ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

PHAN THÀNH NHÂN

SỬ DỤNG TÍNH TOÁN SONG SONG ĐỂ LỰA CHỌN ĐẶC TRƯNG DỰA TRÊN TIÊU CHÍ TỈ LỆ VẾT

LUẬN VĂN THẠC SĨ TOÁN HỌC

TP. Hồ Chí Minh - 2022

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

PHAN THÀNH NHÂN

SỬ DỤNG TÍNH TOÁN SONG SONG ĐỂ LỰA CHỌN ĐẶC TRƯNG DỰA TRÊN TIÊU CHÍ TỈ LỆ VẾT

LUẬN VĂN THẠC SĨ TOÁN HỌC

CHUYÊN NGÀNH CƠ SỞ TOÁN CHO KHOA HỌC DỮ LIỆU

Mã ngành: 8480109

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. Nguyễn Thị Thu

TP. Hồ Chí Minh - 2022

Lời cam đoan

Tôi cam đoan luận văn thạc sĩ chuyên ngành Cơ sở Toán cho Khoa học Dữ liệu, với đề tài "*Sử dụng tính toán song song để lựa chọn đặc trưng dựa trên tiêu chí tỉ lệ vết*" là công trình khoa học do Tôi thực hiện dưới sự hướng dẫn của TS. Nguyễn Thị Thu và sự hỗ trợ, góp ý của PGS.TS Nguyễn Thanh Bình

Những kết quả nghiên cứu của luận văn hoàn toàn trung thực và chính xác.

Học viên cao học

Phan Thành Nhân

Lời cảm ơn

Lời đầu tiên, tôi xin gửi lời cảm ơn chân thành tới TS. Nguyễn Thị Thu, người vừa là người thầy, vừa là người chị đã giúp đỡ tôi rất nhiều, người đã từng bước định hướng và trang bị cho tôi những kiến thức cần thiết, hướng dẫn và chỉ bảo tận tình cho tôi trong suốt quá trình học tập, nghiên cứu cũng như thực hiện luận văn này. Tôi cũng xin gửi lời cảm ơn đến PGS. TS. Nguyễn Thanh Bình, người đã tạo điều kiện và hỗ trợ tôi rất nhiều, thầy thường chia sẻ và cho tôi rất nhiều lời khuyên và góp ý trong việc định hướng nghiên cứu trong suốt thời gian tôi theo học chương trình thạc sĩ ở khoa Toán. Trong quá trình thực hiện luận văn này, tôi đã gặp không ít khó khăn do thời gian không nhiều cũng như kiến thức còn hạn chế, tuy nhiên tôi luôn nhận được sự quan tâm, giúp đỡ, động viên đến từ gia đình, thầy cô, bạn bè.

Tôi xin chân thành cảm ơn đến các thầy cô trong khoa Toán và các thầy cô được mời giảng dạy trong chương trình thạc sĩ này, các thầy cô đã trực tiếp trang bị cho tôi kiến thức toán cũng như phương pháp tự học, tự nghiên cứu. Lời cuối cùng, để hoàn thành luận văn, tôi đã sử dụng, tham khảo một số tài liệu và bài viết, xin cảm ơn các tác giả.

TP.HCM, Ngày 3 tháng 4 năm 2023

Tác giả

Phan Thành Nhân

TRANG THÔNG TIN LUẬN VĂN

Tên đề tài luận văn: Sử dụng tính toán song song để lựa chọn đặc trưng dựa trên tiêu chí tỉ lệ vết
Ngành: Cơ sở Toán cho Khoa học Dữ liệu
Mã số ngành: 8480109
Họ tên học viên cao học: Phan Thành Nhân
Khóa đào tạo: 2020 - 2022
Người hướng dẫn khoa học: TS. Nguyễn Thị Thu.
Cơ sở đào tạo: Trường Đại học Khoa học Tự nhiên, ĐHQG.HCM

1. TÓM TẮT NỘI DUNG LUẬN VĂN:

- Chương 1: Giới thiệu tổng quan về lựa chọn đặc trưng và các nghiên cứu liên quan.
- Chương 2: Trình bày về tiêu chí vết - tiếu chí lựa chọn đặc trưng. Sau đó chứng minh các kết quả và cuối cùng trình bày Thuật toán PSFT - thuật toán lựa chọn đặc trưng song song dựa trên tiêu chí vết.
- Chương 3: Trình bày về các tập dữ liệu được sử dụng khi làm thực nghiệm. Thông tin về phần cứng được sử dụng và thảo luận kết quả thực nghiệm.
- Cuối cùng: Phần kết luận và danh mục các tài liệu tham khảo.

2. NHỮNG KẾT QUẢ MỚI CỦA LUẬN VĂN:

Xây dựng một thuật toán lựa chọn đặc trưng song song dựa trên tiêu chí vết (thuật toán PFST) trên các tập dữ liệu lớn. Thuật toán sử dụng tiêu chí vết để đánh giá các đặc trưng tốt. Thực nghiệm cho thấy thuật toán cho kết quả tốt không chỉ về mặt hiệu suất của mô hình LDA cho bài toán phân loại mà còn tiết kiệm chi phí tính toán, khi thời gian chạy và đạt kết quả nhanh chóng.

3. CÁC ỨNG DỤNG/ KHẢ NĂNG ỨNG DỤNG TRONG THỰC TIỄN HAY NHỮNG VẤN ĐỀ CÒN BỎ NGỎ CẦN TIẾP TỤC NGHIÊN CỨU:
Kết quả luận văn có thể được dùng để lựa chọn đặc trưng trên các tập dữ liệu thực tế. Tuy nhiên, một trong những nhược điểm của PFST là tiêu chí vết chỉ có thể sử dụng cho các đặc trưng liên tục. Do đó, trong tương lai, chúng tôi vẫn sẽ nỗ lực làm việc, và hi vọng sẽ khám phá ra cách mở rộng cho các trường hợp còn lại trong bài toán phân loại.

CÁN BỘ HƯỚNG DẪN
(Ký tên, họ tên)

HỌC VIÊN CAO HỌC
(Ký tên, họ tên)

XÁC NHẬN CỦA CƠ SỞ ĐÀO TẠO
HIỆU TRƯỞNG

THESIS INFORMATION

Thesis title: Parallel feature selection based on the trace ratio criterion
Speciality: Data science
Code: 8480109
Name of Master Student: Phan Thành Nhân
Academic year: 2020 - 2022
Supervisor: Dr. Nguyen Thi Thu
At: VNUHCM - University of Science

1. SUMMARY:

- Chapter 1: An overview of feature selection and related works.
- Chapter 2: Present trace criterion as a feature selection criterion. Prove the results and present parallel Feature Selection using Trace criterion (PFST Algorithm).
- Chapter 3: Presentation of the experiment's data sets. Discussion of the experimental findings and information on the hardware utilized.
- Finally: Conclusion and list of references.

2. NOVELTY OF THESIS: The main results in this thesis are as follows.

Presents a novel parallel feature selection approach for classification, called PFST, for feature selection on large-scale datasets. The method uses the trace criterion. The experiments show that our PFST method achieves good accuracy using the LDA model and less running time to select relevant features.

