Giải thuật DBSCAN cải tiến cho gom cụm các tập dữ liệu lớn

Nguyễn Huỳnh Anh Duy
Khoa CNTT-TT, ĐH Cần Thơ
Số 1, Lý Tự Trọng, Ninh Kiều, TP. Cần Thơ
Email: nhaduyag89@gmail.com

Trần Nguyễn Minh Thư, Đỗ Thanh Nghị
Khoa CNTT-TT, ĐH Cần Thơ
Số 1, Lý Tự Trọng, Ninh Kiều, TP. Cần Thơ
Email: dtnghi@cit.ctu.edu.vn

***Tóm tắt*—Trong bài viết này, chúng tôi trình bày giải thuật DBSCAN cải tiến cho gom cụm các tập dữ liệu lớn. DBSCAN là giải thuật gom cụm hiệu quả, cho phép phát hiện các cụm có hình dạng bất kỳ và có khả năng phát hiện nhiễu tốt. Tuy nhiên, giải thuật DBSCAN có độ phức tạp trung bình bậc hai theo số lượng phần tử, làm cho nó không hiệu quả về tốc độ xử lý các tập dữ liệu lớn. Chúng tôi đề xuất sử dụng cấu trúc chỉ mục Cover Trees, tăng tốc quá trình tìm kiếm *k* láng giềng, cải thiện tốc độ giải thuật DBSCAN. Kết quả thực nghiệm trên các tập dữ liệu cho thấy giải thuật DBSCAN cải tiến sử dụng cấu trúc chỉ mục Cover Trees chạy nhanh hơn 12 lần so với giải thuật DBSCAN gốc.**

Từ khóa: Gom cụm dữ liệu, giải thuật DBSCAN, cấu trúc chỉ mục Cover Trees.

I. GIỚI THIỆU

Gom cụm dữ liệu hay tiếp cận máy học không giám sát thực hiện việc nhóm dữ liệu (không có nhãn) sao cho các dữ liệu cùng cụm có các tính chất tương tự nhau và dữ liệu của hai nhóm khác nhau phải rất khác nhau. Dựa trên thể hiện của dữ liệu để đi tìm hiểu bản chất của dữ liệu. Bằng việc quan sát dữ liệu thu thập được (thể hiện của dữ liệu), người ta giả định rằng các các dữ liệu này tuân theo một quy luật nào đó (bản chất của dữ liệu). Kế đến xây dựng mô hình dựa trên quy luật giả định, thiết lập các phương trình, tìm các tham số, sao cho mô hình phù hợp với dữ liệu ta thu thập được nhất. Mô hình thu được có thể được dùng để hiểu dữ liệu, phát hiện tri thức mới từ dữ liệu hoặc cũng có thể dùng để giải thích dữ liệu mới có liên quan. Gom cụm dữ liệu được ứng dụng trong nhiều lĩnh vực như: mô hình túi từ trong phân lớp ảnh, gom cụm dữ liệu gen trong tin sinh học, phát hiện nhóm khách hàng trong kinh doanh, v.v. Học để khám phá các cụm dữ liệu từ các cơ sở dữ liệu lớn [5], [6], [8], trong thực tiễn luôn đặt ra các vấn đề cần quan tâm như: xác định tham số đầu vào cho giải thuật, khám phá các cụm có hình dạng tùy ý, khả năng xử lý cơ sở dữ liệu lớn. Giải thuật DBSCAN [7] là giải thuật gom cụm hiệu quả, cho phép phát hiện các cụm có hình dạng bất kỳ và có khả năng phát hiện nhiễu tốt. Trong thực hiện, giải thuật cần bước tìm kiếm *k* láng giềng, nên DBSCAN có độ phức tạp bậc hai theo số lượng phần tử. Để cải tiến độ phức tạp giải thuật, chúng tôi đề xuất sử dụng cấu trúc chỉ mục Cover Trees [3], tăng tốc quá trình tìm kiếm *k* láng giềng, cải thiện thời gian thực thi của giải thuật DBSCAN trên các tập dữ liệu lớn. Kết quả thực nghiệm trên các tập dữ liệu thực từ kho UCI [1] và các tập dữ liệu lớn cho thấy giải thuật cải tiến do chúng tôi đề xuất nhanh hơn giải thuật gốc khoảng 12 lần.

