Кафедра інформаційних систем

Пояснювальна записка

для курсового проекту на тему:

“Візуалізація кластеризації з використанням метода expectation maximization”

студента групи КН-309 Скворцова І. В.

Керівник проекту Засоба Є.О. (підпис)_____

Завідувач кафедри Шаховська Н.Б. (підпис)_____

“__”_____20__р.

Вступ

Кластеризація(eng. Clustering) - це процес поділу популяції або точок на декілька груп таких, що точки в одній групі є більш подібними один до одного, ніж до точок в інших групах. Іншими словами, мета кластеризації полягає в тому, що сегрегувати точки зі схожими властивостями в окремі групи(кластери).

Кластерний аналіз, сам по собі, не є безпосереднім алгоритмом, а натомість постає у вигляді окремої задачі, яку потрібно розв’язати. Існує досить велика кількість різноманітних алгоритмів, що відрізняються в розумінні того, що являє собою кластер і в тому, як ефективно їх знаходити. Деякими з популярних понять кластера є: групи з малими відстанями між членами кластера, щільні області простору даних, інтервали чи певні статистичні розподіли. Таким чином, кластеризацію можна сформувати як багатозадачну проблему оптимізації. Так відповідний алгоритм кластеризації та параметри налаштувань залежать від на вхідних даних та на призначенні вихідних даних. Кластерний аналіз як такий - це ітеративний процес пошуку

знань та багатоцільової оптимізації, що триває допоки результат не досягне бажаних властивостей.

Кластеризація належить до “без наглядових”(unsupervised) навчальних методів, що означає, що він оперує на не класифікованих і/або на не помічених даних і які не потребують постійного нагляду.

Розділ 1.

Сфера використання кластерного аналізу та його типи

У напрямку Data science кластерний аналіз застосовується для того, щоб отримати цінну інформацію з вхідних даних про те, які групи дата пойнти утворюють. Іншими словами, кластерний аналіз визначає внутрішнє групування між елементами в неозначеній системі.

Кластеризація може бути поділена на дві підгрупи:

- **Жорстка кластеризація:** В жосткій кластеризації, кожна точка або належить до одного кластера або до іншого.

  Так, ймовірність того, що точка належить до кластера до кластера a: P(a) = 0 || 1.

- **М’яка кластеризація**: в М’якій кластеризації кожна точка буде належати певному кластеру з певною ймовірністю 0 <= P(a) <= 1. Також важливою деталлю є той факт, що одна точка може належати до декількох кластерів з різною ймовірністю/вагою.

Розділ 2.

Mixture models and the Gauss mixture model(GMM).

Сумішна модель(mixture model, MM) - це ймовірнісна модель, призначена для представлення під популяцій без потреби попередньої класифікації елементів. MM визначає під популяцію, до якої належить спостережуваний об’єкт. Стандартно MM відповідає розподілу суміші(mixture distribution), який, в свою чергу, відповідає ймовірнісному розподілу(probability distribution).

Є три основні типи MM:

- Гауссова(Gaussian mixture model)
- Мультиваріативна(Multivariate mixture model)
- Категоральна(Categorical mixture model)

В даній роботі буде розглядатися Гауссова модель, оскільки її результати є найбільш корисними в реальному житті.

Гауссова модель(GMM) - це модель, що використовує нормальний імовірнісний розподіл і вимагає апроксимації середнього значення та відхилення для кожного кластера.

Розділ 3.

Використання алгоритму Expectation maximization(EM) в GMM і як він працює.

Для знаходження середнього значення(μ) та відхилення(σ) буде використаний алгоритм максимізації очікування(expectation maximization algorithm).

Алгоритм максимізації очікування - це алгоритм, що відноситься до підгрупи м'яких алгоритмів кластеризації та використовується для максимізації оцінки ймовірності, що застосовується за присутності невідомих параметрів(latent variables) в Вибірці даних. В даному випадку, невідомим параметром виступають закони/функції, за якими генеруються точки.

