**DỊP KIM YẾN**

**Mã só SV: 21DH112239**

**TRƯỜNG ĐẠI HỌC NGOẠI NGỮ- TIN HỌC TPHCM KHOA CÔNG NGHỆ THÔNG TIN**

**PHÂN VÙNG ẢNH DỰA TRÊN KỸ THUẬT GOM CỤM**

**(CLUSTERING BASED SEGMENTATION)**

**ĐỒ ÁN MÔN HỌC: THỊ GIÁC MÁY TÍNH**

**Giảng viên hướng dẫn:**

**TS.NGUYỄN THANH BÌNH**

**Sinh viên thực hiện:**

**DƯƠNG TẤN ĐẠT**

**Mã số SV: 21DH113556**

**TRƯƠNG HUY HOÀNG**

**Mã số SV: 21DH114404**

**TP.Hồ Chí Minh, tháng 12 năm 2023**



# MỤC LỤC

Contents

[MỤC LỤC 2](#_Toc148035601)

[1. Giới Thiệu 4](#_Toc148035602)

[1.1. Lý do lựa chọn chủ đề 4](#_Toc148035603)

[1.2. Mục tiêu và phạm vi của dự án 4](#_Toc148035604)

[1.3. Tổng quan về phân vùng ảnh và gom cụm 4](#_Toc148035605)

[2. Nền tảng lý thuyết 4](#_Toc148035606)

[2.1. Nguyên lý cơ bản của phân vùng ảnh 4](#_Toc148035607)

[2.2. Gom cụm (Clustering) trong thị giác máy tính 9](#_Toc148035608)

[2.3. Các phương pháp phân vùng ảnh dựa trên gom cụm 11](#_Toc148035609)

[2.4. Những thuật toán phổ biến 12](#_Toc148035610)

[3. Các bước thực hiện 13](#_Toc148035611)

[4. Phương pháp đề xuất 13](#_Toc148035612)

[4.1. K-means Clustering 13](#_Toc148035613)

[4.2. Bài toán phân vùng ảnh dựa trên kỹ thuật gom cụm áp dụng K-means Clustering 24](#_Toc148035614)

[5. Ứng dụng 33](#_Toc148035615)

[6. Kết luận 34](#_Toc148035616)

[6.1. Đánh giá độ hiệu quả và hạn chế của phương pháp 34](#_Toc148035617)

[6.2. Hướng phát triển tương lai 35](#_Toc148035618)

[7. Tài liệu tham khảo 35](#_Toc148035619)

PHỤ LỤC

[Hình 2. 1 Phân vùng dựa trên cạnh 6](#_Toc148035480)

[Hình 2. 2 Phân vùng dựa trên ngưỡng 7](#_Toc148035481)

[Hình 2. 3 Phân vùng theo khu vực 7](#_Toc148035482)

[Hình 2. 4 Phân vùng theo lưu vực 8](#_Toc148035483)

[Hình 2. 5 Phân vùng dựa trên cụm 9](#_Toc148035484)

[Hình 4. 1 Sự dịch chuyển tâm của cụm sau mỗi vòng lặp. 17](#_Toc148035485)

[Hình 4. 2 Đồ thị hàm biến dạng của thuật toán k-Means. 21](#_Toc148035486)

[Hình 4. 3 Biểu diễn của các cụm trên không gian 2D bằng phương pháp giảm chiều dữ liệu 24](#_Toc148035487)

[Hình 4. 4 Vẽ biểu đồ SSE theo số lượng cụm k 28](#_Toc148035488)

[Hình 4. 5 Biểu đồ gom cụm dựa trên K-Means (OXYZ 32](#_Toc148035489)

[Hình 4. 6 ảnh gốc và ảnh đã phân vùng 33](#_Toc148035490)

# Giới Thiệu

## Lý do lựa chọn chủ đề

Trong thời đại số hóa và công nghiệp hóa ngày càng phát triển, xử lý hình ảnh và phân vùng ảnh đang trở thành một phần quan trọng và tương lai của nhiều ứng dụng trong thị giác máy tính. Sự cần thiết của việc tự động phân loại và hiểu hình ảnh đã thúc đẩy sự phát triển nhanh chóng của các kỹ thuật và phương pháp trong lĩnh vực này. Một trong những phương pháp quan trọng để phân tách hình ảnh thành các phần tử có ý nghĩa là "Phân vùng ảnh dựa trên kỹ thuật gom cụm".

## Mục tiêu và phạm vi của dự án

Mục tiêu chính của dự án này là nghiên cứu và phát triển một hệ thống phân vùng ảnh dựa trên kỹ thuật gom cụm, có khả năng chia ảnh thành các khu vực có tính chất hoặc đặc điểm tương tự. Chúng tôi sẽ nghiên cứu các phương pháp và thuật toán gom cụm phổ biến và áp dụng chúng vào việc phân vùng hình ảnh.

Phạm vi của dự án này sẽ bao gồm việc xác định các ứng dụng tiềm năng của phân vùng ảnh dựa trên kỹ thuật gom cụm trong nhiều lĩnh vực khác nhau như thị giác máy tính, y tế, công nghiệp và nhiều lĩnh vực khác.

## Tổng quan về phân vùng ảnh và gom cụm

Phân vùng ảnh là một nhiệm vụ quan trọng trong thị giác máy tính, có nhiệm vụ chia ảnh thành các khu vực có ý nghĩa, giúp máy tính hiểu và xử lý hình ảnh một cách hiệu quả. Gom cụm là một kỹ thuật quan trọng trong phân vùng ảnh, nó tập trung vào việc chia ảnh thành các cụm có tính chất tương tự, dựa trên sự tương đồng giữa các điểm dữ liệu.

Trong phần tiếp theo của dự án, chúng tôi sẽ giới thiệu chi tiết về các phương pháp và thuật toán gom cụm, đồng thời trình bày cách chúng có thể được áp dụng vào việc phân vùng hình ảnh.

# Nền tảng lý thuyết

## Nguyên lý cơ bản của phân vùng ảnh

### *2.1.1. Định nghĩa của phân vùng ảnh*

Phân vùng ảnh là phương pháp chia hình ảnh kỹ thuật số thành các nhóm con, làm giảm độ phức tạp của hình ảnh và cho phép xử lý hoặc phân tích thêm từng phân vùng ảnh.

Một ứng dụng phổ biến của phân vùng ảnh là phát hiện đối tượng. Thay vì xử lý toàn bộ hình ảnh, cách thực hành phổ biến trước tiên là sử dụng thuật toán phân vùng ảnh để tìm các đối tượng quan tâm trong hình ảnh. Sau đó, bộ phát hiện đối tượng có thể hoạt động trên hộp giới hạn đã được xác định bằng thuật toán phân vùng. Điều này ngăn cản máy dò xử lý toàn bộ hình ảnh, cải thiện độ chính xác và giảm thời gian suy luận.

Phân vùng ảnh là một khối xây dựng quan trọng của các thuật toán và công nghệ thị giác máy tính. Nó được sử dụng cho nhiều ứng dụng thực tế bao gồm phân tích hình ảnh y tế, thị giác máy tính cho xe tự hành, nhận dạng và phát hiện khuôn mặt, giám sát video và phân tích hình ảnh vệ tinh.

### *2.1.2. Phân vùng ảnh hoạt động như thế nào?*

Phân vùng ảnh là một chức năng lấy đầu vào hình ảnh và tạo ra đầu ra. Đầu ra là một mặt nạ hoặc một ma trận với nhiều phần tử khác nhau xác định lớp đối tượng hoặc thể hiện của mỗi pixel.

Một số phương pháp phỏng đoán có liên quan hoặc các đặc điểm hình ảnh cấp cao có thể hữu ích cho việc phân vùng ảnh. Các tính năng này là cơ sở cho các thuật toán phân vùng ảnh tiêu chuẩn sử dụng các kỹ thuật gom cụm như cạnh và biểu đồ.

Một ví dụ về phương pháp heuristic phổ biến là màu sắc. Người tạo đồ họa có thể sử dụng màn hình xanh để đảm bảo nền hình ảnh có màu đồng nhất, cho phép phát hiện và thay thế nền theo chương trình trong quá trình xử lý hậu kỳ.

Một ví dụ khác về phương pháp phỏng đoán hữu ích là độ tương phản—các chương trình phân vùng hình ảnh có thể dễ dàng phân biệt giữa hình tối và nền sáng (tức là bầu trời). Chương trình xác định ranh giới pixel dựa trên các giá trị có độ tương phản cao.

Các kỹ thuật phân vùng ảnh truyền thống dựa trên các phương pháp phỏng đoán như vậy có thể nhanh chóng và đơn giản nhưng chúng thường yêu cầu tinh chỉnh đáng kể để hỗ trợ các trường hợp sử dụng cụ thể với các phương pháp phỏng đoán được thiết kế thủ công. Chúng không phải lúc nào cũng đủ chính xác để sử dụng cho những hình ảnh phức tạp. Các kỹ thuật phân vùng mới hơn sử dụng học máy và học sâu để tăng độ chính xác và tính linh hoạt.

Các phương pháp phân vùng ảnh dựa trên học máy sử dụng đào tạo mô hình để cải thiện khả năng của chương trình trong việc xác định các tính năng quan trọng. Công nghệ mạng lưới thần kinh sâu đặc biệt hiệu quả đối với các tác vụ phân vùng ảnh.

