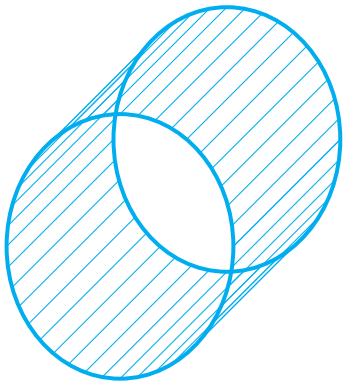
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM

Khoa Toán - Tin Học

BÁO CÁO GIỮA KỲ

K-MEANS CLUSTERING ALGORITHM



Khoa Toán - Tin học

Fac. of Math. & Computer Science

Giảng viên phụ trách: Nguyễn Tấn Trung

TP.Hồ Chí Minh - Tháng 12, 2020

Nhóm sinh viên thực hiện

- Đinh Anh Huy - MSSV : 18110103
- Nguyễn Đức Vũ Duy - MSSV : 18110004

Contents

1 Ý tưởng khởi đầu

Bài toán gom nhóm (Clustering) là một trong những bài toán quan trọng khi chúng ta cần làm việc với máy học (Machine Learning), đặc biệt với việc học không giám sát (Unspervised Learning). Clustering là một kỹ thuật mà cho phép chúng ta chia dữ liệu $\mathcal{X}$ thành các cụm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi cụm. Trong bài toán này, dữ liệu huấn luyện không có nhãn, mô hình tự phân chia dữ liệu thành các cụm khác nhau.

Một ví dụ cho ứng dụng của Clustering là ứng dụng trong kinh doanh. Chúng ta cần dùng máy học để khi chúng ta có một dữ liệu về khách hàng, doanh số, sản phẩm, chúng ta có thể tìm ra những nhóm khách hàng có sở thích tương tự nhau dựa vào những hành vi tiêu dùng chung, từ đó có thể nâng cao doanh số sản phẩm.

Thuật toán K-Means là một trong những thuật toán phổ biến nhất của bài toán Clustering. K-Means được ứng dụng rất nhiều cả ở trong giới học thuật và giới kinh doanh. Để giải thích cho sự phổ biến của nó, thuật toán K-Means dễ thực thi và cũng cho ra tính toán hiệu quả hơn so với các thuật toán gom nhóm khác.

Mục đích của thuật toán K-Means chính là từ dữ liệu đầu vào và một số lượng cụm cần tìm, ta xác định được tâm của mỗi cụm và phân các điểm dữ liệu vào các cụm tương ứng. Ta giả sử rằng, mỗi điểm dữ liệu chỉ thuộc đúng 1 cụm. Chi tiết của thuật toán sẽ được đi sâu hơn ở phần sau.

Một cụm có thể được định nghĩa là tập hợp các điểm có vector đặc trưng gần nhau. Việc tính toán khoảng cách có thể phụ thuộc vào từng loại dữ liệu, trong đó khoảng cách Euclid được sử dụng phổ biến nhất. Trong bài báo cáo này, các tính toán được thực hiện dựa trên khoảng cách Euclid. Tuy nhiên, quy trình thực hiện thuật toán có thể được áp dụng cho các loại khoảng cách khác.

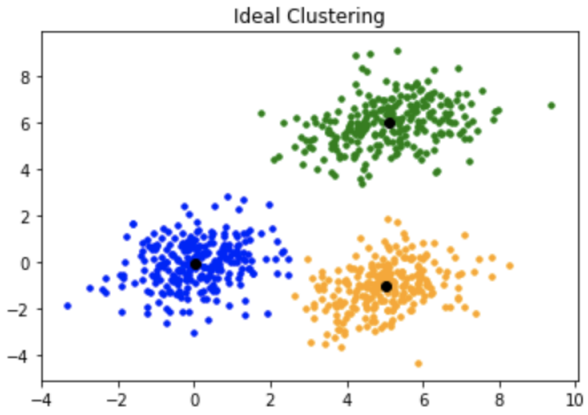


Figure 1: Ví dụ với ba cụm dữ liệu trong không gian hai chiều.

