BÁO CÁO ĐỒ ÁN 1 - COLOR COMPRESSION

Đỗ Hoàng Duy Hưng - 23127049

***Tóm tắt nội dung*—Trong đồ án này, em sẽ trình bày về bài toán nén ảnh màu thành ảnh có kích thước nhỏ hơn với thuật toán KMeans Clutering. Và xem xét tính hiệu quả của thuật toán đối với bài toán bằng nhiều cách.**

***Index Terms*—keywords, temperature, xxxx equation, etc.**

I. Tổng quan

Trong các môn học trước như Nhập môn Trí Tuệ Nhân Tạo thì sinh viên chúng em đã được tìm hiểu về bản chất của ảnh là gì trong máy tính. Mỗi bức ảnh là tập hợp của nhiều pixel có giá trị từ 0 đến 255 (đối với ảnh màu thì là tổ hợp từ 0 đến 255 của 3 màu đỏ, xanh lá, xanh dương (RGB)). Tuy nhiên, hiện nay với nhu cầu ngày càng lớn thì ảnh phải có độ sắc nét cao hơn điều này dẫn đến việc số lượng pixel của mỗi bức ảnh càng lớn. Ví dụ : một tấm ảnh full HD có kích thước 1920×1080 pixel thì có đến khoảng 2.1 Triệu pixel. Điều này khiến cho không gian lưu trữ của 1 bức ảnh trở nên lớn hơn mà hệ thống lưu trữ có giới hạn. Từ đó ta đặt giả thuyết về một bài toán nén bức ảnh 1 cách hiệu quả sao cho ảnh vẫn giữ được thông tin nhưng có kích thước nhỏ. Trong bài báo cáo này chúng ta sẽ tìm hiểu về cách nén ảnh thông qua thuật toán KMeans Clustering.

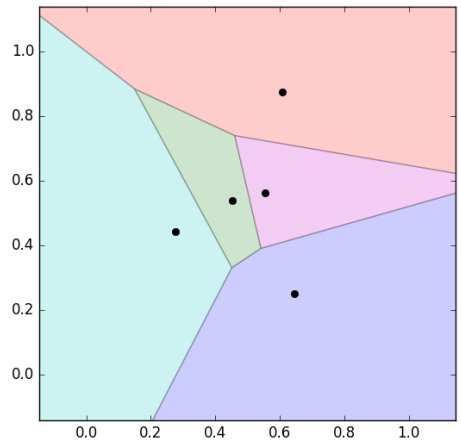
II. Dữ Liệu và Giải Pháp

A. Dữ Liệu

Trong Bài toán này em sẽ tự tạo bộ dữ liệu của riêng mình với số lượng bức ảnh là 10 gồm nhiều loại ảnh khác nhau như ảnh chân dung, phong cảnh, tranh vẽ,..., với số lượng màu ít chi tiết cho đến nhiều chi tiết. Và dataset này em sẽ để nó ở đây [1]

B. Thuật Toán KMeans Clustering

Bản chất thuật toán Kmeans Clutering ta sẽ đi phân cụm những điểm dữ liệu chưa có nhãn sao cho các dữ liệu của cùng một cụm sẽ có tính chất gần giống nhau và ta sẽ dùng 1 centroid để đại diện cho tính chất của cụm đó. Giả sử mỗi cluster có một điểm đại diện (center) màu vàng. Và những điểm xung quanh mỗi center thuộc vào cùng nhóm với center đó. Một cách đơn giản nhất, xét một điểm bất kỳ, ta xét xem điểm đó gần với center nào nhất thì nó thuộc về cùng nhóm với center đó.Hình dưới đây là một hình minh họa cho việc phân chia lãnh hải nếu có 5 đảo khác nhau được biểu diễn bằng các hình tròn màu đen:[2]



Hình 1. Phân vùng lãnh hải của mỗi đảo. Các vùng khác nhau có màu sắc khác nhau.