Có nhiều thiết kế và triển khai mạng thần kinh khác nhau phù hợp cho việc phân vùng hình ảnh. Chúng thường chứa các thành phần cơ bản giống nhau:

* Bộ mã hóa —một loạt các lớp trích xuất các đặc điểm hình ảnh bằng cách sử dụng các bộ lọc ngày càng sâu hơn, hẹp hơn. Bộ mã hóa có thể được đào tạo trước về một nhiệm vụ tương tự (ví dụ: nhận dạng hình ảnh), cho phép nó tận dụng kiến ​​thức hiện có để thực hiện các nhiệm vụ phân vùng.
* Bộ giải mã —một loạt các lớp chuyển đổi dần dần đầu ra của bộ mã hóa thành mặt nạ phân vùng tương ứng với độ phân giải pixel của hình ảnh đầu vào.
* Bỏ qua kết nối —nhiều kết nối mạng thần kinh tầm xa cho phép mô hình xác định các tính năng ở các quy mô khác nhau để nâng cao độ chính xác của mô hình.

### *2.1.3. Các kỹ thuật phân vùng hình ảnh phổ biến*

* + Phân vùng dựa trên cạnh (Edge-Based Segmentation)

Hình 2. 1 Phân vùng dựa trên cạnh

Phân vùng dựa trên cạnh là một kỹ thuật xử lý hình ảnh phổ biến nhằm xác định các cạnh của các đối tượng khác nhau trong một hình ảnh nhất định. Nó giúp xác định vị trí các đặc điểm của đối tượng liên quan trong ảnh bằng cách sử dụng thông tin từ các cạnh. Tính năng phát hiện cạnh giúp loại bỏ các thông tin dư thừa trên hình ảnh, giảm kích thước của chúng và tạo điều kiện thuận lợi cho việc phân tích.

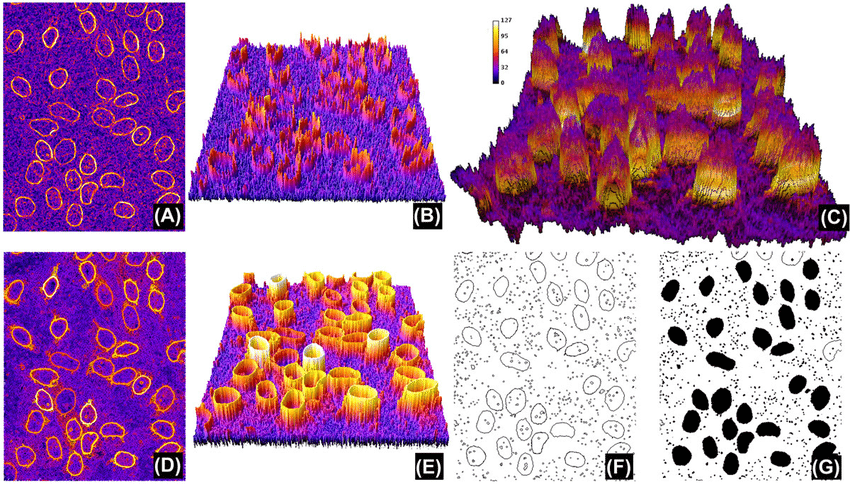
Các thuật toán phân vùng dựa trên cạnh xác định các cạnh dựa trên các biến thể về độ tương phản, kết cấu, màu sắc và độ bão hòa. Chúng có thể biểu diễn chính xác đường viền của các đối tượng trong ảnh bằng cách sử dụng chuỗi cạnh bao gồm các cạnh riêng lẻ.

* + Phân vùng dựa trên ngưỡng (Threshold-Based Segmentation)

Hình 2. 2 Phân vùng dựa trên ngưỡng

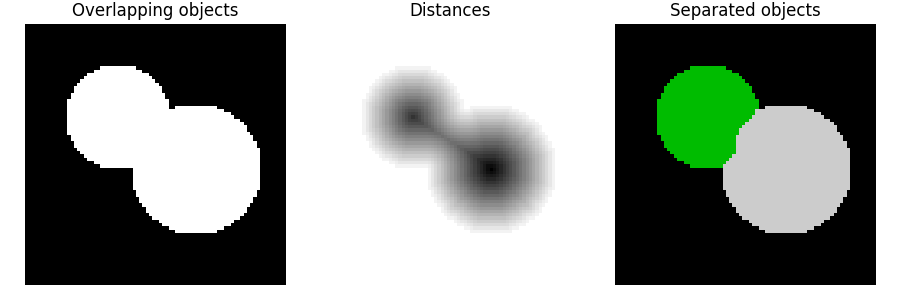
Ngưỡng là phương pháp phân vùng hình ảnh đơn giản nhất, phân chia các pixel dựa trên cường độ của chúng so với một giá trị hoặc ngưỡng nhất định. Nó phù hợp để phân vùng các đối tượng có cường độ cao hơn các đối tượng hoặc nền khác.

Giá trị ngưỡng T có thể hoạt động như một hằng số trong ảnh có độ nhiễu thấp. Trong một số trường hợp, có thể sử dụng ngưỡng động. Ngưỡng chia hình ảnh thang độ xám thành hai phân vùng dựa trên mối quan hệ của chúng với T, tạo ra hình ảnh nhị phân.

* + Phân vùng theo khu vực (Region-Based Segmentation)

Hình 2. 3 Phân vùng theo khu vực

Phân vùng dựa trên vùng liên quan đến việc chia hình ảnh thành các vùng có đặc điểm tương tự. Mỗi vùng là một nhóm pixel mà thuật toán định vị thông qua một điểm giống. Khi thuật toán tìm thấy các điểm giống, nó có thể phát triển các vùng bằng cách thêm nhiều pixel hơn hoặc thu nhỏ và hợp nhất chúng với các điểm khác.

* +  Phân vùng lưu vực (Watershed Segmentation)

Hình 2. 4 Phân vùng theo lưu vực

Lưu vực sông là các phép biến đổi trong ảnh thang độ xám. Các thuật toán phân vùng lưu vực xử lý hình ảnh giống như bản đồ địa hình, với độ sáng pixel xác định độ cao (chiều cao). Kỹ thuật này phát hiện các đường tạo thành các rặng núi và lưu vực, đánh dấu các khu vực giữa các đường lưu vực. Nó chia hình ảnh thành nhiều vùng dựa trên chiều cao pixel, nhóm các pixel có cùng giá trị xám.

Kỹ thuật lưu vực có một số trường hợp sử dụng quan trọng, bao gồm xử lý hình ảnh y tế. Ví dụ: nó có thể giúp xác định sự khác biệt giữa vùng sáng hơn và vùng tối hơn trong quá trình quét MRI, có khả năng hỗ trợ chẩn đoán.

* + Phân vùng dựa trên cụm (Cluster-Based Segmentation)

A group of oranges and apples

Description automatically generated

Hình 2. 5 Phân vùng dựa trên cụm

Thuật toán gom cụm là thuật toán phân loại không giám sát giúp xác định thông tin ẩn trong hình ảnh. Chúng tăng cường tầm nhìn của con người bằng cách cô lập các cụm, bóng và cấu trúc. Thuật toán chia hình ảnh thành các cụm pixel có đặc điểm tương tự nhau, tách các phần tử dữ liệu và nhóm các phần tử tương tự thành các cụm.

## Gom cụm (Clustering) trong thị giác máy tính

### *2.2.1. Gom cụm là gì*

Gom cụm, còn được gọi là phân tích cụm, là một nhiệm vụ học máy không giám sát nhằm gán dữ liệu vào các nhóm. Các nhóm (hoặc cụm) này được tạo bằng cách khám phá các mẫu ẩn trong dữ liệu, để kết thúc việc nhóm các điểm dữ liệu có mẫu tương tự trong cùng một cụm. Ưu điểm chính của gom cụm nằm ở khả năng hiểu được dữ liệu chưa được gắn nhãn.

Dữ liệu không được gắn nhãn là loại dữ liệu phong phú và tương đối dễ lấy. Nó có thể là một tập hợp các bức ảnh được lấy từ internet, một tập hợp các tweet hoặc bất kỳ tập hợp điểm dữ liệu nào khác không có nhãn. Mặt khác, dữ liệu được gắn nhãn đi kèm với nhãn. Đây có thể là một tập hợp văn bản với mỗi văn bản được gắn nhãn là “tiểu thuyết về kẻ giết người” hoặc “chủ nghĩa hiện thực huyền diệu” hoặc một tập dữ liệu hình ảnh có nhãn như “mèo” và “chó”. Dữ liệu được gắn nhãn có nhiều thông tin hơn nhưng khan hiếm vì việc ghi nhãn là một quá trình khó khăn thường đòi hỏi con người phải cung cấp nhãn theo cách thủ công cho từng điểm dữ liệu.

Tất cả các thuật toán học máy đều yêu cầu dữ liệu để học, nhưng chính dạng dữ liệu - cụ thể là, dù được gắn nhãn hay không gắn nhãn - sẽ quyết định thuật toán nào chúng ta có thể áp dụng cho dữ liệu đó. Dựa trên điều này, chúng ta có thể phân biệt giữa hai cách tiếp cận chính đối với học máy: học có giám sát và học không giám sát.

Đúng như tên gọi của nó, học máy có giám sát học dưới sự giám sát được cung cấp dưới dạng nhãn dữ liệu. Kết quả học tập của các thuật toán này là ánh xạ các điểm dữ liệu tới nhãn. Ngược lại, học không giám sát thiếu tín hiệu do nhãn dữ liệu cung cấp. Thay vào đó, các thuật toán học không giám sát sử dụng nhiều phương pháp thống kê khác nhau để tự lấy ra các nhãn.

Ví dụ: thuật toán học có giám sát áp dụng cho tập dữ liệu email có thể yêu cầu mọi email phải có nhãn như “email chuyên nghiệp” hoặc “email cá nhân”. Mục tiêu học tập của thuật toán như vậy là tìm hiểu các mẫu ngôn ngữ chung cho hai loại email.

Mặt khác, một thuật toán gom cụm được áp dụng cho cùng một kho dữ liệu có thể tạo ra các cụm tương ứng với “email chuyên nghiệp” và “email cá nhân”. Ngoài ra, nếu tập dữ liệu chứa các email được gửi dưới dạng quảng cáo thì thuật toán gom cụm cũng có thể phát hiện ra những email này.