Алгоритм максимізації очікування є, за своєю природою, ітеративним алгоритмом, що при кожній ітерації покращує вихідний результат, іншими словами, для цієї задачі, наближає значення параметрів середнього значення(μ) та відхилення(σ) кожного кластера до істинних, допоки не досягне достатньої точної відповіді.

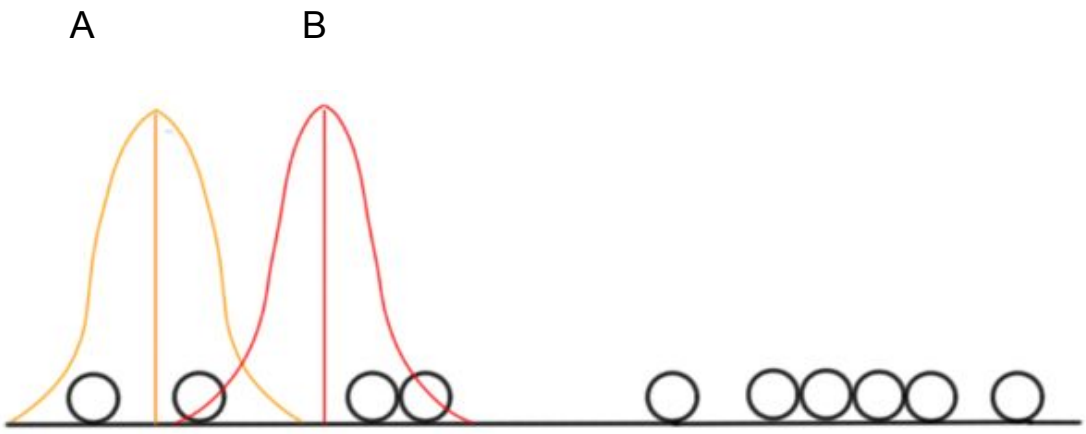
Під час виконання Алгоритм максимізації очікування оперує у двох режимах:

- E-step
- M-step

Та постійно переходить з одного в інший.

Нехай дана певна кількість точок на координатній прямій(розглядатиметься одновимірний приклад заради полегшення пояснення та підвищення зрозумілості) і мета задачі полягає в тому, щоб поділити дані точки на два кластери, K = 2. Нехай A - перший кластер, B - другий кластер.

Тоді першим кроком в даному алгоритмі буде довільно розмістити кластери на координатній прямій та вказати їхні середні значення та відхилення.



Далі почнеться перша ітерація:

E - step:

Основна задача цього етапу - це знаходження невідомих параметрі системи, в даному випадку належності кожної точки до одного з двох даних кластерів.

Оскільки немає можливості визначити належність точки до кластера з абсолютною точністю, то натомість буде здійснюватися пошук ймовірності належності кожної точки до кожного кластера.

Цей пошук буде здійснюватися за даними формулами:

$$P(x_i|b) = \frac{1}{\sqrt{2\pi^2}} e^{(-\frac{(x_i - \mu_b^2)}{2_b^2})}$$

Дана формула формула, фактично, каже:"Яка ймовірність того, що дана точка x_i належить до кластера B."