Hình (1) là một ví dụ về dữ liệu được phân vào ba cụm. Giả sử mỗi cụm có một điểm đại diện được gọi là *tâm cụm*, được minh hoạ bởi các điểm màu đen. Mỗi điểm thuộc vào cụm có tâm gần nó nhất.

2 Phân tích toán học

Giả sử N điểm dữ liệu trong tập huấn luyện được ghép lại thành ma trận $\mathbf{X} = [x_1, x_2, ..., x_N] \in \mathbb{R}^{d \times N}$ và K ($K < N$) là số cụm được xác định trước. Ta cần tìm các tâm cụm $\mu_1, \mu_2, ..., \mu_K \in \mathbb{R}^{d \times 1}$ và nhãn của mỗi điểm dữ liệu. Ở đây, mỗi cụm được đại diện bởi một nhãn, thường là một số tự nhiên từ 1 đến K. Nhắc lại rằng các điểm dữ liệu trong bài toán phân cụm K-Means ban đầu không có nhãn cụ thể.

Với mỗi điểm dữ liệu x_i, ta cần tìm nhãn $y_i = k$ của nó, trong đó $k \in \{1, 2, ..., K\}$. Nhãn của một điểm cũng thường được biểu diễn dưới dạng một vector hàng với K phần tử $y_i = (y_{i1}, y_{i2}, ..., y_{iK}) \in \mathbb{R}^{1 \times K}$, trong đó $y_{ij} = 0, \forall j \neq k; \quad y_{ik} = 1$. Khi ghép các vector y_i lại với nhau, ta được một ma trận nhãn $\mathbf{Y} \in \mathbb{R}^{N \times K}$. Khi đó điều kiện của y_i có thể viết dưới dạng toán học như sau

$$y_{ij} \in \{0, 1\}, \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1, \forall i \tag{1}$$

2.1 Hàm mất mát và bài toán tối ưu

Gọi $\mu_k \in \mathbb{R}^d$ là tâm của cụm thứ k. Giả sử một điểm dữ liệu x_i được phân vào cụm k. Vector sai số nếu thay x_i bằng m_k là $(x_i - m_k)$. Ta muốn vector sai số này gần với với vector không, tức là x_i gần với m_k. Việc này có thể được thực hiện thông qua việc làm tối thiểu bình phương khoảng cách Eulid $\|x_i - m_k\|_2^2$. Hơn nữa, vì x_i được phân vào cụm k nên $y_{ik} = 1$, $y_{ij} = 0$, $\forall j \neq k$. Khi đó, biểu thức khoảng cách Euclid có thể được viết lại thành

$$\|x_i - \mu_k\|_2^2 = y_{ik}\|x_i - \mu_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|x_i - \mu_k\|_2^2 \tag{2}$$

Như vậy, sai số trung bình cho toàn bộ dữ liệu sẽ là

$$\mathcal{L}(Y, M) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|x_i - \mu_j\|_2^2 \tag{3}$$

Trong đó $M = [\mu_1, \mu_2, ..., \mu_K] \in \mathbb{R}^{d \times K}$ là ma trận tạo bởi K tâm cụm. Hàm mất mát trong bài toán phân cụm K-Means là $\mathcal{L}(Y, M)$ với ràng buộc như nêu trong (1). Để tìm các tâm cụm và cụm tương úng của mỗi điểm dữ liệu, ta cần giải bài toán tối ưu có ràng buộc

$$\begin{aligned} Y, M &= \underset{Y,M}{argmin} \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|x_i - \mu_j\|_2^2 \\ \text{thoả mãn } & y_{ij} \in \{0, 1\}, \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1, \forall i \end{aligned} \tag{4}$$

