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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Паничева

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 1:** рассмотреть пример решения задачи аппроксимации данных с использованием искусственной нейронной сети на основе радиально симметричных функций. Дана экспериментальная зависимость в виде набора из 9 пар точек: значений независимой переменной x и соответствующих им значений функции отклика y, представленных в таблице.

Для решения задачи использовалась нейросеть на основе радиально симметричных функций. Обучающая выборка состояла из 9 пар точек со значениями переменной x и соответствующими значениями y. Центры нейронов были выбраны как значения переменной x в опытах 1, 3, 5, 7 и 9.

Алгоритм работы нейросети:

1. Нейросеть инициализируется с заданными центрами нейронов.

2. Характеристическая матрица G вычисляется на основе входных значений и центров нейронов.

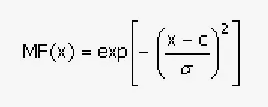
3. Вектор весов w вычисляется по специальной формуле: 

4. Выходные значения сети вычисляются для каждого входного значения с использованием весов и значений функций активации.

5. Оценивается ошибка аппроксимации.

6. Результаты визуализируются на графике.

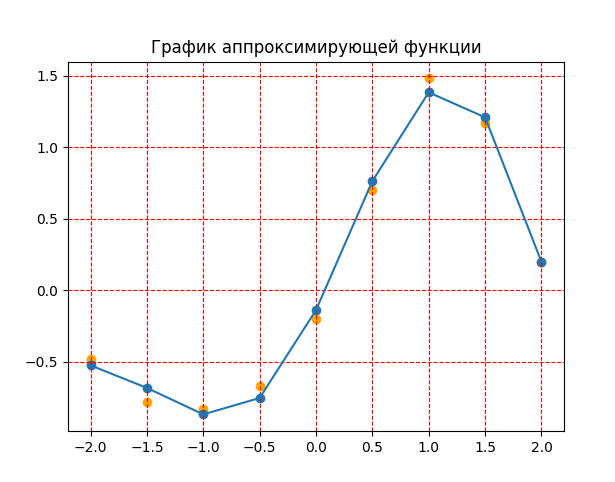
В процессе вычислений также использовалась функция Гаусса для вычисления характеристической матрицы:



Код программы:

**from** math **import** \*  
**import** numpy **as** np  
**from** numpy **import** linalg **as** la  
**import** matplotlib.pyplot **as** plt  
  
**class** Neuron:  
 center = 0  
  
 **def** \_\_init\_\_(self, center): # центр нейрона  
 self.center = center  
  
 **def** calculate(self, x): # функция Гаусса  
 **return** exp(-((x - self.center) \*\* 2) / (2 \* 1.5 \*\* 2))  
  
# обучающие примеры  
inputs = [  
 [-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 = [i[0] **for** i **in** inputs]  
y = np.array([[inputs[i][1]] **for** i **in** range(len(inputs))])  
# исходные точки на графике  
plt.scatter(x, y, c="orange")  
  
# инициализация центров скрытых нейронов в опытах 1, 3, 5, 7 и 9  
centers = [-2.0, -1.0, 0.0, 1.0, 2.0]  
  
# создаем нейроны  
neurons = [Neuron(centers[i]) **for** i **in** range(len(centers))]  
  
# характеристическая матрица  
G = [[neurons[i].calculate(x[j]) **for** i **in** range(len(centers))] **for** j **in** range(len(x))]  
G = np.array(G)  
print("Характеристическая матрица:")  
print(G)  
  
# веса по формуле:  
# w = (G^T \* G)^(-1) \* G^T \* y  
w = np.dot(la.inv(np.dot(G.transpose(), G)), np.dot(G.transpose(), y))  
print("Веса")  
print(w)  
  
# выходные значения на основе весовых коэффициентов  
outputs = [sum([neurons[i].calculate(val[0]) \* w[i][0] **for** i **in** range(5)]) **for** val **in** inputs]  
  
outputs = np.array(outputs)  
print("Выходные значения " + str(outputs))  
  
# считаем среднюю относительную ошибку аппроксимации  
n = len(y)  
summ = 0  
**for** i **in** range(n):  
 summ += fabs(1 - (y[i][0] / outputs[i]))  
print("Средняя ошибка аппроксимации " + str(round((summ / n ), 5)))  
  
#график  
plt.scatter(x, outputs)  
plt.plot(x, outputs)  
plt.grid(color='r', linestyle='--')  
plt.title("График аппроксимирующей функции")  
plt.show()

График результатов:



**Вывод:** В ходе работы была реализована нейронная сеть на основе радиально симметричных функций для аппроксимации заданной функции. Программа корректно вычисляет характеристическую матрицу, определяет вектор весовых коэффициентов и строит график полученной аппроксимации. Результаты демонстрируют высокую точность, с относительной ошибкой вычислений 0.11, подтверждая эффективность данной нейросети для решения задач аппроксимации.