МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №2

з дисципліни: “Нейронні мережі”

Виконали:

студенти групи КА-83

Самошин А.О.

Цепа О.Ю.

Перевірив:

Данилов В.Я.

КИЇВ 2020

Завдання

1. Реалізувати нейронну мережу на основі радіальних базисних функцій, використовуючи такі мови програмування як Python, C++, Java.

2. За допомогою побудованої нейроної мережі розв’язати задачу згідно з номером варіанту. (Номер варіанту визначається за номером у списку групи.) Для цього на основі відповідного файлу необхідно випадковим чином сформувати навчальну та тестову вибірки (у співвідношенні 4:1). Навчити нейронну мережу на навчальній вибірці, використовуючи різні активаційні функції. Порівняти результати.

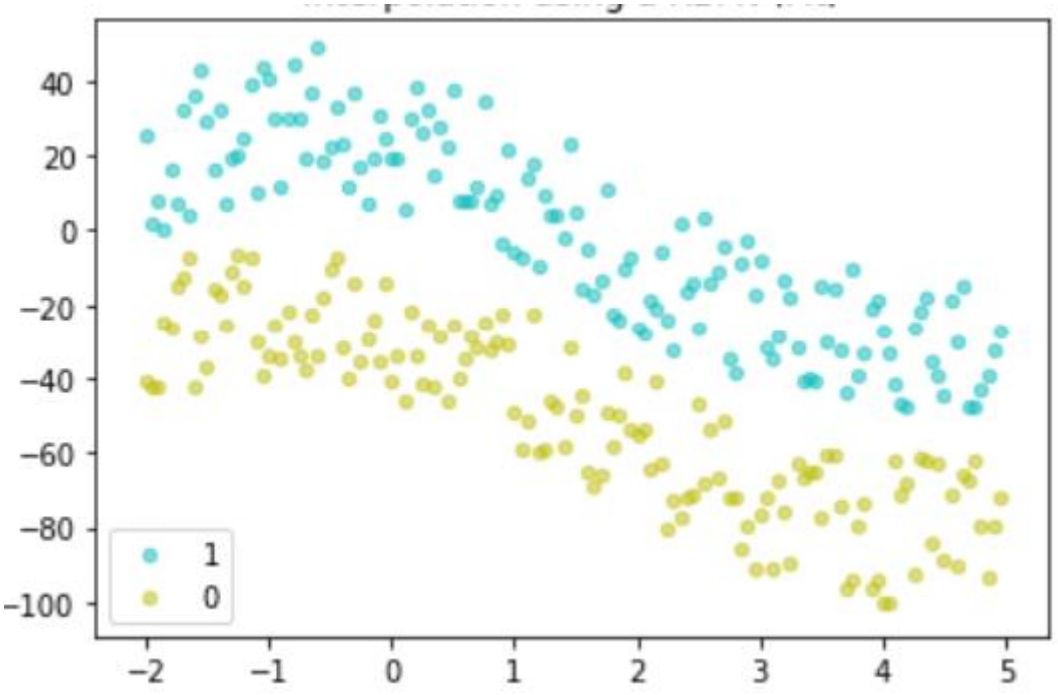
3. Перевірити роботу нейронної мережі на тестових даних.

Мета: отримати навички розв’язання практичних задач за допомогою радикальних базисних функцій.

Розв’язання

На вході отримаємо масив даних – координати точок та приналежність до певного класу (0 або 1).

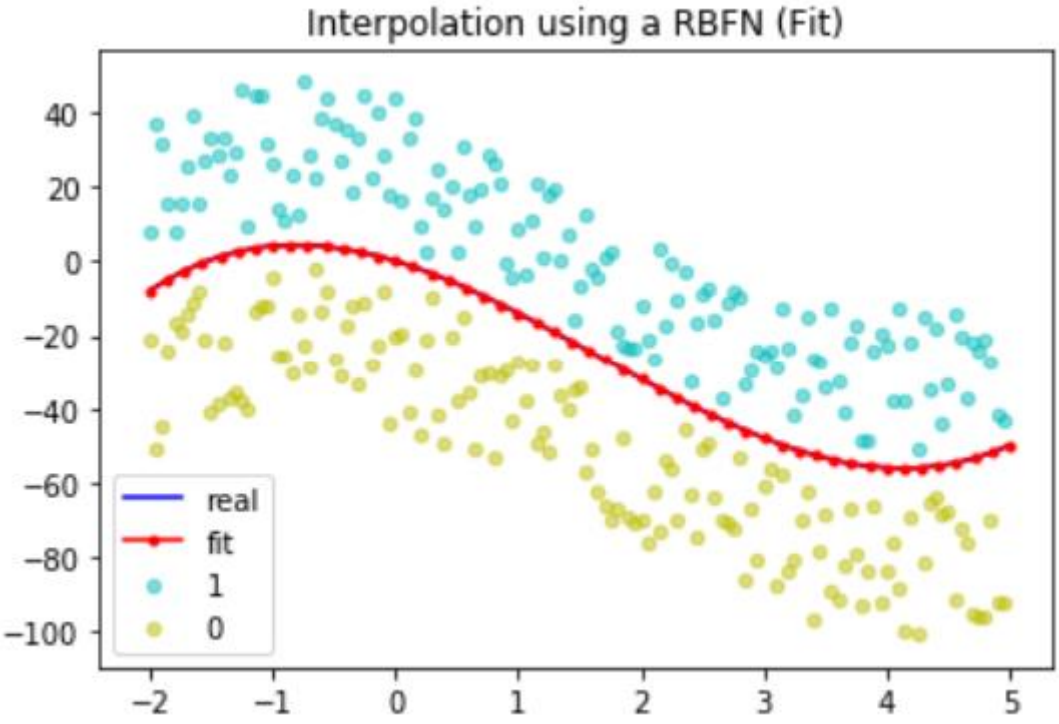
Для кращого розуміння як саме розподілені дані та можливість використання RDF – функцій зобразимо їх на площині.



