Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 4

по курсу «Нейроинформатика»

Студент: Аксенов А. Е.
Группа: М80-408Б-20
Преподаватель: Горохов М. А.
Оценка:

Москва, 2023

Цель работы

Целью работы является исследование свойств некоторых видов сетей с радиальными базисными элементами, алгоритмов обучения, а также применение сетей в задачах классификации и аппроксимации функции.

Основные этапы работы:

1. Использовать вероятностную нейронную сеть для классификации точек в случае, когда классы не являются линейно разделимыми.
2. Использовать сеть с радиальными базисными элементами (RBF) для классификации точек в случае, когда классы не являются линейно разделимыми.
3. Использовать обобщенно-регрессионную нейронную сеть для аппроксимации функции. Проверить работу сети с рыхлыми данными.

Оборудование

Процессор : Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz
ОЗУ : 16 ГБ

Программное обеспечение

Python 3.8 + Jupyter Notebook

Сценарий выполнения работы

1. Использовать вероятностную нейронную сеть для классификации точек в случае, когда классы не являются линейно разделимыми.

Вариант

1.	Эллипс: $a = 0.3$, $b = 0.3$, $\alpha = 0$, $x_0 = 0$, $y_0 = 0$ Эллипс: $a = 0.7$, $b = 0.7$, $\alpha = 0$, $x_0 = 0$, $y_0 = 0$ Эллипс: $a = 1$, $b = 1$, $\alpha = 0$, $x_0 = 0$, $y_0 = 0$

1.1 Инициализируем область определения и значения функций. Отрисуем фигуры и раскрасим согласно классу

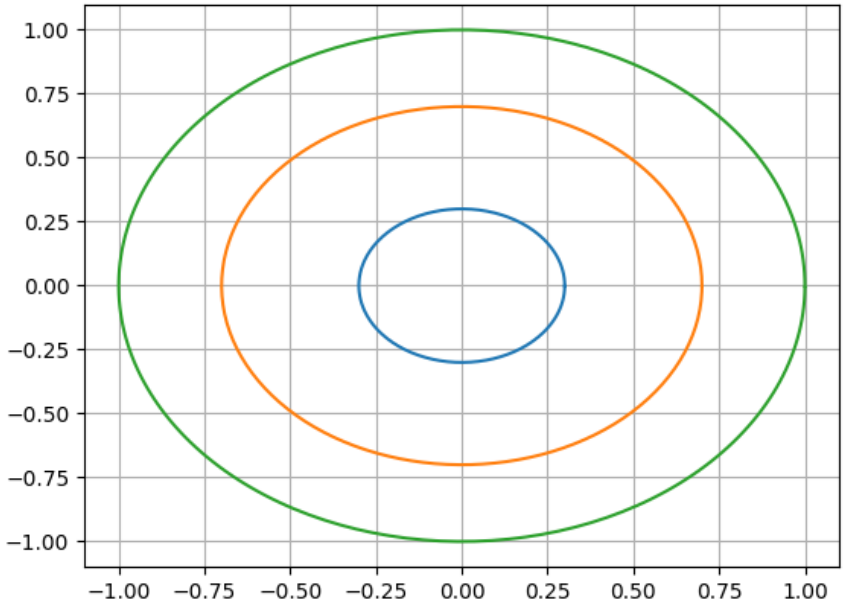

Рис. 1 Вся выборка

1.2 Для первого класса, второго класса и третьего класса возьмем соответственно 60, 100, 120 элементов выборки случайным образом. Отрисуем их

