ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

Мета роботи: *використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи неконтрольованої класифікації даних у машинному навчанні*.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Теоретичні відомості подані на лекціях. Також доцільно вивчити матеріал поданий в літературі:

Джоши Пратик. Искусственный интеллект с примерами на Python. : Пер. с англ. - СПб. : ООО "Диалектика", 2019. - 448 с. - Парал. тит. англ. ISBN 978-5-907114-41-8 (рус.)

Можна використовувати Google Colab або Jupiter Notebook.

Термін *навчання без вчителя (unsupervised learning)* відноситься до процесу побудови моделі машинного навчання, що не вимагає залучення розмічених тренувальних даних. Машинне навчання без вчителя знаходить застосування у багатьох галузях, включаючи сегментування ринку, торгівля акціями, обробка природної мови, машинний зір та ін.

У попередніх ЛР ми мали справу з даними, з якими асоціювалися позначки (маркери). У разі помічених навчальних даних алгоритми вчаться класифікувати по цих мітках.

Алгоритми навчання без вчителя намагаються будувати моделі, які здатні знаходити підгрупи в заданому наборі даних, використовуючи різні метрики подібності.

Розглянемо, як формулюється завдання навчання, якщо воно проводиться без учителя. Коли у нас є набір даних, які не асоціюються з будь-якими мітками, ми припускаємо, що ці дані генеруються під впливом прихованих змінних, що управляють їх розподілом. У такому разі процес навчання може наслідувати якусь ієрархічну схему, використовуючи на початковому етапі індивідуальні точки даних. Далі можна створювати більш глибокі рівні представлення даних.

2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

Завдання 2.1. Кластеризація даних за допомогою методу k-середніх

Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data_clustering.txt.

РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Кластеризація - один із найпопулярніших методів навчання без вчителя. Ця методика застосовується для аналізу даних та виділення кластерів серед них. Для знаходження кластерів застосовують різні метрики подібності, такі як евклідова відстань, що дозволяють виділяти підгрупи даних. Використовуючи міру подібності, можна оцінити складність кластера. Таким чином, кластеризація - це процес організації даних у підгрупи, елементи яких подібні між собою відповідно до деяких критеріїв.

Наше завдання полягає в тому, щоб ідентифікувати приховані властивості точок даних, що визначають їхню приналежність до однієї і тієї ж підгрупи. Універсальних метричних параметрів подібності, які б у всіх випадках працювали, не існує. Все визначається конкретикою завдання. Наприклад, нас може цікавити знаходження представницької точки даних для кожної підгрупи або викидів. Залежно від ситуації ми вибираємо ту або іншу метрику, яка, на нашу думку, найбільш повно навчає специфіку завдання.

Метод k-середніх (k-means) - це добре відомий алгоритм кластеризації. Його використання передбачає, що кількість кластерів заздалегідь відома. Далі ми сегментуємо дані до підгруп, застосовуючи різні атрибути даних. Ми починаємо з того, що фіксуємо кількість кластерів та, виходячи з цього, класифікуємо дані. Основна ідея полягає в оновленні положень центроїдів (центрів тяжіння кластеру, або головні точки) на кожній ітерації. Ітеративний процес продовжується до тих пір, поки всі центроїди не займуть оптимального положення.

Як неважко здогадатися, у цьому алгоритмі вибір початкового розташування центроїдів відіграє дуже важливу роль, оскільки це безпосередньо впливає на кінцеві результати. Одна із стратегій полягає в тому, щоб центроїди розташовувалися на якомога більшій відстані один від одного. Базовому методу k-середніх відповідає випадкове розташування центроїдів, тоді як у вдосконаленому варіанті методу (k-means++) ці точки вибираються алгоритмічно з списку вхідних точок даних. На початку процесу робиться спроба розташувати центри кластерів на великих відстанях один від одного, щоб забезпечити швидку збіжність. Потім ми перебираємо

дані навчального набору та покращуємо стартове розбиття на кластери за допомогою віднесення кожної точки до найближчого кластерного центру.

Завершення перебору всіх точок набору даних означає закінчення першої ітерації. У цьому етапі точки виявляються згрупованими виходячи з початкових положень центрів кластерів. Далі нам необхідно заново вирахувати положення центроїдів, відштовхуючись від нових кластерів, отриманих наприкінці першої ітерації. Отримавши новий набір до центрів, ми повторюємо весь процес, знову ітеруючи по набору даних і відносячи кожну точку до найближчого центроїду.

У процесі повторення описаних кроків центри кластерів поступово зміщуються до своїх стійких положень. Після виконання певної кількості ітерацій центри кластерів перестануть зміщуватися. Це свідчить про те, що ми досягли сталого розташування центрів кластерів. Отримані ***K*** центроїдів і являють собою остаточну модель k-середніх, які будуть використовуватися для виведення суджень (inference).

