****

**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Институт № 3**

**Кафедра 311**

**Интеллектуальные системы**

**Лабораторная работа № 3**

**«Нейронная сеть Хоппфилда»**

**Выполнил студент  
Веденеев Максим Кириллович**

**Группа М3З-501-БК**

**Дата 06.10.2023 г.**

**Принял преподаватель  
Кос Оксана Игоревна**

Оглавление

[Цель лабораторной работы 3](#__RefHeading___1)

[Глава 1. Принцип нейронной сети Хопфилда 4](#__RefHeading___2)

[Глава 3. Реализация 5](#__RefHeading___3)

[Глава 4. Листинг программы 7](#__RefHeading___4)

[Итоги лабораторной работы 12](#__RefHeading___5)

[Список литературы 13](#__RefHeading___6)

# Цель лабораторной работы

Изучить:

1. Принцип работы нейронной сети Хопфилда.
2. Реализовать его.

# Глава 1. Принцип нейронной сети Хопфилда

Нейро́нная сеть Хо́пфилда (англ. Hopfield network) (Рисунок 1.1) — полносвязная нейронная сеть с симметричной матрицей связей. В процессе работы динамика таких сетей сходится (конвергирует) к одному из положений равновесия. Эти положения равновесия определяются заранее в процессе обучения, они являются локальными минимумами функционала, называемого энергией сети (в простейшем случае — локальными минимумами отрицательно определённой квадратичной формы на n-мерном кубе). Такая сеть может быть использована как автоассоциативная память, как фильтр, а также для решения некоторых задач оптимизации. В отличие от многих нейронных сетей, работающих до получения ответа через определённое количество тактов, сети Хопфилда работают до достижения равновесия, когда следующее состояние сети в точности равно предыдущему: начальное состояние является входным образом, а при равновесии получают выходной образ.

Нейронная сеть Хопфилда устроена так, что её отклик на запомненные m эталонных «образов» составляют сами эти образы, а если образ немного исказить и подать на вход, он будет восстановлен и в виде отклика будет получен оригинальный образ. Таким образом, сеть Хопфилда осуществляет коррекцию ошибок и помех.

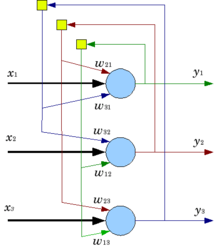


Рисунок 1.1 – Схема сети Хопфилда с тремя нейронами

# Глава 3. Реализация

Обучим сеть распознавать следующие образы:

@-----@  
@---@@-  
@--@---  
@@@----  
@--@---  
@---@@-  
@-----@

-@@@@@-  
-@---@-  
-@---@-  
-@---@-  
-@---@-  
-@---@-  
-@---@-

@@@@@@@  
---@---  
---@---  
---@---  
---@---  
---@---  
---@---

На вход будем подавать искаженный образ буквы К:

@@--@--  
@---@@-  
@--@---  
@-@----  
@--@---  
@-@-@@-  
@@----@

Алгоритм берёт случайный нейрон из матрицы и из входного вектора и пытается определить необходимость инверсии значения нейрона.

В случае, если алгоритм 200 раз подряд не смог найти нейрон для инверсии, то сеть принимает решение прервать поиск и сообщает, что исходный образ определить не удалось.

Результат выполнения программы:

Success. Shape recognized in 200 iterations:

@-----@

@---@@-

@--@---

@@@----

@--@---

@---@@-

@-----@

# Глава 4. Листинг программы

#!/usr/bin/python

# -\*- coding: cp1251 -\*-

import sys

from hopfield.parser import parse, parse\_shape, SHAPE\_SIDE

import random

import math

dictionary = {-1: '-', 1: '@'}

def cmp(a, b):

if a < b:

return -1

if a == b:

return 0

if a > b:

return 1

def charfor(x):

return dictionary[x]

def printshape(obraz, size):

i = 0

out\_str = ""

for o in obraz:

out\_str += str(charfor(o))

i += 1

if i % size == 0:

print(out\_str)

out\_str = ""

# Нейронная сеть Хопфилда

class HopfieldNet:

# Инициализация сети

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

self.drawsteps = False

self.W = [] # матрица весовых коэффициентов

self.shapes = [] # образы, которым обучена сеть

self.neurons = int(math.pow(shapeside, 2)) # количество нейронов

self.shapeside = shapeside # количество бит в стороне образа

r = range(0, self.neurons)

for i in r:

self.W.append([0 for x in r])

self.maxjunkiters = 200 # максимальное число безрезультатных итераций

self.junkiter = 0 # текущая безрезультатная итерация

# Обучение сети образу

def teach(self, shape):

self.shapes.append(shape)

self.X = shape

r = range(0, self.neurons)

for i in r:

for j in r:

if i == j:

self.W[i][j] = 0

else:

self.W[i][j] += self.X[i] \* self.X[j]

#

# Распознать изменённый образ

#

def recognize(self, shape):

self.Y = shape

iter = 0

# пока не совпадёт с одним из известных...

while (self.shapes.count(self.Y) == 0):

self.recstep()

iter += 1

# ... или количество безрезультатных итераций не истечёт

if self.junkiter >= self.maxjunkiters:

return (False, self.Y, iter)

return (True, self.Y, iter)

# Шаг распознования.

# Случайно выбирается нейрон для обновления

def recstep(self):

signum = lambda x: cmp(x, 0)

r = random.randrange(0, self.neurons, 1)

net = 0

for i in range(0, self.neurons):

net += self.Y[i] \* self.W[i][r]

signet = signum(net)

if signet != self.Y[r]: # заменяем текущий нейрон

print(f"Neuron {r} : {self.Y[r]} -> {signet}")

self.Y[r] = signet

if self.drawsteps:

printshape(self.Y, self.shapeside)

self.junkiter = 0

else:

self.junkiter += 1

knows\_shapes\_dir = "known\_shapes"

modified\_shape\_path = "modified\_shapes/modified.txt"

# Главная функция

def main(argv):

shapes = parse(knows\_shapes\_dir)

shape = parse\_shape(modified\_shape\_path)

print("Known Shapes:")

for o in shapes:

printshape(o, SHAPE\_SIDE)

print(o)

print("Teaching...")

hopfield = HopfieldNet(SHAPE\_SIDE)

for o in shapes:

hopfield.teach(o)

print("Modified shape:")

printshape(shape, SHAPE\_SIDE)

print("Shape recognizing...")

(recognized, recshape, iters) = hopfield.recognize(shape)

if recognized:

print("Success. Shape recognized in %g iterations:" % iters)

else:

print("Fail. Shape not recognized in %g iterations..." % iters)

printshape(recshape, SHAPE\_SIDE)

# Точка входа

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

main(sys.argv[1:])

# Итоги лабораторной работы

Мы изучили и научились реализовывать нейронные сети Хопфилда.

# Список литературы

1. [[bitsofmind.org](https://bitsofmind.wordpress.com/2008/08/05/hopfield_net/)] – Нейронная сеть Хопфилда