BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

TÌM HIỂU PHÂN CỤM BIRCH

Giảng viên hướng dẫn: TS. Nguyễn Đức Thuần

Sinh viên thực hiện: Nguyễn Sơn Tùng

Mã số sinh viên : 60137435

LỜI CAM ĐOAN

Tôi xin cam đoan rằng đề tài tốt nghiệp "Tìm hiểu về thuật toán phân cụm BIRCH" là đồ án nghiên cứu của riêng cá nhân tôi. Đồ án đã được thực hiện và hoàn thành dựa vào những tìm hiểu và có sự hướng dẫn vô cùng nhiệt tình và chuyên sâu của GVHD TS.Nguyễn Đức Thuần. Tôi xin chịu hoàn toàn trách nhiệm đối với đồ án nghiên cứu riêng của mình.

Khánh Hòa, ngày…tháng…năm 2022

Tác giả chuyên đề

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên, tôi xin chân thành cảm ơn các quý thầy cô Khoa Công nghệ Thông Tin Trường Đại học Nha Trang đã tạo điều kiện và cơ hội để có thể tiếp nhận đề tài này một cách dễ dàng và các thầy cô cũng nhiệt tình giúp đỡ giải đáp các thắc mắc trong quá trình tìm hiểu và cách thức tiếp nhận đề tài.

Bên cạnh đó, khoảng thời gian tìm hiểu và nghiên cứu về thuật toán phân cụm BIRCH thì em cũng nhận được sự hỗ trợ nhiệt tình và những sự hướng dẫn vô cùng bổ ích của thầy Nguyễn Đức Thuần, nhờ thầy mọi thắc mắc về thuật toán được giải đáp cũng cách vô cùng rõ ràng và dễ hiểu, từ đó mà quá trình nghiên cứu và triển khai đề tài luôn đi đúng tiến độ đã đề ra.

Nội dung trong đồ án này là một kiến thức khá mới đối với em nên có thể có sẽ có một vài thiếu sót. Vì thế em mong nhận được các ý kiến đánh giá, nhận xét từ các quý thầy cô.

Khánh Hòa, ngày…tháng…năm 2022

Tác giả chuyên đề

(Ký và ghi rõ họ tên)

MỤC LỤC

MỞ ĐẦU

Với tốc độ phát triển một cách nhanh chóng của Công nghệ thông tin dẫn đến sự bùng nổ dữ liệu. Các doanh nghiệp và tổ chức chú trọng hơn về việc xử lý và khai thác dữ liệu một cách hiệu quả, chính vì vậy mà khai phá dữ liệu và phát hiện tri thức của dữ liệu là một nhu cầu thiết yếu. Quá trình khai thác dữ liệu cho phép doanh nghiệp thu thập nhiều thông tin hữu ích, giúp thúc đẩy sự phát triển của doanh nghiệp và cắt giảm chi phí.

Một trong những kỹ thuật trong khai phá dữ liệu và phát hiện tri thức là phân cụm dữ liệu. Phân cụm dữ liệu là quá trình gom nhóm các đối tượng có độ tương tự với nhau. Đem lại cái nhìn trực quan hơn về ý nghĩa của dữ liệu. Nội dung chính của đồ án này là giới thiệu với thuật toán phân cụm **BIRCH**, những lý thuyết liên quan đến thuật toán phân cụm **BIRCH**, các công thức, cách triển khai và giải quyết một bài toán phân cụm với thuật toán phân cụm **BIRCH**. Cuối cùng vận dụng các kiến thức trên để tạo ra một ứng dụng minh họa cơ bản cho thuật toán.

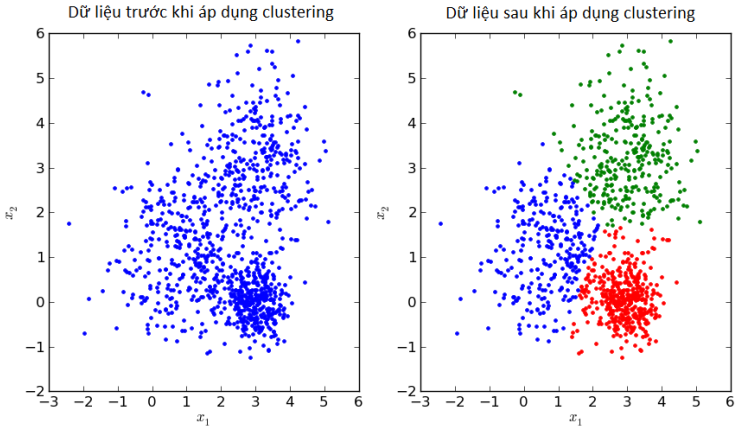
CHƯƠNG I. TỔNG QUAN VỀ THUẬT TOÁN PHÂN CỤM

I.1. Khái niệm về khai phá dữ liệu

Khai phá dữ liệu (Data mining) là công cụ phát hiện ra "tri thức" của dữ liệu, tri thức là những luật và nguyên tắc của dữ liệu. Cơ sở dữ liệu không chỉ đơn thuần là các con số, kí tự được lưu trữ trong một hệ thống máy tính, mà ở đó chúng còn đem lại cho chúng ta những "tri thức" [1].

I.2. Khái niệm về phân cụm

Phân cụm dữ liệu (Data Clustering) là hình thức học không giám sát (Unsupervised learning) trong đó các mẫu học chưa được gán nhãn [1]. Mục đích của phân cụm là gom nhóm các đối tượng có độ tương tự nhau (theo một tiêu chuẩn đánh giá) thành các cụm tương ứng. Các đối tượng dữ liệu trong cùng một cụm có độ tương tự cao hơn đối với các đối tượng dữ liệu tại các cụm khác.



Hình 1. Hình ảnh minh họa phân cụm

I.3. Bài toán phân cụm dữ liệu

Bài toán phân cụm là một nhánh ứng dụng chính của lĩnh vực Học không giám sát (Unsupervised Learning), tại đó các dữ liệu trong bài toán không được gán nhãn. Thuật toán phân cụm sẽ chia dữ liệu thành từng cụm có các đặc điểm tương tự nhau và cũng đồng thời đặc điểm đó phải khác biệt so với các cụm còn lại càng nhiều càng tốt.

Cho một tập dữ liệu $X = \{x_1, x_2, x_3 \dots x_n\}$ với *n* phần tử. Bây giờ bài toán phân cụm cần chia *X* thành *k* cụm với mỗi phần tử chỉ thuộc về một cụm duy nhất và các phần tử trong cụm phải có độ tương tự cao hơn với các phần tử trong cụm khác. Số lượng *k* cụm có thể được xác định bằng cách cho trước giá trị *k* tùy ý hoặc nhờ vào một phương pháp toán học để xác định *k*.

I.4. Các yêu cầu và ứng dụng của phân cụm dữ liệu

I.4.1. Yêu cầu đối với thuật toán phân cụm dữ liệu

- Khả năng mở rộng: thuật toán phân cụm không chỉ cho tập DL nhỏ đôi khi còn cho tập DL lớn.

- Khả năng xử lý các loại thuộc tính khác nhau: thuật toán phải phân cụm được các tập dữ liệu với các kiểu dữ liệu khác nhau như nhị phân, phân loại (nominal), thứ tự (ordinal data) hoặc là các kiểu dữ liệu có kiểu tổng hợp.

- Phát hiện các cụm có hình dạng tùy ý: do các độ đo khoảng cách khác nhau thì có thể dẫn đến phân cụm có hình dạng khác nhau.

- Khả năng đối phó với dữ liệu nhiễu: dữ liệu nhiễu là không thể tránh khỏi trong tập dữ liệu và thuật toán cần xử lý chúng để đạt kết quả mong đợi.

- Không nhạy cảm với các thứ tự của các bản ghi đầu vào: thứ tự đầu vào khác nhau dẫn đến các kết quả khác nhau, vì thế cần xử lý dữ liệu để kết quả trở nên tin cậy hơn.

- Khả năng xử lý nhiều chiều: Dữ liệu để phân cụm có thể có nhiều chiều, thuật toán phân cụm phải thoải mãn xử lý dữ liệu nhiều chiều.

I.4.2. Ứng dụng của thuật toán phân cụm

Thuật toán phân cụm được sử dụng rộng rãi trong nhiều lĩnh vực như trong kinh tế, phân tích dữ liệu và xủ lý hình ảnh. Một số ứng dụng của phân cụm như sau:

- Phân cụm giúp các nhà tiếp thị khám phá các nhóm khách hàng tiềm năng.

- Ở lĩnh vực sinh học, phân cụm có thể được dùng để xác định đơn vị phân loại thực vật và động vật, phân loại gen có chức năng tương tự.

- Phân cụm cũng được sử dụng trong các ứng dụng phát hiện gian lận thẻ tín dụng.

I.5. Kỹ thuật giải quyết các bài toán phân cụm

I.5.1. Kỹ thuật phân cụm phân hoạch

Mục đích của kỹ thuật này là gom dữ liệu thành các cụm rời nhau.

Bài toán: Giả sử ta có một tập hợp dữ liệu có N phần tử và mong muốn phân thành K cụm trong đó $K \leq N$ và K được quyết định bởi hoặc chỉ định bởi người thực hiện phân cụm. Với điều kiện các cụm phải thỏa mãn:

- Mỗi cụm phải chứa ít nhất một phần tử

- Mỗi phần tử chỉ thuộc về một cụm duy nhất

Khuyết điểm của kỹ thuật này là đồ phức tạp rất lớn, để khắc phục điều này thì người thực hiện sẽ sử dụng heuristic là chiến lược tham ăn (Greedy) và hàm tiêu chuẩn để đánh giá chất lượng của các cụm. Kỹ thuật này sẽ không hiệu quả nếu như các cụm có hình dạng phức tạp hoặc mật độ phân bố của các điểm dữ liệu tại một cụm dầy đặc.

Hai thuật toán phổ biến trong kỹ thuật này là K-Mean và K-Medoids

I.5.2. Kỹ thuật phân cụm phân cấp

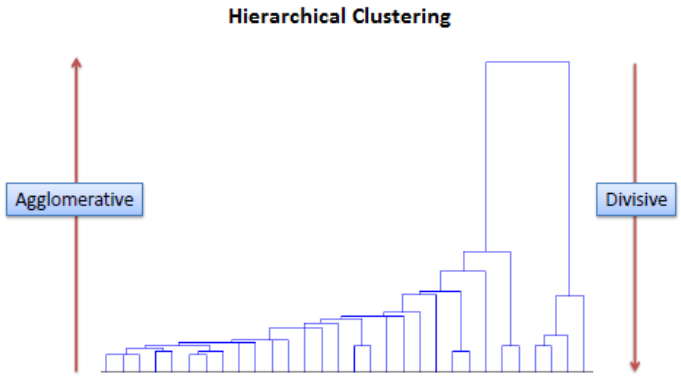
Kỹ thuật phân cụm phân cấp này sẽ xây dựng một phân cấp trên tập dữ liệu. Các dữ liệu đang xét duyệt sẽ được bố trí đưa vào vị trí hoặc cấp bậc hợp lí trên một mô hình phân cấp cụ thể là dạng cây, cây phân cụm phân cấp này được xây dựng theo kỹ thuật đệ quy. Có hai hướng tiếp cận phổ biến trên kỹ thuật phân cụm này là:

- Kết hợp hay còn được gọi là Bottom-Up

- Phân chia hay còn được gọi là Top-Down

Ưu điểm của kỹ thuật phân cụm phân cấp là dễ hiểu, dễ thực hiện.

