ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

NGUYỄN VĂN DƯƠNG

SO SÁNH HAI PHƯƠNG PHÁP THU GỌN TẬP HUẤN LUYỆN RHC VÀ NAIVE RANKING TRONG PHÂN LỚP DỮ LỆU CHUỖI THỜI GIAN

ĐỀ CƯƠNG LUẬN VĂN THẠC SĨ

TP. HỒ CHÍ MINH - 2017

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

NGUYỄN VĂN DƯƠNG

SO SÁNH HAI PHƯƠNG PHÁP THU GỌN TẬP HUẤN LUYỆN RHC VÀ NAIVE RANKING TRONG PHÂN LỚP DỮ LỆU CHUỖI THỜI GIAN

CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH

MÃ SỐ CHUYÊN NGÀNH: 60.48.01

ĐỀ CƯƠNG LUẬN VĂN THẠC SĨ

PGS. TS. DƯƠNG TUẤN ANH

TP. HỒ CHÍ MINH - 2017

MỤC LỤC

Chương 1. GIỚI THIỆU ĐỀ TÀI **1**
1.1 LÝ DO NGHIÊN CỨU 1
1.2 MỤC ĐÍCH NGHIÊN CỨU 1
1.3 Ý NGHĨA ĐỀ TÀI 1
1.4 MÔ TẢ BÀI TOÁN 2

Chương 2. CƠ SỞ LÝ THUYẾT **3**
2.1 ĐẶC ĐIỂM CỦA DỮ LIỆU CHUỖI THỜI GIAN 3
2.1.1 Chuỗi con 3
2.1.2 Trùng khớp 4
2.2 ĐỘ ĐO KHOẢNG CÁCH 4
2.3 CÁC PHƯƠNG PHÁP PHÂN LỚP DỮ LIỆU CHUỖI THỜI GIAN . . . 7
2.3.1 Giải thuật phân lớp k-lân cận gần nhất (k-NN) 7
2.3.2 Cây quyết định 8

Chương 3. TỔNG QUAN CÁC CÔNG TRÌNH LIÊN QUAN **9**
3.1 KỸ THUẬT THU GỌN SỐ PHẦN TỬ TẬP HUẤN LUYỆN NAÏVE RANK 9
3.2 PHƯƠNG PHÁP THU GỌN TẬP HUẤN LUYỆN INSIGHT 11
3.2.1 Hàm tính trọng số 11
3.2.1.1 Thuộc tính trung tâm 11
3.2.1.2 Hàm tính trọng số dựa trên thuộc tính trung tâm 12
3.3 PHÂN LỚP k-NN HIỆU QUẢ VỚI KỸ THUẬT GỌN DỮ LIỆU DỰA VÀO GOM CỤM PHI THAM SỐ 13
3.3.1 Thu gọn dựa vào gom cụm (RHC) 13
3.3.2 RHC động (dRHC) 15
3.4 ÁP DỤNG CÁC GIẢI THUẬT LỰA CHỌN ĐẠI DIỆN VÀ TRÍCH YẾU ĐẠI DIỆN ĐỂ PHÂN LỚP DỮ LIỆU CHUỖI THỜI GIAN HIỆU QUẢ . 19
3.4.1 Các giải thuật lựa chọn đại diện (Prototype Selection Algorithms) . 19
3.4.1.1 Giải thuật CNN-rule 19
3.4.1.2 Giải thuật IB2 20
3.4.2 Các giải thuật trích yếu đại diện (Prototype Abstraction Algorithms) 21

3.4.2.1 Giải thuật trích yếu IB2 (AIB2) 21
3.4.2.2 Giải thuật CJA 22

Chương 4. GIẢI PHÁP **24**

Chương 5. NỘI DUNG VÀ KẾ HOẠCH NGHIÊN CỨU **25**
5.1 NỘI DUNG NGHIÊN CỨU 25
5.2 KẾ HOẠCH NGHIÊN CỨU 25

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 LÝ DO NGHIÊN CỨU

Khai phá dữ liệu (hay còn gọi là khám phá tri thức hoặc dữ liệu) đã trở thành lĩnh vực ngày càng quan trọng vì nó giúp phân tích và tổng kết dữ liệu thành những thông tin hữu ích. Có nhiều kỹ thuật trong quá trình khai thác dữ liệu trong đó *phân lớp* (classification) được biết đến như một kỹ thuật quan trọng. Mục đích của kỹ thuật này là phân tích dữ liệu thu thập được thành các mô hình mô tả nhằm phục vụ cho công việc dự đoán [5]. Ở đề tài này chúng tôi tiến hành thực hiện phân lớp trên *dữ liệu chuỗi thời gian* (time series data).

Bài toán phân lớp dữ liệu chuỗi thười gian là vấn đề luôn thu hút được sự quan tâm của giới nghiên cứu trong lĩnh vực khai phá dữ liệu chuỗi thời gian từ trước đến nay. Nó được ứng dụng trong nhiều lĩnh vực, y tế, tài chính, công nghiệp, giải trí. Đã có nhiều giải thuật được đưa ra để giải quyết bài toán này như: Cây quyết định, Mạng nơ-ron,...[12]

Hầu hết các giải thuật phân lớp đều đối mặt với vấn đề là chi phí tính toán và chi phí lưu trữ cao do phải xử lý trên tập dữ liệu lớn mà trong đó có nhiều dữ liệu không thật sự liên quan đến mục đích của bài toán. Do đó, thu gọn tập huấn luyện trước khi thực hiện phân lớp dữ liệu đã trở thành bước hết sức cần thiết. Đã có nhiều phương pháp thu gọn tập huấn luyện được giới thiệu, ở đây chúng tôi đề xuất áp dụng kỹ thuật *RHC thu gọn dữ liệu trước khi sử dụng k-NN để phân lớp tập dữ liệu chuỗi thời gian*.

1.2 MỤC ĐÍCH NGHIÊN CỨU

Nhằm giúp cho việc phân lớp dữ liệu chuỗi thời gian được giảm chi phí tính toán, hiệu quả hơn, tỉ lệ thu gọn dữ liệu đạt hiệu quả cao mà vẫn duy trì độ chính xác của kết quả phân lớp.

- Tìm hiểu và vận dụng giải thuật thu gọn tập huấn luyện RHC để thu gọn dữ liệu trước khi phân lớp bằng giải thuật k-NN.
- Hiện thực và thử nghiệm với bộ dữ liệu mẫu để đánh giá hiệu quả của các phương pháp đề xuất.

1.3 Ý NGHĨA ĐỀ TÀI

Đề tài đóng góp về ý nghĩa về mặt thực tiễn cũng như về mặt học thuật.

- Về mặt học thuật: đề tài đề xuất việc sử dụng được kỹ thuật thu gọn tập huấn luyện RHC vào bài toán phân lớp dữ liệu chuỗi thời gian bằng k-NN.

- Về mặt thực tiễn: góp phần nâng cao hiệu quả của khai phá dữ liệu chuỗi thời gian trong các lãnh vực môi trường, y tế, khoa học kỹ thuật, kinh tế, tài chính.

1.4 MÔ TẢ BÀI TOÁN

Thông tin đầu vào và đầu ra của bài toán được xác định như sau:

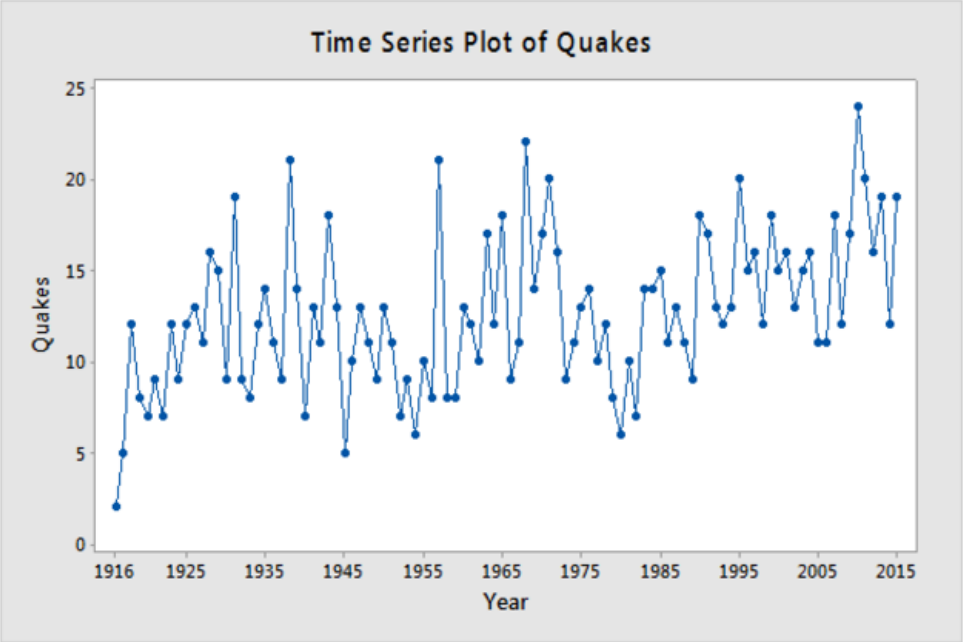
- Input: Tập dữ liệu chuỗi thời gian
- Các bước thực hiện:
  - Bước 1: Xử lý tập dữ liệu
  - Bước 2: Áp dụng kỹ thuật RHC, dRHC để thu gọn tập dữ liệu huấn luyện
  - Bước 3: Tiến hành phân lớp k-NN
  - Bước 4: So sánh, đánh giá kết quả phân lớp với giải thuật k-NN có áp dụng Naïve Ranking để thu gọn tập huấn luyện.
- Ouput:
  - Tập dữ liệu chuỗi thời gian đã được phân lớp
  - Các độ đo đánh giá chất lượng phân lớp của các phương pháp

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Phân lớp là kỹ thuật khai phá dữ liệu quan trọng. Nó cố gắng gán dữ liệu mới tới một lớp đã được xác định dựa trên bộ huấn luyện đã được phân loại sẵn. Một ví dụ phân lớp như gán một email vào lớp "spam" hay lớp "non-spam".

