Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 1

по курсу «Теория искусственных нейронных сетей»

«Реализация однослойного персептрона»

Студентка группы ИУ9-72Б Самохвалова П. С.

Преподаватель Каганов Ю. Т.

Москва 2023

1 Цель работы

Реализовать однослойный персептрон.

2 Задание

- Реализовать на языке высокого уровня однослойный персептрон и проверить его работоспособность на примере искусственных данных типа цифр от 0 до 9 и букв русского алфавита. Размер поля 5x4.
- Исследовать работу персептрона на основе использования различных функций активации. (Линейной, сигмоиды, гиперболического тангенса, ReLu).

3 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Однослойный персептрон

```
1  import math
2  from num_methods import *
3
4
5  def sigmoid_function(x):
6      return 1 / (1 + math.exp(-x))
7
8
9  def tanh_function(x):
10     return math.tanh(x)
11
12
13 def linear_function(x):
14     return x
15
16
17 def relu_function(x):
18     return max(0, x)
19
20
21 def sigmoid_function_der(x):
```

```
22      return sigmoid_function(x) * (1 - sigmoid_function(x))
23
24
25  def tanh_function_der(x):
26      return 1 - tanh_function(x) ** 2
27
28
29  def linear_function_der(x):
30      return 1
31
32
33  def relu_function_der(x):
34      if x > 0:
35          return 1
36      else:
37          return 0
38
39
40  def mse_loss(y_true, y_received):
41      y = sub_vec(y_true, y_received)
42      err = 0
43      for i in range(len(y)):
44          err += y[i] ** 2
45      err /= len(y)
46      return err
47
48
49  num0 = [1, 1, 1, 1,
50          1, 0, 0, 1,
51          1, 0, 0, 1,
52          1, 0, 0, 1,
53          1, 1, 1, 1]
54
55  num1 = [0, 0, 0, 1,
56          0, 0, 0, 1,
57          0, 0, 0, 1,
58          0, 0, 0, 1,
59          0, 0, 0, 1]
60
61  num2 = [1, 1, 1, 1,
62          0, 0, 0, 1,
63          1, 1, 1, 1,
64          1, 0, 0, 0,
65          1, 1, 1, 1]
66
67  num3 = [1, 1, 1, 1,
```

```
68            0, 0, 0, 1,
69            1, 1, 1, 1,
70            0, 0, 0, 1,
71            1, 1, 1, 1]
72
73 num4 = [1, 0, 0, 1,
74            1, 0, 0, 1,
75            1, 1, 1, 1,
76            0, 0, 0, 0,
77            1, 1, 1, 1]
78
79 num5 = [1, 1, 1, 1,
80            1, 0, 0, 0,
81            1, 1, 1, 1,
82            0, 0, 0, 1,
83            1, 1, 1, 1]
84
85 num6 = [1, 1, 1, 1,
86            1, 0, 0, 0,
87            1, 1, 1, 1,
88            1, 0, 0, 1,
89            1, 1, 1, 1]
90
91 num7 = [1, 1, 1, 1,
92            0, 0, 0, 1,
93            0, 0, 0, 1,
94            0, 0, 0, 1,
95            0, 0, 0, 1]
96
97 num8 = [1, 1, 1, 1,
98            1, 0, 0, 1,
99            1, 1, 1, 1,
100           1, 0, 0, 1,
101           1, 1, 1, 1]
102
103 num9 = [1, 1, 1, 1,
104           1, 0, 0, 1,
105           1, 1, 1, 1,
106           0, 0, 0, 1,
107           1, 1, 1, 1]
108
109 nums = [num0, num1, num2, num3, num4, num5, num6, num7, num8, num9]
110
111 learning_rate = 0.1
112 epochs = 1000
113
```

```
114
115 class Perceptron:
116     def __init__(self, size, activation_function):
117         self.activation_function = activation_function
118         self.size = size
119         self.weights = [[0] * size for _ in range(10)]
120         self.loss = []
121
122     def calculation(self, num, weights_ind):
123         return scalar_mult_vec(num, self.weights[weights_ind])
124
125     def calculate_res(self, num, weights_ind):
126         return self.activation_function(scalar_mult_vec(num, self.
    weights[weights_ind]))
127
128     def train(self, nums, learning_rate, epochs):
129         for epoch in range(epochs):
130             for num_ind in range(len(nums)):
131                 for weights_ind in range(len(self.weights)):
132                     y_true = int(num_ind == weights_ind)
133                     y_received = self.calculate_res(nums[num_ind],
    weights_ind)
134                     error = y_true - y_received
135                     for i in range(len(self.weights[weights_ind])):
