Практическое задание 4
Вариант 6

Задание

Необходимо реализовать нейронную сеть вычисляющую результат заданной логической операции. Затем реализовать функции, которые будут симулировать работу построенной модели. Функции должны принимать тензор входных данных и список весов. Должно быть реализовано 2 функции:

1. Функция, в которой все операции реализованы как поэлементные операции над тензорами
2. Функция, в которой все операции реализованы с использованием операций над тензорами из NumPy

Для проверки корректности работы функций необходимо:

1. Инициализировать модель и получить из нее веса (Как получить веса слоя, Как получить список слоев модели)
2. Прогнать датасет через не обученную модель и реализованные 2 функции. Сравнить результат.
3. Обучить модель и получить веса после обучения
4. Прогнать датасет через обученную модель и реализованные 2 функции. Сравнить результат.

Функция (a and not b) or (c xor b)

Выполнение

Была реализована нейросеть вычисляющая результат заданной логической операции.

```
model = Sequential()
model.add(Dense(10, activation='relu', input_shape=(3,)))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Затем были реализованы функции, которые будут симулировать работу построенной модели.

Функция numpy_func, в которой все операции реализованы с использованием операций над тензорами из NumPy.

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))


def numpy_func(weights, train_data):
        layer1 = np.maximum(np.dot(train_data, weights[0].get_weights()[0]) + weights[0].get_weights()[1], 0.)
        layer2 = np.maximum(np.dot(layer1, weights[1].get_weights()[0]) + weights[1].get_weights()[1], 0.)
        layer3 = sigmoid(np.dot(layer2, weights[2].get_weights()[0]) + weights[2].get_weights()[1])
    return layer3
```

Функция elementwise_func, в которой все операции реализованы как поэлементные операции над тензорами.

```
def elementwise_func(weights, train_data):
    layer1 = naive_relu(naive_add_matrix_and_vector(naive_vector_dot(train_data, weights[0].get_weights()[0]),
    weights[0].get_weights()[1]))
    layer2 = naive_relu(naive_add_matrix_and_vector(naive_vector_dot(layer1, weights[1].get_weights()[0]),
    weights[1].get_weights()[1]))
    layer3 = naive_sigmoid(naive_add_matrix_and_vector(naive_vector_dot(layer2, weights[2].get_weights()[0]),
    weights[2].get_weights()[1]))
    return layer3
```

Для работы данной функции были реализованы функции для работы с матрицами, в которых все операции реализованы как поэлементные операции над тензорами.

```
def naive_relu(x):
    assert len(x.shape) == 2
```

```
    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i, j] = max(x[i, j], 0)
    return x

def naive_sigmoid(x):
    assert len(x.shape) == 2
    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i, j] = 1 / (1 + math.exp(-x[i, j]))
    return x

def naive_add_matrix_and_vector(x, y):
    assert len(x.shape) == 2
    assert len(y.shape) == 1
    assert x.shape[1] == y.shape[0]
    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i, j] += y[j]
    return x

def naive_vector_dot(x, y):
    assert len(x.shape) == 2
    assert len(y.shape) == 2
    assert x.shape[1] == y.shape[0]
```

```
    z = np.zeros([x.shape[0], y.shape[1]])
    for i in range(x.shape[0]):
        for j in range(y.shape[1]):
            s = 0
            for k in range(x.shape[1]):
                s += x[i, k] * y[k, j]
            z[i, j] += s
    return z
```

Датасет был прогнан через не обученную модель:

```
numpy_func
[[0.5       ]
 [0.58513208]
 [0.47792075]
 [0.64522508]
 [0.50915563]
 [0.65823998]
 [0.55321806]
 [0.72474086]]
elementwise_func
[[0.5       ]
 [0.58513208]
 [0.47792075]
 [0.64522508]
 [0.50915563]
 [0.65823998]
 [0.55321806]
 [0.72474086]]
```

Получили случайные значения около 0.5.

Датасет был прогнан через обученную модель:

```
numpy_func
[[0.05346096]
 [0.98550589]
 [0.99295277]
 [0.00457701]
 [0.99764034]
 [0.99804052]
 [0.99844086]
 [0.00459677]]
elementwise_func
[[0.05346096]
 [0.98550589]
 [0.99295277]
 [0.00457701]
 [0.99764034]
 [0.99804052]
 [0.99844086]
 [0.00459677]]
```

Значения близки к верным. Различия видны только на 3 знаке после запятой.