$$b_i = P(b|x_i) = \frac{P(x_i|b)P(b)}{P(x_i|b)P(b) + P(x_i|a)P(a)}$$

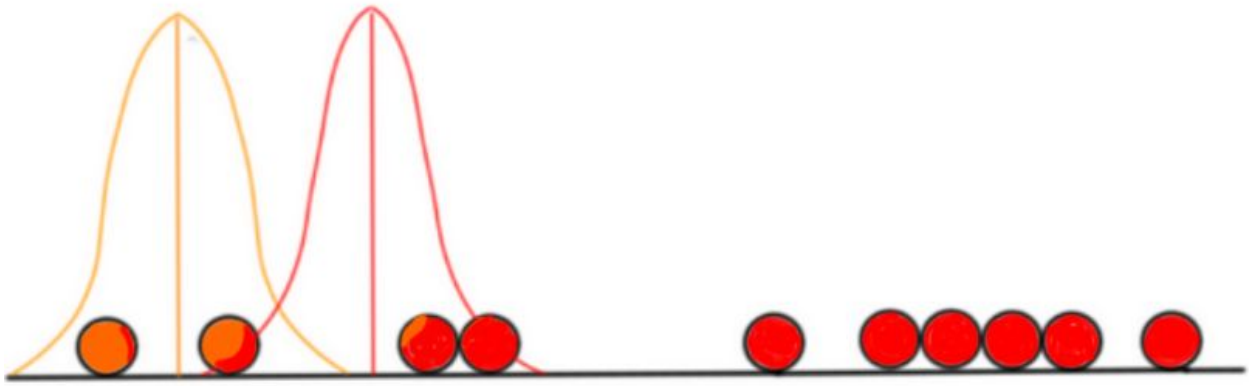
b_i - це ймовірність того, що кластер B містить в собі точку x_i.

Ймовірність a_i рахується як різниця між загальним полем ймовірностей та ймовірністю того, що кластер B містить дані точку.

$$a_i = P(b|x_i) = 1 - P(b|x_i)$$

Такі розрахунки повинні бути проведені для кожної точки.

Після того, як було визначено для всіх b_i та a_i, стануть помітними ймовірності належності точок до кластерів.:



M-step:

На даному етапі максимізуються параметри моделі(середнє значення(μ) та відхилення(σ)) відносно визначених під час попереднього етапу даних.

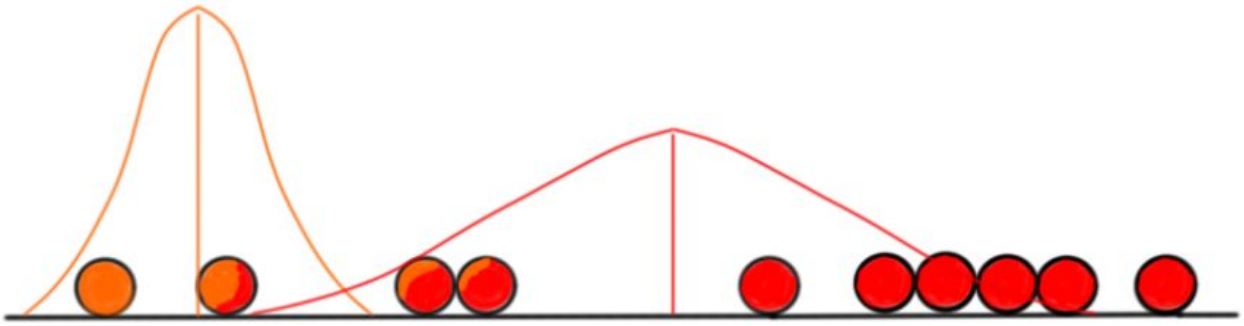
Середнє значення шукається за наступною формулою:

$$\mu_b = \frac{\sum_{i=1}^{n} b_i x_i}{\sum_{i=1}^{n} b_i}$$

Та відхилення:

$$\frac{2}{b} = \frac{\sum_{i=1}^{n} b_i(x_i - \mu_i)}{\sum_{i=1}^{n} b_i}$$

Після зміни значень параметрів моделі графічне зображення прикладу набуде наступного значення:



Цей процес переходу від етапу знаходження невідомих параметрі системи до максимізації параметрів моделі буде тривати доки не буде досягнуто бажаної точності. Іншими словами, процес триватиме доки не виконаються умови:

$\Delta\mu = \mu_n - \mu_{n-1} < e$

та

$\Delta = {}_n - {}_{n-1} < e$, де e - точність.

