Лабораторная работа 3

Вариант 13

Аминов С.С. М8О-408Б-19

Целью работы является исследование свойств многослойной нейронной сети прямого распространения и алгоритмов ее обучения, применение сети в задачах классификации и аппроксимации функции.

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras import layers
import tensorflow as tf
```

Классификация

In [29]:
```
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
```

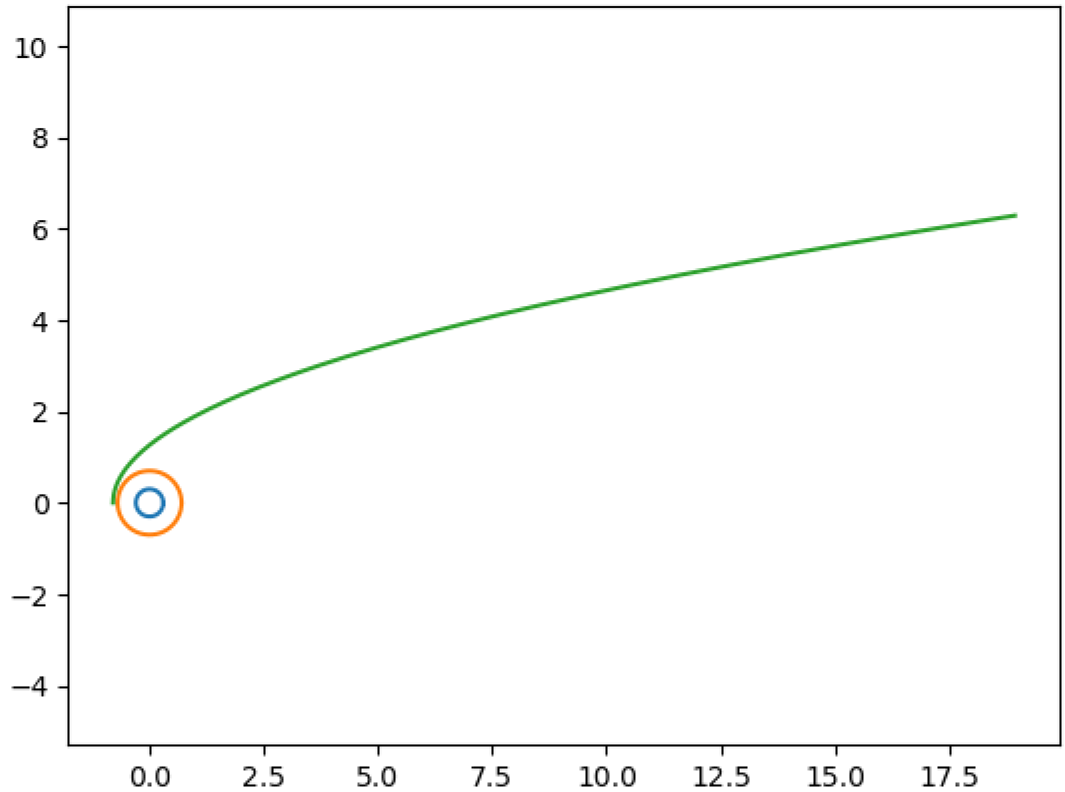
In [30]:
```
t = np.linspace(0, 2*np.pi, 200)
x1, y1 = ellipse(t, 0.3, 0.3, 0, 0)

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

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

In [31]:
```
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.axis('equal')
```

Out[31]:
```
(-1.7869604401089358, 19.92616924228765, -1.049136367826184, 6.632343482179861)
```



Готовим датасет

In [32]:
```
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 [33]:
```
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 [34]:
```
model =  keras.Sequential([
        layers.Dense(100,input_dim=2, activation="tanh", name="tanh"),
        layers.Dense(3,activation='sigmoid', name="sigmoid")
    ]
)
```

