ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Đồ án 1: Color Compression

Toán ứng dụng và thống kê

< Lab01 >

Huỳnh Sỉ Kha - 21127734
Lớp: 21CLC02

Giảng viên:

Vũ Quốc Hoàng
Nguyễn Văn Quang Huy
Lê Thanh Tùng
Phan Thị Phương Uyên

18th July 2023

Mục lục

1 Giới thiệu thuật toán K-Means:

- K-Means là một thuật toán phân cụm (clustering) được sử dụng rộng rãi trong lĩnh vực học máy và khai thác dữ liệu, giúp phân loại điểm dữ liệu thành các nhóm (clustering) dựa trên sự tương đồng của chúng.
- Ứng dụng: thường được sử dụng trong nhiều lĩnh vực, bao gồm xử lý ảnh, phân tích dữ liệu, và các ứng dụng về nhóm khách hàng, nhận dạng chủ đề.
- Thuật toán hoạt động theo các bước sau:
  + Khởi tạo centroids: Centroids được khởi tạo bằng 1 trong 2 cách: chọn ngẫu nhiên k số có giá trị từ 0 → đến 255; hoặc chọn ngẫu nhiên k điểm ảnh trong bức ảnh được chọn. (K là số cụm được người dùng chọn).
  + Gán nhãn cho từng điểm dữ liệu: Sau khi khởi tạo centroids, thuật toán sẽ gán nhãn cho từng điểm ảnh dựa trên khoảng cách Euclidean từ điểm ảnh đó đến các centroids, khoảng cách ngắn nhất sẽ được chọn để gán nhãn cho điểm ảnh.
  + Cập nhật lại centroids: Sau khi gán nhãn, thuật toán sẽ cập nhật lại vị trí của các centroids bằng cách tính trung bình của các điểm ảnh thuộc về centroids đó.
  + Lặp lại quá trình gán nhãn và cập nhật centroids cho tới khi các centroids mới không còn thay đổi hoặc đạt tới số lượng vòng lặp tối đa (hay hội tụ).

  → Kết quả: Sau khi thuật toán đã hội tụ, sẽ thu được bức ảnh có số lượng điểm ảnh giảm xuống còn k điểm, nhưng cơ bản vẫn sẽ bảo toàn được nội dung của ảnh ban đầu.

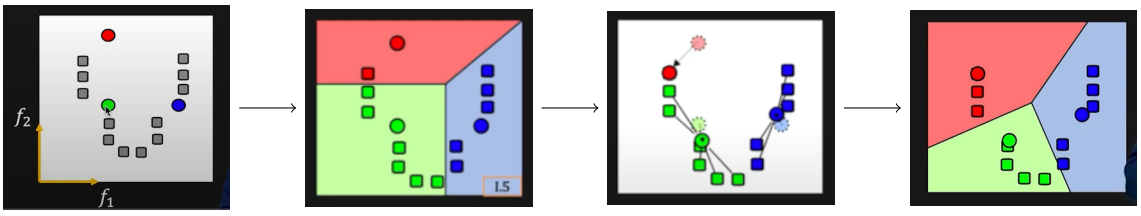

Figure 1: Các bước thực hiện KMeans

2 Ứng dụng K-Means vào bài toán nén ảnh:

2.1 Thư viện hỗ trợ:

- numpy: Tính toán trên ma trận.
- PIL: đọc, lưu ảnh.
- matplotlib: Hiển thị ảnh, vẽ biểu đồ phân tích.
- timeit: Đo thời gian chạy hàm

2.2 Mô tả hàm:

a def get_image_info(image)

Hàm nhận biến image như tham số đầu vào và trả về thông tin của ảnh gồm: chiều rộng, chiều cao và số lượng kênh màu. Biến image được lưu ở định dạng mảng của thư viện numpy.

b def rand_k_centroids(image_1d, k_clusters, random_type):

Hàm trả về mảng có số dòng là số cụm (k_cluster) và số cột là chiều rộng của ảnh được chọn ngẫu nhiên theo 2 kiểu:

- Random: Chọn ngẫu nhiên (số cụm * số cột của ảnh) số tự nhiên từ $0 \rightarrow 255$
- In_pixels: Chọn ngẫu nhiên số cụm dựa trên điểm ảnh của ảnh truyền vào.

c def update_centroids(labels, centroids, image_1d, k_clusters):

Hàm trả về centroids mới dựa trên giá trị trung bình của điểm ảnh thuộc về centroids đó. Trường hợp tồn tại một centroids mà không có điểm ảnh nào thuộc về, thì centroids mới sẽ được gán giá trị bằng centroids đó.

d def visualization(image_1d)

Hàm được dùng để minh hoạ cho quá trình tìm kiếm số cụm tối ưu dựa trên phương pháp Elbow.

- Gọi hàm Kmeans lặp lại k lần với số cụm (k_clusters) từ $1 \rightarrow 10$.
- Với mỗi kết quả trả về, tính toán giá trị tổng bình phương khoảng cách trong cụm (wcss) dựa trên khoảng cách từ điểm đó tới trung tâm của cụm mà nó thuộc về.
- Vẽ biểu đồ minh hoạ dựa trên wcss, từ đó tìm ra điểm 'khuỷu tay' - là vị trí mà giá trị wcss thay đổi không quá đáng kể khi tăng số cụm lên.

$\rightarrow$ Kết quả: Từ đồ thị trên sẽ tìm được số cụm (k_cluster) tối ưu cho bức ảnh.

e def kmeans(image_1d, k_clusters, max_iter=100, random_type='random'):

- **Tham số truyền vào:**
  + image_1d: Hình ảnh được ép về mảng của thư viện NumPy có số dòng bằng (số dòng * số cột) của ảnh ban đầu, và số cột bằng số kênh màu của ảnh đó.
  + k_cluster: Số cụm màu trong ảnh kết quả.
  + max_iter: Số vòng lặp tối đa nếu như chưa thể tìm được điểm hội tụ của các điểm ảnh.
  + random_type: Kiểu chọn ngẫu nhiên.
- **Giá trị trả về:**
  + labels: Mảng gồm chỉ số của centroids mà điểm ảnh tại dòng đó thuộc về.
  + centroids: Mảng gồm k cụm (k_cluster) giá trị của các điểm ảnh được chọn làm centroids.
- **Nội dung thuật toán:**
  + Khởi tạo mảng chứa các centroids.
  + Tính khoảng cách Euclidean dựa trên thuật toán tính Manhattan. Sử dụng thư viện **numpy.newaxis** để khởi tạo 1 chiều không gian mới, chứa khoảng cách từ điểm ảnh đó tới các centroids.

$$D(x, y) = \sum_{i=1}^{k} \|x_i - y_i\| \tag{1}$$

  + Khởi tạo mảng labels gồm các phần tử biểu diễn chỉ số của centroids mà điểm ảnh thuộc về.
  + Tính toán lại giá trị của các centroids, sau đó kiểm tra xem đã hội tụ chưa. Nếu đã hội tụ thì dừng vòng lặp và trả về kết quả.
  + Lặp lại từ bước tính khoảng cách cho tới khi đạt số vòng tối đa.

3 Hướng dẫn sử dụng chương trình:

- Tinh chỉnh giá trị của biến k_cluster để có thể đạt được bức ảnh với số cụm màu mong muốn.
- Chạy thuật toán, nhập vào tên ảnh, và chọn định dạng đầu ra.

```
PS D:\.vscode\HK3\ALA> & C:/Users/huynh/AppData/Local/Programs/Python/Python310/python.exe
Enter image name: test_img.jpg
Output format (png, pdf): png
```

Figure 2: Ảnh gốc

- Kết quả khi chọn phép random 'in_pixels':

Figure 3: cluster = 3, 5, 7

4 Nhận xét kết quả:

4.1 Ứng dụng phương pháp Elbow để tìm K tối ưu:

a Định nghĩa phương pháp Elbow

- Elbow là một phương pháp được sử dụng để tối ưu hóa số lượng cụm trong thuật toán K-Means.
- **Ý tưởng:** Thử nghiệm K-Means với các giá trị K khác nhau và đánh giá hiệu suất của môĩ hình ứng với mỗi giá trị K.
- Giá trị K tối ưu được chọn dựa trên sự biến đổi của giá trị tổng bình phương khoảng cách trong cụm (WCSS - Within-Cluster Sum of Squares) khi K tăng lên.