### 2.2.2. *Các loại gom cụm*

Mặc dù có rất nhiều kiến ​​thức phong phú về các thuật toán gom cụm nhưng có rất ít sự đồng thuận về cách phân loại chúng. Các nguồn khác nhau sẽ phân loại chúng dựa trên các tiêu chí khác nhau. Theo quan điểm của chúng tôi, có hai cách phân loại hữu ích cho ứng dụng thực tế: (i) dựa trên số lượng cụm mà một điểm dữ liệu có thể thuộc về và (ii) dựa trên hình dạng của các cụm kết quả.

Sự khác biệt đầu tiên về cơ bản là giữa các thuật toán gom cụm “cứng” và “mềm”:

* Gom cụm cứng: Trong trường hợp này, mỗi điểm dữ liệu đầu vào hoàn toàn thuộc về một cụm hoặc không. Ví dụ mỗi khách hàng được xếp vào 1 nhóm trong 10 nhóm.
* Gom cụm mềm: Trong trường hợp này, thay vì đặt từng điểm dữ liệu đầu vào một cụm riêng biệt, xác suất hoặc khả năng điểm dữ liệu đó nằm trong các cụm đó sẽ được chỉ định. Ví dụ mỗi khách hàng được ấn định một xác suất thuộc một trong 10 cụm của cửa hàng bán lẻ.

Cụ thể hơn, một điểm dữ liệu có thể là thành viên của chỉ một cụm (gom cụm cứng) hoặc nó có thể thuộc nhiều cụm với các mức độ khác nhau (gom cụm mềm). Tùy thuộc vào ứng dụng của bạn, bạn có thể muốn các cụm của mình cứng nhắc hoặc bạn có thể muốn chúng chồng lên nhau, vì vậy đây là một điều quan trọng cần cân nhắc.

Cách phân loại khác dựa trên hình dạng và loại cụm mà thuật toán tạo ra. Bộ phận này chứa nhiều loại cụm khác nhau với một số thuật toán được thiết lập nhiều nhất là thuật toán phân cấp, dựa trên trọng tâm và dựa trên mật độ. Chúng tôi sẽ đề cập đến những điều này chi tiết hơn trong phần tiếp theo.

Lưu ý rằng hai cách phân loại này không loại trừ lẫn nhau. Ví dụ: một thuật toán gom cụm phổ biến được gọi là gom cụm k-means vừa cứng vừa dựa trên centroid. Tuy nhiên, những cách phân loại này cần được coi trọng vì nhiều thuật toán gom cụm không hoàn toàn thuộc về một cách tiếp cận gom cụm. Do đó, việc phân loại đóng vai trò như một hướng dẫn để chọn thuật toán phù hợp nhất cho ứng dụng của bạn.

### *2.2.3. Tại sao gom cụm quan trọng trong thị giác máy tính*

Gom cụm có vai trò quan trọng trong lĩnh vực thị giác máy tính vì nó giúp xử lý và hiểu cấu trúc của hình ảnh một cách tự động và trực quan. Dưới đây là một số lý do tại sao gom cụm quan trọng trong thị giác máy tính:

* Phân vùng ảnh: Gom cụm có thể được sử dụng để phân tách hình ảnh thành các vùng có tính chất hoặc đặc điểm tương tự. Điều này giúp cho việc phân vùng ảnh trở nên dễ dàng hơn và hữu ích trong việc hiểu và xử lý hình ảnh.
* Phát hiện đối tượng: Trong việc phát hiện đối tượng trong hình ảnh, gom cụm có thể giúp xác định các điểm dữ liệu thuộc về cùng một đối tượng hoặc phần của hình ảnh.
* Nén dữ liệu hình ảnh: Gom cụm có thể giúp nén dữ liệu hình ảnh bằng cách biểu diễn hình ảnh bằng cụm dữ liệu thay vì sử dụng mỗi điểm dữ liệu riêng lẻ.
* Tìm kiếm hình ảnh: Gom cụm có thể được sử dụng trong tìm kiếm hình ảnh để tìm các hình ảnh tương tự dựa trên cấu trúc và nội dung.
* Xử lý dữ liệu hình ảnh lớn: Trong việc xử lý dữ liệu hình ảnh lớn và dữ liệu video, gom cụm giúp làm giảm kích thước và phức tạp của dữ liệu, từ đó cải thiện hiệu suất xử lý.

## Các phương pháp phân vùng ảnh dựa trên gom cụm

Vì nhiệm vụ gom cụm mang tính chủ quan nên có rất nhiều phương tiện có thể được sử dụng để đạt được mục tiêu này. Mỗi phương pháp đều tuân theo một bộ quy tắc khác nhau để xác định ' sự tương đồng' giữa các điểm dữ liệu. Trên thực tế, có hơn 100 thuật toán gom cụm đã được biết đến. Nhưng rất ít thuật toán được sử dụng phổ biến. Chúng ta hãy xem xét chúng một cách chi tiết:

* **Connectivity models (**Các mô hình kết nối): Như tên cho thấy, các mô hình này dựa trên khái niệm rằng các điểm dữ liệu ở gần hơn trong không gian dữ liệu thể hiện sự giống nhau hơn so với các điểm dữ liệu nằm ở xa hơn. Những mô hình này có thể theo hai cách tiếp cận. Trong cách tiếp cận đầu tiên, họ bắt đầu bằng cách phân loại tất cả các điểm dữ liệu thành các cụm riêng biệt và sau đó tổng hợp chúng khi khoảng cách giảm dần. Trong cách tiếp cận thứ hai, tất cả các điểm dữ liệu được phân loại thành một cụm duy nhất và sau đó được phân vùng khi khoảng cách tăng lên. Ngoài ra, việc lựa chọn hàm khoảng cách là chủ quan. Những mô hình này rất dễ diễn giải nhưng thiếu khả năng mở rộng để xử lý các tập dữ liệu lớn. Ví dụ về các mô hình này là các thuật toán gom cụm theo cấp bậc và các biến thể của chúng.
* **Centroid models** (Mô hình trung tâm): Đây là các thuật toán gom cụm lặp trong đó khái niệm về sự tương tự được bắt nguồn từ sự gần gũi của một điểm dữ liệu với tâm hoặc trung tâm cụm của cụm. Thuật toán gom cụm k-Means là một thuật toán phổ biến thuộc loại này. Trong các mô hình này, không. Các tham số cụm được yêu cầu ở cuối phải được đề cập trước, điều này rất quan trọng để có kiến ​​thức trước về tập dữ liệu. Các mô hình này chạy lặp đi lặp lại để tìm ra điểm tối ưu cục bộ.
* **Distribution models:** (Mô hình phân phối): Các mô hình gom cụm này dựa trên khái niệm về khả năng tất cả các điểm dữ liệu trong cụm thuộc về cùng một phân phối (Ví dụ: Bình thường, Gaussian). Những mô hình này thường bị trang bị quá mức. Một ví dụ phổ biến của các mô hình này là thuật toán Tối đa hóa kỳ vọng sử dụng phân phối chuẩn đa biến.
* **Density models** (Mô hình mật độ): Các mô hình này tìm kiếm không gian dữ liệu cho các khu vực có mật độ điểm dữ liệu khác nhau trong không gian dữ liệu. Họ cô lập các vùng dày đặc khác nhau và gán các điểm dữ liệu trong các vùng này vào cùng một cụm. Các ví dụ phổ biến về mô hình mật độ là DBSCAN và OPTICS. Những mô hình này đặc biệt hữu ích để xác định các cụm có hình dạng tùy ý và phát hiện các điểm bất thường, vì chúng có thể phát hiện và phân tách các điểm nằm trong vùng thưa thớt của không gian dữ liệu cũng như các điểm thuộc vùng dày đặc.

## Những thuật toán phổ biến

* **K-Means Clustering:** K-Means là một trong những thuật toán gom cụm phổ biến nhất. Nó chia dữ liệu thành các cụm dựa trên sự tương tự giữa các điểm dữ liệu và đặt số lượng cụm trước đó (K).
* **Hierarchical Clustering:** Hierarchical Clustering tạo ra một cây phân cấp của các cụm bằng cách sử dụng các thuật toán như Agglomerative hoặc Divisive. Nó không yêu cầu xác định số lượng cụm trước và có thể tạo ra biểu đồ dendrogram để hiển thị cấu trúc gom cụm.
* **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** DBSCAN dựa trên mật độ dữ liệu. Nó tự động xác định số lượng cụm và có khả năng xử lý dữ liệu có nhiễu và cụm có hình dạng không đều.
* **Mean-Shift Clustering:** Mean-Shift tìm các điểm tối ưu của hàm mật độ dữ liệu để xác định các cụm. Nó không yêu cầu xác định số lượng cụm trước đó và có khả năng tìm ra các cụm có hình dạng không đều.
* **Gaussian Mixture Model (GMM):** GMM mô hình hóa dữ liệu bằng các phân phối Gaussian và sử dụng thuật toán EM (Expectation-Maximization) để xác định cụm và tham số của phân phối.
* **Spectral Clustering:** Spectral Clustering sử dụng phân tích giá trị riêng của ma trận tương tự dữ liệu để gom cụm. Nó thường được sử dụng cho dữ liệu có cấu trúc phức tạp hoặc không tuyến tính.
* **Affinity Propagation:** Affinity Propagation sử dụng ma trận độ tương tự dữ liệu để xác định các điểm dữ liệu được gửi thông qua mạng và những điểm trung tâm của mỗi cụm.
* **Birch (Balanced Iterative Reducing and Clustering Using Hierarchies**): Birch là một thuật toán gom cụm dựa trên cấu trúc cây và được sử dụng cho dữ liệu lớn. Nó tự động xác định số lượng cụm và có khả năng xử lý dữ liệu dạng dữ liệu dạng số lượng lớn.

