Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №3**

**Дисциплина: Приложение нейросетевых алгоритмов**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. В. Яценко

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Шиян

**Цель работы**

Получение знаний и практических навыков построения

однослойных нейронных сетей.

***Задача – Аппроксимация данных***

1. **Описание решения**

Для решения задачи была создана нейросеть на основе радиально симметричных функций.

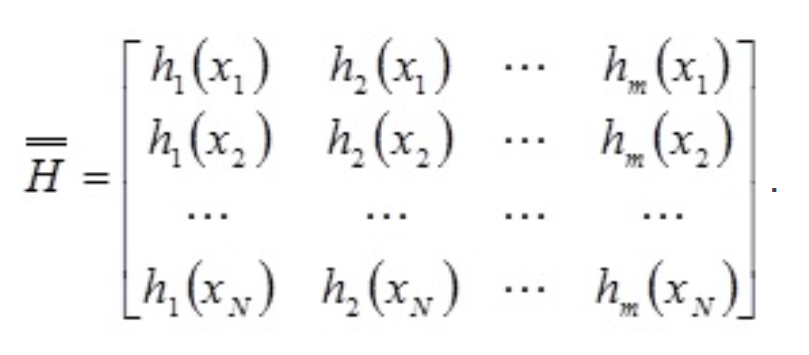
Такие нейросети используют для аппроксимации, классификации и кластеризации данных. Их очень удобно использовать в том случае, когда доступно большое количество обучающих векторов.

В качестве обучающей выборки дан набор из 9 пар точек: значений независимой переменной x и значение отклика y.

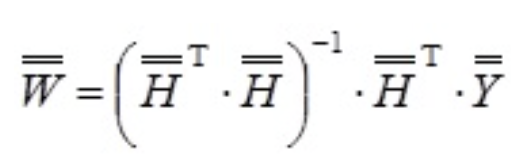
Также в качестве центров нейронов выбраны значения независимой переменной в опытах 1, 3, 5, 7 и 9.

Алгоритм решения задачи можно описать так:

1. Создаем нейросеть с 5 нейронами, нейроны инициализируем центрами
2. Для данных нейронов считаем характеристическую матрицу



1. С помощью алгебры рассчитывается матрица весовых коэффициентов выходного слоя нейронов



В процессе вычислений также использовалась функция Гаусса.

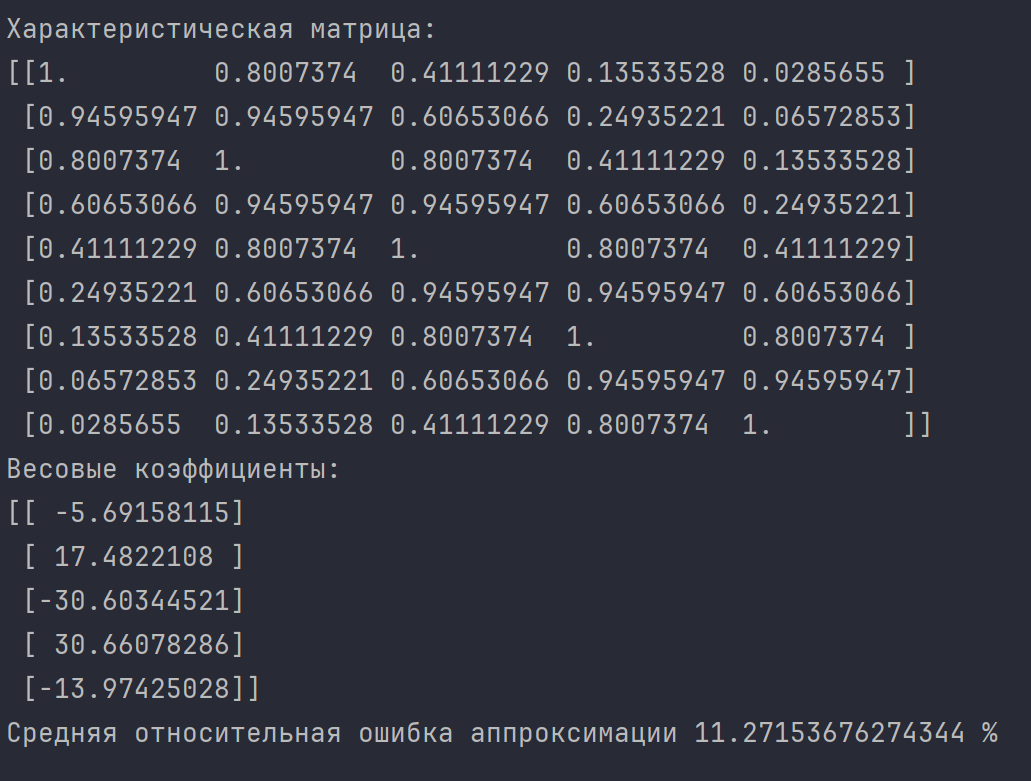
В самой программе был создан один класс Neuron, инициализация которого происходит с помощью задания центра. В этом классе есть один метод – count – для подсчета. Класс нейросети не создавался, так как в этом не было особой необходимости.