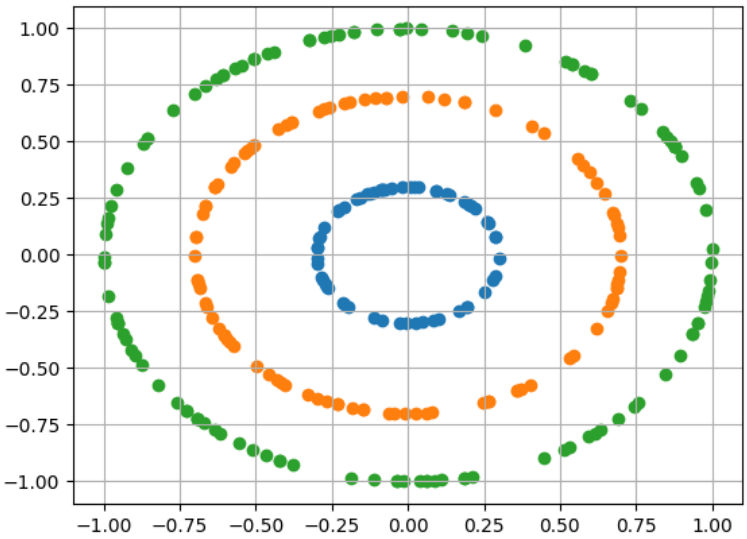

Рис. 2 Подвыборка

1.3 Создадим вероятностную нейронную сеть с параметром SPREAD = 0.3. Обучим сеть и отобразим структуру сети.

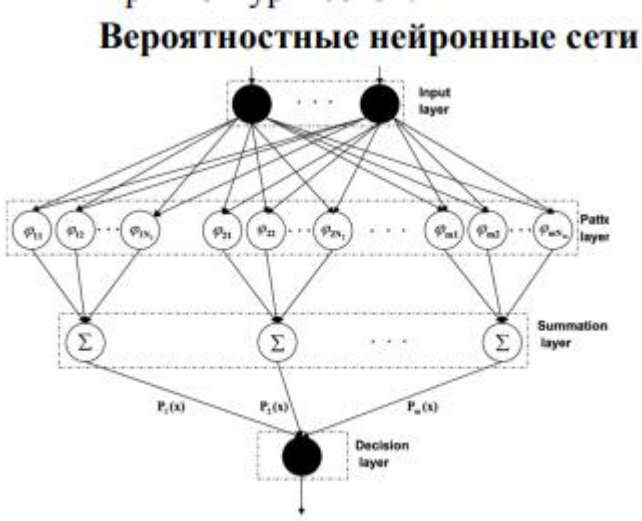

Рис. 3 Структура сети

1.4 Посчитаем долю верный ответов для обучающей и тестовой выборке.

```
# Точностьность на обучающей выборке
accuracy_score(pnn.predict(x_train), y_train)
```

```
1.0
```

```
# Точность на тестовой выборке
accuracy_score(pnn.predict(x_test), y_test)
```

```
1.0
```

Рис. 4 Доля верных ответов на обучающей и тестовой выборки

1.5 Классифицируем точки для области [-1.2,1.2]x[-1.2,1.2] и отобразим результат

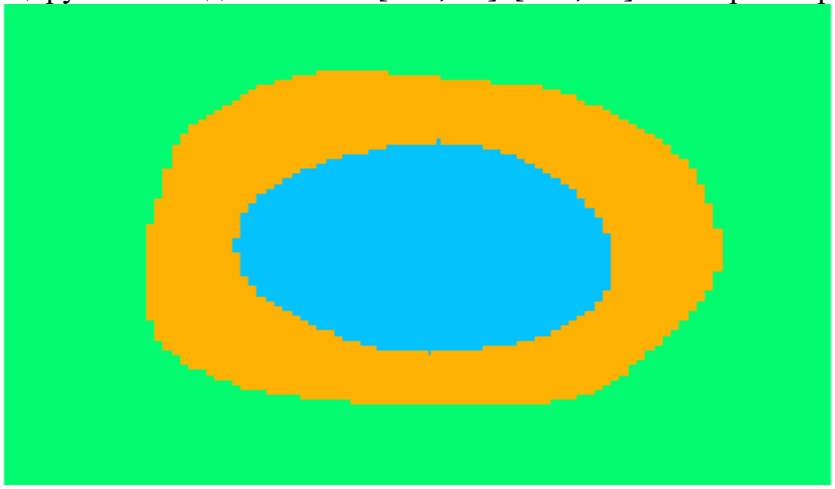

Рис. 5 Классификация точек в области [-1.2,1.2]X[-1.2,1.2]

