Đại học Khoa học tự nhiên
Đại học Quốc gia Thành phố Hồ Chí Minh
Khoa Công nghệ thông tin

AMAS-IT
Báo cáo lab project 1 - Color compression

Lớp: 22CLC04

Thực hiện:
Ngũ Kiệt Hùng - 22127134

Giảng viên hướng dẫn:
Thầy Trần Hà Sơn
Thầy Nguyễn Ngọc Toàn

Tháng 6, 2024

Mục lục

1 Giới thiệu chủ đề

Bài toán phân nhóm (Clustering) là một chủ đề kinh điển trong lĩnh vực học máy. Với lịch sử ra đời đến từ các phương pháp tối ưu trong điện toán và hình học, thuật toán phân nhóm K-Means là một trong các phương pháp gán nhãn (labeling) *K* nhóm dữ liệu trên một tập các dữ liệu trong cùng không gian. Mục đích của thuật toán nhằm tối ưu hóa sai số giữa các điểm phân nhóm (ở đây có thể hiểu là "trung tâm" của các nhóm dữ liệu) và các điểm dữ liệu thuộc nhóm đó. Nói cách khác, thuật toán chọn *K* điểm sao cho tổng bình phương khoảng cách giữa các điểm dữ liệu cho trước là tối thiểu [1].

Giải quyết bài toán K-Means được các nhà nghiên cứu khác cho rằng là NP-Hard, tức rằng bài toán có thể không thể được giải quyết trong thời gian đa thức (ở đây dùng từ *có thể* vì ta vẫn chưa có thể chắc giả thuyết P ≠ NP là đúng). Vì thể, đa phần các cài đặt của bài toán K-Means thường dựa vào các thuật toán sấp xỉ như thuật toán Lloyd có thể tìm kiếm các điểm tối ưu địa phương (local search) vì tính đơn giản cũng như hiệu quả của nó.

Tuy nhiên, các thuật toán thường phải chọn *K* điểm bắt đầu một cách ngẫu nhiên. Điều này có thể dẫn đến vấn đề các phép sấp xỉ sẽ rơi vào các cực trị địa phương, khiến phép phân nhóm không chính xác đối với *K* nhỏ và tập dữ liệu có tính phân hóa cao.

Bài thực hành này có vai trò cài đặt cũng như áp dụng các kỹ thuật lập trình, tính toán ma trận nhằm tối ưu hóa các phép toán của bài toàn phân nhóm K-Means. Đồng thời, bài thực hành sẽ khám phá một số các phương pháp, thuật toán đơn giản nhằm tối ưu hóa độ chính xác của kết quả.

2 Ý tưởng tiếp cận vấn đề

Với lịch sử hình thành và phát triển dày đặc, thuật toán phân nhóm K-Means được tin rằng là *"một trong các thuật toán phân nhóm được sử dụng nhiều nhất trong các ứng dụng khoa học và công nghiệp."* [2]

Trước hết, nhằm phục vụ cho việc hiểu rõ cũng như áp dụng các phương pháp tối ưu cho vấn đề, bài thực hành sẽ đi sơ qua về một số các cơ sở, phân tích toán. Đa số các phân tích và hình họa đều được tham khảo từ [3] của Tiến sĩ Vũ Hữu Tiệp, ngành Học máy Và Thị giác Máy tính, tại Đại học Bang Pennsylvania (Pennsylvania State University), Hoa Kỳ.

2.1 Cơ sở, phân tích toán

Khi nói đến bài toán phân nhóm, cụ thể ở đây là với thuật toán K-Means, ta mong muốn khi cho một tập N các điểm $\boldsymbol{X}=[x_1, x_2, ..., x_n] \in \mathbb{R}^{d \times N}$ cùng một số $\boldsymbol{K}$, ta sẽ thu về một tập các điểm $\boldsymbol{C}=[c_1, c_2, ..., c_k] \in \mathbb{R}^{d \times K}$ sao cho hàm mất mát *được tính bằng tổng bình phương khoảng cách giữa các điểm dữ liệu và điểm trung tâm thuộc* $\boldsymbol{C}$ *gần với điểm dữ liệu đó nhất* [4].