Vậy để có thể tìm ra k cluster từ các điểm dữ liệu cho trước thì ta nên làm thế nào. Trong cách tiếp cận phổ biến nhất của Machine Learning ta sẽ đi tìm hàm mất mát của việc Clustering và đi tối ưu sao cho hàm mất mát nhỏ nhất. Vậy, ta sẽ định nghĩa hàm mất mát như sau: Nếu ta coi center $\mathbf{m}_k$ là center (hoặc representative) của mỗi cluster và *ước lượng* tất cả các điểm được phân vào cluster này bởi $\mathbf{m}_k$, thì một điểm dữ liệu $\mathbf{x}_i$ được phân vào cluster k sẽ bị sai số là $(\mathbf{x}_i - \mathbf{m}_k)$. Ta sẽ tìm cách để đại lượng sau đây đạt giá trị nhỏ nhất:[2]

$$\|\mathbf{x}_i - \mathbf{m}_k\|_2^2$$

Hơn nữa, vì $\mathbf{x}_i$ được phân vào cluster k nên $y_{ik} = 1$, $y_{ij} = 0,\ \forall j \neq k$. Khi đó, biểu thức bên trên sẽ được viết lại là:

$$y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Sai số cho toàn bộ dữ liệu sẽ là:

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Trong đó $\mathbf{Y} = [\mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_N]$, $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K]$ lần lượt là các ma trận được tạo bởi label vector của mỗi điểm dữ liệu và center của mỗi cluster. Hàm số mất mát trong bài toán K-means clustering của chúng ta là hàm $\mathcal{L}(\mathbf{Y}, \mathbf{M})$ với ràng buộc như được nêu trong phương trình (1).

Tóm lại, chúng ta cần tối ưu bài toán sau:[2]

$$\mathbf{Y}, \mathbf{M} = \arg\min_{\mathbf{Y},\mathbf{M}} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \tag{2}$$

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

Tuy nhiên, bài toán trên là 1 bài toán tối ưu một hàm 2 biến Y và M vậy thì ta thử cách giải quyết như bài toán Linear Regression. Ta sẽ tối ưu bằng cách tính giá trị nhỏ nhất của 1 hàm 2 biến bằng cách tính đạo hàm riêng từng biến sẽ ra được điểm dừng sau đó tính toán được giá trị nhỏ nhất của hàm như sau:

Giả sử đã tìm được các centers, hãy tìm các label vector để hàm mất mát đạt giá trị nhỏ nhất. Điều này tương đương với việc tìm cluster cho mỗi điểm dữ liệu.[2]

Khi các centers là cố định, bài toán tìm label vector cho toàn bộ dữ liệu có thể được chia nhỏ thành bài toán tìm label vector cho từng điểm dữ liệu $\mathbf{x}_i$ như sau:[2]

$$\mathbf{y}_i = \arg\min_{\mathbf{y}_i} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \tag{3}$$

subject to: $y_{ij} \in \{0, 1\} \quad \forall j; \quad \sum_{j=1}^{K} y_{ij} = 1$

Vì chỉ có một phần tử của label vector $\mathbf{y}_i$ bằng 1 nên bài toán (3) có thể tiếp tục được viết dưới dạng đơn giản hơn:[2]

$$j = \arg\min_{j} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Vì $\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$ chính là bình phương khoảng cách từ điểm $\mathbf{x}_i$ tới center $\mathbf{m}_j$, ta có thể kết luận rằng mỗi điểm $\mathbf{x}_i$ thuộc vào cluster có center gần nó nhất! Từ đó ta có thể dễ dàng suy ra label vector của từng điểm dữ liệu.[2]

Giả sử đã tìm được cluster cho từng điểm, hãy tìm center mới cho mỗi cluster để hàm mất mát đạt giá trị nhỏ nhất.

