**Министерство образования и науки 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 «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

**ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Исследование однослойных нейронных сетей на примере моделирования булевых выражений**

**Вариант № 7**

**Преподаватель:** Коннова Н.С.

**Студент:** Кустов И. А.

**Группа:** ИУ8-61

Оглавление

[Цель работы 3](#_Toc5313934)

[Постановка задачи 3](#_Toc5313935)

[Практическая часть 4](#_Toc5313936)

[Вывод 10](#_Toc5313937)

[Приложения 10](#_Toc5313938)

# **Цель работы** - исследовать функционирование простейшей нейронной сети (НС) на базе нейрона с нелинейной функцией активации и обучить ее по правилу Видроу – Хоффа

**Постановка задачи** - получить модель булевой функции (БФ) на основе однослойной НС (единичный нейрон) с двоичными входами *x1, x2, x3, x4 ∈ {0, 1}*, единичным входом смещения *x0 =1*, синаптическими весами *w0, w1, w2, w3, w4*двоичным выходом *ye ∈ {0,1}* и заданной нелинейной функцией активации (ФА)

*f : R → (0,1)*

Для заданной БФ реализовать обучение НС с использованием:

1) всех комбинаций переменных *x1, x2, x3, x4*;

2) части возможных комбинаций переменных *x1, x2, x3, x4*;

Остальные комбинации являются тестовыми.

# **Практическая часть**

Получим нейронную модель БФ(таблица 1.1):

F(*x1, x2, x3, x4) = +*

Таблица 1.1

**Таблица истинности БФ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | F |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Обучение будет проведено с нормой обучения 0.3 и нулевыми начальными весами.

1. Используем пороговую ФА:

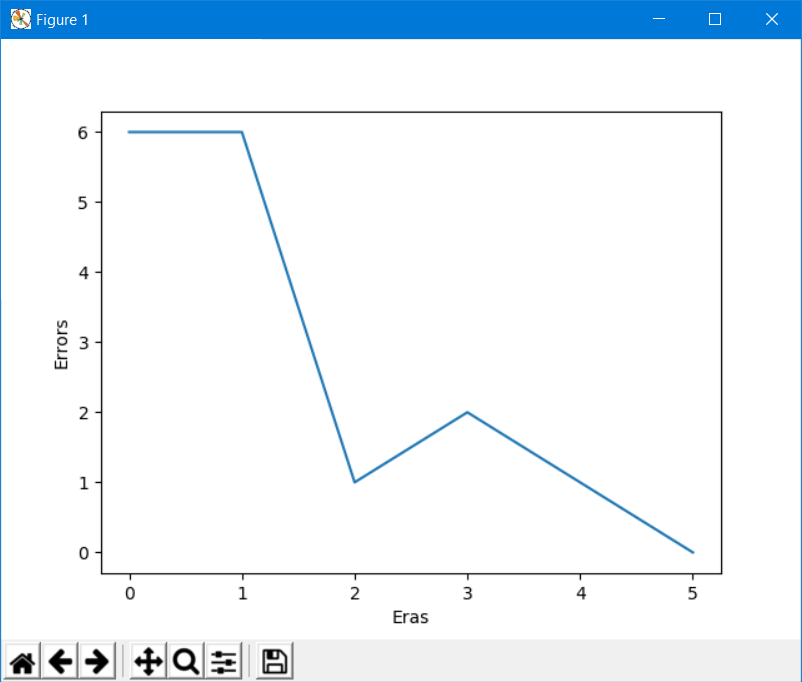
*f* (*net*) =

На таблице 1.2 представлена динамика НС. График суммарной ошибки приведен на рисунке 1.1

Таблица 1.2

**Параметры НС на последовательных эпохах (пороговая ФА)**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Вектор весов w** | **Выходной вектор y** | **Суммарная ошибка E** |
| 0 | [0, 0, 0, 0, 0] | [1, 1 ,1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1] | 6 |
| 1 | [0, 0, 0, 0, 0.9] | [1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1] | 6 |
| 2 | [0, 0, 0, 0.9, 0.9] | [1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 1 |
| 3 | [-0.3, 0, -0.3, 0.9, 0.9] | [0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1] | 2 |
| 4 | [-0.3, -0.3, -0.3, 0.9, 0.9] | [0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 1 |
| 5 | [ 0, -0.3, -0.3, 0.9, 0.9] | [1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 0 |



