МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Уральский государственный экономический университет»

В.П. Часовских

Интеллектуальные технологии и кибербезопасность цифрового предприятья

38.04.05 – бизнес-информатика направленность интеллектуальное управление цифровым предприятием»

Лабораторная работа № 3

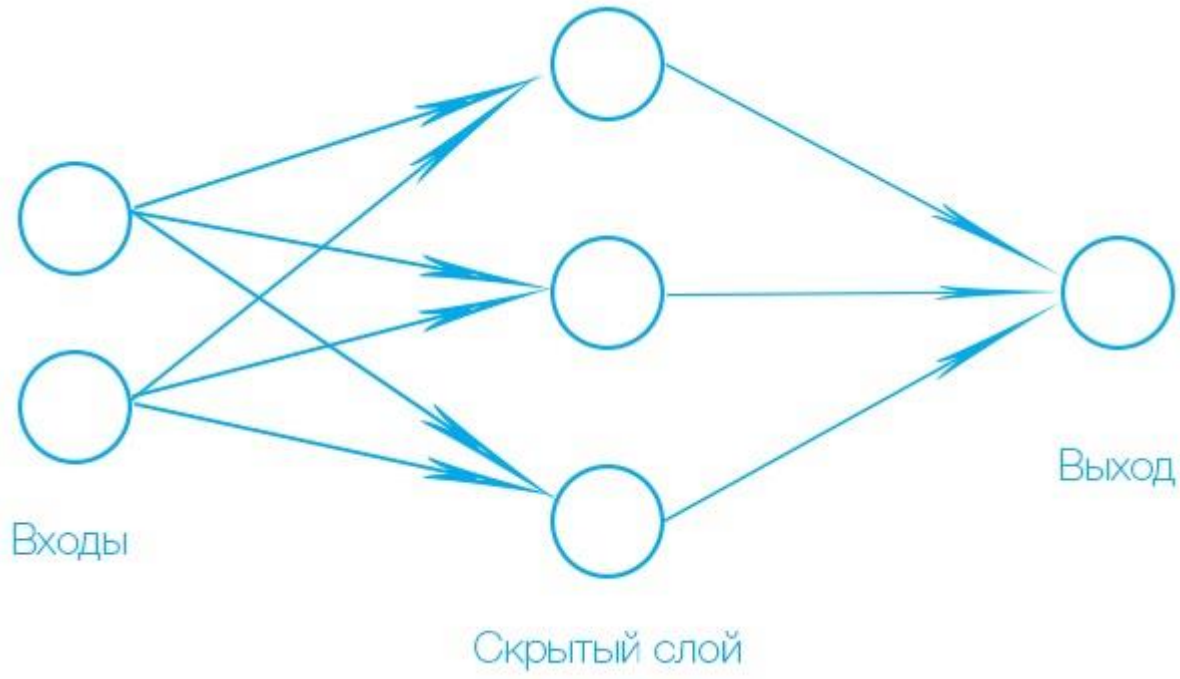
Нейронные сети

Екатеринбург 2022

Введение

Рассмотрим, как написать простую нейросеть на Python в среде Visual Studio (VS). Для более глубокого понимания проблематики приводятся примеры различных версий кода — от простого из 7 строк до более профессионального.

Любая искусственная нейронная сеть состоит из слоёв. Первый слой — входной, последний — выходной. Любой слой, расположенный между входным и выходным — скрытый. Количество слоев и нейронов в них может быть разное.



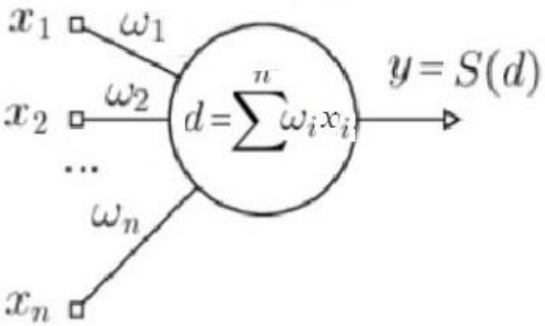
Во входном слое расположены нейроны, которые принимают сигнал, но не обрабатывают его.