Компилируем

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

Тренеруем

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

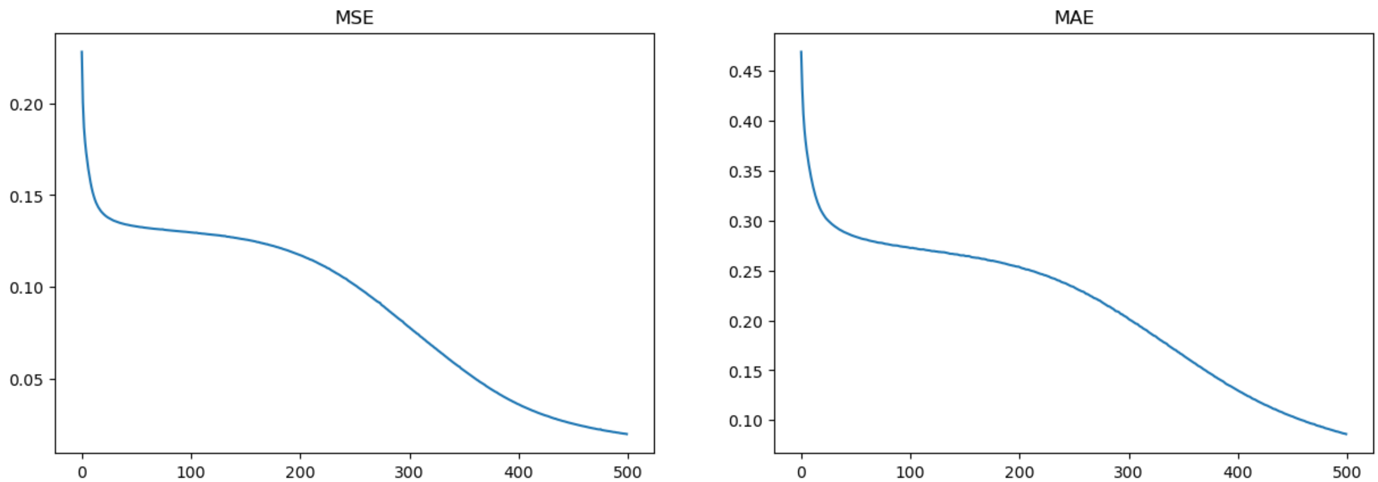
In [13]:
```
fig, ax = plt.subplots(1, 2)
fig.set_figwidth(15)

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[13]:

```
[<matplotlib.lines.Line2D at 0x1a26baa02b0>]
```



Создаем поле точек и скалярное поле

In [14]:

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

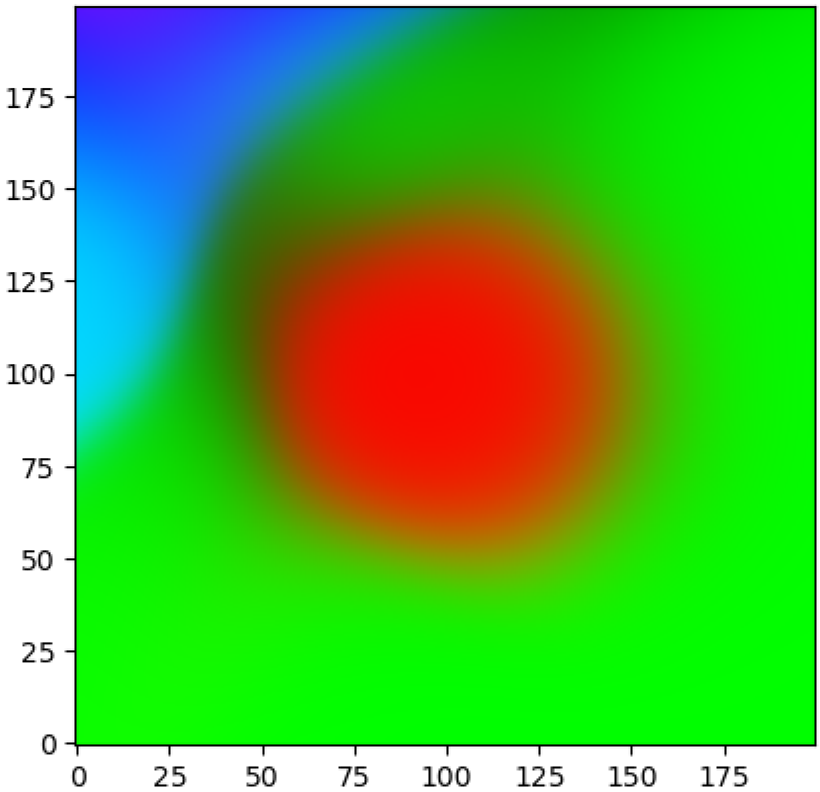
In [15]:

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