1.6 Создадим вероятностную нейронную сеть с параметром SPREAD = 0.1. Посчитаем долю верный ответов для трейна и теста.

```
Доля верный ответов на трейне при Spread = 0.1
     1

Доля верный ответов на тесте при Spread = 0.1
     1
```

Рис. 6 Доля верных ответов на обучающей и тестовой выборки

1.7 Классифицируем точки для области [-1.2,1.2]x[-1.2,1.2] и отобразим результат

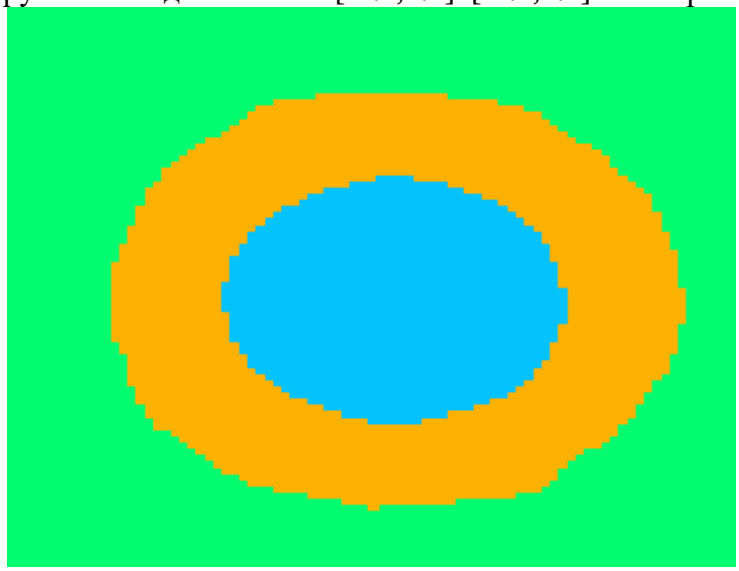

Рис. 7 Классификация точек в области [-1.2,1.2]X[-1.2,1.2]

Обе модели идеально классифицировали обучающее множества. Но модель с параметром std=0.1 более ровные границы множеств чем модель с параметром std=0.3

2. Использовать сеть с радиальными базисными элементами (RBF) для классификации точек в случае, когда классы не являются линейно разделимыми.

2.1 Инициализируем область определения и значения функций. Отрисуем фигуры и раскрасим согласно классу