2.1 ĐẶC ĐIỂM CỦA DỮ LIỆU CHUỖI THỜI GIAN

Chuỗi thời gian (time series) là chuỗi trị số thực, mỗi trị số biểu diễn một giá trị đo tại những thời điểm cách đều nhau. Những tập dữ liệu chuỗi thời gian rất lớn xuất hiện trong nhiều lãnh vực khác nhau như y khoa, kỹ thuật, kinh tế, tài chính, v.v... [3]. Ví dụ kết quả quan sát hiện tượng động đất (đơn vị Richter) qua 99 năm liên tục là chuỗi thời gian như trong hình 2.1.

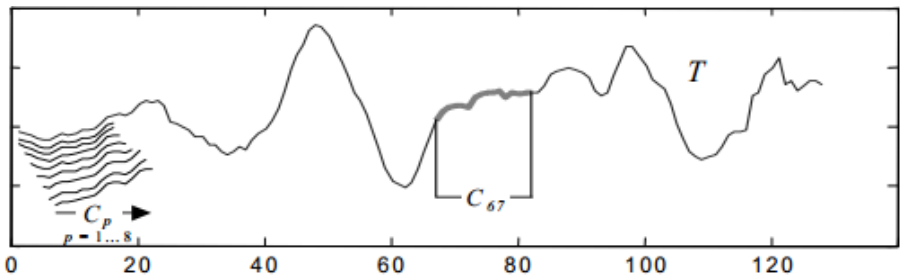


Hình 2.1: Ví dụ về chuỗi thời gian động đất

Ngoài ra, còn có các định nghĩa về *chuỗi thời gian đơn biến* (univariate time series) và *chuỗi thời gian đa biến* (multivariate time series). Chuỗi thời gian đơn biến là một chuỗi thời gian chỉ chứa một quan sát được ghi nhận một cách tuần tự tại những khoảng thời gian cách đều nhau. Chuỗi thời gian đa biến là chuỗi thời gian mà trong đó tại một thời điểm ta có nhiều quan sát (biến) khác nhau.

2.1.1 Chuỗi con

Cho một chuỗi thời gian T có chiều dài n, một *chuỗi con* (Subsequence) C của T là một dãy có chiều dài m ($1 \leq m \leq n$) có vị trí liền nhau trong chuỗi thời gian T.



Hình 2.2: Ví dụ về chuỗi con và cửa sổ trượt [6].

Một chuỗi con C của T cũng có thể được xem là một chuỗi thời gian với chiều dài m. Một điều cần lưu ý là khái niệm "chuỗi con" khác với khái niệm "chuỗi tuần tự". Nếu khái niệm "chuỗi tuần tự" cho phép các phần tử của chuỗi có thể không liên tục so với chuỗi ban đầu, thì khái niệm "chuỗi con C" của một chuỗi thời gian T chỉ chấp nhận những phần tử liên tiếp nhau trong chuỗi thời gian T. Bên dưới là một ví dụ về chuỗi con của một chuỗi thời gian.

Cho chuỗi thời gian $T = (3, 5, 1, 12, 4, 7)$. Khi đó, $C_1 = (12, 4, 7)$ được gọi là một chuỗi con của chuỗi thời gian T. Tuy nhiên, $C_2 = (3, 1, 12, 4)$ không được xem là một chuỗi con của chuỗi thời gian T, vì 3 và 1 là các giá trị không liên tiếp nhau trong T.

Các công trình nghiên cứu thường áp dụng phương pháp *cửa sổ trượt* (Sliding Windows) để lấy các chuỗi con trong một chuỗi thời gian để phục vụ cho bài toán nghiên cứu. Số lượng của các chuỗi con lấy được là bằng nhau và bằng độ dài của cửa sổ trượt. Hình 2.2 minh họa chuỗi con được xác định bằng phương pháp cửa sổ trượt:

2.1.2 Trùng khớp

Cho một số thực dương R (do người dùng định nghĩa) và một chuỗi thời gian T. Biết rằng T chứa một chuỗi con C bắt đầu tại thời điểm p và một chuỗi con M bắt đầu tại q, nếu khoảng cách D giữa 2 chuỗi nhỏ hơn hoặc bằng R, tức $D(C, M) \leq R$, thì M là một *chuỗi con trùng khớp* với C và ngược lại.

2.2 ĐỘ ĐO KHOẢNG CÁCH

Có nhiều loại độ đo khoảng cách để hỗ trợ tính toán trong quá trình khai phá dữ liệu. Nhưng đối với quá trình phân lớp với tập dữ liệu chuỗi thời gian thì thường sử dụng 2 loại độ đo khoảng cách chủ yếu là độ đo khoảng cách Euclid và và độ đo khoảng cách xoắn thời gian động (Dynamic Time Warping - DTW). Ở phần này chúng tôi tìm hiểu cách tính của 2 độ đo này trong dữ liệu chuỗi thời gian [4].

* ***Khoảng cách Euclid***

Cho 2 chuỗi thời gian $Q = Q_1, Q_2, \ldots, Q_n$ và $C = C_1, C_2, \ldots, C_n$. Khi đó khoảng cách Euclid giữa hai chuỗi thời gian Q và C được xác định bằng công thức sau:

$$D(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \tag{2.1}$$

Độ đo khoảng cách Euclid có ưu điểm là dễ hiểu, dễ tính toán, dễ mở rộng cho nhiều bài toán khai phá dữ liệu chuỗi thời gian khác như gom cụm, phân lớp, nhận dạng mô típ, v.v.. Nhưng độ đo khoảng cách này có nhược điểm là nhạy cảm với nhiễu và thiếu sự mềm dẻo khi so trùng [3].

* ***Độ đo xoắn thời gian động (Dynamic Time Warping - DTW)***

Trên nhiều nghiên cứu đã chứng minh DTW là độ đo xoắn thời gian động tính toán khoảng cách giữa 2 mẫu dữ liệu chuỗi thời gian cho độ chính xác nhiều hơn so với khoảng cách Euclid. Cho 2 chuỗi thời gian Q và C có chiều dài lần lượt là n và m theo với $Q = Q_1, Q_2, \ldots, Q_n$ và $C = C_1, C_2, \ldots, C_m$ công thức DTW được tính như sau:

- Bước 1: Xây dựng một ma trận có kích thước $n * m$ với n là số hàng của ma trận và m là số cột của ma trận. Trong đó phần tử (i^{th}, j^{th}) của ma trận chứa khoảng cách $d(q_i, c_j)$ giữa hai điểm q_i, c_j trên 2 chuỗi dữ liệu Q và C. Với $d(q_i, c_j) = (q_i - c_j)^2$.

- Bước 2: Ma trận chứa khoảng cách $d(q_i, c_j)$ là ma trận xoắn. Một đường xoắn (warping path) là một tập các phần tử liên tục của ma trận định nghĩa một đường ánh xạ giữa Q và C

  $W = w_1, w_2, \ldots, w_k$ với $max(m, n) \leq K \leq m + n - 1$

- Bước 3: Có rất nhiều đường xoắn thỏa mãn các điều kiện trên nhưng chúng ta chỉ quan tâm đến đường xoắn có chi phí tối thiểu

$$DTW(Q, C) = \min \left\{ \sqrt{\sum_{k=1}^{K} W_k} \right. \tag{2.2}$$

  Đối với việc tính toán DTW chúng ta có một số ràng buộc sau:

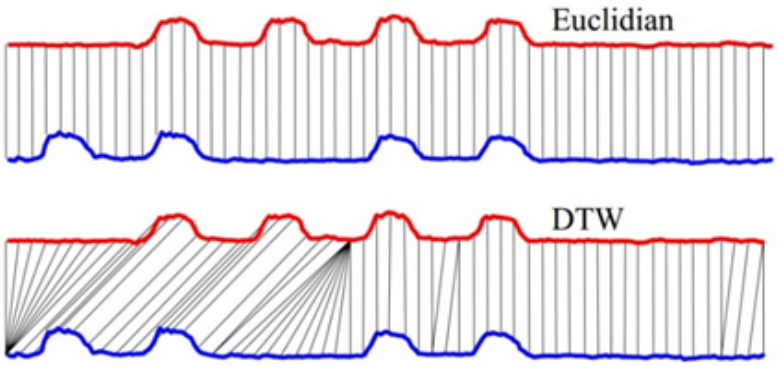
  – Điều kiện biên: $w_1 = (1, 1)$ và $w_k = (m, n)$ ràng buộc này yêu cầu đường xoắn phải bắt đầu và kết thúc ở hai góc đối diện của ma trận.

  – Tính liên tục: $w_k = (a, b)$ thì $w_{k-1} = (a', b')$ trong đó $a - a' \leq 1$ và $b - b' \leq 1$. Ràng buộc này yêu cầu đường xoắn phải di chuyển giữa những ô liền kề (kể cả những ô liền kề theo đường chéo).

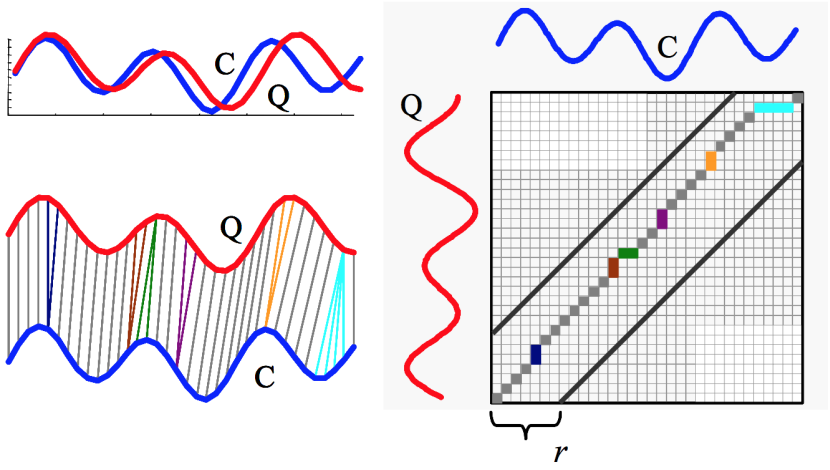
– Tính đơn điệu tăng: $w_k = (a, b)$ thì $w_{k-1} = (a', b')$, với $a - a' \geq 0$ và $b - b' \geq 0$. Ràng buộc này yêu cầu các điểm trong đường xoắn W phải có tính đơn điệu tăng theo thời gian.