$$\phi = \sum_{x \in \boldsymbol{X}} \min_{c \in \boldsymbol{C}} \|x - c\|^2.$$

Đồng thời, với các điểm dữ liệu đầu vào, ta cũng sẽ thu được nhãn (*label*) cho mỗi điểm dữ liệu x_i thông qua việc tương ứng với mỗi điểm dữ liệu, ta có

$$\boldsymbol{L}_i = [l_{i1}, l_{i2}, ..., l_{ij}, ..., l_{iK}],\ l_{ij} \in \{0, 1\},\ \boldsymbol{L} = [\boldsymbol{L}_1{}^T, \boldsymbol{L}_2{}^T, ..., \boldsymbol{L}_N{}^T] \in \mathbb{R}^{N \times K}.$$

Nếu điểm dữ liệu đó thuộc nhóm i thì $l_i = 1,\ l_j = 0, \forall\, j \neq i$. Vậy, ta có thể tổng quát hóa biểu thức cho giá trị giữa một điểm dữ liệu x_i đến trung tâm k gần nhất:

$$\sum_{j=1}^{K} l_{ij} \|x_i - c_j\|^2 \ = \ l_{ik} \|x_i - c_k\|^2$$

Như thế, ứng với mỗi tập dữ liệu đầu vào và một tập các trung tâm, tập nhãn, ta có hàm sai số:

$$\varphi(\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{C}) = \sum_{i=1}^{N} \sum_{j=1}^{K} l_{ij} \|x_i - c_j\|^2 \quad (*)$$

2.2 Thuật toán K-Means tối ưu hàm mất mát

Như đã đề cập đến phía trên, bài toán tối ưu biểu thức *(*)* chính là mục tiêu của thuật toán gom nhóm. Tuy nhiên, việc tìm các điểm tối ưu sao cho *(*)* có giá trị nhỏ nhất không phải là điều dễ dàng. Đã có nhiều thuật toán chính xác [5] cho nghiệm của bài toán, nhưng đa phần đều có độ phức tạp $O(N^{Kd})$ trong khoảng thời gian lũy thừa, khiến chúng mất đi tính thực tiễn kể cả với d (chiều của các vector điểm dữ liệu) nhỏ.

K-Means là một thuật toán tìm các nghiệm hoặc các điểm cực tiểu cho bài toán bằng cách liên tục tập hợp các điểm dữ liệu gần với các trung tâm và gắn nhãn cho chúng ứng với mỗi trung tâm gần nhất, sau đó thực hiện việc *"kéo"* điểm trung tâm ấy gần về trung vị của các điểm dữ liệu ứng với nhãn của

trung tâm ấy. Bằng cách lặp lại quá trình này liên tục, ta có thể ước lượng được các trung tâm tối ưu địa phương với mỗi *seed (các điểm trung tâm chọn ban đầu)*.

Sử dụng cách giải thích của Tiến Sĩ Tiệp, thì ta có thể giải bài toán *(*)* bằng các bước:

Cố định các điểm trung tâm C. Với mỗi điểm dữ liệu x_i, gắn nhãn cho nó tương ứng với gần vớ mỗi c_i gần nó nhất:

$$k = arg\min_j ||x_i - c_j||^2.$$

*"Vì $||x_i - c_j||^2$ chính là bình phương khoảng cách tính từ điểm x_i tới trung tâm c_j, ta có thể kết luận rằng **mỗi điểm x_i thuộc vào cluster có center gần nó nhất**! Từ đó ta có thể dễ dàng suy ra label vector của từng điểm dữ liệu"*. [3]

Cố định các nhãn L. Với mỗi $c_j \in \boldsymbol{C}$, tìm điểm trung tâm mới sao cho hàm mất mát $\varphi(\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{C}_{new})$ đạt giá trị nhỏ nhất:

$$c_j = \frac{\sum_{i=1}^{N} l_{ij} x_i}{\sum_{i=1}^{N} l_{ij}}$$

Có thể thấy, phần mẫu chính là phép đếm số thành phần trong tập điểm dữ liệu ban đầu $|\boldsymbol{X}|$ (vì mỗi x_i tương ứng với một $l_{ik} = 1$), tử số cũng chính là tổng của các điểm dữ liệu. Hay nói cách khác, c_j được tính bằng **trung bình cộng của các điểm đang trong nhóm *j*.** Đó cũng giải thích có tên K-Means của thuật toán.