3. APPLICATIONS/ APPLICABILITY/ PERSPECTIVE:

The results of this thesis can be used to select the features of the real data sets. However, one of the drawbacks of the algorithm is that the trace criterion can only be used for continuous features. Therefore, in the future, it would be desirable to explore how to extend this work to the case where there are categorical features in the model as well.

SUPERVISOR **Master STUDENT**

CERTIFICATION
UNIVERSITY OF SCIENCE
PRESIDENT

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh sách các thuật toán

Lời nói đầu

Sự phát triển của dữ liệu ngày nay đặt ra thách thức trong quản lý, lưu trữ và áp dụng vào các bài toán cụ thể. Trong khi các phương pháp trích xuất đặc trưng có thể giảm kích thước dữ liệu để thực hành, nhưng các phương pháp này không giúp giảm thiểu chi phí lưu trữ dữ liệu. Trong khi đó, việc lựa chọn đặc trưng giúp loại bỏ các đặc trưng dư thừa và do đó hữu ích không chỉ trong thực hành nghiên cứu dữ liệu mà còn trong giảm chi phí quản lý và lưu trữ. Trong đề tài luận văn cao học này, chúng tôi phát triển một phương pháp "Sử dụng tính toán song song để lựa chọn đặc trưng dựa trên tiêu chí tỉ lệ vết (Parallel feature selection based on the trace ratio criterion - PFST)"cho bài toán phân loại. Tiêu chí được sử dụng là một thước đo về khả năng tách lớp được sử dụng trong LDA, để đánh giá tính hữu dụng của đặc trưng. Dựa trên tiêu chí này, PFST sẽ nhanh chóng tìm thấy các đặc trưng quan trọng từ một tập hợp các đặc trưng gốc của tập dữ liệu lớn bằng cách sử dụng sức mạnh của tính toán song song để thực hiện lựa chọn đặc trưng và loại bỏ các đặc trưng dư thừa. Sau khi các đặc trưng quan trọng nhất được đưa vào mô hình, chúng tôi đánh giá phương pháp thông qua các thử nghiệm khác nhau bằng cách sử dụng LDA làm mô hình phân loại. Thực nghiệm cho thấy rằng phương pháp của chúng tôi có thể chọn ra một tập hợp con nhỏ các đặc trưng trong thời gian ngắn, tiết kiệm thời gian hơn so với một số phương pháp khác. Ngoài ra, độ chính xác khi phân loại dựa trên các đặc trưng được lựa chọn bằng PFST cũng đạt độ chính xác cao và tốt hơn các phương pháp khác, và tốt hơn việc phân loại dựa trên tập các đặc trưng gốc ban đầu.

Nội dung khóa luận này bao gồm 4 chương. Trong đó,

Chương 1: Chương này giới thiệu tổng quan về lý do chọn đề tài và các nghiên cứu khoa học liên quan.

Chương 2: Chương này trình bày các kiến thức nền tảng về vấn đề lựa chọn đặc trưng cho lớp các bài toán phân loại.

Chương 3: Chương này trình bày về tiêu chí vết - tiếu chí lựa chọn đặc trưng. Thuật toán PSFT - thuật toán lựa chọn đặc trưng song song dựa trên tiêu chí vết.

Chương 4: Chương này trình bày về các tập dữ liệu được sử dụng khi làm thực nghiệm và thông tin về phần cứng được sử dụng. Trình bày các thuật toán được dùng để so sánh với PFST và thảo luận kết quả

Cuối cùng: Phần kết luận và danh mục các tài liệu tham khảo.

Chương 1

Tổng quan

1.1 Lý do chọn đề tài

Trong kỷ nguyên của dữ liệu lớn, sự phát triển của dữ liệu đặt ra những thách thức đối với việc quản lý và sử dụng dữ liệu hiệu quả. Ví dụ: một tập dữ liệu gen có thể chứa hàng trăm nghìn đặc điểm [19]. Do đó, việc xử lý trực tiếp các tập dữ liệu như vậy có thể gặp sự khó khăn về kích thước. Hơn nữa, các đặc trưng dư thừa có thể làm giảm hiệu suất học tập của các thuật toán phân loại. Để giải quyết vấn đề này, nhiều kỹ thuật giảm chiều dữ liệu đã được phát triển [34, 9, 15, 17, 32] và chúng được phân loại thành các phương pháp trích xuất đặc trưng hoặc lựa chọn đặc trưng [30, 28]. Kỹ thuật trích xuất đặc trưng (ví dụ: Phân tích thành phần chính (Principal Component Analysis - PCA) [24], Phân tích phân biệt tuyến tính (Linear Discriminant Analysis - LDA) [24]) liên quan đến việc chiếu dữ liệu vào một không gian đối tượng mới với số chiều nhỏ hơn thông qua một vài bước biến đổi tuyến tính hoặc phi tuyến từ các đặc trưng gốc. Tuy nhiên, điều này tạo ra một loạt các đặc trưng mới mà không thể diễn giải trực tiếp. Hơn nữa, vì những cách tiếp cận đó sử dụng tất cả các đặc trưng có sẵn trong quá trình trích xuất đặc trưng nên nó không giúp giảm chi phí lưu trữ dữ liệu và chi phí thu thập dữ liệu trong tương lai. Mặt khác, các phương pháp lựa chọn đặc trưng ([42, 23],. . .) chỉ chọn một tập hợp con các đặc trưng hữu ích để xây dựng mô hình. Do đó, điều này giúp giữ các tính chất của các đặc trưng ban đầu trong khi giảm chi phí lưu trữ và thu thập dữ liệu trong tương lai bằng cách loại bỏ các đặc trưng không liên quan. Tuy nhiên, các dữ liệu từ các lĩnh vực khác nhau như khai thác văn bản, phân tích kinh doanh và sinh học, thường được đo bằng gigabyte hoặc terabyte với hàng triệu đặc trưng [6, 29]. Ví dụ: tập dữ liệu Amazon Review [38] là tập dữ liệu 34 gigabyte. Trong những trường hợp như vậy, hiệu suất của các kỹ thuật lựa chọn đặc trưng mới nhất có thể bị ảnh hưởng [29]. Điều này là do không

gian tìm kiếm cho một tập hợp con các đặc trưng hữu ích bị tăng lên đáng kể. Một cách để giải quyết vấn đề này là sử dụng tính toán song song, cho phép sử dụng tốt hơn tài nguyên tính toán của máy tính bằng cách phân vùng dữ liệu và chạy các lựa chọn đặc trưng trên nhiều lõi cùng một lúc.