Phương pháp DTW có ưu điểm là cho kết quả chính xác hơn so với độ đo Euclid và cho phép nhận dạng mẫu có hình dạng giống nhau nhưng chiều dài hình dạng về thời gian có thể khác nhau. Phương pháp này có nhược điểm là thời gian chạy lâu, tuy nhiên cũng đã có những công trình tăng tốc độ tìm kiếm tương tự dùng độ đo DTW.

Hình 2.3 mô tả sự khác biệt giữa độ đo Euclid và độ đo xoắn thời gian động.



Hình 2.3: Độ đo Euclid và độ đo xoắn thời gian động [4]



Hình 2.4: Ma trận xoắn thời gian động [12]

Hình 2.4 (bên trái) biểu diễn hai chuỗi thời gian Q và C tương tự nhau nhưng bị lệch giai đoạn (out of phase). Để sắp xếp lại hai chuỗi này ta xây dựng ma trận xoắn và tìm đường xoắn tối ưu (hình bên phải).

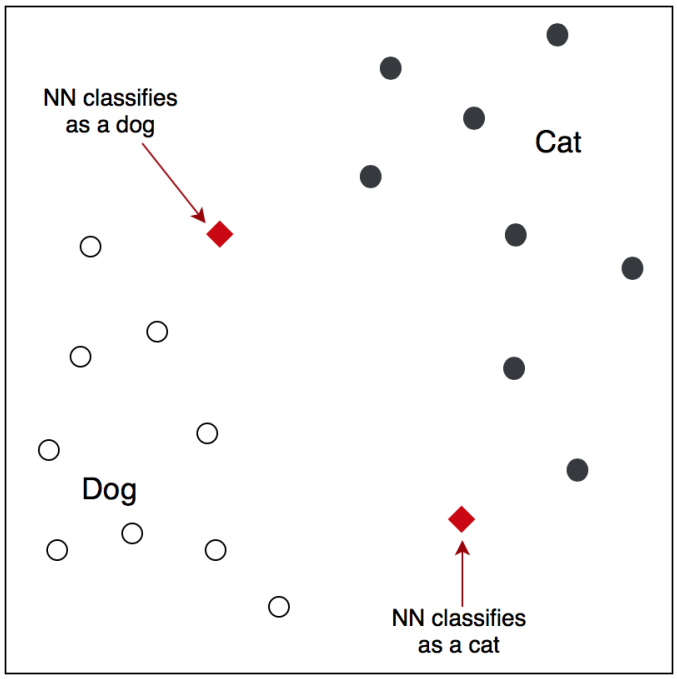
2.3 CÁC PHƯƠNG PHÁP PHÂN LỚP DỮ LIỆU CHUỖI THỜI GIAN

2.3.1 Giải thuật phân lớp k-lân cận gần nhất (k-NN)

k-NN là giải thuật được sử dụng rộng rãi trong nhiều bài toán phân lớp và dự báo. Quy trình phân lớp của k-NN với một đối tượng x_{new} được mô tả như sau:

- Tính khoảng cách giữa x_{new} và tất cả các mẫu trong tập huấn luyện
- Chọn k mẫu gần nhất với x_{new} trong tập huấn luyện
- Gán x_{new} vào lớp có nhiều mẫu nhất trong số k mẫu lân cận đó (hoặc x_{new} nhận giá trị trung bình của k mẫu)

  Hình 2.5 minh họa ví dụ phân lớp đối tượng mới vào lớp Dog hoặc Cat dựa vào k-lân cận gần nhất với trường hợp $k = 1$



Hình 2.5: Phân lớp k-lân cận gần nhất

Ưu điểm:

- Dễ sử dụng và cài đặt
- Xử lý tốt với dữ liệu nhiễu (do dựa trên khoảng cách để quyết định phân lớp)

Khuyết điểm:

- Cần lưu tất cả các mẫu để phân lớp

- Cần nhiều thời gian để xác định lớp cho một mẫu mới (cần tính và so sánh khoảng cách đến tất cả các mẫu huấn luyện)
- Phụ thuộc vào giá trị k do người dùng lựa chọn. Nếu k quá nhỏ, nhạy cảm với nhiễu. Nếu k quá lớn thì vùng lân cận có thể chứa các điểm của lớp khác.
- Cần chuyển đổi kiểu dữ liệu đối với thuộc tính rời rạc (categorical)

2.3.2 Cây quyết định

Cây quyết định là phương pháp phân lớp dưới dạng cấu trúc cây mà nó thực hiện kiểm tra phân tách tại mỗi nút nội và dự đoán phân lớp cho nút lá, nhánh từ một node nội là kết quả của một phép thử trên thuộc tính tương ứng. Đã có nhiều giải thuật khác nhau xây dựng cây quyết định áp dụng thành công trên nhiều lĩnh vực [11]. Đặc điểm của các giải thuật này là:

- Giải thuật tham lam (không có quay lui), chia để trị, đệ qui, từ trên xuống.
- Độ phức tạp với tập huấn luyện D gồm $|D|$ phần tử (đối tượng), mỗi phần tử gồm n thuộc tính $O(n * |D| * log|D|)$.
  - Mỗi thuộc tính ứng với mỗi cấp (level) của cây.
  - Cho mỗi cấp của cây, $|D|$ phần tử huấn luyện được duyệt qua.
- Một bước quan trọng trong giải thuật là lựa chọn tiêu chí rẽ nhánh nghĩa là phân hoạch tập huấn luyện D thành các phân hoạch con với các nhãn phù hợp.
  - Xếp hạng mỗi thuộc tính
  - Thuộc tính được chọn để rẽ nhánh là thuộc tính có trị số điểm (score) lớn nhất
  - Độ đo chọn thuộc tính phân tách (splitting attribute): độ lợi thông tin, chỉ số Gini

Do dữ liệu chuỗi thời gian có những đặc điểm riêng (khối lượng dữ liệu lớn, giữa các điểm dữ liệu có mối tương quan với nhau và dữ liệu có thể có nhiễu) làm cho việc khai phá dữ liệu chuỗi thời gian trở nên khó khăn hơn. Nhiều giải thuật phân lớp kinh điển làm việc tốt với dữ liệu thường nhưng không thể làm việc hiệu quả với dữ liệu chuỗi thời gian. Qua các nghiên cứu đã cho thấy giải thuật phân lớp 1-NN sử dụng độ đo DTW phù hợp, có thể nói là tốt nhất với loại dữ liệu này (theo Wang và các cộng sự [13]).

CHƯƠNG 3. TỔNG QUAN CÁC CÔNG TRÌNH LIÊN QUAN

Thu gọn dữ liệu chuỗi thời gian trở thành một bước tất yếu để tăng tốc quá trình phân lớp. Nhiều công trình nghiên cứu đã cho ra đời những kỹ thuật thu gọn tập huấn luyện hiệu quả. Trong chương này chúng ta sẽ điểm qua một kỹ thuật thu gọn dữ liệu chuỗi thời gian tiên tiến hiện nay.

3.1 KỸ THUẬT THU GỌN SỐ PHẦN TỬ TẬP HUẤN LUYỆN NAÏVE RANK

Có nhiều giải thuật đề xuất cho vấn đề phân lớp với dữ liệu là chuỗi thời gian. Tuy nhiên, rõ ràng giải thuật một phần tử lân cận gần nhất (1-NN) với độ khoảng cách xoắn thời gian động (Dynamic Time Warping - DTW) là giải thuật đặc biệt khó có giải thuật nào đánh bại được [13]. Cách tiếp cận này có một điểm yếu, đòi hỏi tính toán nhiều trong thời gian thực. Có một cách khắc phục điểm yếu này là tăng tốc độ tính toán độ đo DTW. Xi và các cộng sự (2006) đã đưa ra kỹ thuật thu gọn số lượng (numerosity reduction) tập huấn luyện để tăng tốc độ giải thuật phân lớp 1-NN mà không làm mất tính chính xác phân lớp.

Kỹ thuật thu gọn số lượng tập huấn luyện có thể hiểu đơn giản là loại bỏ một số dữ liệu trong tập huấn luyện nhằm để cải thiện hiệu suất phân lớp. Nếu chúng ta cẩn thận lựa chọn đối tượng loại bỏ, chúng ta đặc biệt rút ngắn được thời gian phân lớp trong khi độ chính xác vẫn cao, trong một vài trường hợp tăng hiệu suất phân lớp một cách rất đáng kể.

Việc tăng tốc phân lớp dựa vào nền tảng kiến thức giải thuật k-NN và độ đo DTW đã trình bày ở Chương 2. Và quan trọng nhất là kỹ thuật thu gọn tập huấn luyện Naïve Rank (Naïve Rank Reduction) [12]. Giải thuật Naïve Rank thu gọn tập huấn luyện gồm 2 bước là xếp hạng (ranking) và xác định ngưỡng (thresholding). Đầu tiên giải thuật xếp hạng cho tất cả các đối tượng trong tập huấn luyện. Sau đó giải thuật xác định ngưỡng n để quyết định có bao nhiêu đối tượng được giữ lại và n đối tượng xếp hạng cao nhất được giữ lại để phân lớp.

Hạng của các đối tượng được xác định theo công thức 3.1:

$$rank(x) = \sum_j \begin{cases} 1 & \text{if class}(x) = \text{class}(x_j) \\ -2 & \text{otherwise} \end{cases} \tag{3.1}$$

Với x_j là đối tượng có x là lân cận gần nhất.