Khuyết điểm của kỹ thuật phân cụm phân cấp là kết quả phụ thuộc vào thứ tự duyệt của các phần tử trong tập dữ liệu muốn phân cụm, phụ thuộc vào việc lựa chọn độ đo tương tự, ngoài ra tốc độ cũng khá chậm so với các kỹ thuật khác, bởi vì kỹ thuật này tiến hành duyệt bộ dữ liệu nhiều lần.



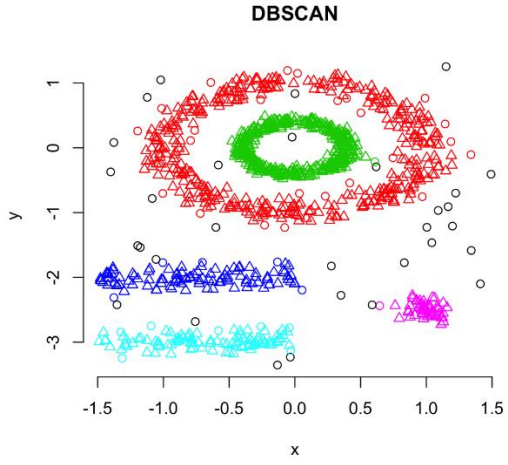
Hình 2. Kỹ thuật phân cụm phân cấp

I.5.3. Kỹ thuật phân cụm dựa trên mật độ

Kỹ thuật phân cụm này là sẽ gom nhóm các dữ liệu trên hàm mật độ. Ý tưởng của của thuật toán là tiếp tục phát triển cụm cho trước với điều kiện là mật độ lân cận không vượt ngưỡng quy định trước đó. Kỹ thuật này dựa trên mật độ của các đối tượng để xác định các cụm dữ liệu có thể phát hiện ra các cụm dữ liệu với hình thù phức tạp.

Kỹ thuật này phân cụm khắc phục được các phần tử ngoại lai hoặc các giá trị nhiễu rất tốt. Tuy nhiên việc xác định các tham số mật độ của thuật toán là rất khó, trong khi kỹ thuật này phụ thuộc vào các tham số này rất nhiều, ảnh hưởng rõ nét đến kết quả.

DBSCAN, OPTICS DENCLUE, CLIQUE là những thuật toán phân cụm điển hình áp dụng kỹ thuật này.



Hình 3. Thuật toán DBSCAN

Tóm lại, các kỹ thuật phân cụm được đề cập ở trên là các kỹ thuật đang được sử dụng rộng rãi trên thực tế để giải quyết một bài phân cụm, bên cạnh những kỹ thuật đã được đề cập bên trên, có thể còn có các kỹ thuật khác hoặc thậm chí là các phiên bản cải tiến của các kỹ thuật trên. Suy cho cùng mỗi phương pháp sẽ có những ưu điểm và khuyết riêng, vì thế dựa vào mục đích, yêu cầu của việc phân cụm mà chúng ta lựa chọn một kỹ thuật thật hợp lý.

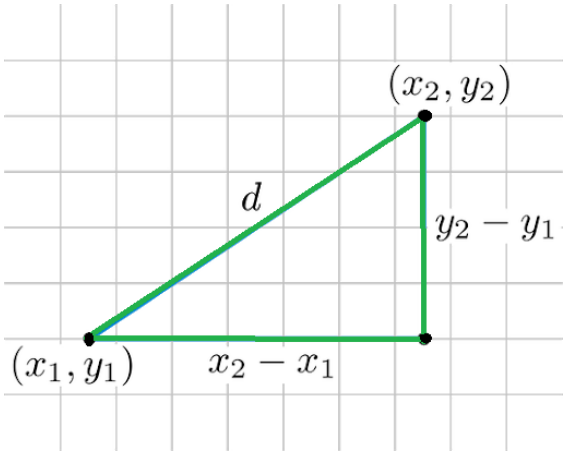
I.6. Các độ đo khoảng cách trong khai phá dữ liệu

Trong khai phá dữ liệu (Data mining) ta thường dùng các độ đo tương tự để đánh giá mức độ tương tự (Similarity) của hai đối tượng. Điều có còn có nghĩa là nếu khoảng cách giữa hai đối tượng dữ liệu càng nhỏ thì mức độ tương tự càng cao và ngược lại. Hầu hết các kỹ thuật phân cụm sử dụng độ đo khoảng cách để đánh giá sự giống nhau hoặc khác nhau giữa một cặp đối tượng, các độ đo phổ biến được sử dụng phổ biến nhất được đề cập bên dưới.

I.6.1. Độ đo Euclidean

Cho hai điểm $x = (x_1, x_2, x_3, \dots x_n)$ và $y = (y_1, y_2, y_3, \dots y_n)$. Khoảng cách giữa hai điểm x và y được xác định theo độ đo Euclidean với công thức:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

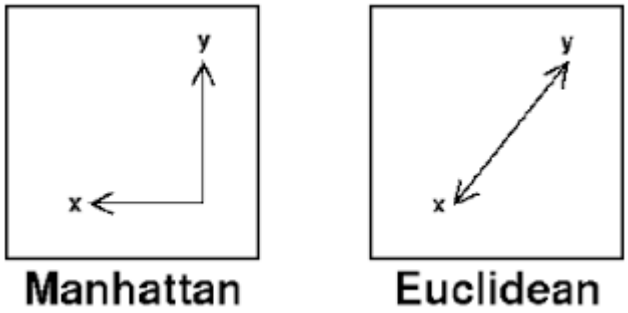


Hình 4. Minh họa về độ đo Euclidean

I.6.2. Độ đo Manhattan

Cho hai điểm $x = (x_1, x_2, x_3, \dots x_n)$ và $y = (y_1, y_2, y_3, \dots y_n)$. Khoảng cách giữa hai điểm x và y được xác định theo độ đo Manhattan với công thức:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$



Hình 5. Sự khác biệt giữa Manhattan và Euclidean

I.6.3. Độ đo Minkowski

Đây là dạng tổng quát của phép đo Euclid và Manhattan. Trong không gian n chiều, hai điểm được biểu diễn dưới dạng $x = (x_1, x_2, x_3, \dots x_n)$ và $y = (y_1, y_2, y_3, \dots y_n)$. Khoảng cách giữa hai điểm x và y được tính theo công thức

$$d(x, y) = \sqrt[p]{(x_1 - y_1)^p + (x_2 - y_2)^p + \cdots + (x_n - y_n)^p}$$

Trong đó với $p = 2$ Minkowski chính là độ đo Euclidean

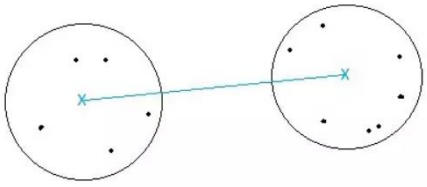
Với $p = 1$ Minkowski chính là độ đo Manhattan.

I.7. Các tiêu chí chọn kết hợp cụm

Trong cách tiếp cận kết hợp (Bottom-up) dựa vào khoảng cách giữa các cụm, có bốn tiêu chí kết hợp khác nhau được trình bày bên dưới đây.

I.7.1. Liên kết tâm (Centroid-linkage)

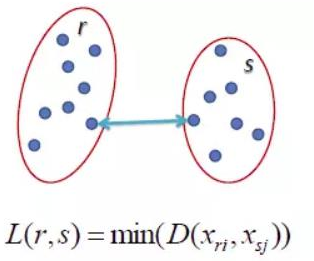
Khoảng cách giữa hai tâm của hai cụm đó là nhỏ nhất.



Hình 6. Minh họa liên kết tâm

I.7.2. Liên kết đơn (Single-linkage)

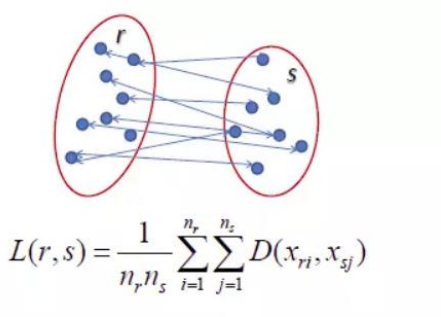
Khoảng cách giữa hai điểm gần nhau nhất của hai cụm là nhỏ nhất.



Hình 7. Minh họa liên kết đơn

I.7.3. Liên kết trung bình (Average-linkage)

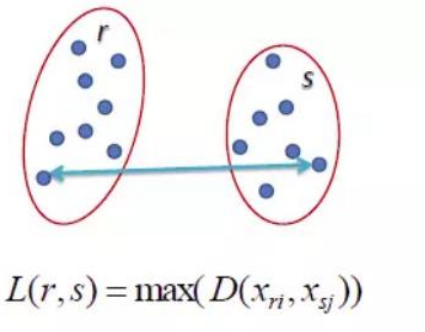
Khoảng cách trung bình giữa các cặp điểm bất kì thuộc hai cụm đó là nhỏ nhất.



Hình 8. Minh họa liên kết trung bình

I.7.4. Liên kết hoàn chỉnh (Complete-linkage)

Khoảng cách của cặp điểm xa nhau nhất của hai cụm là nhỏ nhất.



Hình 9. Minh họa liên kết hoàn chỉnh

CHƯƠNG II. THUẬT TOÁN PHÂN CỤM BIRCH

II.1. Tổng quan về thuật toán phân cụm BIRCH

Thuật toán phân cụm BIRCH là một trong những thuật toán áp dụng kỹ thuật phân cụm phân cấp. BIRCH là viết tắt của Balanced Iterative and Clustering, được phát triển vào năm 1996 bởi Tian Zhang, Raghu Ramakrishnan và Miron Livny.

BIRCH đặc biệt thích hợp cho các tập dữ liệu rất lớn hoặc để truyền trực tuyết dữ liệu, vì khả năng tìm ra giải pháp phân cụm chỉ tốt với một lần quét dữ liệu. Bên cạnh đó, thuật toán có thể quét thêm dữ liệu để cải thiện chất lượng phân cụm. BIRCH xử lý các tập dữ liệu lớn với độ phức tạp về thời gian và hiệu quả không quá vượt trội so với các thuật toán khác.

Thuật toán phân cụm BIRCH được chia làm 2 giai đoạn chính như sau:

- Giai đoạn 1: Xây dựng cây CF
- Giai đoạn 2: Phân cụm dựa trên kết quả của cây CF ở giai đoạn 1

II.2. Đầu vào, đầu ra và kiểu dữ liệu

Input: (X, B, T, L) với

- X là một tập dữ liệu với n phần tử được biểu diễn $X = (x_1, x_2, x_3, \dots x_n)$
- B (Branching Factor) số nút con tối đa được cho phép trong cây CF.
- T (Threshold) ngưỡng bán kính cho phép trong cây CF.
- L (Number of entries) số lượng thực thể tối đa được chứa trong nút lá.

