Практика 4. Вариант 8. Кобенко Влад 8382

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:

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

Для проверки корректности работы функций необходимо:

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

Выполнение работы:

Было реализовано 2 функции:

```
def np_res(W, B, input): # функция с использованием numpy
    a = input.copy()
    for i in range(len(W)):
        a = np.dot(a, W[i])
        a += B[i]
        a = activation['relu'](a) if i != range(len(W))[-1] else activation['sigmoid'](a)
    return a
```

В массиве(размером == размеру тензора) активируем relu, пока не дошли до последнего слоя, после активируем sigmoid. Таким образом происходит обучение нейросети.

```
def elem_res(W, B, input): # функция с использованием поэлементных операций
    a = input.copy()
    for i in range(len(W)):
        a = np.array([matrix_dot(el, W[i]) for el in a])
        a = np.array([matrix_add_vector(el, B[i]) for el in a])
        a = [activation['relu'](el) for el in a] if i != range(len(W))[-1] else [activation['sigmoid'](el) for el in a]
    return np.array(a)
```

Здесь же все устроено по похожему принципу, различие заключается в том, что используются поэлементные операции с тензорами без использования numpy.

Пример работы программы:

Модель:

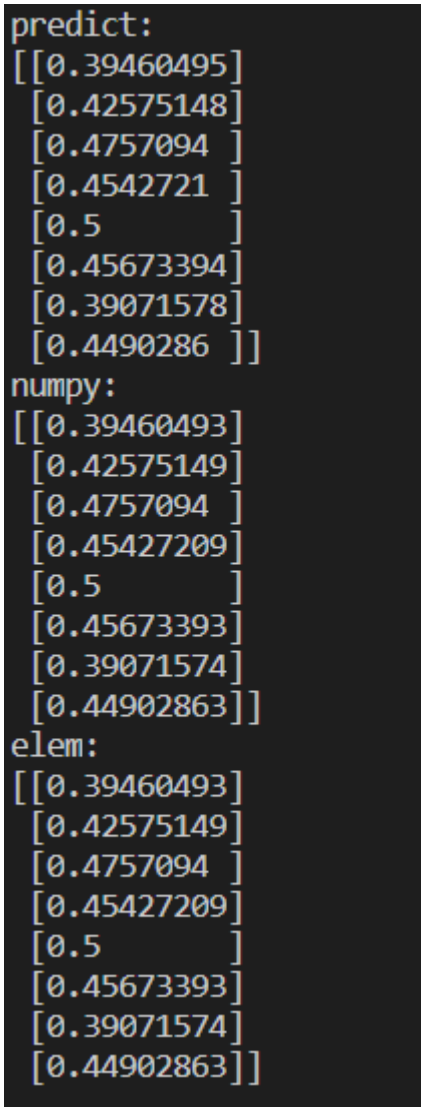
```
model = Sequential()
model.add(Dense(16, activation='relu', input_dim=3))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print('Необученная модель\n')
result(a, model)
H = model.fit(a, b, epochs=100, batch_size=1, verbose=0)
print('\nОбученная модель\n')
result(a, model)
```

Dataset:

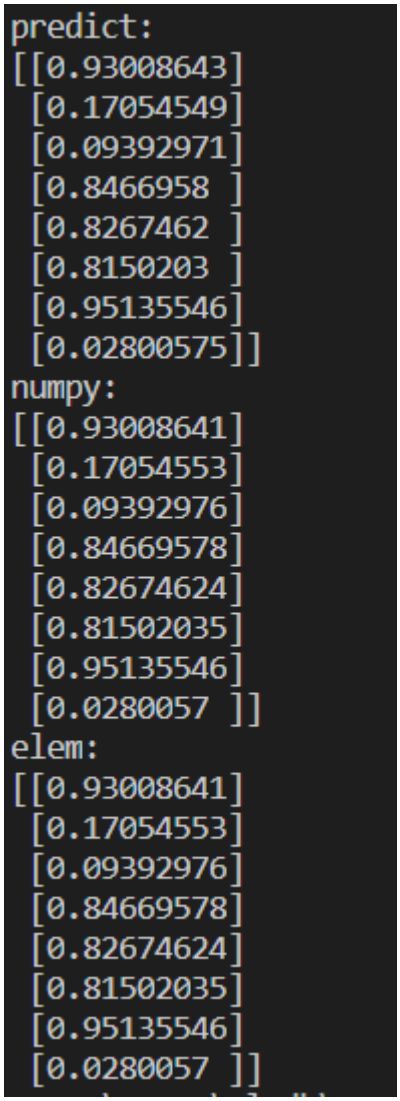
```
a = np.array([[1, 0, 1], [1, 1, 1], [0, 1, 0], [1, 1, 0],
              [0, 0, 0], [0, 0, 1], [1, 0, 0], [0, 1, 1]])
b = np.array([bool_operation(i[0],i[1],i[2]) for i in a])
```

Результат:

Необученная модель:



Обученная модель:



Вывод:

Программа работает корректно, так данные практически не отличаются