**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **СШІ**

**ЗВІТ**

До лабораторної роботи №2

**З дисципліни:**

**Машинне Навчання**

**Виконав:**

ст. гр. КН-308

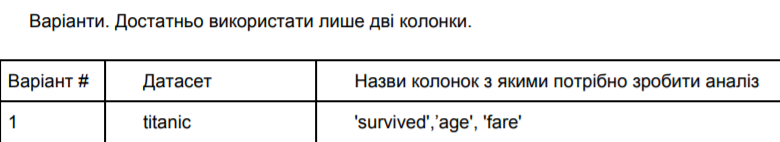
Матвіїв М.А.

**Прийняв:**

Якимишин Х.М.

Львів – 2020

Варіант 1



Мета: засвоїти основні відомості про роботу з алгоритмом метод головних компонент(PCA). Навчитись реалізовувати даний алгоритм.

Теоретичні відомості

Опис методу головних компонент. Застосування.

Ме́тод головни́х компоне́нт (PCA) — метод аналізу в статистиці, який використовує ортогональне перетворення множини спостережень з можливо пов'язаними змінними (сутностями, кожна з яких набуває різних числових значень) у множину змінних без лінійної кореляції, які називаються головними компонентами.

Загалом Основні сфери застосування PCA:

- Візуалізація даних

- Стиснення даних

- Зменшення шуму

- Анонімізація даних

Покроковий опис алгоритму обрахунку PCA.

1 Підготуйте дані. При потребі розділіть на залежні і незалежні змінні(Y і X). Нехай X дані мають розмірність d x n, де d - кількість ознак, а n - кількість рядків даних

2 Відцентруйте дані. Порахуйте вектор середніх ознак який матиме розмірність d і міститиме середні значення усіх ознак. Відніміть цей вектор від кожного рядка даних.

3 Якщо у вас є залежні ознаки(Y): a. Обрахуйте вектор стандартного відхилення усіх ваших ознак b. Розділіть кожну одиницю даних на цей вектор. c. Цю відцентровану і нормалізовану матрицю назвемо Z.

4. Обрахуйте матрицю коваріації ZᵀZ/(n-1)

5. Обрахуйте власні значення і власні вектори отриманої матриці

6. Посортуйте значення та вектори згідно спадання значень. Посортовану матрицю векторів назвемо P\*.

7. Обрахуйте нові ознаки. Z\* = ZP\*.

**Завдання**: Реалізувати метод головних компонент і використати його для аналізу даних згідно варіантів, а саме:

1. Частина 1. Реалізація алгоритму

a. Згідно варіанту, обрахувати PCA використовуючи sklearn. Звізуалізувати основні компоненти згідно прикладу.

b. Реалізувати PCA алгоритм використовуючи PyTorch. Візуалізувати результати алгоритму.

c. Порівняти результати отримані компоненти, порівняти їхні результати. Вони повинні співпадати при правильній реалізації.

2. Частина 2. Обезшумлення

a. Добавити до даних шум.

b. Обрахувати на нових зашумлених даних PCA.

c. Реконстроювати дані використовуючи PCA для знешумлення.

d. Звізуалізувати зашумлені і знешумлені дані

**Хід Роботи**

**Код програми буде прикріплено до звіту**

**Частина 1. Реалізація алгоритму**

Підключимо всі необхідні модулі

import torch

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.decomposition import PCA

використаємо наші функції з 1 лабораторної для заміни nan

та нормалізації даних

# find average value without nan

def get\_average(array):

sum1 = 0

elements = 0

for i in array:

if not np.isnan(i):

sum1 += i

elements += 1

return sum1/elements

# replace nan by average value

def replace\_nan\_by\_average(array, average\_number):

for en, i in enumerate(array):

if np.isnan(i):

array[en] = average\_number

array[en] -= average\_number

return array

# draw vector :)

def draw\_vector(v0, v1, ax=None):

ax = ax or plt.gca()

arrowprops=dict(

linewidth=2,

shrinkA=0, shrinkB=0)

ax.annotate('', v1, v0, arrowprops=arrowprops)

Завантажимо дані та підготуємо їх

# load data

tit = sns.load\_dataset("titanic")

age = tit['age']

fare = tit['fare']

# replace nan

age\_avg = get\_average(age)

age = replace\_nan\_by\_average(age, age\_avg)

fare\_avg = get\_average(fare)

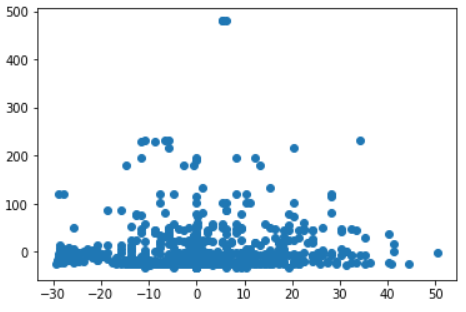
fare = replace\_nan\_by\_average(fare, fare\_avg)

# prepare data for PCA function

X = [[x,y] for x, y in zip(age, fare)]

X = np.array(X)

Візуалізуємо наші дані



Використаємо PCA алгоритм з бібліотеки sklearn

# use sklearn PCA function

pca = PCA(n\_components=2)

pca.fit(X)

# pca.components\_ is the set of all eigenvectors

# власні значення

print(pca.components\_)

print()

print(pca.explained\_variance\_)

Та візуалізуємо його результат

# Visualize PCA using sklearn

# plot data

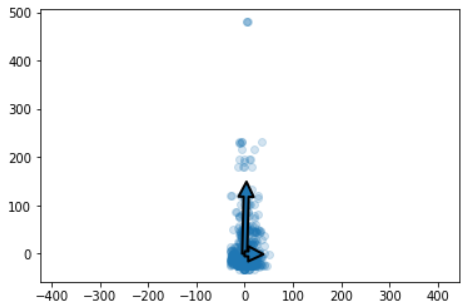
plt.scatter(X[:, 0], X[:, 1], alpha=0.2)

for length, vector in zip(pca.explained\_variance\_, pca.components\_):

v = vector \* 3 \* np.sqrt(length)

draw\_vector(pca.mean\_, pca.mean\_ + v)

plt.axis('equal');



Тепер напишемо власну функцію PCA використовуючи pytorch

def my\_PCA(X, k, center=True):

n = X.size()[0]

ones = torch.ones(n).view([n,1])

# 1/n tensor

h = ((1/n) \* torch.mm(ones, ones.t())) if center else torch.zeros(n\*n).view([n,n])

# Віднімаю від діагональної матрці з діагональними елементами рівними 1 матрицю h

H = torch.eye(n) - h

# множення матриць H і X

X\_center = torch.mm(H.double(), X.double())

# Сингулярний розклад матриці за допомогою метода svd (сингулярий розклад матриці)

u, s, v = torch.svd(X\_center)

# Транспонування матриці v і це є наші компоненти

components = v[:k].t()

for i in components:

i[0], i[1] = -i[0], -i[1]

# Поелементне піднесення до квадрату матриці s після чого діль елементи кінцевої матриці на (n-1)

explained\_variance = torch.mul(s[:k], s[:k])/(n-1)

return {'components':components,

'explained\_variance':explained\_variance }

Застосуємо її

my\_pca = my\_PCA(torch.IntTensor(X), 2)

Та візуалізуємо результат

# visualize PCA using pytorch

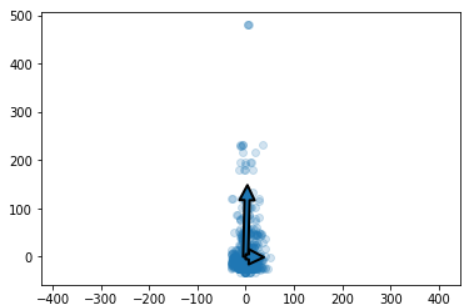
plt.scatter(X[:, 0], X[:, 1], alpha=0.2)

for length, vector in zip(my\_pca['explained\_variance'], my\_pca['components']):

v = vector \* 3 \* np.sqrt(length)

draw\_vector(pca.mean\_, pca.mean\_ + v.data.numpy())

plt.axis('equal');

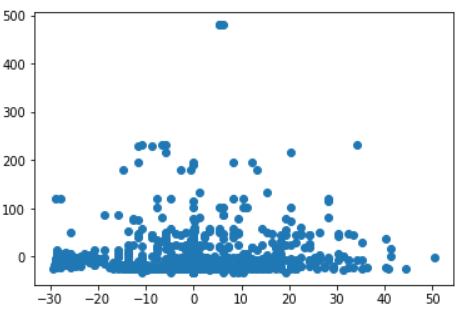


Як ми бачимо, результат ідентичний, отже наша реалізація алгоритму працює коректно

**Частина 2 Знешумлення**

Візуалізуємо наші дані

plt.scatter(X[:, 0], X[:, 1])



Виконаємо зашумлення до наших даних

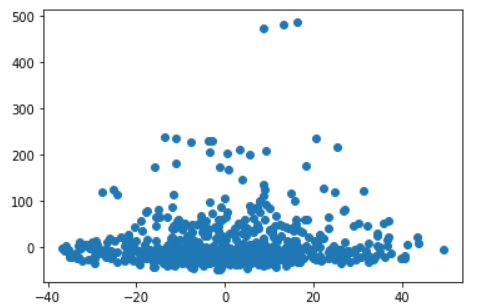
np.random.seed(42)

noisy = X

noisy[:, 0] = np.random.normal(X[:, 0], 4)

noisy[:, 1] = np.random.normal(X[:, 1], 4)

plt.scatter(noisy[:, 0], noisy[:, 1]



Як ми бачимо, на даних тепер є багато шуму

Обрахуємо на зашумленних даних PCA

pca = PCA(0.95).fit(noisy)

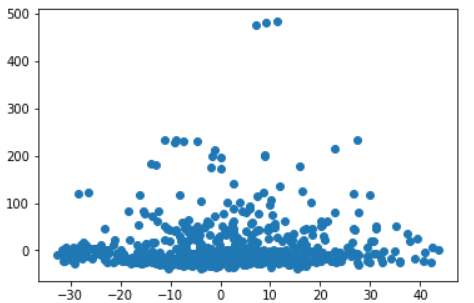
pca.n\_components\_

Та виконаємо обезшумлення

components = pca.transform(noisy)

filtered = pca.inverse\_transform(components)

plt.scatter(filtered[:, 0], filtered[:, 1])



**Висновок**

# На даній лабораторній роботі я ознайомився з principal component analysis алгоритмом. Навчився застосовувати його за допомогою бібліотеки sklearn та реалізовувати його власноруч, використовуючи інструменти pytorch. Також навчився проводити зашумлення та знешумлення.