Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Рекуррентные нейронные сети»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

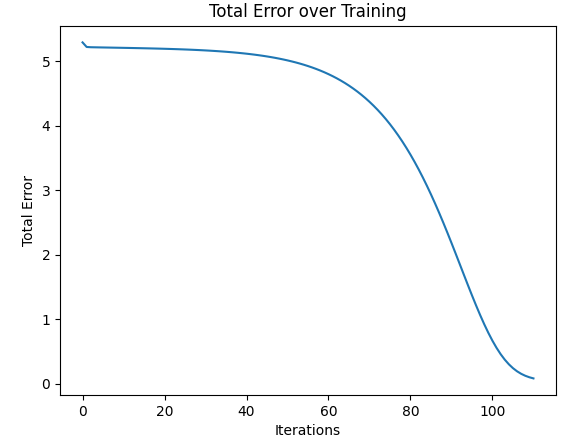
Туз И. С.

Брест 2024

**Цель:** реализовать и обучить рекуррентную нейронную сеть для прогнозирования периодической функции.

**Ход работы**

import numpy as np

import matplotlib.pyplot as plt

a, b, c, d = 0.3, 0.1, 0.06, 0.1

num\_time\_steps = 110

input\_size = 1

hidden\_size = 12

output\_size = 1

lr = 0.0073

start = 0

time\_steps = np.linspace(start, start+200, num\_time\_steps)

data = a \* np.cos(b \* time\_steps) + c \* np.sin(d \* time\_steps)

class RNN:

    def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

        self.hidden\_size = hidden\_size

        self.W\_xh = np.random.randn(hidden\_size, input\_size) \* 0.01

        self.W\_hh = np.random.randn(hidden\_size, hidden\_size) \* 0.01

        self.W\_hy = np.random.randn(output\_size, hidden\_size) \* 0.01

        self.b\_h = np.random.randn(hidden\_size, 1) \* 0.01

        self.b\_y = np.random.randn(output\_size, 1) \* 0.01

    def forward(self, x, h\_prev):

        h\_next = np.tanh(np.dot(self.W\_xh, x) + np.dot(self.W\_hh, h\_prev) + self.b\_h)

        y = np.dot(self.W\_hy, h\_next) + self.b\_y

        return y, h\_next

rnn = RNN(input\_size, hidden\_size, output\_size)

h\_prev = np.zeros((hidden\_size, 1))

errors = []

for iter in range(800):

    x = data[:-1].reshape(num\_time\_steps - 1, 1)

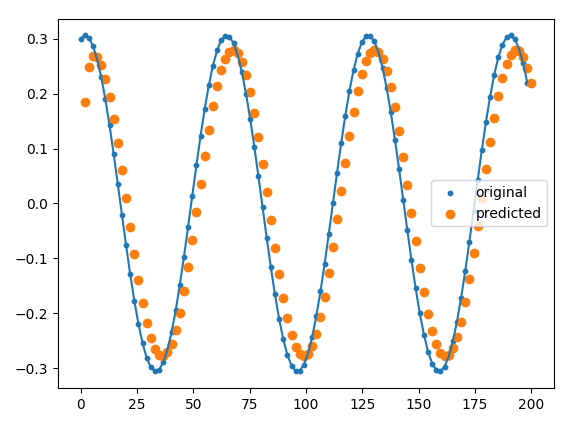
    y = data[1:].reshape(num\_time\_steps - 1, 1)

    dW\_xh, dW\_hh, dW\_hy = np.zeros\_like(rnn.W\_xh), np.zeros\_like(rnn.W\_hh), np.zeros\_like(rnn.W\_hy)

    db\_h, db\_y = np.zeros\_like(rnn.b\_h), np.zeros\_like(rnn.b\_y)

    dh\_next = np.zeros\_like(h\_prev)

    total\_error = 0



    for i in range(num\_time\_steps - 2, -1, -1):

        x\_t = x[i].reshape(1, 1)

        y\_t = y[i].reshape(1, 1)

        pred, h\_prev = rnn.forward(x\_t, h\_prev)

        dy = pred - y\_t

        total\_error += np.sum(dy\*\*2)

        dW\_hy += np.dot(dy, h\_prev.T)

        db\_y += dy

        dh = np.dot(rnn.W\_hy.T, dy) + dh\_next

        dh\_raw = (1 - h\_prev \* h\_prev) \* dh

        db\_h += dh\_raw

        dW\_xh += np.dot(dh\_raw, x\_t.T)

        dW\_hh += np.dot(dh\_raw, h\_prev.T)

        dh\_next = np.dot(rnn.W\_hh.T, dh\_raw)

    print(total\_error)

    errors.append(total\_error)

    for dparam in [dW\_xh, dW\_hh, dW\_hy, db\_h, db\_y]:

        np.clip(dparam, -7, 7, out=dparam)

    rnn.W\_xh -= lr \* dW\_xh

    rnn.W\_hh -= lr \* dW\_hh

    rnn.W\_hy -= lr \* dW\_hy

    rnn.b\_h -= lr \* db\_h

    rnn.b\_y -= lr \* db\_y

    if total\_error < 0.1:

        print(iter, total\_error)

        break

x = data[:-1].reshape(num\_time\_steps-1, 1)

preds = []

h\_prev = np.zeros((hidden\_size, 1))

for i in range(num\_time\_steps - 1):

    x\_t = x[i].reshape(1, 1)

    pred, h\_prev = rnn.forward(x\_t, h\_prev)

    preds.append(pred.ravel()[0])

**Вывод:** в ходе лабораторной работы я реализовал рекуррентную нейронную сеть для прогнозирования значений функции.