UNIVERSITY OF SCIENCE

VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

Báo cáo đồ án
Color Compression

TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CNTT

Giảng viên lý thuyết Vũ Quốc Hoàng
Giảng viên thực hành Phan Thị Phương Uyên
Nguyễn Văn Quang Huy
Lê Thanh Tùng

Sinh viên thực hiện Nguyễn Quốc Huy 21127511

Mục lục

1 LỜI NÓI ĐẦU

- Đây là **đồ án 1 - Color Compression** môn **Toán Ứng dụng và Thống kê cho Công nghệ Thông tin**, khoa Công nghệ Thông tin, trường Đại học Khoa học Tự nhiên - Đại học Quốc Gia Hồ Chí Minh. Được thực hiện bởi sinh viên **Nguyễn Quốc Huy, MSSV 21127511**.
- Toàn bộ giải thích, ứng dụng, ví dụ, so sánh, đúc kết và tham khảo về đồ án đều được nêu chi tiết và đầy đủ qua báo cáo.

2 MÔI TRƯỜNG TIẾN HÀNH

- Chương trình được thực hiện trên **Jupyter Notebook**.
- Các thư viện đã được sử dụng trong bài:
  - import numpy as np
  - from PIL import Image
  - import matplotlib.pyplot as plt
  - import time

3 Ý TƯỞNG THỰC HIỆN

3.1 Thuật toán K-means

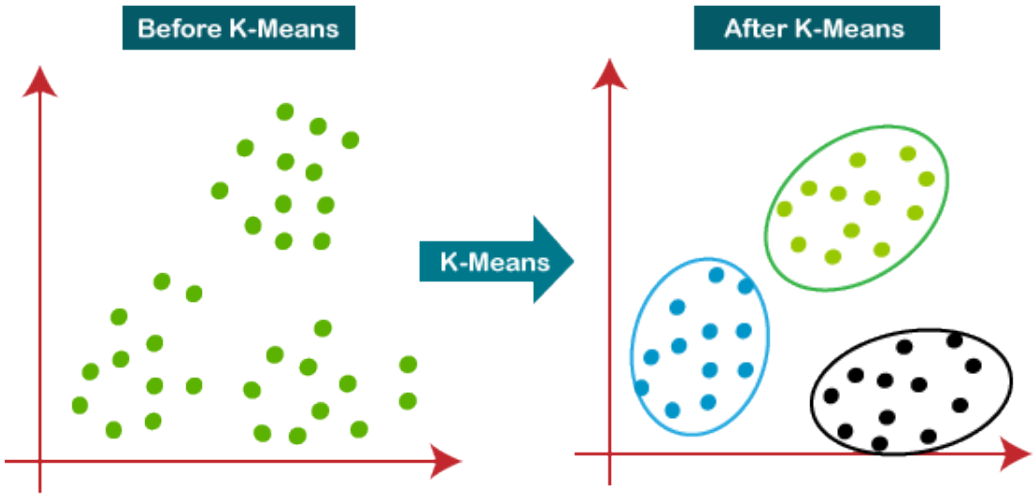
Thuật toán phân cụm **K-means** là một phương pháp được sử dụng trong phân tích tính chất cụm của dữ liệu. Nó đặc biệt được sử dụng nhiều trong khai phá dữ liệu và thống kê. Nó phân vùng dữ liệu thành k cụm khác nhau. Giải thuật này giúp chúng ta xác định được dữ liệu của chúng ta nó thực sử thuộc về nhóm nào.

Mục tiêu của các thuật toán là từ tập dữ liệu khổng lồ, chúng ta biến đổi theo những nhóm dữ liệu đặc trưng trong đó. Từng dữ liệu trong đó thuộc vào nhóm nào? Đó là cái mà thuật toán **K-means** của cần đi tìm câu trả lời.

Ý tưởng của thuật toán phân cụm **K-means** là phân chia 1 bộ dữ liệu thành các cụm khác nhau. Trong đó số lượng cụm được cho trước là k. Công việc phân cụm được xác lập dựa trên nguyên lý: Các điểm dữ liệu trong cùng 1 cụm thì phải có cùng 1 số tính chất nhất

định. Tức là giữa các điểm trong cùng 1 cụm phải có sự liên quan lẫn nhau. Đối với máy tính thì các điểm trong 1 cụm đó sẽ là các điểm dữ liệu gần nhau.