2.2 Thuật toán tối ưu hàm mất mát

Bài toàn (4) là một bài toán khó tìm điểm tối ưu vì có thêm các điều kiện ràng buộc. Bài toán này thuộc loại *mix-integer programming* (điều kiện biến là số nguyên) - là loại rất khó tìm nghiệm tối ưu toàn cục. Tuy nhiên, trong một số trường hợp chúng ta vẫn có phương pháp để tìm nghiệm gần đúng. Một kỹ thuật đơn giản và phổ biến để giải bài toán (4) là xen kẽ giải Y và M khi biến còn lại được cố định cho đến khi hàm mất mát hội tụ. Chúng ta sẽ lần lượt giải quyết hai bài toán sau.

2.2.1 Cố định M, tìm Y

Giả sử đã tìm được các tâm cụm, hãy tìm các vector nhãn để hàm mất mát đạt giá trị nhỏ nhất.

Khi các tâm cụm là cố định, bài toán tìm vector nhãn cho toàn bộ dữ liệu có thể được chia nhỏ thành bài toán tìm vector nhãn cho từng điểm dữ liệu x_i như sau

$$\begin{aligned} y_i &= \underset{y_i}{argmin} \frac{1}{N} \sum_{j=1}^{K} y_{ij} \|x_i - \mu_j\|_2^2 \\ \text{thoả mãn } & y_{ij} \in \{0, 1\}, \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1, \forall i \end{aligned} \tag{5}$$

Vì chỉ có một phần tử của vector nhãn y_i bằng 1 nên bài toàn (5) chính là bài toán đi tìm tâm cụm gần điểm x_i nhất

$$j = \underset{j}{argmin} \|x_i - \mu_j\|_2^2 \tag{6}$$

Vì $\|x_i - \mu_j\|_2^2$ là bình phương khoảng cách Euclid từ điểm x_i đến centroid μ_j, ta có thể kết luận rằng *mỗi điểm x_i thuộc vào cụm có tâm gần nó nhất*. Từ đó có thể suy ra vector nhãn của từng điểm dữ liệu.

2.2.2 Cố định Y, tìm M

Giả sử đã biết cụm của từng điểm, hãy tìm các tâm cụm mới để hàm mất mát đạt giá trị nhỏ nhất.

Khi vector nhãn cho từng điểm dữ liệu đã được xác định, bài toán tìm tâm cụm được rút gọn thành

$$m_j = \underset{m_j}{argmin} \frac{1}{N} \sum_{i=1}^{N} y_{ij} \|x_i - \mu_j\|_2^2 \tag{7}$$

Để ý thấy rằng hàm mục tiêu là một hàm liên tục và có đạo hàm xác định tại mọi điểm m_j. Vì vậy, ta có thể tìm nghiệm bằng phương pháp giải phương trình đạo hàm bằng không. Đặt $\ell(m_j) = \frac{1}{N} \sum_{i=1}^{N} y_{ij} \|x_i - \mu_j\|_2^2$,

ta cần giải phương trình sau đây

$$\nabla_{m_j} \ell(m_j) = \frac{2}{N} \sum_{i=1}^{N} y_{ij}(\mu_j - x_i) = 0 \tag{8}$$

$$\Leftrightarrow \mu_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij} x_i \Leftrightarrow \mu_j = \frac{\sum_{i=1}^{N} y_{ij} x_i}{\sum_{i=1}^{N} y_{ij}} \tag{9}$$

Để ý thấy rằng mẫu số chính là tổng số điểm dữ liệu trong cụm j, tử số là tổng các điểm dữ liệu trong cụm j. Nói cách khác, μ_j là cần tìm là trung bình cộng (mean) của các điểm trong cụm j.
Tên gọi *K-Means Clustering* xuất phát từ đây.

