Лабораторная работа 4

Вариант 13

Аминов С.С. М8О-408Б-19

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

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras import layers
import tensorflow as tf
```

Классификация

In [2]:
```
def ellipse(t, a, b, x0, y0):
    x = x0 + a*np.cos(t)
    y = x0 + b*np.sin(t)
    return x, y

def parabola(t, p, x0, y0):
    x = x0 + t ** 2 / (2. * p)
    y = y0 + t
    return x, y

epochs = 500
```

In [3]:
```
t = np.linspace(0, 2*np.pi, 200)
x1, y1 = ellipse(t, 0.3, 0.3, 0, 0)

x2, y2 = ellipse(t, 0.7, 0.7, 0, 0)

x3, y3 = parabola(t, 1, -0.8, 0)
```

Описываемм класс RBF слоя

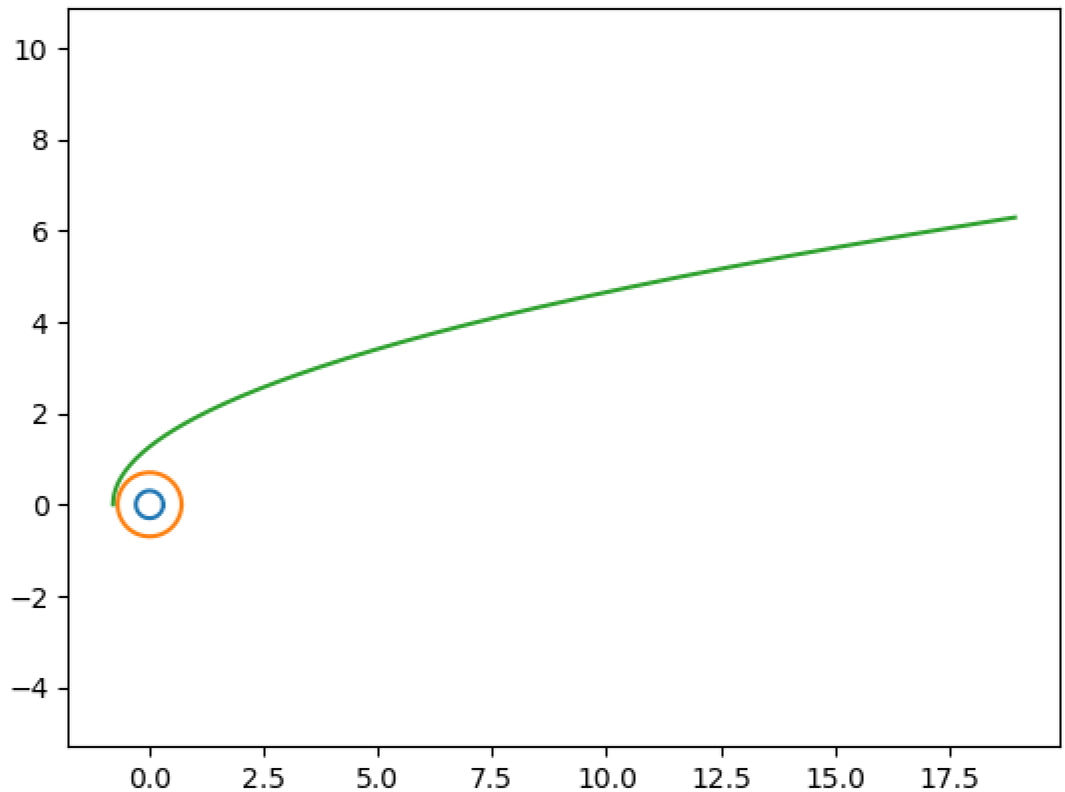
In [4]:
```
class RBF(layers.Layer):
    def __init__(self, units, **kwargs):
        self.units = units
        super(RBF, self).__init__(**kwargs)

    def build(self, input_shape):
        self.mu = self.add_weight(
            shape=(input_shape[1], self.units), initializer="random_normal", trainable=T
        )
        self.sigm = self.add_weight(
            shape=(self.units,) , initializer="random_normal", trainable=True
        )
        super().build(input_shape)