Đối với ba tham số B, T, L sẽ được tìm hiểu ở giai đoạn 1 của thuật toán phân cụm BIRCH.

Output: (C, k) với

- C là tập hợp các cụm dữ liệu được biểu diễn $C = \{C_1, C_2, \dots C_k\}$ với các phần tử trong tập thỏa mãn.

(i) $C_i \cap C_j = \emptyset$ với mọi $0 < i, j \leq k$ và $i \neq j$

(ii) $C_1 \cup C_2 \cup \dots \cup C_k = X$ với X là tập dữ liệu đầu vào ở phần input

- *k* là số lượng cụm đạt được sau khi thực hiện thuật toán phân cụm

II.3. Giai đoạn 1 – Xây dựng cây CF

II.3.1. CF là gì?

Phân cụm BIRCH đạt hiệu quả cao bằng cách sử dụng thông minh một tập thống kê sơ lược nhỏ để đại diện cho một tập hợp các điểm dữ liệu lớn hơn. Đối với mục đích phân cụm, các thống kê sơ lược này tạo thành CF và đại diện cho một đặc trưng của dữ liệu thực tế. CF là viết tắt của Cluster Feature hay còn được gọi là đặc trưng cụm. Nó là một tập hợp của 3 đại số thống kê cho các điểm dữ liệu trong cụm. Các thống kê này như sau:

- Count: thống kê này mô tả có bao nhiều trá trị dữ liệu trong cụm.
- Linear Sum: tổng các giá trị riêng lẻ trong cụm với nhau. Đây còn là thước đo vị trí của cụm.
- Squared Sum: tổng các bình phương của giá trị trong cụm. Đây còn là thước đo mức độ lan truyền của cụm.

II.3.2. Cây CF

Cây CF là một cấu trúc bao gồm các CF của các cụm được bố trí hoặc sắp xếp vào cây. Cây CF trình bày cho một dạng nén dữ liệu, bảo toàn bất kì cấu trúc nào trong dữ liệu. Một cây CF có các tham số như sau:

- Yếu tố phân nhánh (kí hiệu là B) – tham số này xác định số con tối đa được cho phép cho một nút trong.
- Ngưỡng phạm vi bán kính (kí hiệu là T) – tham số này xác định ngưỡng mà các dữ liệu khi đưa vào nút không được tạo ra một cụm có bán kính vượt quá ngưỡng này.
- Số thực thể trong một nút lá (kí hiệu là L) – tham số này xác định số thực thể tối đa được cho phép bên trong một nút lá.

II.3.3. Quá trình xây dựng cây

Giai đoạn 1 của thuật toán BIRCH bao gồm việc xây dựng cây CF. Điều này được thực hiện bằng cách sử dụng phương pháp phân cụm tuần tự, theo đó thuật toán quét dữ

liệu một bản ghi cụ thể nên được gán cho một cụm hiện có hay một cụm mới nên được xây dựng. Quy trình xây dựng cây gồm 4 bước như sau:

Bước 1: Đối với bản ghi đã cho, BIRCH so sánh vị trí của bản ghi đó với vị trí của mỗi CF trong nút gốc, sử dụng tổng tuyến tính – một trong những đại số thống kê của CF hoặc giá trị trung bình của CF. BIRCH chuyển bản ghi đến các nút CF gần nhất.

Bước 2: Bản ghi sau đó đi xuống các nút trong con của nút CF đã chọn ở bước 1. BIRCH so sánh vị trí bản ghi với vị trí của mỗi CF của nút trong. BIRCH chuyển bản ghi đến nút trong gần với bản ghi nhất, cứ tiếp tục cho đến khi gặp nút trong có nút con là nút lá thì chuyển sang bước 3.

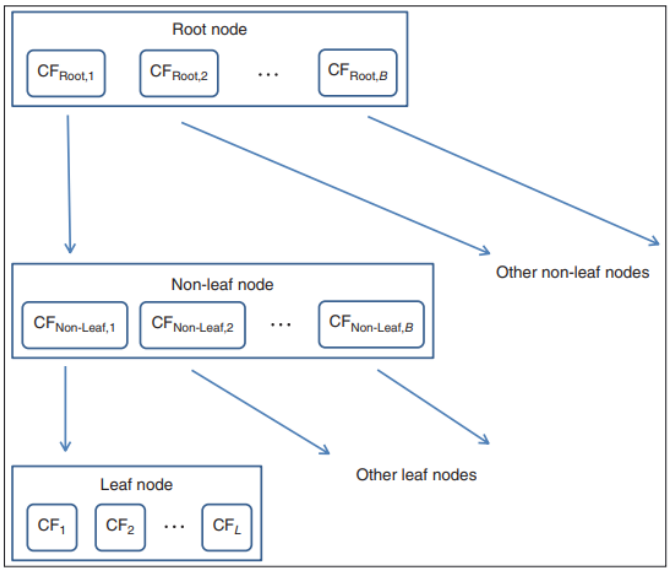
Bước 3: Bản ghi sau đó lại tiếp tục đi xuống các nút con của nút vừa chọn ở bước 2. BIRCH so sánh vị trí của bản ghi với vị trí của mỗi nút lá. BIRCH chuyển bản ghi đến nút lá gần nhất.

Bước 4: Sau khi đi đến nút trong chứa các nút lá thì thực hiện 1 trong 2 hành động dưới đây:

(a) Nếu bán kính của nút lá đã chọn bao gồm cả bản ghi không vượt quá ngưỡng T thì bản ghi sẽ được gắn vào nút lá đó. Lá và tất cả các CF mẹ của nó phải được cập nhật lại các CF bởi vì có thêm điểm dữ liệu mới.

(b) Nếu bán kính của lá vượt quá T thì một nút lá mới được hình thành, chỉ bao gồm bản ghi đang xét. Các CF mẹ của nút lá đó phải cập nhật lại CF.

Bây giờ, nếu bước 4b được thực hiện và trong trường hợp nút lá đã có tối đa L thì sau đó nút lá được tách thành hai nút lá. Các CF của nút lá xa nhất được chọn làm mốc của nút lá, với các CF còn lại được gán cho bất kỳ nút lá nào gần hơn. Nếu nút cha đã đầy thì ta tiếp tục tách nút cha,… Quá trình được minh họa bên dưới



Hình 10. Quá trình xây dựng cây CF

Mỗi nút lá có thể được xem như một cụm con. Trong bước phân cụm, các cụm con này sẽ được kết hợp thành từng cụm. Đối với một cụm nhất định, thì tâm cụm được xác định theo công thức như sau:

$$\bar{x} = \frac{\sum x_i}{n}$$

Trong đó:

x_i là giá trị các phần tử trong cụm

n là số lượng các phần tử trong cụm

Bán kính của một cụm được tính như sau:

$$R = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$$

Trong đó:

x_i là giá trị các phần tử trong cụm

$\bar{x}$ là tâm của cụm

n là số lượng phần tử trong cụm

Có một điều quan trọng trong công thức tính bán kính của một cụm, ngay cả khi không biết được các điểm dữ liệu là những giá trị gì, nhưng chỉ cần thông qua CF là ta có thể vẫn tính được bán kính của một cụm. Điều này cho phép BIRCH đánh giá xem một điểm dữ liệu nhất định có thuộc về một cụm con cụ thể mà không cần quét dữ liệu gốc hay không. Công thức như sau:

$$\sum(x_i - \bar{x})^2$$

$$= \sum(x_i^2 - 2\bar{x}x_i + \bar{x}^2)$$

$$= \sum x_i^2 - 2\bar{x}\sum x_i + n\bar{x}^2$$

$$= \sum x_i^2 - \frac{(\sum x_i)^2}{n}$$

$$= SS - \frac{LS^2}{n}$$

$$R = \sqrt{\frac{SS - \frac{LS^2}{n}}{n}}$$

Ngưỡng T cho phép thay đổi kích thước cây CF khi nó phát triển quá lớn, tức là khi vượt quá B hoặc L. Trong trường hợp này, T có thể tăng lên, cho phép nhiều bảng ghi được gắn vào cụm riêng lẻ hơn. Do đó giảm kích thước tổng thể của cây CF và cho phép nhiều bản ghi được ghi được nhập vào hơn.

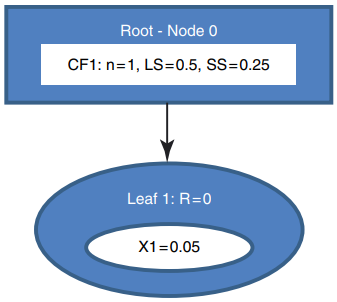
II.3.4. Ví dụ minh họa về giai đoạn xây dựng cây CF

Xét trong một bài toán như sau, ta có một tập dữ liệu một chiều có các phần tử sau: $x_1 = 0.5$, $x_2 = 0.25$, $x_3 = 0.5$, $x_3 = 0$, $x_4 = 0.65$, $x_5 = 1$, $x_6 = 1.4$, $x_7 = 1.1$

Xác định 3 tham số cho quá trình xây dựng cây CF

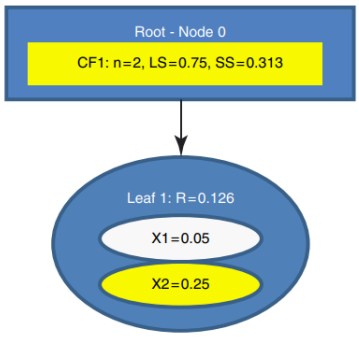
- T = 0.15
- B = 2
- L = 2

Giá trị dữ liệu đầu tiên $x_1 = 0.5$ được duyệt. Nút gốc được khởi tạo với các giá trị CF của giá trị dữ liệu đầu tiên. Một nút lá mới Leaf1 được tạo, và BIRCH gán bảng ghi đầu tiên x1 cho Leaf1. Bởi vì nó chỉ chứa bản ghi duy nhất, bán kính của Leaf1 là 0, do đó nhỏ hơn T = 0.15. Cây CF sau khi dữ liệu đầu tiên được gắn vào cây được hiển thị trong hình.



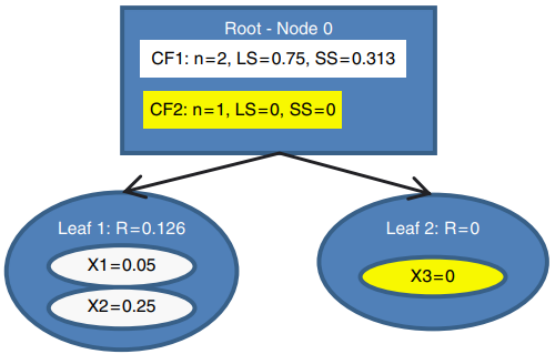
Hình 11. Kết quả khi đưa x1 vào cây

Giá trị dữ liệu thứ hai $x_2 = 0.25$ được duyệt. BIRCH chuyển x_2 đến Leaf1. Bán kính của Leaf1 bây giờ $R = 0.126 < T = 0.15$, do đó x_2 được gán cho Leaf1. Giá trị CF của CF1 sau đó được cập nhật như trong bên dưới.