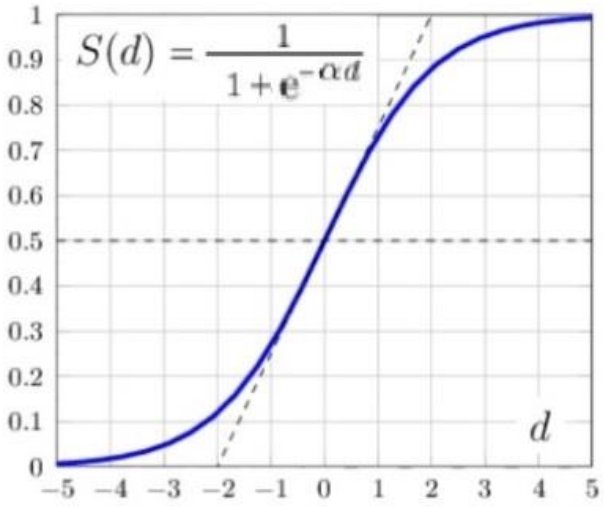
Стрелочки, передающие сигналы — синапсы. Они умножают входной сигнал *xi* на синаптический вес w_i. В каждом из нейронов определяется сумма значений входящих сигналов

$d = w_0 + w_1 x_1 + \ldots + wn\ xn$

где w_0 — параметр смещения.

Результат *d* приводится к диапазону [0 … 1] при помощи функции активации $y=S(d)$ (нелинейной сигмоидальной функции).





alpha – параметр наклона сигмоидальной функции S(d). Чем больше этот параметр, тем круче функция (угол касательной в точке перегиба функции будет больше).

Обучить нейронную сеть — значит, сообщить ей, чего мы от нее добиваемся. Допустим, мы хотим, чтобы для нашей сети при сигналах на входе 0 и 1 на выходе была 1.

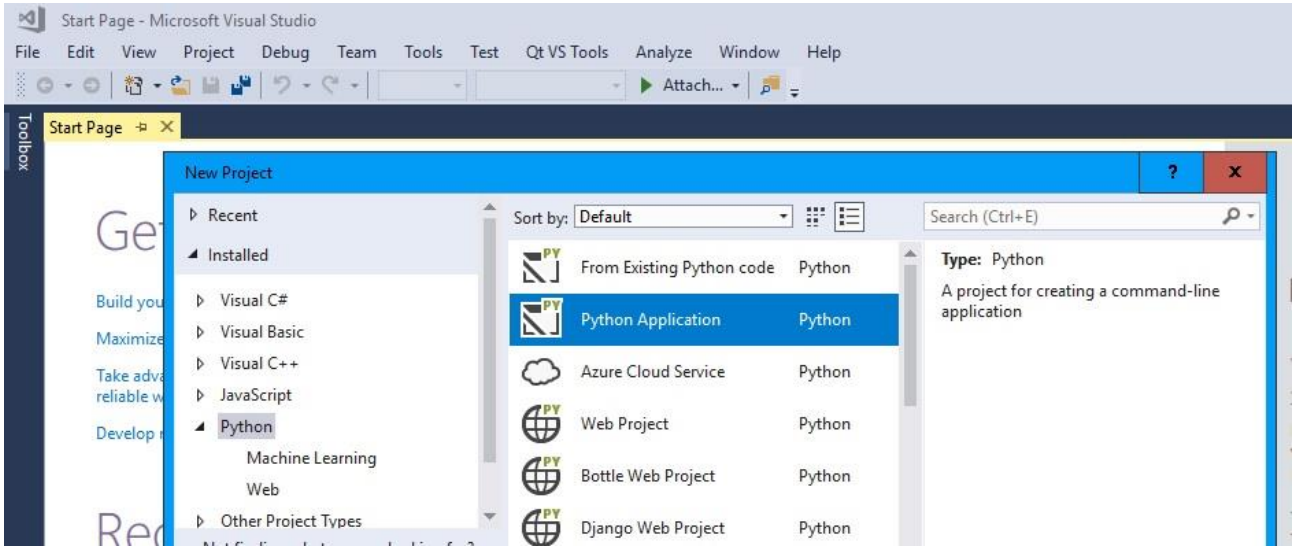
В общем виде алгоритм обучения с учителем будет выглядеть следующим образом:

1. Инициализировать синаптические веса маленькими случайными значениями.
2. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
3. Вычислить выход сети.
4. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
5. Подкорректировать веса сети для минимизации ошибки.
6. Повторять шаги с 2 по 5 для каждой пары обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.
7.

