

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по практической работе №4
по дисциплине «Искусственные нейронные сети»
Тема: Операции с тензорами в библиотеке Keras
Вариант 5

Студентка гр. 8383

Максимова А.А.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

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.
2. Прогнать датасет через не обученную модель и 2 реализованные функции. Сравнить результаты.
3. Обучить модель и получить веса после обучения.
4. Прогнать датасет через обученную модель и 2 реализованные функции. Сравнить результаты.

Примечание: так как множество всех наблюдений ограничен, то обучение проводить можно на всем датасете без контроля.

Задание по варианту

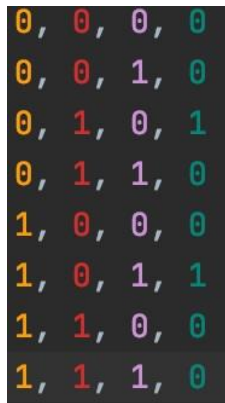
(a xor b) and (b xor c)

Реализация

1. Были импортированы все необходимые для работы классы и функции.

```
import pandas
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
```

2. Набор входных данных содержится в файле "dataset.csv".



0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

3. Была определена функция для создания модели ИНС прямого распространения, состоящей из четырех полносвязанных слоев: первый (входной) содержит 3 нейрона, второй и третий (скрытые) - соответственно 32 и 16 нейронов, функция активации - Relu: $\max(0, x)$; выходной слой - 1 нейрон, функция активации - Sigmoid: $\frac{1}{1 + e^{-x}}$.

Параметры обучения сети: в качестве функции потерь используется "binary_crossentropy", оптимизатор - "adam", метрика - точность.

```
def build_model(sizeLayers):  
    model = Sequential()  
    model.add(Dense(sizeLayers[1], input_dim=sizeLayers[0], activation='relu'))  
    model.add(Dense(sizeLayers[2], activation='relu'))  
    model.add(Dense(sizeLayers[3], activation='sigmoid'))  
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  
    return model
```

4. Была реализована рекурсивная функция, симулирующая работу нейронной сети, в которой все операции реализованы с использованием операций над тензорами из NumPy.

```
def f2(weights, bias, input_layer, sizeLayers, index_layer):  
    if(index_layer == len(weights)-1):  
        return 1 / (1 + np.exp(-(np.dot(input_layer, weights[index_layer]) + bias[index_layer])))  
    else:  
        return f2(weights, bias, np.maximum(np.dot(input_layer, weights[index_layer]) + bias[index_layer], 0),  
                  sizeLayers, index_layer + 1)
```

5. Была реализована рекурсивная функция, симулирующая работу нейронной сети, в которой все операции реализованы как поэлементные операции над тензорами.

```
def f1(weights, bias, input_layer, sizeLayers, index_layer):
    new_layer = np.zeros(sizeLayers[index_layer + 1]) # для значений нейронов след слоя
    if(index_layer == len(weights)-1): # sigmoid = 1/(1+e^(-x))
        dot_ = 0
        for j in range(0, len(weights[index_layer])):
            dot_ += weights[index_layer][j] * input_layer[j]
        return 1 / (1 + np.exp(-(dot_ + bias[index_layer])))
    else: # relu = max(0, x)
        for i in range(0, len(new_layer)): # по нейронам нового слоя
            dot_ = 0
            for j in range(0, len(weights[index_layer])):
                dot_ += weights[index_layer][j][i] * input_layer[j]
            new_layer[i] = max(dot_ + bias[index_layer][i], 0)
    return f1(weights, bias, new_layer, sizeLayers, index_layer + 1)
```

6. Была написана функция для получения весов и смещений модели:

```
def getWeightsBias(model):
    weight_bias = model.get_weights() # получение весов и смещений
    weights = [weight_bias[i] for i in range(0, len(weight_bias), 2)]
    bias = [weight_bias[i] for i in range(1, len(weight_bias), 2)]
    return weights, bias
```

7. Также была написана функция, печатающая результаты прогонки входных данных через модель и две реализованные функции в удобном формате:

```
def start(weights, bias, train_data, train_targets, sizeLayers, case): # прогон датасета
    if(case):
        print("Прогон датасета через необученную модель и 2 реализованные функции:\n")
    else:
        print("Прогон датасета через обученную модель и 2 реализованные функции:\n")

    ins_res = model.predict(train_data)
    index = 0
    for obj in train_data:
        print("Дано: (", obj[0], "xor", obj[1], ") and (", obj[1], "xor", obj[2], ") = ", train_targets[index])
        print("Предсказание нейронной сети:", ins_res[index][0])
        print("Предсказание function1 (поэлементные операции):", f1(weights, bias, obj, sizeLayers, 0))
        print("Предсказание function2 (с использованием NumPy):", f2(weights, bias, obj, sizeLayers, 0))
        print("\n")
        index += 1
```