b Tìm số cụm tối ưu cho bức ảnh

Dùng hàm visualization d để vẽ đồ thị giúp tìm được số cụm tối ưu cho bức ảnh 2 là k $\approx$ 6

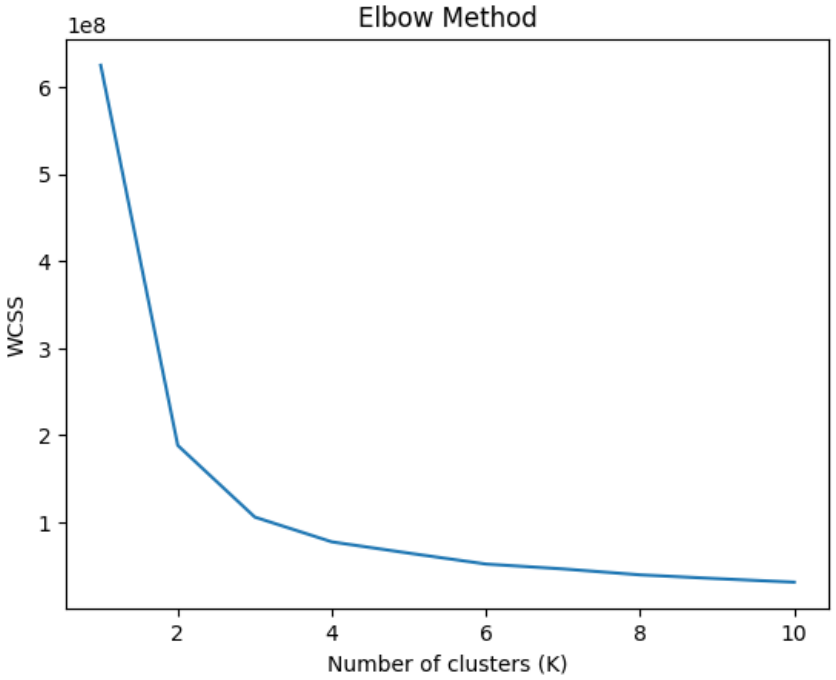

Figure 4: Đồ thị Elbow

c Điểm mạnh của việc chọn được số cụm tối ưu:

Thuật toán KMeans dùng trong quá trình giảm số lượng màu của ảnh nên việc chọn được số cụm tối ưu sẽ có các lợi ích sau:

1. Giảm kích thước dữ liệu: Thay vì phải lưu tất cả pixel của bức ảnh thì chỉ cần lưu các centroids và nhãn các điểm ảnh thuộc về centroids. Điều này giúp giảm được kích thước của dữ liệu, tiết kiệm không gian lưu trữ.
2. Giảm lượng thông tin cần phải truyền đi: Khi chuyển dữ liệu sang nơi khác, nhờ vào đặc tính giảm kích thước dữ liệu, giúp giảm tải lượng thông tin phải truyền tải.
3. Giảm mất mát về chi tiết: Do quá trình chạy thuật toán KMeans sẽ gom nhóm các pixel về còn một số cụm nhất định, điều này khiến cho chi tiết ảnh bị giảm đáng kể. Vì vậy, việc lựa chọn số cụm tối ưu rất quan trọng, giúp phần nào giải quyết được tình trạng mất mát.
4. Đơn giản hoá quá trình phân tích và xử lý ảnh: Thay vì phải xử lí toàn bộ các loại dữ liệu có trong ảnh, KMeans giúp người dùng bỏ qua các giá trị không quan trọng, chỉ chú ý tới các phần điểm ảnh chiếm chủ đạo. Bên cạnh đó, việc phân cụm cũng giúp đơn giản hoá quá trình phân tích, phân loại, và xử lí bức ảnh đó.