Một khi chúng ta đã xác định được label vector cho từng điểm dữ liệu, bài toán tìm center cho mỗi cluster được rút gọn thành:[2]

$$\mathbf{m}_j = \arg\min_{\mathbf{m}_j} \sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Tới đây, ta có thể tìm nghiệm bằng phương pháp giải đạo hàm bằng 0, vì hàm cần tối ưu là một hàm liên tục và có đạo hàm xác định tại mọi điểm.[2]

Đặt $l(\mathbf{m}_j)$ là hàm bên trong dấu $\arg\min$, ta có đạo hàm:

$$\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = 2 \sum_{i=1}^{N} y_{ij} (\mathbf{m}_j - \mathbf{x}_i)$$

Giải phương trình đạo hàm bằng 0 ta có:

$$\mathbf{m}_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij} \mathbf{x}_i \quad \Rightarrow \quad \mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

Vậy ta đã tìm ra được nghiệm của bài toán này là một nghiệm tương đối đẹp đó là trung bình cộng của các điểm trong cluster. Từ đó ta xây dựng thuật giải như sau:[2]

Algorithm 1 Thuật toán K-Means Clustering

Require: Dữ liệu $\mathbf{X}$ gồm N điểm dữ liệu, số lượng cluster K

Ensure: Các center $\mathbf{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_K\}$ và label vector $\mathbf{Y}$

Khởi tạo K center ban đầu $\mathbf{M}^{(0)}$ bằng cách chọn ngẫu nhiên K điểm từ $\mathbf{X}$

repeat

Gán mỗi điểm dữ liệu $\mathbf{x}_i$ vào cluster có center gần nó nhất:

$$y_{ij} = \begin{cases} 1 & \text{nếu } j = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|_2^2 \\ 0 & \text{ngược lại} \end{cases}$$

Cập nhật lại các center mới:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

until Không có sự thay đổi trong các label vector $\mathbf{Y}$

C. Áp dụng thuật toán KMeans Clustering vào bài toán Color Compression

Đối với bài toán Color Compression thì ta sẽ định nghĩa mỗi màu sắc của pixel là 1 cluster riêng biệt. Vì vậy ta có tới hơn 16 triệu màu khác nhau. Đây là con số quá lớn nên ta sẽ tìm cách giảm số lượng màu xuống. Ta sẽ tìm cách chọn 1 cách hiệu quả K cluters 1 cách hiệu quả và tìm cách đưa hình ảnh về không gian con này bằng thuật toán Kmeans Clutering như sau:[3]

- Chọn ngẫu nhiên K điểm ban đầu làm center màu: $\mathbf{m}_1, \ldots, \mathbf{m}_K$.
- Gán mỗi điểm dữ liệu $\mathbf{x}_i$ vào cluster có center gần nhất:

$$j = \arg\min_{k} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2$$

- Cập nhật lại mỗi center:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

- Lặp lại hai bước trên cho đến khi không còn thay đổi (hội tụ).

D. Một số lưu ý

Thuật toán KMeans Clustering đôi khi có thể xuất hiện một số hạn chế không mong muốn như bức ảnh test có thể khiến thuật toán hội tụ chậm, hoặc là clusters nằm bên trong cluster. Tôi sẽ giải quyết những hạn chế này bằng cách giới hạn số lần lặp của thuật toán cũng như số lượng cluster.

III. Chi tiết hoạt động

- `def read_img(img_path)`: Đọc ảnh từ đường dẫn và chuyển ảnh sang định dạng RGB. Trả về ảnh dưới dạng mảng `NumPy array` 3 chiều có kích thước `(height, width, 3)`.[4][5]
- `def show_img(img_2d)`: Hiển thị ảnh RGB đầu vào bằng thư viện `matplotlib`, tắt trục tọa độ.[6]

