|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **Московский государственный технический университет**  **имени Н.Э. Баумана** |

Факультет: Информатика и системы управления

Кафедра: Информационная безопасность

**Интеллектуальные технологии информационной безопасности**

**Лабораторная работа №1 на тему:**

«Исследование однослойных нейронных сетей на примере моделирования булевых выражений»

Вариант 4

Студент: Евула А. С.

Группа: ИУ8-63

Москва 2021

Цель работы

Исследовать функционирование простейшей нейронной сети на базе нейрона с нелинейной функцией активации и обучить ее по правилу Видроу-Хоффа.

Постановка задачи

Получить модель булевой функции на основе однослойной НС с двоичными входами , единичным входом смещения 0 = 1, синоптическими весами 01234, двоичным выходом и заданной функцией активации , реализовать обучение с использованием всех комбинаций входов и с частью возможных комбинаций.

Ход работы

Заданная функция

Таблица истинности БФ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1 | x2 | x3 | x4 | f |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Функции активации:

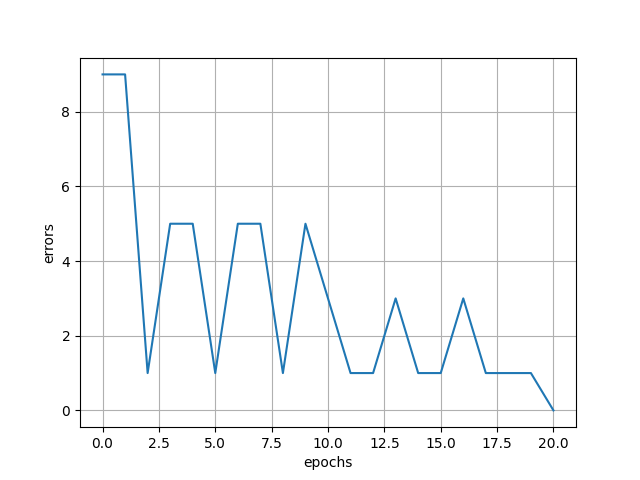
Для обучения использовалась норма обучения

Использование пороговой функции активации

Параметры НС на последовательных эпохах

|  |  |  |  |
| --- | --- | --- | --- |
| Номер эпохи k | Вектор весов w | Выходной вектор y | Суммарная ошибка E |
| 0 | 0 0 0 0 0 | 1111111111111111 | 9 |
| 1 | 0 0 0.6 0 0.3 | 1111111111111111 | 9 |
| … | … | … | … |
| 20 | -1.8 -0.3 1.8 0.9 0.9 | 0000111100000111 | 0 |

График суммарной ошибки НС по эпохам обучения

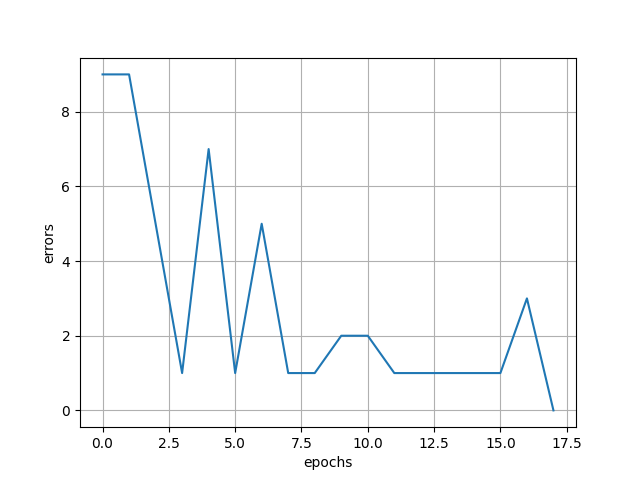


Использование сигмоидальной функции активации

Параметры НС на последовательных эпохах

|  |  |  |  |
| --- | --- | --- | --- |
| Номер эпохи k | Вектор весов w | Выходной вектор y | Суммарная ошибка E |
| 0 | 0 0 0 0 0 | 1111111111111111 | 9 |
| 1 | 0.06 0.09 0.21 0.14 0.09 | 1111111111111111 | 9 |
| … | … | … | … |
| 8 | -0.49 -0.03 0.52 0.12 0.21 | 0000111100000111 | 0 |

График суммарной ошибки НС по эпохам обучения



Обучение с неполным набором входов

Используя пороговую функцию:

Для заданной функции будет найден минимальный набор из 8 векторов:

x1 = 0100 x2 = 0101 x3 = 0110 x4 = 0111

x5 = 1101 x6 = 1100 x7 = 1110 x8 = 1111

Для обучения потребовалось 8 эпох, а конечный вектор коэффициентов

0 -0.3 0 0.9 0.9

Используя сигмоидальную функцию:

Для заданной функции будет найден минимальный набор из 8 векторов:

x1 = 0100 x2 = 0101 x3 = 0110 x4 = 0111

x5 = 1100 x6 = 1101 x7 = 1110 x8 = 1111

Для обучения потребовалось 13 эпох, а конечный вектор коэффициентов

0.03 -0.15 0.03 0.36 0.5

Приложение 1 – код программы

# Copyright 2021, Evula A. S., All rights reserved.

# IC8-63 BMSTU

from matplotlib import pyplot as plt

class NN:

def \_\_init\_\_(self, foo, border, d, maxEpoch=30):

self.W = [0] \* 5

self.t = [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1]

self.Rate = 0.3

self.epoch = 0

self.x = [

[0, 0, 0, 0],

[0, 0, 0, 1],

[0, 0, 1, 0],

[0, 0, 1, 1],

[0, 1, 0, 0],

[0, 1, 0, 1],

[0, 1, 1, 0],

[0, 1, 1, 1],

[1, 0, 0, 0],

[1, 0, 0, 1],

[1, 0, 1, 0],

[1, 0, 1, 1],

[1, 1, 0, 0],

[1, 1, 0, 1],

[1, 1, 1, 0],

[1, 1, 1, 1]

]

self.stable = [i for i in range(16)]

self.foo = foo

self.border = border

self.d = d

self.maxEpoch = maxEpoch

def kick(self):

cont = True

while cont:

stable = self.stable.copy()

res = False

for i in range(len(self.stable)):

del(self.stable[i])

res = self.work(visible=False)

if res: # got result

break

if not res: # no res -> reset

self.W = [0] \* 5

self.stable = stable.copy()

if not res:

cont = False

else:

output = f'''------------------------------------------------------------------

Epochs: {res['epochs']}

used vectors = {self.stable}'''

output += '\nW = '

for w in self.W:

output += str(round(w, 2)) + ' '

self.W = [0] \* 5

print(output)

def work(self, visible=True):

self.epoch = 0

E = 1

self.errors = []

while (E > 0):

if (self.epoch > self.maxEpoch):

return False

if visible:

print(f'''

---------------------------------

Epoch: {self.epoch}

---------------------------------''')

E = self.tick(visible)

self.errors.append(E)

self.epoch += 1

return {'epochs' : self.epoch - 1, 'weights' : self.W}

def tick(self, visible):

y = [0] \* 16

y\_K = [0] \* 16

w\_K = self.W.copy()

output = 'x 0 1 2 3 4 y t\n'

for ROW in self.stable:

x = self.x[ROW]

net = self.W[0]

net\_K = w\_K[0]

for i in range(len(x)):

net += x[i] \* self.W[i+1]

net\_K += x[i] \* w\_K[i+1]

out = int(self.foo(net) >= self.border)

out\_K = int(self.foo(net\_K) >= self.border)

y[ROW] = out

y\_K[ROW] = out\_K

output += ' 1 '

for i in x:

output += str(i) + ' '

output += ' ' + str(y\_K[ROW]) + ' ' + str(self.t[ROW]) + '\n'

self.correctWeights(x, y[ROW], self.t[ROW], d=self.d(net))

output += '\nW = '

for w in self.W:

output += str(round(w, 2)) + ' '

E = self.dist(y\_K)

output += '\nE = ' + str(E)

if visible:

print(output, end='')

return E

def correctWeights(self, x, y, t, d):

x = [1] + x

for i in range(len(self.W)):

self.W[i] += self.Rate \* (t - y) \* x[i] \* d

def dist(self, y\_K):

e = 0

for i in range(len(y\_K)):

e += abs(self.t[i] - y\_K[i])

return e

def graph(self, f\_name):

plt.plot([k for k in range(self.epoch)], self.errors)

plt.grid()

plt.xlabel('epochs')

plt.ylabel('errors')

plt.savefig(f'{f\_name}.png')

plt.cla()

def threshold(net):

return net

def thr\_d(net):

return 1

def logistic(net):

return 0.5\*(net/(1+abs(net)) + 1)

def log\_d(net):

return -0.5 \* (net/(net\*net+2\*net+1) - 1/(net+1))

print(''' (not x1 + x3)x2 + x2x4

0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1\n''')

print('''

=================================

THRESHOLD

=================================''', end='')

nn\_thr = NN(threshold, 0, thr\_d)

res\_thr = nn\_thr.work(visible=True)

nn\_thr.graph('threshold')

print('''

=================================

LOGISTIC

=================================''', end='')

nn\_log = NN(logistic, 0.5, log\_d)

res\_log = nn\_log.work(visible=True)

nn\_log.graph('logistic')

print('''

=================================

MIN VECTOR THRESHOLD

=================================''')

nn\_min = NN(threshold, 0, thr\_d, maxEpoch=res\_thr['epochs'])

nn\_min.kick()

print('''

=================================

MIN VECTOR LOGISTIC

=================================''')

nn\_min = NN(logistic, 0.5, log\_d, maxEpoch=res\_log['epochs'])

nn\_min.kick()