Конкретный вид математических операций, выполняемых на этапе 5, определяет разновидность алгоритма обучения. Например, для однослойных сетей применяют простейший алгоритм, основанный на т. н. дельта-правиле.

Создание нейронной сети в Visual Studio (версия кода 1)

Запускаем редактор VS и создаем проект (File>New>Project>Python>Python Application)



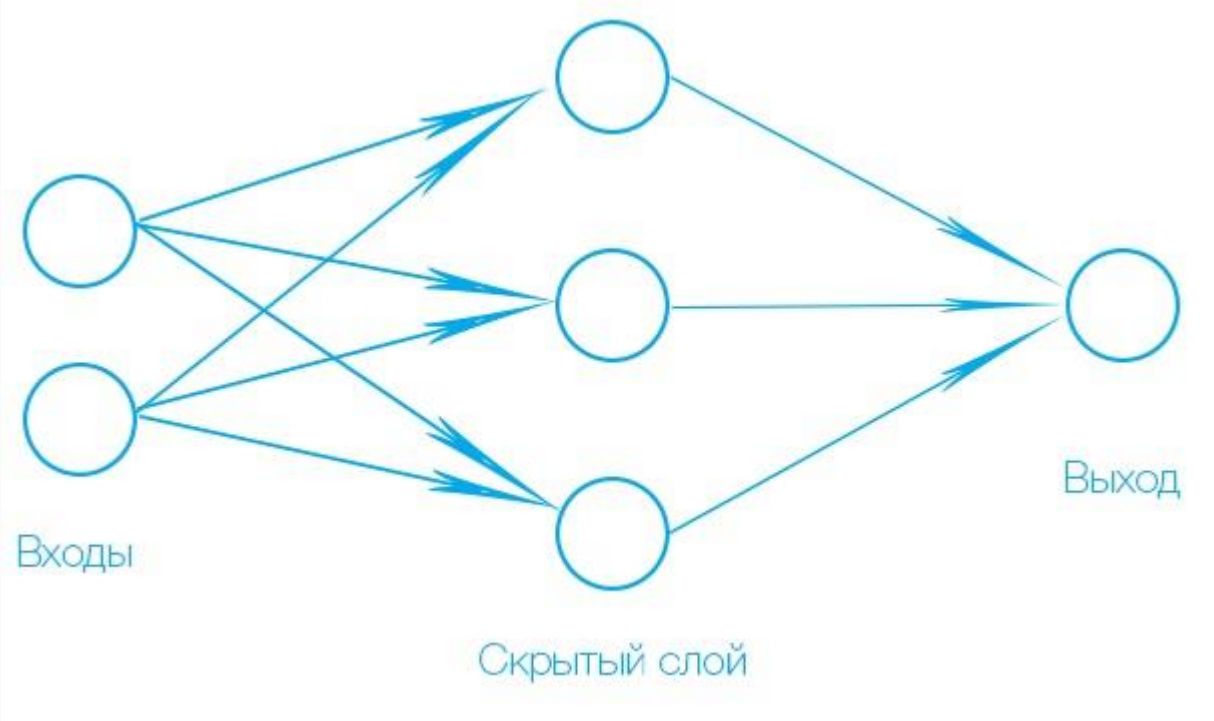
Вставляем в исходный файл (*.py*) код:

```
import numpy as np
# обучающая выборка входных и выходных данных
X = np.array([[3,5],[5,1],[10,2]])
y = np.array([[75, 82, 93]]).T   # T-транспонирование матрицы
# нормализация данных - делим на максимальное из них число
X = X / np.amax(X, axis = 0)
y = y / 100
# инициализация случайных весов (от -1 до +1) синапсов
np.random.seed(1)
synapses_hidden = 2 * np.random.random((2,3)) - 1  # 2*3 - hidden слой
synapses_output = 2 * np.random.random((3,1)) - 1  # 3*1 - output слой
# обучение сети - цикл из 10000 повторений
for j in range(10000):
    # Входной слой ( 2 входа )
    l0 = X
    # Скрытый слой ( 3 скрытых нейрона )
    l1 = 1 / (1 + np.exp(-(l0.dot(synapses_hidden))))
    # Выходной слой ( 1 выходной нейрон )
    l2 = 1 / (1 + np.exp(-(l1.dot(synapses_output))))
    # вычисляем ошибку (используем дельта-правило)
    l2_delta = (y - l2) * (l2 * (1 - l2))
    # получаем ошибку на скрытом слое (используем дельта-правило)
    l1_delta = l2_delta.dot(synapses_output.T) * (l1 * (1 - l1))
    # корректируем веса от скрытых нейронов к выходу
    synapses_output += l1.T.dot(l2_delta)
    # корректируем веса от входов к скрытым нейронам
    synapses_hidden += l0.T.dot(l1_delta)
# Печать сигналов на выходе после последнего цикла обучения
# Сигналы на выходе умножаем на коэффициент нормализации (100)
```

```
print(l2 * 100)
```

