Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный университет”
Факультет электронно-информационных систем
Кафедра ИИТ

Отчет по лабораторной работе №2
По дисциплине “Методы и алгоритмы принятия решений”
Тема: “Адаптивный шаг обучения”
Вариант №7

Выполнил:
Кравцевич Г.А.
студент группы ПО-7

Проверил:
А. А. Крощенко,
ст.преп.кафедры ИИТ
20 сентября 2021 г.

Брест 2021

Цель работы:
Изучить обучение и функционирование линейной ИНС с применением адаптивного шага.

Задание:
Модифицировать программу из лабораторной №1, используя правило адаптивного шага обучения. Произвести исследование получившейся модели ИНС на задачах прогнозирования, согласно варианту лабораторной работы №1.

Код программы:
```
import math
import random

VECTOR_SIZE = 3
EXPECTED_ERROR = 0.0000001

weights = [random.uniform(-1, 1) for _ in range(VECTOR_SIZE)]
T = random.uniform(-1, 1)


def get_func_value(x: float) -> float:
  a, b, d = 3, 6, 0.1
  return a * math.sin(b * x) + d


def get_training_data() -> list:
  data_size = 30
  training_data = []
  current_x = 1
  step = 0.1
  for _ in range(data_size):
    line = []
    vector = []

    for _ in range(VECTOR_SIZE):
      vector.append(get_func_value(current_x))
```

```
        current_x += step

      line.append(vector)
      line.append(get_func_value(current_x))
      current_x -= step * 2

      training_data.append(line)

  return training_data


def train_model() -> None:
  common_error = 0
  iteration = 0
  while True:
    for line in get_training_data():
      vector = line[0]
      expected_result = line[-1]
      obtained_result = predict(vector)

      error = obtained_result - expected_result
      common_error += error ** 2

      change_weights(error, vector)

    if math.fabs(common_error) <= EXPECTED_ERROR:
      print('Iteration: ', iteration)
      break

    iteration += 1
    common_error = 0


def predict(vector: list) -> float:
  output = 0

  for index, value in enumerate(vector):
```

```
    output += value * weights[index]

  output -= T
  return output


def change_weights(error: float, vector: list) -> None:
  global T, weights

  step = get_step(vector)
  for w_index in range(len(weights)):
    weights[w_index] -= step * error * vector[w_index]

  T += step * error


def get_step(vector):
  return 1 / (1 + sum([x ** 2 for x in vector]))


if __name__ == '__main__':
  print(*get_training_data(), sep='\n')
  train_model()
  print('The training is complete!')

  test_vector = [get_func_value(1 + 0.1 * n) for n in range(3)]
  right_answer = get_func_value(1 + 0.1 * 3)
  print('Test vector:', *test_vector, sep=' ')
  print('Obtained result: ', predict(test_vector))
  print('Right result: ', right_answer)
```

Результат работы программы:


C:\Windows\system32\cmd.exe
[[-0.7382464945967776, 1.034624090540136, 2.481003591547461], 3.0956300361238154]
[[1.0346240905401387, 2.481003591547464, 3.0956300361238154], 2.663796724264836]
[[2.481003591547466, 3.095630036123816, 2.663796724264833 5], 1.3363554557252553]
[[3.0956300361238163, 2.663796724264833 5, 1.3363554557252504], -0.42298034366896]
[[2.663796724264830 4, 1.3363554557252504, -0.42298034366896 52], -1.9996240627806499]
[[1.3363554557252455, -0.42298034366897 04, -1.9996240627806499], -2.842808690199481]
[[-0.42298034366897574, -1.9996240627806534, -2.842808690199481], -2.657985576994012]
[[-1.9996240627806574, -2.842808690199482, -2.657985576994 01], -1.5097187540012689]
[[-2.842808690199482, -2.657985576994 01, -1.5097187540012689], 0.2008691416634527]
[[-2.657985576994 01, -1.5097187540012689, 0.2008691416634527], 1.8762205441217035]
[[-1.5097187540012689, 0.2008691416634527, 1.8762205441217035], 2.931087008332329]
[[0.2008691416634527, 1.8762205441217035, 2.931087008332329], 2.9969733296478203]
[[1.8762205441217035, 2.931087008332329, 2.9969733296478203], 2.050863520471314]
[[2.931087008332329, 2.9969733296478203, 2.050863520471314], 0.42326095689828447]
[[2.9969733296478203, 2.050863520471314, 0.42326095689828447], -1.3172659591954454]
[[2.050863520471314, 0.42326095689828447, -1.3172659591954454], -2.562701100744538]
[[0.42326095689828447, -1.3172659591954454, -2.562701100744538], -2.8779781414118912]
[[-1.3172659591954454, -2.562701100744538, -2.8779781414118912], -2.152961740314993]
[[-2.562701100744538, -2.8779781414118912, -2.152961740314993], -0.6409209852098111]
[[-2.8779781414118912, -2.152961740314993, -0.6409209852098111], 1.1299447864597463]
[[-2.152961740314993, -0.6409209852098111, 1.1299447864597463], 2.5410212125213474]
[[-0.6409209852098111, 1.1299447864597463, 2.5410212125213474], 3.099378700428009]
[[1.1299447864597463, 2.5410212125213474, 3.099378700428009], 2.6099669156081333]
[[2.5410212125213474, 3.099378700428009, 2.6099669156081333], 1.2437514749647613]
[[3.099378700428009, 2.6099669156081333, 1.2437514749647613], -0.5220092618203492]
[[2.6099669156081333, 1.2437514749647613, -0.5220092618203492], -2.070484268132779]
[[1.2437514749647613, -0.5220092618203492, -2.070484268132779], -2.8607466743619585]
[[-0.5220092618203492, -2.070484268132779, -2.8607466743619585], -2.6167350860198444]
[[-2.070484268132772, -2.8607466743619563, -2.6167350860198444], -1.4236897711718202]
[[-2.8607466743619563, -2.6167350860198533, -1.4236897711718295], 0.30162421757645674]
Iteration: 3
The training is complete!
Test vector: -0.7382464945967776 1.034624090540136 2.481003591547458
Obtained result: 3.0956209520436264
Right result: 3.095630036123815
D:\Projects\Лабораторные\МиАПР\miapr_po7\reports\Кравцевич\2\src>