Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Бинарная классификация»

**Выполнил:**

Студент 3 курса

Группы ИИ-23

Романюк А. П.

**Проверил:**

Туз И. С.

Брест 2024

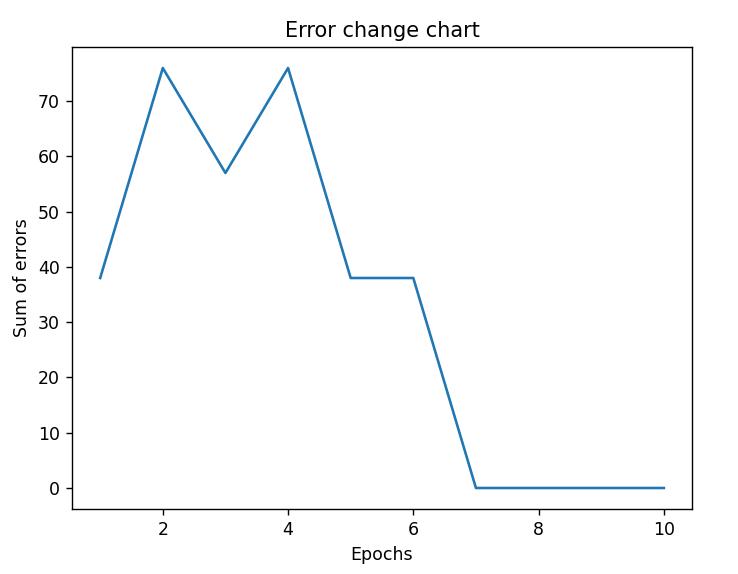
**Цель:** реализовать однослойную нейронную сеть для решения задачи классификации с использованием пороговой функции активации.

**Ход работы****Вариант 9**

**Изображение выглядит как текст, число

Автоматически созданное описание**

import numpy as np  
import matplotlib.pyplot as plt  
class Perceptron:  
 def \_\_init\_\_(self, input\_size):  
 self.weights = np.random.rand(input\_size)  
 # self.weights = np.array([1., 1])  
 self.bias = np.random.rand(1)  
 def activation\_function(self, x):  
 return -1 if x < 0 else 18  
 def predict(self, inputs):  
 weighted\_sum = self.weights.transpose()@inputs+self.bias  
 return self.activation\_function(weighted\_sum)  
 def train(self, training\_inputs, labels, learning\_rate, epochs):  
 errors = []  
 for \_ in range(epochs):  
 total\_error = 0  
 for inputs, label in zip(training\_inputs, labels):  
 prediction = self.predict(inputs)  
 error = label - prediction  
 total\_error += abs(error)  
 self.weights += learning\_rate \* error \* inputs  
 self.bias += learning\_rate \* error  
 errors.append(total\_error)  
 plt.plot(range(1, epochs + 1), errors)  
 plt.xlabel('Epochs')  
 plt.ylabel('Sum of errors')  
 plt.title('Error change chart')  
 plt.show()  
 def plot\_decision\_boundary(self, training\_inputs, labels):  
 x\_min, x\_max = min(training\_inputs[:, 0]) - 1, max(training\_inputs[:, 0]) + 1  
 y\_min, y\_max = min(training\_inputs[:, 1]) - 1, max(training\_inputs[:, 1]) + 1  
 xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01),  
 np.arange(y\_min, y\_max, 0.01))  
  
 z = np.array([self.predict(np.array([x, y])) for x, y in zip(xx.ravel(), yy.ravel())])  
 z = z.reshape(xx.shape)  
  
 plt.contourf(xx, yy, z, alpha=0.3, cmap=plt.cm.coolwarm)  
 plt.scatter(training\_inputs[:, 0], training\_inputs[:, 1], c=labels, cmap=plt.cm.coolwarm)  
 plt.xlabel('Feature 1')  
 plt.ylabel('Feature 2')  
 plt.title('Condition plot (decision boundary and points)')  
 plt.show()  
if \_\_name\_\_ == "\_\_main\_\_":  
 training\_inputs = np.array([[-3, -3], [-3, -2], [-2, -3], [-2, -2]])  
 labels = np.array([-1, 18, -1, 18])  
 perceptron = Perceptron(input\_size=2)  
 learning\_rate = 0.003  
 epochs = 10  
 perceptron.train(training\_inputs, labels, learning\_rate, epochs)  
 perceptron.plot\_decision\_boundary(training\_inputs, labels)

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

**Вывод:** в ходе лабораторной работы я научился реализовывать однослойную нейронную сеть для решения задачи классификации с использованием пороговой функции активации.