Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 13

«СОГЛАСОВАНИЕ КРИПТОГРАФИЧЕСКИХ КЛЮЧЕЙ НА ОСНОВЕ ТЕХНОЛОГИЙ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ»

Выполнила:

студентка 3 курса 2 группы

Шастовская М. С.

Вариант: 12

Преподаватель:

Сазонова Д.В.

Минск 2023

**Цель:** изучение основ построения и функционирования искусственных нейронных сетей (ИНС), а также использования ИНС в криптографии; приобретение практических навыков программной реализации алгоритма согласования ключевой информации на основе технологии ИНС.

**Задачи:**

* Закрепить теоретические знания по основам построения и функционирования ИНС.
* Усвоить особенности построения, основные алгоритмы взаимного обучения двух связанных нейронных сетей на основе модели ТРМ.
* Разработать приложение для реализации модели ТРМ с целью согласования двумя сторонами совместного тайного ключа.
* Познакомиться с методиками оценки криптостойкости алгоритма на основе ТРМ.
* Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**1 Теоретические сведения**

Как известно, мозг человека состоит, главным образом, из большого числа соединенных друг с другом элементарных нервных клеток, называемых нейронами.

Спроектирован искусственный нейрон, называемый также персептроном или перцептроном (англ. perceptron), симулирующий работу биологического нейрона.

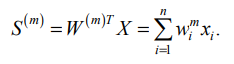
Эта модель содержит главные элементы биологического нейрона. Вот их эквиваленты: входные значения – дендриты, весовые коэффициенты – синапсы, суммирующий блок – ядро, функция активации – основание аксона, а выходное значение – это аксон.

Математически работу персептрона можно описать так:

S wx = = ∑ w I x i,

где xi и wi – соответственно i-e входной сигнал и весовой коэффициент нейрона.

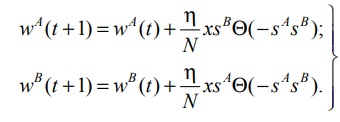
Выходной сигнал m-го нейрона может быть подсчитан следующим образом:



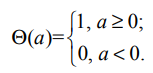
Пусть даны два персептрона, каждый из которых получает на вход случайно выбранный вектор входных значений x. Оба преобразовывают свои внутренние векторы весов в соответствии с принятым правилом обучения. При этом также учитываются собственные выходные величины s. Вычисляется эта величина следующим образом:



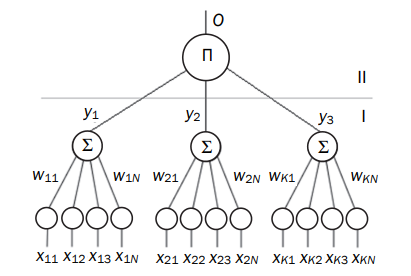
Начальное состояние векторов весов wA и wB в обеих сетях случайное. Затем на каждом шаге обучения на вход обеих сетей подается один (случайно сгенерированный) входной вектор x, с помощью которого обе сети вычисляют состояния своих выходов (sA и sB). На каждом шаге обучения весовые коэффициенты подвергаются следующему преобразованию:



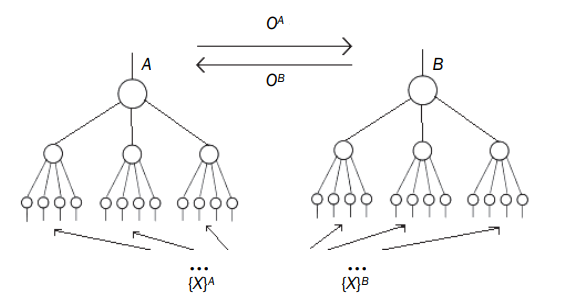
В соответствии с вышеуказанными формулами функция Θ определяется следующим образом:



Архитектуру сети составляют K классических персептронов, принадлежащих внутреннему уровню; внешний уровень (обозначен римской цифрой II) объединяет выходы персептронов внутреннего уровня (I):



Общая схема процесса синхронизации двух сетей в графическом виде представлена. Кратко проанализируем особенности процесса синхронизации сетей на основе архитектуры ТРМ.



Самый простой способ атаки на синхронизирующиеся сети А и В (со стороны С) основывается на том, что оппонент может знать архитектуру обеих сетей, а также иметь всю информацию, которой обмениваются между собой сети.

2 Практическая часть

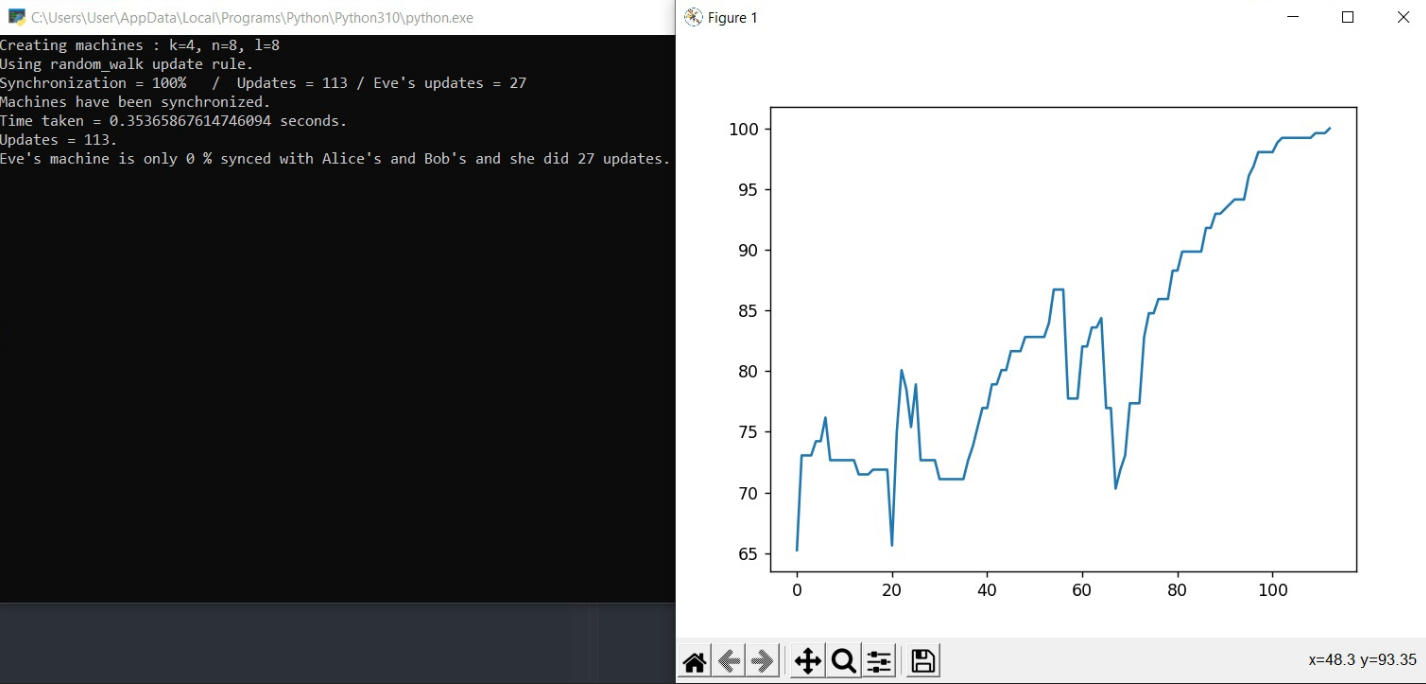


Рисунок 2.1 – Результат разработки приложения

На скриншоте представлено приложение, с помощью которого было реализована модель TCP – эмулятор процесса синхронизации весовых коэффициентов двух ИНС.

Основные функции программы представлены на рисунке 2.2.

\_\_author\_ = "Fariz Rahman"

from machine import Machine

import numpy as np

import time

import sys

#Machine parameters

k = 100

n = 10

l = 10

#Update rule

update\_rules = ['hebbian', 'anti\_hebbian', 'random\_walk']

update\_rule = update\_rules[2]

#Create 3 machines : Alice, Bob and Eve. Eve will try to intercept the communication between

#Alice and Bob.

print("Creating machines : k=" + str(k) + ", n=" + str(n) + ", l=" + str(n))

print("Using " + update\_rule + " update rule.")

Alice = Machine(k, n, l)

Bob = Machine(k, n, l)

Eve = Machine(k, n, l)

#Random number generator

def random():

return np.random.randint(-l, l + 1, [k, n])

#Function to evaluate the synchronization score between two machines.

def sync\_score(m1, m2):

return 1.0 - np.average(1.0 \* np.abs(m1.W - m2.W)/(2 \* l))

#Synchronize weights

sync = False # Flag to check if weights are sync

nb\_updates = 0 # Update counter

nb\_eve\_updates = 0 # To count the number of times eve updated

start\_time = time.time() # Start time

sync\_history = [] # to store the sync score after every update

while(not sync):

X = random() # Create random vector of dimensions [k, n]

tauA = Alice(X) # Get output from Alice

tauB = Bob(X) # Get output from Bob

tauE = Eve(X) # Get output from Eve

Alice.update(tauB, update\_rule) # Update Alice with Bob's output

Bob.update(tauA, update\_rule) # Update Bob with Alice's output

#Eve would update only if tauA = tauB = tauE

if tauA == tauB == tauE:

Eve.update(tauA, update\_rule)

nb\_eve\_updates += 1

nb\_updates += 1

score = 100 \* sync\_score(Alice, Bob) # Calculate the synchronization of the 2 machines

sync\_history.append(score) # Add sync score to history, so that we can plot a graph later.

sys.stdout.write('\r' + "Synchronization = " + str(int(score)) + "% / Updates = " + str(nb\_updates) + " / Eve's updates = " + str(nb\_eve\_updates))

if score == 100: # If synchronization score is 100%, set sync flag = True

sync = True

end\_time = time.time()

time\_taken = end\_time - start\_time # Calculate time taken

#Print results

print ('\nMachines have been synchronized.')

print ('Time taken = ' + str(time\_taken)+ " seconds.")

print ('Updates = ' + str(nb\_updates) + ".")

#See if Eve got what she wanted:

eve\_score = 100 \* int(sync\_score(Alice, Eve))

if eve\_score > 100:

print("Oops! Nosy Eve synced her machine with Alice's and Bob's !")

else:

print("Eve's machine is only " + str(eve\_score) + " % " + "synced with Alice's and Bob's and she did " + str(nb\_eve\_updates) + " updates.")

#Plot graph

import matplotlib.pyplot as mpl

mpl.plot(sync\_history)

mpl.show()

Вывод

В ходе выполнения лабораторной работы были изучены основ построения и функционирования искусственных нейронных сетей.