Синім та жовтим кольором виділені два різних класи даних, тобто клас 0 та клас 1.

Результатом роботи мережі є апроксимована функція, за допомогою якої ми можемо розділити два класи даних, таким чином усі нові дані які будуть знаходитися вище функції будуть належати до одного класу, а усі

що знаходяться під функцією – до іншого.



Апроксимована функція майже дорівнює заданій, далі це побачимо на графіку нев’язки.

Після навчання мережі вихідним параметром є 50-вимірний вектора, кожна координата якого є «вагою» відповідних координат.

Лістинг програми

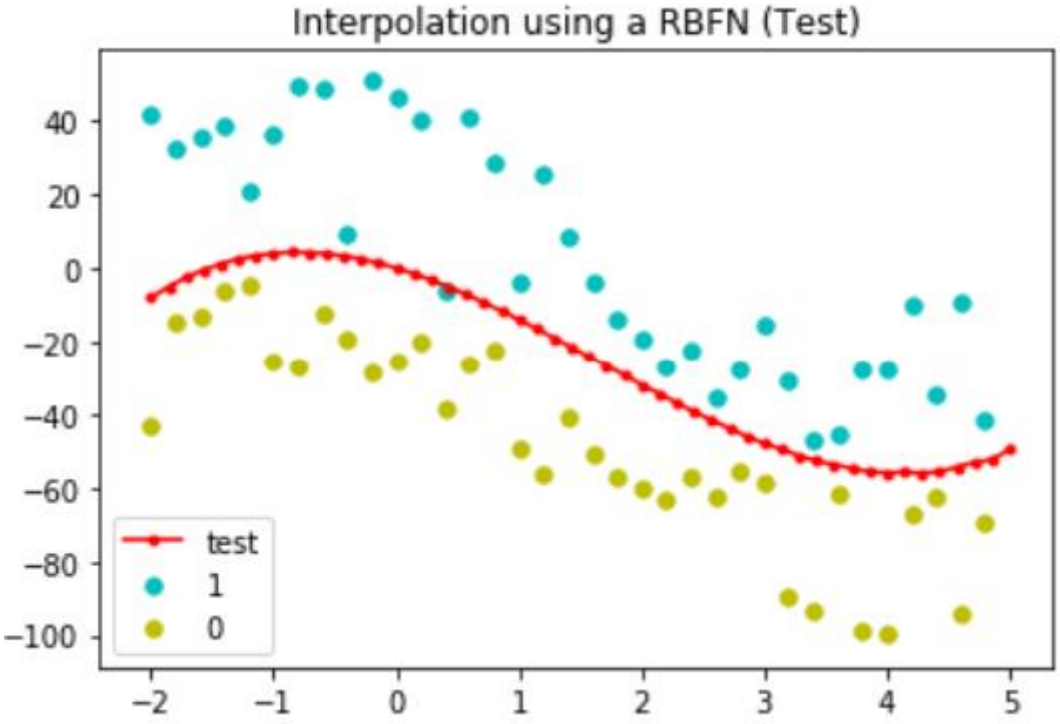
```
import numpy as np
import matplotlib.pyplot as plt


class RBFN(object):

    def __init__(self, hidden_shape, sigma=1.0):

        self.hidden_shape = hidden_shape
        self.sigma = sigma
        self.centers = None
        self.weights = None

    def _kernel_function(self, center, data_point):
        return np.exp(-self.sigma*np.linalg.norm(center-data_point)**2)

    def _calculate_interpolation_matrix(self, X):
        """ Calculates interpolation matrix using a kernel_function
        # Arguments
            X: Training data
        # Input shape
            (num_data_samples, input_shape)
```

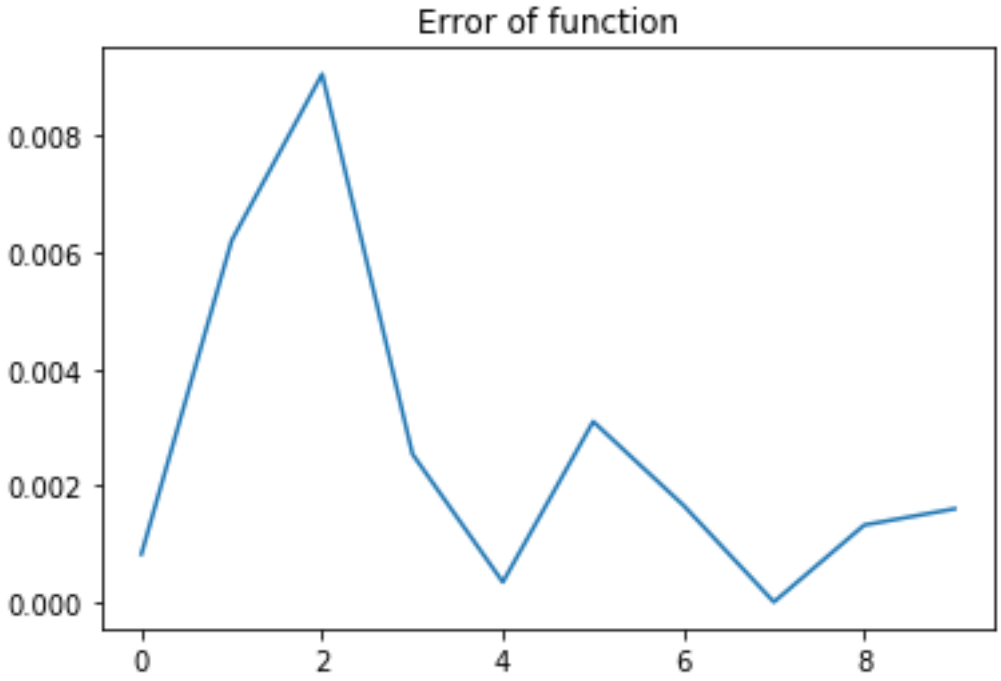
```
        # Returns
            G: Interpolation matrix
        """
        G = np.zeros((len(X), self.hidden_shape))
        for data_point_arg, data_point in enumerate(X):
            for center_arg, center in enumerate(self.centers):
                G[data_point_arg, center_arg] = self._kernel_function(center, data_point)
        return G

    def _select_centers(self, X):
        random_args = np.random.choice(len(X), self.hidden_shape)
        centers = X[random_args]
        return centers

    def fit(self, X, Y):

        self.centers = self._select_centers(X)
        G = self._calculate_interpolation_matrix(X)
        self.weights = np.dot(np.linalg.pinv(G), Y)
        print (self.weights)

    def predict(self, X):

        G = self._calculate_interpolation_matrix(X)
        predictions = np.dot(G, self.weights)
        return predictions


# generating data
x = np.linspace(-2, 5, 50)
y = x**3-5*x**2-10*x


# fitting RBF-Network with data
model = RBFN(hidden_shape=50, sigma=5)
model.fit(x, y)
y_pred = model.predict(x)

err = []
for i in range(50):

    error = ((y_pred[i] - (x[i]**3-5*x[i]**2-10*x[i]))**2)/(i+1)
    err.append(error)
print(err)
plt.plot(x, err)
plt.show()

# plotting 1D interpolation
plt.plot(x, y, 'b-', label='real')
plt.plot(x, y_pred, 'r-', label='fit', marker='.')
plt.margins(0.05)
```

```
X1 = []
Y1 = []
X2 = []
Y2 = []
for i in np.arange(-2, 5, 0.05):
    f = i**3-5*i**2-10*i
    Y1.append(np.random.uniform(f+5, f+45))
    X1.append(i)
    Y2.append(np.random.uniform(f-45, f-5))
    X2.append(i)
plt.scatter(X1, Y1, color='c', label='1', s=20, alpha=0.5)
plt.scatter(X2, Y2, color='y', label='0', s=20, alpha=0.5)
plt.title('Interpolation using a RBFN (Fit)')
plt.legend(loc=3)
plt.show()

plt.plot(x, y_pred, 'r-', label='test', marker='.')
plt.margins(0.05)

X1 = []
Y1 = []
X2 = []
Y2 = []
for i in np.arange(-2, 5, 0.2):
    f = i**3-5*i**2-10*i
    Y1.append(np.random.uniform(f-2, f+50))
    X1.append(i)
    Y2.append(np.random.uniform(f-45, f-5))
    X2.append(i)
plt.scatter(X1, Y1, color='c', label='1', s=30)
plt.scatter(X2, Y2, color='y', label='0', s=30)
plt.title('Interpolation using a RBFN (Test)')
plt.legend(loc=3)
plt.show()
```

Результати роботи програми

Після навчання мережі перевіримо результати на тестовій вибірці.





Висновки

У даній лабораторній роботі я навчився працювати з нейронною мережею, використовуючи RBF - функції та застосовувати отримані навички на практиці.