1. **Реализация на языке Python**

from math import exp, fabs  
import numpy as np  
from numpy import linalg as la  
import matplotlib.pyplot as plt  
  
class Neuron:  
 center = 0  
  
 def \_\_init\_\_(self, center):  
 # инициализируем нейрон центром  
 self.center = center  
  
 def count(self, x):  
 return exp(-((x - self.center) \*\* 2) / ((2 \* 1.5 \*\* 2)))  
  
# обучающая выборка  
values = [  
 [-2.0, -0.48],  
 [-1.5, -0.78],  
 [-1.0, -0.83],  
 [-0.5, -0.67],  
 [0.0, -0.20],  
 [0.5, 0.70],  
 [1.0, 1.48],  
 [1.5, 1.17],  
 [2.0, 0.20]  
]  
  
x = [value[0] for value in values]  
y = np.array([[values[i][1]] for i in range(9)])  
  
# центры (пока не понятно че это такое)  
centers = [-2.0, -1.0, 0.0, 1.0, 2.0]  
# радиус тоже не понятно зачем  
radius = 1.5  
# тогда параметр a будет равен 0.22  
  
# создаем нейроны  
neurons = [Neuron(centers[i]) for i in range(len(centers))]  
  
# после создания нейронов нужно создать характеристическую матрицу  
# количество строк этой матрицы = количеству примеров из выборки  
# количество столбцов - количество нейронов  
h = np.array([[neurons[i].count(x[j]) for i in range(len(centers))] for j in range(len(x))])  
print("Характеристическая матрица:")  
print(h)  
  
# теперь определяем синаптические коэффициенты выходного нейрона  
# для этого используем еще одну формулу  
# w = (h^T \* h)^(-1) \* h^T \* y  
  
# коротко о том, что обозначает каждая функция  
# dot - скалярное произведение массивов  
# inv - обратная матрица  
# transpose - транспонированная  
w = np.dot(la.inv(np.dot(h.transpose(), h)), np.dot(h.transpose(), y))  
print("Весовые коэффициенты:")  
print(w)  
  
# теперь считаем выходные значения на основе весовых коэффициентов  
answers = [sum([neurons[i].count(value[0]) \* w[i][0] for i in range(5)]) for value in values]  
  
# считаем среднюю относительную ошибку аппроксимации  
n = len(x)  
s = 0  
for i in range(n):  
 s += fabs(1 - (y[i][0] / answers[i]))  
print("Средняя относительная ошибка аппроксимации " + str(s / n \* 100) + " %")  
  
  
# теперь нужно все нарисовать  
plt.scatter(x, y, c="slateblue", label="Исходные точки выборки")  
plt.plot(x, answers, color="darkturquoise", label="Полученная аппроксимация")  
plt.legend()  
plt.grid(True)  
plt.title("Положение исходных точек\n и график аппроксимации")  
plt.show()

1. **Запуск программы**

В консоли мы видим промежуточные матрицы, а также относительную ошибку аппроксимации:



Еще один результат работы программы – график, он выглядит так:

Изображение выглядит как текст, диаграмма, линия, График

Автоматически созданное описание

**Вывод**

Таким образом, в ходе работы удалось написать нейросеть на основе радиально симметричных функций. Написанная программа верно считает характеристическую матрицу, матрицу весовых коэффициентов, а также строит график полученной аппроксимации. Результаты получились близки к реальным, относительная ошибка вычислений составила 11 %