Sơ đồ dưới đây giải thích hoạt động của Thuật toán phân cụm **K-means** với K = 3:



Các bước thuật toán K-Means có thể giải thích theo các bước dưới đây:

- **Bước 1**: Chọn số K để quyết định số cụm.
- **Bước 2**: Chọn ngẫu nhiên K điểm hoặc trọng tâm. (Có thể khác với tập dữ liệu đầu vào).
- **Bước 3**: Gán từng điểm dữ liệu với trọng tâm gần nhất của chúng, tâm này sẽ tạo thành cụm K được xác định trước.
- **Bước 4**: Tính toán trung bình và đặt giá trị của trọng tâm mới vào mỗi cụm.
- **Bước 5**: Lặp lại bước thứ ba, nghĩa là gán lại từng điểm dữ liệu cho tâm mới gần nhất của mỗi cụm cho đến khi hết điểm cần gán.
- **Bước 6**: Nếu có bất kỳ sự thay đổi nào giữa trọng tâm lúc đầu và lúc sau, quay lại bước 4, nếu không thì dừng.
- **Bước 7**: Thuật toán đã hoàn tất.

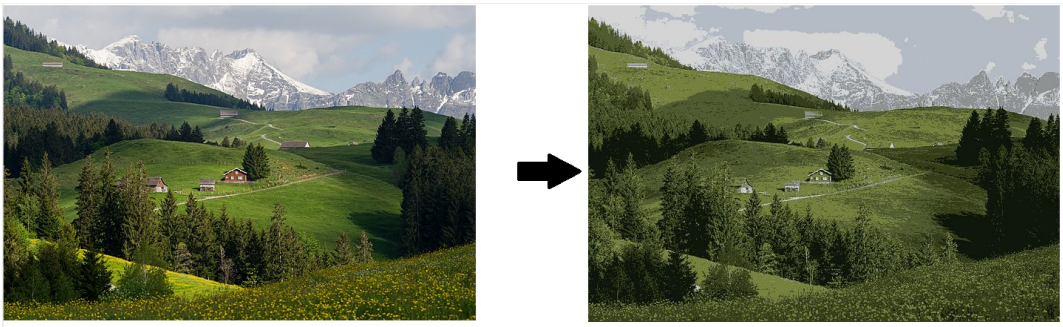
3.2 Hướng thực hiện

Ứng dụng trong vấn đề Color Compression lần này, thuật toán **K-means** sẽ có nhiệm vụ đi tìm ra K cluster tượng trưng cho K màu trong một bức ảnh. Từ một bức ảnh hàng nghìn màu, ta có nhiệm vụ phải đưa chúng về K cụm tượng trưng cho K màu khác nhau nhưng vẫn giữ được đặc trưng cũng như nội dung của bức hình.

Hướng thực hiện như sau:

- Đầu tiên, ta cần tạo dữ liệu khởi tạo ban đầu của K trọng tâm (Ở trong đồ án này, ta có thể chọn ngẫu nhiên 255 màu hoặc lấy màu từ dãy dữ liệu màu có sẵn trong hình.)
- Tiếp theo, ứng với mỗi điểm dữ liệu trên ảnh, ta cần tìm trọng tâm gần nhất ứng với điểm đó để đưa về 1 cụm.
- Sau đó, tính toán lại trọng tâm dựa trên trung bình của các điểm trong một cụm.
- Lặp đi lặp lại các bước này cho đến khi các điểm ảnh đã được gán vào K cụm.
- Các trọng tâm cũng được tính lại cho đến khi không có sự thay đổi đáng kể nào giữa trọng tâm cũ và giá trị trọng tâm đã được tính lại. Các điểm trọng tâm mới có thể xấp xỉ trọng tâm ban đầu, nên chúng ta có thể đặt khoảng hội tụ như một sai số cho mỗi lần tính giá trị trọng tâm.
- Khi trọng tâm mới được giữ nguyên, ta có thể dừng thuật toán và tìm được kết quả.
- Ngoài ra, để có thể thao tác dễ dàng hơn, ta có thể đưa các ảnh từ ma trận nhiều chiều về dạng 1 chiều, đồng thời thu hẹp giá trị của các điểm màu trong khoảng từ [0 đến 1].

Đây là ví dụ về thuật toán **K-means** khi được áp dụng trên 1 bức ảnh:



Ảnh ban đầu

Ảnh giảm số lượng màu

4 MÔ TẢ CÁC HÀM

4.1 find_closest_centroids

Hàm **find_closest_centroids** tính toán khoảng cách giữa mỗi điểm pixel và từng centroid, sau đó trả về chỉ số của centroid gần nhất cho mỗi điểm dữ liệu. Quá trình này là quan trọng trong việc cập nhật các nhóm (clusters) khi thực hiện thuật toán K-means.

