TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC #1

REPORT

Tài liệu này mô tả nội dung đồ án môn học cho môn học Toán ứng dụng và thống kê cho Công nghệ thông tin

Họ tên sinh viên: Nguyễn Văn Đạt

MSSV: 20127132

Khoa Công nghệ Thông tin
Đại học Khoa học Tự nhiên TP HCM
Tháng Thg6-22

MỤC LỤC

1

Tổng quan

Thông tin nhóm

MSSV	Họ tên	Email
20127132	Nguyễn Văn Đạt	20127132@student.hcmus.edu.vn

Thông tin đồ án

Tên đồ án: Color Compression	
Công cụ hướng dẫn	Visual Studio Code Jupyter Notebook

2

Nội dung đồ án

2.1

Giới thiệu

Một bức ảnh có thể lưu trữ dưới ma trận của các điểm ảnh. Có nhiều loại ảnh được sử dụng trong thực tế, ví dụ: ảnh xám, ảnh màu,...

Đối với ảnh xám, một điểm ảnh sẽ là được biểu diễn bằng giá trị không âm.

Ví dụ ta có thể dùng ma trận này:

$$\begin{bmatrix} 255 & 0 & 0 & 0 & 255 \\ 255 & 0 & 255 & 0 & 255 \\ 255 & 0 & 255 & 0 & 255 \\ 255 & 0 & 255 & 0 & 255 \\ 255 & 0 & 0 & 0 & 255 \end{bmatrix}$$

có thể biểu diễn cho ảnh xám có nội dung như sau:

```
import matplotlib.pyplot as plt
plt.imshow(np.array([[255, 0, 0, 0, 255], [255, 0, 255, 0, 255], [255, 0, 255, 0, 255],
                     [255, 0, 255, 0, 255], [255, 0, 0, 0, 255]]), cmap='gray', vmin=0,
vmax=255);
```

Ảnh màu được sử dụng phổ biến là ảnh RGB, trong đó, mỗi điểm ảnh sẽ lưu trữ 3 thông tin kênh màu (mỗi kênh màu 1 byte) là: R (red - đỏ), G (green - xanh lá), B (blue - xanh dương). Ta có thể sử dụng ma trận:

$$\begin{bmatrix} [255 & 255 & 255] & [255 & 0 & 0] & [255 & 0 & 0] & [255 & 0 & 0] & [255 & 255 & 255] \\ [255 & 255 & 255] & [255 & 0 & 0] & [255 & 255 & 255] & [255 & 0 & 0] & [255 & 255 & 255] \\ [255 & 255 & 255] & [255 & 0 & 0] & [255 & 255 & 255] & [255 & 0 & 0] & [255 & 255 & 255] \\ [255 & 255 & 255] & [255 & 0 & 0] & [255 & 255 & 255] & [255 & 0 & 0] & [255 & 255 & 255] \\ [255 & 255 & 255] & [255 & 0 & 0] & [255 & 0 & 0] & [255 & 0 & 0] & [255 & 255 & 255] \end{bmatrix}$$

để biểu diễn cho ảnh màu có nội dung sau:

```
plt.imshow(np.array(
      [[[255, 255, 255], [255, 0, 0], [255, 0, 0], [255, 0, 0], [255, 255, 255]],
      [[255, 255, 255], [255, 0, 0], [255, 255, 255], [255, 0, 0], [255, 255, 255]],
      [[255, 255, 255], [255, 0, 0], [255, 255, 255], [255, 0, 0], [255, 255, 255]],
      [[255, 255, 255], [255, 0, 0], [255, 255, 255], [255, 0, 0], [255, 255, 255]],
      [[255, 255, 255], [255, 0, 0], [255, 0, 0], [255, 0, 0], [255, 255, 255]]]));
```

Như vậy, số màu trong ảnh RGB có thể là $256^3 \approx 1.7 \times 10^7$. Với số lượng màu khá lớn, khi lưu trữ ảnh có thể sẽ tốn chi phí lưu trữ. Do đó bài toán đặt ra là giảm số lượng màu để biểu diễn ảnh sao cho nội dung ảnh được bảo toàn nhất có thể.

Cho ảnh như sau:

Ảnh ban đầu Ảnh giảm số lượng màu

Trong ví dụ trên, số lượng màu cho ảnh ban đầu là 439 màu. Sau khi giảm số lượng màu xuống còn 5, ảnh không còn được chi tiết nhưng cơ bản vẫn bảo toàn nội dung của ảnh ban đầu.

