

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО КОНТРОЛЬНОЙ РАБОТЕ № 3

Дисциплина: Методы машинного обучения

Студент: Маслова Анастасия

Группа: НКНбд-01-21

Москва 2024

Вариант №12

1. Функция одной переменной $f(x) = e^{-x}/(1+x)$ на отрезке $[0, 4]$
2. Порядок производной функции одной переменной 4
3. Функция двух переменных $f(x,y) = x^3 \sin(y)$ в области $[0, 2] \times [0, 2]$
4. Порядок смешанной производной функции двух переменных $d^3/dx dy^2$
5. Показатель качества регрессии: максимальная ошибка (MaxErr)

1. Постройте тензор ранга 1 (вектор) со значениями заданной в индивидуальном задании функции одной переменной на заданном в индивидуальном задании отрезке и определите максимальное и минимальное значения функции.

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

```
In [2]: def func(x):
return (math.e**(-x))/(1+x)
```

```
In [3]: ▶ segment = tf.Variable(np.linspace(0, 4, 50), dtype=tf.float32)
tensor = tf.constant(func(segment), dtype=tf.float32)
max_value = tf.reduce_max(tensor)
min_value = tf.reduce_min(tensor)
print("Максимальное значение функции: ", max_value.numpy())
print("\nМинимальное значение функции: ", min_value.numpy())
print(tensor)
```

Максимальное значение функции: 1.0

Минимальное значение функции: 0.0036631282

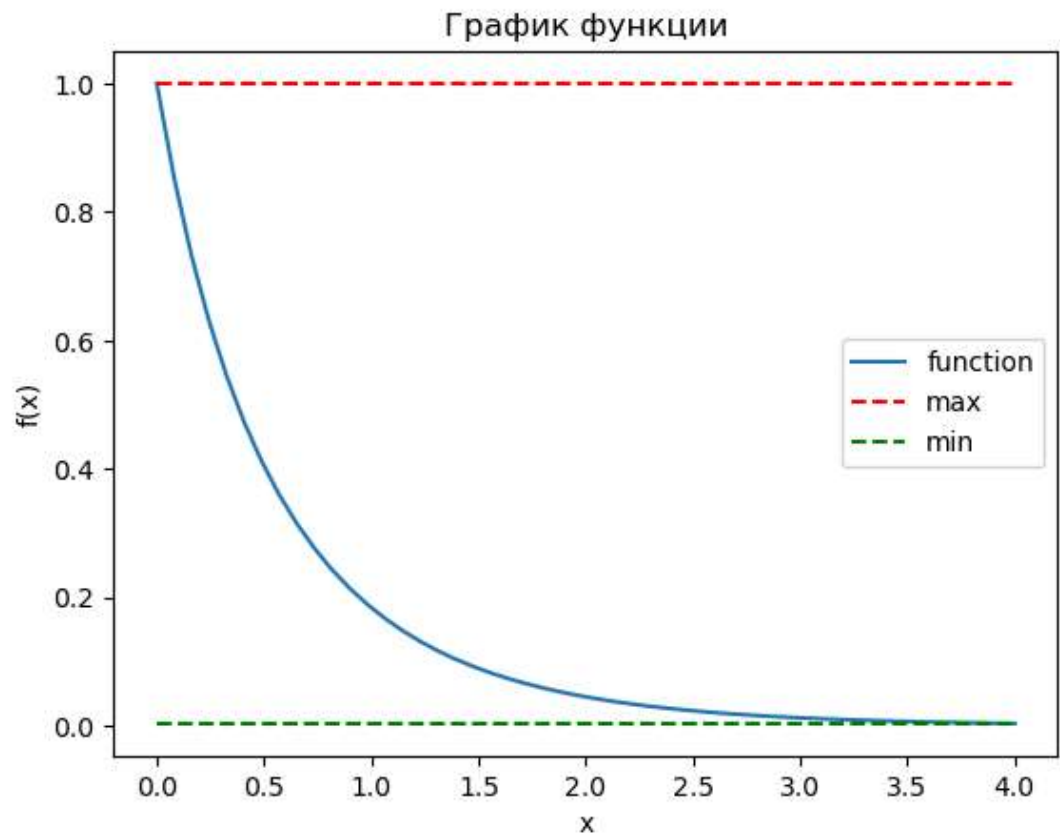
```
tf.Tensor(
[1.          0.852055  0.73015654 0.628794   0.5438415  0.47215426
 0.4112989  0.3593661  0.3148402  0.27650544 0.24337724 0.21465173
 0.18966754 0.16787684 0.14882305 0.13212362 0.1174562  0.10454802
 0.09316734 0.08311635 0.07422567 0.0663498  0.05936331 0.05315779
 0.04763931 0.04272623 0.03834748 0.03444102 0.0309526  0.02783469
 0.02504558 0.02254857 0.02031137 0.01830549 0.01650576 0.01488994
 0.01343831 0.01213342 0.01095976 0.00990354 0.00895252 0.00809579
 0.00732363 0.00662738 0.00599927 0.00543242 0.00492062 0.00445836
 0.00404067 0.00366313], shape=(50,), dtype=float32)
```