5 Điểm yếu của K-Means:

K-Means là một thuật toán giúp phân loại, đơn giản hoá quá trình xử lý ảnh. Tuy nhiên, nó vẫn tồn tại một số khuyết điểm:

1. Khởi tạo centroids không tốt: KMeans phụ thuộc vào quá trình khởi tạo centroids ban đầu. Nếu các centroids được khởi tạo gần nhau hoặc gần các điểm dữ liệu đặc biệt, thuật toán sẽ rơi vào trường hợp cực tiểu địa phương và không tìm được cấu trúc phân cụm tốt. Việc này xảy ra thường xuyên hơn khi sử dụng số ngẫu nhiên thay vì chọn ngẫu nhiên từ những điểm ảnh.

```
Average time for 'in_pixel':  0.33418351000000257
Average time for 'random':  0.41958980999916096
```

Figure 5: Trung bình thời gian chạy 10 lần với k = 6 của random so với in-pixel

2. Cấu trúc dữ liệu phức tạp: KMeans gặp khó khăn trong việc phân cụm các điểm ảnh phức tạp hoặc các cụm có kích thước khác nhau nhưng nằm gần nhau.
3. Số lượng cụm không phù hợp: Việc chọn số lượng cụm quá lớn khiến cho thuật toán trở nên ì ạch, tạo ra phân loại mà người dùng không cần tới. Ngược lại, số cụm quá nhỏ sẽ làm mất đi các chi tiết quan trọng của bức ảnh.

6 Một số kỹ thuật cải thiện hiệu suất của KMeans:

6.1 Các kỹ thuật toàn vẹn cao

Các vấn đề của KMeans được xoay quanh việc chọn số cụm, vì vậy, đưa ra phương pháp chọn số cụm thích hợp sẽ giúp phần nào giải quyết được vấn đề của nó. Một số giải pháp phân cụm:

- Phương pháp Elbow để tìm số cụm tối ưu. Mục 4.1.b
- Phương pháp KMeans++
- Phương pháp chia để trị bisecting

6.2 Phương pháp tương đối:

Phương pháp tương đối thay thế điều kiện hội tụ từ hàm so sánh **numpy.all()** thành hàm so sánh gần bằng **numpy.allclose()** với giá trị tương đối cần được xác định trước. Trong phạm vi đồ án, giá trị tương đối được cài mặc định là: 0.5

- **Điểm mạnh:**
  + Giảm thời gian biên dịch chương trình, nhưng vẫn có thể tạo ra sản phẩm với mức độ sai màu không quá chênh lệch so với phương pháp toàn vẹn.
- **Điểm yếu:**
  + Việc chọn giá trị sai số tương đối lớn sẽ ảnh hưởng tiêu cực đến ảnh kết quả.
  + Trong một số trường hợp, việc so sánh tương đối sẽ dẫn tới kết quả bị sai lệch.

Figure 6: Ảnh gốc và phép so sánh toàn vẹn

Figure 7: Ảnh gốc và phép so sánh tương đối với mức sai số = 0.5 và k = 7

Figure 8: Ảnh gốc và phép so sánh tương đối với mức sai số = 0.5 và k = 100

+ **Nhận xét:** : Độ chênh lệch màu giữa hai kết quả phụ thuộc vào việc chọn giá trị sai số. Nếu có thể chấp nhận sai số cao, thời gian biên dịch sẽ giảm đáng kể. Tuy nhiên, sẽ đánh đổi bằng độ chênh lệch màu, chi tiết của bức ảnh.
+ **Kết luận:** Với một mức độ k = 100, phương pháp này giúp giảm phần lớn thời gian đồng thời trong nhiều tình huống có thể bảo đảm chi tiết và màu sắc.

7 Tài liệu tham khảo:

- Giới thiệu thuật toán K-Means
- Hướng dẫn đồ án 01 - GV: Phan Thị Phương Uyên
- Hình ảnh tham khảo hướng dẫn thuật toán1
- Ứng dụng Manhattan để tính khoảng cách Euclidean.
- Chọn phần tử ngẫu nhiên từ mảng hai chiều
- Phương pháp Elbow để tìm số cụm tối ưu
- Thư viện Numpy
- Thư viện Matplotlib