Từ những lý do trên, trong đề tài luận văn cao học này, chúng tôi nghiên cứu và trình bày một phương pháp "Sử dụng tính toán song song để lựa chọn đặc trưng dựa trên tiêu chí tỉ lệ vết (Parallel feature selection based on the trace ratio criterion - PFST)"cho bài toán phân loại. Tiêu chí đánh giá mức độ hữu dụng của đặc trưng được sử dụng là một thước đo về khả năng tách lớp được sử dụng trong bài toán Linear discriminant analysis (LDA - chúng tôi sẽ trình bày bài toán này bên dưới). Dựa trên tiêu chí này, PFST sẽ nhanh chóng tìm thấy các đặc trưng quan trọng từ một tập hợp các đặc trưng gốc của tập dữ liệu lớn và loại bỏ các đặc trưng dư thừa. Bên cạnh đó, chúng tôi thiết lập và sử dụng tính toán song song để tối ưu nguồn tài nguyên của máy tính nhằm tăng hiệu suất tính toán và thời gian chạy. Cuối cùng, chúng tôi sẽ sử dụng LDA làm mô hình phân loại để đánh giá độ hiệu quả của thuật toán PFST. Bên cạnh đó, chúng tôi cũng so sánh với một số thuật toán khác. Kết quả thu được rất tốt, PFST có thời gian chạy và có độ chính xác cao hơn các phương pháp khác. Bên cạnh đó, tập đặc trưng thu được từ PFST cũng cho ra kết quả tốt hơn nếu sử dụng toàn bộ đặc trưng ban đầu.

1.2 Các công trình nghiên cứu khoa học có liên quan

Đối mặt với các thách thức về kích thước của dữ liệu ngày càng tăng, đã có nhiều nỗ lực trong hướng nghiên cứu về lựa chọn đặc trưng để phát triển các kỹ thuật mới. Bên cạnh các phương pháp lai để kết hợp các chiến lược lựa chọn đặc trưng khác nhau [40, 2], hầu hết các phương pháp lựa chọn đặc trưng có thể được chia thành ba loại.

Đầu tiên, cách tiếp cận "wrapper" dựa trên hiệu suất của một thuật toán học máy cụ thể để đánh giá tầm quan trọng của các đặc trưng được chọn. Một phương pháp "wrapper" điển hình sẽ tìm kiếm một tập con các đặc trưng dựa trên một thuật toán học máy trước, sau đó sẽ đánh giá chúng. Các bước này được lặp lại cho đến khi thỏa mãn một số tiêu chí dừng. Các phương pháp trong loại này thường rất tốn kém về chi phí tính toán vì việc đánh giá tập con các đặc trưng yêu cầu nhiều lần lặp lại. Mặc dùng nhiều các tiếp cận tìm kiếm được đề xuất chẳng hạn như thuật toán tìm kiếm best-first [3] và thuật toán di chuyền (genetic) [16]. Tuy nhiên, việc sử dụng các thuật toán này cho dữ liệu nhiều chiều vẫn không cho thấy sử cải thiện về chi phí tính toán.

Thứ hai, cách tiếp cận "filter" bao gồm các kỹ thuật đánh giá các tập hợp con đặc bằng việc xếp hạng với một số tiêu chí như tiêu chí thông tin [37, 41], khả năng tái tạo [13, 33]. Các phương pháp này chọn các đặc trưng độc lập với thuật toán học máy và thường hiệu quả hơn về chi phí tính toán so với các phương pháp "wrapper" [29]. Tuy nhiên, vì không được tối ưu hóa cho bất kỳ thuật toán học máy mục tiêu nào, nên chúng có thể không tối ưu cho một tuật toán học máy cụ thể.

Thứ ba, các phương pháp "embedded" sử dụng các tiêu chí độc lập để tìm ra tập con tối ưu cho một tập hợp nhất định. Sau đó, một thuật toán học máy được sử dụng để lựa chọn tập con tối ưu cuối cùng trong số các tập con tối ưu trên các tập hợp khác nhau. Vì thế, chúng hiệu quả hơn về chi phí tính toán so với các phương pháp "wrapper" vì chúng không đánh giá đặc trưng dựa trên việc lặp lại các tập con đặc trưng. Ngoài ra, chúng cũng được huấn luyện từ các thuật toán học máy. Vì thế, chúng có thể được coi như sự đánh đổi giữa phương pháp "filter" và phương pháp "wrapper"[29].

Mặc dù, cho đến này, các nhà khoa học đã nỗ lực rất nhiều trong hướng nghiên cứu lựa chọn đặc trưng, nhưng dữ liệu từ các trường, các ngành khác nhau có thể quá phong phú ngay cả đối với các phương pháp "filter"hiệu quả về chi phí tính toán. Điều này đã thúc đẩy nhiều nghiên cứu khác trong việc lựa chọn đặc trưng song song. Một số phương pháp đã được đề xuất trong [34, 9, 15, 17, 32] sử dụng quy trình xử lý song song để đánh giá đồng thời nhiều đặc trưng. Tuy nhiên, các thuật toán này yêu cầu quyên truy cập vào toàn bộ dữ liệu. Mặc khác, trong trong [43], các tác giả đã đề xuất một thuật toán lựa chọn đặc trưng song song cho hồi quy logistic dựa trên framework MapReduce và các đặc trưng được đánh giá thông qua hàm mục tiêu của mô hình hồi quy logistic. Trong khi đó, các tác giả của bài báo [46] đã đề xuất *Song song, Tiến–Lùi với thuật toán Tỉa (Parallel, Forward–Backward with Pruning algorithm)* (PFBP) để lựa chọn đặc trưng bằng cách bỏ sớm một số đặc trưng trong các lần lặp lại tiếp theo và sớm trả ra kết quả đặc trưng tốt nhất trong mỗi lần lặp. Tuy nhiên, các tiếp cận này yêu cầu tính toán bootstrap của p-giá trị, rất tốn kém chi phí tính toán. Trong bài báo [51], Zhao và các cộng sự đã giới thiệu một thuật toán lựa chọn đặc trưng song song để chọn các đặc trưng dựa trên khả năng giải thích phương sai của dữ liệu. Tuy nhiên, theo các tiếp cận của họ, việc xác định số lượng các đặc trưng trong mô hình dựa trên việc chuyển đổi các nhãn phân loại thành các giá trị số và sử dụng tổng bình phương sai số. Việc tính tổng bình phương sai số đòi hỏi phải điều chỉnh mô hình và do đó thuật toán vẫn còn tốn kém rất nhiều chi phí tính toán.