Обучается сеть с 2-я входами, с 3-я нейронами на скрытом слое и одним нейроном на выходе

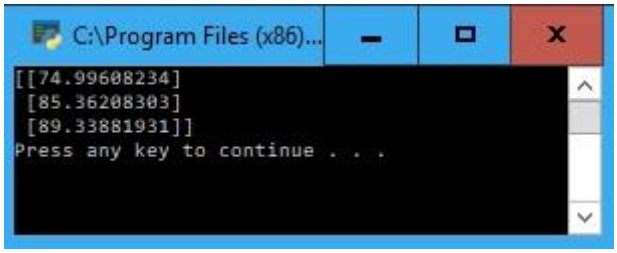
.



Обучающая выборка для обучения из 3-х примеров:

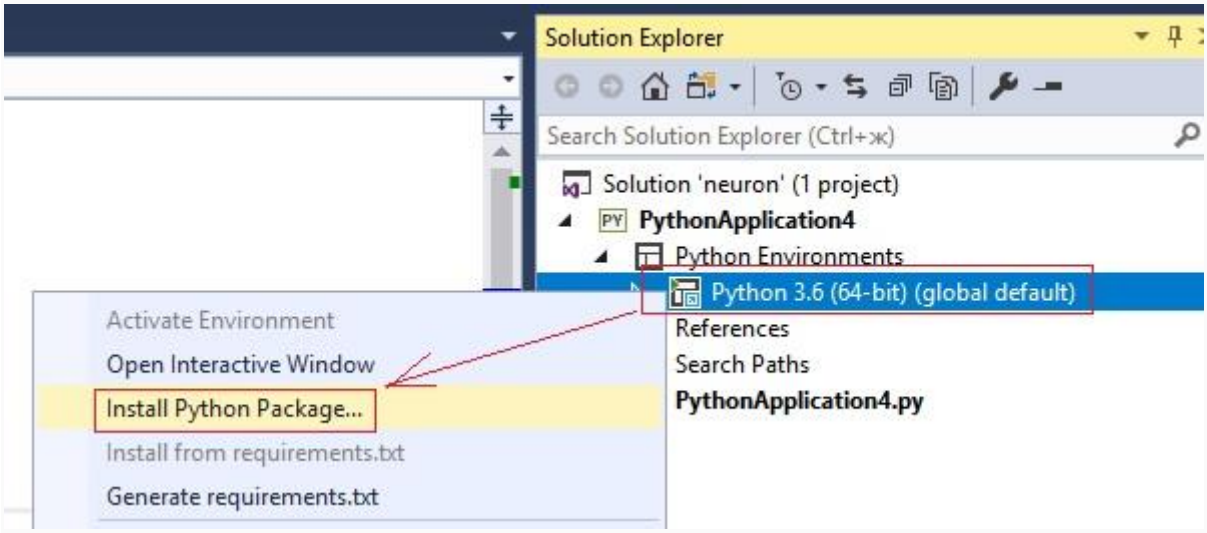
- *На входе сигналы: [3,5],[5,1],[10,2];*
- *На выходе сигналы: 75,82,93;*

