МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № 1

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил:

Туз И. С.

Брест – 2024

**Ход работы**

**Задание:** в рамках данной работы необходимо реализовать PCA и автоэнкодер и сравнить время и качество их обучения.

|  |  |
| --- | --- |
| import torch  import torch.nn as nn  import torch.optim as optim  class Autoencoder(nn.Module):  def \_\_init\_\_(self, input\_size, hidden\_size):  super(Autoencoder, self).\_\_init\_\_()  self.encoder = nn.Sequential(  nn.Linear(input\_size, hidden\_size),  nn.LeakyReLU()  )  self.decoder = nn.Sequential(  nn.Linear(hidden\_size, input\_size),  nn.LeakyReLU()  )  def forward(self, x):  x = self.encoder(x)  x = self.decoder(x)  return x | import numpy as np    def PCA(X, num\_components, return\_error=False):  X\_meaned = X - np.mean(X, axis=0)  cov\_mat = np.cov(X\_meaned, rowvar=False)  eigen\_values, eigen\_vectors = np.linalg.eigh(cov\_mat)  sorted\_index = np.argsort(eigen\_values)[::-1]  sorted\_eigenvectors = eigen\_vectors[:, sorted\_index]  eigenvector\_subset = sorted\_eigenvectors[:, 0:num\_components]  X\_reduced = np.dot(eigenvector\_subset.transpose(), X\_meaned.transpose()).transpose()  if return\_error:  X\_restored = np.dot(X\_reduced, eigenvector\_subset.transpose()) + np.mean(X, axis=0)  reconstruction\_error = np.mean(np.square(X - X\_restored))  return X\_reduced, reconstruction\_error  else:  return X\_reduced |
| *Реализация автоэнкодера* | *Реализация PCA* |

|  |  |
| --- | --- |
| import pandas as pd  df = pd.read\_csv("./WineQT.csv")  X = df.iloc[:, :11].values  y = df.iloc[:, 11].values.reshape(-1, 1)  print("Размер обучающего набора: ", y.shape[0]) | from sklearn.preprocessing import StandardScaler  import time  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)    start\_time = time.time()  X\_pca, pca\_error = PCA(X\_scaled, 2, return\_error=True)  end\_time = time.time()  print("Время расчёта:", end\_time - start\_time)  print("Ошибка для PCA:", pca\_error) |
| *Загрузка датасета WineQT* | *Нормализация входных данных* |

|  |  |
| --- | --- |
| input\_size = X\_scaled.shape[1]  hidden\_size = 2  autoencoder = Autoencoder(input\_size, hidden\_size)  criterion = nn.MSELoss()  optimizer = optim.Adam(autoencoder.parameters(), lr=0.001)  num\_epochs = 100  batch\_size = 16  X\_torch = torch.tensor(X\_scaled, dtype=torch.float32)  loss\_torch = []  for epoch in range(num\_epochs):  loss\_epoch = 0  for i in range(0, len(X\_torch), batch\_size):  batch\_x = X\_torch[i:i+batch\_size]  output = autoencoder(batch\_x)  loss = criterion(output, batch\_x)  loss\_epoch += loss.item()  optimizer.zero\_grad()  loss.backward()  optimizer.step()  loss\_torch.append(loss\_epoch | import matplotlib.pyplot as plt  plt.plot(np.arange(num\_epochs), loss\_torch)  plt.title('График обучения Autoencoder')  plt.xlabel('Epoch')  plt.ylabel('Loss')  plt.show() |
| *Обучение автоэнкодера* | *График изменения ошибки при обучении* |

start\_time = time.time()

with torch.no\_grad():

X\_torch\_restored = autoencoder(X\_torch).numpy()

end\_time = time.time()

print("Время расчёта:", end\_time - start\_time)

autoencoder\_error = np.mean((X\_scaled - X\_torch\_restored) \*\* 2)

print("Ошибка для автоэнкодера:", autoencoder\_error)

*Рассчёт ошибки для автоэнкодера*

|  |  |  |
| --- | --- | --- |
|  | **PCE** | **Автоэнкодер** |
| **Ошибка преобразования** | 0.5423255152723097 | 0.6966298103371684 |
| **Время расчёта** | 0.0019986629486083984 | 0.0009853839874267578 |

*Размер обучающего набора: 1143*