Mô tả chi tiết:

- Input: ma trận X và một tập hợp các centroids ban đầu.
- Output: một mảng numpy chứa chỉ số của centroid gần nhất cho mỗi điểm trong X. Chúng sẽ được sử dụng để gán các điểm vào các nhóm (clusters) tương ứng trong thuật toán **K-means**.
- Sử dụng hàm **np.linalg.norm** để tính khoảng cách giữa mỗi điểm và centroid tương ứng. Đồng thời tạo ma trận **distances** có kích thước (m, k), trong đó m và k là số hàng và số cột của ma trận X và centroids tương ứng.
- Do đó, ma trận **distances** chứa các khoảng cách giữa mỗi điểm và từng centroid. Khoảng cách được tính đó còn gọi là chuẩn (norm).
- Ngoài ra, sử dụng **np.argmin(distances, axis=1)** để trả về một mảng chứa chỉ số của centroid gần nhất cho mỗi điểm trong X, theo chiều dọc (axis=1). Trong NumPy, các tham số **axis** được sử dụng để xác định trục (axis) mà một phép toán sẽ được thực hiện qua. Cụ thể:
  - **axis=0** đại diện cho trục dọc (vertical). Khi áp dụng một phép toán, nó sẽ được thực hiện qua các phần tử cùng chỉ số trên các trục ngang khác nhau.
  - **axis=1** đại diện cho trục ngang (horizontal). Khi áp dụng một phép toán, nó sẽ được thực hiện qua các phần tử cùng chỉ số trên các trục dọc khác nhau.
  - **axis=2**, **axis=3**, và những giá trị cao hơn tương ứng với các trục khác (other dimensions) nếu ma trận có nhiều hơn 2 chiều. Trong trường hợp này, **axis=2** được sử dụng để chỉ định trục thứ 3 (third dimension). Khi áp dụng phép toán, nó sẽ tính norm (độ lớn) của các vectơ trong trục thứ 3, tức là tính norm của từng vectơ chứa các giá trị khác nhau giữa mỗi điểm và từng centroid. Tùy thuộc vào bối cảnh sử dụng, các giá trị axis có thể thay đổi để áp dụng phép toán qua các trục khác nhau của ma trận hoặc mảng.

- Ví dụ cách hoạt động:
  Giả sử chúng ta có một tập dữ liệu X và danh sách các centroid centroids như sau:

```
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
centroids = np.array([[2, 2], [6, 6]])
```

  Ta sẽ xét lần lượt điểm đầu tiên [1, 2] được gán vào centroid thứ nhất (centroids[0]) do khoảng cách từ điểm thứ nhất đến centroid thứ nhất gần hơn khoảng cách đến centroid thứ 2 Tiếp theo điểm thứ hai [3, 4] sẽ được gán vào centroid thứ nhất (centroids[0]), Cứ như vậy, điểm thứ ba [5, 6] và thứ tư [7, 8] được gán vào centroid thứ hai (centroids[1]).
  Và đây là kết quả:

```
[0 0 1 1]
```

- Trong hàm này còn sử dụng thêm một thuộc tính đặc biệt: **np.newaxis**
  Trong NumPy, mỗi chiều của một mảng được xác định bởi số lượng phần tử trong chiều đó. Tuy nhiên, có thể có những tình huống khi chúng ta muốn thêm một chiều mới vào mảng để thực hiện các phép toán hoặc tính toán theo các chiều khác nhau. Thì lúc này, **np.newaxis** là một cách đơn giản để thêm một chiều mới vào mảng. Khi sử dụng **np.newaxis**, một chiều mới sẽ được thêm vào vị trí đó và kích thước của mảng sẽ được thay đổi tương ứng.
  Ví dụ:
  Giả sử có một mảng một chiều **a** với 3 phần tử:

```
a = np.array([1, 2, 3])
```

  Nếu muốn biến **a** thành một mảng hai chiều, trong đó chiều thứ hai (axis=1) có kích thước 1, ta có thể sử dụng **np.newaxis** như sau:

```
a_2d = a[:, np.newaxis]
```

  Kết quả là một mảng hai chiều **a_2d** với kích thước (3, 1). Ta có thể thấy **np.newaxis** đã thêm một chiều mới vào mảng **a**, biến nó từ mảng một chiều thành mảng hai chiều.

```
array([[1],
       [2],
       [3]])
```

4.2 calculate_centroids

Hàm **calculate_centroids** sẽ tính toán các centroid cho các nhóm dữ liệu trong một thuật toán gom cụm (clustering). Thuật toán gom cụm như K-means yêu cầu xác định các centroid ban đầu và sau đó cập nhật chúng theo cách tốt nhất để phù hợp với dữ liệu.