Trường hợp 2 đối tượng có xếp hạng như nhau, thì sẽ dựa vào độ ưu tiên của chúng để phân biệt. Công thức tính độ ưu tiên của x như sau:

$$priority(x) = \sum_j \frac{1}{d(x, x_j)^2} \tag{3.2}$$

Với x_j là đối tượng có x là lân cận gần nhất và $d(x, x_j)$ là khoảng cách giữa x và x_j. Nếu đối tượng ở cách xa lân cận gần nhất thì nó có thể là nhiễu hoặc không là đối tượng thuộc lớp thì nó được loại bỏ đi. Giữa 2 đối tượng có cùng xếp hạng nhưng đối tượng nào có độ ưu tiên thấp hơn sẽ bị loại bỏ đầu tiên.

Giải thuật xếp hạng có tính lặp. Giả sử tập huấn luyện có kích thước N, giải thuật sẽ chạy N lần, tại mỗi lần lặp nó sẽ loại bỏ đối tượng có xếp hạng thấp và tiếp tục xếp hạng lại các đối tượng.

Algorithm 1. Naïve Rank Reduction

Input: T
Output: S

Remove any duplicate instances from T
leave -one-out 1-NN classification on T
$N \leftarrow$ size of T
$S \leftarrow \varnothing$
$loop\_num \leftarrow 0$
while $loop\_num < N$ **do**
    for each instance x_i in T - S **do**
        rank(x_i) is calculated using eq. 3.1
    end for
    adjust the rank ties by priorities using eq. 3.2
    $worst\_index = arg_i(min(rank[x_i] \& min(priority[x_i])))$
    push(S, x_{worst_index}) *// discard instance with lowest rank*
    $loop\_num \leftarrow loop\_num + 1$
end while
return S

Trong bước xác định ngưỡng, n phần tử có hạng cao nhất được giữ lại trong S như là tập huấn luyện để làm đầu vào cho giải thuật phân lớp (n là tham số đầu vào được xác định bởi người dùng). Việc xác định ngưỡng có thể được dựa trên không gian hoặc thời gian tối đa mà người dùng có cho việc giải quyết bài toán. Ví dụ, việc phân lớp côn trùng với tài nguyên hạn chế của các cảm biến (sensors), người dùng chỉ có tối đa 200k bộ nhớ để lưu toàn bộ tập huấn luyện (Wei & Keogh, 2005). Ngoài ra còn có một số cách xác định ngưỡng khác như *"cho tập dữ liệu huấn luyện nhỏ nhất với một tỉ lệ lỗi mong đợi ít hơn 5%"* hoặc *"cho tập dữ liệu huấn luyện nhỏ nhất với tỉ lệ lỗi leave-one-out không khác với tỉ lệ lỗi trên toàn bộ tập dữ liệu quá 1%"*.

Algorithm 1 trình bày mã giả của giải thuật thu gọn dữ liệu Naïve Rank, nhận vào tập huấn luyện T và trả ra tập thu gọn S. Giải thuật bắt đầu bằng việc loại bỏ các phần tử trùng lặp trong tập huấn luyện T (dòng 1). Sau đó tiến hành phân lớp sử dụng giải thuật 1-NN trên tập huấn luyện T (dòng 2). Mỗi phần tử trong tập huấn luyện được xác định thứ hạng sử dụng công thức 3.1 (dòng 7-9). Nếu thứ hạng các phần tử bị trùng nhau, thứ hạng sẽ được điều chỉnh bằng công thức 3.2 (dòng 10). Những phần tử có thứ hạng nhỏ nhất sẽ không được đưa vào tập kết quả S (dòng 12). Giải thuật dừng khi đã duyệt qua tất cả các phần tử trong tập huấn luyện T.

3.2 PHƯƠNG PHÁP THU GỌN TẬP HUẤN LUYỆN INSIGHT

Trong công trình này, Buza và các cộng sự [1] đề xuất một phương pháp lựa chọn phần tử mới mà nó khai thác khái niệm của tính trung tâm (hubness), là một vài phần tử có xu hướng là lân cận gần nhất rất thường xuyên hơn những phần tử còn lại. Dựa trên tính trung tâm, họ đề xuất một khung sườn cho việc lựa chọn phần tử dựa trên điểm (score), nó được kết hợp với nguyên tắc tối ưu hóa độ bao phủ (coverage) của dữ liệu huấn luyện. Nghĩa là một chuỗi thời gian x bao phủ một chuỗi thời gian y, nếu y có thể được phân lớp đúng đắn dự trên x. Phương pháp này không chỉ cho phép hiểu tốt hơn bài toán lựa chọn phần tử mà còn giúp cho việc phân tích các đặc điểm của hướng tiếp cận từ quan điểm của việc tối đa hóa sự bao phủ. Với những lý do trên, hướng tiếp cận này được gọi là "Lựa chọn phần tử dựa trên sự bao phủ đồ thị và tính trung tâm của chuỗi thời gian" (Instance Selection based on Graph-coverage and Hubness for Time-series), được viết tắt là INSIGHT.

INSIGHT được đánh giá thực nghiệm với 37 tập dữ liệu phân lớp chuỗi thời gian và được so sánh với FastAWARD, một phương pháp lựa chọn phần tử tiên tiến cho dữ liệu chỗi thời gian. INSIGHT được đánh giá là tốt hơn FastAWARD một cách đáng kể về cả độ chính xác phân lớp cũng như thời gian thực thi việc lựa chọn phần tử [1].

3.2.1 Hàm tính trọng số

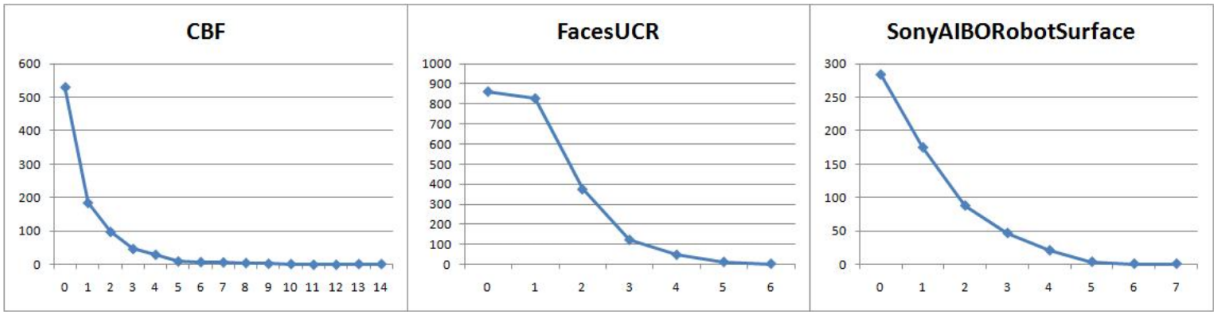
3.2.1.1 Thuộc tính trung tâm

Để xây dựng *hàm trọng số* (score function) phục vụ việc lựa chọn phần tử đại diện cho phân lớp chuỗi thời gian bằng phương pháp lân cận gần nhất, ta phải kể đến phát hiện gần đây về thuộc tính trung tâm. Thuộc tính này chỉ ra rằng với dữ liệu cao chiều như hầu hết dữ liệu chuỗi thời gian, một vài đối tượng có xu hướng trở thành lân cận gần nhất thường xuyên hơn những đối tượng khác. Để diễn tả trung tâm một cách chính xác, cho một tập dữ liệu $\mathscr{D}$ chúng ta định nghĩa lần xuất hiện như là lân cận thứ k ($k-occurrence$)

của một phần tử $x \in \mathscr{D}$, ký hiệu $f_N^k(x)$ là số phần tử của $\mathscr{D}$ có x ở giữa k lân cận gần nhất. Với thuật ngữ *trung tâm* (hubness) đề cập tới hiện tượng mà phân bố của $f_N^k(x)$ lệch đáng kể về bên phải. Độ lệch được ký hiệu: $\mathscr{S}_{f_N^k(x)}$

$$\mathscr{S}_{f_N^k(x)} = \frac{E[(f_N^k(x) - \mu_{f_N^k(x)})^3]}{\sigma^3_{f_N^k(x)}} \tag{3.3}$$

Trong đó $\mu_{f_N^k(x)}$ và $\sigma_{f_N^k(x)}$ là trung vị và độ lệch chuẩn của $f_N^k(x)$. Khi $\mathscr{S}_{f_N^k(x)}$ lớn hơn một thì phân bố tương ứng lệch về bên phải tạo thành đuôi dài. Với dữ liệu có nhãn, trung tâm tốt và trung tâm xấu được phân biệt như sau: phần tử y là một *k-lân cận gần nhất* tốt (xấu) của phần tử x nếu (i) y là một trong k lân cận gần nhất của x, và (ii) chúng có cùng (khác) nhãn. Điều này cho phép định nghĩa *k-occurence* tốt (xấu) của một chuỗi thời gian x, tương ứng là $f_G^k(x)$ (và $f_B^k(x)$) là số lượng những chuỗi thời gian khác mà có x là một trong những *k-lân cận gần nhất* tốt (xấu) của chúng.



Hình 3.1: Biểu đồ phân bố của hàm $f_G^1(x)$ trên một số chuỗi thời gian. Trục hoành thể hiện giá trị $f_G^1(x)$, trục tung là số lượng phần tử có giá trị đó.

Đối với dữ liệu chuỗi thời gian, cả hai phân bố $f_G^k(x)$ và $f_B^k(x)$ đều thường lệch về bên phải. Ví dụ ở hình 3.1 mô tả sự phân bố của hàm $f_G^1(x)$ trên một số chuỗi thời gian, biểu đồ phân bố tạo thành đuôi dài trong trường hợp trung tâm tốt (good hubs) xuất hiện.

