Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «МРЗИС»

Тема: “Понижение размерности входного образа”

Выполнил:

Студент 3 курса

Группы ИИ-21

Романко Н.А.

Проверил:

Туз И.С.

Брест 2023

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

Код программы:

import pandas as pd

import numpy as np

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

file = pd.read\_csv('1 semestr\\MRZIS\\lab3\\WineQT.csv')

#модификация датасета

file = file[file['volatile acidity'] < 1]

file = file[file['sulphates'] < 1]

file = file[file['chlorides'] < 0.14]

file = file[file['free sulfur dioxide'] < 35]

file = file[(file['quality'] != 3) & (file['quality'] != 4) & (file['quality'] != 8)]

col = ['fixed acidity', 'volatile acidity', 'citric acid', 'chlorides', 'total sulfur dioxide', 'pH','sulphates', 'alcohol']

X = pd.DataFrame()

for i in col:

    X[i] = file[i]

Y = file['quality'].apply(lambda x: x - 5 if x>=5 else x)

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

activation = Softmax()

autoencoder\_input\_layer = Layer\_Dense(len(col), 4)

autoencoder\_hidden\_layer = Layer\_Dense(4, 4)

autoencoder\_output\_layer = Layer\_Dense(4, 8)

loss\_func = Loss\_CCE\_and\_Softmax()

optimazer = Optimazer\_Adam(learning\_rate=0.12, decay=3e-5)

print("Обчение автоенкодера")

for epoch in range(500):

    autoencoder\_input\_layer.forward(X\_train)

    activation.forward(autoencoder\_input\_layer.output)

    autoencoder\_hidden\_layer.forward(activation.output)

    activation.forward(autoencoder\_hidden\_layer.output)

    autoencoder\_output\_layer.forward(activation.output)

    loss = loss\_func.forward(autoencoder\_output\_layer.output, y\_train)

    predictions = np.argmax(loss\_func.output, axis=1)

    if len(y\_train.shape) == 2:

        y\_train = np.argmax(y\_train, axis=1)

    accuracy = np.mean(predictions==y\_train)

    if not epoch % 100:

        print(f'epoch: {epoch}, ' + f'acc: {accuracy:.3f}, ' + f'loss: {loss:.3f}, ' + f'lr: {optimazer.current\_learning\_rate}')

    loss\_func.backward(loss\_func.output, y\_train)

    autoencoder\_output\_layer.backward(loss\_func.dinputs)

    activation.backward(autoencoder\_output\_layer.dinputs)

    autoencoder\_hidden\_layer.backward(activation.dinputs)

    activation.backward(autoencoder\_hidden\_layer.dinputs)

    autoencoder\_input\_layer.backward(activation.dinputs)

    optimazer.pre\_update\_params()

    optimazer.update\_params(autoencoder\_input\_layer)

    optimazer.update\_params(autoencoder\_hidden\_layer)

    optimazer.update\_params(autoencoder\_output\_layer)

    optimazer.post\_update\_params()

autoencoder\_input\_layer.forward(X)

activation.forward(autoencoder\_input\_layer.output)

autoencoder\_hidden\_layer.forward(activation.output)

# print("hidden: ", autoencoder\_hidden\_layer.output)

activation.forward(autoencoder\_hidden\_layer.output)

# print("after activation: ", activation.output)

autoencoder\_output\_layer.forward(activation.output)

# print("output: ", autoencoder\_output\_layer.output)

X\_enc = autoencoder\_output\_layer.output

input\_layer = Layer\_Dense(len(col), 8)

hidden\_layer = Layer\_Dense(8, 8)

output\_layer = Layer\_Dense(8, 3)

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

print("обучение перептрона на данных с автоенкодра")

for epoch in range(500):

    input\_layer.forward(X\_train)

    activation.forward(input\_layer.output)

    hidden\_layer.forward(activation.output)

    activation.forward(hidden\_layer.output)

    output\_layer.forward(activation.output)

    loss = loss\_func.forward(output\_layer.output, y\_train)

    predictions = np.argmax(loss\_func.output, axis=1)

    if len(y\_train.shape) == 2:

        y\_train = np.argmax(y\_train, axis=1)

    accuracy = np.mean(predictions==y\_train)

    if not epoch % 100:

        print(f'epoch: {epoch}, ' + f'acc: {accuracy:.3f}, ' + f'loss: {loss:.3f}, ' + f'lr: {optimazer.current\_learning\_rate}')

    loss\_func.backward(loss\_func.output, y\_train)

    output\_layer.backward(loss\_func.dinputs)

    activation.backward(output\_layer.dinputs)

    hidden\_layer.backward(activation.dinputs)

    activation.backward(hidden\_layer.dinputs)

    input\_layer.backward(activation.dinputs)

    optimazer.pre\_update\_params()

    optimazer.update\_params(input\_layer)

    optimazer.update\_params(hidden\_layer)

    optimazer.update\_params(output\_layer)

    optimazer.post\_update\_params()

input\_layer.forward(X\_test)

activation.forward(input\_layer.output)

hidden\_layer.forward(activation.output)

activation.forward(hidden\_layer.output)

output\_layer.forward(activation.output)

loss = loss\_func.forward(output\_layer.output, y\_test)

predictions = np.argmax(loss\_func.output, axis=1)

if len(y\_test.shape) == 2:

    y\_test = np.argmax(y\_test, axis=1)

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

print(f'final acc: {accuracy:.3f}, ' + f'loss: {loss:.3f}')

Вывод программы:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Вывод: научился реализовывать и применять на практике автоенкодер. Изучил влияние изменения входного образа на результат работы персептрона.