Запускаем приложение, получаем после обучения сигналы на выходе, близкие к ожидаемым (заданным):

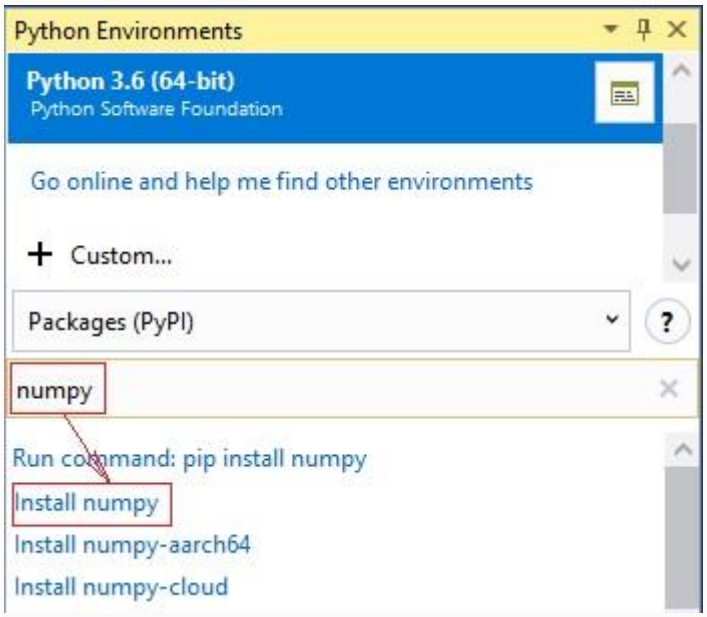


Если отладчик подчеркивает 1-й рядок кода, значит, у Вас еще не инсталлирован пакет (package) ***numpy****. Его можно инсталлировать не выходя из проекта (Шаг 5. Установка пакетов в окружении Python). Для*

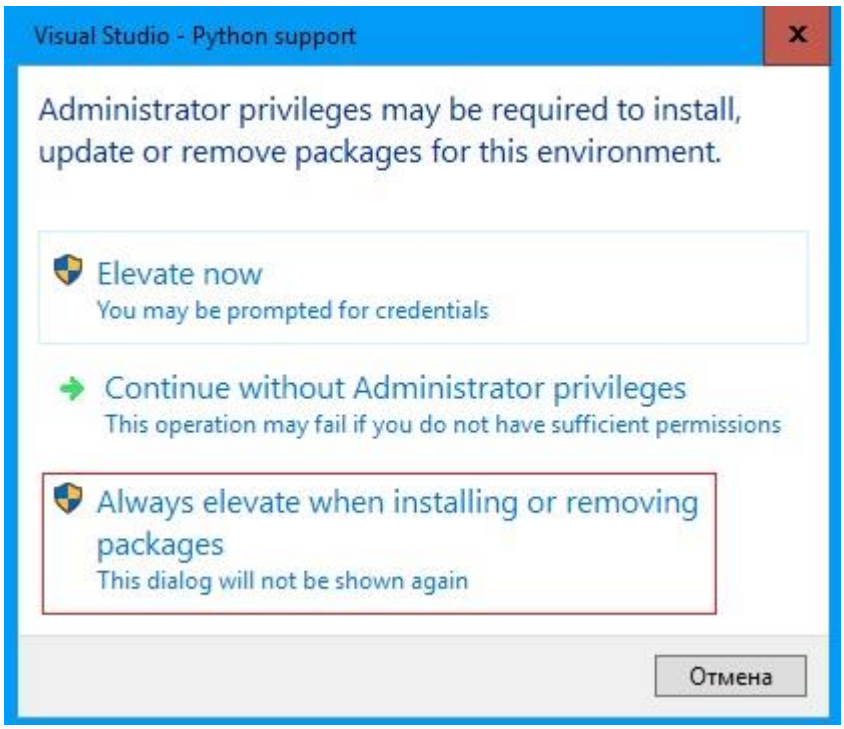
этого открываете контекстное меню в Solution Explore и выбираете в нем «Install Python Package»:



Откроется окно «Python Environments». В поисковом окошке набираете «numpy» и, затем, запускаете «Install numpy».