Một chuỗi thời gian x là một trung tâm tốt (xấu), nếu phân bố $f_G^k(x)$ ($f_B^k(x)$) rộng một cách đặt biệt với x. Đối với việc phân lớp chuỗi dữ liệu thời gian bằng phương pháp lân cận gần nhất thì việc biểu đồ phân bố lệch của các thể hiện tốt là rất quan trọng, bởi vì một số chuỗi thời gian có thể giúp phân lớp các chuỗi thời gian khác một cách đúng đắn. Vì thế, đây là bằng chứng quan trọng để ta nên đặc biệt chú ý vào các trung tâm tốt.

3.2.1.2 Hàm tính trọng số dựa trên thuộc tính trung tâm

Dựa vào **Good 1-occurence score** đã tìm thấy trước đó - giải thuật INSIGHT sử dụng trọng số của phần tử là 1-xuất hiện tốt. Công thức tính trọng số 1-xuất hiện tốt như sau:

$$f_G(x) = f_G^1(x) \tag{3.4}$$

Trọng số tương đối (Relative Score) $f_R(x)$ của một chuỗi thời gian x là phân số của 1-xuất hiện tốt và tổng xuất hiện cộng với một (để tránh trường hợp mẫu số là 0):

$$f_R(x) = \frac{f_G^1(x)}{f_N^1(x) + 1} \tag{3.5}$$

Trọng số Xi (Xi's score), dựa vào $f_G^k(x)$ và $f_B^k(x)$ cho phép chúng ta giải thích tiêu chí xếp hạng trong công trình của Xi bằng cách diễn tả chúng như là một dạng khác của trọng số tương đối.

$$f_{Xi}(x) = f_G^1(x) - 2f_B^1(x) \tag{3.6}$$

Giải thuật INSIGHT lựa chọn những phần tử có thứ hạng cao (top-ranked). Tuy nhiên, trong khi xếp hạng các phần tử, chúng ta cần xem xét sự tương quan giữa chúng. Ví dụ, giải sử rằng phần tử có thứ hạng cao đầu tiên cho phép độ chính xác phân lớp 1-NN hầu như giống với phần tử có thứ hạng cao thứ hai. Vì thế sự đóng góp của phần tử có thứ hạng cao thứ 2 là không quan trọng trong tổng thể quá trình phân lớp. Giải thuật INSIGHT có 3 bước cơ bản được mô tả trong đoạn mã giả Algorithm 2 sau:

- Bước 1: Tính trọng số cho tất cả các phần tử x trong D.
- Bước 2: Sắp xếp tất cả các chuỗi thời gian trong D theo trọng số của chúng.
- Bước 3: Chọn N chuỗi thời gian có thứ hạng cao đưa vào tập hợp kết quả.

Algorithm 2. INSIGHT

Require: Time-series dataset D, Score Function f , Number of selected instances N
Ensure: Set of selected instances (time series) D

1: Calculate score function $f(x)$ for all $x \in D$
2: Sort all the time series in D according to their scores $f(x)$
3: Select the top-ranked N time series and return the set containing them

3.3 PHÂN LỚP K-NN HIỆU QUẢ VỚI KỸ THUẬT GỌN DỮ LIỆU DỰA VÀO GOM CỤM PHI THAM SỐ

3.3.1 Thu gọn dựa vào gom cụm (RHC)

RHC là giải thuật *trích yếu đại diện* (prototype abstraction) phi tham số. Nó dựa vào ý tưởng đơn giản là áp dụng đệ quy vào giải thuật gom cụm k-means. Đặc biệt RHC xây dựng cụm cho đến khi chúng thuần nhất nghĩa là tất cả đối tượng chỉ thuộc về cùng một lớp. Ban đầu RHC xem toàn bộ tập huấn luyện là *không thuần nhất* (non-homogeneous). Giải thuật bắt đầu tìm mean (centroid - trung tâm cụm) của mỗi lớp bằng cách tính toán

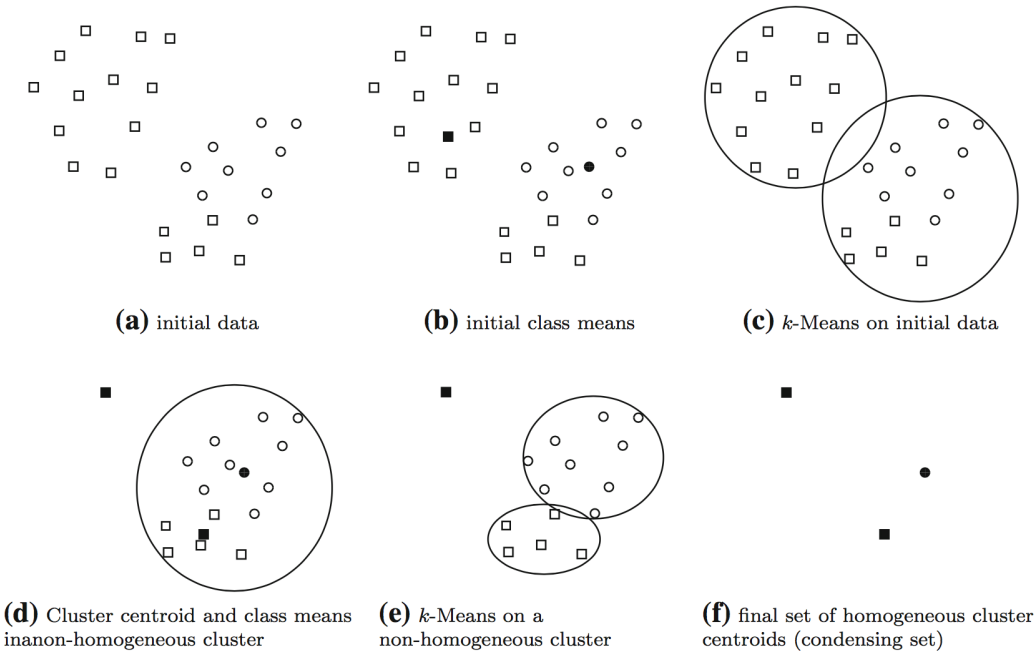
trung bình các giá trị thuộc tính của các mẫu tương ứng của tập huấn luyện. Vì vậy một tập dữ liệu có n lớp, giải thuật sẽ tính n centroid. RHC thực thi giải thuật gom cụm k-means sử dụng n centroid ban đầu và xây dựng các cụm. Đối với mỗi cụm thuần nhất centroid thuộc về *tập ngưng tụ* (condensing set), còn cụm chưa thuần nhất tiếp tục thực hiện đệ quy lại thủ tục trên. Giải thuật ngừng khi tất cả các cụm đã thuần nhất. Khi kết thúc, tập ngưng tụ gồm tất cả các centroid của các cụm đã thuần nhất. Chú ý sử dụng mean của các lớp như là các centroid ban đầu cho giải thuật k-means, số lượng cụm sẽ được xác định tự động. Phần tử trung bình m của một cụm hoặc lớp C tính bằng cách lấy trung bình giá trị của n thuộc tính của các phần tử $x_j, j = 1, 2, ..., |C|$ trong C. Cụ thể, thuộc tính thứ j của phần tử trung bình m được xác định bởi công thức 3.7 sau đây:

$$m.d_j = \frac{1}{|C|} \sum_{x_i \in C} x_i.d_j, j = 1, 2, ..., n \tag{3.7}$$

Với

- m: một đối tượng nào đó trong cụm hoặc trong lớp tương ứng $x_i, i = 1, 2...|C|$
- n: là số thuộc tính.

Hình 3.2 mô tả trực quan qui trình thực hiện giải thuật RHC.



Hình 3.2: Quy trình thực hiện của giải thuật RHC [8].

Algorithm 3 trình bày mã giả của giải thuật RHC. Nó tận dụng cấu trúc dữ liệu hàng đợi, $Queue$ để lưu các cụm. Đầu tiên, giải thuật RHC xem toàn bộ tập huấn luyện TS như là một cụm chưa được xử lý. Do đó nó được đặt vào hàng đợi $Queue$ (dòng 3). Ở mỗi

Algorithm 3. RHC

Input: TS
Output: CS

1: **{Stage 1: Queue Initialization}**
2: $Queue \leftarrow \varnothing$
3: $Enqueue(Queue, TS)$
4: **{Stage 2: Construction of the condensing set}**
5: $CS \leftarrow \varnothing$
6: **repeat**
7: $C \leftarrow$ Dequeue($Queue$)
8: **if** C is homogenious **then**
9: $r \leftarrow$ mean of C
10: $CS \leftarrow CS \cup \{r\}$
11: **else**
12: $M \leftarrow \varnothing$ {M is the set of class means}
13: **for** each class L in C **do**
14: $m_L \leftarrow$ mean of L
15: $M \leftarrow M \cup \{m_L\}$
16: **end for**
17: $NewClusters \leftarrow$ K-Means(C, M)
18: **for** each cluster $NC \in NewClusters$ **do**
19: $Enqueue(Queue, NC)$
20: **end for**
21: **end if**
22: **until** IsEmpty($Queue$)
23: **return** CS

bước lặp, giải thuật RHC xét phần tử C đầu hàng đợi $Queue$ (dòng 7). Sau đó nó kiểm tra xem C có phải là cụm thuần nhất hay không. Nếu C là cụm thuần nhất (dòng 8) thì phần tử trung bình r của C được thêm vào tập ngưng tụ (dòng 10). Ngược lại nếu C là tập không thuần nhất thì giải thuật RHC tính các trung bình lớp M: một trung bình lớp cho mỗi lớp trong C (dòng 13-16). Sau đó, giải thuật RHC sử dụng giải thuật gom cụm k-means với tham số đầu vào là C và M, tạo ra một tập hợp các cụm mới (dòng 17) và thêm vào hàng đợi $Queue$ (dòng 18-20). Giải thuật dừng lại khi không còn phần tử nào trong hàng đợi $Queue$ (dòng 22) hay nói cách khác là khi tất cả các cụm đều thuần nhất.