# Các bước thực hiện

* Thu thập dữ liệu ảnh:

Xác định nguồn dữ liệu ảnh của bạn. Đây có thể là hình ảnh chụp từ máy ảnh, dữ liệu ảnh từ cơ sở dữ liệu công khai, hoặc các nguồn dữ liệu khác.

Lựa chọn các hình ảnh phù hợp với mục tiêu của bạn và đảm bảo rằng chúng đủ đa dạng để đại diện cho các trường hợp khác nhau.

* Tiền xử lý dữ liệu:

Chuẩn hóa kích thước hình ảnh: Đảm bảo rằng tất cả các hình ảnh có cùng kích thước để đảm bảo tính nhất quán trong dữ liệu đầu vào.

Giảm nhiễu: Loại bỏ nhiễu từ hình ảnh bằng cách sử dụng các kỹ thuật xử lý hình ảnh như làm mịn ảnh.

Chuyển đổi không gian màu sắc : Đối với dự án về phân vùng ảnh màu sắc, ta cần chuyển đổi không gian màu sắc của hình ảnh (ví dụ: từ RGB sang GRAY) để tăng cường tính tương đồng màu sắc giữa các điểm.

* Đưa vào mô hình huấn luyện:

Chia dữ liệu ảnh thành tập huấn luyện và tập kiểm tra. Tập huấn luyện sẽ được sử dụng để huấn luyện mô hình phân vùng, trong khi tập kiểm tra sẽ được sử dụng để đánh giá hiệu suất của mô hình.

Đảm bảo rằng tập kiểm tra đủ đại diện và không chứa dữ liệu đã được sử dụng trong quá trình huấn luyện.

* Tạo nhãn:

Nếu ta có dữ liệu dán nhãn (ground truth) để đánh giá kết quả phân vùng, ta sẽ tạo nhãn cho các hình ảnh trong tập huấn luyện.

Nhãn này sẽ giúp đối chiếu với kết quả phân vùng của mô hình để đánh giá độ chính xác.

# Phương pháp đề xuất

## K-means Clustering

**a) Các bước của thuật toán k-Means Clustering**

Trong thuật toán k-Means mỗi cụm dữ liệu được đặc trưng bởi một tâm (centroid). tâm là điểm đại diện nhất cho một cụm và có giá trị bằng trung bình của toàn bộ các điểm nằm trong cụm. Chúng ta sẽ dựa vào khoảng cách từ mỗi điểm tới các tâm để xác định nhãn cho chúng thuộc về tâm gần nhất. Ban đầu thuật toán sẽ khởi tạo ngẫu nhiên một số lượng điểm xác định trước cho tâm cụm. Sau đó tiến hành xác định nhãn cho từng điểm và tiếp tục cập nhật lại tâm cụm. Thuật toán sẽ dừng khi toàn bộ các điểm dữ liệu được phân về đúng cụm hoặc số lượt cập nhật tâm chạm ngưỡng.

Cụ thể các bước của thuật toán k-Means được tóm tắt như sau:

1.- Khởi tạo ngẫu nhiên k tâm cụm 1, 2, …,k.

2.- Lặp lại quá trình cập nhật tâm cụm cho tới khi dừng:

a. Xác định nhãn cho từng điểm ci dựa vào khoảng cách tới từng tâm cụm:

A black and white text

Description automatically generated

b. Tính toán lại tâm cho từng cụm theo trung bình của toàn bộ các điểm trong một cụm:

A close-up of math equations

Description automatically generated

* Trong công thức 2.a thì  là bình phương của chuẩn bậc 2, kí hiệu là L2, chuẩn bậc 2 là một độ đo khoảng cách thường được sử dụng trong machine learning.
* Trong công thức 2.b chúng ta sử dụng hàm 1(.), hàm này có giá trị trả về là 1 nếu nhãn của điểm ci được dự báo thuộc về cụm j, trái lại thì trả về giá trị 0.
* -Tử số: Tổng khoảng cách của toàn bộ các điểm dữ liệu nằm trong cụm j .
* -Mẫu số: Số lượng các điểm dữ liệu thuộc cụm j.
* -j : Vị trí của tâm cụm k mà ta dự báo tại thời điểm hiện tại.

Trong thuật toán trên thì tham số mà chúng ta cần lựa chọn chính là số lượng cụm k. Thời điểm ban đầu ta sẽ khởi tạo k điểm dữ liệu một cách ngẫu nhiên và sau đó gán các tâm bằng giá trị của k điểm dữ liệu này. Các bước trong vòng lặp ở bước 2 thực chất là:

a. Gán nhãn cho mỗi điểm dữ liệu bằng với nhãn của tâm cụm gần nhất.

b. Dịch chuyển dần dần tâm cụm j tới trung bình của những điểm dữ liệu mà được phân về j.

Bên dưới ta sẽ xây dựng một class KMean từ đầu để gom cụm dữ liệu. Hàm quan trọng nhất là fit() có tác dụng huấn luyện theo vòng lặp thuật toán KMean(). Hàm get\_labels() được sử dụng để phân các điểm dữ liệu về đúng nhãn của cụm. Trong khi đó get\_centroids() được sử dụng để xác định tâm của các cụm dựa trên nhãn. Hàm should\_stop() có tác dụng kiểm tra điều kiện dừng của vòng lặp. Ở đây chúng ta sẽ dừng khi số lượng vòng lặp vượt quá max\_iteration hoặc các centroids ngừng thay đổi.

import numpy as np

from sklearn.datasets import make\_blobs

class KMeans():

  def \_\_init\_\_(self, k, max\_iteration=10):

      self.k = k

      self.max\_iteration = max\_iteration

      self.all\_centroids = []

      self.all\_labels = []

  # Hàm thuật toán k-Means lấy đầu vào là một bộ dữ liệu và số lượng cluster k. Trẻ về tâm của k cụm

  def fit(self, dataSet):

      # Khởi tạo ngẫu nhiên k centroids

      numFeatures = dataSet.shape[1]

      centroids = self.get\_random\_centroids(numFeatures, self.k)

      self.all\_centroids.append(centroids)

      self.all\_labels.append(None)

      # Khởi tạo các biến iterations, oldCentroids

      iterations = 0

      oldCentroids = None

      # Vòng lặp cập nhật centroids trong thuật toán k-Means

      while not self.should\_stop(oldCentroids, centroids, iterations):

          # Lưu lại centroids cũ cho quá trình kiểm tra hội tụ

          oldCentroids = centroids

          iterations += 1

          # Gán nhãn cho mỗi diểm dữ liệu dựa vào centroids

          labels = self.get\_labels(dataSet, centroids)

          self.all\_labels.append(labels)

          # Cập nhật centroids dựa vào nhãn dữ liệu

          # print('0ld centroids: ', centroids)

          centroids = self.get\_centroids(dataSet, labels, self.k)

          # print('new centroids: ', centroids)

          self.all\_centroids.append(centroids)

      return centroids

  # Hàm khởi tạo centroids ngẫu nhiên

  def get\_random\_centroids(self, numFeatures, k):

    return np.random.rand(k, numFeatures)

    # return np.array([[-5., -5.],

    #                  [4., 6.]])

  # Hàm này trả về nhãn cho mỗi điểm dữ liệu trong datasets

  def get\_labels(self, dataSet, centroids):

      # Với mỗi quan sát trong dataset, lựa chọn centroids gần nhất để gán label cho dữ liệu.

      labels = []

      for x in dataSet:

        # Tính khoảng cách tới các centroids và cập nhận nhãn

        distances = np.sum((x-centroids)\*\*2, axis=1)

        label = np.argmin(distances)

        labels.append(label)

      return labels

  # Hàm này trả về True hoặc False nếu k-Means hoàn thành. Điều kiện k-Means hoàn thành là

  # thuật toán vượt ngưỡng số lượng vòng lặp hoặc centroids ngừng thay đổi

  def should\_stop(self, oldCentroids, centroids, iterations):

      if iterations > self.max\_iteration:

        return True

      return np.all(oldCentroids == centroids)

  # Trả về toan độ mới cho k centroids của mỗi chiều.

  def get\_centroids(self, dataSet, labels, k):

      centroids = []

      for j in np.arange(k):

        # Lấy index cho mỗi centroids

        idx\_j = np.where(np.array(labels) == j)[0]

        centroid\_j = dataSet[idx\_j, :].mean(axis=0)

        centroids.append(centroid\_j)

      return np.array(centroids)

dataset, \_ = make\_blobs(n\_samples=250, cluster\_std=3.0, random\_state=123)

kmean = KMeans(k=2, max\_iteration=8)

centroids = kmean.fit(dataset)

import matplotlib.pyplot as plt

from matplotlib.gridspec import GridSpec

gs = GridSpec(nrows=3, ncols=3)

plt.figure(figsize = (20, 20))

plt.subplots\_adjust(wspace=0.2,hspace=0.4)

colors = ['green', 'blue']

labels = ['cluster 1', 'cluster 2']

for i in np.arange(len(kmean.all\_centroids)):

  ax = plt.subplot(gs[i])

  if i == 0:

    centroids\_i = kmean.all\_centroids[i]

    plt.scatter(dataset[:, 0], dataset[:, 1], s=50, alpha=0.5, color='red')

    for j in np.arange(kmean.k):

      plt.scatter(centroids\_i[j, 0], centroids\_i[j, 1], marker='x', s=100, color='red')

    plt.title('All points in original dataset')

  else:

    # Lấy centroids và labels tại bước thứ i

    centroids\_i = kmean.all\_centroids[i]

    labels\_i = kmean.all\_labels[i]