Hình 12. Kết quả khi đưa x2 vào cây

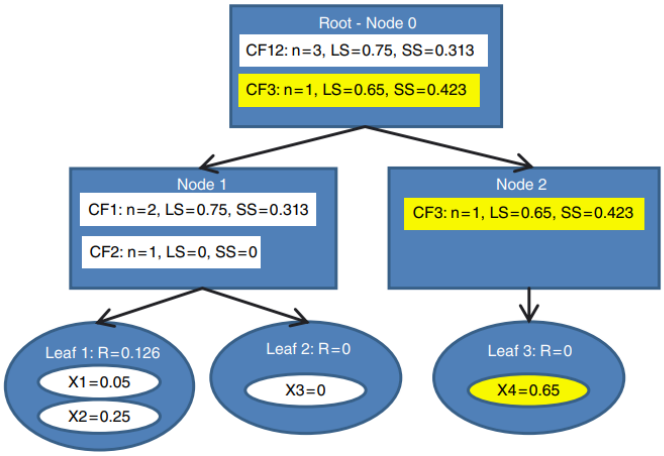
Giá trị dữ liệu $x_3 = 0$ được duyệt. BIRCH chuyển $x_3 = 0$ đến Leaf1. Tuy nhiên, bán kính của Leaf1 giờ đã tăng lên $R = 0.205 > T = 0.15$. Giá trị ngưỡng T = 0.15 bị vượt quá, vì vậy x_3 không được gán cho Leaf1. Thay vào đó, một lá mới được khởi tạo, được gọi là Leaf2, chỉ chứa x_3. CF1 và CF2 được cập nhật lại và hiển thị trong hình bên dưới.



Hình 13. Kết quả khi đưa x3 vào cây

Giá trị dữ liệu thứ 4 $x_4 = 0.65$ được duyệt. BIRCH so sánh x_4 với vị trí của CF1 và CF2. Vị trí được đo bằng x = LS / n. Ta có xCF1 = 0.75/2=0.375 và xCF2 = 0/1 = 0. Điểm dữ liệu $x_4 = 0.65$ do đó gần với CF1 hơn CF2. BIRCH chuyển x_4 đến CF1. Bán kính của CF1 lúc này tăng lên R = 0.166 > T = 0.15. Giá trị ngưỡng T bị vượt quá, vì vậy x_4 không được chỉ định cho CF1.

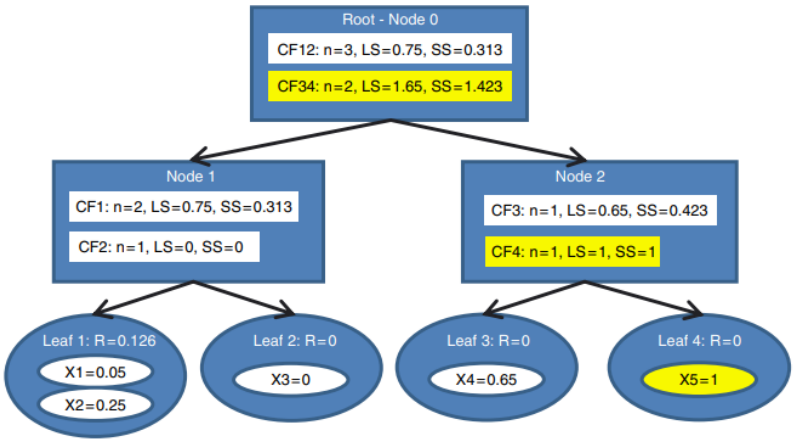
Thay vào đó, chúng ta sẽ khởi lá mới, tuy nhiên L = 2 có nghĩa là chúng ta không thể có 3 thực thể trong một nút lá. Do đó, chúng ta phải chia nút gốc thành Node1, có các nút con là Leaf1 và Leaf2, và Node2 có lá duy nhất là Leaf3 chỉ chứa x_4, như hình minh họa trong hình bên dưới. Các giá trị CF có liên quan đều phải được cập nhật lại.



Hình 14. Kết quả khi đưa x4 vào cây

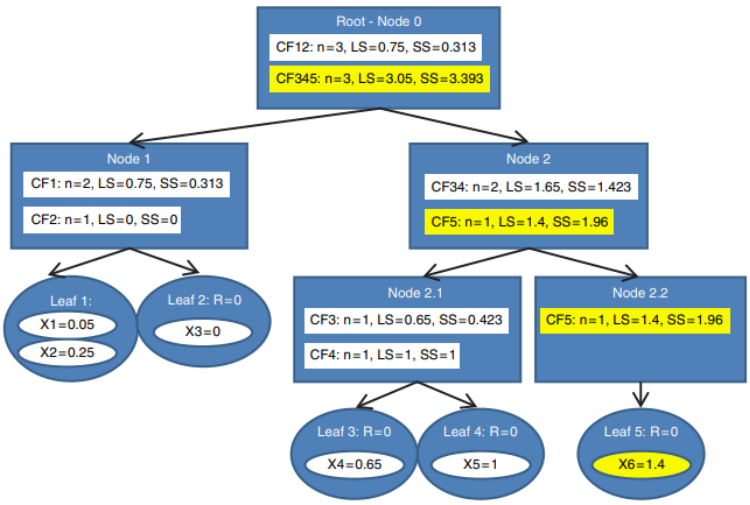
Giá trị dữ liệu thứ 5 $x_5 = 1$ được duyệt. BIRCH so sánh x_5 với vị trí của CF12 và CF3. Ta có xCF12 = 0.75/3=0.25 và xCF4=0.65/1=0.65. Do đó, điểm dữ liệu $x_5 = 1$ gần với CF3 hơn là CF12. BIRCH chuyển x_5 đến CF3. Bán kính của CF3 bây giờ tăng lên R=0.175

> T = 0.15, vì vậy x_5 không thể được gán cho CF3. Thay vào đó, một lá mới trong nút lá Leaf4 được khởi tạo với CF, CF4, chỉ chứa x_5.



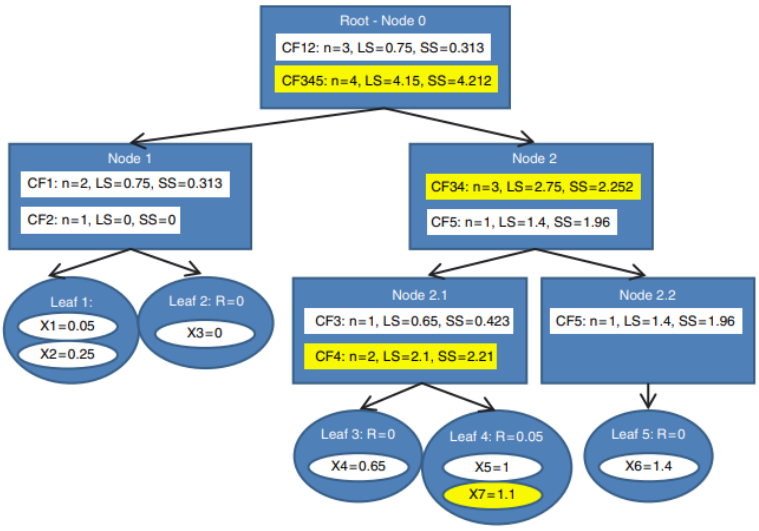
Hình 15. Kết quả khi đưa x5 vào cây

Giá trị dữ liệu thứ sáu x_6 = 1,4 được duyệt. Tại nút gốc, BIRCH so sánh x_6 = 1,4 với vị trí của CF12 và CF34. Ta có xCF12 = 0,75 / 3 = 0,25 và xCF34 = 1,65 / 2 = 0,825. Do đó, điểm dữ liệu x_6 = 1,4 gần với CF34 hơn và BIRCH chuyển x6 đến CF34. Dữ liệu được đưa xuống Node2 và BIRCH so sánh x_6 = 1,4 với vị trí của CF3 và CF4. Ta có xCF3 = 0,65 và xCF4 = 1. Do đó, điểm dữ liệu x_6 = 1,4 gần với CF4 hơn là CF3. BIRCH chuyển x_6 đến CF4. Bán kính của CF4 lúc này tăng lên R = 0,2> T = 0,15. Giá trị ngưỡng T = 0,15 bị vượt quá, vì vậy x_6 không được chỉ định cho CF4. Nhưng hệ số phân nhánh B = 2 có nghĩa là chúng ta có thể có nhiều nhất hai nút lá phân nhánh của bất kỳ nút không phải lá nào. Do đó, chúng ta sẽ cần một tập hợp các nút trong thành nút mới, Node2.1 và Node2.2, phân nhánh từ Node2. Node2.1 chứa CF3 và CF4, trong khi Node2.2 chứa CF5 mới mong muốn và nút lá mới Leaf5 là con duy nhất của nó, chỉ chứa thông tin từ x_6. Cây này được thể hiện trong hình dưới.



Hình 16. Kết quả khi đưa x6 vào cây

Cuối cùng, giá trị dữ liệu cuối cùng $x_7 = 1.1$ được duyệt. Trong nút gốc, BIRCH so sánh $x_7 = 1,1$ với vị trí của CF12 và CF345. Ta có xCF12 = 0,25 và xCF345 = 1,02, do đó x7 = 1,1 gần với CF345 hơn và BIRCH chuyển x_7 đến CF345. Dữ liệu sau đó được đưa xuống Node2. So sánh tại nút này có $x_7 = 1,1$ gần với CF34 hơn là CF5. Dữ liệu sau đó đưa xuống Node 2.1. Ở đây, $x_7 = 1,1$ gần với CF4 hơn CF3. BIRCH chuyển x_7 đến CF4 và đến Leaf4. Bán kính của Leaf4 trở thành R = 0,05, không vượt quá giá trị ngưỡng bán kính T = 0,15. Do đó, BIRCH chỉ định x_7 cho Leaf4. Dạng cuối cùng của cây CF được thể hiện trong hình bên dưới.



Hình 17. Kết quả khi đưa x7 vào cây

II.4. Giai đoạn 2 – Phân cụm dựa trên cây CF được xây dựng ở giai đoạn 1

II.4.1. Đôi nét về giai đoạn 2

Khi cây CF được xây dựng, bất kỳ thuật toán phân cụm nào hiện có cũng có thể được áp dụng cho các cụm con (các nút lá của cây CF), để kết hợp các cụm con này thành các cụm lớn hơn.

Như đã đề cập trước đó, phân cụm BIRCH đạt hiệu quả cao bằng cách thay thế một tập hợp 3 giá trị thống kê bao gồm LS, SS, Count để đại diện cho một tập hợp các điểm dữ liệu. Khi một giá trị dữ liệu mới được thêm vào, những giá trị CF này có thể được cập nhật dễ dàng. Do đó cây CF nhỏ hơn nhiều so với dữ liệu gốc, cho các phép tính toán được thực hiện hiệu quả hơn.

Sau đây là một điều bất lợi của việc phân cụm BIRCH. Do cấu trúc cây vốn có trong cây CF, giải pháp phân cụm có thể phụ thuộc vào thứ tự đầu vào của các bản ghi dữ liệu. Để tránh điều này, nhà phân tích dữ liệu có thể muốn áp dụng phân cụm BIRCH trên một số cách sắp xếp dữ liệu theo nhiều cách khác nhau và tìm sự tương đồng giữa các kết quả.

Tuy nhiên, một lợi ích của phân cụm BIRCH là nhà phân tích không bắt buộc phải chọn lựa chọn số *k* tức là số lượng cụm, điều này khác với một số thuật toán phân cụm khác.

II.4.2. Giải pháp tìm *k* hợp lý cho quá trình phân cụm

Việc lựa chọn *k* tốt nhất sẽ thông qua một phương pháp được giới thiệu ngay đây. Ở đây chúng ta sẽ áp dụng một trong các phương pháp tìm ra *k* hợp lý nhất để phục vụ trong quá trình phân cụm. Hiện nay có nhiều phương pháp xác định số *k* thích hợp như elbow, average silhouette, gap statistic,…, trong phạm vi của đề tài này sẽ sử dụng phương pháp *pseudo-f statistic*.

Pseudo-f statistic thường được dùng để quyết định số lượng các cụm. Tập hợp các cụm có giá trị *pseudo-f* lớn nhất được chọn nhưng ở trong giai đoạn này ta sẽ chọn *p-value* để làm giá trị so sánh và lựa chọn *k* lý tưởng nhất.

Giả sử chúng ta có *k* cụm. Với n_i giá trị dữ liệu tương ứng, sao cho $\sum n_i$ = N tức là tổng cỡ mẫu. Để x_{ij} tham chiếu đến giá trị dữ liệu thứ *j* trong cụm thứ *i*, để m_i tham chiếu đến tâm của cụm thứ *i*, để *M* đại diện cho giá trị trung bình của tất cả dữ liệu trong tập.

Xác định SSB, tổng bình phương giữa các cụm, được định nghĩa như sau:

$$SSB = \sum_{i=1}^{k} n_i . Distance^2(m_i, M)$$

Xác định SSE - tổng các bình phương trong các cụm, như sau:

$$SSE = \sum_{i=1}^{k} \sum_{j=1}^{n_j} Distance^2(x_{ij,} m_j)$$

Với

$$Distance(a, b) = \sqrt{\sum (a_i - b_i)^2}$$

Giá trị pseudo-f statistic được tính bằng

$$F = \frac{MSB}{MSE} = \frac{SSB/(k-1)}{SSE/(N-k)}$$

Pseudo-f statistic là đo lường tỉ lệ sự tách biệt giữa các cụm, được đo bằng *MSB*, bình phương trung bình giữa các cụm, với sự lan truyền của dữ liệu trong các cụm, được đo bằng bình phương trung bình lỗi, *MSE*.

Các giả thuyết đang được kiểm tra như sau:

H_0 : Không có cụm nào trong tập dữ liệu

H_1 : Tồn tại k cụm trong tập dữ liệu

Để bác bỏ H_0 thì ta phải có p-value đủ nhỏ, trong đó

$$p - value = P(F_{k-1,n-k} > pseudo - F\ value)$$

Lý do mà chúng ta gọi là thống kê này là thống kê giả *f (pseudo-f)*, là vì nó bác bỏ giả thuyết vô hiệu một cách quá dễ dàng. Thông thường tùy thuộc vào mục đích hoặc quy định của người phân tích mà việc chọn mốc p-value cũng sẽ khác nhau. Có 3 giá trị p-value thường được dùng là 0.1, 0.05, 0.01[3]. Nếu p-value nhỏ hơn một trong những mức này thì bác bỏ giả thuyết vô hiệu. Với *p-value* gần 0, bác bỏ mạnh mẽ giả thuyết rỗng rằng không có cụm nào trong dữ liệu. Nhưng vì dữ liệu được tạo ngẫu nhiên, nên giả định phải là không có cụm thực nào trong dữ liệu.

II.4.3. Ví dụ minh họa cho giai đoạn 2

Trước khi tiến hành phân cụm thì ta phải đưa ra một bản khảo sát với từng giá trị p-value của từng ứng viên thông phương pháp *pseudo-f statistic* ở trên.

Chúng ta cũng sẽ tiếp tục với bài toán ở giai đoạn 1 và sử dụng kết quả của nó để tiến hành giai đoạn 2.

Tính toán giá trị *pseudo-f* và *p-value* cho mỗi ứng viên, và chọn ứng viên có *p-value* nhỏ nhất làm giải pháp phân cụm tốt nhất. Đối với mỗi giải pháp phân cụm ứng viên được trình bày ở trên, giá trị *pseudo-f* và *p-value* được tính toán và được trình bày ở hình bên dưới.

Value of *k*	MSB	MSE	Pseudo-*F*	*p*-Value
2	**1.1433**	**0.3317**	**17.24**	**0.009**
3	0.6533	0.0408	15.52	0.013
4	0.4628	0.0289	16.02	0.024
5	0.3597	0.0181	19.84	0.049

Hình 18. Kết quả khảo sát các ứng viên

Thông qua kết quả đã thực hiện ở *hình 19* thì ta có thể thấy *p-value* nhỏ nhất chính là ứng viên *k* = 2. Với giá trị *p-value* = 0.009.

Sau khi đã tìm được số k lý tưởng thì ta tiến hành phân cụm theo chiến lược Bottom-Up. Dự vào cây CF ở giai đoạn 1 ta có các giá trị tâm cụm như sau:

$$\bar{x}_{CF_2} = 0;\ \ \bar{x}_{CF_1} = 0.375;\ \ \bar{x}_{CF_3} = 0.65;\ \ \bar{x}_{CF_4} = 1.05;\ \bar{x}_{CF_5} = 1.4$$

Chúng ta tiến hành nhóm 2 cụm có khoảng cách gần nhau nhất vào thành 1 cụm và điều này lặp đi lặp lại cho đến khi kết quả cuối cùng của chúng còn lại *k* cụm. Với số *k* là số cụm mà chúng ta đã có được qua bước khảo sát và tìm ra ứng viên lý tưởng nhất, trong bài toán của chúng ta thì cụ thể *k* = 2.

Hai cụm gần nhất là CF_1 và CF_3. Kết hợp các CF này, chúng ta thu được tâm cụm mới là $x_{CF1,3} = (2 \times 0{,}375 + 1 \times 0{,}65)/3 = 0{,}47$. Các tâm cụm còn lại là:

$$\bar{x}_{CF_2} = 0;\ \ \bar{x}_{CF_{1,3}} = 0.47;\ \ \bar{x}_{CF_4} = 1.05;\ \bar{x}_{CF_5} = 1.4$$

Ở đây, hai cụm gần nhất là CF_4 và CF_5. Khi chúng ta kết hợp các CF này, tâm cụm được kết hợp là $x_{CF4,5} = (2 \times 1{,}05 + 1 \times 1{,}4)/3 = 1{,}17$. Các trung tâm cụm còn lại là:

$$\bar{x}_{CF_2} = 0;\ \ \bar{x}_{CF_{1,3}} = 0.47;\ \bar{x}_{CF_{4,5}} = 1.17$$

Trong số này, hai cụm gần nhất là CF_2 và $CF_{1,3}$. Kết hợp các CF này, chúng ta nhận được tâm cụm mới là $x_{CF2,1,3} = (2 \times 0{,}375 + 1 \times 0 + 1 \times 0{,}65)/4 = 0{,}35$. Điều này để lại cho chúng ta các tâm cụm cuối cùng còn lại:

$$\bar{x}_{CF_{2,1,3}} = 0.35;\ \ \ \bar{x}_{CF_{4,5}} = 1.17$$

Sau bước trên thì ta thấy rằng số cụm còn lại chỉ còn lại 2 và nó đã bằng với số *k* mà ta đã tìm được. Vì thế chúng ta dừng tại đây và đưa ra kết quả phân cụm cuối cùng.

CHƯƠNG III. VÍ DỤ MINH HỌA PHÂN CỤM VỚI THUẬT TOÁN BIRCH

III. 1. Giới thiệu ngôn ngữ lập trình và các thư viện

III.1.1. Ngôn ngữ lập trình C#

Ứng dụng demo sử dụng trong đồ án này được viết bằng ngôn ngữ lập trình C# với phiên bản cụ thể là .NET 5. Ngôn ngữ được phát triển bởi Microsoft vào năm 2000 cho đến nay, C# là một ngôn ngữ hướng đối tượng vì thế chương trình cũng được trình theo mô hình hướng đối tượng (OOP) với mục đích dễ dàng phát triển và mở rộng.

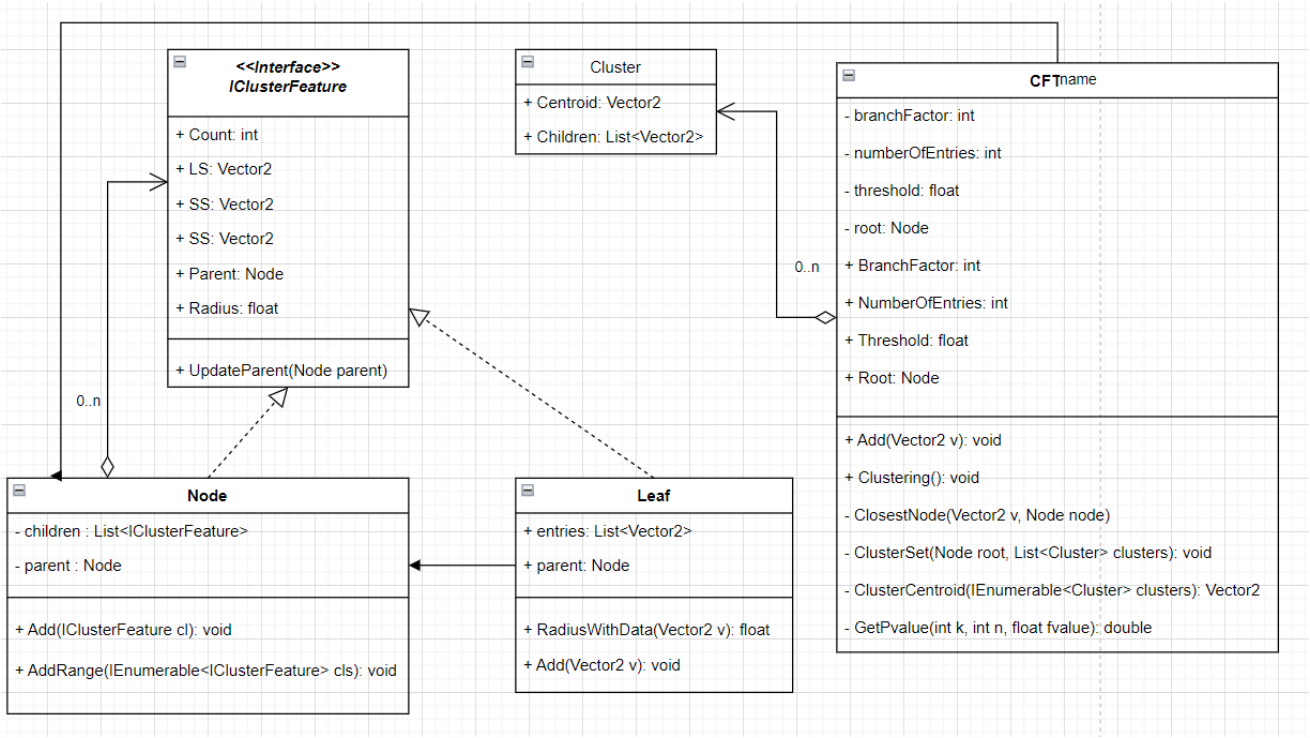
III.1.2. Các gói thư viện được dùng trong chương trình

Chi tiết các thư viện được trình bày ở bảng bên dưới

Tên gói thư viện	Công dụng
Extreme.Statistics	Thư viện cung cấp một số phương thức liên quan đến xác suất thống kê. Cụ thể hơn trong chương trình demo nó được dùng để tính pseudo-f value và p-value
Newtonsoft.Json	Thư viện dùng để làm việc với dữ liệu JSON trong chương trình, bao gồm việc chuyển dữ liệu đối tượng sang kiểu JSON và ngược lại.
ChartJs	Thư viện này giúp hiển thị dữ liệu dưới dạng biểu đồ trực quan, giúp ta dễ dàng quan sát kết quả đạt được
CsvHelper	Thư viện hỗ trợ thao tác với dữ liệu từ tệp .csv. Bao gồm đọc, viết,… tệp .csv. Một điểm đặc biệt là sau khi đọc dữ liệu nó hỗ trợ chúng ta ánh xạ dữ liệu đó sang kiểu đối tượng mà ta chỉ định.

III.2. Các hàm chức năng trong chương trình

Để dễ dàng triển khai và hiểu rõ hơn những gì mà ứng dụng xử lý dữ liệu thì bên dưới sẽ là sơ đồ lớp gồm chi tiết các đối tượng, thuộc tính của từng lớp.



Hình 19. Sơ đồ lớp của ứng dụng demo

III.2.1. Xây dựng cây CF

Trước tiên sẽ cùng tiến hành thực hiện xây dựng cây CF ở giai đoạn 1, kết hợp với sơ đồ lớp ở trên để hiểu rõ hơn về ý nghĩa và cách hoạt động của từng module. Với ứng dụng demo đang thực hiện với dữ liệu hai chiều, nên sẽ dùng kiểu dữ liệu Vector2 tích hợp sẵn trong C# để triển khai bài toán phân cụm.

Tìm nút gần nhất

Ở giai đoạn 1 thì dữ liệu đầu vào của chúng ta sẽ được xét duyệt với từng nút theo từng cấp từ trên xuống dưới. Với mỗi nút gần với dữ liệu đầu vào thì sẽ tiến hành tiếp tục so sánh với các nút con của nút đó cho đến khi gặp nút gần nhất. Ở đoạn code bên dưới với trường hợp cây CF của chúng ta chưa có bất kì đối tượng dữ liệu nào được thêm, thì nút hàm này sẽ trả về nút gốc là nút gần nhất. Còn trong các trường hợp còn lại thì chương trình sẽ thực hiện vòng lặp để tìm nút gần nhất của từng cấp cho đến khi gặp nút gần nhất là nút lá.

```
1 reference
private IClusterFeature ClosestNode(Vector2 data, Node node)
{
    IClusterFeature closestNode = node;
    float minDistance = -1;
    if (node.Count == 0)
        return node;
    while (closestNode is not Leaf)
    {
        foreach (var child in ((Node)closestNode).Children)
        {
            var distance = Vector2.Distance(child.LS / child.Count, data);
            if (distance < minDistance || minDistance == -1)
            {
                minDistance = distance;
                closestNode = child;
            }
        }
        minDistance = -1;
    }
    return closestNode;
}
```

Tính bán kính của cụm

Dựa vào công thức tính bán kính của một cụm được trình bày ở chương II thì đoạn code sẽ được xây dựng như sau

```
1 reference
public float RadiusWithData(Vector2 data)
{
    var ss = SS + data * data;
    var ls = LS + data;
    var n = Count + 1;

    var ts = ss - (ls * ls / n);
    var ms = n;

    return Vector2.SquareRoot(ts / ms).Length();
}
```

Thêm một đối tượng dữ liệu vào cây CF

Hàm này có thể nói là hàm khá quan trọng trong quá trình xây dựng cây. Với tham số là một đối tượng dữ liệu thì module sẽ tiến hành tìm kiếm xem nút nào lá nào là nút gần nhất với dữ liệu đang xét. Sau khi tìm được nút lá gần nhất, ta tiến hành tính bán kính của đối tượng dữ liệu đó với nút lá gần nhất, nếu nút bán kính vượt quá giới hạn tham số *T* thì sẽ tạo ra nút lá mới. Nhưng khi tạo ra nút lá mới cần chú ý rằng, nếu nút lá mới được tạo ra mà lại làm vượt quá giớn hạn của tham số *B* thì ta đi tới nút cha của nút mà ta định gắn nút lá vào để xem xét có thể gắn nút lá mới hay không nếu không thì cứ tiếp tục làm kiểm tra cho đến khi đạt điều

kiện cho phép. Đoạn code bên dưới sẽ mô tả quá trình đó và sẽ có một thuộc tính có sẵn là "parent" để lưu trữ nút cha của nút đó.

```
21 references
public void Add(Vector2 data)
{
    var closestNode = ClosestNode(data, root);
    if (closestNode.Equals(root))
    {
        root.Add(clusterFeature: new Leaf(root, data));
        return;
    }
    var radius = closestNode is Leaf ? ((Leaf)closestNode).RadiusWithData(data)
        : throw new InvalidOperationException(message: "Không phải nút lá");
    if (radius <= threshold)
    {
        if (closestNode.Count < numberOfEntries)
        {
            ((Leaf)closestNode).Add(data);
            return;
        }
    }
    var parentNode = ((Leaf)closestNode).Parent;
    while (parentNode != null)
    {
        if (parentNode.Children.Count < branchingFactor)
        {
            parentNode.Children.Add(item: new Leaf(parentNode, data));
            return;
        }
        parentNode = parentNode.Parent;
    }
    parentNode = root;
    var tmp = new List<IClusterFeature>(parentNode.Children);
    parentNode.Children.Clear();
    var newNode1 = new Node(parentNode);
    newNode1.Children.AddRange(tmp);
    tmp.ForEach(c => c.UpdateParent(newNode1));
    var newNode2 = new Node(parentNode);
    newNode2.Add(clusterFeature: new Leaf(newNode2, data));
    parentNode.Children.Add(newNode1);
    parentNode.Children.Add(newNode2);
    root = parentNode;
}
```

Bây giờ thì ta đã có đủ công cụ để hoàn thành giai đoạn 1 - xây dựng 1 cây CF. Bên trên là đoạn code chỉ cho phép thêm một đối tượng dữ liệu, nếu muốn thêm nhiều ta cho chạy vòng lặp để cây CF của chúng ta có thể tiếp nhận các dữ liệu một cách tuần tự.

III.2.2. Phân cụm dữ liệu

Sau khi có được cây CF thì dựa trên đối tượng cây CF ta tiến hành phân cụm với các module sau

Chuyển các nút lá thành đối tượng Cluster

Mục đích của module này là biến đổi các nút lá trong cây CF về đối tượng được khai báo sẵn trong chương trình nhằm phục vụ cho việc gộp nhóm các cụm các tính toán các đại lượng thống kê.

```
2 references
private void ClusterSet(Node root, List<Cluster> clusters)
{
    if (root == null)
    {
        throw new ArgumentNullException(paramName: "Root không được phép null");
    }
    if (root.Children == null)
    {
        throw new ArgumentNullException(paramName: "Các cụm con của Root không được phép null");
    }
    foreach (var i in root.Children)
    {
        if (i is Leaf leaf)
        {
            clusters.Add(item: new Cluster()
            {
                Children = leaf.Entries,
                Centroid = leaf.Entries.Average(),
            });
        }
        else
        {
            ClusterSet((Node)i, clusters);
        }
    }
}
```

Tính tâm của cụm

Dựa vào công thức tính tâm cụm được trình bày ở chương II ta viết được đoạn code như bên dưới.

```
1 reference
private Vector2 ClusterCentroid(IEnumerable<Cluster> clusters)
{
    Vector2 sum = Vector2.Zero;
    int n = 0;
    foreach (Cluster cluster in clusters)
    {
        sum = sum + cluster.Children.Sum(c => c);
        n += cluster.Children.Count;
    }
    return sum / n;
}
```

Tính giá trị p-value

Sử dụng thư viện đã cung cấp tại phần III.1.2 thư viện này cung cấp nhiều phương thức làm việc với các kiến thức liên quan xác suất thống kê. Module này vận dụng thư viện này để tính p-value phục vụ cho quá trình tìm k (số cụm) tốt nhất cho bài toán.

```
1 reference
private double GetPvalue(int k, int n, float pseudoValue)
{
    var df1 = k - 1;
    var df2 = n - k;
    FDistribution fdis = new FDistribution(df1, df2);
    var pvalue = fdis.RightTailProbability(pseudoValue);
    return pvalue;
}
```

Phân cụm dữ liệu

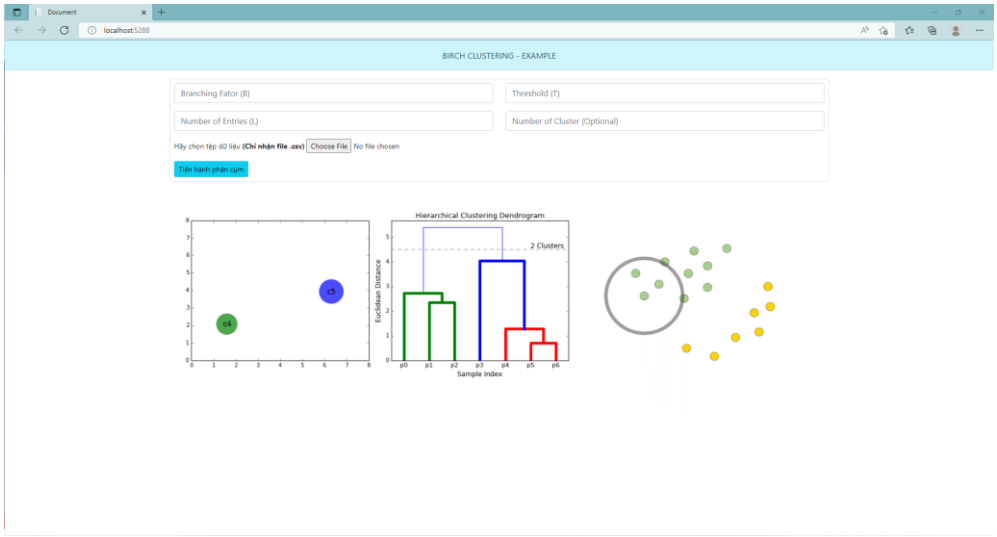
```
2 references
public List<PoolResult> Clustering()
{
    List<Cluster> clusters = new List<Cluster>();
    List<PoolResult> results = new List<PoolResult>();
    bool isInit = true;
    ClusterSet(root, clusters);
    int n = clusters.Sum(c => c.Children.Count);
    var nClusters = clusters.Count;
    var clusterCentroid = ClusterCentroid(clusters);
    while (clusters.Count > 2)
    {
        if (!isInit)
        {
            float minDistance = -1;
            int iInx = -1;
            int jInx = -1;
            for (int i = 0; i < clusters.Count - 1; i++)
            {
                for (int j = i + 1; j < clusters.Count; j++)
                {
                    if (i != j)
                    {
                        var distance = Vector2.Distance(clusters[i].Centroid, clusters[j].Centroid);
                        if (distance < minDistance || minDistance == -1)
                        {
                            minDistance = distance;
                            iInx = i;
                            jInx = j;
                        }
                    }
                }
            }
            clusters[iInx].Children.AddRange(clusters[jInx].Children);
            clusters[iInx].Centroid = clusters[iInx].Children.Average();
            clusters.RemoveAt(jInx);
        }
```

```
            int k = clusters.Count;
            var SSB = clusters.Sum(c => c.Children.Count * Vector2.DistanceSquared(c.Centroid, clusterCentroid));
            var SSE = clusters.Sum(c => c.Children.Sum(ch => Vector2.DistanceSquared(ch, c.Centroid)));
            var MSB = SSB / (k - 1);
            var MSE = SSE / (n - k);
            var pseudoFValue = MSB / MSE;
            var pvalue = (float)GetPvalue(k, n, pseudoFValue);
            results.Add(item: new PoolResult()
            {
                RawData = JsonConvert.SerializeObject(clusters.Select(c => new { c.Children, c.Centroid })),
                MSB = (float)Math.Round(MSB, digits: 3),
                MSE = (float)Math.Round(MSE, digits: 3),
                PseudoF = (float)Math.Round(pseudoFValue, digits: 3),
                Pvalue = (float)Math.Round(pvalue, digits: 15),
                K = k,
                N = n,
                Centroid = clusterCentroid,
            });
            isInit = false;
        }
        return results;
    }
```

III.2.3. Ứng dụng với giao diện trực quan

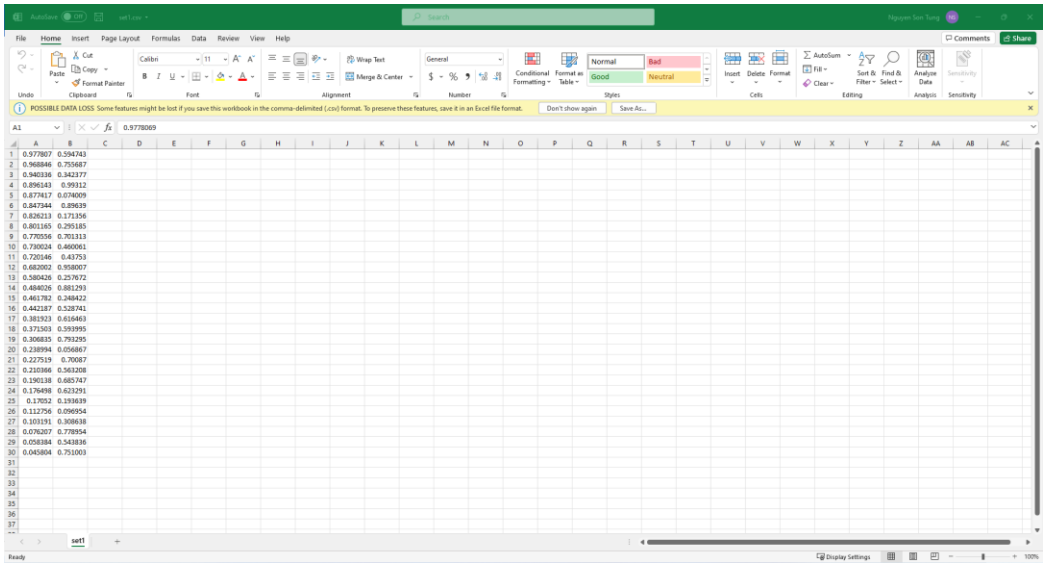
Để có cái nhìn trực quan hơn về kết quả phân cụm sau khi được phân cụm bằng thuật toán BIRCH. Trong đồ án này sử dụng ASP .NET MVC được viết bằng C# để xử lý dữ liệu và đưa dữ liệu lên nền tảng web.

Đầu tiên sẽ là một màn hình nhập liệu để nhập các tham số phục vụ cho quá trình phân cụm. Ở đây có 5 đầu vào bao gồm: B (Branch Factor), T (Threshold), L (Number of Entries) chi tiết về 3 tham số này đã được đề cập tại chương II. Bên cạnh đó có một tham số tùy chọn là “Number of Cluster” có nghĩa là thay vì để chương trình phân cụm và tìm ra số cụm phù hợp nhất, chúng ta có thể chỉ định số cụm kết quả cho dữ liệu đầu vào. Đầu vào cuối cùng một tệp dữ liệu với định dạng với đuôi là ***csv***.



Hình 20. Chương trình minh họa

Với tệp ***csv*** chúng ta sẽ nhập liệu cho tệp với 2 cột dữ liệu, tại sao lại là 2 cột dữ liệu là vì chương trình demo này đang xử lý dữ liệu có 2 thuộc tính. Trong thực tế thì dữ liệu có thể có rất nhiều thuộc tính, vì đây chỉ là chương trình minh họa cho kỹ thuật phân cụm BIRCH, nên chúng ta sẽ tạm thời chấp nhận dữ liệu chỉ có 2 thuộc tính. Bên dưới sẽ là dữ liệu đầu vào mẫu cho chương trình.



Hình 21. Dữ liệu đầu vào với tệp .csv

Sau khi nhập dữ liệu đầy đủ thì nhấn vào nút tiến hành phân cụm để chương trình tính toán và xử lý. Kết quả cuối cùng sẽ được hiển thị dưới dạng biểu đồ, biểu đồ này được dùng từ thư viện chartjs đã được đề cập ở phần III.1.2. Mỗi điểm sẽ được biểu diễn tương

ứng với một chấm tròn ứng với một màu sắc nhất định. Mỗi cụm sẽ có một màu đặc trưng và mỗi phần tử trong cụm đó sẽ có màu tương ứng. Đôi lúc việc nhập đầu vào sẽ không nhập được kết quả nào có thể là do tham số đầu vào của chúng ta không phù hợp với tệp dữ liệu mà ta đã nhập vào. Bên dưới là hình ảnh minh họa cho kết quả phân cụm.

Hình 22. Kết quả sau khi thực thi

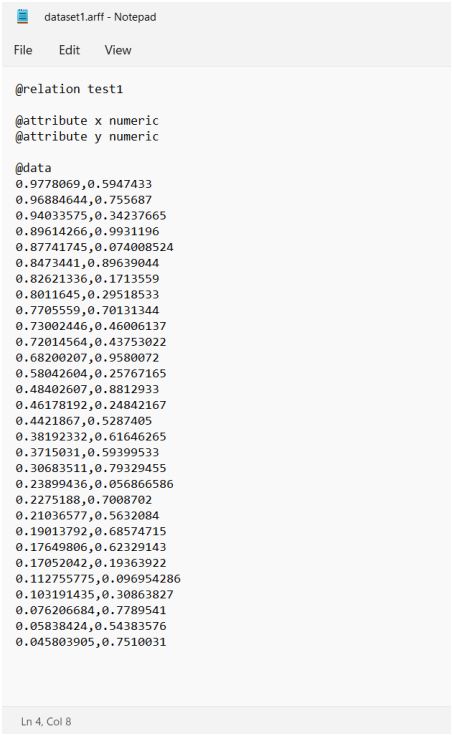
III.4. So sánh kết quả thuật toán phận cụm trong phần mềm WEKA

III.4.1. Giới thiệu về phần mềm WEKA

WEKA có tên đầy đủ là Waikato Environment for Knowledge Analysis. WEKA là phần mềm mã nguồn mở miễn phí thuộc dự án nghiên cứu của Đại học Waikato, New Zealand. WEKA được xây dựng và viết bằng ngôn ngữ lập trình Java theo kiến trúc hướng đối tượng, được tạo ra để phục vụ cho lĩnh vực khai phá dữ liệu và học máy. WEKA hỗ trợ giao diện đồ họa giúp người dùng dễ sử dụng.

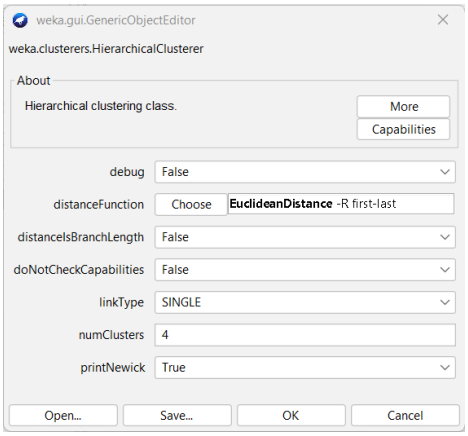
III.4.2. Phân cụm với WEKA

Trước tiên cần chuẩn bị dữ liệu cho chương trình, tập dữ liệu (dataset) mà WEKA hỗ trợ có đuôi tệp là *arff*. Bên dưới sẽ là ví dụ tạo ra một tập dữ liệu mà đã sử dụng minh họa cho chương trình phân cụm BIRCH ở *hình 21*. Có một số câu chỉ thị trong tệp cần chú như sau: @relation dùng để khai báo tên quan hệ của tập dữ liệu, @attribute khai báo tên thuộc tính và kiểu dữ liệu của thuộc tính, @data chỉ thị để ghi dữ liệu của tệp.



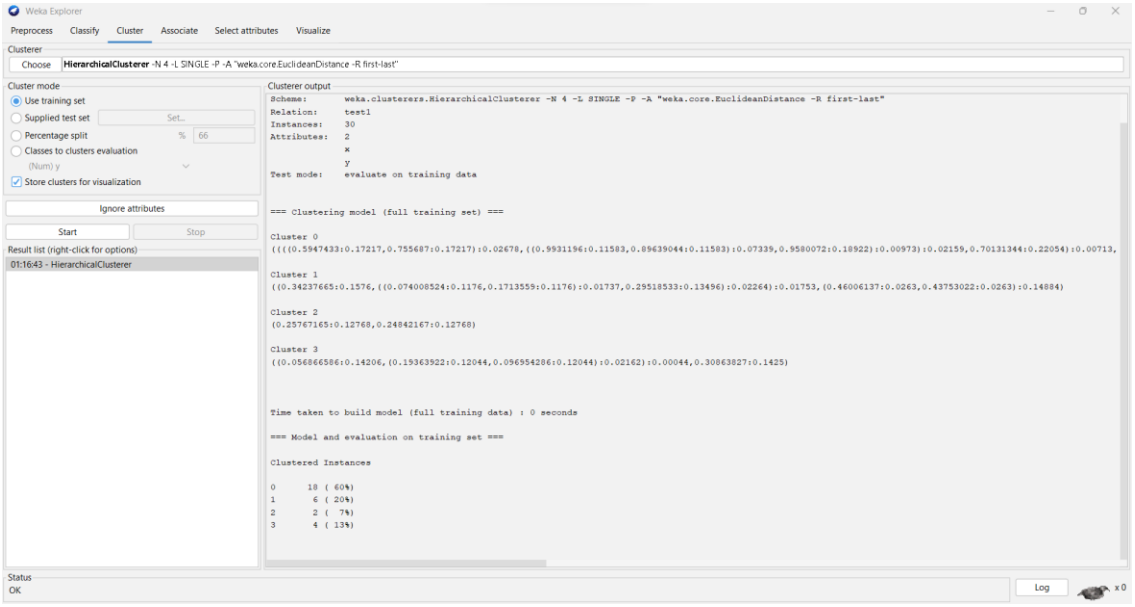
Hình 23. Tạo tập dữ liệu

Sau khi đã có dữ liệu thì tiến hành nhập liệu cho phần mềm WEKA. Vào tab Cluster chọn thuật toán phân cụm mong muốn, mà cụ thể ở đây sẽ là phân cụm phân cấp (Hierarchical clustering). Điều chỉnh chọn số cụm cần chọn là 4 vì ở kết quả *ở hình 22* có 4 cụm. Sau đó chọn những mục tùy chọn ví dụ như distanceFunction tức là chọn độ đo cho thuật toán ta sẽ chọn EuclideanDistance và linkType (đã được giới thiệu tại mục I.7) chọn Single.



Hình 24. Các mục tùy chọn cho phân cụm phân cấp

Sau đó click vào Ok và xem kết quả ở *hình 25*



Hình 25. Kết quả phân cụm phân cấp với WEKA

Dựa vào kết quả ở hình 25 có thể thấy được kết quả phân cụm gồm 4 cụm với

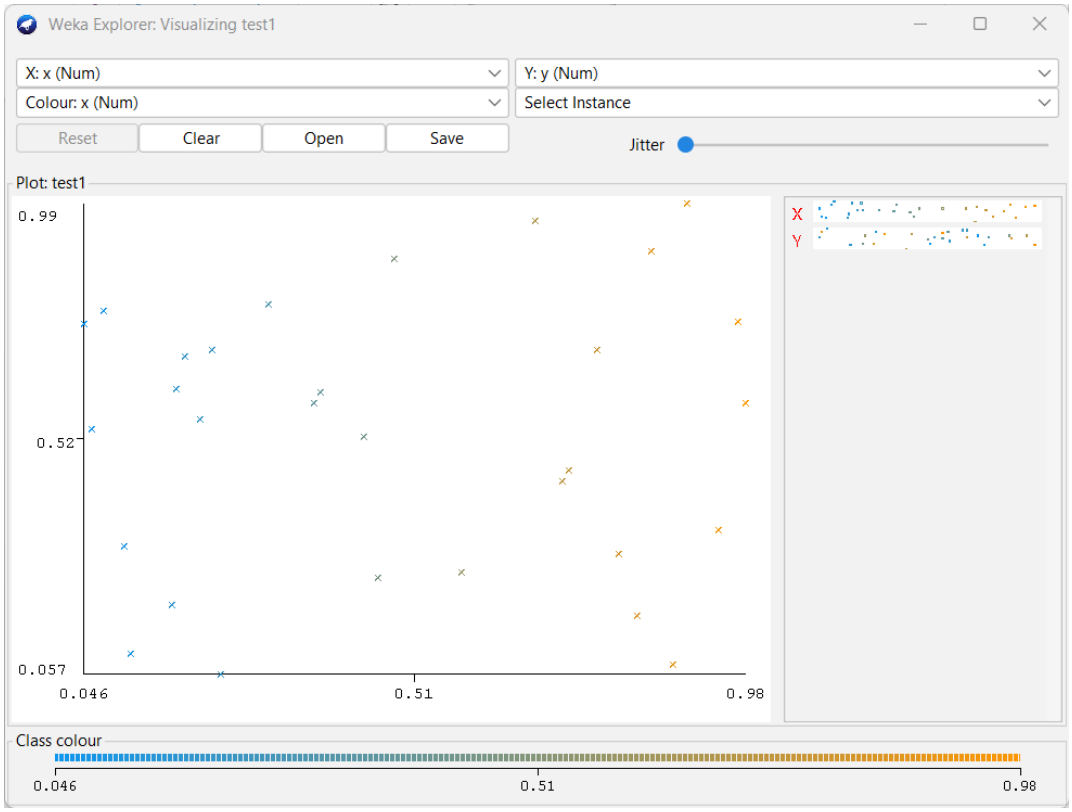
- Cụm 0: chứa 18 phần tử (60%)
- Cụm 1: chứa 6 phần tử (20%)
- Cụm 2: chứa 2 phần tử (7%)
- Cụm 3: chứa 4 phần tử (13%)

Bên cạnh đó thì dựa vào màn hình kết quả của WEKA sẽ còn hiển thị rằng phần tử nào thuộc về cụm nào.

```
Cluster 0
((((0.5947433:0.17217,0.755687:0.17217):0.02678,((0.9931196:0.11583,0.89639044:0.11583):0.07339,0.9580072:0.18922):0.00973):0.02159,0.70131344:0.22054):0.00713,(

Cluster 1
((0.34237665:0.1576,((0.074008524:0.1176,0.1713559:0.1176):0.01737,0.29518533:0.13496):0.02264):0.01753,(0.46006137:0.0263,0.43753022:0.0263):0.14884)

Cluster 2
(0.25767165:0.12768,0.24842167:0.12768)

Cluster 3
((0.056866586:0.14206,(0.19363922:0.12044,0.096954286:0.12044):0.02162):0.00044,0.30863827:0.1425)
```

Hình 26. Kết quả phân cụm

WEKA còn tích hợp thêm biểu đồ trực quan để dễ dàng hình dung kết quả phân cụm. Chọn vào tab Visualize và tiếp đến chọn và biểu đồ muốn xem.



Hình 27. Biểu đồ trực quan của WEKA

III.4.3. So sánh kết quả

Để có nhìn nhận dễ dàng hơn về hai kết quả mà ta đã đạt được tạm gọi kết quả từ chương trình được trình bày trong luận văn này là kết quả phân cụm A, còn kết quả thu được từ phân cụm phân cấp bởi phần mềm WEKA là kết quả phân cụm B.

Kết quả phân cụm A	Kết quả phân cụm B
Cụm 1: chứa 11 phần tử	Cụm 1: chứa 18 phần tử
Cụm 2: chứa 9 phần tử	Cụm 2: chứa 6 phần tử
Cụm 3: chứa 6 phần tử	Cụm 3: chứa 2 phần tử
Cụm 4: chứa 4 phần tử	Cụm 4: chứa 4 phần tử

Bảng 1. Kết quả phân cụm của 2 chương trình

Dựa vào *bảng 1* thì có thể thấy kết quả giữa 2 chương trình là có sự khác biệt. Điều dẫn đến sự khác này bắt nguồn từ các nguyên nhân sau:

- Thuật toán phân cụm phân cấp được thực hiện bằng cách duyệt các phần tử và chọn hai phần tử gần nhau nhất để gộp nhóm lại với nhau để tạo thành cụm lớn hơn tại mỗi vòng lặp. Trong thuật toán BIRCH lại sắp xếp các dữ liệu vào mô hình cây rồi sau đó mới gộp nhóm các cụm con.
- Trong quá trình xây dựng cây thuật toán BIRCH được trình bày trong đề tài có sử dụng thêm một tham số được ký hiệu là L (đã trình bày ở phần II.3). Bởi vì mỗi nút lá sẽ đại diện cho một cụm con, việc xây dựng cây CF với tham số này làm cho thuật toán trở nên linh hoạt và mỗi cụm con trong lúc xây dựng cây có thể chứa nhiều phần tử hơn hoặc ít hơn tùy thuộc vào giá trị mà người dùng truyền vào.

KẾT LUẬN

KẾT QUẢ ĐẠT ĐƯỢC

Sau những thời gian nghiên cứu, tìm hiểu về thuật toán phân cụm BIRCH thì em đã có thể tự mình xây dựng thuật toán phân cụm BIRCH mà không cần sử dụng đến thư viện viết sẵn, từ đó nâng cao được khả năng lập trình, tư duy và khả năng tự học tự của chính bản thân. Cũng qua đề tài này mà em thấy rằng đây là một chủ đề vô cùng thú vị và đem lại nhiều sự hiểu biết hơn về khai phá dữ liệu nói chung và phân cụm dữ liệu nói riêng. Phân cụm dữ liệu hiện nay đã được áp dụng rộng rãi trong nhiều lĩnh vực từ nghiên cứu khoa học, y tế, sinh học,… đến đời sống xã hội. Đến bây giờ công nghệ, truyền thông phát triển một cách "thần tốc", với những công nghệ lưu trữ vô cùng mạnh mẽ và khả năng lưu trữ lẫn tốc độ truy vấn dữ liệu, dữ liệu đạt đến quy mô "khổng lồ" là điều không hiếm gặp nửa, thì ngay lúc này phân cụm dữ liệu là một giải pháp không thể nào tốt hơn và cũng như là chìa khóa để tìm ra những tri thức từ dữ liệu. Việc được tìm hiểu, nghiên cứu và được trình bày chi tiết về thuật toán phân cụm dữ liệu BIRCH là một trong những bước chuyển mình về kiến thức của chính bản thân về khai phá dữ liệu.

HƯỚNG PHÁT TRIỂN

Mở rộng quy mô xử lý dữ liệu của thuật toán, từ tập dữ liệu ít chiều (ít thuộc tính) đến tập dữ liệu nhiều chiều (nhiều thuộc tính). Từ số lượng bản ghi ít đến số lượng bản ghi lớn. Tối ưu giao diện và các đoạn code để chương trình đạt hiệu suất tốt nhất. Thêm một số chức năng mới cho ứng dụng.

TÀI LIỆU THAM KHẢO

[1] TS. Nguyễn Đức Thuần, *Nhập môn Phát hiện tri thức và Khai phá dữ liệu*, NXB Thông tin và Truyền thông.

[2] Daniel T. Larose, Chantal D. Larose, *Data Mining and Predictive Analytics (ebook).*