Введение в глубокое обучение

Лабораторная работа №1

Содержание

- Векторная математика (простейшие операции с тензорами)
- Реализация двухслойной нейронной сети с прямым распространением
- Реализация трехслойной нейронной сети с прямым распространением с использованием библиотеки NumPy

Работа в онлайн среде Jupiter Notebook

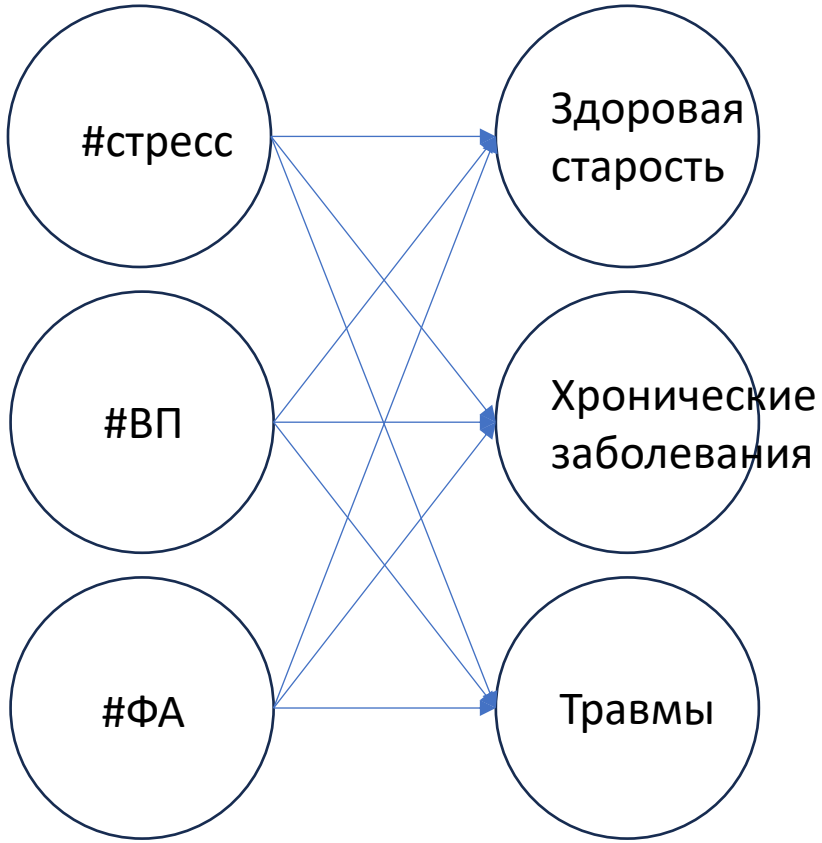
- Зайдите на ресурс https://jupyter.org/
- Создайте новый ноутбук
- Начинаем работать.

Простейшие операции с тензорами

Реализуйте следующие алгоритмы в виде функций:

1. Поэлементное сложение двух векторов
2. Поэлементное умножение двух векторов
3. Сумма вектора
4. Среднее значение вектора
5. Скалярное произведение двух векторов (взвешенная сумма) или умножение с накоплением **def w_sum(a, b)**
6. Умножение вектора на матрицу **def vect_mat_mul(vect, matrix)** с использованием функции def w_sum(a,b). Пусть вектор имеет размер 3, а матрица 3x3 элемента.

Реализуйте двухслойную нейронную сеть.




```
                #стресс   #ВП      #ФА
weights = [[0.01,     0.09,     0.01],  # здоровая старость
           [0.01,     -0.04,     0.00],    # хронические забол.
           [0.00,     0.17,     0.03]]   # травмы

stress = [5, 8, 15, 6, 30]
bad_habits = [8, 4, 2, 6, 1]
phys_active =  [1, 1, 2, 1, 3]
input = [stress[0], bad_habits[0], phys_active[0]]

# для КАЖДОГО выхода вычисляется взвешенная сумма
# входов
def w_sum(a, b):
#...
def vect_mat_mul(vect, matrix):
#...
def neural_network(input, weight):
        prediction = vect_mat_mul(input,weight)
        return prediction
pred = neural_network(input, weights)
print(pred)
```

Подключение библиотеки NumPy для упрощенной работы с тензорами

1. Импортирование библиотеки NumPy: import numpy as np
2. Запись вектора: stress = np.array([5, 8, 15, 6, 30])
3. Запись матрицы:

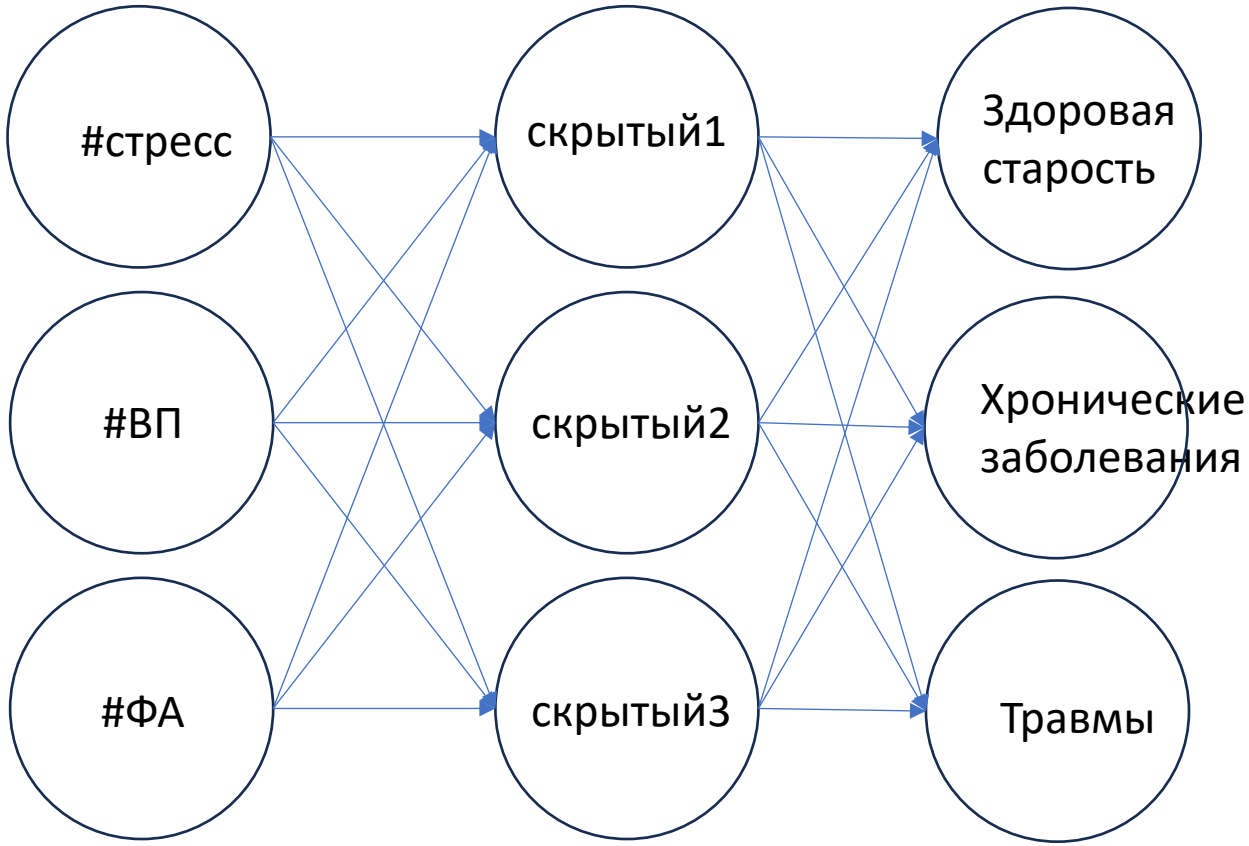
```
weights = np.array([[0.01,      0.09,      0.01],
                    [0.01,      -0.04,      0.00],
                    [0.00,      0.17,      0.03]]).T
```

В реализации нейронной сети матрицу нужно транспонировать!!!

3. Запись скалярного произведения:

```
prediction = input.dot(weights)
```

Реализуйте 3-слойную нейросеть без и с использованием библиотеки NumPy



Данную НС можно представить как три независимых **скалярных произведения – ф-я vect_mat_mul**

```
                #стресс   #ВП      #ФА
weights1 = [[0.01,     0.09,     0.01],  # скрытый1
            [0.01,     -0.04,    0.00],   # скрытый2
            [0.00,     0.17,     0.03]]   # скрытый3

                #скрыт1  #скрыт2  #скрыт3
weights2 = [[0.06,     0.02,     -0.01],  # здоровая старость
            [0.00,     0.10,     0.00],   # хронические забол.
            [0.02,     -0.17,    0.04]]   # травмы

weights = [weights1, weights2]

# входные значения
stress = [5, 8, 15, 6, 30]
bad_habits = [8, 4, 2, 6, 1]
phys_active =  [1, 1, 2, 1, 3]
input = [stress[0], bad_habits[0], phys_active[0]]
def neural_network(input, weights):
            pred_hid = vect_mat_mul(input,weights[0])
            pred_out = vect_mat_mul(pred_hid,weights[1])
            return prediction

pred = neural_network(input, weights)
print(pred)
```