    # Visualize các điểm cho từng cụm

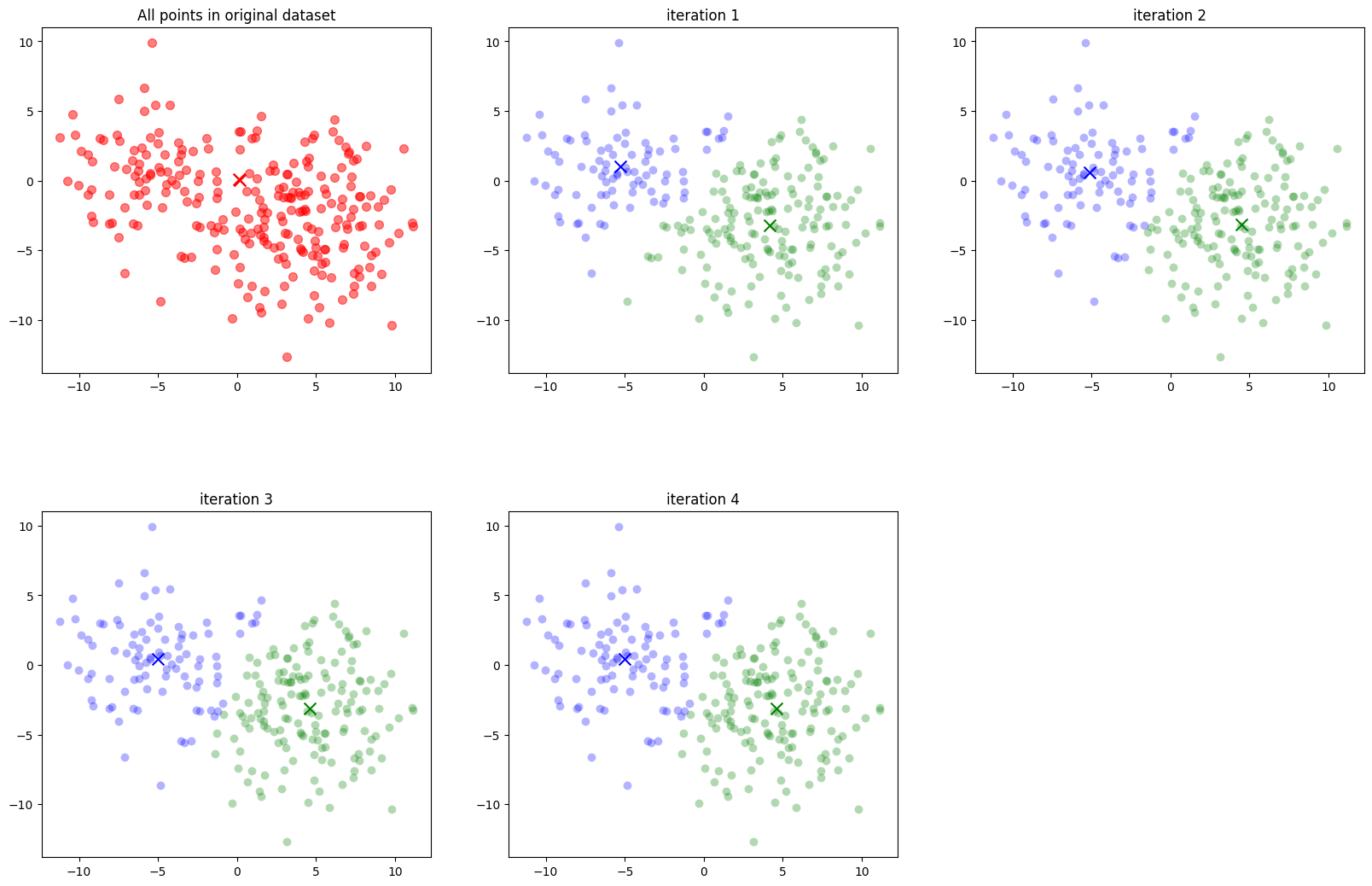
    for j in np.arange(kmean.k):

      idx\_j = np.where(np.array(labels\_i) == j)[0]

      plt.scatter(dataset[idx\_j, 0], dataset[idx\_j, 1], color=colors[j], label=labels[j], s=50, alpha=0.3, lw = 0)

      plt.scatter(centroids\_i[j, 0], centroids\_i[j, 1], marker='x', color=colors[j], s=100, label=labels[j])

    plt.title(r'iteration {}'.format(i))



Hình 4. 1 Sự dịch chuyển tâm của cụm sau mỗi vòng lặp.

Thời điểm ban đầu hai tâm cụm được khởi tạo một cách ngẫu nhiên. Sau đó sau mỗi vòng lặp thì các nhãn được cập nhật lại tuỳ theo vị trí của chúng đối với tâm cụm. Tiếp tục cập nhật lại tâm cụm theo các nhãn dữ liệu mới. Cuối cùng thì thuật toán sẽ hội tụ khi tâm cụm ngừng thay đổi vị trí.

Trên sklearn chúng ta có thể dễ dàng huấn luyện thuật toán k-Means thông qua class KMeans của sklearn.cluster

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2, random\_state=0)

kmeans.fit(dataset)

print('kmeans labels of 5 observation: ', kmeans.labels\_[:5])

print('kmeans centroids: \n', kmeans.cluster\_centers\_)

kmeans labels of 5 observation: [0 0 0 1 0]

kmeans centroids:

[[ 4.58876493 -3.13006162]

[-4.99023469 0.44409831]]

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

Dự báo cụm cho một quan sát mới:

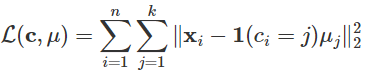
x = np.array([[5, 1]])

kmeans.predict(x)

array([0], dtype=int32)

**b) Sự hội tụ của thuật toán k-Means clustering**

Liệu rằng thuật toán k-Means luôn hội tụ? Điều này là hợp lý, chúng ta sẽ chứng minh thông qua xét một hàm biến dạng (distortion function) dạng MSE:



Trong đó c = (c1, c2, …, cn) là véc tơ nhãn của toàn bộ các quan sát được dữ báo từ thuật toán gom cụm. 1(ci=j)j chính là tâm cụm mà quan sát xi được phân bổ về. Như vậy hàm biến dạng thực tế là đo lường khoảng cách từ toàn bộ các điểm đến các tâm cụm. Theo như bước 2 trong qui trình của thuật toán k-Means thì đầu tiên tại bước 2.a chúng ta sẽ tính khoảng cách từ một điểm đến toàn bộ các tâm cụm nhằm cập nhật lại nhãn. Như vậy giá trị  luôn giảm sau bước này. Tiếp theo sang bước thứ 2.b chúng ta cập nhật lại tâm cụm sao cho khoảng cách của chúng tới các điểm là nhỏ nhất. Vị trí này là nghiệm của phương trình đạo hàm bậc nhất:

A math equations with numbers and symbols

Description automatically generated with medium confidence

Dòng thứ 2 suy ra dòng thứ 3 là theo công thức đạo hàm của 

Dòng thứ 3 suy ra dòng thứ 4 là vì khi nhãn ci = j thì 1(ci = j) =1. Trường hợp nhãn ci≠j thì là khoảng cách được gán về 0 và không ảnh hưởng tới tới hàm biến dạng.

Như vậy đạo hàm bậc nhất bằng 0 khi:

A math equations on a white background

Description automatically generated

Như vậy khi tâm cụm là trung bình của toàn bộ các điểm được phân bổ về cụm thì tổng khoảng cách giữa các điểm tới tâm cụm mà nó thuộc về là nhỏ nhất. Điều đó có nghĩa là hàm biến dạng luôn giảm sau mỗi vòng lặp. Mặt khác hàm biến dạng bị chặn dưới bởi 0 nên là một chuỗi hội tụ. Tức là sau một hữu hạn bước thì thuật toán k-Means sẽ dừng.

**c) Phương pháp Elbow trong lựa chọn số cụm**

Trong thuật toán k-Means thì chúng ta cần phải xác định trước số cụm. Câu hỏi đặt ra là đâu là số lượng cụm cần phân chia tốt nhất đối với một bộ dữ liệu cụ thể? Phương pháp Elbow là một cách giúp ta lựa chọn được số lượng các cụm phù hợp dựa vào đồ thị trực quan hoá bằng cách nhìn vào sự suy giảm của hàm biến dạng và lựa chọn ra điểm khuỷ tay (elbow point). Để tìm hiểu phương pháp Elbow, bên dưới chúng ta cùng thử nghiệm vẽ biểu đồ hàm biến dạng bằng cách điều chỉnh số lượng cụm của thuật toán k-Means.

from scipy.spatial.distance import cdist

losses = []

K = 10

for i in range(1, K):

  # 1.  Huấn luyện với số cụm = i

  kmeans\_i = KMeans(n\_clusters=i, random\_state=0)

  kmeans\_i.fit(dataset)

  # 2. Tính \_hàm biến dạng\_

  # 2.1. Khoảng cách tới toàn bộ centroids

  d2centroids = cdist(dataset, kmeans\_i.cluster\_centers\_, 'euclidean') # shape (n, k)

  # 2.2. Khoảng cách tới centroid gần nhất

  min\_distance = np.min(d2centroids, axis=1) # shape (n)

  loss = np.sum(min\_distance)

  losses.append(loss)

#Vẽ biểu đồ hàm biến dạng

plt.figure(figsize=(12, 8))

plt.plot(np.arange(1, K), losses, 'bx-')

plt.xlabel('Values of K')

plt.ylabel('Distortion')

plt.title('The Elbow Method using Distortion')

plt.show()