3.3.2 RHC động (dRHC)

Giống với hầu hết các kỹ thuật thu gọn dữ liệu, RHC là kỹ thuật dựa vào bộ nhớ, nghĩa là toàn bộ tập dữ liệu thuộc về bộ nhớ. RHC không thể quản lý hết tập dữ liệu vì không thể lưu hết dữ liệu vào bộ nhớ chính. Do vậy, nó không thể thực thi trên thiết bị bị giới hạn về bộ nhớ và cũng không thể chuyển dữ liệu qua các máy khác xử lý thông qua mạng máy tính. Vì đây là thủ tục tốn nhiều thời gian và chi phí. Thêm vào đó, RHC không thể

xử lý khi có những mẫu dữ liệu mới thêm vào. Giả sử RHC được thực thi trên tập dữ liệu D và xây dựng tập ngưng tụ. Sau đó giả sử một tập dữ liệu mới S được đưa thêm vào xem xét. Để xây dựng cập nhật lại tập ngưng tụ, RHC phải thực thi lại từ đầu trên tập $D \cup S$. Thủ tục phải thực hiện lặp lại bất cứ khi nào có tập dữ liệu mới thêm vào, lúc này nhu cầu lưu trữ và chi phí tính toán cao.

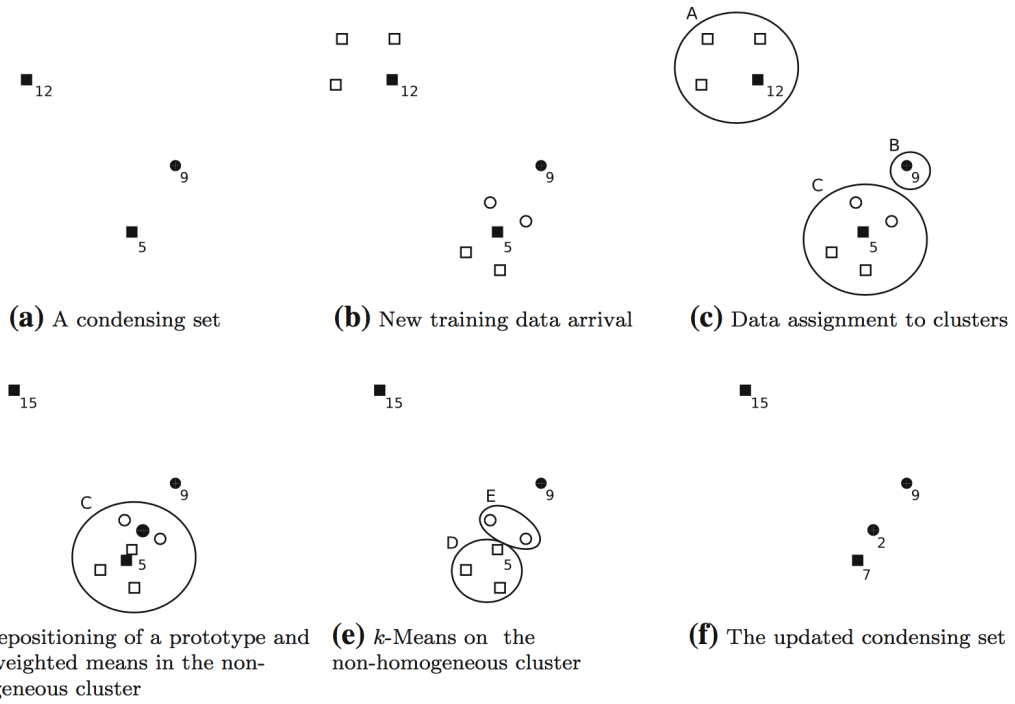
Giải thuật dRHC là phiên bản động của RHC, nó kế thừa lại tất cả ưu điểm của RHC. Nhưng nó khắc phục được điểm yếu của RHC, nếu tập dữ liệu không phù hợp với bộ nhớ chính, nó sẽ chia tập dữ liệu thành các "phân đoạn dữ liệu" (data segments) để thích hợp với kích thước bộ nhớ chính. Ngoài ra đối với môi trường động khi dữ liệu đến liên tục, nó xem dữ liệu này như các "phân đoạn dữ liệu". Trong trường hợp này, khái niệm phân đoạn dữ liệu được thực hiện sử dụng bộ đệm (buffer), nơi tập dữ liệu mới được lưu trữ. Khi bộ đệm đầy, giải thuật dRHC bắt đầu thực thi trên chúng. Sau đó tập dữ liệu được lưu trữ trong bộ đệm bị xóa đi và bộ đệm sẵn sàng nhận tập dữ liệu mới. dRHC thực hiện 2 giai đoạn: (1) xây dựng tập ngưng tụ ban đầu (CS-consending set), (2) cập nhật tập ngưng tụ mỗi lần một phân đoạn dữ liệu đến. Mọi thủ tục đều tương tự RHC, điểm khác biệt duy nhất giữa RHC và dRHC là mỗi lần sinh ra phần tử đại diện (prototype) lưu trữ giá trị trọng số như một thuộc tính bổ sung. Giá trị này là số lượng tập huấn luyện mà được gom lại với nhau và được đại diện bởi prototype đặc biệt trong tập ngưng tụ. Cập nhật tập ngưng tụ CS chỉ dựa vào khái niệm cụm thuần nhất và các trọng số dữ liệu. Công thức 3.8 dưới đây là công thức tìm trung tâm cụm của dRHC:

$$m_C.d_j = \frac{\sum\limits_{x_i \in C} x_i.d_j \times x_i.weight}{\sum\limits_{x_i \in C} x_i.weight} \tag{3.8}$$

- m: đối tượng nào đó trong cụm hoặc trong lớp tương ứng $x_i, i = 1, 2...|C|$
- d_j là mỗi vector thuộc tính của m_C, với $j = 1, 2, ...n$

Quy trình thực hiện giải thuật dRHC được minh hoạ trực quan thông qua hình vẽ 3.3.

Algorithm 4 trình bày mã giả của giai đoạn cập nhật tập ngưng tụ CS của giải thuật dRHC. Nó nhận vào tập ngưng tụ sẵn có $oldCS$ cùng với một phân đoạn dữ liệu mới $dataSeg$ và trả ra tập ngưng tụ đã được cập nhật $newCS$. Giải thuật bắt đầu bằng việc xây dựng hàng đợi $Queue$ để chứa những cụm chưa được xử lý. Đầu tiên, nó khởi tạo một số lượng các cụm $CList$ bằng với số lượng phần tử trong tập ngưng tụ sẵn có $oldCS$ (dòng 3-6). Tiếp theo, mỗi phần tử trong phân đoạn dữ liệu $dataSeg$ sẽ được gán trọng số là 1 và thêm vào một trong các cụm này (dòng 7-11). Tất cả các cụm trong $CList$ được đặt vào hàng đợi $Queue$ (dòng 12-14). Giải thuật dRHC tạo ra tập ngưng tụ mới $newCS$



(a) A condensing set **(b)** New training data arrival **(c)** Data assignment to clusters

(d) Repositioning of a prototype and class weighted means in the non-homogeneous cluster **(e)** k-Means on the non-homogeneous cluster **(f)** The updated condensing set

Hình 3.3: Quy trình thực hiện giải thuật dRHC [8]

bằng cách tương tự như RHC nhưng có xem xét tới giá trị trọng số. Đối với cụm thuần nhất thì phần tử đại diện được lưu trong $newCS$ là phần tử trung tâm cụm (dòng 19-22). Đối với cụm không thuần nhất C, mỗi trung bình lớp được được đánh trọng số là tổng trọng số của các phần tử của lớp (dòng 24-29). Những phần tử trung bình lớp này đóng vai trò là những phần tử trung bình khởi đầu cho giải thuật k-means (dòng 30). Giải thuật dRHC sử dụng giải thuật gom cụm k-means có quan tâm tới trọng số khi xác định trung tâm cụm. Cho một cụm C, mỗi thuộc tính $d_j, j = 1, 2, .., n$ của trung tâm cụm m_c (dòng 20, 26) được tính dựa vào công thức (3.8).

Algorithm 4. dRHC: CS update phase

Input: $oldCS, dataSeg$
Output: $newCS$

{Stage 1: Queue Initialization}
$Queue \leftarrow \varnothing$
$CList \leftarrow \varnothing$ {empty list of clusters}
for each prototype $m \in oldCS$ **do**
    add new cluster $C = \{m\}$ in $CList$
end for
for each item $x \in dataSeg$ **do**
    $x.weight = 1$
    Find $C_x \in CList$ with the nearest to x mean
    $C_x \leftarrow C_x \cup \{x\}$ {do not recompute mean of C_x}
end for
for each cluster C in $CList$ **do**
    Enqueue($Queue, C$)
end for
{Stage 2: Construction of newCS}
$newCS \leftarrow \varnothing$
repeat
    $C \leftarrow$ Dequeue($Queue$)
    if C is homogenious **then**
        $m \leftarrow$ weighted mean of C
        $m.weight \leftarrow \sum_{x_i \in C} x_i.weight$
        $newCS \leftarrow newCS \cup \{m\}$
    else
        $M \leftarrow \varnothing$ {M is the set of weighted class means}
        for each class L in C **do**
            $m_L \leftarrow$ mean of L
            $m_L.weight \leftarrow \sum_{x_i \in L} x_i.weight$
            $M \leftarrow M \cup \{m_L\}$
        end for
        $NewClusters \leftarrow$ K-Means(C, M)
        for each cluster $NC \in NewClusters$ **do**
            $Enqueue(Queue, NC)$
        end for
    end if
until IsEmpty($Queue$)
return $newCS$

3.4 ÁP DỤNG CÁC GIẢI THUẬT LỰA CHỌN ĐẠI DIỆN VÀ TRÍCH YẾU ĐẠI DIỆN ĐỂ PHÂN LỚP DỮ LIỆU CHUỖI THỜI GIAN HIỆU QUẢ

Các kỹ thuật thu gọn dữ liệu (Data Reduction Techniques - DRTs) trong phần này đều dựa trên ý tưởng đơn giản là: những phần tử không đại diện cho ranh giới quyết định giữa các lớp là những phần tử vô ích cho quá trình phân lớp. Do đó, chúng có thể bị loại bỏ. Giải thuật phân lớp k-NN đạt được độ chính xác gần như nhau khi thực hiện phân lớp trên tập huấn luyện và tập ngưng tụ. Việc duyệt qua tập ngưng tụ hiệu quả hơn duyệt qua tập huấn luyện. Do đó, các kỹ thuật thu gọn dữ liệu cố gắng lựa chọn hoặc tạo ra một số lượng vừa đủ các các phần tử nằm trong vùng dữ liệu gần với ranh giới quyết định. Các kỹ thuật thu gọn dữ liệu được trình bày trong phần này là phi tham số. Chúng xác định kích thước của tập ngưng tụ một cách tự động dựa trên mức độ nhiễu (noise) và số lượng lớp trong dữ liệu ban đầu (càng nhiều lớp, các nhiều ranh giới, do đó, càng nhiều phần tử được chọn hoặc tạo ra).

