|  |  |
| --- | --- |
| 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 №3 на тему:**

«Исследование нейронных сетей с радиальными базисными функциями (RBF) на примере моделирования булевых выражений»

Вариант 4

Студент: Евула А. С.

Группа: ИУ8-63

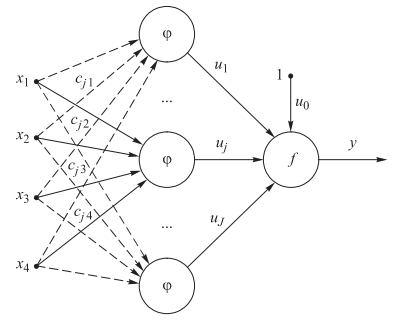
Москва 2021

Цель работы:

Исследовать функционирование НС с радиальными базисными функциями (RBF) и обучить ее по правилу Видроу-Хоффв.

Постановка задачи:

Получить модель булевой функции (БФ) на основе RBF-HC с двоичными входами x1,2,3,4 единичным входом смещения равным 1, синаптическими весами v0,1,2,3,4, двоичным выходом y, пороговой ФА выходного нейрона, J скрытыми RBF-нейронами с гауссовой ФА и координатами центров cj1,j2,j3,j4.



**Моделируемая БФ**:

**Функции активации:**

Ход работы

Вектор данной БФ = [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1]

Исходные веса примем нулевыми, норма обучения равна 0.3 (исходя из тестов, лучшее время обучения).

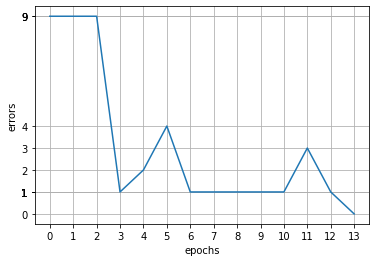


Рисунок 1 - график зависимости ошибки от эпохи

По исходным данным, нейронная сеть обучается за ~13 эпох.

Исходный код программы и ее вывод представлены в Приложении.

Приложение

Исходный код программы (НС)

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

# IC8-63 BMSTU

from math import exp, sqrt

from matplotlib import pyplot as plt

class NN():

def \_\_init\_\_(self):

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

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

zeros, ones = self.t.count(0), self.t.count(1)

el = 0 if zeros<=ones else 1

self.J = min(zeros, ones)

self.c = [self.x[i] for i, e in enumerate(self.t) if e == el]

self.W = [0] \* (self.J+1)

self.Rate = 0.3

self.MaxEpoch = 20

self.MinError = 0

def train(self):

epoch = 0

E = 1

self.res = [[], []]

while E > self.MinError:

if epoch > self.MaxEpoch:

return print('ERR: epoch')

E = round(self.tick(), 3)

self.res[0].append(epoch)

self.res[1].append(E)

epoch += 1

return 0

def tick(self):

y = []

y\_K = []

w\_K = self.W.copy()

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

phi = self.hiden(self.x[i])

net = self.W[0]

net\_K = w\_K[0]

for j in range(len(phi)):

net += self.W[j+1] \* phi[j]

net\_K += w\_K[j+1] \* phi[j]

y.append(int(net >= 0))

y\_K.append(int(net\_K >= 0))

self.correctWeights(phi=[1]+phi, delta=self.t[i]-y[i])

E = 0

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

E += 1 if self.t[i] != y\_K[i] else 0

return E

def hiden(self, x):

res = []

for j in range(self.J):

s = 0

for i in range(len(x)):

s -= (x[i] - self.c[j][i]) \* (x[i] - self.c[j][i])

res.append(exp(s))

return res

def correctWeights(self, phi, delta):

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

self.W[j] += self.Rate \* delta \* phi[j]

def print(self, graph=False):

for j in range(len(self.c)):

print(f'c{j+1} = {self.c[j]}')

print()

for i in range(len(self.res[0])):

print(f'K = {self.res[0][i]}\tE = {self.res[1][i]}')

print()

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

print(f'w{j} = {round(self.W[j], 2)}')

if graph:

plt.plot(self.res[0], self.res[1])

plt.grid()

plt.xlabel('epochs')

plt.ylabel('errors')

plt.xticks(self.res[0])

plt.yticks(self.res[1])

plt.savefig('graph.png')

def main():

nn = NN()

nn.train()

nn.print(graph=True)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Вывод программы

c1 = [0, 1, 0, 0]

c2 = [0, 1, 0, 1]

c3 = [0, 1, 1, 0]

c4 = [0, 1, 1, 1]

c5 = [1, 1, 0, 1]

c6 = [1, 1, 1, 0]

c7 = [1, 1, 1, 1]

K = 0 E = 9

K = 1 E = 9

K = 2 E = 9

K = 3 E = 1

K = 4 E = 2

K = 5 E = 4

K = 6 E = 1

K = 7 E = 1

K = 8 E = 1

K = 9 E = 1

K = 10 E = 1

K = 11 E = 3

K = 12 E = 1

K = 13 E = 0

w0 = -0.9

w1 = 0.42

w2 = 0.52

w3 = 0.53

w4 = 0.33

w5 = 0.42

w6 = 0.44

w7 = 0.53