Розділ 4.
Реалізація

EM_clustering.py

```
import numpy as np
from sklearn.mixture import GaussianMixture
import visualization

N, D = 1000, 3

if D == 3:
  # set gaussian centers and covariances in 3D
  means = np.array([[0.5, 0.0, 0.0],
                    [0.0, 0.0, 0.0],
                    [-0.5, -0.5, -0.5],
                    [-0.8, 0.3, 0.4]])
  covs = np.array([np.diag([0.01, 0.01, 0.03]),
                   np.diag([0.08, 0.01, 0.01]),
                   np.diag([0.01, 0.05, 0.01]),
                   np.diag([0.03, 0.07, 0.01])])
n_gaussians = means.shape[0]


# generate points using a multivariate normal distribution for
points = []
for i in range(len(means)):
  x = np.random.multivariate_normal(means[i], covs[i], N )
  points.append(x)
points = np.concatenate(points)


# Fitting the Gaussian Mixture Model
# fit the gaussian model
gmm = GaussianMixture(n_components=n_gaussians, covariance_type='diag')
gmm.fit(points)

if D == 3:
```

```
  visualization.visualize_3d_gmm(points, gmm.weights_, gmm.means_.T,
np.sqrt(gmm.covariances_).T)
```

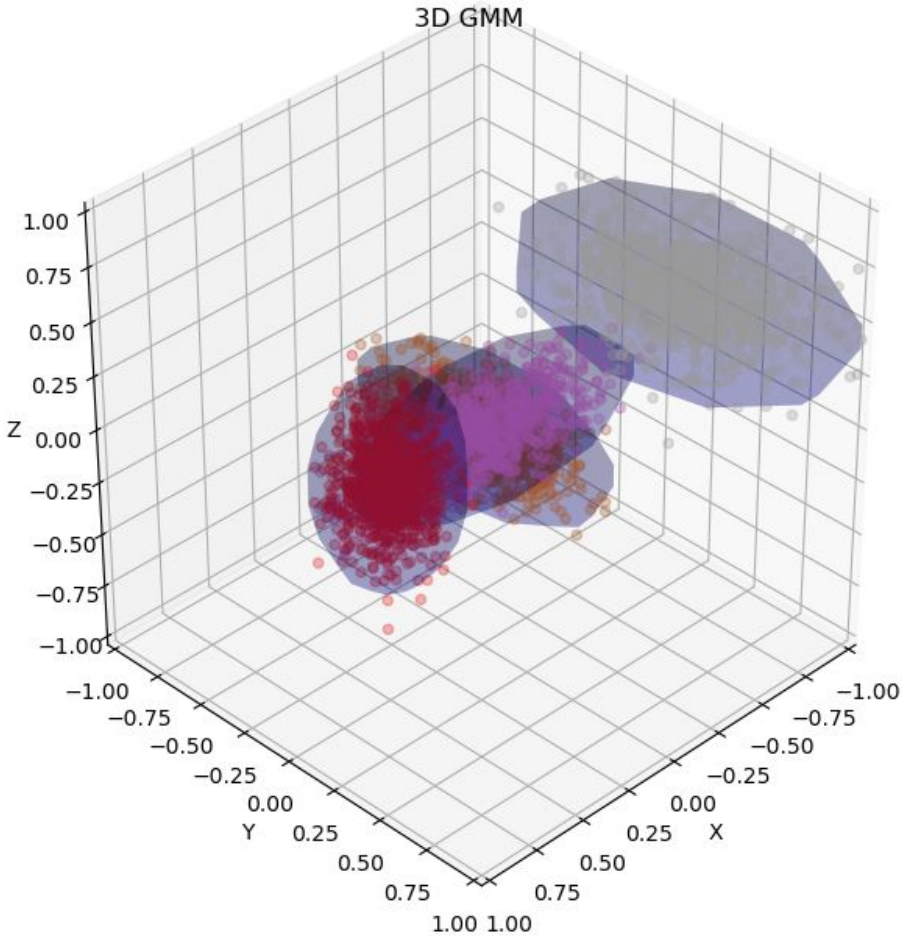
visualization.py

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cmx
import os


def visualize_3d_gmm(points, w, mu, stdev, export=True):
  '''
  plots points and their corresponding gmm model in 3D
  Input:
    points: N X 3, sampled points
    w: n_gaussians, gmm weights
    mu: 3 X n_gaussians, gmm means
    stdev: 3 X n_gaussians, gmm standard deviation (assuming diagonal covariance
matrix)
  Output:
    None
  '''

  n_gaussians = mu.shape[1]
  N = int(np.round(points.shape[0] / n_gaussians))
  # Visualize data
  fig = plt.figure(figsize=(8, 8))
  axes = fig.add_subplot(111, projection='3d')
  axes.set_xlim([-1, 1])
  axes.set_ylim([-1, 1])
  axes.set_zlim([-1, 1])
  plt.set_cmap('Set1')
  colors = cmx.Set1(np.linspace(0, 1, n_gaussians))
  for i in range(n_gaussians):
    idx = range(i * N, (i + 1) * N)
```

```
        axes.scatter(points[idx, 0], points[idx, 1], points[idx, 2], alpha=0.3, c=colors[i])
        plot_sphere(w=w[i], c=mu[:, i], r=stdev[:, i], ax=axes)

    plt.title('3D GMM')
    axes.set_xlabel('X')
    axes.set_ylabel('Y')
