')
# ось y
plt.ylabel('I, А')
# легенда
plt.legend()
plt.grid()  # включаем сетку
plt.show()  # вывод графика
```