Проверка корректности работы функций

Результаты прогона входных данных через не обученную модель и 2 реализованные функции:

Дано: $(0 \text{ xor } 0) \text{ and } (0 \text{ xor } 0) = 0$

Предсказание нейронной сети: 0.5

Предсказание function1 (поэлементные операции): [0.5]

Предсказание function2 (с использованием NumPy): [0.5]

Дано: $(0 \text{ xor } 0) \text{ and } (0 \text{ xor } 1) = 0$

Предсказание нейронной сети: 0.51871204

Предсказание function1 (поэлементные операции): [0.51871204]

Предсказание function2 (с использованием NumPy): [0.51871206]

Дано: $(0 \text{ xor } 1) \text{ and } (1 \text{ xor } 0) = 1$

Предсказание нейронной сети: 0.59520316

Предсказание function1 (поэлементные операции): [0.59520316]

Предсказание function2 (с использованием NumPy): [0.59520316]

Дано: $(0 \text{ xor } 1) \text{ and } (1 \text{ xor } 1) = 0$

Предсказание нейронной сети: 0.5514311

Предсказание function1 (поэлементные операции): [0.5514311]

Предсказание function2 (с использованием NumPy): [0.5514311]

Дано: $(1 \text{ xor } 0) \text{ and } (0 \text{ xor } 0) = 0$

Предсказание нейронной сети: 0.5298151

Предсказание function1 (поэлементные операции): [0.5298151]

Предсказание function2 (с использованием NumPy): [0.5298151]

Дано: $(1 \text{ xor } 0) \text{ and } (0 \text{ xor } 1) = 1$

Предсказание нейронной сети: 0.4940739

Предсказание function1 (поэлементные операции): [0.4940739]

Предсказание function2 (с использованием NumPy): [0.4940739]

Дано: (1 xor 1) and (1 xor 0) = 0

Предсказание нейронной сети: 0.6221652

Предсказание function1 (поэлементные операции): [0.62216514]

Предсказание function2 (с использованием NumPy): [0.6221652]

Дано: (1 xor 1) and (1 xor 1) = 0

Предсказание нейронной сети: 0.58538383

Предсказание function1 (поэлементные операции): [0.58538383]

Предсказание function2 (с использованием NumPy): [0.58538385]

Результаты прогона входных данных через обученную модель и 2 реализованные функции:

Дано: (0 xor 0) and (0 xor 0) = 0

Предсказание нейронной сети: 0.09444839

Предсказание function1 (поэлементные операции): [0.09444835]

Предсказание function2 (с использованием NumPy): [0.09444836]

Дано: (0 xor 0) and (0 xor 1) = 0

Предсказание нейронной сети: 0.10465753

Предсказание function1 (поэлементные операции): [0.10465757]

Предсказание function2 (с использованием NumPy): [0.10465756]

Дано: (0 xor 1) and (1 xor 0) = 1

Предсказание нейронной сети: 0.87333363

Предсказание function1 (поэлементные операции): [0.8733336]

Предсказание function2 (с использованием NumPy): [0.87333362]

Дано: (0 xor 1) and (1 xor 1) = 0

Предсказание нейронной сети: 0.046280295

Предсказание function1 (поэлементные операции): [0.04628034]

Предсказание function2 (с использованием NumPy): [0.04628032]

Дано: (1 xor 0) and (0 xor 0) = 0

Предсказание нейронной сети: 0.023412734

Предсказание function1 (поэлементные операции): [0.02341269]

Предсказание function2 (с использованием NumPy): [0.02341268]

Дано: (1 xor 0) and (0 xor 1) = 1

Предсказание нейронной сети: 0.7068461

Предсказание function1 (поэлементные операции): [0.7068461]

Предсказание function2 (с использованием NumPy): [0.70684609]

Дано: $(1 \text{ xor } 1) \text{ and } (1 \text{ xor } 0) = 0$
Предсказание нейронной сети: 0.031497926
Предсказание function1 (позлементные операции): [0.03149793]
Предсказание function2 (с использованием NumPy): [0.03149793]

Дано: $(1 \text{ xor } 1) \text{ and } (1 \text{ xor } 1) = 0$
Предсказание нейронной сети: 0.076043695
Предсказание function1 (позлементные операции): [0.07604375]
Предсказание function2 (с использованием NumPy): [0.07604373]

Как видно, результаты прогона входных данных через модель и 2 реализованные функции до и после обучения эквиваленты между собой с минимальной погрешностью.