Để thực hiện giảm số lượng màu, ta cần tìm ra các đại diện có thể thay thế cho một nhóm màu. Cụ thể trong trường hợp ảnh RGB, ta cần thực hiện gom nhóm các pixel ($\mathbb{R}^3$) và chọn ra đại diện cho từng nhóm. Như vậy, bài toán trên trở thành gom nhóm các vec-tơ.

2.2

Yêu cầu

Trong đồ án này, bạn được yêu cầu cài đặt chương trình giảm số lượng màu cho ảnh sử dụng thuật toán K-Means.

Các thư viện được phép sử dụng là: NumPy (tính toán ma trận), PIL (đọc, ghi ảnh), matplotlib (hiển thị ảnh).

Một số gợi ý:

- Đọc ảnh: PIL.Image.open(...)

- Hiển thị ảnh: matplotlib.pyplot.imshow(...)

- Thay đổi shape cho np.ndarray: np.reshape(...)

- Khai báo hàm gợi ý cho thuật toán K-Means:

```
def kmeans(img_1d, k_clusters, max_iter, init_centroids='random'):
    '''
    K-Means algorithm

    Inputs:
        img_1d : np.ndarray with shape=(height * width, num_channels)
            Original image in 1d array

        k_clusters : int
            Number of clusters

        max_iter : int
            Max iterator

        init_cluster : str
            The way which use to init centroids
            'random' --> centroid has `c` channels, with `c` is initial random in
[0,255]
            'in_pixels' --> centroid is a random pixels of original image

    Outputs:
        centroids : np.ndarray with shape=(k_clusters, num_channels)
            Store color centroids
```

```
        labels : np.ndarray with shape=(height * width, )
            Store label for pixels (cluster's index on which the pixel belongs)

    '''
```

*Lưu ý: Không sử dụng K-Means đã được cài đặt sẵn trong các thư viện trong bài nộp. Bạn có thể sử dụng K-Means trong scikit-learn để kiểm tra.

*Sinh viên cần viết chương trình main cho phép:

- Người dùng nhập vào tên tập tin ảnh mỗi lần chương trình thực thi (gợi ý sử dụng input() trong Python)

- Lựa chọn định dạng lưu ảnh đầu ra gồm: png, jpg, pdf

2.3

Ý tưởng thực hiện

Tạo một centroids (3 x k_cluster) ngẫu nhiên hoặc được tạo thành từ các pixel (“random” hoặc “in_pixels”) có giá trị trong khoảng [0, 255].

Dùng max_iter để giới hạn số lần tìm mean:

-Dùng hàm norm trong Numpy để tính module hiệu của ndarray của hình ảnh và centroids, rồi đánh dấu các vị trí norm nhỏ nhất của từng cột (shape sau khi norm là (3 x (pixel_dài x pixel_rộng)) bằng Numpy.argmin có giá trị trong khoảng [0, k_cluster - 1] và lưu vào labels.

-Tiếp đó tạo một ndarrray lưu các giá trị trung bình để cập nhật vào centroids bằng việc tìm các vector có vị trí trong ndarrray của hình ảnh trùng với vị trí trong labels, chúng được gom nhóm lại bằng các giá trị nhỏ nhất được đánh dấu bằng Numpy.argmin, rồi dùng hàm mean hình trung bình của từng cột. Hoàn thành việc lưu các giá trị thì cập nhật vào centroids.

Cuối cùng cập nhật lại labels và trả về giá trị centroids, labels. Khi thực hiện hàm thì phải cập nhật centroids vào ndarray hình ảnh, rồi reshape lại và xuất hình ảnh.

2.4
Mô tả hàm

Sử dụng và thiết kế lại khuôn hàm kmeans được yêu cầu:

-Đầu tiên là khởi tạo centroids với hàm random có sẵn trong Numpy.

-Đi vào vòng lặp, khởi tạo labels dựa trên argmin (tìm vị trí nhỏ nhất trong hàng hoặc cột) và linalg.norm (tính module hiệu hai ndarray của hình ảnh và centroids theo hàng hoặc cột tùy theo thuật toán) của Numpy. Tạo mới một ndarray, tìm tất chuyển các array labels về cùng kiểu dữ liệu với centroids rồi cập nhật nó.