Mô tả chi tiết:

- Input: **calculate_centroids** nhận đầu vào là ma trận X, mảng labels chứa nhãn của từng điểm trong X, và số lượng k_clusters của các nhóm cần tạo.
- Ouput: Một ma trận có kích thước (k_clusters, n) chứa các centroid được tính toán từ dữ liệu. Mỗi hàng của ma trận đại diện cho một centroid.
- Trong quá trình tính toán, hàm kiểm tra xem có điểm dữ liệu nào thuộc vào nhóm hiện tại không. Nếu có, hàm tính trung bình của các điểm dữ liệu trong nhóm đó bằng cách sử dụng **np.mean(X[labels == i], axis=0)** và lưu kết quả vào ma trận centroids. Nếu không có điểm dữ liệu nào thuộc vào nhóm, hàm gán centroid tương ứng bằng một mảng zeros.
- Tóm lại, mục đích của hàm **calculate_centroids** là tính toán các centroid cho từng nhóm trong quá trình gom cụm (clustering), để dùng chúng cho các bước tiếp theo của thuật toán.
- Ngoài ra, hàm này được thiết kế theo list comprehension để rút ngắn độ dài các vòng lặp và điều kiện.
- **Ví dụ cách hoạt động:**
  Giả sử có một ma trận X và mảng nhãn labels đã được gán cho từng điểm với K = 3:

```
X = np.array([[2, 3], [2, 4], [3, 5], [6, 2], [7, 3], [6, 4]])
labels = np.array([0, 0, 1, 1, 2, 2])
k_clusters = 3
```

  Đối với labels = 0, các điểm [2, 3] và [2, 4] được lấy trung bình để tính toán centroid ra kết quả [2, 3.5]. Tương tự, centroid cho nhóm 1 là [4.5, 3.5] và centroid cho nhóm 2 là [6.5, 3.5]. Ta được kết quả như hình

```
[[ 2.   3.5]
 [ 4.5  3.5]
 [ 6.5  3.5]]
```

4.3 choose_in_pixels_centroids

Hàm **choose_in_pixels_centroids** sẽ chọn ngẫu nhiên các centroid ban đầu từ dữ liệu của ma trận đầu vào, để sử dụng chúng trong thuật toán gom cụm K-means. Việc chọn ngẫu nhiên giúp khởi tạo các centroid khác nhau và khả năng thuật toán tìm được các nhóm tốt hơn.
Mô tả chi tiết:

- Input: Một ma trận dữ liệu có chứa các điểm. Mỗi hàng của ma trận đại diện cho một điểm dữ liệu. Và số lượng cluster cần chọn (K)
- Output: Một ma trận có kích thước bằng với ma trận ban đầu chứa các centroid được chọn từ dữ liệu.
- Hàm sử dụng **np.random.choice** để chọn k_clusters chỉ số ngẫu nhiên từ phạm vi từ 0 đến **X.shape[0] - 1** (số lượng điểm dữ liệu trong X). Tham số **replace=False** đảm bảo không có sự lặp lại khi chọn các chỉ số.

4.4 choose_random_centroids

Ta cần xác định các centroid ban đầu để bắt đầu quá trình tìm kiếm các nhóm dữ liệu. Hàm **choose_random_centroids** giúp tạo ra các centroid ngẫu nhiên để khởi tạo các nhóm ban đầu.
Mô tả chi tiết:

- Input: Ma trận X và số lượng k_clusters của các centroid cần chọn.
- Output: Kết quả trả về là một ma trận centroids có kích thước của ma trận ban đầu chứa các centroid được chọn ngẫu nhiên.
- Các centroid được chọn ngẫu nhiên trong khoảng giá trị từ 0 đến 255. Mỗi hàng của ma trận centroids đại diện cho một centroid được khởi tạo ngẫu nhiên trong không gian đặc trưng.

4.5 Kmeans

Hàm Kmeans sử dụng các hàm hỗ trợ đã nêu bên trên như **find_closest_centroids** để gán labels cho từng điểm và **calculate_centroids** để tính toán các centroids mới dựa trên từng điểm đã được gán labels.

