Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: «Нелинейные ИНС в задачах прогнозирования»

Вариант 11

Выполнил:

Студент 2-го курса

Группы ПО-7

Лобан К.Ю.

Проверил:

Крощенко А.А.

Брест 2021

**Цель работы**: Изучить обучение и функционирование нелинейной ИНС при решении задач распознавания образов.

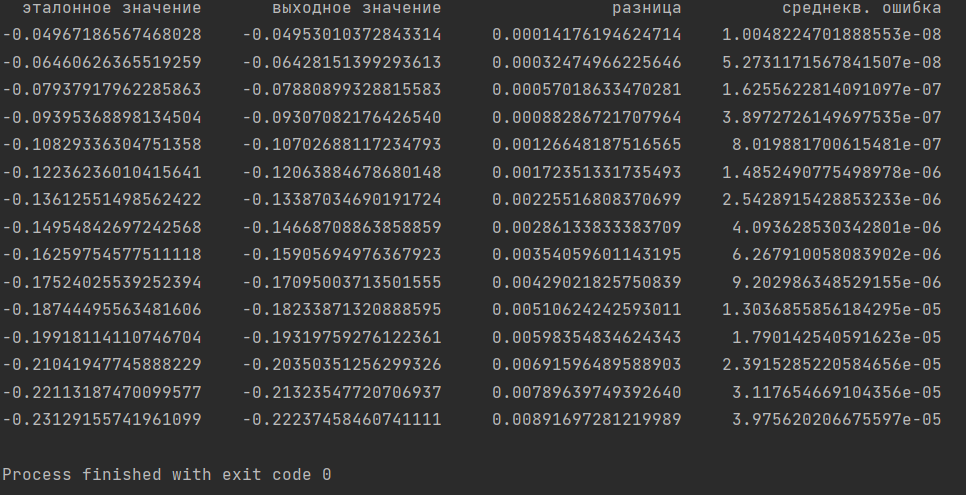
**Задание**

Спрогнозировать нелинейный временной ряд, применяя параметры лабораторной работы №3. При этом необходимо использовать алгоритм обучения многослойной ИНС с адаптивным шагом.

**Код программы:**

import numpy as np  
  
  
def predict\_xe(begin, lenght, count, step):  
 *"""Наборы для предсказания"""* base\_array = np.arange(count + lenght) # Базовый массив из целых чисел (0, 1 ...), из которого будут браться срезы  
 x = np.zeros(shape=(count, lenght)) # Шаблон из нулей длин (count, lenght)  
 for i in range(count):  
 x[i] = base\_array[i: lenght + i] # Срез i-того ряда  
 e = base\_array[lenght: lenght + count] # Массив эталонов (count,)  
  
 x = x \* step + begin # Умножение на шаг и прибавление начала  
 e = e \* step + begin  
 # return x, e  
 return function(x), function(e) # Применение функции  
  
  
class NN:  
 *"""Класс нейронки"""* def \_\_init\_\_(self):  
 *"""Нейронная сеть с количеством нейронов: 8-3-1"""* self.wx = np.random.normal(-0.5, 0.5, (8, 3))  
 self.th = np.random.normal(-0.5, 0.5, 3)  
 self.wh = np.random.normal(-0.5, 0.5, (3, 1))  
 self.ty = np.random.normal(-0.5, 0.5, 1)  
 # Задание весов и порогов нормальным распределением нужных длин  
  
 def go(self, x):  
 *"""Прохождение всей нейронки"""* self.x = x # Входные параметры (1, 8)  
 self.sh = np.dot(self.x, self.wx) - self.th # Вектор сумм ((1, 8) x (8, 3) - (1, 3) = (1, 3))  
 self.h = 1.0 / (1.0 + np.exp(-self.sh)) # Активация (1, 3)  
 self.sy = np.dot(self.h, self.wh) - self.ty # Вектор сумм ((1, 3) x (3, 1) - (1, 1) = (1, 1))  
 self.y = self.sy # Активация и получение выхода нейронки (1, 1)  
 return self.y # (1, 1)  
  
 def back\_propagation(self, error\_y):  
 *"""Обратное распространение ошибки с изменением весов, порога"""* error\_h = np.dot(error\_y, self.wh.transpose())  
 # Ошибка для скрытого слоя (((1,) \* (1,)) x (3, 1).T = (3,))  
  
 alpha = 1 / (1 + np.square(self.h).sum())  
  
 gamma\_y = alpha \* error\_y  
 self.wh -= np.dot(self.h.reshape(-1, 1), gamma\_y.reshape(1, -1))  
 self.ty += gamma\_y  
  
 alpha = (  
 4  
 \* (error\_h \*\* 2 \* self.h \* (1 - self.h)).sum()  
 / (1 + (self.x \*\* 2).sum())  
 / np.square(error\_h \* self.h \* (1 - self.h)).sum()  
 )  
  
 gamma\_h = alpha \* error\_h \* self.y \* (1.0 - self.y)  
 self.wx -= np.dot(self.x.reshape(-1, 1), gamma\_h.reshape(1, -1))  
 self.th += gamma\_h  
  
 def learn(self, x, e):  
 *"""Обучение наборами"""* square\_error = 0 # Среднеквадратичная ошибка  
 for i in range(len(e)):  
 y = self.go(x[i]) # Прогон выборки  
 error = y - e[i] # Ошибка нейронной сети  
 square\_error += error[0] \*\* 2 / 2 # Суммирование среднеквадратичной ошибки  
 self.back\_propagation(error) # Обратное распространение  
 return square\_error  
  
 def test(self, x, e):  
 *"""Тестирование наборами"""* print(" эталонное значение выходное значение разница среднекв. ошибка")  
 for i in range(len(e)):  
 y = self.go(x[i])[0] # Прохождение нейронки  
 delta = y - e[i] # Вычисление ошибки  
 square\_error = delta \*\* 2 / 2 # Подсчет среднеквадратичной ошибки  
 print(f"{e[i]: 19.17f}{y: 24.17f}{delta: 24.17f}{square\_error: 26}")  
  
  
def function(x):  
 *"""Полная функция для варианта 11"""* return 0.3 \* np.cos(0.5 \* x) + 0.05 \* np.sin(0.5 \* x)  
  
  
def main():  
 learn\_x, learn\_e = predict\_xe(0, 8, 30, 0.1) # Получение обучающей выборки  
 nn = NN()  
  
 for i in range(30\_000):  
 error = nn.learn(learn\_x, learn\_e)  
 print(f"Square error {i: 5}: {error: .8f}") # Обучение с выводом ошибки  
 if np.isnan(error):  
 nn = NN() # Если произойдет ошибка, нейронная сеть обновится  
 if error <= 1e-7:  
 break  
  
 test\_x, test\_e = predict\_xe(3, 8, 15, 0.1) # Получение тестовой выборки  
 nn.test(test\_x, test\_e) # Тестирование  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат:**

****

**Вывод:** я изучил нелинейные ИНС в задачах прогнозирования.