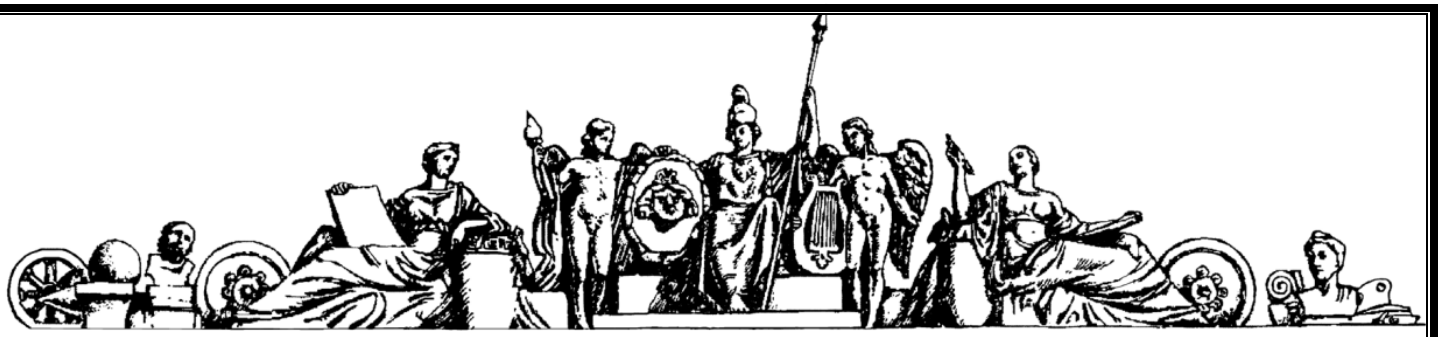

Министерство образования Российской Федерации
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н. Э. БАУМАНА
Факультет: Информатика и системы управления
Кафедра: Информационная безопасность

«Интеллектуальные технологии информационной безопасности»

ЛАБОРАТОРНАЯ РАБОТА № 3

«Изучение алгоритма обратного распространения ошибки (метод Back Propagation)»

Вариант № 6

Преподаватель: Коннова Н.С.
Студент: Кошман А.А.
Группа: ИУ8-61

Москва, 2019

Оглавление

Цель работы

Исследовать функционирование многослойной нейронной сети (МНС) прямого распространения и ее обучение методом обратного распространения ошибки (англ. Back Propagation – BP)

Постановка задачи

На примере МНС архитектуры $N – J – M$ (рис. 1) реализовать ее обучение BP, проведя настройку весов нейронов скрытого $(w_{ij}^{(1)}(k),\ i = \overline{0,\ N},\ j = \overline{1,\ J})$ и выходного $(w_{jm}^{(2)}(k),\ j = \overline{0,\ J},\ m = \overline{1,\ M})$ слоев, где индексы I, j = 0 соответствуют нейронам смещеия; $k = 1,2,\ ...$ - номер эпохи обучения.

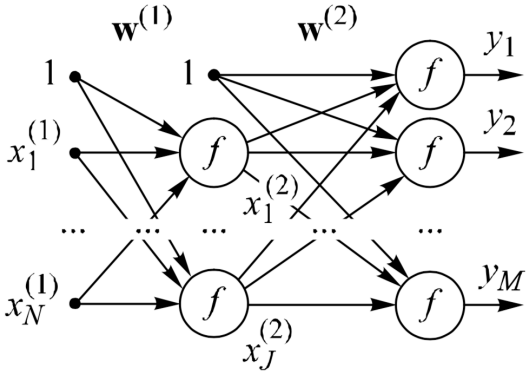

Рисунок 1 – Многослойная НС

Условие

Архитектура: *2 – 1 – 2 (N = 2, J = 1, M = 2)*

Входной вектор: X = *(1, 1, -1)*

Целевой вектор: t = *(0.2, -0.1)*

Результаты эксперимента

В таблице 1 представлены некоторые результаты обучения МНС методом ВР при значении параметра $\eta = 0.3$.