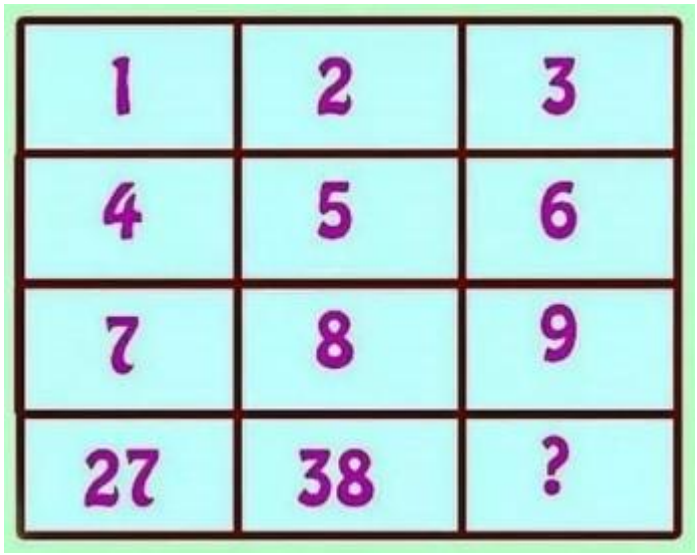


Откроется окно, в котором выбираете права администратора:



Инсталляция длится не более 5 минут.

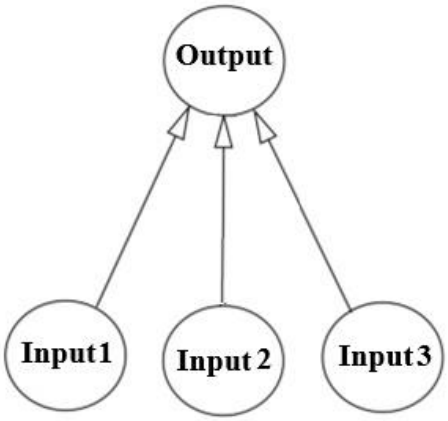
Математическая задача:



Какая цифра должна быть? Проверьте себя и поставьте эту задачу для нейронной сети.

Простая нейронная сеть (версия кода 2)

Постановка задачи. Обучить однослойную нейронную сеть с 3-х входами и с одним выходом на основе тренировочной выборки из 4-х примеров. После обучения системы определить результат на выходе для тестового примера.

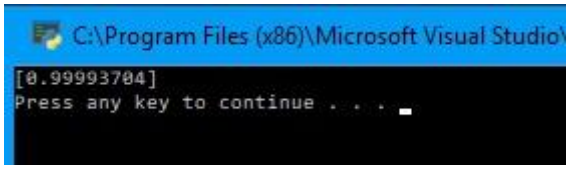


	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0
New situation	1	0	0	?

Вставляем в исходный файл следующий код:

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in range(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) *
output * (1 - output))
print (1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights)))))
```

Запускаем приложение, получаем следующий результат:

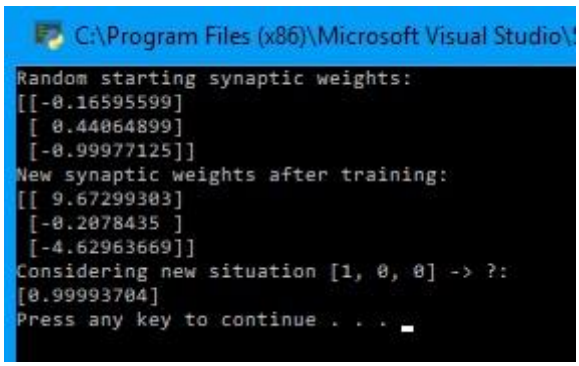


Усложняем приложение. Вставляем в исходный файл следующий код:

```
from numpy import exp, array, random, dot

class NeuralNetwork():
    def __init__(self):
        random.seed(1)
        self.synaptic_weights = 2 * random.random((3, 1)) - 1
    def __sigmoid(self, x):
        return 1 / (1 + exp(-x))
```

```
    def __sigmoid_derivative(self, x):
        return x * (1 - x)

    def train(self, training_set_inputs, training_set_outputs,
number_of_training_iterations):
        for iteration in range(number_of_training_iterations):
            output = self.think(training_set_inputs)
            error = training_set_outputs - output
            adjustment = dot(training_set_inputs.T, error *
self.__sigmoid_derivative(output))
            self.synaptic_weights += adjustment

    def think(self, inputs):
        return self.__sigmoid(dot(inputs, self.synaptic_weights))

if __name__ == "__main__":

    #Intialise a single neuron neural network.
    neural_network = NeuralNetwork()

    print ("Random starting synaptic weights: ")
    print (neural_network.synaptic_weights)

    training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
    training_set_outputs = array([[0, 1, 1, 0]]).T

    # Train the neural network using a training set. Do it 10,000 times.
    neural_network.train(training_set_inputs, training_set_outputs, 10000)

    print ("New synaptic weights after training: ")
    print (neural_network.synaptic_weights)

    # Test the neural network with a new situation.
    print ("Considering new situation [1, 0, 0] -> ?: ")
    print (neural_network.think(array([1, 0, 0])))
```

Запускаем приложение, получаем следующий результат:



Простая нейронная сеть (версия кода 3)

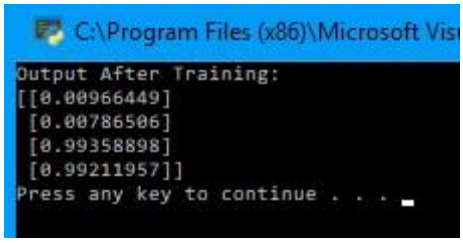
Ниже рассмотрены коды приложений для 2-х и 3-х уровневых нейронных сетей. .

Код к 2-х уровневой нейронной сети:

```
import numpy as np

# sigmoid function
def nonlin(x,deriv=False):
    if(deriv==True):
        return x*(1-x)
    return 1/(1+np.exp(-x))

# input dataset
X = np.array([ [0,0,1],
               [0,1,1],
               [1,0,1],
               [1,1,1] ])
# output dataset
y = np.array([[0,0,1,1]]).T

np.random.seed(1)
syn0 = 2*np.random.random((3,1)) - 1

for i in range(10000):
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    l1_error = y - l1
    l1_delta = l1_error * nonlin(l1,True)
    # update weights
    syn0 += np.dot(l0.T,l1_delta)
```

```
print ("Output After Training:")
print (l1)
```

Запускаем приложение, получаем следующий результат:



Код к 3-х уровневой нейронной сети:

```
import numpy as np

def nonlin(x,deriv=False):
    if(deriv==True):
        return x*(1-x)
    return 1/(1+np.exp(-x))

X = np.array([[0,0,1],
              [0,1,1],
              [1,0,1],
              [1,1,1]])
y = np.array([[0],
              [1],
              [1],
              [0]])

np.random.seed(1)
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1

for j in range(60000):
    l0 = X
    l1 = nonlin(np.dot(l0,syn0))
    l2 = nonlin(np.dot(l1,syn1))
    l2_error = y - l2
    if (j% 10000) == 0:
        print ("Error:" + str(np.mean(np.abs(l2_error))))
    l2_delta = l2_error*nonlin(l2,deriv=True)
    l1_error = l2_delta.dot(syn1.T)
    l1_delta = l1_error * nonlin(l1,deriv=True)
```

```
    syn1 += l1.T.dot(l2_delta)
    syn0 += l0.T.dot(l1_delta)
```

Запускаем приложение, получаем следующий результат: