Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине «МРЗИС»

Тема: “РСА и автоэнкодеры”

Выполнил:

Студент 3 курса

Группы ИИ-21

Романко Н.А.

Проверил:

Туз И.С.

Брест 2024

Цель:реализовать PCA и автоэнкодер и сравнить время и качество их обучения.

Начальные данные: датасет WineQT.csv

Код автоенкодера:

import pandas as pd

import torch

import torch.nn as nn

import torch.optim as optim

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import classification\_report

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

file = pd.read\_csv('C:\\work\\MRZIS\\WineQT.csv')

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)]

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

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, Y, test\_size=0.2)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

X\_train\_tensor = torch.FloatTensor(X\_train\_scaled)

X\_test\_tensor = torch.FloatTensor(X\_test\_scaled)

class Autoencoder(nn.Module):

def \_\_init\_\_(self, input\_dim, encoding\_dim):

super(Autoencoder, self).\_\_init\_\_()

self.encoder = nn.Linear(input\_dim, encoding\_dim)

self.decoder = nn.Linear(encoding\_dim, input\_dim)

def forward(self, x):

encoded = torch.relu(self.encoder(x))

decoded = torch.sigmoid(self.decoder(encoded))

return decoded

input\_dim = X\_train\_tensor.shape[1]

encoding\_dim = 4

autoencoder = Autoencoder(input\_dim, encoding\_dim)

criterion = nn.MSELoss()

optimizer = optim.Adam(autoencoder.parameters(), lr=0.001)

num\_epochs = 2000

for epoch in range(num\_epochs):

optimizer.zero\_grad()

outputs = autoencoder(X\_train\_tensor)

loss = criterion(outputs, X\_train\_tensor)

loss.backward()

optimizer.step()

if (epoch+1) % 10 == 0:

print(f'Epoch [{epoch+1}/{num\_epochs}], Loss: {loss.item():.4f}')

encoded\_X\_train\_tensor = autoencoder.encoder(X\_train\_tensor).detach().numpy()

encoded\_X\_test\_tensor = autoencoder.encoder(X\_test\_tensor).detach().numpy()

mlp = MLPClassifier(hidden\_layer\_sizes=(5,), activation='logistic', max\_iter=2000, learning\_rate\_init=0.12, alpha=3e-5)

mlp.fit(encoded\_X\_train\_tensor, y\_train)

predictions = mlp.predict(encoded\_X\_test\_tensor)

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

accuracy = mlp.score(encoded\_X\_test\_tensor, y\_test)

print(f'Final accuracy: {accuracy:.3f}')

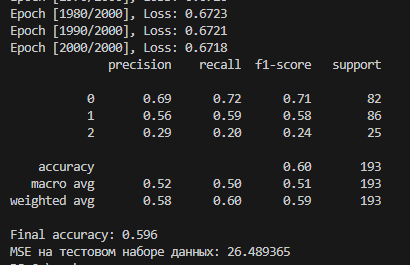
reconstructed\_X\_train\_tensor = autoencoder.decoder(torch.FloatTensor(encoded\_X\_train\_tensor)).detach().numpy()

reconstructed\_X\_test\_tensor = autoencoder.decoder(torch.FloatTensor(encoded\_X\_test\_tensor)).detach().numpy()

mse\_test = mean\_squared\_error(X\_test\_tensor, reconstructed\_X\_test\_tensor)

print("MSE на тестовом наборе данных:", mse\_test)

Результат обучения и тестирования автоенкодера:



Код РСА:

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_squared\_error

X\_train\_scaled = np.array([[-2, 2], [2, -2]])

X\_mean = X\_train\_scaled.mean()

X\_centered = X\_train\_scaled - X\_mean

cov = np.cov(X\_train\_scaled.T)

eigenvalues, eigenvectors = np.linalg.eig(cov)

first\_eigenvector = list(reversed(eigenvectors[0]))

second\_eigenvector = list(reversed(eigenvectors[1]))

first\_PC = X\_centered \* first\_eigenvector

second\_PC = X\_centered \* second\_eigenvector

first\_PC = first\_PC.sum(axis=1)

second\_PC = second\_PC.sum(axis=1)

reconstructed\_X\_train\_scaled = np.dot(np.array([first\_PC, second\_PC]).T, eigenvectors[:2, :]) + X\_mean

mse = mean\_squared\_error(X\_train\_scaled, reconstructed\_X\_train\_scaled)

print("MSE между исходными и восстановленными данными:", mse)

Результат работы РСА:



Вывод: реализовал автоенкодер и РСА, сравнил время и качество обучения на данном датасете.