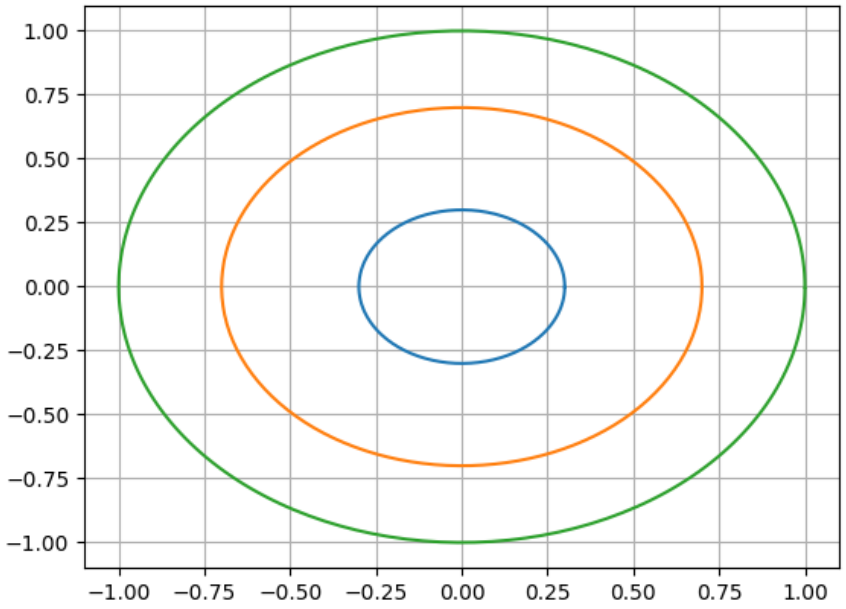

Рис. 8 Вся выборка

2.2 Для первого класса, второго класса и третьего класса возьмем соответственно 60, 100, 120 элементов выборки случайным образом. Разделим данные на выборки для обучения и теста в соотношении 0.8 , 0.2. Отрисуем их

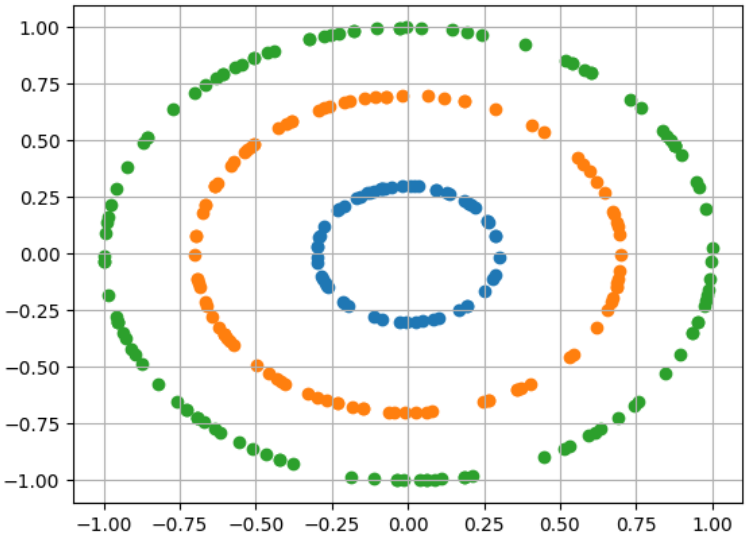

Рис. 9 Подвыборка

2.3 Создадим сеть с радиальными базисными элементами (RBF) с параметром SPREAD = 0.3. Обучим сеть и отобразим структуру сети.

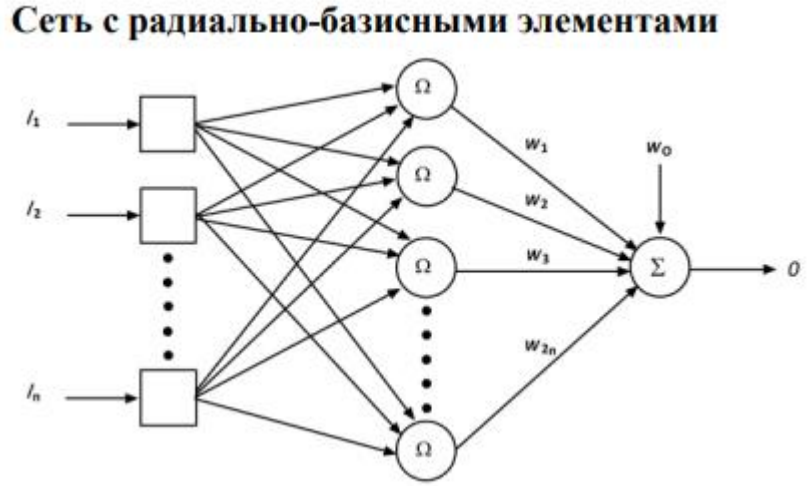

Рис. 10 Структура сети

2.4 Посчитаем долю верный ответов для обучающей и тестовой выборки.

```
# Точность на обучающей выборке
accuracy_score(svc.predict(x_train), y_train)
```

```
1.0
```

```
# Точность на тестовой выборке
accuracy_score(svc.predict(x_test), y_test)
```

```
1.0
```