Таблица 1 – Параметры МНС на последовательных эпохах

k	Вектор весов $w^{(1)}$	Вектор весов $w^{(2)}$	Выходной вектор y	Суммарная ошибка E
0	[[0.3878 0.6344 0.4655]]	[[0.03 0.0189] [-0.015 -0.0095]]	[0. 0.]	0.224
1	[[0.388 0.6346 0.4655]]	[[0.0568 0.0359] [-0.0284 -0.0179]]	[0.021 -0.01]	0.2
2	[[0.3883 0.6349 0.4655]]	[[0.0808 0.0511] [-0.0404 -0.0255]]	[0.04 -0.02]	0.179
3	[[0.3887 0.6353 0.4655]]	[[0.1023 0.0646] [-0.0512 -0.0323]]	[0.056 -0.028]	0.16
4	[[0.3892 0.6358 0.4655]]	[[0.1215 0.0767] [-0.0608 -0.0384]]	[0.071 -0.036]	0.144
5	[[0.3897 0.6362 0.4655]]	[[0.1386 0.0876] [-0.0694 -0.0438]]	[0.085 -0.043]	0.129
6	[[0.3902 0.6368 0.4655]]	[[0.154 0.0973] [-0.0771 -0.0487]]	[0.097 -0.049]	0.115
7	[[0.3907 0.6373 0.4655]]	[[0.1677 0.106] [-0.084 -0.0531]]	[0.107 -0.054]	0.103
…	…	…	…	…

47	[[0.3969 0.6435 0.4655]]	[[0.287 0.1817] [-0.1423 -0.0901]]	[0.199 -0.099]	0.002
48	[[0.397 0.6435 0.4655]]	[[0.2872 0.1818] [-0.1424 -0.0901]]	[0.199 -0.099]	0.0018
49	[[0.397 0.6435 0.4655]]	[[0.2874 0.1819] [-0.1424 -0.0902]]	[0.199 -0.1]	0.0015
50	[[0.397 0.6436 0.4655]]	[[0.2875 0.182] [-0.1425 -0.0902]]	[0.199 -0.103]	0.0013
51	[[0.397 0.6436 0.4655]]	[[0.2877 0.1821] [-0.1425 -0.0902]]	[0.199 -0.109]	0.001
52	[[0.397 0.6436 0.4655]]	[[0.2878 0.1822] [-0.1426 -0.0903]]	[0.199 -0.112]	0.001