Phần tiếp theo của bài viết được trình bày như sau: phần 2 trình bày các giải thuật gom cụm phổ biến hiện nay, phần 3 trình bày giải thuật DBSCAN cải tiến, phần 4 trình bày các kết quả thực nghiệm và phần 5 là kết luận và hướng phát triển.

II. GIẢI THUẬT GOM CỤM

Công trình nghiên cứu liên quan đến giải thuật gom cụm được trình bày trong các tài liệu [2], [6], [10], [14]. Các giải thuật gom cụm thường được chia làm hai nhóm giải thuật: phân cấp và phân vùng.

Các giải thuật phân vùng tìm cách xếp *n* phần tử của tập dữ liệu *D* vào *k* nhóm (*k* là tham số đầu vào của các giải thuật). Giải thuật phân vùng sẽ bắt đầu với một vùng khởi đầu của *D* và sao đó di chuyển lặp đi lặp lại các phần tử giữa các nhóm. Mỗi cụm được biểu diễn bởi tâm của cụm (giải thuật k-means [12]) hoặc bởi nơ-ron chiến thắng (giải thuật bản đồ tự tổ chức SOM [11]) hoặc bởi biến ngẫu nhiên tiềm ẩn z_i (giải thuật cực đại hóa kỳ vọng EM [4]). Các giải thuật phân vùng thường thực hiện 2 bước. Thứ nhất, xác định *k* đại diện. Thứ hai, gán mỗi phần tử vào nhóm có đại diện gần với phần tử đang xét nhất. Với việc gán các phần tử vào nhóm như vậy thì hình dạng của các cụm tìm được bởi các giải thuật gom cụm là đa giác lồi (rất hạn chế).

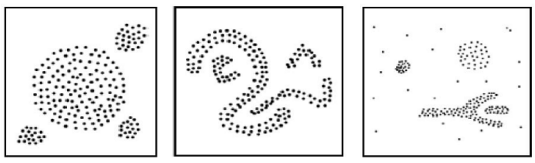
Các giải thuật phân cấp tạo một cây phân cấp cho dữ liệu *D*. Cây phân cấp lặp lại việc chia *D* thành những tập con nhỏ hơn cho đến khi mỗi tập con chứa chỉ một phần tử. Như vậy, mỗi nút của cây biểu diễn một cụm của *D*. Cây phân cấp có thể được tạo nút lá lên nút gốc (agglomerative approach) hoặc từ nút gốc xuống nút lá (divisive approach) bởi việc sáp

nhập hoặc phân chia các cụm tại mỗi bước. Ngược lại với các giải thuật phân vùng, các giải thuật phân cấp không cần tham số đầu vào *k*. Tuy nhiên, một điều kiện dừng phải được xác định để kết thúc quá trình sáp nhập hoặc phân chia. Một ví dụ cho điều kiện dừng trong agglomerative approach là khoảng cách D_{min} giữa tất cả các cụm. Vấn đề chính với các giải thuật phân cấp là việc khó xác định các tham số thích hợp cho điều kiện dừng, giá trị của D_{min} phải đủ nhỏ để phân chia được tất cả các cụm và phải đủ lớn không có cụm nào bị tách thành hai cụm nhỏ hơn.

III. GIẢI THUẬT DBSCAN CẢI TIẾN

A. Giải thuật DBSCAN

Giải thuật DBSCAN (Density Based Spatial Clustering of Application with Noise) [7] được Ester, Kriegel và Sander đề xuất năm 1996 khi nghiên cứu các thuật toán gom cụm dữ liệu không gian dựa trên định nghĩa cụm là tập tối đa các điểm liên thông về mật độ. Giải thuật DBSCAN phát hiện các cụm có hình dạng tùy ý, khả năng phát hiện nhiễu tốt. DBSCAN thực hiện tốt trên không gian nhiều chiều; thích hợp với cơ sở dữ liệu có mật độ phân bố dày đặc kể cả có phần tử nhiễu.



Dữ liệu 1 Dữ liệu 2 Dữ liệu 3

Hình 1: Dạng cụm dữ liệu được khám phá bởi giải thuật DBSCAN (nguồn [7])

Quan sát 3 tập dữ liệu điểm mẫu ở hình 1, chúng ta dễ dàng nhận ra các cụm điểm và các điểm nhiễu.

Ý tưởng chính để phát hiện ra các cụm của giải thuật DBSCAN là bên trong mỗi cụm luôn tồn tại một mật độ cao hơn bên ngoài cụm. Hơn nữa, mật độ ở những vùng nhiễu thì thấp hơn mật độ bên trong của bất kỳ cụm nào. Trong mỗi cụm phải xác định bán kính vùng lân cận (Eps) và số lượng điểm tối thiểu trong vùng lân cận của một điểm trong cụm (MinPts). Hình dạng vùng lân cận của một điểm được xác định dựa vào việc chọn hàm khoảng cách giữa hai điểm *p* và *q*, ký hiệu là *dist(p,q)*. Ví dụ, nếu dùng khoảng cách Mahattan trong không gian 2D thì hình dạng vùng lân cận là hình chữ nhật.

1) Khái niệm sử dụng trong giải thuật DBSCAN

Định nghĩa 1: (Eps-neighborhood)

Gọi *D* là tập cơ sở dữ liệu điểm. Eps-neighborhood của điểm *p*, ký hiệu là $N_{Eps}(p)$, được xác định bởi $N_{Eps}(p) = \{q \in D \mid dist(p,q) \le Eps\}$.

Định nghĩa 2: (directly density – reachable)

Một điểm $p \in D$ là directly density - reachable từ một điểm $q \in D$ khi và chỉ khi:

$p \in N_{Eps}(q)$ và

$|N_{Eps}(q)| \ge MinPts$ (điều kiện điểm lõi).

Định nghĩa 3: (density–reachable)

Một điểm *p* là density–reachable từ *q* khi và chỉ khi có một dây chuyền các điểm $p_1, ..., p_n$ với $p_1 = q$ và $p_n = p$ mà p_{i+1} là directly density – reachable từ p_i.

Hai điểm nằm trên biên của một nhóm thì không density–reachable lẫn nhau, bởi vì điều kiện xác định điểm lõi không chứa chúng. Tuy nhiên, trong nhóm sẽ có một điểm lõi mà hai điểm biên đều density–reachable.

Định nghĩa 4: (density-connected)

Một điểm *p* là density-connected đến một điểm *q* khi và chỉ khi có một điểm *o* mà cả hai điểm *p* và *q* đều density-reachable từ *o*.

Một nhóm là một tập các điểm density-connected lớn nhất. Nhiễu là điểm không thuộc nhóm nào.

Định nghĩa 5: (Nhóm)

Một nhóm *C* là một tập con không rỗng của *D* thỏa các điều kiệu sau:

$\forall p, q$: nếu $p \in C$ và *q* là density-reachable từ *p* thì $q \in C$.

$\forall p, q \in C$: *p* là density-connected từ *q*.

Định nghĩa 6: (Nhiễu)

Gọi $C_1, \ldots, C_k$ là các nhóm của tập dữ liệu *D*. Nhiễu là tập hợp tất cả các điểm không thuộc bất kỳ nhóm C_i nào với $i=1, ..., k$.

Sau đây là một số bổ đề quan trọng trong thuật toán DBSCAN. Một nhóm có thể được tìm thấy thông qua 2 bước. Bước thứ nhất, chọn một điểm bất kỳ từ *D* thỏa điều kiện điểm lõi làm hạt giống. Bước thứ 2, tìm tất cả các điểm density-reachable từ điểm hạt giống.

Bổ đề 1: Gọi *p* là điểm thuộc *D* và $|N_{Eps}(p)| \ge MinPts$ thì tập $O = \{o \mid o \in D$ và *o density-reachable* từ $p\}$ là một nhóm.

Bổ đề 2: Gọi C là một nhóm tìm được thông qua hai tham số *Eps* và *MinPts* và p là một điểm bất kỳ trong C với $|N_{Eps}(p)| \geq MinPts$ thì C bằng tập $O = \{o \mid o$ là *density-reachable* từ $p\}$.

2) Giải thuật DBSCAN

Giải thuật DBSCAN được phát triển để phát hiện các cụm và nhiễu dựa trên hai định nghĩa 5 và 6. Giải thuật cần phải biết trước hai tham số *Eps* và *MinPts*. Để tìm một nhóm, DBSCAN bắt đầu với một điểm p và tìm tất cả các điểm density-reachable từ p. Nếu p là một điểm lõi thì giải thuật hình thành một cụm (theo bổ đề 2). Nếu p là một điểm biên thì không có điểm nào density-reachable từ p và DBSCAN đi đến điểm kế tiếp của tập dữ liệu.

Bảng 1: Giải thuật DBSCAN

Đầu vào:

Tập dữ liệu gồm m phần tử: $X = \{x_i\}_{i=1,m}$ với $x_i \in R^n$

Bán kính vùng lân cận: eps

Số lượng điểm tối thiểu: minPts

Giải thuật:

```
//Khởi tạo
ClusterId = 1
for i = 1 to m do
    if (x_i chưa được duyệt qua)
        N = regionQuery(x_i, X, eps)
        if (|N| < minPts) //x_i không là điểm lõi
            changeCluster(x_i, 0) //Gán x_i vào nhiễu
        else
            changeCluster(N, ClusterId)
            N = N \ {x_i}
            while (N != rỗng)
                N' = regionQuery(p, X, eps) // p ∈ N
                if (|N'| >= minPts)
                    for j= 1 to sizeof(N') do
                        q = N'.get(j)
                        if (q chưa được duyệt hoặc là noise)
                            if (q chưa được duyệt)
                                N = N ∪ {q}
                            changeCluster(q, ClusterId)
                N = N \ {p}
            ClusterId++
```

Giải thuật DBSCAN chi tiết được nêu trong bảng 1. Hàm *regionQuery(x_i, X, eps)* được gọi trong giải thuật DBSCAN trả về các điểm láng giềng trong bán kính *eps* của điểm x_i. Hàm này cần duyệt qua toàn tập dữ liệu để tìm k láng giềng, điều này làm hạn chế tốc độ xử lý của DBSCAN. Do đó, độ phức tạp thời gian của giải thuật DBSCAN gốc là $O(n^2)$ với n là số lượng phần tử của tập dữ liệu. Để cải thiện tốc độ tìm kiếm k láng giềng (không cần duyệt hết dữ liệu), chúng tôi đề xuất sử dụng cấu trúc cây chỉ mục nhiều chiều. Các nghiên cứu thường dựa vào cấu trúc chỉ mục phổ biến kd-trees [9]. Tuy nhiên, cấu trúc kd-trees chỉ thực hiện tốt khi số chiều dữ liệu nhỏ hơn 20 do độ phức tạp của kd-trees tăng theo hàm mũ của số chiều [13]. Cấu trúc kd-trees không hiệu quả cho tìm kiếm k láng giềng khi số chiều dữ liệu lớn. Vì vậy, thay vì sử dụng cấu trúc kd-trees, chúng tôi đề xuất sử dụng cấu trúc Cover Trees [3] (có thể xử lý khi số chiều dữ liệu lớn) để tăng tốc quá trình tìm kiếm k láng giềng cho giải thuật DBSCAN. Điều này cải thiện tốc độ xử lý của giải thuật DBSCAN. Phần tiếp theo, chúng tôi trình bày về cấu trúc Cover Trees, giải thuật tạo một Cover Tree và giải thuật tìm kiếm trên một Cover Tree hỗ trợ tìm kiếm k láng giềng trong giải thuật DBSCAN.

B. Cover Trees

1) Cấu trúc Cover Trees

Một cây cover T trên tập dữ liệu S là một cây phân cấp (leveled) mà tại mỗi cấp là một "cover" cho cấp dưới nó. Mỗi cấp được đánh chỉ mục bởi số nguyên i giảm dần. Mỗi nút trên cây liên kết với một điểm trong S. Mỗi điểm trong S có thể liên kết với nhiều nút trên cây; tuy nhiên, mỗi điểm chỉ được xuất hiện một lần trên mỗi cấp. Gọi C_i là tập điểm trong S liên kết với các nút ở mức i. Cover tree tuân theo những quy luật sau cho tất cả các mức i:

- (Nesting) $C_i \subset C_{i-1}$. Điều này có nghĩa là một lần một điểm $p \in S$ xuất hiện trong C_i thì ở các mức dưới mức i trên cây đều có một nút liên kết với p.
- (Covering tree) Với mọi điểm $p \in C_{i-1}$, luôn tồn tại một điểm $q \in C_i$ sao cho $d(p,q) \leq 2^i$ và nút ở mức i liên kết với q là cha của nút ở mức i - 1 liên kết với p.
- (Separation) Với mọi điểm $p, q \in C_i$ thì $d(p,q) > 2^i$.

Ghi chú quan trọng: Khi chúng ta xác định các nút liên kết với các điểm cần chú ý là một điểm chỉ có thể xuất hiện một lần ở mỗi mức.

Khái niệm đơn giản nhất để mô tả các giải thuật trong giới hạn của sự biểu diễn ẩn của Cover Tree chứa một lượng vô hạn các mức là C_∞ chứa một điểm trong S liên kết với nút gốc và $C_{-\infty} = S$. Tuy nhiên, chúng ta phải sử dụng và phân tích sự biểu diễn một cách tường minh chỉ mất không gian $O(n)$. Nếu một điểm $p \in S$ xuất hiện lần đầu ở mức i thì nó sẽ xuất hiện ở tất cả các mức dưới mức i, và p sẽ là con của chính nó trong tất cả các mức này (nút liên kết với p là con của nút liên kết với p ở mức trên). Sự biểu diễn tường minh của cây hợp nhất tất cả các nút mà con của nó chỉ là nó. Nghĩa là mỗi nút hoặc có cha khác với chính nó hoặc là có con khác với chính nó.

2) Tìm kiếm láng giềng gần nhất

Để tìm láng giềng gần nhất (xem bảng 2) của một điểm p trong một Cover Tree, giải thuật đi xuống theo từng mức, theo dõi tập con $Q_i \subset C_i$ của các nút mà có thể chứa láng giềng gần nhất của p như một hậu duệ. Giải thuật lặp lại công việc xây dựng Q_{i-1} bằng cách mở rộng Q_i cho con của nó trong C_{i-1}, sau đó loại bỏ các con q mà không thể dẫn đến láng giềng gần nhất của p. Sẽ dễ dàng hơn để hình dung cây khi có một số lượng vô hạn các mức (với C_∞ chỉ chứa gốc, và với $C_{-\infty} = S$). Biểu diễn tập con của nút p là Children(p) và gọi $d(p,Q) = \min_{q \in Q} d(p,q)$ là khoảng cách từ p đến điểm gần nhất q trong tập Q.

Bảng 2: Giải thuật tìm láng giềng gần nhất

Find-Nearest (cover tree T, query point p)
(1) set $Q_\infty = C_\infty$.
(2) for i from ∞ down to $-\infty$
(a) consider the set of children of Q_i:
$Q = \{ \text{Children}(q) : q \in Q_i \}$.
(b) form next cover set:
$Q_{i-1} = \{q \in Q: d(p,q) \le d(p,Q) + 2^i\}$
(3) return arg $\min_{q \in Q-\infty} d(p, q)$.

Nếu T là một Cover Tree trên S thì Find-Nearest(T,p) trả về láng giềng gần nhất của p trong S. Độ phức tạp thời gian thực hiện giải thuật tìm kiếm láng giềng gần nhất là $c^{12}logn$.

Để tìm kiếm k láng giềng trong bán kính eps thì ở bước 2(b) thay thế eps cho $d(p,Q)$ và ở bước (3) dùng giải thuật vét cạn để tìm các điểm thỏa điều kiện trong tập Q.

3) Xây dựng cover tree cho một tập dữ liệu

Giải thuật xây dựng cover tree (xem bảng 3) cho một tập dữ liệu sử dụng phương pháp đệ quy. Tại mỗi bước, giải thuật có một điểm p, một tập điểm *NEAR* gồm những điểm phải được thêm vào dưới p, một tập điểm *FAR* gồm những điểm có thể được thêm vào dưới p. Đầu tiên giải thuật tìm kiếm các tập *NEAR* và *FAR* cho chính nó (như là các con) và sau đó làm tương tự cho những phần tử còn lại của *NEAR*. Giải thuật trả về một nút được tạo và những điểm chưa được sử dụng của *FAR*. Độ phức tạp của giải thuật xây dựng một cover tree cho tập dữ liệu là $O(c^6nlogn)$.

C. Tích hợp cấu trúc cây chỉ mục Cover Trees vào giải thuật DBSCAN

Quay trở lại giải thuật DBSCAN gốc trình bày trong bảng 1. Hàm regionQuery(p, X, eps) thực hiện tìm kiếm k láng giềng bán kính eps của điểm p trong tập dữ liệu X. Do hàm này phải duyệt qua toàn bộ dữ liệu để tìm kiếm k láng giềng trong giải thuật DBSCAN gốc nên độ phức tạp là $O(n^2)$.

Bảng 3: Giải thuật xây dựng Cover Tree

Construct(point p,point sets <NEAR,FAR>, level i)
(1) if NEAR = $\emptyset$
(2) then return <p, FAR>
(3) else
(a)<SELF,NEAR>=**Construct**(p,SPLIT(d(p,·), 2^{i-1}, NEAR), i - 1)
(b) add SELF to Children(p_i)
(c) while NEAR $\neq \emptyset$
(i) pick q in NEAR
(ii)<CHILD,UNUSED>=Construct(q ,SPLIT(d(q,·), 2^{i-1}, NEAR, FAR, i-1)
(iii) add CHILD to Children(p_i)
(iv)let <NEW-NEAR, NEW-FAR> = SPLIT(d(p,), 2^i, UNUSED)
(v) add NEW-FAR to FAR, and NEW-NEAR to NEAR.
(d) return <p_i, FAR>

Để cải tiến thời gian chạy giải thuật DBSCAN, chúng tôi đề xuất xây dựng cây chỉ mục Cover Trees cho tập dữ liệu X (xem bảng 4) và cần thay thế hàm regionQuery(p, X, eps) bằng hàm hàm tìm kiếm k láng giềng trong bán kính eps trên Cover Tree (hàm Find-Nearest **(**T, p, eps**)** trình bày trong bảng 2). Do độ phức tạp thời gian trung bình của giải thuật tìm kiếm k láng giềng trên Cover Tree là $O(c^{12}*logn)$ nên độ phức tạp thời gian trung bình của giải thuật DBSCAN cải tiến (giải thuật DBSCAN có kết hợp Cover Trees) là $O(c^{12}n*logn)$. Vì vậy, giải thuật DBSCAN cải tiến có độ phức tạp nhỏ hơn giải thuật DBSCAN gốc khi n lớn (tập dữ liệu lớn).

IV. KẾT QUẢ THỰC NGHIỆM

Chúng tôi tiến hành đánh giá hiệu quả của giải thuật DBSCAN cải tiến được đề xuất để gom cụm trên các tập dữ liệu lớn. Trước tiên, chúng tôi đã cài đặt giải thuật DBSCAN gốc và DBSCAN cải tiến bằng ngôn ngữ lập trình C/C++ có sử dụng thư viện Cover Trees [3].

Bảng 4: Mô tả các tập dữ liệu nhóm 1

Tên tập dữ liệu	Số thuộc tính	Số mẫu
Iris	4	150
Wine	13	178
Segment	19	2 310
Optdigits	64	3 823
Sat Image	36	4 435
Shuttle	9	43 500

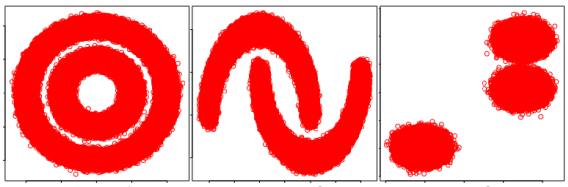
Các tập dữ liệu sử dụng trong thực nghiệm chia thành 2 nhóm. Nhóm các tập dữ liệu thực gồm 6 tập dữ liệu đa chiều (từ 4 đến 64 chiều) với số mẫu từ 150 đến 43500 mẫu được lấy từ website UCI

Machine Learning Repository [1], được mô tả trong bảng 4. Chúng tôi đã loại bỏ đi nhãn của các tập dữ liệu trước khi chạy thực nghiệm.

Bảng 5: Mô tả các tập dữ liệu nhóm 2

Tên tập dữ liệu	Số thuộc tính	Số mẫu
Data 1	2	1.000.000
Data 2	2	1.000.000
Data 3	2	1.000.000

Nhóm thứ hai gồm 3 tập dữ liệu 2 chiều với một triệu mẫu tin không có nhãn (xem bảng 5) được tạo từ thư viện chương trình scikit-learn [15]. Phân bố dữ liệu của nhóm thứ hai được cho trong hình 2.



Data 1 Data 2 Data 3

Hình 2: Phân bố dữ liệu của tập dữ liệu nhóm 2

Chúng tôi lần lượt thực hiện hai giải thuật DBSCAN và DBSCAN cải tiến trên hai nhóm dữ liệu, đo thời gian thực hiện của hai giải thuật này và so sánh kết quả với nhau. Tất cả các kết quả đều được thực hiện trên máy tính cá nhân (intel 2.1GHz x 2, 2GB RAM) chạy hệ điều hành Linux (bản phân phối hệ điều hành là Ubuntu 12.04 LTS 32 bit).

A. Kết quả chạy thực nghiệm trên các tập dữ liệu nhóm 1

Bảng 6: Bộ tham số Eps và MinPts cho các tập dữ liệu nhóm 1

Tên tập dữ liệu	Eps	MinPts
Iris	0.42	5
Wine	67	5
Segment	19	5
Optdigits (train)	19.8	6
Sat Image (train)	26	6
Shuttle (train)	9	8

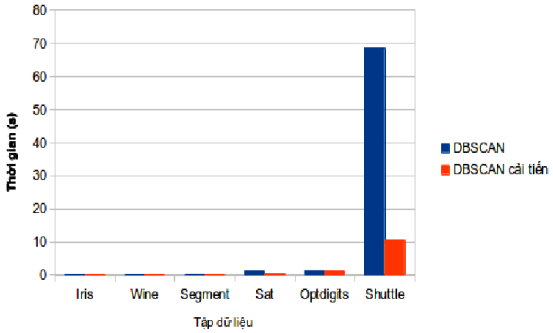
Bảng 7: Thời gian thực thi gom cụm trên các tập dữ liệu nhóm 1 (đơn vị tính bằng giây)

Tên tập dữ liệu	DBSCAN	DBSCAN cải tiến
Iris	0.001462	0.001245
Wine	0.001116	0.001429
Segment	0.220723	0.063479
Optdigits	1.272079	1.046573
Sat Image	1.098793	0.370765
Shuttle	68.7473	10.537758

Chúng tôi áp dụng các bộ tham số (*Eps* và *MinPts*) thu được kết quả tốt cho gom cụm như trong bảng 6 để chạy thực nghiệm trên các tập dữ liệu thuộc nhóm 1.

Kết quả thời gian chạy thực nghiệm (được tính bằng giây) của giải thuật DBSCAN và DBSCAN cải tiến trên các tập dữ liệu thuộc nhóm 1 được trình bày trong bảng 7, hình 3.

Quan sát biểu đồ thời gian thực thi của 2 giải thuật trên hình 3, chúng ta có thể thấy được với các tập dữ liệu Iris, Wine, Segment, Optdigits thì sự chênh lệch là không nhiều do các tập dữ liệu này có số lượng mẫu nhỏ. Với tập dữ liệu lớn hơn như Sat Image thì giải thuật DBSCAN cải tiến chạy nhanh hơn giải thuật DBSCAN nhưng cũng chưa đáng kể. Riêng với tập dữ liệu lớn nhất là Shuttle thì giải thuật DBSCAN cải tiến chạy nhanh hơn khoảng 6 lần so với giải thuật gốc. Điều này phù hợp với mục tiêu cải tiến tốc độ của giải thuật DBSCAN trên tập dữ liệu lớn.



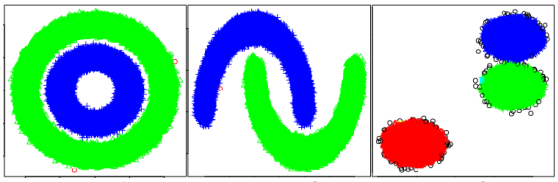
Hình 3: So sánh thời gian thực thi trên các tập dữ liệu nhóm 1

B. Kết quả chạy thực nghiệm trên các tập dữ liệu thuộc nhóm 2

Phần tiếp theo chúng tôi sẽ chạy thực nghiệm trên 3 tập dữ liệu có số mẫu lớn đến một triệu mẫu. Chúng tôi áp dụng các bộ tham số (*Eps* và *MinPts*) nêu trong bảng 8 để chạy thực nghiệm trên các tập dữ liệu. Hình 4 trình bày kết quả gom cụm, hình 5, bảng 9, trình bày thời gian thực thi của các giải thuật.

Bảng 8: Bộ tham số Eps và MinPts cho các tập dữ liệu nhóm 2

Tên tập dữ liệu	Eps	MinPts
Data 1	0.06	4
Data 2	0.06	4
Data 3	0.27	4



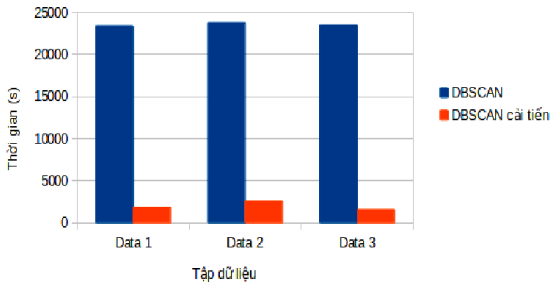
Data 1 Data 2 Data 3

Hình 4: Hiển thị kết quả gom cụm trên ba tập dữ liệu nhóm 2 của DBSCAN

Hình 4 hiển thị kết quả thu được từ gom cụm trên 3 tập dữ liệu nhóm 2. Giải thuật DBSCAN tách tập dữ liệu **Data 1** thành 2 cụm, **Data 2** thành 2 cụm và **Data 3** thành 3 cụm (với điểm hình tròn là nhiễu). Kết quả từ hình 4 cho thấy giải thuật DBSCAN thực hiện gom cụm tốt trên 3 tập dữ liệu nhóm 2.

Bảng 9: Thời gian thực thi gom cụm trên các tập dữ liệu nhóm 2 (đơn vị tính bằng giây)

Tên tập dữ liệu	DBSCAN	DBSCAN cải tiến
Data 1	23360.041639	1825.989121
Data 2	23758.916738	2604.849390
Data 3	23473.789459	1596.198543



Hình 5: So sánh thời gian thực thi trên các tập dữ liệu nhóm 2

Quan sát biểu đồ trong hình 5, có thể thấy rằng giải thuật DBSCAN cải tiến chạy nhanh hơn khoảng 12 lần so với giải thuật gốc trên 3 tập dữ liệu lớn (nhóm 2). Kết quả thu được từ việc chạy thực nghiệm lần nữa cho thấy giải thuật DBSCAN cải tiến do chúng tôi đề xuất cải thiện được thời gian gom cụm của giải thuật gốc trên các tập dữ liệu lớn.

V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chúng tôi vừa trình bày giải thuật DBSCAN cải tiến cho phép gom cụm nhanh các tập dữ liệu lớn. Giải thuật đề xuất sử dụng cấu trúc chỉ mục Cover Trees, tăng tốc quá trình tìm kiếm *k* láng giềng, cải thiện tốc độ giải thuật DBSCAN bằng cách giảm độ phức tạp của giải thuật gốc. Kết quả thực nghiệm trên các tập dữ liệu cho thấy giải thuật DBSCAN cải tiến nhanh hơn giải thuật gốc khi gom cụm các tập dữ liệu lớn.

Trong tương lai, chúng tôi phát triển chiến lược xác định tự động tham số *Eps* và *MinPts* của giải thuật DBSCAN, nâng cao khả năng ứng dụng giải thuật trong thực tiễn.

TÀI LIỆU THAM KHẢO

[1] Asuncion, A. & Newman, D.J.: UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science, 2007. [http://www.ics.uci.edu/~mlearn/-MLRepository.html]

[2] P. Berkhin: A Survey of Clustering Data Mining Techniques. *Grouping Multidimensional Data* 2007: 25-71.

[3] A. Beygelzimer, S. Kakade, and J. Langford. Cover Trees for Nearest Neighbor. In Proc. International Conference on Machine Learning (ICML), 2006.

[4] Dempster A.P., Laird N.M and Rubin D.: Maximum Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B 39(1): 1-38, 1977.

[5] Đỗ T.N. and Lê T.V.: *Các hệ tri thức và khai thác dữ liệu*. NXB Đại học Cần Thơ, 2012.

[6] Đỗ T.N. and Phạm N.K.: *Nguyên lý máy học*. NXB Đại học Cần thơ, 2012.

[7] M. Ester, H.P. Kriegel, J. Sander and X. Xu.: A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press., 1996, pp. 226-231.

[8] Fayyad, U., Piatetsky-Shapiro, G., and Uthurusamy, R.: Summary from the KDD-03 Panel – Data Mining: The Next 10 Years. in *SIGKDD Explorations* 5(2):191-196, 2004.

[9] J. Friedman, J. Bentley, and R. Finkel.: An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3): 209–226, 1977.

[10] Kaufman L. and Rousseeuw P.J.: *Finding group in data. An Introduction to cluster analysis*. Wiley Interscience, 2005.

[11] Kohonen T.: *Self-Organization and Associative Memory*. Springer-Verlag, 1984.

[12] MacQueen J.: Some methods for classification and analysis of multivariate observations. In proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, *University of California Press*, vol. 1, 1967, p. 281-297.

[13] Mount D.: Data Structures. Department of Computer Science, University of Maryland, 2001.

[14] Wu X. and Kumar V.: *Top 10 Algorithms in Data Mining*. Chapman & Hall/CRC, 2009.

[15] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research Vol(12): 2825-2830, 2011.