    def call(self, inputs):
        l_norm = tf.reduce_sum((tf.expand_dims(inputs, axis=2) - self.mu) ** 2, axis = 1
        return tf.exp(l_norm*self.sigm)
```

In [5]:
```
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.axis('equal')
```

Out[5]:
```
(-1.7869604401089358, 19.92616924228765, -1.049136367826184, 6.632343482179861)
```



Готовим датасет

In [6]:
```
data1 = [[cords, [1, 0, 0]] for cords in zip(x1, y1)]
data2 = [[cords, [0, 1, 0]] for cords in zip(x2, y2)]
data3 = [[cords, [0, 0, 1]] for cords in zip(x3, y3)]
dataset = data1 + data2 + data3
np.random.shuffle(dataset)
```

In [7]:
```
train_percent = 0.8
train_num = int(train_percent * len(dataset))
train_X = [x[0] for x in dataset[:train_num]]
train_y = [x[1] for x in dataset[:train_num]]
test_X = [x[0] for x in dataset[train_num:]]
test_y = [x[1] for x in dataset[train_num:]]
```

Создаем модель

In [8]:
```
model =  keras.Sequential([
    RBF(10,input_dim=2),
    layers.Dense(3,activation='sigmoid', name="sigmoid")
    ])
```

Компилируем модель

In [9]:
```
model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

Тренеруем модель

```
hist = model.fit(train_X,train_y,batch_size=len(dataset)//10,epochs=epochs)
```

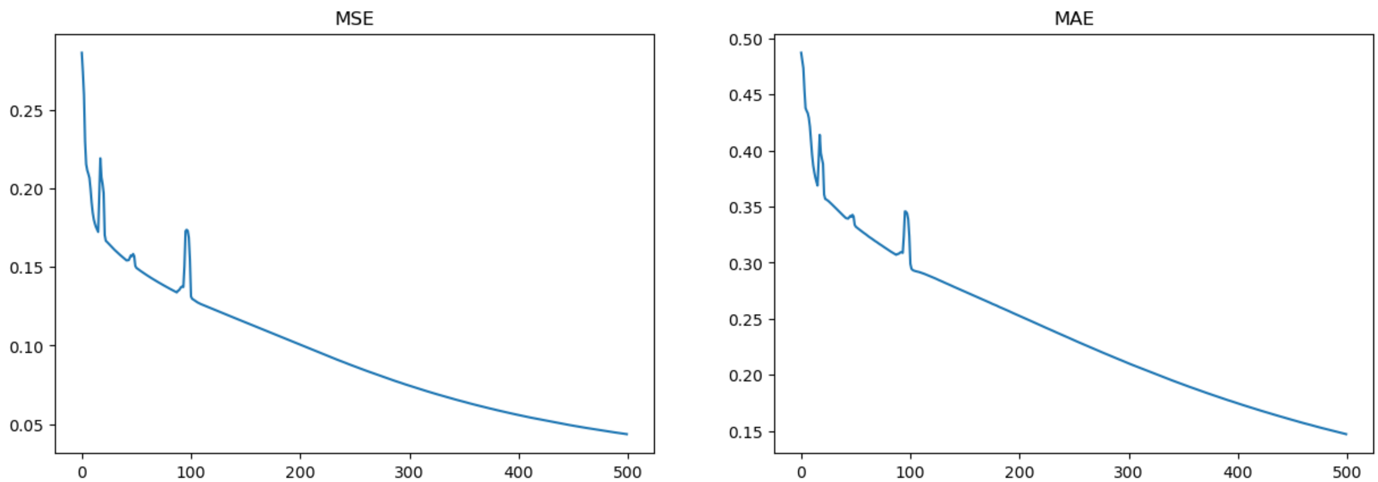
In [ ]:

In [11]:
```
fig, ax = plt.subplots(1, 2)
fig.set_figwidth(15)

ax[0].set_title('MSE')
ax[1].set_title('MAE')

ax[0].plot(range(epochs), hist.history['loss'])
ax[1].plot(range(epochs), hist.history['mae'])
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x226f6b93f70>]
```