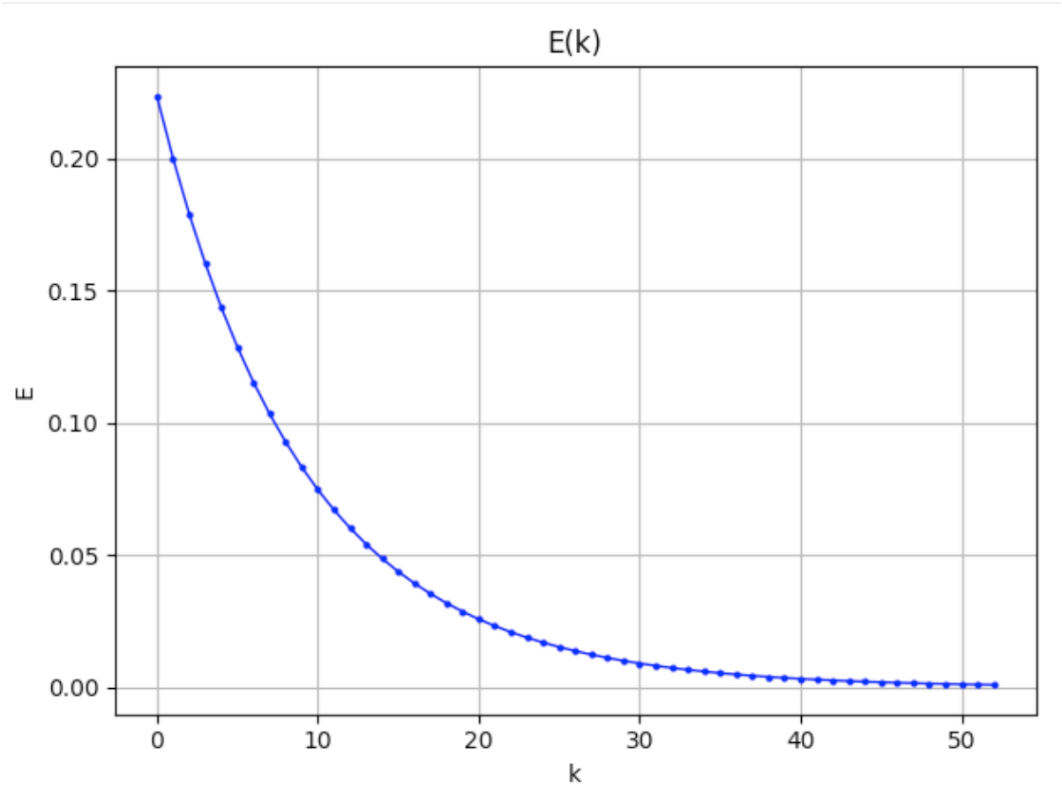

Рисунок 2 - Зависимость средне-квадратичной погрешности E от эпохи k

Выводы

В процессе лабораторной работы было исследовано функционирование многослойной нейронной сети (МНС) прямого распространения и произведено ее обучение методом обратного распространения ошибки.

Основная идея метода BP состоит в распространении сигналов ошибки от выходов сети к её входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. МНС была обучена за 52 эпохи при норме обучения $\eta = 0.3$.

Приложения

Файл ‘script.py’ :

```
import numpy as np
import matplotlib.pyplot as plt
import random

X1 = [1., 1., -1.]   # входной вектор
T = [0.2, -0.1].     # целевой вектор
n = 0.3.             # норма обучения


def Q_error2(Y, m): # находит ошибку для скрытого слоя
    return (T[m] - Y[m]) * (1 - Y[m] ** 2) / 2

def Q_error1(w2, q2, out2):  # находит ошибку для выходного слоя
    return sum([w_i * q_i for w_i, q_i in zip(w2, q2)]) * (1 - out2 ** 2) / 2

def DeltaW(x, q):  # находит величину, на которую изменятся Wi
    return n * x * q

def Net(x, w):  # находит значение сетевого входа НС
    return sum([w_i * x_i for w_i, x_i in zip(w[1:], x)]) + w[0]

def Out(net):     # находит выходной сигнал
    return (1 - np.exp((-1) * net)) / (1 + np.exp((-1) * net))

def MeanSquareError(T, Y):   # находит расстояние Хэмминга между двумя векторами
    summa = 0
    for t_i, y_i in zip(T, Y):
        summa += (t_i - y_i) ** 2
    return summa ** 0.5

def Learning(N, J, M):  # обучение НС методом обратного распространения

    X2 = [0. for j in range(J)]
    Y = [0. for m in range(M)]

    w1 = np.array([[random.uniform(0., 0.9) for i in range(N+1)]  for j in range(J)])
    w2 = np.array([[0.] * (J+1) for m in range(M)])

    q1 = [0. for j in range(J)]
    q2 = [0. for m in range(M)]

    net1 = [0. for j in range(J)]
    net2 = [0. for m in range(M)]
```

```
    E = [MeanSquareError(T, Y)]
    era = 0

    while (E[len(E) - 1] > 10 ** (-3)):

        print("\nera = ", np.round(era, 3))

        # первый этап для скрытого слоя
        for j in range(J):
            net1[j] = Net(X1, w1[j])
            X2[j] = Out(net1[j])

        # первый этап для выходного слоя
        for m in range(M):
            net2[m] = Net(X2, w2[m])
            Y[m] = Out(net2[m])

        # второй этап для выходного слоя
        for m in range(M):
            q2[m] = Q_error2(Y, m)

        # второй этап для скрытого слоя
        for j in range(J):
            q1[j] = Q_error1(q2, w2[:, j+1], X2[j])

        # третий этап для скрытого слоя
            for i in range(N):
                w1[j][i] += DeltaW(X1[i], q1[j])

        # третий этап для выходного слоя
        for m in range(M):
            for j in range(J):
                w2[m][j + 1] += DeltaW(X2[j], q2[m])
            w2[m][0] += DeltaW(1, q2[m])

        E.append(MeanSquareError(T, Y))
        era += 1

        print("w1 : \n", np.round(w1, 4))
        print("w2 : \n", np.round(w2, 4))
        print("Y : ", np.around(Y, 3))
        print("e = ", np.round(E[len(E) - 1], 3))

    return E[1:]


def Graph(E):  # строит график зависимости ошибки
    plt.plot(E, 'bo-', linewidth=1, markersize=2)
    plt.title("E(k)")
    plt.xlabel("k")
    plt.ylabel('E')
    plt.grid(True)
    plt.show()

if __name__ == "__main__":

    E = Learning(2,1,2)
    Graph(E)
```