Vậy, với các cơ sở trên, ta cũng đã có thể mường tượng ra được thuật toán K-Means được thực hiện như thế nào khi tính toán trên máy tính. Cũng từ đó, có thể suy ra thời gian chạy của một lần cập nhật các trung tâm là $O(N \times K \times d)$; độ phức tạp trong trường hợp tốt nhất cũng chính là $O(N \times K \times d)$, trường hợp các điểm trung tâm được chọn ban đầu có thể hội tụ về tập nghiệm tối ưu $\boldsymbol{C}_{OPT}$. Mặt khác, có thể thấy được, rất khó để có thể đo chính xác do tổng sai lệch bình phương tối ưu ϕ_{OPT} và tổng sai lệch bình phương của từng cách chọn điểm ban đầu là không có giới hạn trên hoặc giới hạn dưới, khiến việc tính toán độ phức tạp của những cài đặt thuật toán trở nên khó khăn.

2.3 Tóm tắt thuật toán trong điện toán

Input: tập các điểm dữ liệu $\boldsymbol{X}$, $|\boldsymbol{X}| = N$và một số $\boldsymbol{K}$ nhóm cần tìm.

Output: $\boldsymbol{K}$ điểm trung tâm và $L = \{l_i \mid l_i \in [0, K),\ l_i \in \mathbb{Z},\ i \in [0, N)\}$ là nhãn ứng với mỗi điểm dữ liệu trong $\boldsymbol{X}$.

Algorithm:

- Chọn $\boldsymbol{K}$ điểm bất kì làm các trung tâm ban đầu.
- Duyệt qua từng điểm dữ liệu x_i và gắn nhãn cho chúng ứng với trung tâm gần nó nhất.
- Với mỗi trung tâm, cập nhật lại vị trí mới của nó bằng cách lấy trung bình cộng tất cả các điểm dữ liệu thuộc nhãn của trung tâm đó.

- Nếu các điểm trung tâm mới trùng với các điểm trung tâm cũ, kết thúc thuật toán.
- Nếu không, quay trở lại bước 2.

3 Cài đặt thuật toán

Đáp ứng với yêu cầu, bài thực hành này chỉ sử dụng các thư viện để tính toán trên dãy số và ma trận (NumPy), nhập xuất hình ảnh (Pillow) và matplotlib dùng để hiển thị các kết quả của thuật toán; cùng một số các module built-in của Python3.

Chi tiết cài đặt của các hàm đã được chú thích rõ ràng trong mã nguồn, nên để tránh khiến cho báo cáo trở nên dày đặc không cần thiết, ta chỉ nêu một số điểm nổi bật.

3.1 Nhập xuất hình ảnh

Về việc nhập xuất hình ảnh, có nhiều phương pháp để nhập xuất bằng thư viện built-in của Python, hoặc bằng các thư viện ngoài. Ở đây, bài thực hành sử dụng Pillow nhằm đơn giản hóa mã nguồn cũng như cung cấp một giao diện cho phép ta chuyển đổi các dữ liệu hình ảnh RGB sang một ma trận trong NumPy một cách dễ dàng.

Lưu ý ở đây rằng, vì thuật toán K-Means sử dụng nhiều phép toán tính khoảng cách, có thể dẫn đến các số cần độ chính xác cao khi biểu diễn floating-point, cũng như hỗ trợ việc chuẩn hóa giá trị RGB về miền [0,1] nhằm biểu diễn trên matplotlib, ta phải khai báo kiểu dữ liệu của ma trận là *float64*.

3.2 Hiển thị hình ảnh

Việc hiển thị hình ảnh cũng khá đơn giản, Pillow cũng đã hỗ trợ một số các hàm như Image.show() nhằm bỏ đi các bước nhọc nhằn để hiển thị hình ảnh. Nhưng để có thể hỗ trợ tốt cho đa thiết bị cũng như hỗ trợ Jupyter Notebook, bài thực hành sẽ sử dụng matplotlib để hiển thị các bảng, hình ảnh. Như đã *lưu ý* phía trên, matplotlib yêu cầu dữ liệu đầu vào cho hiển thị ảnh phải được chuẩn hóa từ khoảng [0,255] về [0,1], nên để đơn giản tổng quát hóa giao diện đầu vào, hàm sẽ tự động chia các thành phần vô hướng của ma trận cho 255.