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

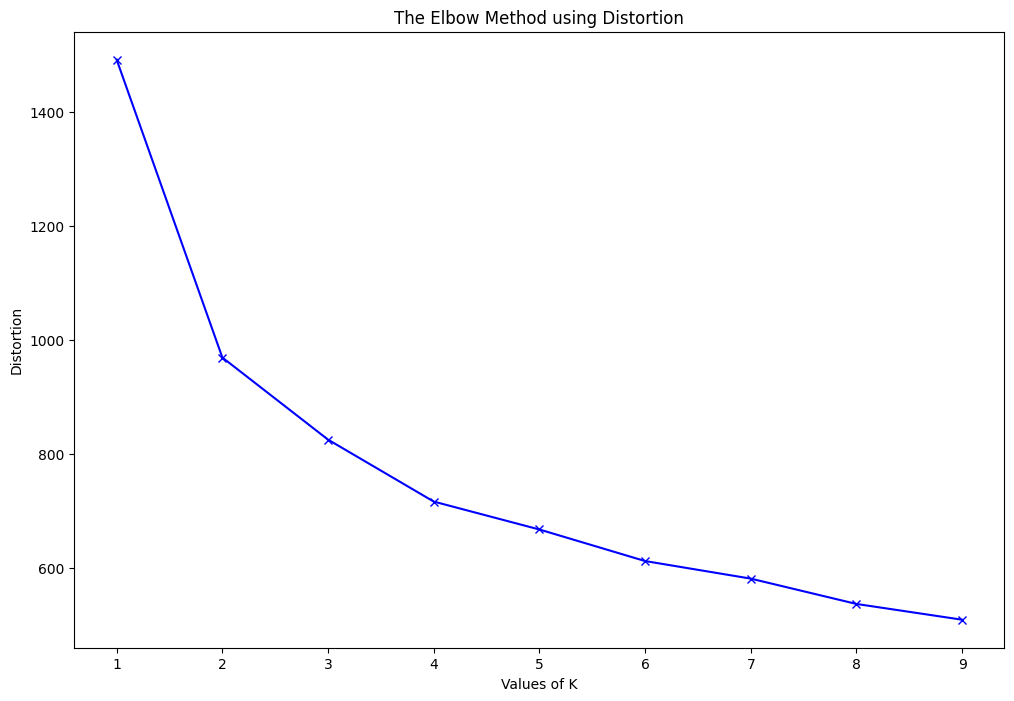
warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(



Hình 4. 2 Đồ thị hàm biến dạng của thuật toán k-Means.

Trục tung là giá trị của hàm biến dạng và trục hoành là giá trị của số lượng cụm cần phân chia trong thuật toán k-Means.

Điểm khuỷ tay là điểm mà ở đó tốc độ suy giảm của hàm biến dạng sẽ thay đổi nhiều nhất. Tức là kể từ sau vị trí này thì gia tăng thêm số lượng cụm cũng không giúp hàm biến dạng giảm đáng kể. Nếu thuật toán phân chia theo số lượng cụm tại vị trí này sẽ đạt được tính chất gom cụm một cách tổng quát nhất mà không gặp các hiện tượng vị khớp (overfitting). Trong hình trên thì ta thấy vị trí của điểm khuỷ tay chính là k=2 vì khi số lượng cụm lớn hơn 2 thì tốc độ suy giảm của hàm biến dạng dường như không đáng kể so với trước đó.

Phương pháp Elbow là một phương pháp thường được sử dụng để lựa chọn số lượng cụm phân chia hợp lý dựa trên biểu đồ, tuy nhiên có một số trường hợp chúng ta sẽ không dễ dàng phát hiện vị trí của Elbow, đặc biệt là đối với những bộ dữ liệu mà qui luật gom cụm không thực sự dễ dàng được phát hiện. Nhưng nhìn chung thì phương pháp Elbow vẫn là một phương pháp tốt nhất được ứng dụng trong việc tìm kiếm số lượng cụm cần phân chia.

**d) Biểu diễn dữ liệu đa chiều trên đồ thị**

Sau khi huấn luyện thuật toán k-Means chúng ta sẽ cần kiểm tra qui luật gom cụm xem chúng đã thực sự hợp lý? Điều này sẽ được thực hiện thông qua biểu đồ gom cụm. Đối với các bộ dữ liệu hai chiều và ba chiều chúng ta có thể dễ dàng biểu diễn chúng trên mặt phẳng hoặc siêu phẳng. Nhưng đối với các bộ dữ liệu nhiều hơn ba chiều thì chúng ta cần áp dụng các phương pháp giảm chiều dữ liệu trước khi đồ thị hoá.

Giảm chiều dữ liệu là phương pháp giúp biến đổi các bộ dữ liệu cao chiều (high dimensional) về các bộ dữ liệu thấp chiều (low dimensional) mà vẫn giữ được nhiều nhất thông tin từ bộ dữ liệu gốc. Về các phương pháp giảm chiều dữ liệu sẽ được giới thiệu ở một bài khác. Trong chương này chúng ta sẽ học cách ứng dụng thuật toán t-SNE để giảm chiều dữ liệu về hai chiều và biểu diễn qui luật các cụm trong không gian hai chiều.

Đầu tiên chúng ta sẽ huấn luyện một mô hình k-Means với 2 cụm trên bộ dữ liệu 3 chiều.

import numpy as np

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

# Khởi tạo dữ liệu một cách ngẫu nhiên

X, \_ = make\_blobs(n\_samples=250, centers=2,

                        n\_features=2, cluster\_std=3.0,

                        random\_state=123)

# Hồi qui mô hình với 2 cụm

kmean = KMeans(n\_clusters=2, random\_state=0)

centroids = kmean.fit(X)

# Dự báo nhãn cho các cụm trên tập X

labels = kmean.predict(X)

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

Để huấn luyện mô hình t-SNE trên sklearn chúng ta sử dụng class [TSNE](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html). Chúng ta cần khai báo số lượng chiều dữ liệu sau khi giảm thông qua đối số n\_components như sau:

from sklearn.manifold import TSNE

import time

time\_start = time.time()

# Biến đổi dữ liệu về 2 chiều

X\_tsne = TSNE(n\_components=2, random\_state=123).fit\_transform(X)

print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time\_start))

t-SNE done! Time elapsed: 4.1357810497283936 seconds

Hàm fit\_transform() có tác dụng vừa huấn luyện vừa biến đổi dữ liệu, kết quả trả ra là bộ dữ liệu được giảm chiều X\_tsne.

print(X\_tsne.shape)

(250, 2)

Thông thường đối với các bộ dữ liệu kích thước lớn thì thuật toán t-SNE tốn khá nhiều thời gian để huấn luyện, chúng ta có thể in ra tổng thời gian huấn luyện để ước tính chi phí thời gian.

Tiếp theo ta biều đồ hoá các cụm bằng biểu đồ scatter.

import matplotlib.patheffects as PathEffects

import seaborn as sns

import numpy as np

def \_plot\_kmean\_scatter(X, labels):

    '''

    X: dữ liệu đầu vào

    labels: nhãn dự báo

    '''

    # lựa chọn màu sắc

    num\_classes = len(np.unique(labels))

    palette = np.array(sns.color\_palette("hls", num\_classes))

    # vẽ biểu đồ scatter

    fig = plt.figure(figsize=(12, 8))

    ax = plt.subplot()

    sc = ax.scatter(X[:,0], X[:,1], lw=0, s=40, c=palette[labels.astype(np.int)])

    plt.xlim(-25, 25)

    plt.ylim(-25, 25)

    # thêm nhãn cho mỗi cluster

    txts = []

    for i in range(num\_classes):

        # Vẽ text tên cụm tại trung vị của mỗi cụm

        xtext, ytext = np.median(X[labels == i, :], axis=0)

        txt = ax.text(xtext, ytext, str(i), fontsize=24)

        txt.set\_path\_effects([

            PathEffects.Stroke(linewidth=5, foreground="w"),

            PathEffects.Normal()])

        txts.append(txt)

    plt.title('t-sne visualization')

\_plot\_kmean\_scatter(X\_tsne, labels)

<ipython-input-13-71af9bea7ecd>:17: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.

Deprecated in NumPy 1.20; for more details and guidance: <https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations>

sc = ax.scatter(X[:,0], X[:,1], lw=0, s=40, c=palette[labels.astype(np.int)])

A screen shot of a graph

Description automatically generated

Hình 4. 3 Biểu diễn của các cụm trên không gian 2D bằng phương pháp giảm chiều dữ liệu

t-SNE.

Như vậy chúng ta nhận thấy rằng phép chiếu trên không gian hai chiều cho thấy thông tin cụm được bảo toàn và giữa các cụm có ranh giới rõ ràng. Bạn có đọc có thể áp dụng hàm \_plot\_kmean\_scatter() ở trên để visualize cho những bộ dữ liệu khác trên không gian hai chiều. Các thông tin cần truyền vào hàm là dữ liệu sau giảm chiều X\_tsne và nhãn dự báo labels.

## Bài toán phân vùng ảnh dựa trên kỹ thuật gom cụm áp dụng K-means Clustering

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import load\_sample\_image

#Load ảnh mẫu từ scikit-learn :

china = load\_sample\_image("china.jpg")

print(china)

print(china.shape)

[[[174 201 231]

[174 201 231]

[174 201 231]

...

[250 251 255]

[250 251 255]

[250 251 255]]

[[172 199 229]

[173 200 230]

[173 200 230]

...

[251 252 255]

[251 252 255]

[251 252 255]]

[[174 201 231]

[174 201 231]

[174 201 231]

...

[252 253 255]

[252 253 255]

[252 253 255]]

...

[[ 88 80 7]

[147 138 69]

[122 116 38]

...

[ 39 42 33]

[ 8 14 2]

[ 6 12 0]]

[[122 112 41]

[129 120 53]

[118 112 36]

...

[ 9 12 3]

[ 9 15 3]

[ 16 24 9]]

[[116 103 35]

[104 93 31]

[108 102 28]

...

[ 43 49 39]

[ 13 21 6]

[ 15 24 7]]]

(427, 640, 3)

#Chuyển đổi ảnh thành mảng 2D của các điểm dữ liệu (pixels)

china\_data = china / 255.0  # Chuẩn hóa giá trị màu sắc về khoảng [0, 1] để phù hợp với KMeans

height, width, channels = china\_data.shape

#Chuyển đổi mảng 3 chiều về thành mảng 2 chiều :

china\_data\_2d = china\_data.reshape((width \* height, channels))

print(china\_data\_2d)

print(china\_data\_2d.shape)

[[0.68235294 0.78823529 0.90588235]

[0.68235294 0.78823529 0.90588235]

[0.68235294 0.78823529 0.90588235]

...