136                         self.weights[weights_ind][i] += learning_rate *
    nums[num_ind][i] * error
137             y_receiveds = []
138             y_trues = []
139             for num_ind in range(len(nums)):
140                 for weights_ind in range(len(self.weights)):
141                     y_trues.append(int(num_ind == weights_ind))
142                     y_receiveds.append(self.calculate_res(nums[num_ind],
     weights_ind))
143             self.loss.append(mse_loss(y_trues, y_receiveds))
144
145     def get_loss(self, i):
146         return self.loss[i]
147
148
149 def check(perceptron, num):
150     max_value = 0
151     res_num = 0
152     for i in range(10):
153         value = perceptron.calculate_res(num, i)
154         if value > max_value:
155             max_value = value
```

```
156                 res_num = i
157         if max_value != 0:
158             return res_num
159         else:
160             return "Digit is not recognized"
161
162
163 perceptron_sigmoid = Perceptron(size=20, activation_function=
        sigmoid_function)
164 perceptron_tanh = Perceptron(size=20, activation_function=tanh_function)
165 perceptron_linear = Perceptron(size=20, activation_function=
        linear_function)
166 perceptron_relu = Perceptron(size=20, activation_function=relu_function)
167
168 perceptron_sigmoid.train(nums, learning_rate, epochs)
169 print("Perceptron sigmoid weights")
170 print(perceptron_sigmoid.weights)
171 perceptron_tanh.train(nums, learning_rate, epochs)
172 print("Perceptron tanh weights")
173 print(perceptron_tanh.weights)
174 perceptron_linear.train(nums, learning_rate, epochs)
175 print("Perceptron linear weights")
176 print(perceptron_linear.weights)
177 perceptron_relu.train(nums, learning_rate, epochs)
178 print("Perceptron relu weights")
179 print(perceptron_relu.weights)
180
181 print(check(perceptron_sigmoid, num2))
182 print(check(perceptron_sigmoid, num5))
183 print(check(perceptron_sigmoid, num7))
184
185 num51 = [1, 1, 1, 1,
186          1, 0, 0, 0,
187          1, 1, 1, 0,
188          0, 0, 0, 1,
189          1, 1, 1, 1]
190
191 num52 = [1, 1, 1, 1,
192          1, 0, 0, 0,
193          0, 1, 1, 1,
194          0, 0, 0, 1,
195          1, 1, 1, 1]
196
197 num53 = [1, 1, 1, 0,
198          1, 0, 0, 0,
199          1, 1, 1, 1,
```

```
200             0, 0, 0, 1,
201             1, 1, 1, 1]
202
203 num54 = [1, 1, 1, 0,
204             1, 0, 0, 0,
205             1, 1, 1, 1,
206             0, 0, 0, 1,
207             0, 1, 1, 1]
208
209 num71 = [1, 1, 1, 1,
210             0, 0, 0, 1,
211             0, 0, 1, 1,
212             0, 0, 0, 1,
213             0, 0, 0, 1]
214
215 num81 = [0, 1, 1, 0,
216             1, 0, 0, 1,
217             0, 1, 1, 0,
218             1, 0, 0, 1,
219             0, 1, 1, 0]
220
221 print(check(perceptron_sigmoid, num51))
222 print(check(perceptron_sigmoid, num52))
223 print(check(perceptron_sigmoid, num53))
224 print(check(perceptron_sigmoid, num54))
225 print(check(perceptron_sigmoid, num71))
226 print(check(perceptron_sigmoid, num81))
227
228
229 plt.plot([i for i in range(epochs)], [perceptron_sigmoid.get_loss(i) for
        i in range(epochs)], label='Sigmoid', color='blue')
230 plt.xlabel('Epochs')
231 plt.ylabel('Loss')
232 plt.title('Loss and Epochs')
233 plt.legend()
234 plt.show()
235 plt.plot([i for i in range(epochs)], [perceptron_tanh.get_loss(i) for i
       in range(epochs)], label='Tanh', color='yellow')
236 plt.xlabel('Epochs')
237 plt.ylabel('Loss')
238 plt.title('Loss and Epochs')
239 plt.legend()
240 plt.show()
241 plt.plot([i for i in range(epochs)], [perceptron_linear.get_loss(i) for
       i in range(epochs)], label='Linear', color='green')
242 plt.xlabel('Epochs')
```

```
plt.ylabel('Loss')
plt.title('Loss and Epochs')
plt.legend()
plt.show()
plt.plot([i for i in range(epochs)], [perceptron_relu.get_loss(i) for i
    in range(epochs)], label='ReLu', color='red')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss and Epochs')
plt.legend()
plt.show()
```

4 Результаты

Результаты работы программы представлены на рисунках 1 – 6.

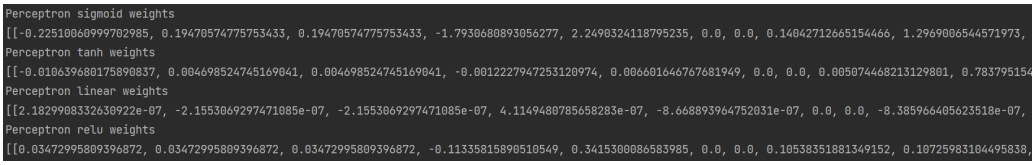

Рис. 1 — Полученные веса

Рис. 2 — Тестирование распознавания цифр

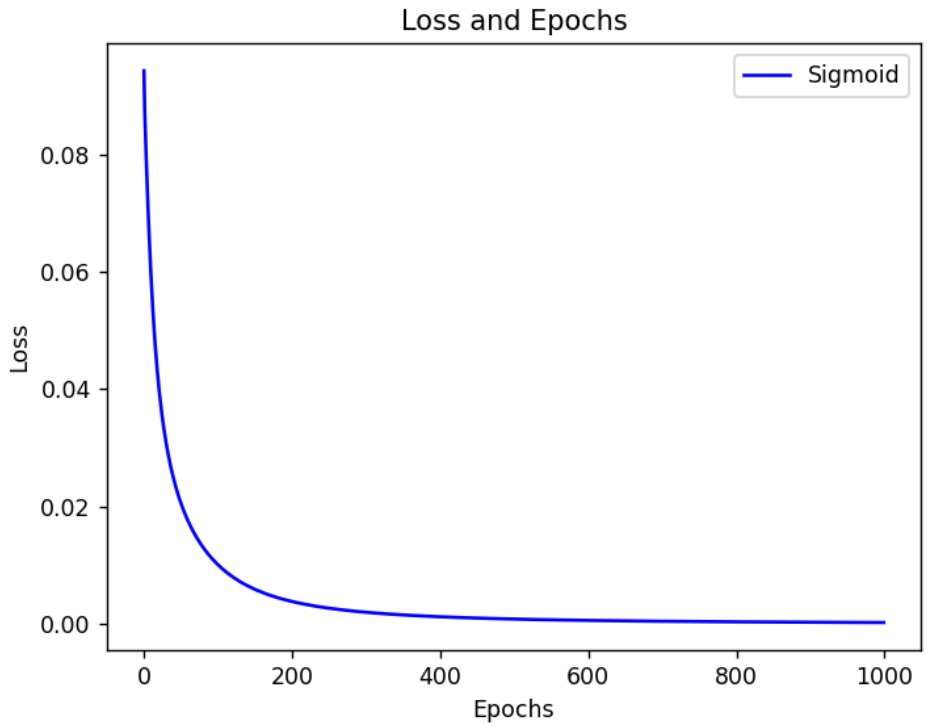