Щоб подивитися, як працює метод кластеризації k-середніх, застосуємо його до двовимірних даних. Будемо використовувати дані, що містяться у файлі data_clustering txt. У цьому файлі кожен рядок містить два числа, розділені комою.

У scikit-learn K-means реалізується як об’єкт кластера, який називається sklearn.cluster.KMeans, і використовується для пошуку кластерів.

Створіть новий файл Python та імпортуйте такі пакети.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics
```

Завантажимо вхідні дані із файлу.

```
# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')
```

Щоб застосувати k-середніх необхідно задати кількість кластерів

```
num_clusters = 5
```

Візуалізуйте вхідні дані, щоб побачити, як виглядає розподіл.

```
# Включення вхідних даних до графіка
```

```
plt.figure()
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none'
            edgecolors='black', s=80)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Входные данные')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
```

Графік занесіть у звіт.

Ми можемо отримати наочне підтвердження, що наші дані складаються з п'яти груп. Створимо об'єкт KMeans, використовуючи параметри ініціалізації.

Параметр init дозволяє встановити спосіб ініціалізації початкових центрів кластерів. Замість того, щоб вибирати їх випадковим чином, ми використовуємо для цього параметра значення k-means++, яке забезпечує покращений спосіб вибору положень центроїдів, що гарантує швидку збіжність алгоритму. Параметр n_clusters визначає кількість кластерів, тоді як параметр n_init дозволяє вказати, скільки разів повинен виконатися алгоритм, перш ніж буде прийнято рішення щодо найкращого результату.

Створення об'єкту KMeans
```
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)
```

Навчимо модель k-середніх на вхідних даних.

Навчання моделі кластеризації KMeans
```
kmeans.fit(X)
```

Щоб візуалізувати межі, ми маємо створити сітку точок та обчислити модель на всіх вузлах сітки. Визначимо крок сітки.

Визначення кроку сітки
```
step_size = 0.01
```

Далі визначимо саму сітку і переконаємось у тому, що вона охоплює всі вхідні значення.

#Відображення точок сітки
```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_vals, y_vals = np.meshgrid(np.arange(x_min, x_max, step_size),
        np.arange(y_min, y_max, step_size))
```

Спрогнозуйте результати всіх точок сітки, використовуючи навчену модель k-середніх.

Передбачення вихідних міток для всіх точок сітки
```
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
```

Відобразіть на графіку вихідні значення та виділіть кожну область своїм кольором.

Графічне відображення областей та виділення їх кольором
```
output = output.reshape(x_vals.shape)
plt.figure()
plt.clf()
plt.imshow(output, interpolation='nearest',
           extent=(x_vals.min(), x_vals.max(),
                   y_vals.min(), y_vals.max()),
           cmap=plt.cm.Paired,
           aspect='auto',
           origin='lower')
```

Відобразіть вхідні дані на виділених кольором областях.

Відображення вхідних точок
```
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none',
            edgecolors='black', s=80)
```

Відобразіть на графіку центри кластерів, отримані з використанням методу k-середніх.

Відображення центрів кластерів
```
cluster_centers = kmeans.cluster_centers_
plt.scatter(cluster_centers[:,0], cluster_centers[:,1],
            marker='o', s=210, linewidths=4, color='black',
            zorder=12, facecolors='black')
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Границы кластеров')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

Ваш графік занесіть у звіт.

Збережіть код робочої програми під назвою LR_4_task_1.py

Код програми, графік функції та результати оцінки якості занесіть у звіт.

Зробіть висновок

Завдання 2.2. Кластеризація K-середніх для набору даних Iris

Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина пелюстки. У цьому завданні використовуйте sklearn.cluster.KMeans для пошуку кластерів набору даних Iris.

РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Можна написати свій власний код програми з поясненнями по аналогії з попереднім завданням або скористатися підказками, але у цьому випадку ви повинні прокоментувати кожен рядок чи функцію коду де є позначка #.

Код підказки (містить помилки):

```
sklearn.svm import SVC
from sklearn.metrics import pairwise_distances_argmin

import numpy as np
iris = load_iris()
X = iris['data']
y = iris['target']
#
sklearn.cluster.KMeans(n_clusters = 8, init = 'k-means + +', n_init = 10, max_iter =
300, tol = 0.0001, precompute_distances = 'auto', verbose = 0, random_state =
None, copy_x = True, n_jobs = None, algorithm = 'auto')
#
kmeans = KMeans(n_clusters = 5)
#
kmeans.fit(X)
#
y_kmeans = kmeans.predict(X);
#
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 50, cmap = 'viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 200, alpha = 0.5);

#
def find_clusters(X, n_clusters, rseed = 2):
#
```

```
rng = np.random.RandomState(rseed)
i = rng.permutation(X.shape[0])[:n_clusters]
centers = X[i]
while True:
#
labels = pairwise_distances_argmin(X, centers)
#
new_centers = np.array([X[labels == i].mean(0)
for i in range(n_clusters)])
#
if np.all(centers == new_centers):
break
centers = new_centers

return centers, labels
centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c = labels,
s = 50, cmap = ‘viridis’);
#
centers, labels = find_clusters(X, 3, rseed = 0)
plt.scatter(X[:, 0], X[:, 1], c = labels,
s = 50, cmap = ‘viridis’);
#
labels = KMeans(3, random_state = 0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c = labels,
s = 50, cmap = ‘viridis’);
```

Збережіть код робочої програми з обов’язковими коментарям під назвою LR_4_task_2.py

Код програми та рисунок занесіть у звіт.

Зробіть висновок

Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву середнього

Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt.

РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Метод зсуву середнього (Mean Shift) - потужний алгоритм, що використовується в навчанні без вчителя. Цей непараметричний алгоритм часто застосовується при вирішенні завдань кластеризації. Він називається не параметричним, оскільки в ньому не використовуються будь-які припущення щодо базового розподілу даних.

Цей метод контрастує з параметричними підходами, у яких передбачається, що базові дані підпорядковуються стандартному розподілу ймовірностей. Метод зсуву середнього знаходить безліч застосувань у таких областях, як відстеження об'єктів та аналіз даних у реальному часі.

У алгоритмі зсуву середнього весь простір ознак сприймається як функція розподілу ймовірності. Ми починаємо з тренувального набору даних і припускаємо, що ця вибірка відповідає функції розподілу ймовірності. В рамках такого підходу кластери відповідають максимуму базового розподілу. Якщо існують *K* кластерів, то в базовому розподілі існують *K* піків, і метод зсуву середнього ідентифікує ці вершини.

Метою методу зсуву середнього є ідентифікація позицій центрів кластерів. Для кожної точки навчального набору визначається оточуюче її вікно. Потім для цього вікна визначається центроїд і положення вікна оновлюється так, щоб воно відповідало стану нового центроїду. Далі процес повторюється для нового центроїду шляхом визначення вікна навколо нього. У міру продовження описаного процесу ми наближаємось до піку кластера. Кожна точка даних переміщатиметься у напрямку кластера, якому вона належить. Це переміщення здійснюється у напрямку області з більш високою щільністю ймовірності.

Ми продовжуємо процес зміщення центроїдів, що також звуться середніми, до піків кожного кластера. Оскільки середні при цьому зміщуються, метод і називається *зсув середнього*. Цей процес триває до того часу, поки алгоритм не зійдеться, тобто. поки центроїди не перестануть зміщуватися.

Створіть новий файл Python та імпортуйте такі пакети.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle
```

Завантажимо вхідні дані.

```
# Завантаження
X = np.loadtxt('data_clustering.txt', delimiter=',')
```

Зверніть увагу на ширину вікна вхідних даних. *Ширина вікна* (bandwidth) - це параметр базового процесу оцінки щільності розподілу ядра в алгоритмі зсуву середньою. Ширина вікна впливає на загальну швидкість збіжності алгоритму та результуючу кількість кластерів. Отже, цей параметр відіграє важливу роль. Вибір занадто малої ширини вікна може призвести до занадто великої кількості кластерів, тоді як завищені значення цього параметра призводять до злиття окремих кластерів.

Параметр quantile впливає на ширину вікна. Вищі значення цього параметра збільшують ширину вікна, тим самим зменшуючи кількість кластерів.

```
# Оцінка ширини вікна для X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))
```

Навчимо модель кластеризації на основі зсуву середнього, використовуючи отриману оцінку ширини вікна.

```
# Кластеризація даних методом зсуву середнього
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)
```

Витягнемо центри всіх кластерів.

```
# Витягування центрів кластерів
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)
```

Витягнемо кількість кластерів.

```
# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
print("\nNumber of clusters in input data =", num_clusters)
```

Візуалізуємо точки даних.

```
# Відображення на графіку точок та центрів кластерів
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    # Отображение на графике точек, принадлежащих
    # текущему кластеру
    plt.scatter(X[labels==i, 0], X[labels==i, 1], marker=marker,
                color='black')
```

Відобразимо на графіку центр поточного кластера.