3.3 Lưu trữ hình ảnh & Tạo dựng lại ma trận dữ liệu

Hàm lưu trữ cũng như san phẳng ma trận 2 chiều cũng khá đơn giản, chỉ việc sử dụng những thủ tục có sẵn của NumPy và Pillow. Lưu ý rằng hình ảnh nên được chuyển qua dạng lưu trữ "RGB" (mặc định ban đầu là dạng tham chiếu điểm ảnh) để quá trình lưu trữ diễn ra suông sẻ.

Cuối cùng chính là thủ tục tạo dựng lại ma trận hai chiều cho dữ liệu ảnh bằng cách gói theo chiều dài dòng của ma trận một chiều kết quả thành các dòng bằng hàm reshape của NumPy.

3.4 Trừu tượng hóa các tham số của thuật toán k-means

Tiếp đến là những thủ tục chính được sử dụng trong chương trình.

Chương trình cho phép 3 phương pháp chọn các trung tâm bắt đầu, tùy thuộc vào nhu cầu cũng như độ phức tạp.

- random: chọn ***K*** điểm bắt đầu một cách ngẫu nhiên, trong không gian giá trị của RGB.
- in_pixels: chọn ***K*** điểm bắt đầu ngẫu nhiên bằng cách chọn một mẫu từ dữ liệu đầu vào.
- kmeans++: phương pháp chọn các điểm bắt đầu bằng cách chọn theo xác suất phân phối của các điểm dữ liệu tỉ lệ theo đóng góp sai số của điểm đó với tổng sai số của tất cả các điểm.

Các phương pháp chọn đều có các đặc tính cũng như những điểm mạnh/điểm yếu riêng, các mục này sẽ được đề cập sau trong báo cáo.

Về phần cài đặt chi tiết:

- random chỉ việc tạo 1 NumPy.ndarray bằng cách chọn ngẫu nhiên $\boldsymbol{K}$ phần tử trong khoảng giá trị của RGB (có thể tổng quát hóa bằng cách cho một biến khoảng giá trị khác). Đối với $\boldsymbol{K}$ nhỏ, việc khởi tạo sẽ không ảnh hưởng nhiều đến thời gian chạy.
- in_pixels cho phép chọn ngẫu nhiên các điểm bắt đầu bằng cách sử dụng thủ tục choice trong module RNG của NumPy. Các điểm được chọn sẽ không được chọn lại.
- k-means++ được cài đặt dựa vào bài báo của Arthur D., Vassilvitskii S. [4] về cách chọn các điểm bắt đầu từng bước một và theo phân phối dựa vào đóng góp khi xét sai số đến trung tâm gần nhất của một điểm so với tổng sai số đến trung tâm gần nhất của tất cả các điểm. Việc chọn các điểm như vậy tăng đáng kể khả năng các điểm bắt đầu của các trung tâm sẽ nằm trong các nhóm phân biệt (trong trường hợp các nhóm được định rõ ràng), có thể một phần nào đó hỗ trợ giải quyết vấn đề dừng ở cực trị địa phương khi giải quyết bài toán *(*)*.

Tuy nhiên, phần cài đặt của k-means++ nếu chỉ sử dụng những tính năng thông thường của Python sẽ tăng thời gian chạy lên đáng kể, vì thể, để thuận tiện thì bài thực hành sẽ đem các dữ liệu về dạng ma trận và thực hiện các phép toán ma trậ thông qua thư viện NumPy, được tăng tốc bằng cách sử dụng điện toán đa chương (multi-threading). Chi tiết về phương pháp k-means++ sẽ được đề cập sau.

Các phương pháp chọn đều được gom vào một cấu trúc dictionary nhằm khiến cho mã nguồn dễ hiểu cũng như tổng quát hóa các biến tham gia vào quá trình khởi tạo. Tất cả mọi hàm khởi tạo chọn các điểm đều chấp nhận tối thiểu 2 tham số: mảng NumPy.ndarray $N \times d$ chiều là mảng một chiều gồm N phần tử, mỗi phần tử có d chiều; tham số thứ hai là số $\boldsymbol{K}$, số nhóm cần phân; tham số thứ 3 là tùy chọn: một số nguyên chỉ giá trị ngẫu nhiên ban đầu cho quá trình chọn.

