ИНТЕЛЛЕКТУАЛЬНЫЕ МЕТОДЫ В ЗАДАЧАХ ЗАЩИТЫ ОТ КИБЕРУГРОЗ

ЛАБОРАТОРНАЯ РАБОТА № 2
Алгоритм обучения по методу обратного определения.

Выполнил:
Мосолков Е.Н.
Преподаватель:
Петров А.А.

Москва 2021 г.

ЦЕЛЬ РАБОТЫ

Освоить на практике применение алгоритма обратного распространения ошибки

ОПИСАНИЕ ЗАДАЧИ

Провести апробацию: индивидуально выбрать начальные веса. Возможен выбор функции активации, шага Lmbd, количества обучающих эпох и иных параметров (поощряется индивидуальность подхода).

КОД

```
import numpy as np


def f(x):
    return 2/(1 + np.exp(-x)) - 1


def df(x):
    return 0.5*(1 + x)*(1 - x)


W1 = np.array([[-0.2, 0.3, -0.4], [0.1, -0.3, -0.4]])
W2 = np.array([0.2, 0.3])


def go_forward(inp):
    sum = np.dot(W1, inp)
    out = np.array([f(x) for x in sum])
    sum = np.dot(W2, out)
    y = f(sum)
    return (y, out)


def train(epoch):
    global W2, W1
    lmd = 0.01 # шаг обучения
    N = 10000 # число итераций при обучении
    count = len(epoch)
    for k in range(N):
        x = epoch[np.random.randint(0, count)] # случайных выбор входного сигнала из
обучающей выборки
        y, out = go_forward(x[0:3]) # прямой проход по НС и вычисление выходных значе
ний нейронов
        e = y - x[-1] # ошибка
        delta = e*df(y) # локальный градиент
        W2[0] = W2[0] - lmd * delta * out[0] # корректировка веса первой связи
        W2[1] = W2[1] - lmd * delta * out[1] # корректировка веса второй связи
        delta2 = W2*delta*df(out) # вектор из 2-х величин локальных градиентов
        # корректировка связей первого слоя
        W1[0, :] = W1[0, :] - np.array(x[0:3]) * delta2[0] * lmd
        W1[1, :] = W1[1, :] - np.array(x[0:3]) * delta2[1] * lmd

# обучающая выборка (она же полная выборка)
epoch = [(-1, -1, -1, -1),
        (-1, -1, 1, 1),
        (-1, 1, -1, -1),
        (-1, 1, 1, 1),
        (1, -1, -1, -1),
        (1, -1, 1, 1),
        (1, 1, -1, -1),
        (1, 1, 1, -1)]
```

```
train(epoch) # запуск обучения сети
# проверка полученных результатов
for x in epoch:
    y, out = go_forward(x[0:3])
    print(f"Выходное значение НС: {y} => {x[-1]}")
```

ВЫВОД

Я освоил на практике применение алгоритма обратного распределения ошибки