Рис. 11 Доля верных ответов

2.5 Классифицируем точки для области [-1.2,1.2]x[-1.2,1.2] и отобразим результат

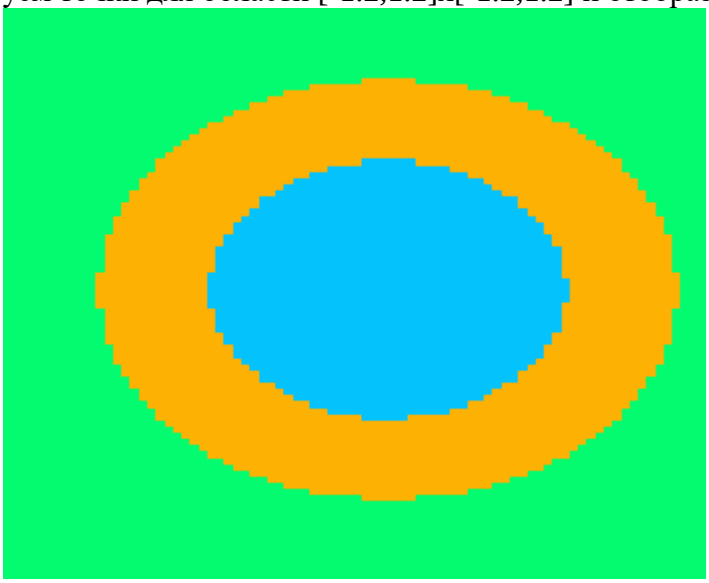

Рис. 12 Классификация точек в области [-1.2,1.2]X[-1.2,1.2]

2.6 Создадим сеть с радиальными базисными элементами (RBF) с параметром SPREAD = 0.1. Посчитаем долю верный ответов для обучающей и тестовых выборок.

```
# Точность на обучающем множестве
accuracy_score(svc.predict(x_train), y_train)
```

```
1.0
```

```
# Точность на тестовой выборке
accuracy_score(svc.predict(x_test), y_test)
```

```
1.0
```

Рис. 13 Доля верных ответов

2.7 Классифицируем точки для области [-1.2,1.2]x[-1.2,1.2] и отобразим результат

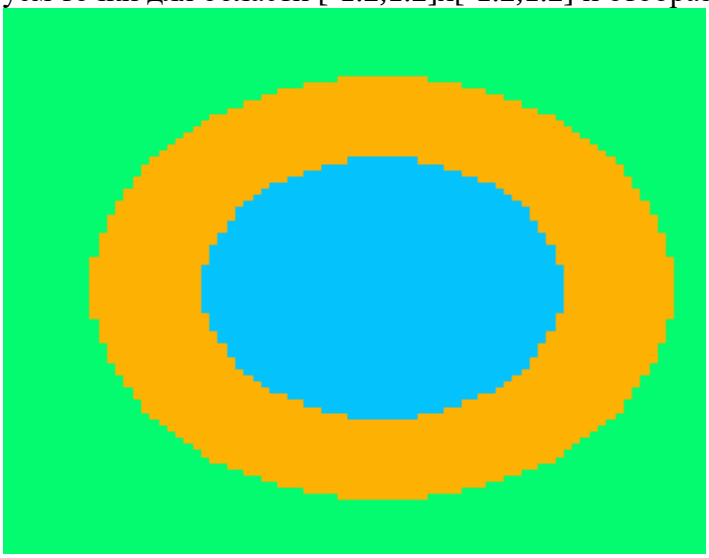

Рис. 14 Классификация точек в области [-1.2,1.2]X[-1.2,1.2]

Обе модели идеально классифицировали обучающее множества. Также обе модели верно классифицировали области

3. Использовать обобщенно-регрессионную нейронную сеть для аппроксимации функции. Проверить работу сети с рыхлыми данными.