Chương 2

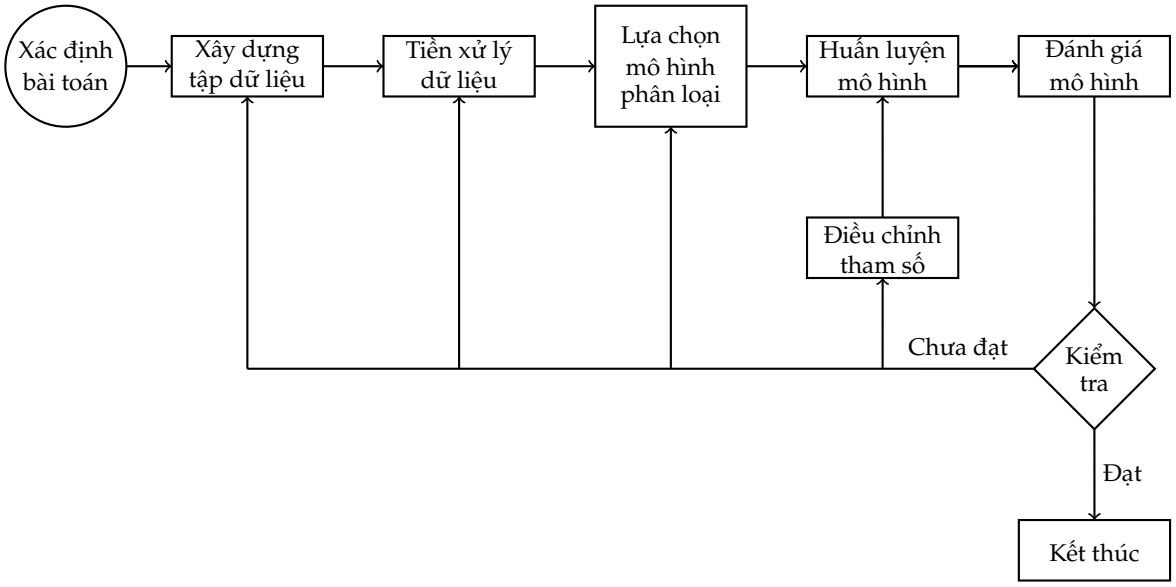
Kiến thức nền tảng

2.1 Quy trình của lớp các bài toán phân loại

Để giải quyết một bài toán phân loại, ta cần phải hiểu dữ liệu, tính chất các đặc trưng và quá trình chọn các đặc trưng phù hợp với mô hình cũng cần phải có một quy trình rõ ràng.

Bước đầu tiên là xác định bài toán, nghĩa là xác định nhãn của bài toán và xây dựng tập dữ liệu (data collection). Phải đảm bảo là tập dữ liệu phải liên quan đến bài toán được mô hình hóa. Bước này rất quan trọng vì sẽ ảnh hưởng rất nhiều đến kết quả phân loại. Ngoài ra, chỉ những đặc trưng có thông tin hữu ích về bài toán là nên được sử dụng. Trong một số trường hợp, khi gặp khó khăn về kiến thức cũng như khả năng thu thập dữ liệu. Chúng ta có thể sử dụng phương pháp brute-force để thay thế. Brute-force là một phương pháp giải quyết vấn đề bằng cách thực hiện tất cả các giải pháp có thể có và chọn ra giải pháp tốt nhất. Nó được sử dụng khi không có một thuật toán cụ thể nào có thể giải quyết vấn đề hoặc khi không có đủ kiến thức về vấn đề để thiết kế một giải pháp tối ưu. Tuy nhiên, phương pháp này có thể rất tốn kém và thời gian, đặc biệt là đối với các vấn đề có kích thước lớn. Do vậy, trong trường hợp này sẽ có một lượng rất lớn các biến được đo, xử lý và thêm vào tập dữ liệu. Tuy nhiên, hi vọng, trong tương lai có thể tác được các đặc trưng tốt nhất và phù hợp nhất với bài toán.

Nếu vấn đề về dữ liệu có thể được giải quyết, chúng ta sẽ nên bước tiếp theo trong quy trình phân loại là tiền xử lý dữ liệu (data pre-processing). Ở bước này, vấn đề chính là thiếu dữ liệu (missing data) và dữ liệu ngoại lai (outlier) cần phải được xử lý. Có một vài phương pháp phân tích thống kê [1, 22] để có thể dử lý các vấn đề này. Hơn nữa, đây là bước mà số lượng các đặc trưng của bài toán có thể giảm đi bằng việc áp dụng thuật toán lựa chọn đặc trưng.



Hình 2.1: Quy trình giải bài toán phân loại

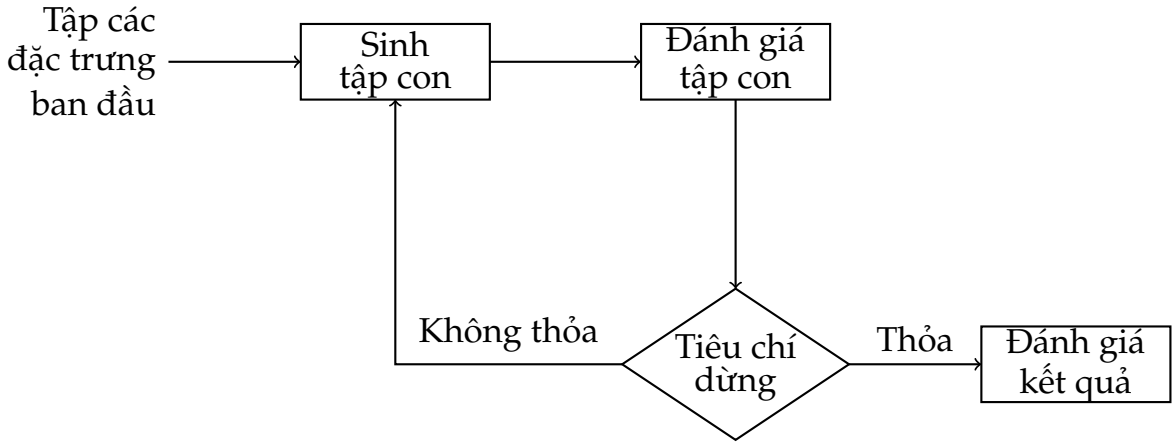
Vấn đề này có thể được giải quyết, đưa chúng ta đến bước tiếp theo trong quy trình phân loại: tiền xử lý dữ liệu. Ở giai đoạn này, các vấn đề chính như giá trị bị thiếu và phát hiện ngoại lệ nên được xử lý. Có nhiều phương pháp phân tích thống kê để giải quyết các vấn đề này [1, 26]. Ngoài ra, đây là giai đoạn trong vấn đề mà số lượng đặc trưng của vấn đề có thể được giảm bằng thuật toán lựa chọn đặc trưng.

Bước tiếp theo là lựa chọn thuật toán phân loại. Có rất nhiều các thuật tóa phân loại, và mặc dù các thuật toán rất đa dạng và khác nhau về ý tưởng, nhưng không dễ dàng chọn được thuật toán nào là tốt nhất cho một bài toán cụ thể. Do đó, việc thử nghiệm và so sánh một số thuật toán là một cách làm tương đối phổ biến, mục tiêu cuối cùng là lựa chọn được thuật toán cho kết quả tốt nhất [27].

Việc đánh giá các thuật toán phân loại thường dựa trên độ chính xác của việc dự đoán. Một kỹ thuật điển hình là chia dữ liệu thành hai phần để huấn luyện mô hình và sử dụng phần còn lại để kiểm tra độ chính xác. Tuy nhiên, quy trình này thường dẫn đến kết quả không tốt khi áp dụng vào tập dữ liệu bên ngoài. Do đó, để giảm thiểu sai số, một số kỹ thuật phức tạp hơn như kiểm tra chéo (cross-validation) [25] có thể được sử dụng.

Để có cái nhìn tổng quan, chúng tôi xin trình bày theo sơ đồ khối của toàn bộ quy trình của một bài toán phân loại.

Trong toàn bộ quy trình, nếu có bất kì bước nào không tốt, quy trình phải quy trở lại bước trước đó. Có nhiều nguyên nhân có thể ảnh hưởng đến hiếu suất của một bài toán phân loại [27] như:



Hình 2.2: Bốn bước của quy trình lựa chọn đặc trưng

- Đặc trưng phù hợp không được lựa chọn tốt.
- Tập dữ liệu không đủ, ít mẫu quan sát.
- Số lượng các đặc trưng quá nhiều.
- Kỹ thuật tiền xử lý dữ liệu chưa được tốt.
- Mô hình phân loiaj được chọn không phù hợp cho vấn đề hoặc cần điều chỉnh tham số.