```
1250/1250 [==============================] - 1s 855us/step
```

In [16]:

```
fig, ax = plt.subplots()
ax.imshow(z)
ax.invert_yaxis()
```



Аппроксимация функции

In [17]:
```
def func(t):
    return np.cos(t**2 - 2*t + 3)
```

In [18]:
```
h = 0.02
X = np.arange(0, 5+h,h)
y = func(X)
```

Создаем модель

In [19]:
```
model2 =  keras.Sequential([
        layers.Dense(100,input_dim=1, activation="tanh", name="tanh"),
        layers.Dense(30, activation="tanh", name="tanh2"),
        layers.Dense(1,activation='linear', name='linear')
    ]
)
```

Компилируем

In [20]:
```
model2.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

Тренеруем

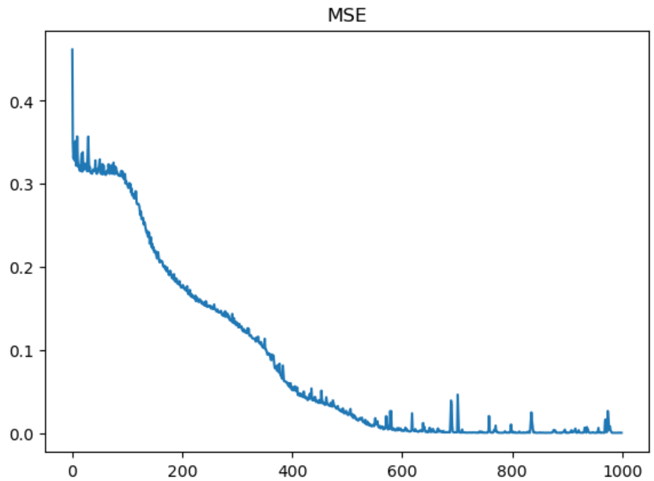
In [ ]:
```
hist2 = model2.fit(X,y,batch_size=10,epochs=1000)
```

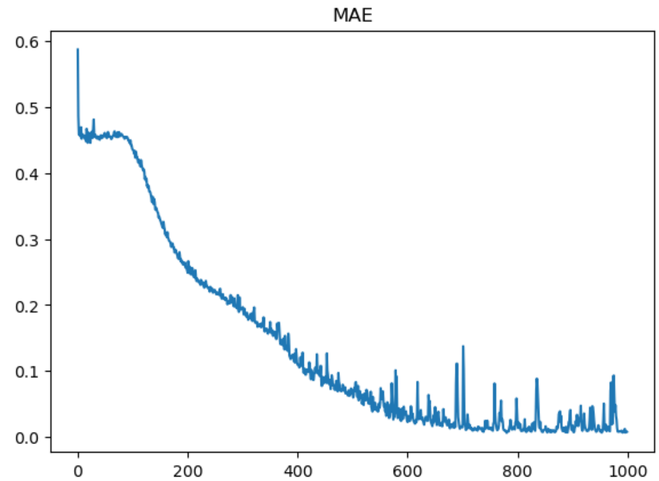
In [25]:
```
fig, ax = plt.subplots(1, 2)
fig.set_figwidth(15)

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'])
```

Out[25]: [<matplotlib.lines.Line2D at 0x1a26aa31700>]





Аппроксимируем функцию

In [27]:
```
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 758us/step
```

Out[27]: [<matplotlib.lines.Line2D at 0x1a267b3b940>]