Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Понижение размерности входного образа»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

Туз И. С.

Брест 2023

**Цель:** изучить влияние изменения размерности входного образа на качество обучения сети.

**Ход работы (Вариант 4)**

В рамках данной работы необходимо реализовать 2 классификатора:

1. Персептрон с одним скрытым слоем.
2. Персептрон с одним скрытым слоем с использованием понижения размерности входного образа с помощью автоэнкодера. Под этим имеется в виду следующее:
   1. Обучение автоэнкодера

Датасет: seeds.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import classification\_report

class Perceptron:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size, learning\_rate=0.1):

self.input\_size = input\_size

self.hidden\_size = hidden\_size

self.output\_size = output\_size

self.learning\_rate = learning\_rate

self.weights\_input\_hidden = np.random.randn(self.input\_size, self.hidden\_size)

self.bias\_hidden = np.zeros((1, self.hidden\_size))

self.weights\_hidden\_output = np.random.randn(self.hidden\_size, self.output\_size)

self.bias\_output = np.zeros((1, self.output\_size))

def sigmoid(self, x):

return 1 / (1 + np.exp(-x))

def sigmoid\_derivative(self, x):

return x \* (1 - x)

def forward(self, inputs):

self.hidden\_input = np.dot(inputs, self.weights\_input\_hidden) + self.bias\_hidden

self.hidden\_output = self.sigmoid(self.hidden\_input)

self.output = np.dot(self.hidden\_output, self.weights\_hidden\_output) + self.bias\_output

return self.output

def backward(self, inputs, target, output):

error = target - output

delta\_output = error

delta\_hidden = delta\_output.dot(self.weights\_hidden\_output.T) \* self.sigmoid\_derivative(self.hidden\_output)

self.weights\_hidden\_output += self.hidden\_output.T.dot(delta\_output) \* self.learning\_rate

self.bias\_output += np.sum(delta\_output, axis=0, keepdims=True) \* self.learning\_rate

self.weights\_input\_hidden += inputs.T.dot(delta\_hidden) \* self.learning\_rate

self.bias\_hidden += np.sum(delta\_hidden, axis=0, keepdims=True) \* self.learning\_rate

def train(self, inputs, targets, epochs):

for epoch in range(epochs):

for i in range(len(inputs)):

input\_data = np.array([inputs[i]])

target\_data = np.array([targets[i]])

output = self.forward(input\_data)

self.backward(input\_data, target\_data, output)

def predict(self, inputs):

output = self.forward(inputs)

return output

def encode(self, inputs):

return np.dot(inputs, self.weights\_input\_hidden) + self.bias\_hidden

if \_\_name\_\_ == "\_\_main\_\_":

dataset = pd.read\_csv("Seed\_Data.csv")

X = dataset.iloc[:, 0:7]

Y = dataset.iloc[:, 7]

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

size = X.shape[1]

autoencoder = Perceptron(input\_size=size, hidden\_size=size // 2, output\_size=size)

inputs, targets = X, X

autoencoder.train(inputs, targets, epochs=100)

X\_enc = autoencoder.encode(X)

X\_df = pd.DataFrame(X\_enc)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X\_df, Y, test\_size=0.2)

size = X\_train.shape[1]

perceptron = Perceptron(input\_size=size, hidden\_size=size, output\_size=1)

inputs, targets = X\_train.to\_numpy(), Y\_train.to\_numpy()

perceptron.train(inputs, targets, epochs=100)

predictions = perceptron.predict(X\_test.to\_numpy())

predictions = abs(np.round(predictions).astype(int)).flatten()

accuracy = np.mean(predictions == Y\_test)

Y\_test = Y\_test.to\_numpy()

print(classification\_report(Y\_test, predictions))

# accuracy = TP + TN / TP + TN + FP + FN

# precision = TP / TP + FP

# recall = TP / TP + FN

**Вывод:** изучил влияние сжатия информации на точность сети.