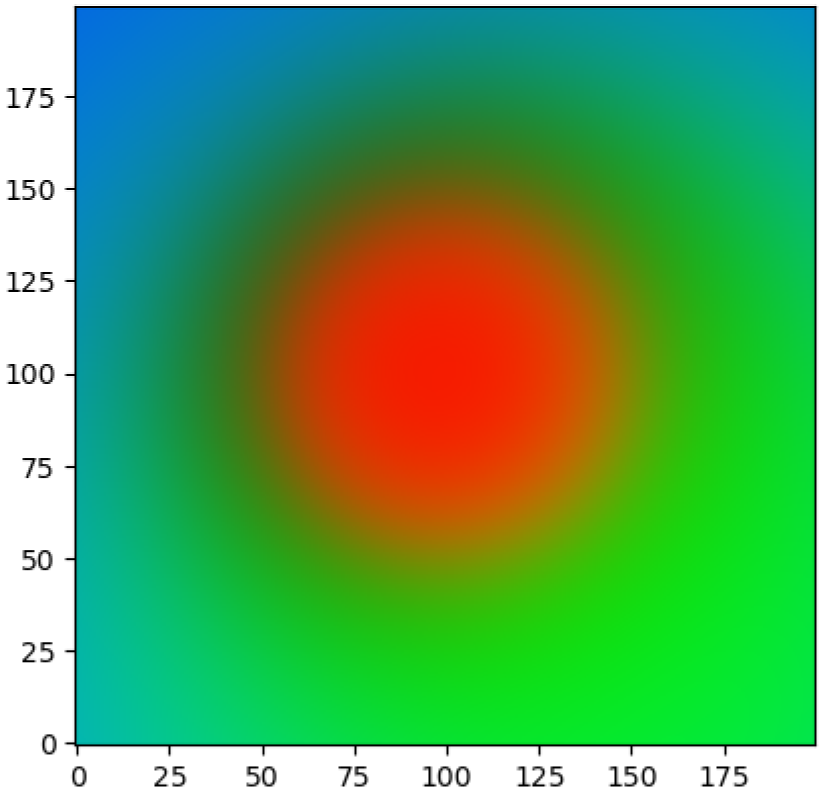
Создаем поле точек и скалярное поле

In [12]:
```
pole = []
for y in np.linspace(-1,1,200):
    for x in np.linspace(-1,1,200):
        pole.append((x,y))
```

In [13]:
```
pred = model.predict(pole)
z = []
for i in range(200):
    z.append(pred[i*200: (i+1)*200])
```

```
1250/1250 [==============================] - 1s 711us/step
```

In [14]:
```
fig, ax = plt.subplots()
ax.imshow(z)
ax.invert_yaxis()
```



Аппроксимация функции

In [15]:
```
def func(t):
    return np.cos(t**2 - 2*t + 3)
```

In [16]:
```
h = 0.02
X = np.arange(0, 5+h,h)
y = func(X)
```

Создаем модель

In [38]:
```
model2 =  keras.Sequential([
        RBF(30,input_dim=1),
        layers.Dense(1,activation='linear', name='linear')
    ]
)
```

Компилируем модель

In [39]:
```
model2.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

Тренеруем модель

In [ ]:
```
hist2 = model2.fit(X,y,batch_size=10,epochs=1000)
```

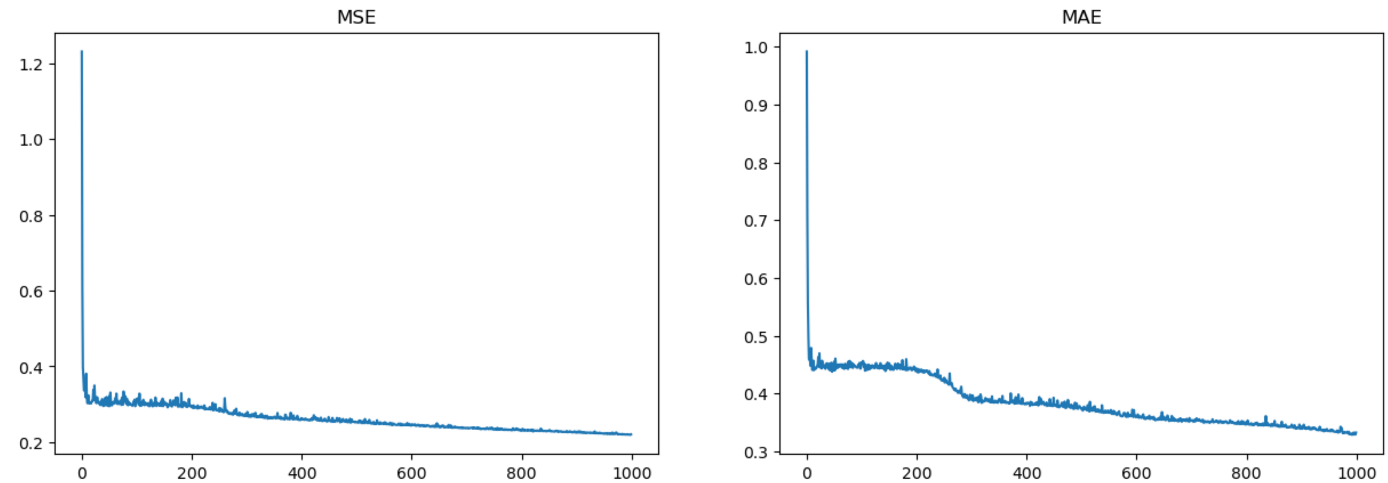
In [41]:
```
fig, ax = plt.subplots(1, 2)
fig.set_figwidth(15)