3.5 Thủ tục k-means

Thủ tục k-means là thủ tục chính để chạy chương trình. Nhằm tạo khả năng sử dụng lại các đoạn mã nguồn, bên trong thủ tục cũng định nghĩa một số hàm như tính toán khoảng cách của từng điểm đến từng trung tâm (*kmeans_data_dist*), gán nhãn cho từng điểm dữ liệu (*kmeans_labeling*), cập nhật tọa độ các điểm trung tâm (*kmeans_iterator*). Tất cả đều nhận các mảng dữ liệu là NumPy.ndarray $N \times d$ đối với tập điểm dữ liệu $\boldsymbol{X}$, NumPy.ndarray $K \times d$ đối với tập các trung tâm $\boldsymbol{C}$. Ngoài ra, cũng còn một thủ tục *convergent* nhằm xét xem các điểm trung tâm đã hội tụ chưa. Phần còn lại của thủ tục được cài đặt theo tóm tắt thuật toán như trên.

Tất cả các hàm tính toán đều được đưa về dạng ma trận nhằm tận dụng tối đa khả năng đa chương có sẵn trong NumPy, việc này đồng thời cũng đã cải thiện tính *"sạch sẽ"* của mã nguồn và cải thiện đáng kể thời gian chạy của thuật toán.

3.6 Hàm main đầu vào

Hàm main đóng vai trò là giao diện về một thủ tục nhập, xử lý và xuất ảnh của chương trình. Các thông số đầu vào đòi hỏi người dùng phải nhập đường dẫn của tập tin ảnh, số nhóm cần gom, số lần chạy tối đa (nhằm tránh trường hợp biến thiên liên tục giữa các giá trị gần nhau khi gom nhóm) và phương pháp khởi tạo. Khi đã xử lý xong, ảnh sẽ được in ra qua matplotlib, đồng thời chương trình cũng sẽ cho phép người dùng lưu ảnh dưới 1 trong 2 định dạng: PDF hoặc PNG.

4 Phương pháp khởi tạo trung tâm k-means++

Với cách cài đặt cũng như thời gian chạy tương đối nhanh, những mặt lợi của thuật toán k-means được đề xuất ban đầu [1] được đánh đổi bằng độ chính xác khi đa số những cách chọn các điểm khởi tạo ban đầu là ngẫu nhiên trong miền giá trị của tập dữ liệu hoặc chỉ dựa vào phương pháp chọn ngẫu nhiên các điểm dữ liệu. Việc này dẫn đến phần lớn các lần chạy thuật toán (được mô tả ở mục thực nghiệm) đều dẫn đến các phân nhóm không chính xác (hoặc đơn giản hơn, $\frac{\phi}{\phi_{OPT}}$ không có giới hạn).

Một đề xuất gần đây do David Arthur và Sergei Vaasilvitskii đã mô tả phương pháp chọn các điểm trung tâm bắt đầu một cách có chọn lọc sao cho ta luôn chọn những điểm có khoảng cách lớn nhất đến trung tâm gần nhất đối với tất cả các điểm dữ liệu. So với việc chỉ chọn ngẫu nhiên các điểm dữ liệu làm điểm bắt đầu (việc này có thể xảy ra trường hợp các trung tâm chọn đều gần nhau), ta cách đều các điểm trung tâm sao cho chúng luôn có thể khám phá những nhóm mới.

Bài báo ban đầu của hai tác giả cũng đã chứng minh rằng thuật toán có tổng bình phương độ sai lệch tối đa trong đa số trường hợp là $O(\log(k))$ [4]; cũng như rằng phương sai của tổng bình phương sai số các điểm dữ liệu ϕ_{D^2} sẽ tuân theo biểu thức: $E(\phi_{D^2}) \leq 8(\ln(k)+2)\phi_{OPT}$. Với những kết quả ấn tượng như vậy, việc lựa chọn k-means++ làm phương pháp chọn khởi điểm có thể coi là chuẩn trong các ứng dụng học máy và cụ thể hơn là bài toán gom nhóm. Tuy nhiên, phương pháp này cũng có những mặt hạn chế, như việc phải tính xác suất chọn cho các điểm dữ liệu với mỗi trung tâm mà ta xét. Việc này dẫn đến quá trình tiền khởi tạo dữ liệu trở nên rất tốn kém về mặt tính toán.