2. Постройте график функции с прямыми, соответствующими максимальному и минимальному значениям, подписывая оси и рисунок и создавая легенду.

```
In [4]: fig, ax = plt.subplots()
ax.plot(segment, func(segment), label='function')

ax.plot([0, 4], [max_value, max_value], 'r--', label='max')
ax.plot([0, 4], [min_value, min_value], 'g--', label='min')

ax.set_xlabel('x')
ax.set_ylabel('f(x)')
ax.set_title('График функции')
ax.legend()
plt.show()
```



3. Найдите значения производной от функции порядка, указанного в индивидуальном задании, и постройте график полученной функции, подписывая оси и рисунок.

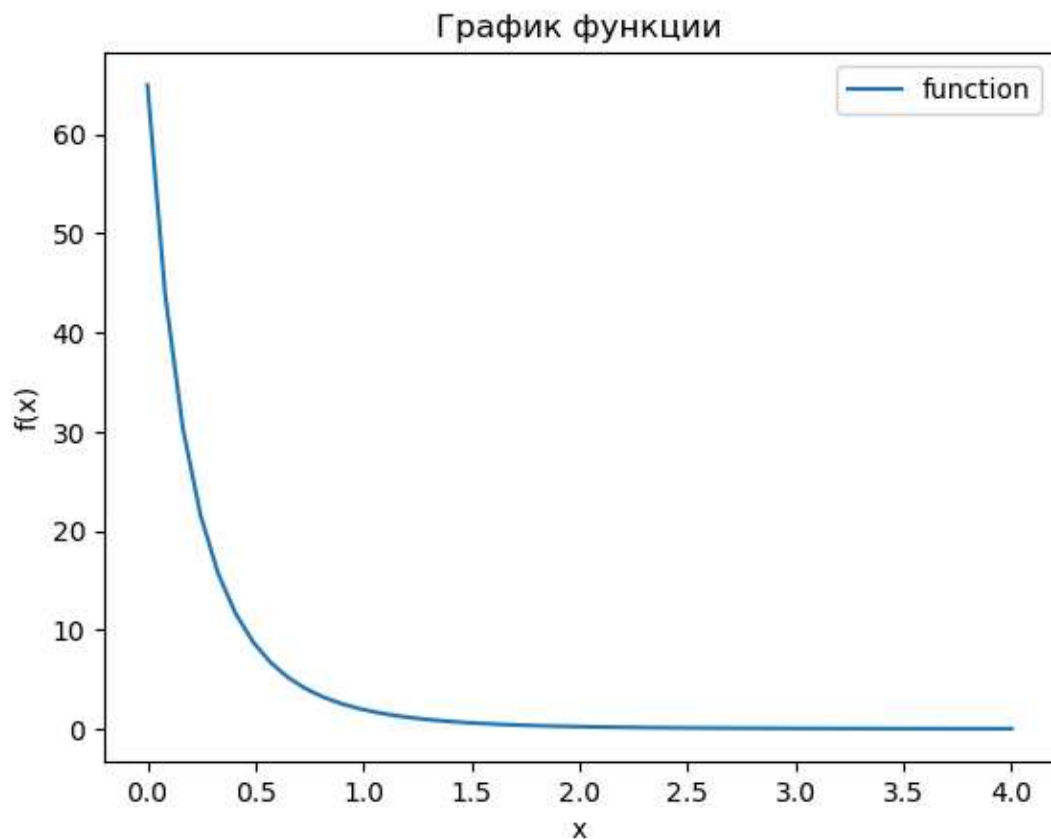
```
In [5]: ▶ def f4(x):
        with tf.GradientTape() as tape1:
            with tf.GradientTape() as tape2:
                with tf.GradientTape() as tape3:
                    with tf.GradientTape() as tape4:
                        y = func(x)
                        dy_dx = tape4.gradient(y, x)
                        d2y_dx2 = tape3.gradient(dy_dx, x)
                        d3y_dx3 = tape2.gradient(d2y_dx2, x)
                        d4y_dx4 = tape1.gradient(d3y_dx3, x)
                    return d4y_dx4

        f4(segment)
```

```
Out[5]: <tf.Tensor: shape=(50,), dtype=float32, numpy=
array([6.5000000e+01, 4.3842854e+01, 3.0418324e+01, 2.1623247e+01,
       1.5699116e+01, 1.1610818e+01, 8.7284899e+00, 6.6574645e+00,
       5.1439486e+00, 4.0209074e+00, 3.1760845e+00, 2.5326047e+00,
       2.0369067e+00, 1.6510894e+00, 1.3479444e+00, 1.1076814e+00,