Рис. 3 — Зависимость функции потерь от числа эпох

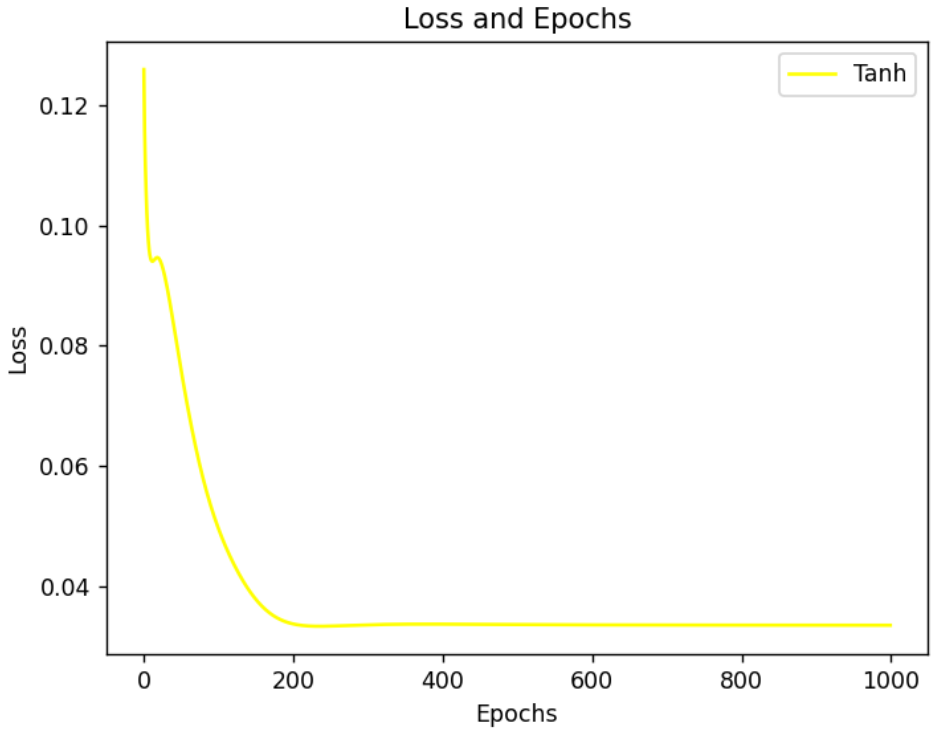

Рис. 4 — Зависимость функции потерь от числа эпох

5 Выводы

В результате выполнения лабораторной работы был реализован однослойный персептрон.