3 Tóm tắt thuật toán

Ta có thể tóm tắt thuật toán phân cụm K-Means như sau

Thuật toán K-Means Clustering

Input: Ma trận dữ liệu $X \in \mathbb{R}^{d \times N}$ và số lượng cụm cần tìm $K (K < N)$.
Output: Ma trận tâm cụm $M \in \mathbb{R}^{d \times K}$ và ma trận nhãn $Y \in \mathbb{R}^{N \times K}$.
Các bước thực hiện:

(1) Chọn K điểm bất kỳ trong tập huấn luyện làm các tâm cụm ban đầu.

(2) Phân mỗi điểm dữ liệu vào cụm có tâm gần nó nhất.

(3) Nếu việc phân cụm dữ liệu vào từng cụm ở bước 2 không thay đổi so với vòng lặp trước thì dừng thuật toán.

(4) Cập nhật tâm cụm bằng cách lấy trung bình cộng của các điểm đã được gán vào cụm đó sau bước 2.

(5) Quay lại bước 2.

Thuật toán này sẽ hội tụ sau một số hữu hạn vòng lặp. Thật vậy, dãy số biểu diễn giá trị của hàm mất mát sau mỗi bước là một đại lượng không tăng và bị chặn dưới. Điều này chỉ ra rằng dãy số này phải hội tụ. Để ý thêm nữa, số lượng các phân cụm cho toàn bộ dữ liệu là hữu hạn (khi số cụm K là cố định) nên đến một lúc nào đó, hàm mất mát sẽ không thể thay đổi, và chúng ta có thể dừng thuật toán tại đây.

Nếu tồn tại một cụm không chứa điểm nào, mẫu số trong (8) sẽ bằng không, và phép chia sẽ không thực hiện được. Vì vậy, K điểm bất kỳ trong tập huấn luyện được chọn làm các tâm cụm ban đầu ở bước 1 phải đảm bảo mỗi cụm có ít nhất một điểm. Trong quá trình huấn luyện, nếu tồn tại một cụm không chứa điểm

nào, có hai cách giải quyết. Cách thứ nhất là bỏ cụm đó và giảm K đi một. Cách thứ hai là thay tâm của cụm đó bằng một điểm bất kỳ trong tập huấn luyện, chẳng hạn như điểm xa tâm cụm hiện tại của nó nhất.

4 Phương pháp Elbow - Tìm số lượng cụm tối ưu

Trong khi thực hiện bài toán phân cụm bằng thuật toán K-Means, ta nhận thấy rằng để thực hiện được thuật toán này, ta cần truyền vào số lượng cụm cần tìm. Vậy câu hỏi được đặt ra là *Phải chọn K như thế nào để thuật toán được tối ưu nhất?*

Một trong những cách giúp trả lời cho câu hỏi trên là *phương pháp Elbow*. Đây là phương pháp phổ biến nhất trong bài toán tìm số lượng cụm tối ưu cho bài toán phân cụm K-Means. Trong phương pháp này, ta sử dụng lại tổng bình phương sai số của toàn bộ dữ liệu $SSE = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|x_i - \mu_j\|_2^2$. Tổng SSE đo lường mức độ méo mó của dữ liệu hay còn gọi nó là hệ số *distortion*.

Phương pháp Elbow xem xét tổng SSE như một hàm của số lượng cụm. Theo trực giác, chúng ta có thể nói rằng, khi K tăng thì độ méo mó của dữ liệu hay hệ số *distortion* sẽ càng giảm. Điều này xảy ra do các mẫu dữ liệu sẽ càng gần với các centroids mà chúng được chỉ định. Phương pháp này sử dụng biểu đồ đường như một công cụ hiển thị, từ đây ta sẽ chọn số lượng cụm K sao cho tại đó hệ số *distortion* bắt đầu biến thiên nhiều nhất.

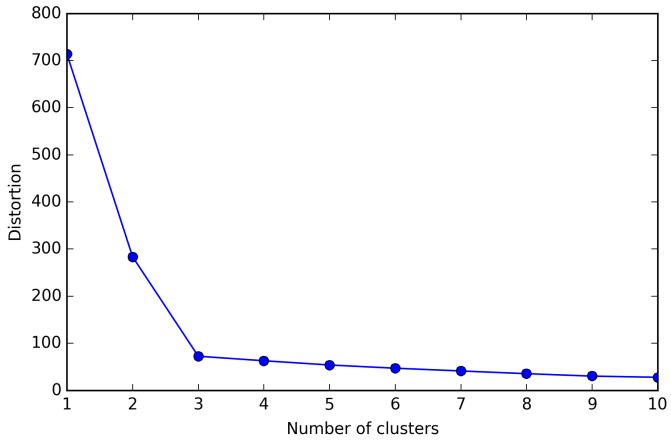


Figure 2: Ví dụ về một dạng biểu đồ elbow.

Như chúng ta có thể thấy trong biểu đồ (2), hệ số *distortion* bắt đầu biến thiên nhiều nhất tại $K = 3$, để ý thấy rằng tại biểu đồ có dạng giống một cánh tay và tại điểm $K = 3$ là khuỷu tay. Tên gọi của phương pháp Elbow cũng bắt nguồn từ đây.

Trong thực tế, không phải lúc nào ta cũng có một bộ dữ liệu tốt để có thể thấy rõ khuỷu tay của biểu đồ. Vì thế phương pháp Elbow trong những trường hợp dữ liệu không đẹp sẽ có hiệu quả không cao. Ngoài phương

pháp Elbow giúp tìm giá trị K tối ưu cho bài toán phân cụm K-Means thì còn một số phương pháp khác như phương pháp *Average Silhouette*, phương pháp *Gap Statistic*,...

5 Thuật toán K-Means++

Trong thuật toán K-Means cổ điển, các điểm tâm cụm ban đầu được chọn ngẫu nhiên từ các điểm dữ liệu. Tuy nhiên, việc chọn ngẫu nhiên các điểm tâm cụm này có thể dẫn đến các cụm không tốt hoặc hội tụ chậm nếu các tâm cụm ban đầu được chọn không tốt. Hình (3) là một ví dụ cho trường hợp các tâm cụm được chọn ban đầu không tốt. Trong hình phía trên, ta có thể quan sát thấy các cụm không được hình thành đúng cách so với các cụm được hình thành ở hình phía dưới.

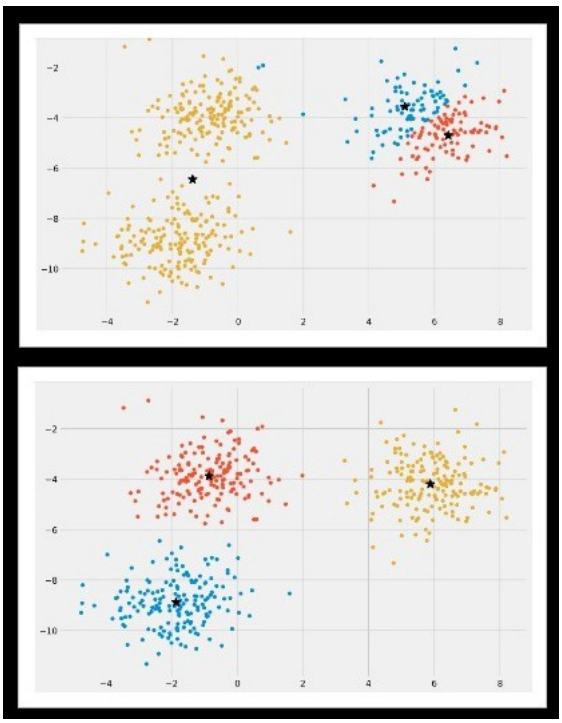


Figure 3: Ví dụ về một dạng phân cụm không tốt.

Một cách để giải quyết vấn đề trên là chạy thuật toán K-Means nhiều lần trên một tập dữ liệu và chọn ra mô hình hoạt động tốt nhất. Một cách khác là đặt các tâm cụm cách xa nhau thông qua thuật toán K-Means++, dẫn đến kết quả tốt hơn và nhất quán hơn K-Means cổ điển.

K-Means++ là một kỹ thuật khởi tạo các tâm cụm thông minh và các bước còn lại của thuật toán hầu như giống như thuật toán K-Means cổ điển. Ta có thể tóm tắt thuật toán này qua các bước thực hiện sau.

Thuật toán K-Means++

Các bước thực hiện:

(1) Khởi tạo một tập rỗng M để lưu trữ K tâm cụm.

(2) Chọn ngẫu nhiên tâm cụm đầu tiền μ_1 từ các mẫu đầu vào và thêm nó vào M.

(3) Đối với mỗi điểm dữ liệu x_i không có trong M, tìm bình phương khoảng cách $d(x_i, M)^2$ đến bất kỳ tâm nào trong M.

(4) Chọn ngẫu nhiên tâm cụm tiếp theo μ_p bằng cách sử dụng phân phối xác suất có trọng số bằng $\frac{d(\mu_p, M)^2}{\sum_i d(x_i, M)^2}$

(4) Lặp lại bước 2 và 3 đến khi tìm đủ K tâm cụm.

(5) Bây giờ các tâm cụm ban đầu đã được chọn, ta tiếp tục thực hiện các bước tiếp theo tương tự thuật toán K-Means cổ điển.

6 Hạn chế của thuật toán K-Means

- Số cụm K cần được xác định trước. Trong trường hợp chúng ta không biết trước giá trị này, ta có thể sử dụng *phương pháp elbow* giúp xác định giá trị K này.
- Nghiệm cuối cùng phụ thuộc vào các tâm cụm được khởi tạo ban đầu. Thuật toán phân cụm K-Means không đảm bảo tìm được nghiệm tối ưu toàn cục, nghiệm cuối cùng phụ thuộc vào các tâm cụm được khởi tạo ban đầu. Trong trường hợp này, thuật toán K-Means++ là giải pháp tối ưu để giải quyết vấn đề trên.

7 Ví dụ trên Python

7.1 Giới thiệu bài toán

Chúng ta sẽ làm một ví dụ đơn giản. Trước hết, ta tạo tâm cụm và dữ liệu cho từng cụm bằng cách lấy mẫu theo phân phối chuẩn có kỳ vọng là tâm của cụm đó và ma trận hiệp phương sai là ma trận đơn vị. Ở đây, hàm **cdist** trong **scipy.spatial.distance** được dùng để tính khoảng cách giữa các cặp điểm trong hai tập hợp một cách hiệu quả.

Dữ liệu được tạo bằng cách lấy ngẫu nhiên 500 điểm cho mỗi cụm theo phân phối chuẩn có kỳ vọng lần lượt

là (2, 2), (8, 3) và (3, 6); ma trận hiệp phương sai giống nhau và là ma trận đơn vị.

```
from __future__ import print_function
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
from matplotlib.backends.backend_pdf import PdfPages
import random
np.random.seed(18)
means = [[2, 2], [8, 3], [3, 6]]
cov = [[1, 0], [0, 1]]
N = 500
X0 = np.random.multivariate_normal(means[0], cov, N)
X1 = np.random.multivariate_normal(means[1], cov, N)
X2 = np.random.multivariate_normal(means[2], cov, N)
X = np.concatenate((X0, X1, X2), axis = 0)
K = 3
original_label = np.asarray([0]*N + [1]*N + [2]*N).T
```

7.2 Các hàm số cần thiết cho phân cụm K-Means

Trước khi viết thuật toán chính phân cụm K-Means, ta cần một số hàm phụ trợ:

- **kmeans_init_centroids** khởi tạo các tâm cụm.
- **kmeans_asign_labels** tìm nhãn mới cho các điểm khi biết các tâm cụm.
- **kmeans_update_centroids** cập nhật các tâm cụm khi biết nhãn của từng điểm.
- **has_converged** kiểm tra điều kiện dừng của thuật toán

```
#Khởi tạo tâm cụm
def kmeans_init_centroids(X, k):
    #Chọn ngẫu nhiên k dòng của X để tạo tâm cụm
    return X[np.random.choice(X.shape[0], k, replace=False)]

#Tìm nhãn mới cho các điểm khi biết tâm cụm
```

```
def kmeans_assign_labels(X, centroids):
    #Tính khoảng cách giữa X và tâm cụm
    D = cdist(X, centroids)
    #Trả về Tâm cụm gần nhất
    return np.argmin(D, axis = 1)

#Kiểm tra tính hội tụ, điều kiện dừng của bài toán
def has_converged(centroids, new_centroids):
    #Trả về True nếu tập hợp 2 tâm cụm là như nhau
    return (set([tuple(a) for a in centroids]) ==set([tuple(a) for a in new_centroids]))

#Cập nhật Tâm cụm khi biết nhãn của từng điểm
def kmeans_update_centroids(X, labels, K):
    centroids = np.zeros((K, X.shape[1]))
    for k in range(K):
        #Tập hợp tất cả các điểm mà ứng với tâm cụm thứ k
        Xk = X[labels == k, :]
        #Tính trung bình
        centroids[k,:] = np.mean(Xk, axis = 0)
    return centroids
```

Phần chính của phân cụm K-Means:

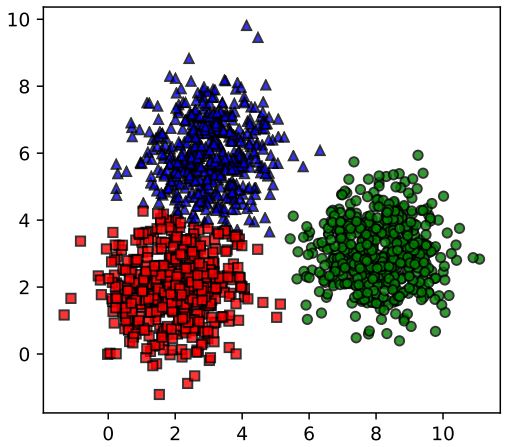
```
def kmeans(X, K):
    centroids = [kmeans_init_centroids(X, K)]
    labels = []
    it = 0
    while True:
        labels.append(kmeans_assign_labels(X, centroids[-1]))
        new_centroids = kmeans_update_centroids(X, labels[-1], K)
        if has_converged(centroids[-1], new_centroids):
            break
        centroids.append(new_centroids)
        it += 1
    return (centroids, labels, it)
```

Áp dụng thuật toán vừa viết vào dữ liệu ban đầu và hiển thị kết quả cuối cùng:

```
(centroids, labels, it) = kmeans(X, K)
print('Centers found by our algorithm:\n', centroids[-1])
kmeans_display(X, labels[-1], 'res.jpg')
```

Kết quả:

```
Centers found by our algorithm:
 [[3.02702878 5.95686115]
 [8.07476866 3.01494931]
 [1.9834967  1.96588127]]
```



Hình (4) minh hoạ thuật toán phân cụm K-means trên tập dữ liệu này sau một số vòng lặp. Nhận thấy rằng tâm cụm và các vùng lãnh thổ của chúng thay đổi qua các vòng lặp và hội tụ chỉ sau sáu vòng lặp. Từ kết quả này ta thấy rằng thuật toán phân cụm K-Means làm việc khá thành công, các tâm cụm tìm được gần với các tâm cụm ban đầu và các nhóm dữ liệu được phân ra gần như hoàn hảo (một vài điểm gần ranh giới giữa hai cụm hình thoi và hình sao có thể lẫn vào nhau).

7.3 Kết quả tìm được bằng thư viện scikit-learn

Để kiểm tra thêm, chúng ta hay so sánh kết quả trên với kết quả thu được bằng cách sử dụng thư viện **scikit_learn**.