[0.16862745 0.19215686 0.15294118]

[0.05098039 0.08235294 0.02352941]

[0.05882353 0.09411765 0.02745098]]

(273280, 3)

#Dùng phương pháp Elbow (PP Khuỷu Tay) để có thể đưa ra số gom cụm tối ưu cho bức ảnh :

#Tạo danh sách số lượng cụm bạn muốn thử nghiệm

k\_values = range(1, 16)  # Ví dụ: từ 1 đến 15 cụm

#Tạo danh sách để lưu giá trị SSE tương ứng với từng số lượng cụm

sse\_values = []

#Dùng vòng lặp qua từng số lượng cụm và tính SSE để rút ra số cụm tối ưu :

for k in k\_values:

    kmeans = KMeans(n\_clusters=k, random\_state=0)

    kmeans.fit(china\_data\_2d)

    sse\_values.append(kmeans.inertia\_)  # inertia\_ chứa SSE

#Vẽ biểu đồ SSE theo số lượng cụm k để xem sự sai số :

plt.figure(figsize=(8, 8))

plt.plot(k\_values, sse\_values, marker='o', linestyle='-', color='b')

plt.xlabel('Số lượng cụm (k)')

plt.ylabel('SSE')

plt.title('Biểu đồ Elbow để xác định số lượng cụm tối ưu')

plt.grid(True)

plt.show()

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

A graph with a line

Description automatically generated

Hình 4. 4 Vẽ biểu đồ SSE theo số lượng cụm k

\* Phương pháp Elbow cho ra chỉ số SSE : SSE càng nhỏ thì cụm đó càng ưu tiên. SSE là tổng bình phương sai số, và mục tiêu của gom cụm là tối thiểu hóa SSE. Nó đo lường sự "đồng nhất" hoặc "gần giống" của các điểm dữ liệu trong cùng một cụm. Điều này có nghĩa là một SSE nhỏ hơn thể hiện rằng các điểm dữ liệu trong cụm đó gần nhau hơn và tập trung hơn xung quanh điểm trung tâm của cụm, điều này cho biết cụm đó có tính chất tốt hơn.SSE được tính bằng cách lấy tổng của bình phương khoảng cách từ mỗi điểm dữ liệu trong cụm đến điểm trung tâm của cụm. Công thức tính SSE cho một cụm C và trung tâm của cụm là μ\_C: SSE(C) = Σ ||x - μ\_C||^2

Trong đó:

x là điểm dữ liệu thuộc cụm C.

μ\_C là trung tâm của cụm C.

Σ là ký hiệu tổng hợp trên tất

Huấn luyện mô hình với k = 2

n\_clusters\_1 = 2

kmeans\_1 = KMeans(n\_clusters= n\_clusters\_1, random\_state=0)

kmeans\_1.fit(china\_data\_2d)

#Lấy ra các giá trị điểm trung tâm cụm :

cluster\_centers\_1 = kmeans\_1.cluster\_centers\_

#Lấy ra các nhãn (hay nói cách khác là lấy ra tên cụm của từng điểm ảnh trong bức ảnh 2D ban đầu) :

cluster\_labels\_1 = kmeans\_1.labels\_

print(cluster\_centers\_1)

print(cluster\_labels\_1)

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

[[0.28302516 0.25523936 0.18494818]

[0.82404948 0.85468805 0.88413402]]

[1 1 1 ... 0 0 0]

Huấn luyện mô hình với k = 6

n\_clusters\_2 = 6

kmeans\_2 = KMeans(n\_clusters= n\_clusters\_2, random\_state=0)

kmeans\_2.fit(china\_data\_2d)

#Lấy ra các giá trị điểm trung tâm cụm :

cluster\_centers\_2 = kmeans\_2.cluster\_centers\_

#Lấy ra các nhãn (hay nói cách khác là lấy ra tên cụm của từng điểm ảnh trong bức ảnh 2D ban đầu) :

cluster\_labels\_2 = kmeans\_2.labels\_

print(cluster\_centers\_2)

print(cluster\_labels\_2)

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

[[0.91120164 0.94009844 0.97666764]

[0.28381957 0.25212185 0.17690952]

[0.09987477 0.08751997 0.06038895]

[0.65322602 0.61502248 0.54640633]

[0.46745764 0.42719508 0.30929023]

[0.763237 0.81821265 0.86898573]]

[5 5 5 ... 1 2 2]

Huấn luyện với k =10

#Nhìn trên biểu đồ Elbow ta thấy được số cụm tối ưu là 10 nên thực hiện phân thành 10 cụm :

n\_clusters\_3 = 10

kmeans\_3 = KMeans(n\_clusters= n\_clusters\_3, random\_state=0)

kmeans\_3.fit(china\_data\_2d)

#Lấy ra các giá trị điểm trung tâm cụm :

cluster\_centers\_3 = kmeans\_3.cluster\_centers\_

#Lấy ra các nhãn (hay nói cách khác là lấy ra tên cụm của từng điểm ảnh trong bức ảnh 2D ban đầu) :

cluster\_labels\_3 = kmeans\_3.labels\_

print(cluster\_centers\_3)

print(cluster\_labels\_3)

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

[[0.92088457 0.94508904 0.97708832]

[0.19315171 0.16825351 0.12597783]

[0.82052066 0.56957268 0.40866218]

[0.74915636 0.77752611 0.77161225]

[0.0704952 0.06337864 0.03922054]

[0.45764274 0.40525478 0.26656587]

[0.77862766 0.84679686 0.92590329]

[0.50984765 0.51286695 0.43283969]

[0.32109047 0.28967414 0.20257762]

[0.60744199 0.64455851 0.62506333]]

[6 6 6 ... 1 4 4]

from mpl\_toolkits.mplot3d import Axes3D

#Tạo biểu đồ 3D cho mỗi mô hình và hiển thị chúng trên cùng một bức ảnh :

fig = plt.figure(figsize=(18, 6))

#Mô hình 1 (Số cụm là 2):

ax1 = fig.add\_subplot(131, projection='3d')

scatter1 = ax1.scatter(china\_data\_2d[:, 0], #Lấy giá trị Red

                       china\_data\_2d[:, 1], #Lấy giá trị Green

                       china\_data\_2d[:, 2], #Lấy giá trị Blue

                       c=cluster\_labels\_1, cmap='viridis', s=5)

ax1.set\_xlabel('Red')

ax1.set\_ylabel('Green')

ax1.set\_zlabel('Blue')

ax1.set\_title(f"Mô hình 1: {n\_clusters\_1} cụm")

#Mô hình 2 (Số cụm là 6):

ax2 = fig.add\_subplot(132, projection='3d')

scatter2 = ax2.scatter(china\_data\_2d[:, 0],

                       china\_data\_2d[:, 1],

                       china\_data\_2d[:, 2],

                       c=cluster\_labels\_2, cmap='viridis', s=5)

ax2.set\_xlabel('Red')

ax2.set\_ylabel('Green')

ax2.set\_zlabel('Blue')

ax2.set\_title(f"Mô hình 2: {n\_clusters\_2} cụm")

#Mô hình 3 (Số cụm là 10):

ax3 = fig.add\_subplot(133, projection='3d')

scatter3 = ax3.scatter(china\_data\_2d[:, 0],

                       china\_data\_2d[:, 1],

                       china\_data\_2d[:, 2],

                       c=cluster\_labels\_3, cmap='viridis', s=5)

ax3.set\_xlabel('Red')

ax3.set\_ylabel('Green')

ax3.set\_zlabel('Blue')

ax3.set\_title(f"Mô hình 3: {n\_clusters\_3} cụm")

#Thêm thanh bar cho từng mô hình :

bar1 = fig.colorbar(scatter1, ax=ax1, pad=0.1, orientation='vertical')

bar1.set\_label('Cluster')

bar2 = fig.colorbar(scatter2, ax=ax2, pad=0.1, orientation='vertical')

bar2.set\_label('Cluster')

bar3 = fig.colorbar(scatter3, ax=ax3, pad=0.1, orientation='vertical')

bar3.set\_label('Cluster')

plt.suptitle("Biểu đồ gom cụm dựa trên K-Means (OXYZ)")

plt.tight\_layout()

plt.show()

A graph of a graph

Description automatically generated with medium confidence

Hình 4. 5 Biểu đồ gom cụm dựa trên K-Means (OXYZ

#Tạo ảnh đã phân vùng :

#Sau khi tìm ra các vị trí điểm trung tâm cụm

#Thì chuyển các giá trị màu của các điểm trong bức ảnh về thành giá trị màu của điểm trung tâm cụm (Dựa vào nhãn đã đánh dấu ban đầu):

#Đồng thời reshape lại ma trận 2 chiều thành ma trận ảnh 3 chiều để có thể hiển thị :

clustered\_image\_1 = cluster\_centers\_1[cluster\_labels\_1].reshape(china.shape)

clustered\_image\_2 = cluster\_centers\_2[cluster\_labels\_2].reshape(china.shape)

clustered\_image\_3 = cluster\_centers\_3[cluster\_labels\_3].reshape(china.shape)

print(clustered\_image\_1.shape)

print(clustered\_image\_2.shape)

print(clustered\_image\_3.shape)

(427, 640, 3)

(427, 640, 3)

(427, 640, 3)

#Hiển thị ảnh gốc và ảnh đã phân vùng cho 3 mô hình K-Means :

plt.figure(figsize=(18, 6))

plt.subplot(141)

plt.imshow(china)

plt.title("Ảnh gốc")

plt.axis('off')

#Hiển thị cho mô hình 1 :

plt.subplot(142)

plt.imshow(clustered\_image\_1)

plt.title(f"Ảnh đã phân vùng (Mô hình 1)")

plt.axis('off')

#Hiển thị cho mô hình 2 :

plt.subplot(143)

plt.imshow(clustered\_image\_2)

plt.title(f"Ảnh đã phân vùng (Mô hình 2)")

plt.axis('off')

#Hiển thị cho mô hình 3 :

plt.subplot(144)

plt.imshow(clustered\_image\_3)

plt.title(f"Ảnh đã phân vùng (Mô hình 3)")

plt.axis('off')

plt.show()

A comparison of a building

Description automatically generated with medium confidence

Hình 4. 6 ảnh gốc và ảnh đã phân vùng

Thuật toán k-Means là phương pháp đơn giản và thường được áp dụng trong các bài toán gom cụm. Thuật toán này dựa trên khoảng cách để cập nhật lại nhãn cho các quan sát về tâm gần nhất và tâm cụm sau đó được tính theo trung bình của toàn bộ các quan sát bên trong cụm. Chúng ta cũng chứng minh được rằng thuật toán sẽ hội tụ sau hữu hạn bước.