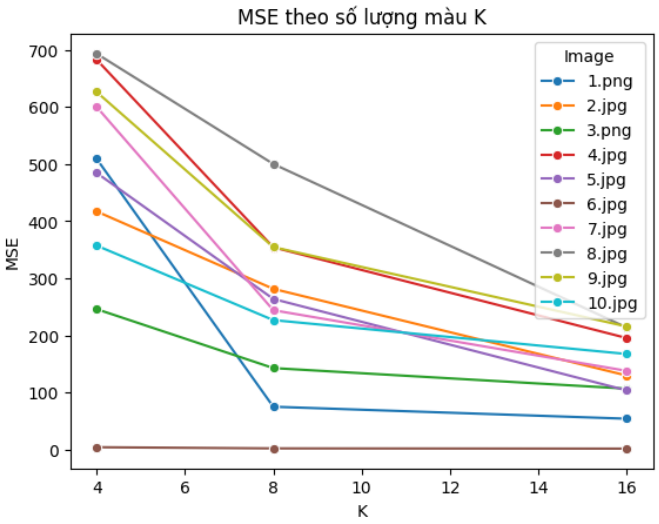
- `def save_img(img_2d, img_path)`: Lưu ảnh 2D dưới dạng file ảnh với tên là `output.png` sử dụng thư viện `PIL`.[4]
- `def convert_img_to_1d(img_2d)`: Chuyển ảnh RGB từ dạng `(height, width, 3)` sang dạng 1 chiều `(height × width, 3)` để thuận tiện cho thuật toán K-Means.[5][4]
- `def kmeans(img_1d, k_clusters, max_iter, init_centroids='random')`: Cài đặt thuật toán K-Means. Hàm nhận vào ảnh dạng 1D, số lượng cụm k, số lần lặp tối đa và phương pháp khởi tạo centroid. Trả về: `centroids` và `labels` ứng với từng pixel.[2][5]
- `def generate_2d_img(img_2d_shape, centroids, labels)`: Sinh lại ảnh 2D ban đầu từ label của từng pixel và centroid tương ứng. Đầu ra là ảnh RGB có cùng kích thước với ảnh gốc.[3][5]
- `def MSE(img_2d, compressed_img)`: Tính sai số trung bình bình phương (Mean Squared Error - MSE) giữa ảnh gốc và ảnh nén. Giá trị MSE càng thấp, ảnh nén càng giống ảnh gốc.[7]
- `def Compress_Ratio(img_2d, compressed_img)`: Tính tỷ lệ nén dựa trên kích thước file ảnh gốc và file ảnh nén. Dùng thư viện `os.path.getsize` để lấy kích thước file (tính theo byte).[8]
- `def main()`: Hàm cho phép nhập vào tên tập tin ảnh mỗi lần chương trình thực thi, cho phép lưu ảnh với tối thiểu 2 định dạng là pdf và png
- `Phần Evaluate Metrics`: Phần evaluate metrics tính toán các đại lượng trong các thang đo như MSE, Compress Ratio sau đó trực quan hoá các giá trị lên các dạng đồ thị phù hợp từ đó rút ra nhận xét.(tqdm dùng để visualize giai đoạn tính toán cho dễ theo dõi)[6][9][4][5]

IV. Đánh giá hiệu năng mô hình

A. Thang Đo MSE

Đầu tiên ta sử dụng độ đo phổ biến nhất với các bài toán machine learning đó là thang đo MSE. Công thức là tính trung bình tổng hiệu bình phương giữa các pixel trong 2 bức ảnh trước và sau khi nén. MSE càng nhỏ thì hai bức ảnh càng giống nhau.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(I_i - \hat{I}_i\right)^2 \tag{1}$$