    axes.set_zlabel('Z')
    axes.view_init(35.246, 45)
    if export:
        if not os.path.exists('images/'): os.mkdir('images/')
        plt.savefig('images/3D_GMM_demonstration.png', dpi=100, format='png')
    plt.show()


def plot_sphere(w=0, c=[0,0,0], r=[1, 1, 1], subdev=10, ax=None, sigma_multiplier=3):
    '''
        plot a sphere surface
        Input:
            c: 3 elements list, sphere center
            r: 3 element list, sphere original scale in each axis ( allowing to draw elipsoids)
            subdiv: scalar, number of subdivisions (subdivision^2 points sampled on the
surface)
            ax: optional pyplot axis object to plot the sphere in.
            sigma_multiplier: sphere additional scale (choosing an std value when plotting
gaussians)
        Output:
            ax: pyplot axis object
    '''

    if ax is None:
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
    pi = np.pi
    cos = np.cos
    sin = np.sin
    phi, theta = np.mgrid[0.0:pi:complex(0,subdev), 0.0:2.0 * pi:complex(0,subdev)]
    x = sigma_multiplier*r[0] * sin(phi) * cos(theta) + c[0]
    y = sigma_multiplier*r[1] * sin(phi) * sin(theta) + c[1]
    z = sigma_multiplier*r[2] * cos(phi) + c[2]
```

```
    cmap = cmx.ScalarMappable()
    cmap.set_cmap('jet')
    c = cmap.to_rgba(w)

    ax.plot_surface(x, y, z, color=c, alpha=0.2, linewidth=1)

    return ax
```

Нижче наведено результат процесу кластеризації:



Висновок

Під час написання цієї курсової роботи та процесу ознайомлення з даною темою, я осягнув спосіб, в який працює алгоритм максимізації очікування та наочно продемонстрував роботу даного алгоритму, шляхом його реалізації на коді. Дані знання мені, однозначно, знадобляться в подальшому навчанні та житті та безпосередня реалізація також стане в нагоду на лабораторних роботах в майбутньому.

Джерела

Expectation Maximization: how it works
EM algorithm: how it works
A Gentle Introduction to Expectation-Maximization (EM Algorithm)
Gaussian Mixture Models Clustering Algorithm Explained
Expectation–maximization algorithm
Mixture model
Clustering in Machine Learning
Лекції з предмету Методи та системи штучного інтелекту.