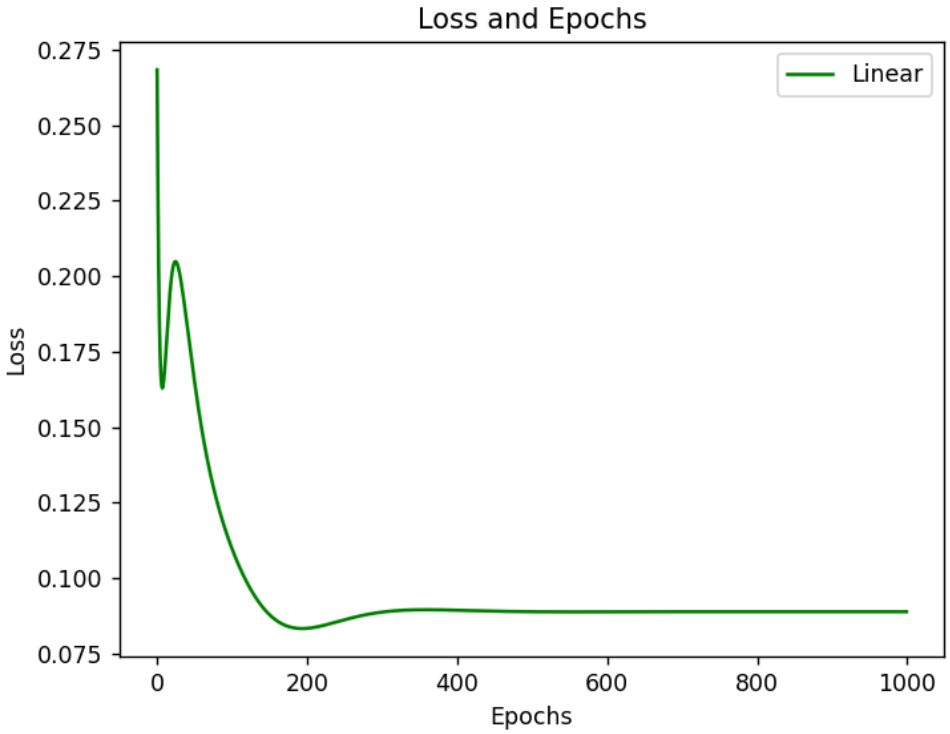

Рис. 5 — Зависимость функции потерь от числа эпох

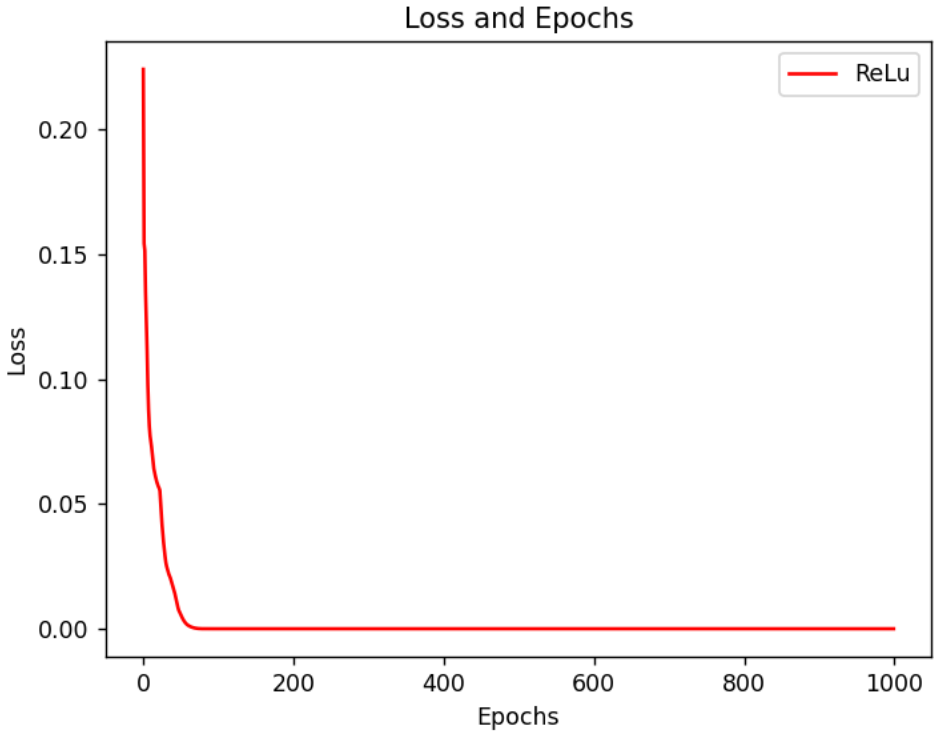

Рис. 6 — Зависимость функции потерь от числа эпох