Лабораторная работа № **3** по Нейроинформатике

Многослойные сети. Алгоритм обратного распространения ошибки

Выполнила: Тимофеева Наталья М**8**О-**408**Б-**19**

Вариант № **16**

Часть **1**

Классификация

In [241]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.python.keras import layers
from matplotlib import pyplot as plt
import itertools
from timeit import default_timer as timer
```

In [242]:

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

In [243]:

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

In [244]:

```
def rotate(x, y, alpha):
    xr = x * np.cos(alpha) - y * np.sin(alpha)
    yr = x * np.sin(alpha) - y * np.cos(alpha)
    return xr, yr
```

In [245]:

```
t = np.arange(0, 2 * np.pi, 0.001)
```

Создаём эллипсы и параболу

In [246]:

```
x1, y1 = ellipse(t, 0.4, 0.15, 0.1, -0.15)
x1, y1 = rotate(x1, y1, np.pi / 6)

x2, y2 = ellipse(t, 0.7, 0.5, 0, 0)
x2, y2 = rotate(x2, y2, np.pi / 3)

x3, y3 = parabola(t, 1, 0, -0.8)
x3, y3 = rotate(x3, y3, np.pi / 2)
```

Преоразуем данные в удобный формат

In [247]:

```
d1 = [[[x, y], [1., 0., 0.]] for x, y in zip(x1, y1)]
```

```
d2 = [[[x, y], [0., 1., 0.]] for x, y in zip(x2, y2)]
d3 = [[[x, y], [0., 0., 1.]] for x, y in zip(x3, y3)]
```

Объединяем и перемешиваем

In [248]:

```
dataset = d1 + d2 + d3
np.random.shuffle(dataset)
```

Разбиваем на обучающую и тестовую выборки

In [249]:

```
separ = int(len(dataset) * 0.8)
train_data = dataset[:separ]
test_data = dataset[separ:]

train_input = [i[0] for i in train_data]
train_output = [i[1] for i in train_data]
```

Создание модели

In [250]:

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(100, input_dim = 2, activation='tanh'))
model.add(keras.layers.Dense(3, activation='sigmoid'))
model.compile(loss = 'mse', optimizer = 'adam', metrics = ['mae'])
```

Обучение модели

In [251]:

```
len(train_data)
```

Out[251]:

```
15081
```

In [252]:

```
epochs = 2500
batch_size = int(len(train_data) * 0.004)

time_start = timer()
hist = model.fit(train_input, train_output, batch_size = batch_size, epochs = epochs)
time_end = timer()
```

```
Epoch 1/2500
252/252 [==============================] - 0s 777us/step - loss: 0.1519 - mae: 0.3314
Epoch 2/2500
252/252 [==============================] - 0s 725us/step - loss: 0.1334 - mae: 0.2753
Epoch 3/2500
252/252 [==============================] - 0s 725us/step - loss: 0.1319 - mae: 0.2657
Epoch 4/2500
252/252 [==============================] - 0s 733us/step - loss: 0.1311 - mae: 0.2611
Epoch 5/2500
252/252 [==============================] - 0s 746us/step - loss: 0.1307 - mae: 0.2585
Epoch 6/2500
252/252 [==============================] - 0s 732us/step - loss: 0.1303 - mae: 0.2561
Epoch 7/2500
252/252 [==============================] - 0s 734us/step - loss: 0.1297 - mae: 0.2543
Epoch 8/2500
252/252 [==============================] - 0s 775us/step - loss: 0.1289 - mae: 0.2526
Epoch 9/2500
252/252 [==============================] - 0s 776us/step - loss: 0.1273 - mae: 0.2500
Epoch 10/2500
252/252 [==============================] - 0s 745us/step - loss: 0.1239 - mae: 0.2464
```

```
Epoch 2495/2500
252/252 [==============================] - 0s 1ms/step - loss: 0.0228 - mae: 0.0312
Epoch 2496/2500
252/252 [==============================] - 0s 947us/step - loss: 0.0228 - mae: 0.0311
Epoch 2497/2500
252/252 [==============================] - 0s 884us/step - loss: 0.0227 - mae: 0.0311
Epoch 2498/2500
252/252 [==============================] - 0s 820us/step - loss: 0.0227 - mae: 0.0312
Epoch 2499/2500
252/252 [==============================] - 0s 866us/step - loss: 0.0227 - mae: 0.0312
Epoch 2500/2500
252/252 [==============================] - 0s 819us/step - loss: 0.0228 - mae: 0.0314
```

In [253]:

```
print('Эпох: {0}'.format(epochs))
print('Время обучения: {0} секунд'.format(int(time_end - time_start)))
print('Функция потерь MSE: {0}'.format(min(hist.history['loss'])))
print('Метрика качества MAE: {0}'.format(min(hist.history['mae'])))
```

```
Эпох: 2500
Время обучения: 558 секунд
Функция потерь MSE: 0.022683124989271164
Метрика качества MAE: 0.031024910509586334
```

In [254]:

```
x = np.linspace(-6, 1, 200)
y = np.linspace(-3, 20, 200)
```

In [255]:

```
line = np.array(list(itertools.product(x, y)))
xs, ys = np.hsplit(line, 2)
```

In [256]:

```
predicted = model.predict(line)
```

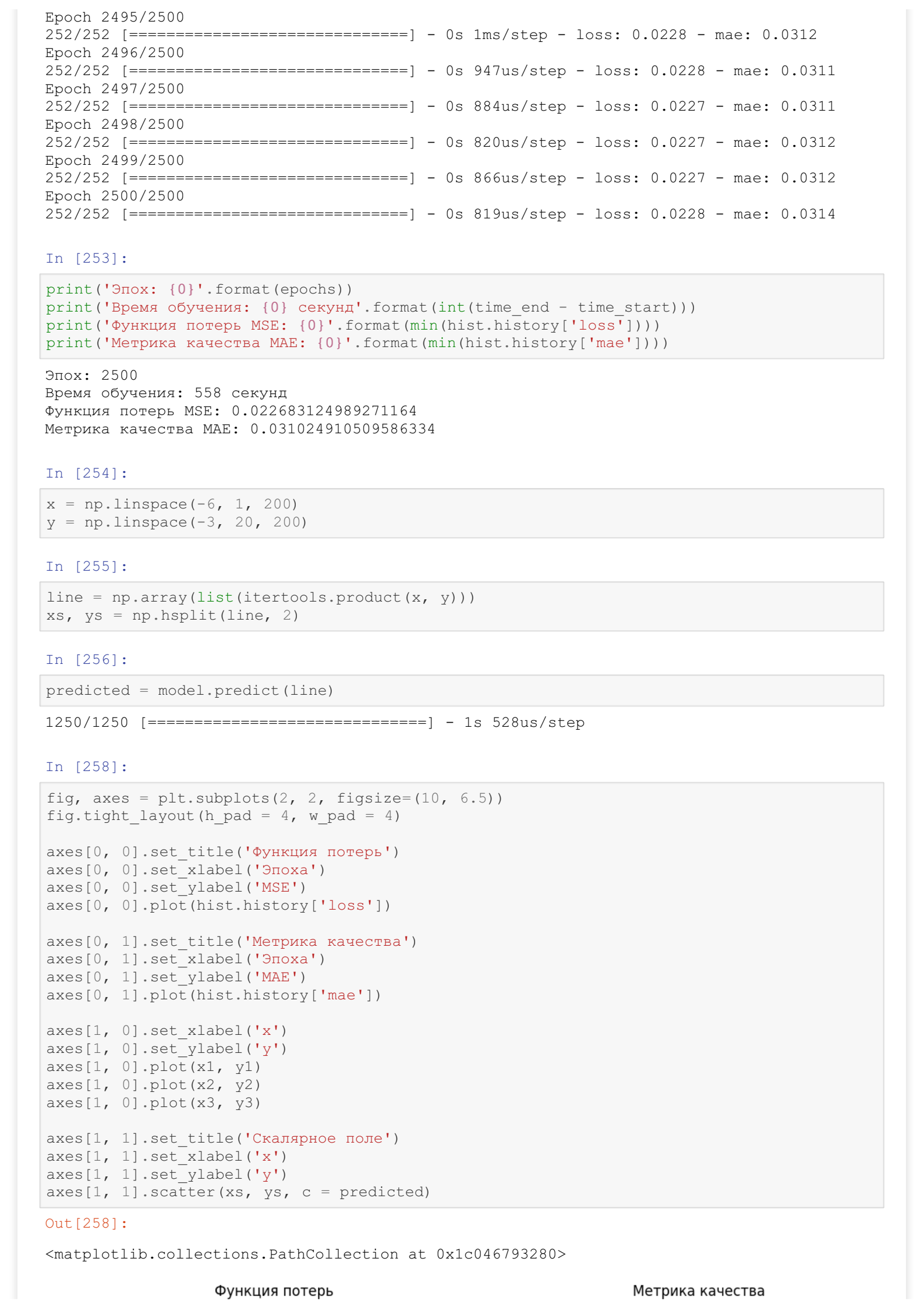
```
1250/1250 [==============================] - 1s 528us/step
```

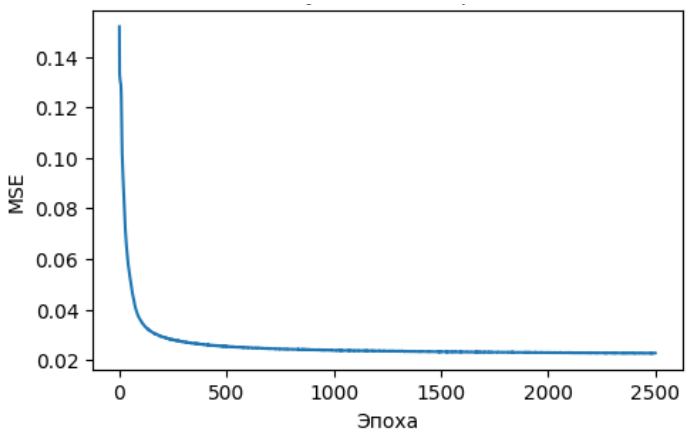
In [258]:

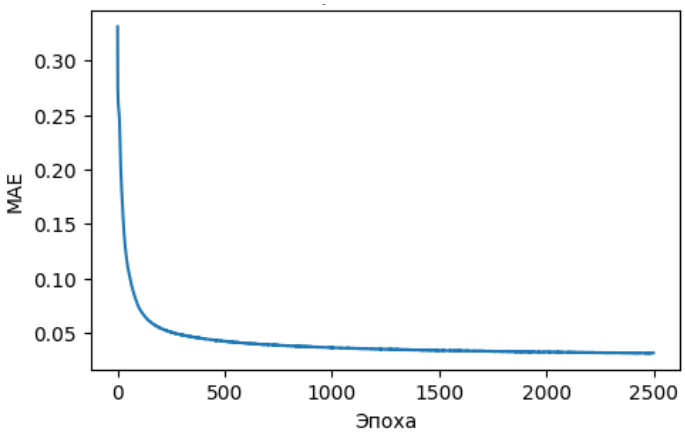
```
fig, axes = plt.subplots(2, 2, figsize=(10, 6.5))
fig.tight_layout(h_pad = 4, w_pad = 4)

axes[0, 0].set_title('Функция потерь')
axes[0, 0].set_xlabel('Эпоха')
axes[0, 0].set_ylabel('MSE')
axes[0, 0].plot(hist.history['loss'])

axes[0, 1].set_title('Метрика качества')
axes[0, 1].set_xlabel('Эпоха')
axes[0, 1].set_ylabel('MAE')
axes[0, 1].plot(hist.history['mae'])

axes[1, 0].set_xlabel('x')
axes[1, 0].set_ylabel('y')
axes[1, 0].plot(x1, y1)
axes[1, 0].plot(x2, y2)
axes[1, 0].plot(x3, y3)

axes[1, 1].set_title('Скалярное поле')
axes[1, 1].set_xlabel('x')
axes[1, 1].set_ylabel('y')
axes[1, 1].scatter(xs, ys, c = predicted)
```

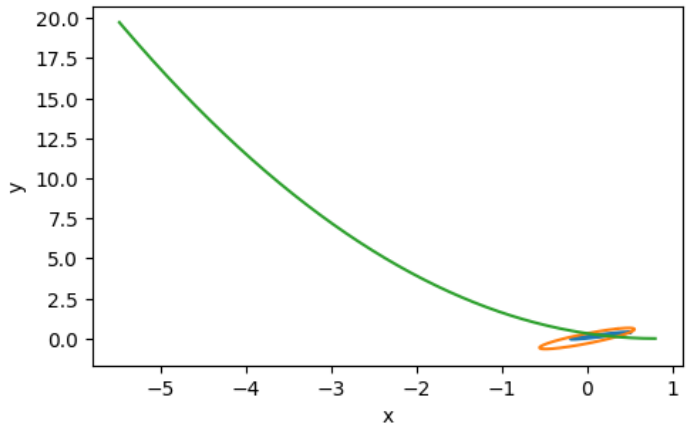
Out[258]:

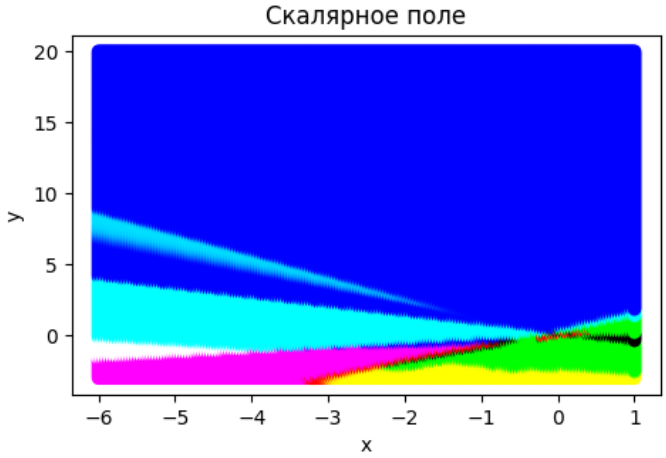
```
<matplotlib.collections.PathCollection at 0x1c046793280>
```




MSE
Эпоха


MAE
Эпоха


y
x


Скалярное поле
y
x

Лабораторная работа № **3** по Нейроинформатике

Многослойные сети. Алгоритм обратного распространения ошибки

Выполнила: Тимофеева Наталья М**8**О-**408**Б-**19**

Вариант № **16**

Часть **2**

Аппроксимация

In [17]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.python.keras import layers
from matplotlib import pyplot as plt
from timeit import default_timer as timer
```

In [18]:

```
def fun(t):
    return np.sin(np.sin(t) * t ** 2 + 3 * t - 10)
```

Создание модели

In [19]:

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(40, input_dim = 1, activation='tanh'))
model.add(keras.layers.Dense(12, activation='tanh'))
model.add(keras.layers.Dense(1, activation='linear'))
model.compile(loss = 'mse', optimizer = 'adam', metrics = ['mae'])
```

In [20]:

```
t1 = np.linspace(0, 3.5, 100)
y1 = fun(t1)
```

Обучение модели

In [21]:

```
epochs = 2500
batch_size = 7

time_start = timer()
hist = model.fit(t1, y1, batch_size = batch_size, epochs = epochs)
time_end = timer()
```

```
Epoch 1/2500
15/15 [==============================] - 0s 1ms/step - loss: 0.5115 - mae: 0.6307
Epoch 2/2500
15/15 [==============================] - 0s 977us/step - loss: 0.4937 - mae: 0.6257
Epoch 3/2500
15/15 [==============================] - 0s 861us/step - loss: 0.4764 - mae: 0.6183
Epoch 4/2500
15/15 [==============================] - 0s 894us/step - loss: 0.4803 - mae: 0.6156
Epoch 5/2500
15/15 [==============================] - 0s 784us/step - loss: 0.4668 - mae: 0.6110
Epoch 6/2500
15/15 [==============================] - 0s 998us/step - loss: 0.4746 - mae: 0.6155
Epoch 7/2500
15/15 [==============================] - 0s 1ms/step - loss: 0.4669 - mae: 0.6102
```

```
Epoch 2492/2500
15/15 [==============================] - 0s 2ms/step - loss: 4.8722e-04 - mae: 0.0145
Epoch 2493/2500
15/15 [==============================] - 0s 2ms/step - loss: 5.9317e-04 - mae: 0.0160
Epoch 2494/2500
15/15 [==============================] - 0s 2ms/step - loss: 0.0027 - mae: 0.0277
Epoch 2495/2500
15/15 [==============================] - 0s 2ms/step - loss: 0.0035 - mae: 0.0351
Epoch 2496/2500
15/15 [==============================] - 0s 2ms/step - loss: 0.0023 - mae: 0.0280
Epoch 2497/2500
15/15 [==============================] - 0s 4ms/step - loss: 7.5137e-04 - mae: 0.0169
Epoch 2498/2500
15/15 [==============================] - 0s 4ms/step - loss: 2.9456e-04 - mae: 0.0131
Epoch 2499/2500
15/15 [==============================] - 0s 4ms/step - loss: 3.3803e-04 - mae: 0.0128
Epoch 2500/2500
15/15 [==============================] - 0s 4ms/step - loss: 2.4447e-04 - mae: 0.0118
```

In [22]:

```
print('Эпох: {0}'.format(epochs))
print('Время обучения: {0} секунд'.format(int(time_end - time_start)))
print('Функция потерь MSE: {0}'.format(min(hist.history['loss'])))
print('Метрика качества MAE: {0}'.format(min(hist.history['mae'])))
```

```
Эпох: 2500
Время обучения: 54 секунд
Функция потерь MSE: 0.00021155335707589984
Метрика качества MAE: 0.010877574793994427
```

Выводим с помощью обученной модели точки с меньшим шагом

In [23]:

```
t2 = np.linspace(0, 3.5, 2000)
y2 = model.predict(t2)
real_y = fun(t2)
```

```
63/63 [==============================] - 0s 724us/step
```

In [24]:

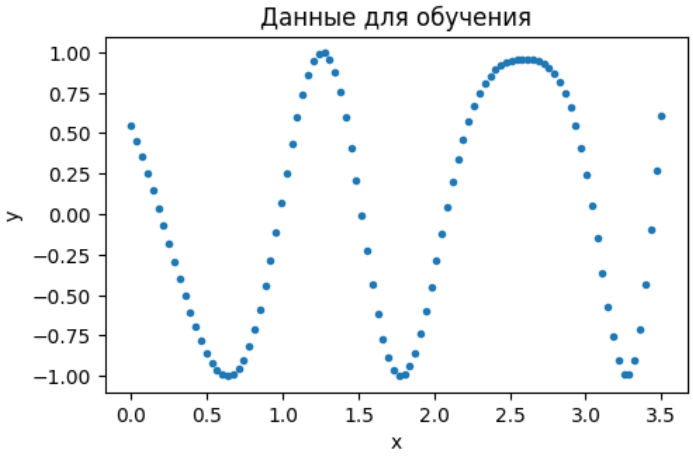
```
fig, axes = plt.subplots(2, 2, figsize=(10, 6.5))
fig.tight_layout(h_pad = 4, w_pad = 4)

axes[0, 0].set_title('Функция потерь')
axes[0, 0].set_xlabel('Эпоха')
axes[0, 0].set_ylabel('MSE')
axes[0, 0].plot(hist.history['loss'])

axes[0, 1].set_title('Метрика качества')
axes[0, 1].set_xlabel('Эпоха')
axes[0, 1].set_ylabel('MAE')
axes[0, 1].plot(hist.history['mae'])

axes[1, 0].set_title('Данные для обучения')
axes[1, 0].set_xlabel('x')
axes[1, 0].set_ylabel('y')
axes[1, 0].plot(t1, y1, '.')

axes[1, 1].set_title('Истинные и предсказанные значения')
axes[1, 1].set_xlabel('x')
axes[1, 1].set_ylabel('y')
axes[1, 1].plot(t2, real_y)
axes[1, 1].plot(t2, y2, '--')
```

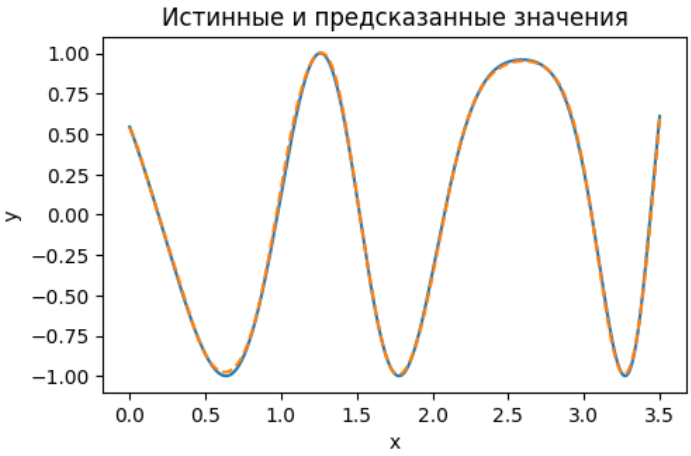
Out[24]:

```
[<matplotlib.lines.Line2D at 0x1ef85ac10f0>]
```


Функция потерь
MSE
Эпоха


Метрика качества
MAE
Эпоха


Данные для обучения
y
x


Истинные и предсказанные значения
y
x