3.1 Зададим область определения и значения функции. Отобразим график

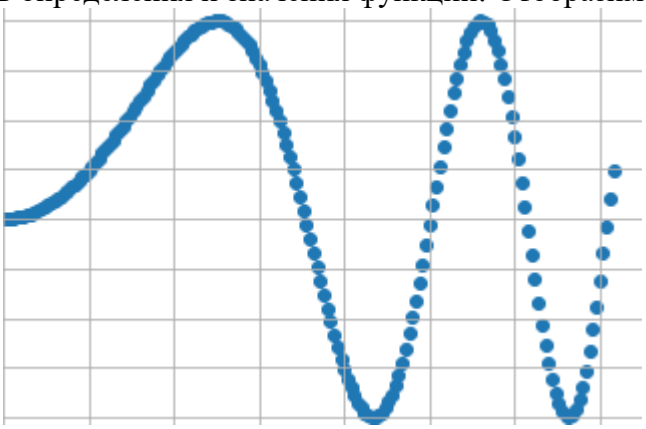

Рис. 15 Эталон

3.2 Разделим выборку на обучающую и тестовую выборку. Для теста возьмем 10% данных с конца, 90% с начала для обучающей выборки. Создадим обобщенно-регрессионную нейронную сеть. Обучим ее и отобразим структуру.

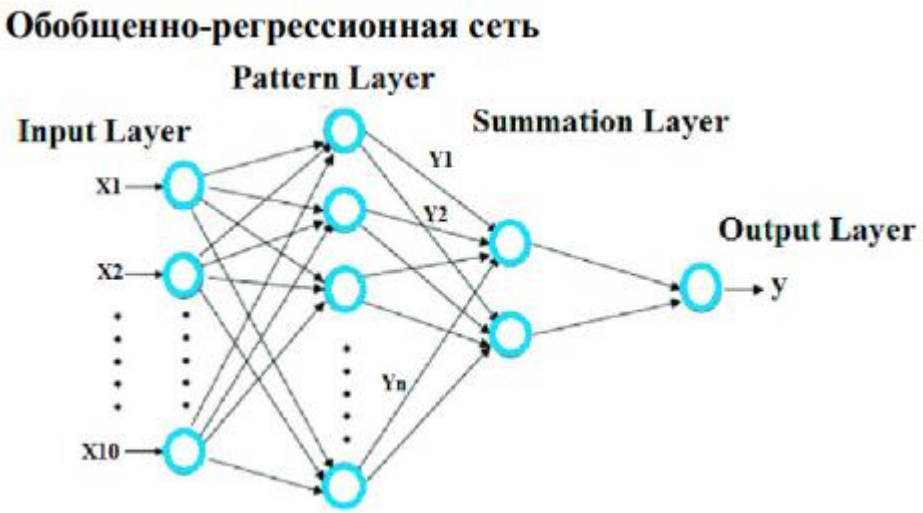

Рис. 16 Структура сети

3.3 Посчитаем MAE для обучающей выборки и тестовой выборок. Построим график предсказанный и истинный кривых.

```
# MAE на обучающей выборке
mean_squared_error(y_train, grnn.predict(x_train))
```

```
0.00264546652656753
```

```
# MAE на тестовой выборке
mean_squared_error(y_test, grnn.predict(x_test))
```

```
0.3811834261073895
```