       9.1573012e-01, 7.6124120e-01, 6.3605380e-01, 5.3396773e-01,
       4.5023048e-01, 3.8116872e-01, 3.2392049e-01, 2.7623948e-01,
       2.3635083e-01, 2.0284246e-01, 1.7458445e-01, 1.5066715e-01,
       1.3035445e-01, 1.1304766e-01, 9.8257229e-02, 8.5581228e-02,
       7.4688062e-02, 6.5303259e-02, 5.7198457e-02, 5.0183252e-02,
       4.4098061e-02, 3.8808845e-02, 3.4202591e-02, 3.0183759e-02,
       2.6671318e-02, 2.3596397e-02, 2.0900253e-02, 1.8532671e-02,
       1.6450647e-02, 1.4617244e-02, 1.3000670e-02, 1.1573523e-02,
       1.0312108e-02, 9.1959154e-03], dtype=float32)>
```

```
In [6]: fig, ax = plt.subplots()
ax.plot(segment, f4(segment), label='function')

ax.set_xlabel('x')
ax.set_ylabel('f(x)')
ax.set_title('График функции')
ax.legend()
plt.show()
```



4. Постройте тензор ранга 2 (матрицу) со значениями заданной в индивидуальном задании функции двух переменных на заданном в индивидуальном задании прямоугольнике и определите максимальное и минимальное значения функции.

$f(x,y) = x^3 \sin(y)$ в области $[0, 2] \times [0, 2]$

```
In [7]: def func1(x,y):
        return (x**3)*tf.math.sin(y)

segment1 = tf.Variable(np.linspace(0, 2, 20))
segment2 = tf.Variable(np.linspace(0, 2, 20))

X, Y = tf.meshgrid(segment1, segment2)
Z = func1(X,Y)
```

```
In [8]: ▶ print("Матрица:\n", Z.numpy(), "\n")
print("Максимальное значение функции:", tf.reduce_max(Z).numpy())
print("\nМинимальное значение функции:", tf.reduce_min(Z).numpy())
Z.shape
```