5 Thực nghiệm

Tất cả các cài đặt thuật toán đều được thực hiện trong Python thuần, cùng sự kết hợp của một số thư viện nhập xuất và tính toán khác. Nhằm thể hiện rõ được mối tương quan giữa các thông số đầu vào cũng như tính hiểu quả của các phương pháp chọn điểm trung tâm khởi đầu, các thực nghiệm *test* đều được tính toán và lưu lại. Các tính toán đều có kết quả quang nghiệm cũng như các số liệu về tổng sai số và tổng thời gian chạy tính bằng giây.

5.1 Dataset

Dataset được sử dụng là bức ảnh được lưu trữ dưới dạng RGB gồm các màu tự nhiên, được chọn sao cho các dải màu có sự lặp lại nhưng cũng có sự khác biệt đáng kể trong dải tương phản của nó.

Với mỗi phương pháp chọn điểm bắt đầu {random, in_pixels, k-means++}, bài thực hành sử dụng các thông số K = 3, 5, 7, 10 và 25, với số lần lặp tối đa là 2000 nhằm đo thời gian thực thi, độ chính xác quang trắc và độ chính xác theo hàm mất mát. Với mỗi phương pháp chọn, bài thực hành sẽ chạy 20 trials và cho ra một kết quả quang trắc cho trial cuối cùng và bảng số liệu của mỗi test.

Figure 1: Ảnh mẫu dữ liệu (1200x630x3)

5.2 Kết quả thực nghiệm

Figure 2: Kết quả phân nhóm k-means, phương pháp chọn random, K = 3

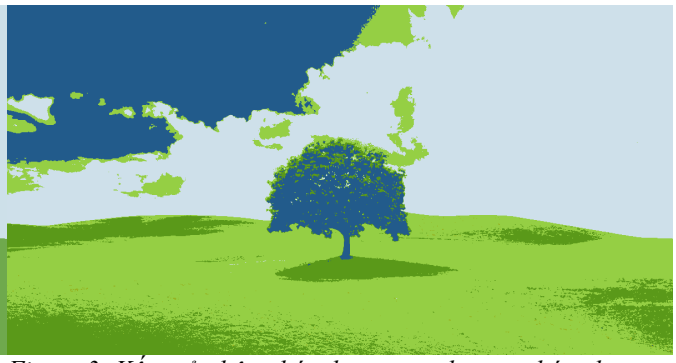

Figure 3: Kết quả phân nhóm k-means, phương pháp chọn random, K = 5

Figure 4: Kết quả phân nhóm k-means, phương pháp chọn random, K = 7

Figure 5: Kết quả phân nhóm k-means, phương pháp chọn random, K = 10

Figure 6: Kết quả phân nhóm k-means, phương pháp chọn random, K = 25

Figure 7: Kết quả phân nhóm k-means, phương pháp chọn in_pixels, K = 3

Figure 8: Kết quả phân nhóm k-means, phương pháp chọn in_pixels, K = 5

Figure 9: Kết quả phân nhóm k-means, phương pháp chọn in_pixels, K = 7

Figure 10: Kết quả phân nhóm k-means, phương pháp chọn in_pixels, K = 10

Figure 11: Kết quả phân nhóm k-means, phương pháp chọn in_pixels, K = 25

Figure 12: Kết quả phân nhóm k-means, phương pháp chọn k-means++, K = 3

Figure 13: Kết quả phân nhóm k-means, phương pháp chọn k-means++, K = 5

Figure 14: Kết quả phân nhóm k-means, phương pháp chọn k-means++, K = 7

Figure 15: Kết quả phân nhóm k-means, phương pháp chọn k-means++, K = 10

Figure 16: Kết quả phân nhóm k-means, phương pháp chọn k-means++, K = 25

K	*random*		*in_pixels*		*k-means++*	
	Average ϕ	Average T	Average ϕ	Average T	Average ϕ	Average T
3	5.060E+09	0.25	4.439E+09	0.24	2.954E+09	0.43
5	3.680E+09	0.38	2.426E+09	0.38	1.384E+09	0.88
7	2.541E+09	0.5	1.317E+09	0.5	7.467E+08	1.38
10	2.052E+09	0.66	8.092E+08	0.65	4.239E+08	2.21
25	6.844E+08	1.27	2.254E+08	1.26	1.326E+08	8.01

Table 1: Kết quả thực nghiệm trên dataset PNG 1200x630x3 (n = 756000, d = 3). Mỗi thử nghiệm được ghi nhận lại tổng sai số bình phương và thời gian chạy T (s).