Hàm cũng hỗ trợ người dùng bằng cách theo dõi lịch sử của các centroid qua các bước lặp bằng cách lưu trữ chúng trong centroid_history.

Mô tả chi tiết:

- Input:
  - **img_1d**: Một mảng 1D đại diện cho ảnh được rút trích thành dữ liệu 1 chiều. Mảng này chứa các giá trị pixel của ảnh.
  - **k_clusters**: Số lượng centroid (cluster) cần tạo trong quá trình gom cụm.
  - **max_iter**: Số lần lặp tối đa cho thuật toán **K-means**.
  - **init_centroids**: Phương pháp khởi tạo ban đầu cho các centroid. Giá trị mặc định là 'random' (chọn ngẫu nhiên).
- Output:
  - **labels**: Một mảng chứa label của từng điểm trong **img_1d**. Label này xác định nhóm mà điểm thuộc về sau khi hoàn thành thuật toán **K-means**.
  - **centroid_history**: Một danh sách lưu trữ lịch sử các centroid trong quá trình thực hiện thuật toán. Mỗi phần tử của danh sách đại diện cho một bước lặp và chứa các centroid tương ứng với bước đó. Nếu ta muốn lấy **final_centroid** ta chỉ cần truy xuất đến phần tử được cập nhật gần nhất trong **centroid_history**
- Ban đầu, hàm khởi tạo danh sách centroid_history để lưu trữ lịch sử của các centroid trong quá trình thực hiện thuật toán.
- Tiếp theo, hàm khởi tạo các centroid ban đầu bằng cách sử dụng init_centroids.
- Trong vòng lặp, thuật toán K-means được thực hiện max_iter lần hoặc cho đến khi sự thay đổi giữa centroid vừa tính và centroid gần nhất xấp xỉ nhau 1 khoảng hội tụ atol = 0.1. Trong mỗi lần lặp, sử dụng hàm **find_closest_centroids** để gán nhãn cho từng điểm dữ liệu trong img_1d dựa trên các centroid hiện tại.
- Sau đó, sử dụng hàm **calculate_centroids** để tính toán các centroid mới dựa trên dữ liệu đã được gán nhãn.
- Nếu sự thay đổi giữa các centroid liên tiếp là nhỏ hơn ngưỡng cho trước, thuật toán dừng lại.
- Kết quả trả về là labels, mảng chứa nhãn của các điểm dữ liệu, và centroid_history, danh sách lịch sử các centroid trong quá trình thực hiện thuật toán.

4.6 Các hàm phụ trợ

- input_information
  Hàm này sẽ yêu cầu người dùng nhập vào các thông tin cần thiết của chương trình như: tên hình, k cluster mong muốn, số lần lặp tối đa (max_iter), cách khởi tạo centroid ('random' hay 'in_pixels') và định dạng của kết quả đầu ra (pdf hay png)

- print_image_information
  Hàm này sẽ dựa vào file hình ảnh mà người dùng cung cấp để trả về các thông tin của ảnh nhu: độ dài, độ rộng, loại ảnh, ...

- image_handler
  Hàm này sẽ làm nhiệm vụ chính trong việc đọc ảnh, đưa ảnh về dạng ma trận 1 chiều. Sau đó thực hiện k-means dựa trên các thông tin khởi tạo mà người dùng đã cung cấp. Và cuối cùng là lấy labels, final_centroids để in ra kết quả trên màn hình cho người dùng.

- save_image
  Hàm này sẽ lưu ảnh đã giảm theo định dạng đã nhập từ người dùng, tên của ảnh đầu ra được quy định là 'output.pdf' hoặc 'output.png'

4.7 Mô tả chương trình

- Đầu tiên, người dùng sẽ được yêu cầu nhập vào tên ảnh, số lượng cluster, số lần lặp tối đa, giá trị khởi tạo (random hay in_pixels) và định dạng file output đầu ra (pdf hay png).

- Tiếp đó, chương trình sẽ hiện ra những thông tin của ảnh đầu vào như: shape, width, height, num_channel.

- Sau đó, ảnh sẽ được sử lý lại thành ảnh 1 chiều, giảm giá trị trong khoảng từ [0 đến 1] và tùy thuộc vào giá trị khởi tạo rồi đưa vào hàm kmeans xử lý.