ax[0].set_title('MSE')
ax[1].set_title('MAE')

ax[0].plot(range(1000), hist2.history['loss'])
ax[1].plot(range(1000), hist2.history['mae'])
```

```
[<matplotlib.lines.Line2D at 0x2268307f5b0>]
```

Out[41]:

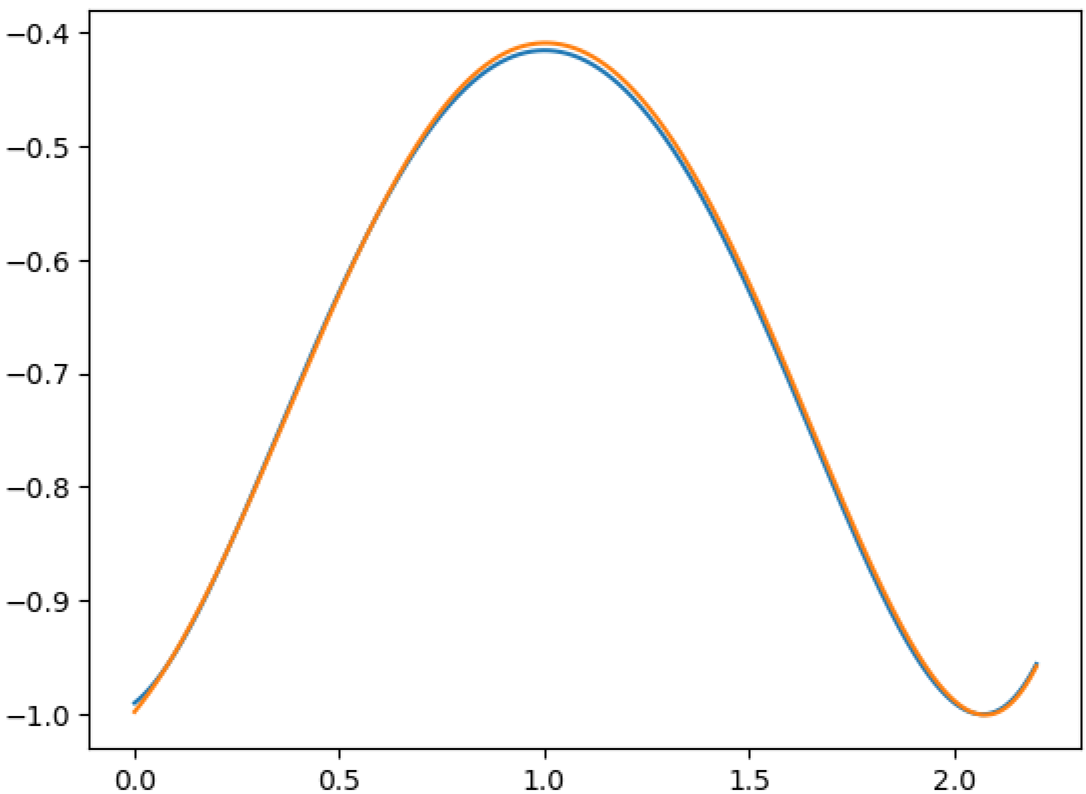


Аппроксимируем функцию

In [33]:
```
t = np.linspace(0, 2.2, 2000)
y_ans = func(t)
y_pred = model2.predict(t)
plt.plot(t, y_ans)
plt.plot(t, y_pred)
```

```
63/63 [==============================] - 0s 790us/step
```

Out[33]:
```
[<matplotlib.lines.Line2D at 0x226fbf0a3a0>]
```



In [ ]: