K-means Algorithm Visualization with SDL2/C++

Nguyen Dinh Hoang, *20021361*
University of Engineering and Technology - VNU

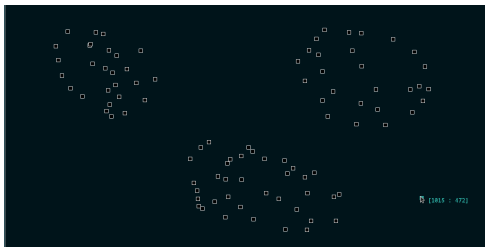
***Tóm tắt nội dung*— Bài toán phân cụm có nhiệm vụ phân chia một tập hợp các đối tượng (còn gọi là các thành viên) thành các nhóm khác nhau (gọi là các cụm) dựa trên các đặc điểm của đối tượng. Các thành viên của một nhóm sẽ có nhiều điểm tương đồng hơn so với những thành viên trong nhóm khác. Bài viết này nói về một phương pháp phân cụm được gọi là thuật toán K-Means dưới góc độ toán học và quá trình tạo lên nó từ thư viện SDL2 và ngôn ngữ C++.**

I. Giới Thiệu

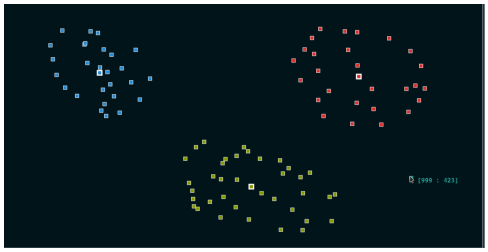
BÀI toán phân cụm có nhiều ứng dụng khác nhau: khai thác dữ liệu học máy và khám phá kiến thức, nén dữ liệu, nhận dạng mẫu và phân loại mẫu.

Mục tiêu của Thuật toán K-Means là phân tách chính xác các đối tượng trong tập dữ liệu thành các nhóm dựa trên thuộc tính của đối tượng. Ví dụ, các đối tượng có thể là ngôi nhà và thuộc tính của chúng là kích thước, số tầng, vị trí, mức tiêu thụ điện năng mỗi năm, v.v. Mục đích là phân loại tập dữ liệu nhà thành các nhóm giàu, trung bình, nghèo. Trong trường hợp đó, tất cả các thuộc tính của các ngôi nhà phải được xử lý để chuyển thành số để tạo ra một vector, quá trình này được gọi là vectơ hóa.

Một ví dụ khác, lấy các điểm trong bảng làm đối tượng và mỗi đối tượng có hai thuộc tính là vị trí trục x và trục y, với đầu vào K = 3.



Hình 1. Các đối tượng chưa được chia nhóm.



Hình 2. Các đối sau khi được chia nhóm.

II. Giải Thích Thuật Toán

Thuật toán nhận vào một tập hợp:

$$X = [x_1, x_2, ..., x_n] \in \mathbf{R}^{d.n}$$

d là chiều của mỗi vector, n là số vector. Cuối cùng nhận đầu vào K là số nhóm muốn chia. Thuật toán sẽ xuất ra K điểm trung tâm $[m_1, m_2, ..., m_K] \in \mathbf{R}^{d.K}$ của K nhóm và gắn mác(nhãn, nhóm thuộc về) cho từng điểm đầu vào.
Thuật toán có thể áp dụng cho tập hợp các dữ liệu vector nhiều chiều, vì phần mêm mô phỏng trên mặt phẳng Oxy hai chiều nên bài viết sẽ nói về hai chiều để có một cách tiếp cận dễ nhất, đối với nhiều chiều các bước làm cũng tương tự.

Tập hợp nhận vào là các vector có tọa độ (x, y) là tọa độ của chuột tại vị trí nhấp chuột.
Bước đầu chúng ta cần tạo một mảng, vector hai chiều để lưu trữ các điểm đầu vào,sau đó sẽ nhận vào vị trí của chuột và input vào mảng, vector hai chiều đó.

```
// Khai bao mang luu tru cac diem dau vao
vector<vector<double>> points;
    ...
while(window.isRunning) {
        ...

    int mouse_x, mouse_y;
    SDL_GetMouseState(&mouse_x, &mouse_y);

       ...
    if (event.type == SDL_MOUSEBUTTONDOWN) {
        if ([mouse in points space]) {
            labels.clear();
            vector<double> tmp;
            int x = mouse_x - 11; // nhan vao toa
 do x
            int y = mouse_y - 11; // nhan vao toa
 do y
            tmp.push_back(x);
            tmp.push_back(y);
            points.push_back(tmp);
            tmp.clear();
        }
    }
}
```

Listing 1. Nhận đầu vào

Bước tiếp theo chúng ta sẽ khai báo và nhận đầu vào K, và tạo một mảng, vector hai chiều rồi random ra K điểm có tọa độ (x, y) bất kỳ và nhập nó vào trong mảng, vector hai chiều vừa tạo.

```
#include <time.h> // ham random
int K = 0;
vector<vector<double>> clusters;
...
while(window.isRunning()) {
        ...

    if ([mouse in random space]) {
```

```
        clusters.clear(); // make sure = []
        labels.clear();
        for (int i = 0; i < K; i++) {
            // random 10 < r1 < 1178 - 10
            // random 10 < r2 < 598 - 10
            int r1 = 10 + rand() % (1178 - 10 + 1
 - 10);
            int r2 = 10 + rand() % (598 - 10 + 1
- 10);
            vector<double> rand;
            rand.push_back(r1);
            rand.push_back(r2);
            clusters.push_back(rand);
            rand.clear();
        }
        cerr << "Random exc" << endl;
    }
}
```

Listing 2. Random các điểm

Bước tiếp theo là chạy thuật toán. Nói một cách đơn gian chúng ta đang có $points = [p_1, p_2, ..., p_n] \in \mathbf{R}^{2.n}$ là tập hợp các điểm ta nhấp lên màn hình. $clusters = [c_1, c_2, ..., c_K] \in \mathbf{R}^{2.K}$ là số điểm mà chúng ta vừa random. Bây giờ chúng ta sẽ tính khoảng cách của từng điểm trong $points$ tới các điểm trong $clusters$ rồi lưu nó vào một list mới $distancetocl$.

$$distancetocl = [d_1, d_2, ..., d_K] \in \mathbf{R}^{2.K}$$

Sau khi có list $distancetocl$ ta sẽ tìm ra (giá trị nhỏ nhất) khoảng cách nhỏ nhất của điểm đầu (points[0]) đến từng điểm trong $clusters$ ghi nhớ vị trí (index) của nó rồi lưu vào list $labels$. Tiếp tục tính tương tự đối với các phần tử tiếp theo trong list $points$.

$$labels = [l_1, l_2, ..., l_n] \in \mathbf{R}^{1.n}$$

Mỗi giá trị trong $labels$ sẽ đánh dấu các điểm K
Múc đích của việc này là đẻ gắn nhãn cho các điểm.

Giờ hãy khai báo mảng và tạo hàm tính khoảng cách:

```
double dis(vector<double> vect1, vector<double>
    vect2) {
    return double(sqrt(pow((vect1[0] - vect2[0]),
     2) + pow((vect1[1] - vect2[1]), 2)));
}
```

Listing 3. Hàm tính khoảng cách

```
if ([mouse in run space]) {
    // change points color
    labels.clear();
    for (int i = 0; i < points.size(); i++) {
        vector<double> distance_to_cl;
        for (int j = 0; j < clusters.size(); j++)
     {
            double tmp1 = dis(points[i], clusters
    [j]);
            distance_to_cl.push_back(tmp1);
        }
            double min_dis = *min_element(
    distance_to_cl.begin(), distance_to_cl.end())
    ;
            labels.push_back(getIndex(
    distance_to_cl, min_dis));
    }
    ...
}
```

Listing 4. Chạy thuật toán

Tiếp theo chúng ta sẽ cập nhật tọa độ của từng điểm K trong $clusters$, cứ tiếp tục vòng lặp cho đến khi phân chia được kết quả.

```
...
 // change clusters pos
for (int i = 0; i < clusters.size(); i++) {
    double sum_x = 0;
    double sum_y = 0;
    double count = 0;
    for (int j = 0; j < points.size(); j++) {
        if (labels[j] == i) {
            sum_x = sum_x + points[j][0];
            sum_y = sum_y + points[j][1];
            count = count + 1;
        }
    }
    if (count != 0) {
        clusters[i][0] = sum_x/count;
        clusters[i][1] = sum_y/count;
    }
}
...
```

Listing 5. Chạy thuật toán

Tọa độ mới của từng điểm K là trung bình cộng của các điểm trong $points$ được gán vào nhóm K đó. Sau khi có vị trí K mới ta sẽ cập nhật lại cho đến khi vị trí của K được cố định.