```
# Відображення на графіку центру кластера
cluster_center = cluster_centers[i]
plt.plot(cluster_center[0], cluster_center[1], marker='o',
         markerfacecolor='black', markeredgecolor='black',
         markersize=15)
plt.title('Кластеры')
plt.show()
```

Після виконання цього коду на екрані відобразиться графік. У вікні термінала відобразяться координати центрів кластерів.

Збережіть код робочої програми з обов'язковими коментарям під назвою LR_4_task_3.py

Код програми та рисунок занесіть у звіт.

Зробіть висновок

Завдання 2.4. Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

Використовуючи модель поширення подібності, знайти підгрупи серед учасників фондового ринку. У якості керуючих ознак будемо використовувати варіацію котирувань між відкриттям і закриттям біржі.

Використовувати файл вхідних даних фондового ринку, що доступний в бібліотеці matplotlib. Прив'язки символічних позначень компаній до повних назв містяться у файлі company_symbol_mapping.json.

РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Поширення подібності (affinity propagation) - це алгоритм кластеризації, виконання якого вимагає попередньої вказівки використовуваної кількості кластерів . У силу своєї спільності та простоти реалізації він широко застосовується у різних галузях. Цей алгоритм знаходить представницькі елементи кластерів, що звуться зразки (exemplars), використовуючи техніку "обміну повідомленнями " між точками даних.

Ми починаємо з визначення мір подібності, які має використовувати алгоритм. Спочатку як потенційні зразки розглядаються всі навчальні точки даних. Далі точки даних «спілкуються» між собою до тих пір поки не вдасться визначити представницький набір зразків.

Елементи кластерів попарно обмінюються повідомленнями двох категорій, що містять інформацію про придатність (responsibility) та доступність (availability) елементів для ролі зразків. Повідомлення першої категорії надсилаються елементами кластера потенційним зразкам і вказують на те, наскільки добре точка даних підходила б для того, щоб бути елементом кластера даного зразка. Повідомлення другої категорії надсилаються потенційними зразками потенційним елементам кластера і вказують на те, наскільки добре вони підійшли б для того, щоб служити зразком. Цей процес триває до того часу, поки алгоритм не зійдеться оптимального набору зразків.

Також є параметр preference, що управляє кількістю зразків, які повинні бути знайдені. Якщо ви виберете для нього завищене значення, це призведе до того, що алгоритм знайде занадто велику кількість кластерів. Наслідком заниженого значення цього параметра буде занадто мала кількість кластерів.

Створіть новий файл Python та імпортуйте такі пакети.

```
import datetime
import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn import covariance, cluster
from matplotlib.finance import quotes_historical_yahoo_ochl
as quotes_yahoo


# Вхідний файл із символічними позначеннями компаній
input_file = 'company_symbol_mapping.json'
```

Завантажте з файлу масив відповідності символів компаній їх повним назвам.

Завантаження прив'язок символів компаній до їх повних назв

```
with open(input_file, 'r') as f:
    company_symbols_map = json.loads(f.read())

symbols, names = np.array(list(company_symbols_map.items())).T
```

Завантажте дані котирувань із бібліотеки matplotlib.

```
# Завантаження архівних даних котирувань
start_date = datetime.datetime(2003, 7, 3)
end_date = datetime.datetime(2007, 5, 4)
quotes = [quotes_yahoo(symbol, start_date, end_date,
          asobject=True) for symbol in symbols]
```

Обчисліть різниці між котируваннями при відкритті та закритті біржі.

```
# Вилучення котирувань, що відповідають
# відкриттю та закриттю біржі
opening_quotes = np.array([quote.open for quote in
quotes]).astype(np.float)
closing_quotes = np.array([quote.close for quote in
quotes]).astype(np.float)

# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes
```

Нормалізуйте дані.

```
X = quotes_diff.copy().T
X /= X.std(axis=0)
```

Створіть модель графа.

```
# Створення моделі графа
edge_model = covariance.GraphLassoCV()
```

Навчимо модель.

```
# Навчання моделі
with np.errstate(invalid='ignore'):
    edge_model.fit(X)
```

Створіть модель кластеризації на основі поширення подібності, використовуючи щойно навчену крайову модель.

Створення моделі кластеризації на основі поширення подібності

```
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()
```

Виведіть результати.

```
for i in range(num_labels + 1):
    print("Cluster", i+1, "==>", ', '.join(names[labels == i]))
```

Код програми та результати занесіть у звіт.

Зверніть увагу що отримані кластери можуть відрізнятися у різних студентів для різних налаштувань.

Програмний код збережіть під назвою LR_4_task_4.py

Коди комітити на GitHab. У кожному звіті повинно бути посилання на GitHab.

Назвіть бланк звіту СШІ-ЛР-4-NNN-XXXXX.doc
де NNN – позначення групи
XXXXX – позначення прізвища студента.

Переконвертуйте файл звіту в СШІ-ЛР-4-NNN-XXXXX.pdf