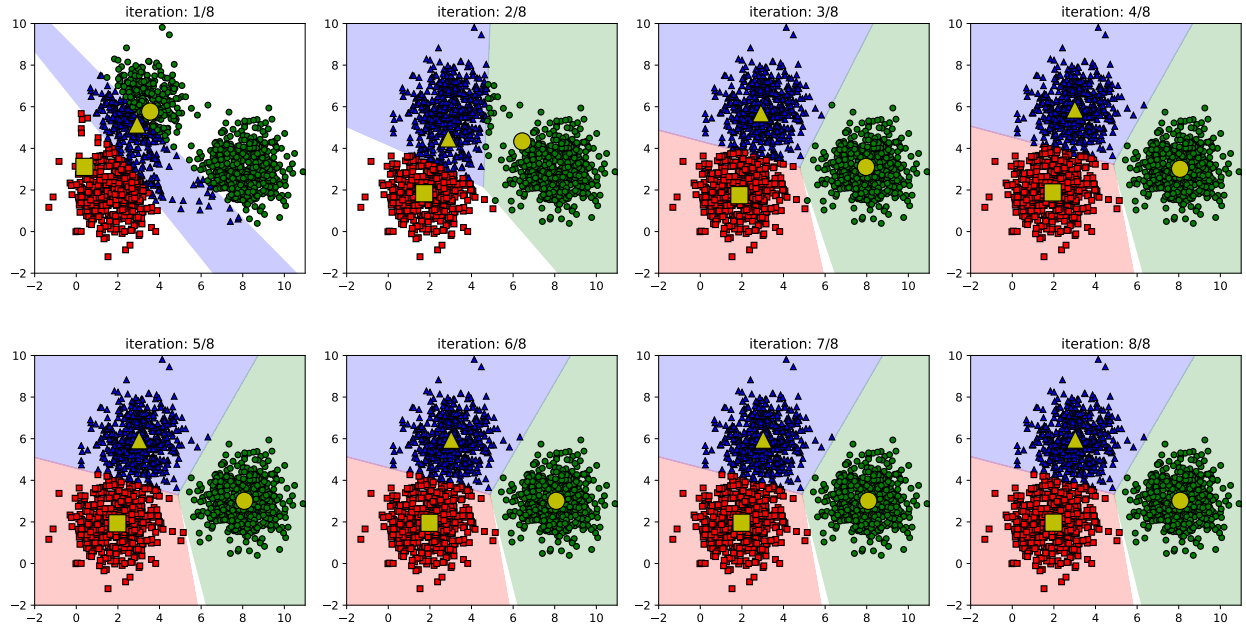
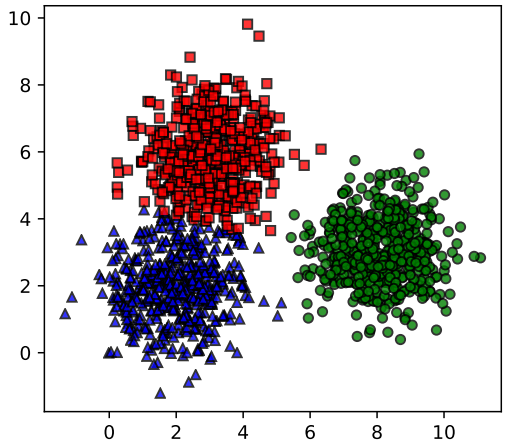


Figure 4: Thuật toán phân cụm K-Means qua các vòng lặp

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3, random_state=0).fit(X)
print('Centers found by scikit-learn:')
print(model.cluster_centers_)
pred_label = model.predict(X)
kmeans_display(X, pred_label, 'res_scikit.pdf')
```

Kết quả:

```
Centers found by scikit-learn:
[[1.98417154 1.96141961]
 [8.07476866 3.01494931]
 [3.02429957 5.95334038]]
```



Ta nhận thấy rằng các tâm cụm tìm được rất gần với kết quả kỳ vọng.