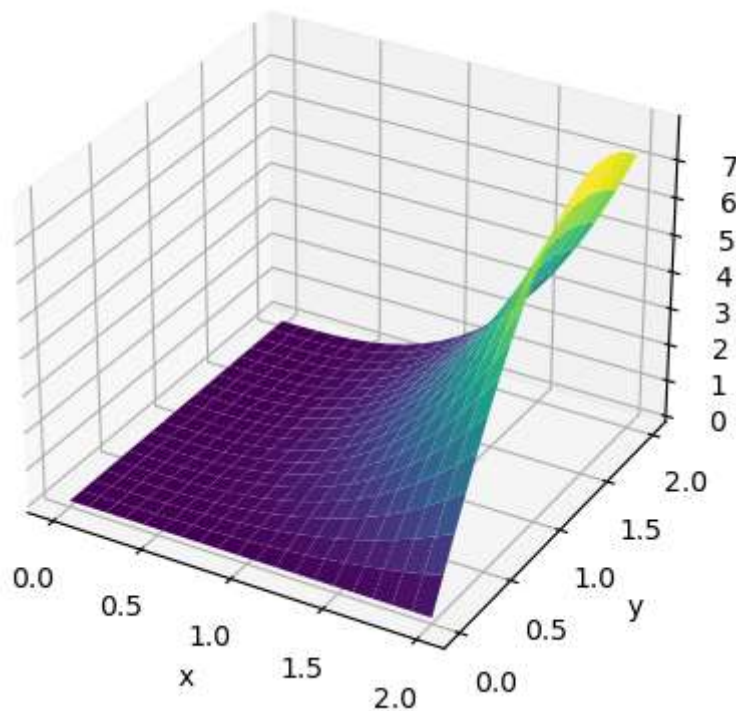
Матрица:

```
[ [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.22547163e-04 9.80377303e-04 3.30877340e-03
  7.84301842e-03 1.53183954e-02 2.64701872e-02 4.20336769e-02
  6.27441474e-02 8.93368817e-02 1.22547163e-01 1.63110274e-01
  2.11761497e-01 2.69236117e-01 3.36269415e-01 4.13596675e-01
  5.01953179e-01 6.02074211e-01 7.14695054e-01 8.40550990e-01]
 [0.00000000e+00 2.43737716e-04 1.94990173e-03 6.58091833e-03
  1.55992138e-02 3.04672145e-02 5.26473466e-02 8.36020365e-02
  1.24793710e-01 1.77684795e-01 2.43737716e-01 3.24414900e-01
  4.21178773e-01 5.35491762e-01 6.68816292e-01 8.22614791e-01
  9.98349684e-01 1.19748340e+00 1.42147836e+00 1.67179699e+00]
 [0.00000000e+00 3.62230067e-04 2.89784053e-03 9.78021180e-03
  2.31827243e-02 4.52787583e-02 7.82416944e-02 1.24244913e-01
  1.85461794e-01 2.64065719e-01 3.62230067e-01 4.82128219e-01
  6.05000000e-01 7.05000000e-01 8.00000000e-01 9.00000000e-01
  1.00000000e+00 1.00000000e+00 1.00000000e+00 1.00000000e+00]
```

5. Постройте 3d график поверхности функции двух переменных, подписывая оси и рисунок.

```
In [9]: fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('График функции  $f(x,y) = x^3 \sin(y)$ ')
plt.show()
```