K	*in_pixels*	*k-means++*	
	Average ϕ difference margin from random	Average ϕ difference margin from random	Average ϕ difference margin from in_pixels
3	12.27%	41.63%	33.46%
5	34.06%	62.38%	42.95%
7	48.17%	70.61%	43.30%
10	60.56%	79.34%	47.62%
25	67.06%	80.62%	41.16%

Table 2: Kết quả so sánh tỉ lệ khác biệt giữa sai số của các phương pháp chọn trung tâm khởi điểm. Kết quả được tính bằng $\Delta\,\phi_{(\text{from},o)} = \frac{\phi_{\text{from}} - \phi_o}{\phi_{\text{from}}}$.

5.3 Nhận xét kết quả

Các kết quả của các lần chạy thực nghiệm để thể hiện trong các bảng và các hình [2] - [16]. Có thể nhận thấy ở các K thấp, hầu như các phương pháp chọn ngẫu nhiên đều không thể hiện được chính xác các nhóm màu của hình ảnh. Mặt khác, có thể thấy phương pháp k-means++ vượt trội hơn nhiều về độ chính xác ở cả sai số mất mát (ở một vài trường hợp là gấp vài lần) và có thời gian chạy tương đối nhanh hơn (khi không xét thời gian khởi tạo ban đầu) do khả năng hội tụ đến các cực trị tương đối nhanh.

Cũng có thể nhận thấy tại một số trường hợp, các phương pháp chọn ngẫu nhiên hoàn toàn không thể phân biệt giữa các dải/nhóm màu mặc dù bức ảnh có độ phức tạp về giá trị màu sắc tương đối thấp, cũng như đây là một bức ảnh có tương đối nhiều điểm dữ liệu phục vụ cho việc gom nhóm.

Với các phương pháp chọn ngẫu nhiên, các vị trí gần các giới hạn của miền giá trị (đen, trắng) rất dễ bị ảnh hưởng bởi nhiễu, và đa số các trường hợp đều đưa đến các kết quả mà tại các vùng chứa nhiều điểm gần giá trị cực ấy, thuật toán không thể phân nhóm một cách tối ưu ($K = \{3, 5, 7\}$). Nếu ta nâng số nhóm lên đáng kể $K = \{10, 25\}$, có thể nhận thấy thuật toán làm việc tốt hơn trong việc phân

biệt mối tương phản trong các dải màu, cũng như có thể định danh một số điểm dữ liệu tốt hơn (bóng cây được thể hiện rõ gradient của nó khi tiến càng xa khỏi cây, các nếp gấp của mây được nêu bật hơn). Tuy nhiên, sự cải thiện này được đánh đổi bởi thời gian chạy thực tế của thuật toán.

Nhưng, nếu xét về tổng thời gian chạy, có thể thấy việc triển khai các điểm trung tâm ban đầu chiếm phần lớn khi sử dụng phương pháp k-means++. Điều này cũng dễ hiểu được vì với mỗi trung tâm cần khởi tạo, ta phải tính tỉ lệ đóng góp sai số của từng điểm với các trung tâm đã khởi tạo trước đó, khiến thời gian khởi tạo có thể nằm trong thời gian $O(K \times N)$. Nhưng về tổng thể, nếu có thể đa chương hóa việc tính toán các điểm trung tâm ban đầu, k-means++ sẽ là phương pháp vượt trội hơn k-means thuần trong rất nhiều trường hợp.