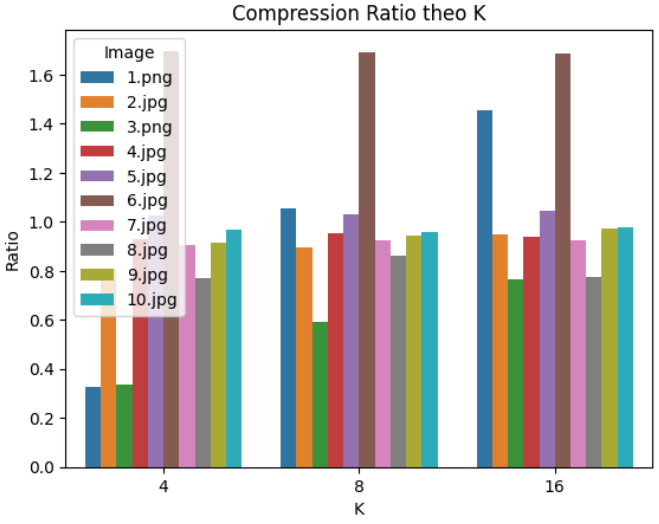
Hình 2. Đo MSE theo số lượng màu K

Có thể thấy từ biểu đồ, MSE có xu hướng giảm khi số lượng màu K tăng. Xu hướng này đúng với hầu hết các ảnh thử nghiệm, ngoại trừ test số 6. Nguyên nhân là do ảnh số 6 là một logo chỉ chứa hai màu cơ bản: màu nền và màu biểu tượng, nên việc tăng K không làm giảm sai số nhiều.

B. Thang Đo Compress Ratio

Thang đo này tính toán hiệu quả của việc nén ảnh. Công thức là tính tỉ lệ của bức ảnh sau khi nén chia cho bức ảnh trước khi nén. Compress Ratio càng nhỏ chứng tỏ tỉ lệ ném càng cao => hiệu quả nén ảnh càng tốt.

$$\text{Compression Ratio} = \frac{\text{Size}_{\text{compressed}}}{\text{Size}_{\text{original}}} \tag{2}$$



Hình 3. Đo Compression Ratio theo số lượng màu K

Hình minh họa cho thấy tỉ lệ nén (Compression Ratio) có xu hướng tăng khi số lượng màu K tăng, do ảnh sau nén giữ nhiều thông tin hơn. Một số ảnh đơn giản như `5.jpg`, `7.jpg`,

`10.jpg` có tỉ lệ gần như không đổi, trong khi ảnh `6.jpg` có tỉ lệ cao ở mọi mức K, cho thấy cấu trúc màu phức tạp. Các ảnh PNG như `1.png`, `3.png` đạt tỉ lệ nén thấp nhất khi $K = 4$, do đặc điểm nén không mất dữ liệu ban đầu. Biểu đồ hỗ trợ đánh giá mức K phù hợp để cân bằng giữa chất lượng và dung lượng. Code có thể khiến ảnh lớn hơn ảnh ban đầu dù việc này nghe sai với việc nén ảnh vì nhiều nguyên nhân như : Ảnh gốc có định dạng nén mạnh như JPEG, dùng K quá lớn hoặc ảnh đầu vào có đặc tính khiến PNG nén kém.[10]

V. Acknowledgement

Đồ án có sự giúp đỡ của bạn Bùi Huy Giáp trong việc suy nghĩ nên chọn Metrics nào, trong quá trình đó chúng em đã bỏ phần evaluate phân bố màu bằng histogram. Ngoài ra, đồ án còn có sự hỗ trợ của ChatGPT trong việc def hàm Main.

Tài liệu

[1] Đ. H. D. Hưng, "Dataset nhiều ảnh," Data mới được crawl https://github.com/RitaKaniska/Applied-Math-Project-2---Color-Compression, 2025.

[2] V. H. Tiep, *Bài 4: K-means clustering*, https://machinelearningcoban.com/2017/01/01/kmeans/, Accessed: June 22, 2025, 2017.

[3] V. H. Tiep, *Bài 5: K-means clustering: Simple applications*, https://machinelearningcoban.com/2017/01/04/kmeans2/, Accessed: June 22, 2025, 2017.

[4] J. A. Clark and contributors, *Pil document*, https://pillow.readthedocs.io/en/stable/reference/Image.html, 2017.

[5] T. Oliphant, *Pil document*, https://numpy.org/neps/, 2017.

[6] J. D. Hunter, *Matplotlibpyplot document*, https://matplotlib.org/stable/tutorials/pyplot.html, 2017.

[7] C. F. Gauss and P.-S. Laplace, *Mean squared error*, https://www.geeksforgeeks.org/python/python-mean-squared-error/, 2025.

[8] https://en.wikipedia.org/wiki/Data_compression_ratio, 2025.

[9] https://seaborn.pydata.org/, 2025.

[10] ://chatgpt.com/c/685980ea-e768-800d-8ed0-2d2503673c2c.