График функции $f(x,y) = x^3 \sin(y)$



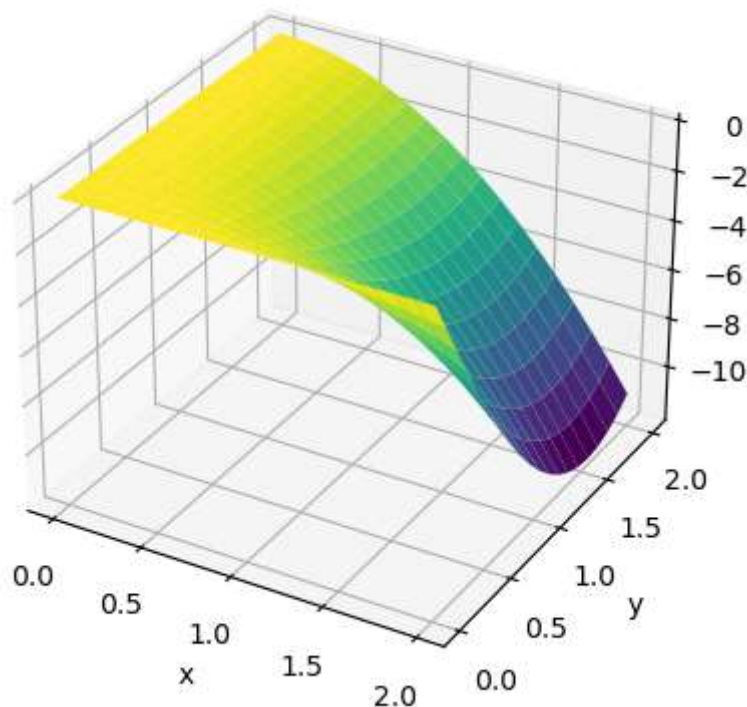
6. Найдите значения смешанной производной от функции порядка, указанного в индивидуальном задании, и постройте 3d график поверхности полученной функции, подписывая оси и рисунок.

Порядок смешанной производной функции двух переменных $d^3/dxdy^2$

```
z1 = f_mixed(X, Y)
z1
```


```
In [11]: fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, z1, cmap='viridis')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_title('График функции  $f(x,y) = x^3 \sin(y)$ ')
plt.show()
```

График функции $f(x,y) = x^3 \sin(y)$ 

7. Решите задачу парной линейной регрессии при помощи модели TensorFlow, рассматривая тензор ранга 1 из пункта 1 задания как значения зависимой переменной (отклика), а точки отрезка из индивидуального задания как значения независимой переменной (предиктора). Предварительно масштабируйте независимую и зависимую переменные на интервал $[0, 1]$. Оцените качество полученной модели по показателю качества регрессии, указанному в индивидуальном задании. Количество эпох, скорость обучения и начальные значения весов выберите самостоятельно, обеспечивая сходимость итерационной процедуры.
8. Постройте кривую обучения для показателя качества регрессии, указанного в индивидуальном задании, с зависимостью от количества эпох. Показатель качества регрессии реализуйте как функцию с использованием функций модуля `tf.math`.
9. Изобразите на графике точки набора данных (независимой и зависимой переменных) и линию построенной парной регрессии, подписывая оси и рисунок и создавая легенду.

In [12]: `tensor`

Out[12]: `<tf.Tensor: shape=(50,), dtype=float32, numpy=
array([1. , 0.852055 , 0.73015654, 0.628794 , 0.5438415 ,
 0.47215426, 0.4112989 , 0.3593661 , 0.3148402 , 0.27650544,
 0.24337724, 0.21465173, 0.18966754, 0.16787684, 0.14882305,
 0.13212362, 0.1174562 , 0.10454802, 0.09316734, 0.08311635,
 0.07422567, 0.0663498 , 0.05936331, 0.05315779, 0.04763931,
 0.04272623, 0.03834748, 0.03444102, 0.0309526 , 0.02783469,
 0.02504558, 0.02254857, 0.02031137, 0.01830549, 0.01650576,
 0.01488994, 0.01343831, 0.01213342, 0.01095976, 0.00990354,
 0.00895252, 0.00809579, 0.00732363, 0.00662738, 0.00599927,
 0.00543242, 0.00492062, 0.00445836, 0.00404067, 0.00366313],
 dtype=float32)>`

In [13]: `class Model(object):
 def __init__(self):
 # Инициализируем вес как `2.0` и смещение как `1.0`
 # На практике инициализация должна быть случайными значениями (`tf.
 self.w = tf.Variable(3.0)
 self.b = tf.Variable(1.0)