Vì vậy, không thể chỉ ra rõ ràng bước nào trong quy trình cần trở lại. Tuy nhiên, mục tiêu cuối cùng là giải quyết bài toán phân loại đạt kết quả tốt nhất cho dữ liệu chưa được quan sát. Đây là bài toán khó, và mỗi bước thường được thực hiện trong thời gian dài, thông thường chúng ta cần phải liên tục thực hiện nhiều thử nghiệm mới để cải thiện khả năng dự đoán của mô hình.

2.2 Quy trình lựa chọn đặc trưng

Các thuật toán lựa chọn đặc trưng hoạt động bằng cách kết hợp một ý tưởng tìm kiếm để xác định tập hợp các đặc trưng, với một phương pháp đánh giá chúng. Mục tiêu cuối cùng của quá trình này là xác định các tập con có điểm số cao nhất. Mặc dù có nhiều thuật toán khác nhau, nhưng chúng đều tuân theo một quy trình chung gồm bốn bước: Sinh ra tập con (subset generation), đánh giá tập con (subset evaluation), tiêu chí dừng (stopping criteria), đánh giá kết quả (result validation) [35, 31, 28]

Bước đầu tiên sẽ xác định những tập con nào sẽ được kiểm tra, bước tiếp theo đánh giá đặc trưng bằng việc gán điểm cho các tập con, cho phép xếp

hạng chúng. Tiêu chí dừng điều chỉnh độ tập trung của thuật toán. Cuối cùng, xác nhận kết quả là đánh giá chất lượng của quy trình lựa chọn đặc trưng. Hình 2.2 mô tả các bước của quy trình này, chúng tôi sẽ giới thiệu chi tiết hơn trong phần tiếp theo.

Bước đầu tiên xác định những tập con nào sẽ được kiểm tra trong quá trình, bước tiếp theo đại diện cho hàm gán điểm cho các tập con, cho phép xếp hạng chúng. Tiêu chí dừng điều chỉnh độ tập trung của tìm kiếm. Cuối cùng, xác nhận kết quả là phần đánh giá chất lượng của giải pháp. Hình 2.3 minh họa các bước này.

2.2.1 Sinh tập con

Việc tạo các tập con đại diện cho thuật toán tìm kiếm đặc trưng và xác định một tập con tốt để đánh giá. Do đó, hai vấn đề chính được giải quyết ở bước này là: successor generation và search organization.

Successor generation

Successor generation là quá trình tạo ra các tập con mới từ tập con hiện tại. Trong thuật toán lựa chọn đặc trưng, các tập con được tạo ra bằng cách loại bỏ hoặc bổ sung các đặc trưng vào tập con hiện tại. Quá trình này được lặp lại để tạo ra các tập con mới cho đến khi đạt được một số tiêu chí dừng nhất định. Successor generation đóng vai trò quan trọng trong thuật toán lựa chọn đặc trưng, vì nó xác định các tập con mới mà thuật toán sẽ đánh giá để tìm ra tập con tốt nhất. Theo các tác giả của [31], có bốn toán tử cơ bản để giải quyết vấn đề này:

- **Chuyển tiếp (Forward):** Tập con mới được tạo ra bằng cách thêm từng đặc trưng một vào tập con.
- **Chuyển lùi (Backward):** Tập con mới được tạo ra bằng cách loại bỏ từng đặc trưng một khỏi tập con.
- **Kết hợp (Compound):** Toán tử này áp dụng k bước chuyển tiếp, theo sau bởi l bước chuyển lùi. Bằng cách làm như vậy, các vòng lặp mới giữa các đặc trưng khác nhau được khám phá [28].
- **Ngẫu nhiên (Random):** Các tập con được lựa chọn ngẫu nhiên.

Search Organization

Search Organization là cách thức tổ chức và quản lý quá trình tìm kiếm trong các thuật toán tối ưu hóa, trong đó các trạng thái và hành động được xác định để tìm kiếm đặc trưng tốt nhất. Search Organization đóng vai trò quan trọng trong việc tối ưu hóa hiệu suất thuật toán, đảm bảo rằng nó không bỏ sót đặc trưng tốt nhất và tránh tình trạng bị tối ưu hóa địa phương (local optima). Do đó, có thể xem đây là phần quan trọng nhất của toàn bộ quy trình. Các tác giả của [31] phân loại tìm kiếm thành ba loại:

- **TÌm kiếm vét cạn (Complete Search):** Hướng tìm kiếm này đảm bảo tìm ra được nghiệm tối ưu. Tìm kiếm vét cạn là một kỹ thuật tốt nếu có đủ thời gian để đi qua hết tất cả các lời giải, vì việc tìm kiếm thường dễ để thực thi và nó luôn cho ra lời giải chính xác. Nếu tìm kiếm vét cạn là quá chậm, thì có thể tính đến các thuật toán tham lam hoặc quy hoạch động. Thuật toán Fast brand & bound trong [44] là một ví dụ về tìm kiếm vét cạn.

- **Tìm kiếm tuần tự (Sequence Search)**: là một phương pháp tìm kiếm một phần tử cho trước trong một danh sách bằng cách duyệt lần lượt từng phần tử của danh sách đó cho đến lúc tìm thấy giá trị mong muốn hay đã duyệt qua toàn bộ danh sách. Các tìm kiếm này dễ dàng triển khai và thường cung cấp kết quả rất nhanh chóng. Tuy nhiên, chất lượng của các giải pháp thường là kém. Một số ví dụ về tìm kiếm này được đề cập trong [21].

- **Tìm kiếm ngẫu nhiên (Random Search):** Trong phương pháp tiếp cận này, ý tưởng là dẫn dắt tìm kiếm một cách ngẫu nhiên. Cả thuật toán Las Vegas và Las Vegas Incremental [35] đều là các ví dụ thuộc loại này.

- **Tìm kiếm di truyền (Genetic Search)**: Đây là một loại tìm kiếm khác đã tích hợp sẵn việc tạo ra các trạng thái tiếp theo. Ý tưởng của chúng là mô phỏng quá trình lựa chọn tự nhiên gồm ba toán tử: lựa chọn, đột biến và ghép cặp. Ban đầu, một số nghiệm tiêu biểu được sinh ra. Sau đó, chất lượng của từng nghiệm được đánh giá và các nghiệm tốt nhất được chọn. Trong bước tiếp theo, các nghiệm tiềm năng mới được tạo ra bằng cách kết hợp các nghiệm được bầu chọn từ giai đoạn trước. Các toán tử di truyền như ghép cặp và đột biến được sử dụng ở giai đoạn này. Quá trình này lặp lại cho đến khi kết thúc. Có nhiều loại tìm kiếm kiểu này được đề cập trong tài liệu [49].

2.2.2 Đánh giá tập con

Quá trình đánh giá và cho điểm các tập con được thực hiện. Tiêu chí đánh giá định nghĩa chất lượng của một tập con và ảnh hướng đến nghiệm tối ưu của bài toán. Cụ thể hơn, nghiệm tối ưu toàn cục (global optima) đối với một tiêu chí đánh giá nhất định, có thể không phải là nghiệm tối ưu địa phương trên một tiêu chí đánh giá khác. Có hai nhóm phân loại hàm đánh giá: đánh độc lập (independent) và đánh giá phụ thuộc (dependent) [31].