**Рисунок 1.1.** График суммарной ошибки НС

по эпохам обучения (пороговая ФА)

1. Используем логистическую ФА:

*,*

производная которой равна:

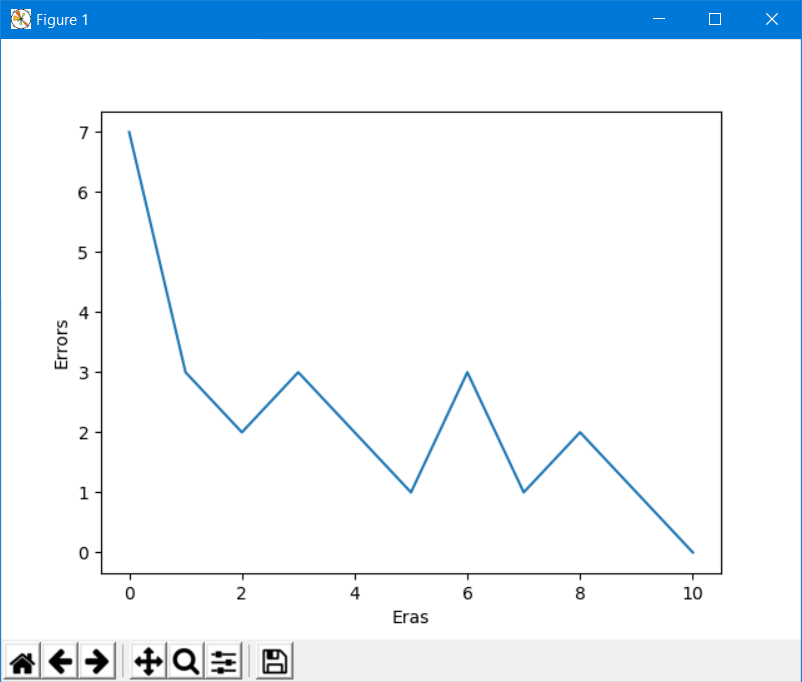
На обучение НС потребовалось 10 эпох. На таблице 1.3 представлена динамика НС. График ошибки приведен на рисунке 1.2

Таблица 1.3

**Параметры НС на последовательных эпохах**

**(логистическая ФА)**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Вектор весов w** | **Выходной вектор y** | **Суммарная ошибка E** |
| 0 | [0, 0, 0, 0, 0] | [1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1] | 7 |
| 1 | [ 0.043, -0.015, 0.057, 0.119, 0.335] | [1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1] | 3 |
| 2 | [-0.077, -0.152, 0.075, 0.261, 0.335] | [0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 2 |
| 3 | [-0.065, -0.152, -0.043, 0.261, 0.335] | [0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 3 |
| … | … | … | … |
| 10 | [0.026 -0.155 -0.184 0.555 0.469] | [1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 0 |



**Рисунок 1.2.** График суммарной ошибки НС

по эпохам обучения (логистическая ФА)

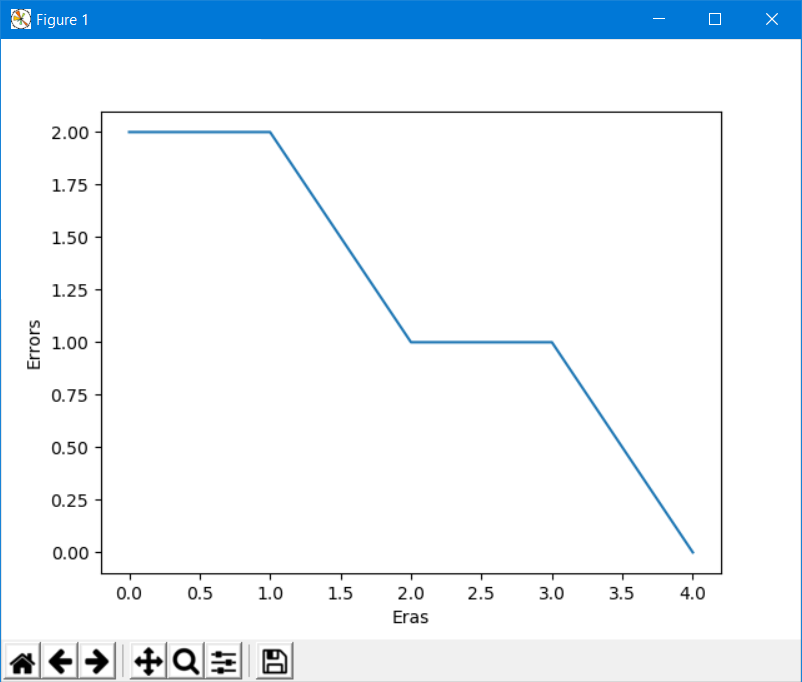
**3.**Обучение НС с использованием минимальной обучающей выборки c пороговой ФА. Была использована выборка [0, 0, 0, 0], [1, 1, 0, 0], [1, 1, 1, 1]. На обучение НС потребовалось 4 эпохи. На таблице 1.4 представлена динамика НС. График ошибки приведен на рисунке 1.3