Рис. 17 MAE на обучающей и тестовой выборке

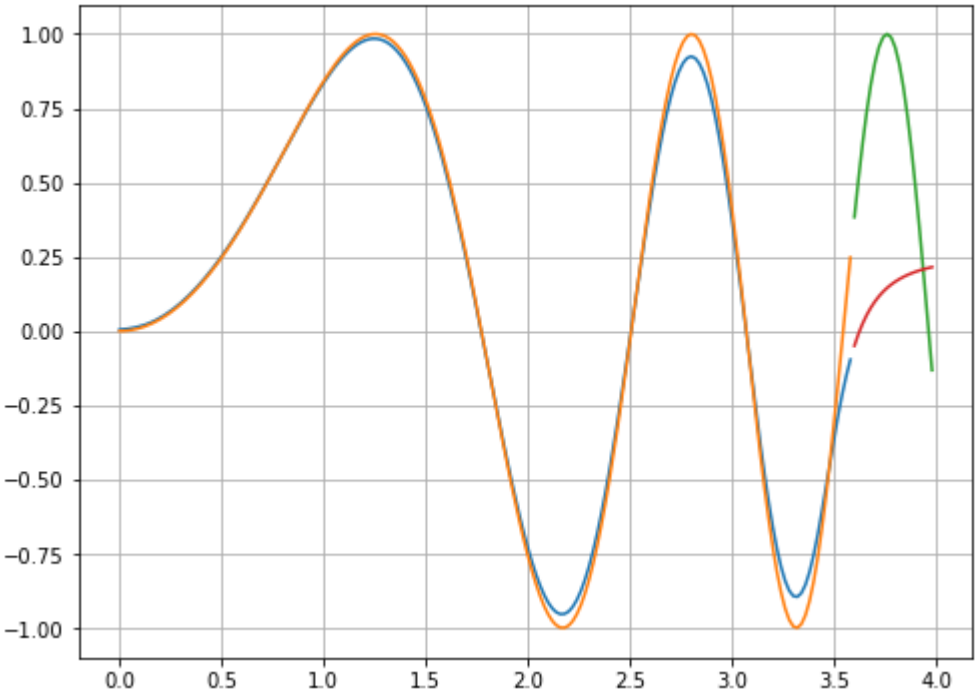

Рис. 18 График истинной и предсказанной кривых

Как видно из графика модель достаточно хорошо справилась с задачей. На тестовой выборке модель смогла предсказать общее направление, но не изгиб кривой.

Код программы

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from neupy.algorithms.rbfn.pnn import PNN
from neupy.algorithms.rbfn.grnn import GRNN
import warnings
warnings.filterwarnings("ignore")


# In[2]:


get ipython().system('python --version')


# In[3]:


# Создание элипса
def generate_points(a: float, b: float, x0: float, y0: float):
    t = np.arange(0, 2*np.pi, 0.025)
    points = np.zeros((t.shape[0], 2))
    points[:, 0] = a * np.cos(t) + x0
    points[:, 1] = b * np.sin(t) + y0
    return points


# In[4]:


# Классификация области
def classify_square3(model_func, a1=-1.2, b1=1.2, a2=-1.2, b2=1.2,
step=0.025):
    X_paint = np.array([[a1, a2]]).reshape((1, 2))
    for i in np.arange(a1, b1, step):
        for j in np.arange(a2, b2, step):
            X_paint = np.append(X_paint, np.array([[i, j]]), axis=0)
    Y_paint = model_func(X_paint)
    x1 = X_paint[(Y_paint == 0)]
    x2 = X_paint[(Y_paint == 1)]
    x3 = X_paint[(Y_paint == 2)]
    plt.figure(figsize=(8, 6))
    plt.scatter(x1.T[0], x1.T[1], marker='s', color='#03c2fc')
    plt.scatter(x2.T[0], x2.T[1], marker='s', color='#fcb103')
    plt.scatter(x3.T[0], x3.T[1], marker='s', color='#03fc6f')
    plt.title("Классификация заданной области на 3 класса")


# In[5]:


# Создание выборки
points1 = generate_points(0.3, 0.3, 0, 0)
points2 = generate_points(0.7, 0.7, 0, 0)
```

```
points3 = generate_points(1, 1, 0, 0)
plt.plot(points1[:, 0], points1[:, 1])
plt.plot(points2[:, 0], points2[:, 1])
plt.plot(points3[:, 0], points3[:, 1])
plt.grid(True)
plt.show()


# In[6]:


# Подготовка множеств
points1 = points1[np.random.choice(points1.shape[0], 60),:]
points2 = points2[np.random.choice(points2.shape[0], 100),:]
points3 = points3[np.random.choice(points3.shape[0], 120),:]

plt.scatter(points1[:, 0], points1[:, 1])
plt.scatter(points2[:, 0], points2[:, 1])
plt.scatter(points3[:, 0], points3[:, 1])
plt.grid(True)
plt.show()