Đánh giá độc lập

Tiêu chí độc lập đánh giá chất lượng của một tập con đặc trưng dựa trên các đặc điểm của dữ liệu. Hầu hết các lần, các thước đo này được sử dụng để đánh giá chất lượng của một đặc trưng riêng biệt. Bên cạnh đó, chúng liên quan đến các phương pháp filter (đã được nhắc đến trong 1.2). Dựa trên các chỉ số được sử dụng, các tiêu chí này được chia thành nhiều loại khác nhau [35]:

- **Các chỉ số khoảng cách hoặc độ khác biệt (Distance or divergence measures)**: đánh giá khả năng của các đặc trưng để phân biệt xác suất có điều kiện giữa các lớp. Các ví dụ về các chỉ số này bao gồm độ khác biệt Jeffrey và độ khác biệt Kaga [28].
- **Các chỉ số thông tin hoặc sự không chắc chắn (Infomation or uncertainly measures)**: xác định thông tin mà một đặc trưng đóng góp cho các lớp. Khái niệm này được gọi là lợi ích thông tin và các ví dụ bao gồm entropy Shannon và tất cả các biến thể của nó [48].
- **Các chỉ số xác suất lỗi (Probability of error measures)**: ước lượng khả năng của một đặc trưng để giảm thiểu xác suất lỗi phân loại. Xác suất Bayes là ví dụ phổ biến nhất của kỹ thuật này [47].
- **Các chỉ số phụ thuộc (Dependency measures):** đánh giá khả năng của các đặc trưng trong việc dự đoán nhãn. Hệ số tương quan có thể được coi là một ví dụ [18].
- **Các chỉ số khoảng cách giữa các lớp (Interclass distance measures)**: khoảng cách trong không gian dữ liệu được sử dụng để xác định các đặc trưng tốt nhất để phân tách các lớp khác nhau. Khoảng cách Euclid là một ví dụ về kỹ thuật này [28].
- **Các chỉ số tính nhất quán (Consistency measures)**: dựa trên nguyên tắc các đặc trưng có cùng giá trị thì thuộc về cùng một lớp. Vi phạm quy tắc

này dẫn đến sự trừ điểm cho đặc trưng. Một số cách tiếp cận của loại này được đề cập trong [35].

Đánh giá phụ thuộc

Các đánh giá phụ thuộc sử dụng các thuật toán để học sau đó ước tính chất lượng của các đặc trưng. Mỗi tập con được đánh giá được sử dụng để huấn luyện mô hình, sau đó hiệu suất của mô hình được sử dụng để gán một điểm cho tập con. Trong lớp các bài toán phân loại, cách thường được sử dụng để định lượng hiệu suất của thuật toán học là sử dụng độ chính xác của dự đoán.

2.2.3 Tiêu chí dừng

Tiêu chí dừng được hiểu là điều kiện để kết thúc tìm kiếm. Một vài tiêu chí là [28]:

- Tìm kiếm đã vét cạn hoàn toàn.
- Đạt đến một ngưỡng được xác định. Ngưỡng có thể là số lượng tối thiểu hoặc tối đa của tập con đặc trưng hoặc số lượng lặp tối đa.
- Tìm thấy một tập con đủ tốt.
- Các tập con mới tiếp theo không cải thiện đáng kể các tiêu chí đánh giá.

2.2.4 Đánh giá kết quả

Sau khi quá trình hoàn tất, một tập con cuối cùng của các đặc trưng được chọn ra. Để kiểm tra xem tập các đặc trưng này có đủ tốt cho bài toán phân loại hay không, quá trình đánh giá kết quả là rất quan trọng. Một phương pháp trực tiếp để đánh giá kết quả thu được bằng việc sử dụng kiến thức trước đó về bộ dữ liệu. Nếu có thông tin về các đặc trưng quan trọng, có thể đánh giá bằng cách so sánh các đặc trưng được chọn với những đặc trưng được biết là quan trọng. Trong các trường hợp này, thông tin về các đặc trưng không quan trọng hoặc trùng lặp cũng quan trọng, chủ yếu bởi vì sự hiện diện của chúng phản ánh chất lượng của tập các đặc trưng được chọn.

Tình huống phổ biến nhất là thiếu thông tin về bộ dữ liệu. Do đó, cần sử dụng các kỹ thuật khác nhau. Ví dụ, có thể sử dụng hiệu suất của tập đặc trưng cuối cùng trên mô hình và so sánh nó với với toàn bộ tập đặc trưng hoặc bất kỳ tập con đặc trưng nào khác. Ngoài ra, một cách thông thường là sử dụng các mô hình thuật toán khác nhau và so sánh hiệu suất [8].

2.3 Sử dụng tính toán song song để lựa chọn đặc trưng

Như đã biết, việc lựa chọn ra một tập hợp các đặc trưng tốt cho bài toán phân loại cụ thể là một bài toán không dễ. Thông thường, điều này yêu cầu các nhà nghiên cứu phải thực hiện nhiều lần thử nghiệm cho đến khi đạt được kết quả như mong muốn. Một cách tiếp cận thông thường là thay đổi các tham số trên các mô hình hoặc kiểm tra các mô hình khác nhau để so sánh kết quả. Ngoài ra, quá trình lựa chọn đặc trưng có thể cần một lượng lớn thời gian tùy thuộc vào kích thước của bộ dữ liệu và mô hình được chọn.

Vì thế, tính toán song song đã nổi lên như một giải pháp tiềm năng để giải quyết vấn đề này. Việc thực hiện nhiều tính toán đồng thời để giải quyết một bài toán dựa trên nguyên tắc rằng các bài toán lớn có thể được chia thành các vấn đề nhỏ hơn [10].Xem xét kỹ hơn về quy trình tổng quát cho lựa chọn đặc trưng, nó bao gồm việc tạo ra và đánh giá một lượng lớn các tập con cho đến khi đạt được điều kiện dừng, và thu được tập con tốt nhất. Vấn đề này có thể dễ dàng chia thành các tác vụ nhỏ hơn, trong đó mỗi tác vụ được xác định là quy trình tổng thể trên mỗi tập con, điều này lý tưởng cho các kỹ thuật tính toán song song, và do vậy kích thích các nhà nghiên cứu khai thác tính song song trong các thuật toán lựa chọn đặc trưng để cải thiện thời gian thực thi của chúng. Ví dụ, Azmandian và các cộng sự [4] đã sử dụng đơn vị xử lý đồ họa để tăng tốc thuật toán lựa chọn đặc trưng của họ. Lopez và các cộng sự [32] cũng sử dụng tính song song để tăng tốc một tìm kiếm phân tán để đạt được hiệu suất tốt hơn về thời gian thực thi và chất lượng.

2.4 Sơ lược về hướng tiếp cận

Có rất nhiều thuật toán lựa chọn đặc trưng đã được phát triển từ rất lâu đến tận thời điểm hiện tại. Trong đó, lựa chọn tiến (forward feature selection), lựa chọn lùi (backward feature selection) và lựa chọn từng bước (stepwise feature selection) là ba thuật toán rất phổ biến. Trong phần này, chúng tôi sẽ tóm tắt lại ba kĩ thuật này, và trình bày chi tiết cách chúng tôi sử dụng trong các thuật toán 1, 2, và 3.

