Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №16»**

«Согласование криптографических ключей на основе технологий искусственных нейронных сетей»

**Выполнил:** студент 3 курса

4 группы специальности ПОИТ

Супрунюк Евгений Андреевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования Python, в котором реализована реализация синхронизация сетей с помощью алгоритма обучения Хэбба.

1. **Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, которое реализует реализацию синхронизацию сетей с помощью алгоритма обучения Хэбба. На листинге 2.1 представлены функции, реализующие данную функциональность.

|  |
| --- |
| from update\_rules import hebbian, anti\_hebbian, random\_walk  import numpy as np  class Machine:      def \_\_init\_\_(self, k=3, n=4, l=6):          self.k = k          self.n = n          self.l = l          self.W = np.random.randint(-l, l + 1, [k, n])      def get\_output(self, X):          k = self.k          n = self.n          W = self.W          X = X.reshape([k, n])          # Compute inner activation sigma Dimension:[K]          sigma = np.sign(np.sum(X \* W, axis=1))          tau = np.prod(sigma)  # The final output          self.X = X          self.sigma = sigma          self.tau = tau          return tau      def \_\_call\_\_(self, X):          return self.get\_output(X)      def update(self, tau2, update\_rule='hebbian'):          X = self.X          tau1 = self.tau          sigma = self.sigma          W = self.W          l = self.l          if (tau1 == tau2):              if update\_rule == 'hebbian':                  hebbian(W, X, sigma, tau1, tau2, l)              elif update\_rule == 'anti\_hebbian':                  anti\_hebbian(W, X, sigma, tau1, tau2, l)              elif update\_rule == 'random\_walk':                  random\_walk(W, X, sigma, tau1, tau2, l)              else:                  raise Exception("Invalid update rule. Valid update rules are: " +                                  "\'hebbian\', \'anti\_hebbian\' and \'random\_walk\'.")  import numpy as np  def theta(t1, t2):      return 1 if t1 == t2 else 0  def hebbian(W, X, sigma, tau1, tau2, l):      k, n = W.shape      for (i, j), \_ in np.ndenumerate(W):          W[i, j] += X[i, j] \* tau1 \* theta(sigma[i], tau1) \* theta(tau1, tau2)          W[i, j] = np.clip(W[i, j] , -l, l)  def anti\_hebbian(W, X, sigma, tau1, tau2, l):      k, n = W.shape      for (i, j), \_ in np.ndenumerate(W):          W[i, j] -= X[i, j] \* tau1 \* theta(sigma[i], tau1) \* theta(tau1, tau2)          W[i, j] = np.clip(W[i, j], -l, l)  def random\_walk(W, X, sigma, tau1, tau2, l):      k, n = W.shape      for (i, j), \_ in np.ndenumerate(W):          W[i, j] += X[i, j] \* theta(sigma[i], tau1) \* theta(tau1, tau2)          W[i, j] = np.clip(W[i, j] , -l, l)  import matplotlib.pyplot as mpl  import sys  import time  import numpy as np  from machine import Machine  # Machine parameters  k = 5  n = 8  l = 3  # Update rule  update\_rules = ['hebbian', 'anti\_hebbian', 'random\_walk']  update\_rule = update\_rules[0]  # 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      # Calculate the synchronization of the 2 machines      score = 100 \* sync\_score(Alice, Bob)      # Add sync score to history, so that we can plot a graph later.      sync\_history.append(score)      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  mpl.plot(sync\_history)  mpl.show() |

Листинг 2.1 –код программы, реализующие заданную ранее функциональность

**3. Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение. Рисунок 3.1 показывают требуемые в данной лабораторной работе результаты.

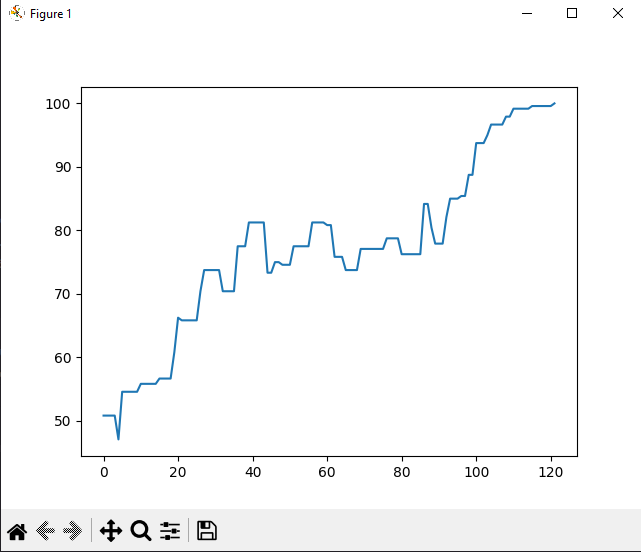


Рисунок 3.1 – Результат работы

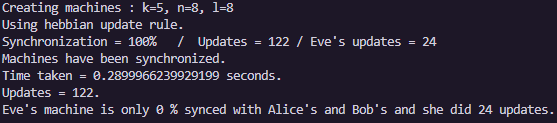


Рисунок 3.2 – Результат работы

**Вывод**

В ходе лабораторной работы было разработано приложение для реализации синхронизация сетей с помощью алгоритма обучения Хэбба.