# In[7]:


features = np.concatenate((points1, points2, points3))
labels = np.concatenate((np.full(60, 0), np.full(100, 1), np.full(120, 2)))


# In[8]:


x_train, x_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, random_state=23)
print(x_train.shape[0] / 280, x_test.shape[0] / 280)


# In[9]:


# Обучение с параметром std=0.1
pnn = PNN(std=0.1)
pnn.fit(x_train, y_train)


# In[10]:


# Точность на обучающей выборке
accuracy_score(pnn.predict(x_train), y_train)


# In[11]:


# Точность на тестовой выборке
accuracy_score(pnn.predict(x_test), y_test)


# In[12]:


# Классификация областей
```

```
classify_square3(pnn.predict)


# In[13]:


# Обучение с параметром std=0.3
pnn = PNN(std=0.3)
pnn.fit(x_train, y_train)


# In[14]:


# Точностьность на обучающей выборке
accuracy_score(pnn.predict(x_train), y_train)


# In[15]:


# Точность на тестовой выборке
accuracy_score(pnn.predict(x_test), y_test)


# In[16]:


# Классификация области
classify_square3(pnn.predict)


# ###### Обе модели идеально классифицировали обучающее множества.
# ###### Но модель с параметром std=0.1 более ровные границы множеств чем
модель с параметром std=0.3

# In[17]:


# Модель с параметром гамма = 0.3
svc = SVC(kernel='rbf', C=1e2, gamma=0.3)
svc.fit(x_train, y_train)


# In[18]:


# Точность на обучающей выборке
accuracy_score(svc.predict(x_train), y_train)


# In[19]:


# Точность на тестовой выборке
accuracy score(svc.predict(x test), y test)


# In[20]:


# Классификация области
classify_square3(svc.predict)
```

```
# In[21]:


# Модель с параметром гамма = 0.1
svc = SVC(kernel='rbf', C=1e2, gamma=0.1)
svc.fit(x_train, y_train)


# In[22]:


# Точность на обучающем множестве
accuracy_score(svc.predict(x_train), y_train)


# In[23]:


# Точность на тестовой выборке
accuracy_score(svc.predict(x_test), y_test)


# In[24]:


# Классификация области
classify_square3(svc.predict)


# ###### Обе модели идеально классифицировали обучающее множества.
# ###### Также обе модели верно классифицировали области.

# In[25]:


# Выборка
xt = lambda t: np.sin(t*t)
features = np.arange(0, 4, 0.02)
print(features.shape[0])
targets = xt(features)
n = int(0.9 * features.shape[0])
x_train, y_train = features[:n], targets[:n]
x_test, y_test = features[n:], targets[n:]
plt.scatter(x_train, y_train)
plt.grid(True)
plt.show()


# In[26]:


#Создание модели
grnn = GRNN(std=0.1)
grnn.fit(x_train, y_train)


# In[27]:


# MAE на обучающей выборке
mean_squared_error(y_train, grnn.predict(x_train))
```

```
# In[28]:


# MAE на тестовой выборке
mean_squared_error(y_test, grnn.predict(x_test))


# In[29]:


# Предсказание на тесте и на трейне
train_predictions = grnn.predict(x_train)
test_predictions = grnn.predict(x_test)


# In[30]:


# Рисунок истинных и предсказанных значений на тестовой и обучающей выборке
plt.figure(figsize=(8, 6))
plt.plot(x_train, train_predictions)
plt.plot(x_train, y_train)
plt.plot(x_test, y_test)
plt.plot(x_test, test_predictions)
plt.grid(True)
plt.show()


# In[ ]:




# In[ ]:
```

Выводы

В лабораторной работе было проведено исследование свойств некоторых видов сетей с радиальными базисными элементами, а также применение сетей в задачах классификации и аппроксимации функции.