2.4.1 Thuật toán lựa chọn chuyển tiến

Thuật toán lựa chọn chuyển tiến được sử dụng rất rộng rãi vì sự hiệu quả trong việc tính toán của nó, cùng với khả năng xử lý hiệu quả các vấn đề bao

gồm việc số lượng đặc trưng vượt quá số lượng quan sát. Tuy nhiên, một số đặc trưng có thể xuất hiện dư thừa sau khi đã lựa chọn các đặc trưng khác. Về các điều kiện đủ để lựa chọn tiến nhằm khôi phục mô hình ban đầu và tính ổn định của nó, chúng tôi tham khảo từ [45] và [11]. Dưới đây là chi tiết thuật toán

---

Thuật toán 1. Lựa chọn tiến
(Forward Feature Selection)

Input: Một tập dữ liệu gồm p đặc trưng $f_1, f_2, ..., f_p$, tham số tiến là α.
Output: Một tập hợp R gồm các đặc trưng được chọn.

$R \leftarrow \emptyset$
$S \leftarrow \{f_1, f_2, \ldots, f_p\}$
while True **do**
    $f_j \leftarrow$ đặc trưng hữu ích nhất trong S
    if mô hình được cải thiện tốt hơn một lượng là α sau khi thêm vào f_j **then**
        $R \leftarrow R \cup \{f_j\}$
        $S \leftarrow S \setminus \{f_j\}$
    else
        return R
    end if
end while

2.4.2 Thuật toán lựa chọn chuyển lùi

Thuật toán lựa chọn chuyển lùi [23] được trình bày chi tiết trong thuật toán 2. Bắt đầu với toàn bộ đặc trưng, sau đó lần lượt loại bỏ các đặc trưng ít hữu ích nhất từ từ, mỗi lần một đặc trưng. Yêu cầu của bfs bảo đảm những đặc trưng dư thừa được loại bỏ khỏi mô hình. Tuy nhiên, thuật toán lựa chọn chuyển lùi thì có tốc độ tính toán khá chậm, và chậm hơn nhiều khi so với thuật toán lựa chọn chuyển tiến.

Thuật toán 2. Lựa chọn lùi
(Backward Feature Selection)

Input: Một tập dữ liệu gồm p đặc trưng $f_1, f_2, ..., f_p$, tham số lùi là β.
Output: Một tập hợp R gồm các đặc trưng được chọn.

$R \leftarrow \{f_1, f_2, \ldots, f_p\}$
while True, **do**

3: $f_j \leftarrow$ là đặc trưng ít hữu ích nhất trong R.
4: **if** giá trị mất mát của mô hình sau khi loại f_j là nhỏ hơn β **then**
5: $R \leftarrow R \setminus \{f_j\}$
6: **else**
7: **return** R
8: **end if**
9: **end while**

2.4.3 Thuật toán lựa chọn từng bước

Ngoài hai thuật toán được nêu ra ở trên, một thuật toán khác, là phiên bản kết hợp của cả thuật toán lựa chọn chuyển tiến và thuật toán lựa chọn chuyển lùi là thuật toán lựa chọn từng bước, được trình bày chi tiết trong thuật toán 3. Trong cách tiếp cận này, các đặc trưng được thêm vào mô hình một cách có tuần tự như trong lựa chọn tiến. Tuy nhiên, sau khi thêm vào các đặc trưng mới, phương pháp này có thể loại bỏ bất kỳ đặc trưng nào mà có vẻ không còn phù hợp.

Thuật toán 3. Lựa chọn từng bước
(Stepwise Feature Selection)

Input: Một tập hợp gồm p đặc trưng $f_1, f_2, ..., f_p$, tham số tiến là α, tham số lùi là β.
Output: Một tập hợp R gồm các đặc trưng được chọn.

1: $R \leftarrow \emptyset$
2: $S \leftarrow \{f_1, f_2, \ldots, f_p\}$
3: **while** True, **do**
4: $f_j \leftarrow$ là đặc trưng hữu ích nhất trong S
5: **if** mô hình cải thiện hơn một lượng là α sau khi thêm f_j **then**
6: $R \leftarrow R \cup \{f_j\}$
7: $S \leftarrow S \setminus \{f_j\}$
8: **while** True **do**
9: $f_k \leftarrow$ là đặc trưng ít hữu ích nhất trong R
10: **if** hiệu suất của mô hình giảm một lượng nhỏ hơn β sau khi loại bỏ f_j. **then**
11: $R \leftarrow R \setminus \{f_k\}$
12: **else** break
13: **end if**
14: **end while**
15: **else**
16: **return** R

17: **end if**
18: **end while**

2.5 Một số khái niệm toán học

2.5.1 Vết của ma trận

Định nghĩa 2.5.1. Vết của ma trận (trace) là tổng các phần tử trên đường chéo chính của ma trận vuông. Ký hiệu là $trace(A)$, trong đó A là ma trận vuông.

Cho A là ma trận vuông kích thước $n \times n$, với các phần tử a_{ij}, vết của A được tính như sau:

$$trace(A) = \sum_{i=1}^{n} a_{ii} \tag{2.1}$$

Vết của ma trận có một số tính chất quan trọng:

- $trace(A \pm B) = trace(A) \pm trace(B)$ với A và B là hai ma trận cùng kích thước.
- $trace(cA) = c * trace(A)$, trong đó c là một hằng số và A là một ma trận.
- $trace(A') = trace(A)$, trong đó A' là ma trận chuyển vị của A.
- $trace(AB) = trace(BA)$, trong đó A và B là hai ma trận hợp lệ để nhân với nhau.

Vết của ma trận được sử dụng trong nhiều ứng dụng trong toán học, đại số tuyến tính, thống kê và học máy, bao gồm tính toán độ đo Frobenius norm, tìm dạng đường chéo hóa của ma trận và phân tích thành phần chính (PCA).

2.5.2 Trung bình bình phương sai số

Trung bình bình phương sai số (Mean Squared Error - MSE) là một phép đo chất lượng trong dự đoán, được sử dụng để đánh giá độ chính xác của mô hình dự đoán hoặc ước lượng. Nó đo lường mức độ khác biệt giữa các giá trị dự đoán và giá trị thực tế của dữ liệu, và thường được sử dụng trong học máy và thống kê để đánh giá hiệu suất của các mô hình hồi quy, phân loại và dự đoán.

Định nghĩa 2.5.2. Cho tập dữ liệu gồm n mẫu với các giá trị là $(y_1, \ldots, y_n)$ và các giá trị dự đoán là $(\hat{y}_1, \ldots, \hat{y}_n)$. Khi đó,

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

MSE là một phép đo không âm, với giá trị càng gần 0, mô hình dự đoán càng chính xác. Một ưu điểm của MSE so với các phép đo khác như trung bình của sai số tuyệt đối (Mean Absolute Error - MAE) là nó đánh giá nặng hơn các lỗi lớn hơn, do tính chất của bình phương sai số. Tuy nhiên, do vậy mà MSE không cùng đơn vị giống với đại lượng cần dự đoán. Để khắc phục điều này, người ta thường sử dụng căn bậc hai của trung bình bình phương sai số (Root Mean Squared Error - RMSE), giúp đưa về cùng đơn vị với giá trị dự đoán và giá trị thực tế.