3.4.1 Các giải thuật lựa chọn đại diện (Prototype Selection Algorithms)

3.4.1.1 Giải thuật CNN-rule

CNN-rule [7] là giải thuật lựa chọn đại diện (PS) ra đời sớm nhất và nổi tiếng nhất. Nó sử dụng hai tập hợp CS và TS. Đầu tiên, một phần tử trong tập huấn luyện được đưa vào CS, những phần tử còn lại trong tập huấn luyện để trong tập TS. Sau đó, CNN-rule cố gắng phân lớp tập TS bằng cách sử dụng giải thuật phân lớp 1-NN trên nội dung của tập CS. Khi một phần tử bị phân loại sai, nó được xem xét như nằm trong vùng dữ liệu gần ranh giới quyết định. Vì thế nó được chuyển từ tập TS sang tập CS. Giải thuật ngừng lại khi không còn có sự chuyển dữ liệu từ tập TS sang tập CS.

Mã giả của giải thuật CNN-rule được trình bày trong Algorithm 5: nó bắt đầu với một tập huấn luyện TS và trả về một tập ngưng tụ CS. Ban đầu, CS chỉ chứa một phần tử huấn luyện (dòng 1, 2). Sau đó, với mỗi phần tử huấn luyện $x \in TS$ (dòng 5), giải thuật truy xuất và kiểm tra nhãn lớp của lân cận gần nhất của nó (dòng 6). Nếu nhãn lớp của x khác với nhãn lớp của lân cận gần nhất của nó (dòng 7), x sẽ được chuyển vào tập CS (dòng 8, 9). Quá trình này được lặp (dòng 4,10,13) đến khi không còn phần tử nào trong tập TS được chuyển vào tập CS.

Giải thuật CNN-rule dựa trên ý tưởng đơn giản là: những phần tử mà được phân lớp một cách đúng đắn bởi giải thuật 1-NN thì được xem là đang ở trung tâm lớp của vùng dữ liệu và vì thế chúng sẽ được bỏ qua. Nói cách khác, những phần tử mà phân lớp sai (misclassified) thì được xem là nằm gần đường biên lớp của vùng dữ liệu và vì thế chúng

Algorithm 5. CNN-rule

Input: TS
Output: CS

1: $CS \leftarrow \varnothing$
2: pick an item of TS and move it to CS
3: **repeat**
4: $stop \leftarrow TRUE$
5: **for** each $x \in TS$ **do**
6: $NN \leftarrow$ Nearest Neighbour of x in CS
7: **if** $NN_{class} \neq x_{class}$ **then**
8: $CS \leftarrow CS \cup \{x\}$
9: $TS \leftarrow TS - \{x\}$
10: $stop \leftarrow FALSE$
11: **end if**
12: **end for**
13: **until** $stop == TRUE$
14: discard TS
15: **return** CS

được đưa vào tập ngưng tụ. Điểm yếu của giải thuật CNN-rule là việc tạo ra kết quả tập ngưng tụ phụ thuộc vào thứ tự các phần tử trong tập huấn luyện. Điều này có nghĩa là có thể có nhiều tập ngưng tụ khác nhau được tạo ra bởi cùng một tập huấn luyện với các thứ tự phần tử khác nhau.

3.4.1.2 Giải thuật IB2

Giải thuật IB2 thuộc họ các giải thuật học dựa trên phần tử (Instance-Based Learning - IBL) nổi tiếng và dựa trên giải thuật CNN-rule. Algorithm 6 trình bày mã giả của giải thuật IB2. Mỗi phần tử trong tập huấn luyện $x \in TS$ được phân lớp bằng giải thuật phân lớp 1-NN dựa trên tập ngưng tụ CS tại thời điểm hiện tại (dòng 4). Nếu x được phân lớp đúng, nó sẽ bị loại bỏ (dòng 8). Ngược lại, x sẽ được chuyển vào tập ngưng tụ CS (dòng 6). Trái với giải thuật CNN-rule, giải thuật IB2 không đảm bảo mọi phần tử bị loại bỏ đều có thể được phân lớp một cách đúng đắn dựa vào tập ngưng tụ. Tuy nhiên, IB2 là giải thuật duyệt một lần (one-pass algorithm) nên nó rất nhanh, chi phí tính toán tiền xử lý thấp. Thêm vào đó, giải thuật IB2 xây dựng tập ngưng tụ một cách gia tăng nên rất thích hợp cho môi trường động, phát trực tuyến nơi mà các phần tử huấn luyện mới được thêm vào một cách liên tục. Ngoài ra, giải thuật IB2 còn khác với giải thuật CNN-rule và các kỹ thuật thu gọn dữ liệu khác ở chỗ giải thuật IB2 không đòi hỏi toàn bộ tập huấn luyện phải được lưu trong bộ nhớ. Vì thế, giải thuật IB2 có thể được ứng dụng trên các thiết bị có bộ nhớ hạn chế không thể chứa toàn bộ tập huấn luyện. Và cũng giống như giải thuật CNN-rule, IB2 cũng là giải thuật phụ thuộc vào thứ tự của tập huấn luyện.

Algorithm 6. IB2

Input: TS
Output: CS

$CS \leftarrow \varnothing$
an item is chosen at random to migrate from TS to CS
for each $x \in TS$ **do**
    $NN \leftarrow$ Nearest Neighbour of x in CS
    if $NN_{class} \neq x_{class}$ **then**
        $CS \leftarrow CS \cup \{x\}$
    end if
    $TS \leftarrow TS - \{x\}$
end for
return CS

3.4.2 Các giải thuật trích yếu đại diện (Prototype Abstraction Algorithms)

3.4.2.1 Giải thuật trích yếu IB2 (AIB2)

Giải thuật AIB2 là phiên bản trích yếu đại diện (PA) của giải thuật IB2. Vì thế, nó kế thừa tất cả các đặc điểm của giải thuật IB2. Ý tưởng của giải thuật AIB2 rất đơn giản: những phần tử đại điện phải nằm ở trung tâm của vùng dữ liệu mà nó đại diện. Do đó, những phần tử được phân lớp đúng không bị loại bỏ. Những phần tử này góp phần tạo ra tập ngưng tụ bằng cách định vị lại những đại diện gần nhất của chúng bằng cách áp dụng khái niệm trọng số. Mỗi đại diện được đặc trưng bởi một trọng số biểu thị số lượng phần tử mà nó đại diện.

Algorithm 7. AIB2

Input: TS
Output: CS

$CS \leftarrow \varnothing$
move a random item y from TS to CS
$y_{weight} \leftarrow 1$
for each $x \in TS$ **do**
    $NN \leftarrow$ Nearest Neighbour of x in CS
    if $NN_{class} \neq x_{class}$ **then**
        $x_{weight} \leftarrow 1$
        $CS \leftarrow CS \cup \{x\}$
    else
        for each attribute $attr(i)$ **do**
            $NN_{attr(i)} \leftarrow \frac{NN_{attr(i)} \times NN_{weight} + x_{attr(i)}}{NN_{weight}+1}$
        end for
        $NN_{weight} \leftarrow NN_{weight} + 1$
    end if
    $TS \leftarrow TS - \{x\}$
end for
return CS

Algorithm 7 hiện thực giải thuật AIB2 bằng mã giả. Đầu tiên, tập ngưng tụ CS chỉ có một phần tử duy nhất với trọng số khởi tạo là 1 (dòng 1-3). Với mỗi phần tử huấn luyện x, giải thuật AIB2 truy xuất đại diện gần nhất của nó NN từ tập ngưng tụ hiện hành (dòng 6-8). Những thuộc tính của NN được cập nhật bằng cách tính đến trọng số hiện tại của nó và các thuộc tính của x. Kết quả là NN tiến về phía x (dòng 10-12). Cuối cùng, trọng số của NN được tăng lên một đơn vị (dòng 13) và x bị loại bỏ (dòng 15).

Giải thuật AIB2 hướng tới mục đích nâng cao hiệu của giải thuật IB2 bằng cách xây dựng tập ngưng tụ với những phần tử đại diện tốt hơn giải thuật IB2. Mỗi đại diện nằm gần với trung tâm của vùng dữ liệu mà nó đại diện. Do vậy, AIB2 có thể đạt được độ chính xác phân lớp cao hơn. Hơn thế nữa, những đại diện được định vị lại góp phần làm giảm số lượng phần tử trong tập ngưng tụ cuối cùng do đó AIB2 có thể đạt được tỉ lệ thu gọn cao hơn và chi phí tiền xử lý tốt hơn IB2.

3.4.2.2 Giải thuật CJA

Giải thuật trích yếu đại diện của Chen và Jozwik (CJA) [2] làm việc như sau: Đầu tiên, giải thuật xác định đường kính của tập huấn luyện bằng cách tìm khoảng cách cách xa nhất của hai phần tử A và B bất kỳ trong tập huấn luyện. Sau đó, tập huấn luyện được phân chia thành hai tập huấn luyện con, S_A và S_B. S_A chứa những phần tử huấn luyện nằm gần A hơn, S_B chứa những phần tử huấn luyện nằm gần B hơn. Tiếp theo, giải thuật CJA chọn tập huấn luyện con nào có chứa phần tử của nhiều hơn một lớp (tập con không thuần nhất) để tiếp tục phân chia. Tập con có đường kính lớn nhất sẽ được chia trước. Nếu tất cả các tập con đều thuần nhất, giải thuật CJA chia tập con thuần nhất lớn nhất. Thủ tục này được thực hiện cho đến khi số lượng tập con tạo ra bằng với giá trị được chỉ định bởi người dùng. Cuối cùng, với mỗi tập con S, giải thuật CJA tính trung bình của các phần tử trong S và tạo ra phần tử trung bình (mean) được gán nhãn là lớp chính trong S. Các phần tử trung bình được tạo ra sẽ tạo thành tập ngưng tụ.

Algorithm 8 trình bày mã giả của giải thuật CJA. Nó nhận vào một tập huấn luyện TS và một số n là số lượng đại điện sẽ được tạo ra. Đầu tiên, toàn bộ tập huấn luyện TS được lưu trữ trong S (dòng 2). Sau đó, tập con không thuần nhất C với đường kính lớn nhất được chia làm hai tập con (dòng 4, 8). Nếu tất cả các tập con đều thuần nhất, giải thuật CJA chia tập con thuần nhất C với đường kính lớn nhất (dòng 5-7). Cả hai tập con được thêm vào S và loại bỏ tập C ra khỏi S (dòng 9-11). Quá trình tạo ra tập con dừng lại khi số lượng tập con được tạo ra bằng với n (dòng 3). Bước cuối cùng là tính trung bình (hoặc tạo ra đại diện) cho mỗi tập con và lưu vào tập ngưng tụ CS (dòng 13-18).

Giải thuật CJA chọn tập con để chia bằng cách xét đường kính của nó. Ý tưởng chính cho phương pháp này là một tập con với đường kính lớn thì khả năng nó chứa nhiều phần

Algorithm 8. CJA

Input: TS, n
Output: CS

$S \leftarrow \varnothing$
$add(S, TS)$
for $i = 2$ to n **do**
    $C \leftarrow$ select the non-homogeneous subset $\in S$ with the largest diameter
    if $C == \varnothing$ {All subsets are homogeneous} **then**
        $C \leftarrow$ select the homogenous subset $\in S$ with the largest diameter
    end if
    $(S_x, S_y) \leftarrow$ devide C into two subsets
    $add(S, S_x)$
    $add(S, S_y)$
    $remove(S, C)$
end for
$CS \leftarrow \varnothing$
for each subset $T \in S$ **do**
    $r \leftarrow$ compute the mean item by averaging the items in T
    $r.label \leftarrow$ find the most common class label in T
    $CS \leftarrow CS \cup \{r\}$
end for
return CS

tử huấn luyện hơn. Do đó, nếu tập con này được chia trước thì sẽ đạt được tỉ lệ thu gọn cao hơn. Ưu điểm của giải thuật CJA là nó xây dựng ra cùng một tập ngưng tụ không bị ảnh hưởng bởi thứ tự của các phần tử trong tập huấn luyện. Tuy nhiên, giải thuật CJA có hai nhược điểm chính. Thứ nhất là đây là giải thuật có tham số. Người dùng phải chỉ định trước số lượng đại diện được tạo ra. Điều này thường dẫn đến việc tốn nhiều chi phí tinh chỉnh với thủ tục thử và lỗi. Trong một số lĩnh vực cụ thể, nhược điểm này là điều mong muốn bởi vì nó cho phép người dùng kiểm soát được kích thước của tập ngưng tụ. Tuy nhiên, nhược điểm này đã ngăn cản việc tìm ra tập ngưng tụ chính xác phù hợp với bản chất của dữ liệu. Điểm yếu thứ hai là những phần tử không thuộc vào lớp phổ biến nhất của tập con thì không được đại diện trong tập ngưng tụ. Bởi vì phần tử trung bình của mỗi tập con được gán nhãn của lớp phổ biến nhất, những phần tử thuộc các lớp khác sẽ bị bỏ qua.

Các công trình nghiên cứu trên của Ougiaroglou và cộng sự đều đạt được những kết nhất định. Tuy nhiên trong quá trình phân lớp sử dụng giải thuật k-NN, các công trình nghiên cứu này lại sử dụng độ đo Euclid là chưa thật sự phù hợp với dữ liệu chuỗi thời gian. Đọ đo xoắn thời gian động (DTW) được đánh giá là cho kết quả tối ưu khi phân lớp với giải thuật k-NN trên dữ liệu chuỗi thời gian [1].

CHƯƠNG 4. GIẢI PHÁP

- Hiện thực giải thuật phân lớp k-NN thuần túy để phân lớp dữ liệu chuỗi thời gian
- Hiện thực giải thuật phân lớp k-NN có kết hợp sử dụng kỹ thuật thu gọn tập huấn luyện RHC, dRHC, Naïve Ranking để phân lớp dữ liệu chuỗi thời gian

So sánh kết quả của 3 phương pháp trên: cho các giải thuật chạy trên cùng tập huấn luyện, so kết quả thu gọn và độ chính xác phân lớp sau khi thu gọn.

CHƯƠNG 5. NỘI DUNG VÀ KẾ HOẠCH NGHIÊN CỨU

5.1 NỘI DUNG NGHIÊN CỨU

Để hiện thực theo mục tiêu đã đề ra ban đầu là so sánh hai phương pháp thu gọn tập huấn luyện RHC và Naïve ranking trong phân lớp dữ liệu chuỗi thời gian cần thực hiện những nội dung sau:

- Tìm hiểu giải thuật phân lớp k-NN
- Tìm hiểu các giải thuật: RHC, dRHC
- Tìm hiểu giải thuật: Naïve Ranking
- Hiện thực giải thuật phân lớp k-NN thuần túy trên tập dữ liệu chuỗi thời gian
- Hiện thực giải thuật phân lớp k-NN có sử dụng kỹ thuật thu gọn tập huấn luyện RHC, dRHC, Naïve Ranking

  So sánh kết quả của 3 phương pháp trên: cho các giải thuật chạy trên cùng tập huấn luyện, so kết quả thu gọn và như độ chính xác phân lớp sau khi thu gọn.

Dữ liệu mẫu cho bài toán phân lớp dữ liệu chuỗi thời gian được lấy từ trang web http://www.cs.ucr.edu/ eamonn/.

5.2 KẾ HOẠCH NGHIÊN CỨU

STT	Nội dung	Thời gian (Tuần)
1	Tìm hiểu giải thuật phân lớp k-NN	1
2	Tìm hiểu giải thuật: RHC và dRHc	4
3	Tìm hiểu giải thuật: Naïve Ranking	2
4	Hiện thực giải thuật phân lớp k-NN thuần túy	3
5	Hiện thực giải thuật k-NN + RHC, k-NN + dRHC	4
6	Hiện thực giải thuật k-NN + Naïve Ranking	2
7	Thực nghiệm và so sánh kết quả thực nghiệm của 3 phương pháp trên	4
8	Viết luận văn	4
	Tổng cộng	24

TÀI LIỆU THAM KHẢO

[1] Buza, K., Nanopoulos A., and Schmidt-Thieme L. (2011), "INSIGHT: Efficient and Effective Instance Selection for Time-Series Classification", Proceedings of PAKDD 2011, Shenzen, China, May 24-27.

[2] Chen, C.H., Jozwik, A. (1996), "A sample set condensation algorithm for the class sensitive artificial neural network. Pattern Recogn". Lett. 17(8), 819–823.

[3] Duong Tuan Anh (2009), "An Overview of Similarity Search in Time Series Data", in Proceedings of the 11th Conference on Science and Technology - Section of Computer Science and Engineering, Ho Chi Minh City University of Technology, 21-23 October, 2009, pp. 86-95.

[4] Keogh E., (2002), "Exact indexing of dynamic time warping". In Proc.VLDB, Proceedings of the 28th VLDB Conference, Hong Kong, China.

[5] Han J., Kamber M., Pei J. (2012), Data Mining: Concepts and Techniques, Third Edition, Morgan Kaufmann Publishers.

[6] Lin J., Keogh E., Lonardi S., and Chiu B. (2003), "A symbolic representation of time series, with implications for streaming algorithms", in SIGMOD'03, 2003.

[7] Hart,P.E., "The condensed nearest neighbor rule. IEEE Transactions on Information Theory" 14(3), 515–516 (1968)

[8] Ougiaroglou S. and Evangelidis G. (2016), "RHC: a non-parametric cluster-based data reduction for efficient k-NN classification". Pattern Analysis and Applications, Vol 19, 93-109

[9] Ougiaroglou S., Karamitopoulos L., Tatoglou C. (2015), "Applying Prototype Selection and Abstraction Algorithms for Efficient Time-Series Classification". Artificial Neural Networks Methods and Applications in Bio-/Neuroinformatics pp.333-348

[10] Sánchez, J.S. (2004), "High training set size reduction by space partitioning and prototype ab straction". Pattern Recognition 37(7), 1561–1564

[11] Yamada, Y., Suzuki, E., Yokoi H., Takabayashi K. (2003), "Decision-tree induction from time-series data based on a standard-example split test". In: Proc. Twentieth International Conference on Machine Learning (ICML), pp. 840–847

[12] Xi X., Keogh E., Shelton C., Wei L., and Ratanamahatana C. A. (2006), "Fast Time Series Classification using Numerosity Reduction." In the Proc of the 23rd ICML, 1033-1040.

[13] Wang X., Mucen A., Ding H., Trjcevske G., Scheuermann P., Keogh E. (2013), "Experiment Comparision of Representation Methods and Distance Measures for Time Series, Data Mining and Knowledge Discovery", Vol. 26, pp.275-309.