 def __call__(self, x):
 return self.w * x + self.b

model = Model()`

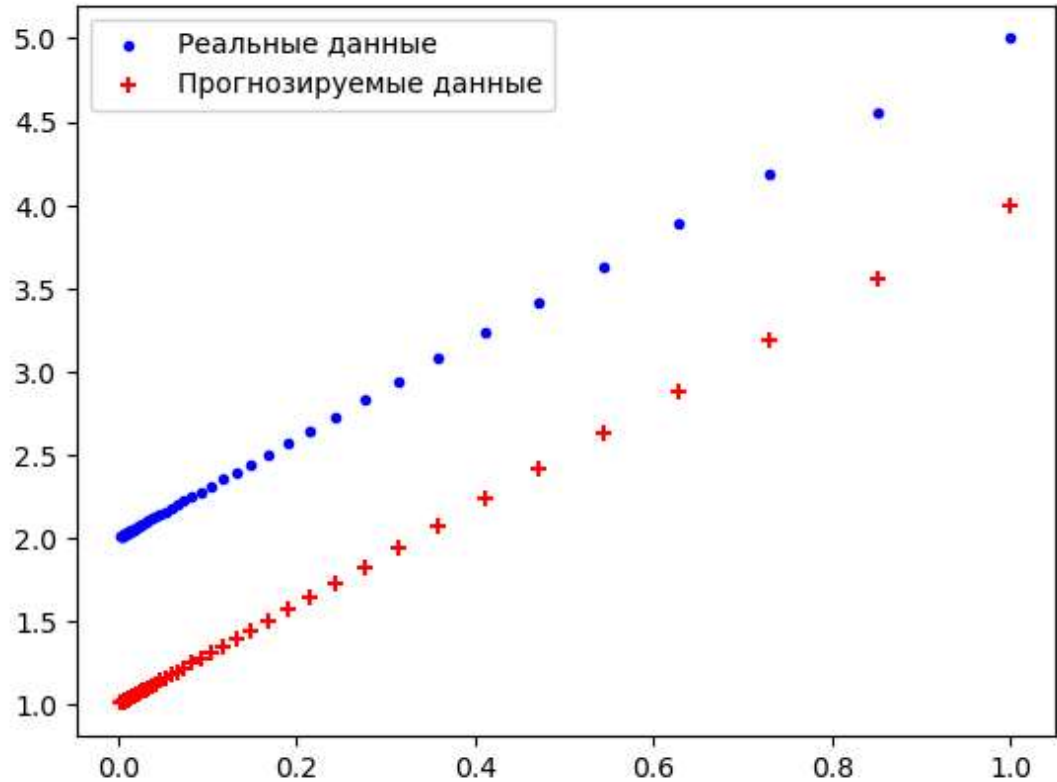
In [14]: `def loss(predicted_y, target_y):
 return tf.reduce_mean(tf.square(predicted_y - target_y))`

In [15]: `TRUE_w = 3.0
TRUE_b = 2.0
NUM_EXAMPLES = 1000

ys = (TRUE_w * tensor) + TRUE_b`

In [16]: `def plot_data(inputs, outputs, predicted_outputs):
 real = plt.scatter(inputs, outputs, c='b', marker='.')
 predicted = plt.scatter(inputs, predicted_outputs, c='r', marker='+')
 plt.legend((real, predicted), ('Реальные данные', 'Прогнозируемые данн
 plt.show()`

```
In [17]: plot_data(tensor, ys, model(tensor))
print('Текущие потери (ошибка): %1.6f' % loss(model(tensor), ys).numpy())
```



Текущие потери (ошибка): 1.000000

```
In [18]: def train(model, inputs, outputs, learning_rate):
    with tf.GradientTape() as t:
        current_loss = loss(model(inputs), outputs)
        dw, db = t.gradient(current_loss, [model.w, model.b])
        model.w.assign_sub(learning_rate * dw)
        model.b.assign_sub(learning_rate * db)

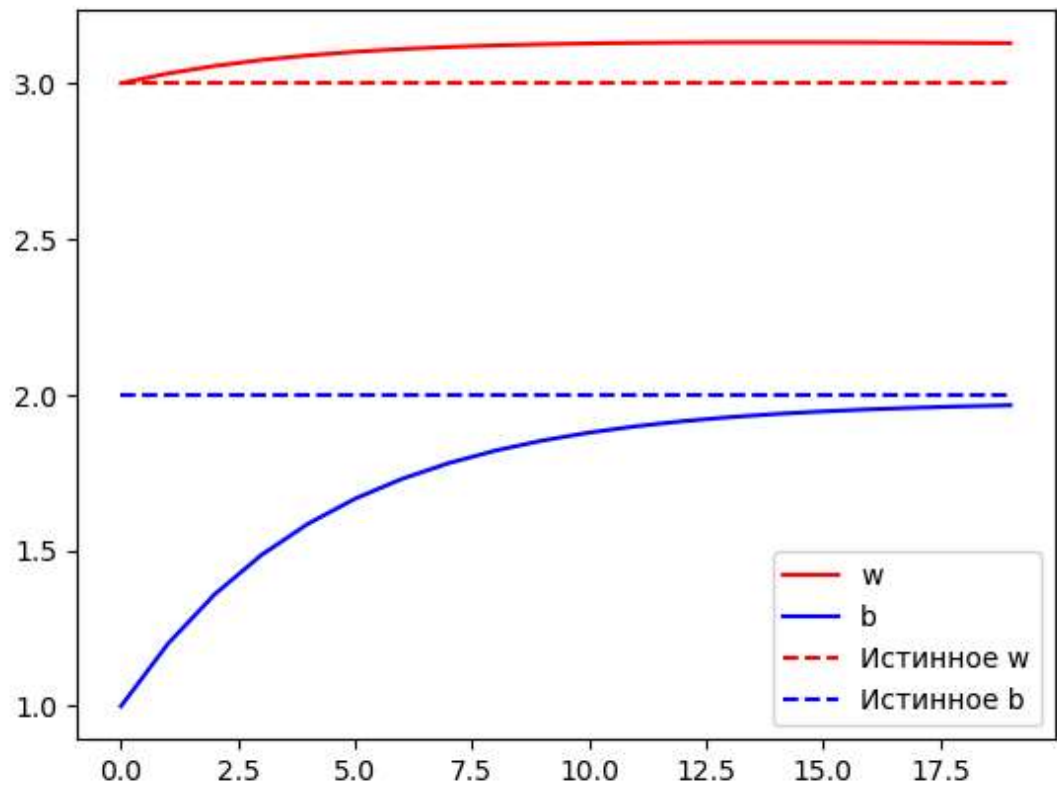
    return current_loss
```

```
In [19]: model = Model()

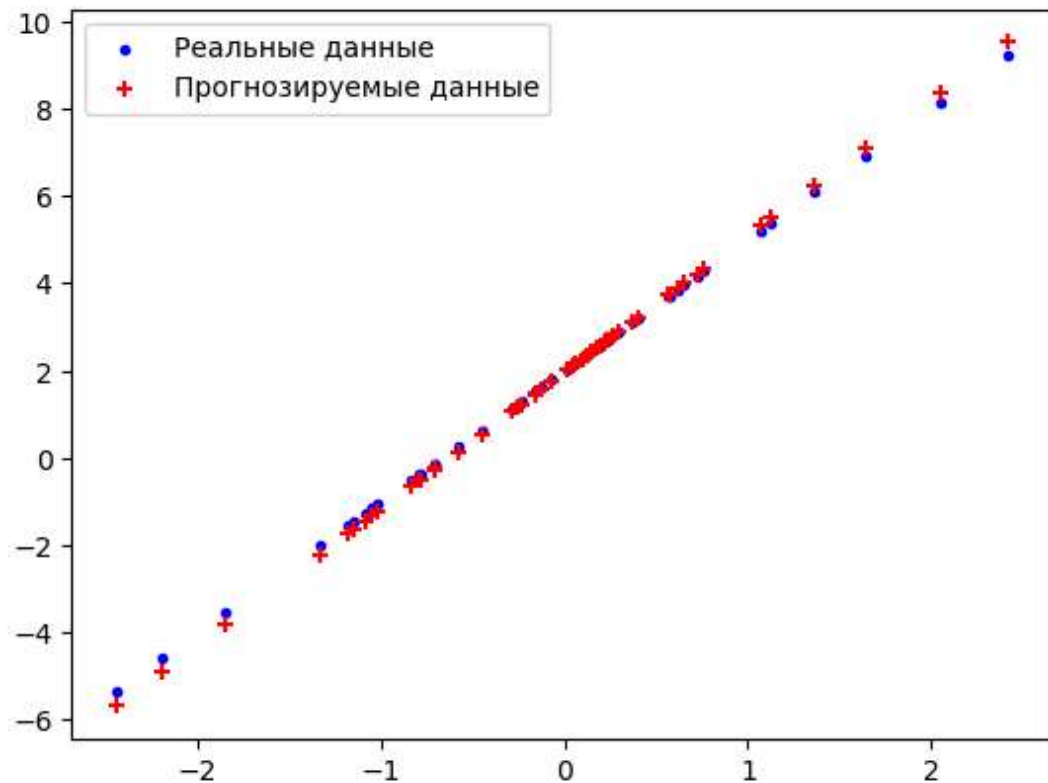
# Запоминаем историю значений 'w' и 'b' для визуализации
list_w, list_b = [], []
epochs = range(20)
losses = []
for epoch in epochs:
    list_w.append(model.w.numpy())
    list_b.append(model.b.numpy())
    current_loss = train(model, tensor, ys, learning_rate=0.1)
    losses.append(current_loss)
    print('Эпоха %2d: w=%1.2f b=%1.2f, потери=%2.5f' %
          (epoch, list_w[-1], list_b[-1], current_loss))
```

```
Эпоха 0: w=3.00 b=1.00, потери=1.00000
Эпоха 1: w=3.03 b=1.20, потери=0.63232
Эпоха 2: w=3.06 b=1.36, потери=0.40000
Эпоха 3: w=3.07 b=1.49, потери=0.25320
Эпоха 4: w=3.09 b=1.59, потери=0.16043
Эпоха 5: w=3.10 b=1.67, потери=0.10181
Эпоха 6: w=3.11 b=1.73, потери=0.06476
Эпоха 7: w=3.12 b=1.78, потери=0.04135
Эпоха 8: w=3.12 b=1.82, потери=0.02654
Эпоха 9: w=3.13 b=1.85, потери=0.01718
Эпоха 10: w=3.13 b=1.88, потери=0.01125
Эпоха 11: w=3.13 b=1.90, потери=0.00750