-Cuối cùng, cập nhật lại labels và trả về giá trị như khuôn hàm đã quy định.

Có tạo thêm hàm:

```
def initCentroids(img_1d, k_clusters, init_centroids)
```

initCentroids là hàm dùng để khởi tạo centroids thành một ndarray.

```
    def OneImage(img, k_cluster, img_height, img_width, init_centroids = 'random')
```

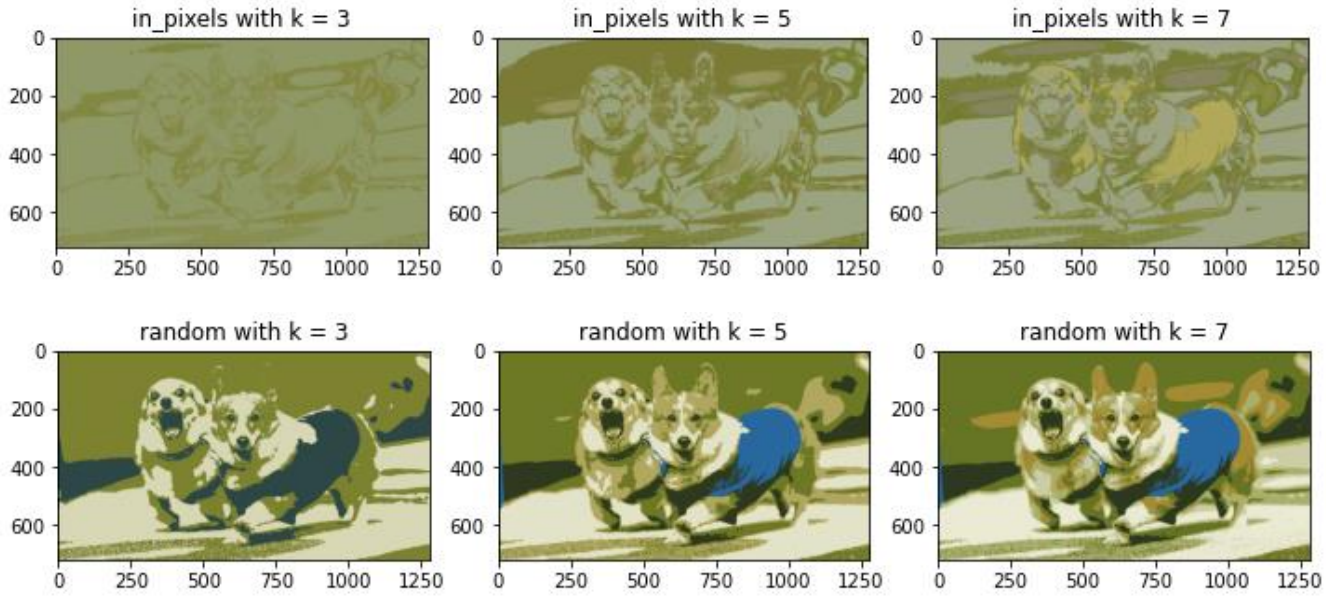
OneImage là hàm xuất **một** ảnh, nó được tạo ra nhằm giảm thời gian nhập xuất, chạy thử chương trình. Đồng thời làm gọn mã nguồn hơn.

```
     def ListTestImage(img,img_load, img_height, img_width)
```

ListTestImage là hàm xuất, gộp **nhiều** ảnh, nó được tạo ra nhằm giảm thời gian nhập xuất, chạy thử chương trình. Đồng thời làm gọn mã nguồn hơn.

2.5
Kết quả


Original Image
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


in_pixels with k = 3
in_pixels with k = 5
in_pixels with k = 7
random with k = 3
random with k = 5
random with k = 7
0
200
400
600
0
250
500
750
1000
1250

3

Kết Luận

-Kết quả là chấp nhận được, tuy nhiên nếu so sánh với Kmeans của scikit-learn thì độ hiệu quả khi giảm màu là không bằng. Với các ảnh giảm màu sử dụng max-iter thấp thường bị lỗi, khó nhận ra được hình ảnh gốc.

-Với init_centroids là “random” thì cả tốc độ và chất lượng màu ảnh tốt hơn rất nhiều so với “in_pixels”.

4

Tài liệu tham khảo

- Tài liệu do giảng viên cung cấp

- Tài liệu từ web:

https://numpy.org/

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