Таблица 1.3

**Параметры НС на последовательных эпохах (пороговая ФА)**

**при наборе из 4 векторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Вектор весов w** | **Выходной вектор y** | **Суммарная ошибка E** |
| 0 | [0, 0, 0, 0, 0] | [1, 1, 0] | 2 |
| 1 | [0, 0, 0, 0.3, 0.3] | [1, 1, 0] | 2 |
| 2 | [0, 0, 0, 0.6, 0.6] | [1, 1, 1] | 1 |
| 3 | [-0.3, -0.3, -0.3, 0.6, 0.6] | [0, 0, 1] | 1 |
| 4 | [0, -0.3, -0.3, 0.6, 0.6] | [1, 0, 1] | 0 |



**Рисунок 1.3.** График суммарной ошибки НС по эпохам обучения с минимальным количеством наборов (пороговая ФА)

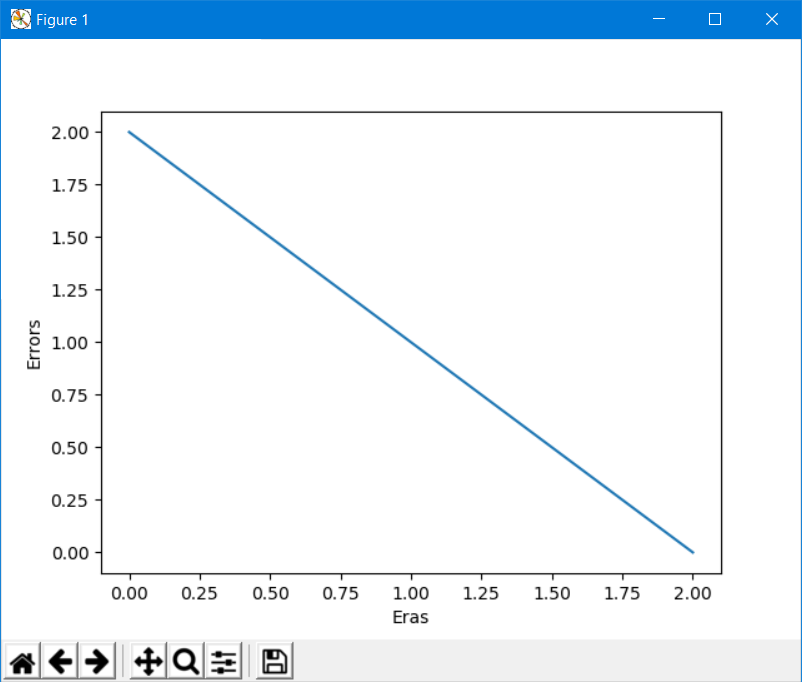
**4.** Обучение НС с использованием минимальной обучающей выборки c логистической ФА. Была использована выборка [0, 0, 1, 1], [1, 1, 0, 0], [1, 1, 0, 1]. На обучение НС потребовалось 2 эпохи. На таблице 1.5 представлена динамика НС. График ошибки приведен на рисунке 1.4

Таблица 1.5

**Параметры НС на последовательных эпохах (логистическая ФА)**

**при наборе из 4 векторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Вектор весов w** | **Выходной вектор y** | **Суммарная ошибка E** |
| 0 | [0, 0, 0, 0, 0] | [1, 1, 0] | 2 |
| 1 | [-0.079, -0.079, -0.079, 0.0, 0.071] | [0, 0, 1] | 1 |
| 2 | [0.069, -0.079, -0.079, 0.148, 0.219] | [1, 0, 1] | 0 |



**Рисунок 1.4.** График суммарной ошибки НС по эпохам обучения с минимальным количеством наборов (логистическая ФА)

# **Вывод**

В процессе работы было проведено 4 обучения НС по правилу Видроу-Хоффа с разными ФА (пороговая и логистическая) и выборками обучения (полный вектор и минимальная выборка). На пороговой ФА выборка обучается быстрее, чем на логистической. При минимальной выборке при логистической ФА обучение прошло быстрее, чем при пороговой. При минимальной выборке при обеих ФА обучение прошло быстрее, чем на полных наборах.