- Hàm kmeans thực hiện xong sẽ trả về giá trị của các điểm ảnh cuối cùng và hàm xử lý cũng sẽ in ra màn hình file gốc và file đã giảm màu (có nêu rõ chi tiết k, max-iter và init-centroid). Đồng thời bộ đếm thời gian cũng sẽ được thực hiện từ đầu hàm kmeans cho đến khi hàm được thực thi xong.

- Cuối cùng, bức ảnh cuối cùng sẽ được lưu vào hệ thống dựa trên định dạng đầu ra mà người dùng đã chọn (pdf hay png). Hệ thống cũng sẽ thông báo thành công khi file đã được lưu

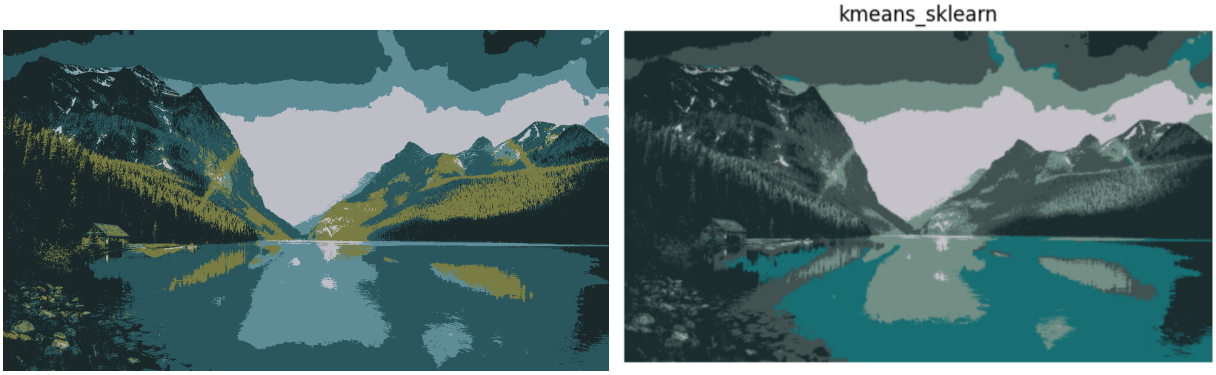
5 SO SÁNH VÀ NHẬN XÉT

5.1 So sánh thuật toán

Sau khi chạy thuật toán trên nhiều hình với số lượng K khác nhau, rút ra được các kết quả. Để đảm bảo tính chính xác tương đối, thư viện scikit-learn cũng được sử dụng để kiểm tra (**Thư viện Scikit-learn không sử dụng trong đồ án này, chỉ dùng để so sánh**):

Hình sau khi đưa về 3 cluster với giá trị khởi tạo là random

Hình sau khi đưa về 5 cluster với giá trị khởi tạo là in_pixels



Hình sau khi đưa về 7 cluster với giá trị khởi tạo là in_pixels

Thời gian đo được ở các trường hợp với các cluster khác nhau

K-cluster	random	in_pixels	Scikit-learn
3	0,851s	0,817s	0.831s
5	1,338s	1,344s	1.213s
7	1,666s	1,207s	1.605s

5.2 Nhận xét và đánh giá

Dựa vào các kết quả chạy được, nhận xét thời gian để cho r bức ảnh phụ thuộc vào số cụm (K-cluster) chỉ định và còn phụ thuộc vào khoảng hội tụ do người dùng chỉ định. Nếu khoảng hội tụ càng nhỏ thì thời gian ra kết quả càng lâu do sự chênh lệch của các centroid mới và centroid cũ. Nhưng đổi lại, chất lượng cũng như độ chính xác về màu sẽ được cải thiện hơn.

Càng nhiều cluster, ta sẽ có được nhiều màu hơn, nhưng thời gian cũng dài hơn. Về khía cạnh giá trị khởi tạo 'random' và 'in_pixels', random sẽ có thể cho ra các giá trị không có trong ma trận ảnh ban đầu nên độ chính xác có thể sẽ không tương đối bằng in_pixels - được khởi tạo từ các giá trị có sẵn từ ma trận ban đầu.

Nếu khoảng hội tụ được người dùng chỉ định là không thay đổi thì mới dừng cập nhật -> thì thời gian cũng sẽ lâu và có khả năng chạm đến max-iter.

6 THAM KHẢO

- Tìm hiểu K-means

- Thuật toán K-means và các ví dụ
- Kiểm tra bằng Scikit-learn
- Sự trợ giúp của cô và thầy trong tiết thực hành

Hết!