Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

ОТЧЕТ

по дисциплине «Интеллектуальные системы»

на тему «Нейронная сеть, меняющая два числа местами»

Выполнили студенты команды №1

Квашнина А., Мирошник С., Киреев Д.,
Ковалева Н., Маслова Е., Урбановский Е.

Томск—2020

1. Цель работы

Написать нейронную сеть, получающую два числа на входе, которая должна поменять их местами.

2. Постановку задачи.

Для формулирования верной постановки задачи необходимо:

А) Выбрать подход к обучению НС.

Б) Выбрать алгоритм обучения.

3. Метод решения задачи.

Для решения поставленной задачи необходимо выполнить следующие действия:

1. Создать многослойную НС.
2. Создать тестовые данные для обучения с учителем.
3. Протестировать НС различной структуры.
4. Выбрать лучшую.

4. Структурную схему алгоритма.

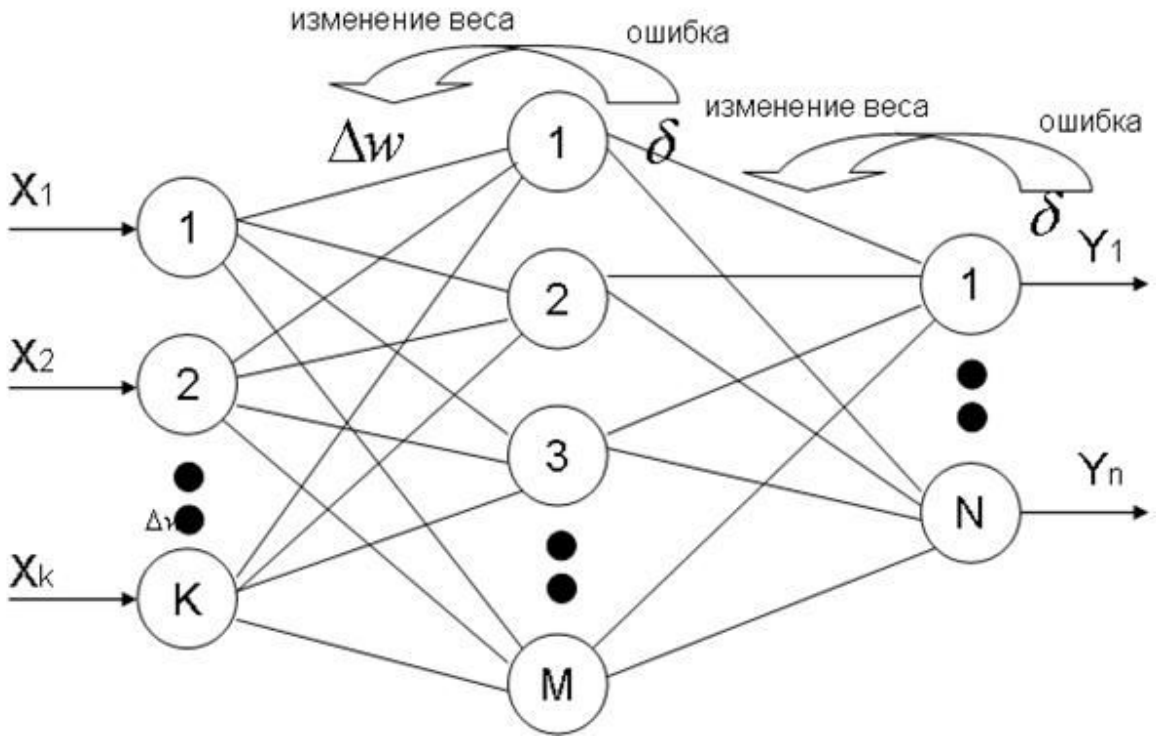

Рис 1. Схема работы алгоритма обратного распространения ошибки

5. Листинг программы.

см. Приложение

6. Результаты работы генетического алгоритмы.

По итогу обучения сети удалось выйти на результат с точностью около 0.05.

```
net.feedforward(np.array([[0.1, 0.9],
                          [0.15, 0.21],
                          [0.65, 0.45],
                          [0.97, 0.72]]))
```

```
array([[0.8739431 , 0.13773251],
       [0.19907859, 0.14580485],
       [0.43939136, 0.69699563],
       [0.73081566, 0.90833274]])
```

7. Выводы.

Результат получился неидеальный, но обучить НС лучше не вышло.

ПРИЛОЖЕНИЕ

```
import numpy as np

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

def d_sigmoid(x):
    f = sigmoid(x)
    return f*(1-f)


X_init = [[np.random.random(), np.random.random()] for i in range(5000)]
y_true = []
for i in range(5000):
    y_true.append(X_init[i].copy())
    X_init[i].reverse()
X_init = np.array(X_init)
y_true = np.array(y_true)
```

```
class OurNeuralNetwork:

    def __init__(self):

        self.hide_1 = Neuron_layer(2*np.random.random((2,4)) - 1)
        self.hide_2 = Neuron_layer(2*np.random.random((4,4)) - 1)
        self.out = Neuron_layer(2*np.random.random((4,2)) - 1)

    def feedforward(self, x):
        out_h1 = self.hide_1.feedforward(x)
        out_h2 = self.hide_2.feedforward(np.array(out_h1))
        out_o = self.out.feedforward(np.array(out_h2))

        return out_o
```

```
class Neuron_layer:
    def __init__(self, weights):
        self.weights = weights
        self.prev = weights
    def feedforward(self, inputs):
        # Вводные данные о весе, добавление смещения
        # и последующее использование функции активации

        self.prev = np.dot(inputs, self.weights )
        return sigmoid(self.prev)
```

```
net = OurNeuralNetwork()

for j in range(4000):
    error =np.array([[0,0]])
    flag = 1
    for X, y in zip(X_init, y_true):

        y_real = net.feedforward(np.array([X]))

        error = np.append(error, (y - y_real), axis=0)
        if flag:
            error = np.delete(error, [0,0], axis=0)
            flag = 0

        delta = error[-1]*d_sigmoid(net.out.prev)
        h2_error = delta.dot(net.out.weights.T)
        h2_delta = h2_error * d_sigmoid(net.hide_2.prev)
        h1_error = h2_delta.dot(net.hide_2.weights.T)
        h1_delta = h1_error * d_sigmoid(net.hide_1.prev)

        net.out.weights += 0.6*net.hide_2.prev.T.dot(delta)
        net.hide_2.weights += 0.3*net.hide_1.prev.T.dot(h2_delta)
        net.hide_1.weights += 0.3*np.array([X]).T.dot(h1_delta)
```