Tuy nhiên thuật k-Means vẫn là thuật toán tồn tại những hạn chế đó là cần phải xác định trước tâm cụm, vị trí tâm của cụm chịu sự phụ thuộc vào vị trí khởi tạo ban đầu của chúng, thuật toán cũng bị ảnh hưởng bởi outliers và sự khác biệt về đơn vị của biến đầu vào. Trong trường hợp các bộ dữ liệu có phân phối phức tạp và mất cân bằng thì thuật toán sẽ không gom cụm chính xác.

# Ứng dụng

Phân vùng hình ảnh dựa trên K-Means clustering có nhiều ứng dụng trong xử lý hình ảnh và thị giác máy tính. Dưới đây là một số ví dụ về các ứng dụng của phân vùng hình ảnh dựa trên K-Means clustering:

* **Giảm số lượng màu sắc:** K-Means clustering có thể được sử dụng để giảm số lượng màu sắc trong hình ảnh. Điều này hữu ích trong việc giảm kích thước tệp ảnh và tối ưu hóa việc lưu trữ hình ảnh trên web.
* **Phân vùng hình ảnh vệ tinh:** K-Means clustering có thể được sử dụng để phân vùng hình ảnh vệ tinh thành các vùng đất, nước và hoa màu, giúp theo dõi các biến đổi theo thời gian.
* **Phát hiện biên:** K-Means clustering có thể được sử dụng để phát hiện biên trong hình ảnh. Sau khi phân vùng, biên giữa các vùng được xác định bằng cách so sánh các vùng gần nhau.
* **Loại bỏ nhiễu:** K-Means clustering có thể giúp loại bỏ nhiễu khỏi hình ảnh bằng cách tạo các cụm màu cho các vùng nhiễu và gán lại màu sắc cho các vùng này.
* **Phân tích hình ảnh y tế:** Trong y tế, K-Means clustering có thể được sử dụng để phân vùng hình ảnh y tế, ví dụ như hình ảnh X-quang hoặc MRI, để phát hiện và phân loại các cấu trúc y tế.
* **Trích xuất đặc trưng:** K-Means clustering có thể được sử dụng để trích xuất các đặc trưng quan trọng từ hình ảnh, ví dụ như các điểm đặc trưng được đại diện bởi các tâm cụm.
* **Phân loại nội dung hình ảnh:** K-Means clustering có thể được sử dụng để phân loại và nhóm nội dung hình ảnh dựa trên đặc điểm chung, giúp tạo ra thư viện hình ảnh hoặc phân loại hình ảnh trực tuyến.

# Kết luận

## Đánh giá độ hiệu quả và hạn chế của phương pháp

* + 1. Ưu điểm
* **Đơn giản và hiệu quả:** K-Means clustering là một thuật toán đơn giản và hiệu quả, dễ hiểu và triển khai. Điều này làm cho nó trở thành một công cụ mạnh mẽ để gom cụm dữ liệu trong nhiều tình huống.
* **Có tính linh hoạt:** K-Means clustering có thể được sử dụng trong nhiều tình huống và dữ liệu khác nhau. Nó có thể hoạt động với dữ liệu số học, màu sắc, hình dạng, và nhiều loại dữ liệu khác.
* **Sự hiểu biết về dữ liệu:** Khi người dùng tham gia vào việc chọn K, họ có cơ hội hiểu sâu hơn về dữ liệu và nhu cầu cụ thể của họ. Việc lựa chọn K có thể phản ánh những kiến thức cụ thể về dữ liệu và mục tiêu gom cụm.
* **Hiệu năng tương đối tốt:** K-Means clustering hoạt động tốt trên các tập dữ liệu lớn và có thể được tối ưu hóa để xử lý nhanh chóng dữ liệu hàng loạt.
* **Ứng dụng thực tế rộng rãi:** K-Means clustering được sử dụng rộng rãi trong nhiều lĩnh vực như xử lý hình ảnh, xử lý ngôn ngữ tự nhiên, khoa học dữ liệu, nhiều lĩnh vực khác.
  + 1. Nhược điểm
* **Cần chọn số lượng cụm K tối ưu:** Một trong những hạn chế quan trọng của K-Means là bạn cần xác định số lượng cụm K trước. Sự lựa chọn không đúng về K có thể dẫn đến kết quả phân vùng kém và cần lặp lại quy trình nhiều lần để tìm số lượng cụm tối ưu.
* **Không xử lý dữ liệu nhiễu tốt:** K-Means Clustering không xử lý dữ liệu nhiễu tốt. Dữ liệu nhiễu có thể ảnh hưởng lớn đến kết quả phân vùng và cần phải được xử lý trước hoặc sử dụng các biến thể của K-Means có khả năng chống nhiễu hơn.
* **Không phân biệt kích thước và hình dạng của cụm:** K-Means không nhận biết được sự khác biệt về kích thước hoặc hình dạng của các cụm. Điều này có thể dẫn đến việc cụm có hình dạng không thể tách biệt hoặc bị trộn lẫn.
* **Không thể xác định đối tượng cụ thể trong ảnh:** K-Means Clustering chỉ có khả năng phân vùng ảnh thành các cụm mà không xác định được đối tượng cụ thể trong ảnh. Điều này làm cho nó không phù hợp cho các ứng dụng yêu cầu phát hiện và nhận dạng đối tượng cụ thể.
* **Không xem xét thông tin không gian:** K-Means không xem xét thông tin không gian và tương quan giữa các điểm dữ liệu. Điều này có nghĩa rằng nó có thể phân vùng ảnh dựa trên màu sắc mà không xem xét cấu trúc không gian của đối tượng trong ảnh.

## Hướng phát triển tương lai

* Sử dụng Deep Learning và CNNs: Một trong những xu hướng quan trọng là kết hợp phân vùng ảnh với deep learning, đặc biệt là mạng nơ-ron tích chập (Convolutional Neural Networks - CNNs). CNNs có khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh và có thể cải thiện đáng kể hiệu suất phân vùng.
* Phân vùng đối tượng đa lớp: Phát triển các phương pháp để phân vùng đối tượng trong hình ảnh thành nhiều lớp khác nhau. Điều này có ứng dụng trong thị giác máy tính cho xe tự hành, giám sát video, và thậm chí trong phát hiện các sự kiện đa lớp.
* Sử dụng học tăng cường (reinforcement learning): Sử dụng học tăng cường để tối ưu hóa quá trình phân vùng hình ảnh. Điều này có thể giúp mô hình tự động tìm hiểu cách tăng cường chất lượng phân vùng qua nhiều vòng lặp.
* Phát triển ứng dụng trong y tế và y học: Áp dụng phân vùng ảnh và gom cụm vào lĩnh vực y tế để phát hiện và phân loại tế bào, tổn thương, và các dấu hiệu bệnh lý. Điều này có thể giúp cải thiện chẩn đoán y tế và nghiên cứu trong lĩnh vực này.
* Tích hợp thông tin thời gian: Phát triển phương pháp phân vùng hình ảnh đối tượng đối với dữ liệu dạng video hoặc chuỗi thời gian. Điều này hữu ích trong việc theo dõi và phân tích sự thay đổi của đối tượng qua thời gian.
* Ứng dụng trong thực tế ảo và tăng cường: Sử dụng phân vùng ảnh và gom cụm để tạo ra trải nghiệm thực tế ảo và tăng cường hình ảnh (VR/AR) đáng kể hơn. Điều này có thể bao gồm việc phân vùng không gian thực và tích hợp đối tượng ảo.
* Tích hợp với IoT và dự đoán sự kiện: Sử dụng phân vùng ảnh để phân tích dữ liệu từ cảm biến và camera IoT để dự đoán sự kiện và tình huống thời gian thực, chẳng hạn như phát hiện tai nạn giao thông hoặc sự kiện an ninh.
* Phát triển công cụ và thư viện mở rộng: Xây dựng các thư viện và công cụ mã nguồn mở để giúp cộng đồng phát triển và thử nghiệm các thuật toán và ứng dụng phân vùng ảnh và gom cụm một cách dễ dàng hơn.

# Tài liệu tham khảo

<https://datagen.tech/guides/image-annotation/image-segmentation/>

[https://phamdinhkhanh.github.io/deepai-book/ch\_ml/index\_KMeans.html#](https://phamdinhkhanh.github.io/deepai-book/ch_ml/index_KMeans.html)

<http://cs229.stanford.edu/notes2020spring/cs229-notes7a.pdf>

<https://www.sciencedirect.com/science/article/pii/S0167865504000996>

<https://www.datacamp.com/community/tutorials/introduction-t-sne>

<https://machinelearningcoban.com/2017/01/01/kmeans/>

https://www.udacity.com/blog/2021/05/clustering-algorithms-explained.html