6 Mở rộng

Kể từ sự ra đời chính thức từ vài thập kỉ trước, k-means đã trở thành một trong những tượng đài, cũng như một trong những thuật toán gom nhóm phổ biến nhất trong các ứng dụng cả về khoa học lẫn công nghiệp. Kèm theo đó, đã có nhiều bài báo đề xuất các cách cải tiến đáng kể cho các vấn đề liên quan đến khả năng tìm nghiệm tối ưu cho bài toán [6][7], cũng như áp dụng các phương pháp sấp xỉ theo xác suất dẫn đến nghiệm tối ưu hoặc sử dụng các đặc tính tỉ lệ của các điểm dữ liệu nhằm chọn các điểm một cách có chọn lọc (ở ngành Khoa học máy tính thường gọi là phương pháp tham lam - greedy). Bên cạnh đó, một số phương pháp gom nhóm hiệu quả khác có thể kể đến như *Gaussian expectation-maximization qua Gaussian Mixture Model.* [8]

Ngoài ra, đa số các cài đặt của thuật toán gom nhóm k-means trong các thư viện học máy như scikit đều thực hiện tối ưu hóa các kết quả tính toán bằng cách lặp đi lặp lại thuật toán nhiều lần với cùng một thông số, chỉ khác các trung tâm khởi tạo ban đầu do mỗi lần thực thi sẽ cho ra một tập điểm khởi tạo mới.

7 Kết luận

Bài thực hành đã cài đặt và thực nghiệm trên một tập dữ liệu đầu vào là một bức ảnh cùng các điểm dữ liệu là một vector chứa ba giá trị R, G, B đại diện cho điểm ảnh. Bài thực hành áp dụng các phương pháp tối ưu trong bài toán hồi quy nhằm tìm các vị trí tối ưu cho các điểm trung tâm cần gom K nhóm.

Cùng với đó, bài thực hành cũng khám phá một vài cải tiến, cách thức tối ưu phương pháp sấp xỉ nghiệm của bài toán bằng các phương pháp chọn điểm ban đầu hợp lý như chọn ngẫu nhiên theo phân phối đều đối với tập điểm dữ liệu đầu vào, hoặc chọn theo tỉ lệ đóng góp vào sai số tổng của mỗi điểm dữ liệu ứng với các trung tâm gần nhất đã tìm được theo quá trình lặp đi lặp lại (*iterative process*). Bài thực hành sử dụng các kỹ thuật lập trình cũng như ma trận hóa các dữ liệu nhằm tận dụng tối đa sức mạnh của các thư viện tính toán đại số tuyến tính (NumPy), kết hợp với các thư viện nhập xuất nhằm đơn giản hóa quá trình chạy chương trình.

Lời cảm ơn

Xin cảm ơn các tài liệu liên quan đến NumPy từ thầy Nguyễn Ngọc Toàn cũng như chi tiết các phương pháp thao tác, biến đổi ma trận từ thầy Trần Hà Sơn. Ngoài ra, em cũng xin chân thành cảm ơn Tiến sĩ Vũ Hữu Tiệp vì các tài liệu vô cùng chi tiết cũng như cho phép những đọc giả hứng thú có thể tiếp cận chủ đề Học máy qua cái nhìn tổng quan về lịch sử cũng như các phương pháp thông dụng.

Tham khảo

[1] : Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, pages 129-136. 1982.
[2]: Pavel Berkhin. Survey of clustering data mining (Technical report). *Accrue Software, San Jose*. 2002.
[3]: Vũ Hữu Tiệp. K-means Clustering. . https://machinelearningcoban.com/2017/01/01/kmeans/.
[4] : David Arthur, Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. *Soda*, pages 1027-1035. 2007.
[5] : Mary Inaba, Naoki Katoh, and Hiroshi Ima. Applications of weighted voronoi diagrams and randomization. *SCG '94: Proceedings of the tenth annual symposium*, pages 332-339. 1994.
[6]: J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms. *New York: Plenum Press*. 1981.
[7]: T. Zhang, M. Hsu, U. Dayal. K-harmonic means - a data clustering algorithm (Technical Report HPL-1999-124). *Hewlett-Packard Labs*. 1999.
[8] : Sotirios P. Chatzis, Dimitrios I. Kosmopoulos, Theodora A. Varvarigou. Signal Modeling and Classification Using a Robust Latent Space Model Based on t Distributions. *IEEE Transactions on Signal Processing*, pages 949–963. 2008.