Эпоха 12: w=3.13 b=1.91, потери=0.00512
Эпоха 13: w=3.13 b=1.93, потери=0.00361
Эпоха 14: w=3.13 b=1.94, потери=0.00265
Эпоха 15: w=3.13 b=1.95, потери=0.00203
Эпоха 16: w=3.13 b=1.95, потери=0.00163
Эпоха 17: w=3.13 b=1.96, потери=0.00138
Эпоха 18: w=3.13 b=1.96, потери=0.00121
Эпоха 19: w=3.13 b=1.97, потери=0.00109
```

```
In [20]: ▶ plt.plot(epochs, list_w, 'r', epochs, list_b, 'b')  
plt.plot([TRUE_w] * len(epochs), 'r--', [TRUE_b] * len(epochs), 'b--')  
plt.legend(['w', 'b', 'Истинное w', 'Истинное b'])  
plt.show()
```



```
In [21]: test_inputs = tf.random.normal(shape=[tensor.shape[0]])  
test_outputs = test_inputs * TRUE_w + TRUE_b  
  
predicted_test_outputs = model(test_inputs)  
plot_data(test_inputs, test_outputs, predicted_test_outputs)
```



```
In [22]: ▶ def plot_loss_for_weights(weights_list, losses):
    for idx, weights in enumerate(weights_list):
        plt.subplot(120 + idx + 1)
        plt.plot(weights['values'], losses, 'r')
        plt.plot(weights['values'], losses, 'bo')
        plt.xlabel(weights['name'])
        if idx==0:
            plt.ylabel('Ошибка')

    weights_list = [{ 'name' : "w",
                      'values' : list_w
                    },
                    {
                      'name' : "b",
                      'values' : list_b
                    }
                ]

    plot_loss_for_weights(weights_list, losses)
```