# **Приложения**

Ссылка на гитхаб с кодом:

<https://github.com/Kustov-Ilya/lab_work_II/blob/master/lab1.py>

Листинг №1

import numpy as np

import itertools

import matplotlib.pyplot as plt

import math

#подсчет колва шагов при поиске комбинации

count = 0

#Обучение НС

def Train(logistic, select):

global count

count += 1

nu = 0.3

W = np.zeros(5, dtype = float)

OutText = ''

list\_errors = []

era = 0

while not list\_errors or list\_errors[era - 1] > 0: #Начало эпохи

if era == 50:

break

OutText += f'Era: {era}\nw: {", ".join([str(w) for w in np.around(W, 3)])}\n'

era += 1

errors = 0

vector\_y = []

for i in select:

net = np.dot(W, i)

t = Function(i)

y = Activate\_func(logistic, net)

vector\_y.append(str(y))

sigma = t - y

if sigma != 0:

errors += 1

#корректировка весов

W += [nu \* sigma \* Derivative\_act\_func(logistic, net) \* xi for xi in i]

OutText += f"Errors: {errors}\nVector: {', '.join(vector\_y)}\n{'-' \* 70}\n"

list\_errors.append(errors)

#проверка обучения

if Real\_vector() == Neuron\_vector(W, logistic):

OutText += f"Test accessed\n{'\*' \* 70}\n"

print(f'Step {str(count)}\nSelection:\n{[x[1:] for x in select]}\n{OutText}')

count = 0

Drow\_plot(era, list\_errors)

return era

else:

print(f"{count}\nTest don't accessed\nNeuron can't learn\n{'\*' \* 70}")

return None

#Вывод графика

def Drow\_plot(era, list\_errors):

X = [i for i in range(era)]

Y = [list\_errors[i] for i in X]

plt.plot(X, Y)

plt.xlabel('Eras')

plt.ylabel('Errors')

plt.show()

#Выбор режима работы программы. Пороговая или логистическая.

#Со всеми наборами или с минимальной выборкой

def Selection(isselect, islogistic):

Set = [[1] + list(i) for i in itertools.product([0, 1], repeat = 4)]

if not isselect:

Train(islogistic, Set)

else:

for i in range(1, 16): #поиск минимальной обучающей выборки

for rezult\_selection in itertools.permutations(Set, i):

if Train(islogistic, list(rezult\_selection)) != None:

return None

#Булевый вектор, который высчитала НС

def Neuron\_vector(W, logistic):

out = []

for i in [[1] + list(i) for i in itertools.product([0, 1], repeat = 4)]:

net = np.dot(W, i)

y = Activate\_func(logistic, net)

out.append(str(y))

return ''.join(out)

#Функция активации

def Activate\_func(logistic, net):

if logistic:

return 1 if 0.5 \* (net / (1 + abs(net)) + 1) >= 0.5 else 0

else:

return 1 if net >= 0 else 0 #для пороговой

#Производная функции активации

def Derivative\_act\_func(logistic, net):

return 1 / (2 \* (abs(net) + 1) \*\* 2) if logistic else 1

#Реальный будевый вектор

def Real\_vector():

real = []

for i in [[1] + list(i) for i in itertools.product([0, 1], repeat = 4)]:

real.append(str(Function(i)))

return ''.join(real)

#Функция, которой необходимо обучить НС

def Function(x):

return int(not (x[1] or x[2]) or x[3] or x[4])

if \_\_name\_\_ == "\_\_main\_\_":

#Пороговая на полном наборе

Selection(False, False)

#Логистическая на полном наборе

Selection(False, True)

#Пороговая на минимальной выборке

Selection(True, False)

#Логистическая на минимальной выборке

Selection(True, True)