2.5.3 Khoảng cách Mahalanobis

Khoảng cách Mahalanobis là một phép đo khoảng cách giữa các điểm trong không gian đa chiều, được đặt theo tên của P. C. Mahalanobis, một nhà thống kê Ấn Độ. Khoảng cách Mahalanobis có tính chất đặc biệt so với các phép đo khoảng cách khác, như khoảng cách Euclid, bởi vì nó tính toán khoảng cách giữa các điểm dựa trên mối tương quan giữa các biến và độ lớn của chúng.

Khoảng cách Mahalanobis được tính bằng công thức sau:

$$D^2 = (x - \mu)' \cdot S^{-1} \cdot (x - \mu), \tag{2.2}$$

trong đó:

- D^2 là bình phương của khoảng cách Mahalanobis.
- x là một vector đại diện cho một điểm trong không gian đa chiều.
- μ là vector giá trị trung bình của từng biến trong tập dữ liệu.
- S là ma trận hiệp phương sai (covariance) của tập dữ liệu.

Ưu điểm của khoảng cách Mahalanobis so với các phép đo khoảng cách khác là nó có thể xử lý hiệu quả các biến có tính chất tương quan cao và độ lớn khác nhau. Khoảng cách Mahalanobis thường được sử dụng trong các bài toán nhận dạng mẫu, phân tích thành phần chính (PCA) và phát hiện ngoại lai (outlier detection).

2.5.4 Phân kỳ Kullback-Leibler

Phân kỳ Kullback-Leibler là một phép đo không đối xứng giữa hai phân phối xác suất. Nó được sử dụng để đo mức độ khác biệt giữa hai phân phối xác suất và chủ yếu được áp dụng trong lý thuyết thông tin, thống kê và học máy. Phân kỳ Kullback-Leibler được đặt theo tên của hai nhà khoa học Solomon Kullback và Richard Leibler, người đã đề xuất nó vào năm 1951.

Định nghĩa 2.5.3. Cho $P(x)$ và $Q(x)$ là xác suất của một sự kiện x lần lượt trong phân phối P và Q, ta có

$$D_{KL}\left(P||Q\right) = \sum_{x} P(x) \cdot \log \frac{P(x)}{Q(x)}$$

Ký hiệu $D_{KL}\left(P||Q\right)$ là phân kỳ Kullback-Leibler đo lường mức độ bất đồng về thông tin khi sử dụng phân phối Q để ước lượng hoặc mô phỏng phân phối P.

Lưu ý rằng phân kỳ Kullback-Leibler không đối xứng, tức là $D_{KL}\left(P||Q\right)$ không bằng $D_{KL}\left(Q||P\right)$. Nó cũng không thỏa mãn bất đẳng thức tam giác, do đó không phải là một khoảng cách theo nghĩa toán học chặt chẽ.

Phân kỳ Kullback-Leibler được sử dụng trong nhiều ứng dụng như phân tích phân cụm, phát hiện ngoại lai, nén dữ liệu, ước lượng hợp lý tối đa và các thuật toán học máy liên quan đến phân phối xác suất.

2.6 Mô hình LDA (Linear Discriminant Analysis)

Mô hình LDA (Linear Discriminant Analysis) là một mô hình học máy giám sát được sử dụng trong các bài toán phân loại. LDA tìm một đường phẳng tuyến tính để phân tách các lớp dữ liệu sao cho khoảng cách giữa các lớp là lớn nhất và phương sai bên trong mỗi lớp là nhỏ nhất. Nói cách khác, LDA cố gắng tối đa hóa giữa tỷ lệ giữa phương sai giữa các lớp và phương sai bên trong mỗi lớp. Để đạt được mục tiêu này, LDA sử dụng hai ma trận quan trọng: ma trận phân tán giữa các lớp (between-class scatter matrix) và ma trận phân tán trong lớp (within-class scatter matrix).

Định nghĩa 2.6.1 (Ma trận phân tán giữa các lớp)**.** Cho một tập dữ liệu gồm C lớp, và có n_i quan sát cho lớp thứ i, và đặt $\boldsymbol{x}_{ij}$ là mẫu thứ j của lớp thứ i. Khi đó, ma trận phân tán giữa các lớp được tính như sau

$$S_b = \sum_{i=1}^{C} n_i(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'. \tag{2.3}$$

Ma trận phân tán giữa các lớp dùng để đo lường mức độ phân tán giữa các lớp trong không gian dữ liệu và được sử dụng để tìm ra một không gian chiếu tối ưu, giúp giữ cho các lớp phân biệt tốt hơn.

Định nghĩa 2.6.2 (Ma trận tán xạ trong lớp)**.** Cho một tập dữ liệu gồm C lớp, và có n_i quan sát cho lớp thứ i, và đặt $\boldsymbol{x}_{ij}$ là mẫu thứ j của lớp thứ i. Khi đó, ma trận tán xạ trong lớp được tính như sau

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)'. \tag{2.4}$$

Ma trận phân tán trong lớp dùng để đo lường mức độ phân tán của các điểm dữ liệu bên trong mỗi lớp và được sử dụng để tìm ra một không gian chiếu tối ưu, giúp giữ cho các điểm dữ liệu trong cùng một lớp gần nhau.

Định nghĩa 2.6.3. Cho một tập dữ liệu gồm C lớp, phép chiếu tập dữ liệu xuống một đường thẳng là một ma trận hệ số $\boldsymbol{W}$. Khi đó, ta có hàm mục tiêu

$$J(\boldsymbol{W}) = \frac{\boldsymbol{W}' S_b \boldsymbol{W}}{\boldsymbol{W}' S_w \boldsymbol{W}} \tag{2.5}$$

Thuật toán LDA đi tìm giá trị lớn nhất của $J(\boldsymbol{W})$. Nghiệm W được tìm bằng cách giải phương trình đạo hàm hàm mục tiêu bằng 0. Phương trình này tương đương với $J(\boldsymbol{W})\boldsymbol{W} = \left(\boldsymbol{S}_w^{-1}\boldsymbol{S}_b\right)\boldsymbol{W}$.

Từ đây suy ra, mỗi cột của $\boldsymbol{W}$ là một véc-tơ riêng của $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ ứng với trị riêng lớn nhất.

Vậy các cột của $\boldsymbol{W}$ cần phải độc lập tuyến tính. Câu hỏi đặt ra là: Có nhiều nhất bao nhiêu vector riêng độc lập tuyến tính ứng với trị riêng lớn nhất của $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$?

Số lượng lớn nhất các vector riêng độc lập tuyến tính ứng với 1 trị riêng chính là số chiều của không gian riêng ứng với trị riêng đó, và không thể lớn hơn hạng của ma trận. Ta có một bổ đề quan trọng:

Bổ đề 2.6.4. $\text{rank}\,(\boldsymbol{S}_B) \leq C - 1$

Chương 3

Tiêu chí vết - Tiêu chí lựa chọn đặc trưng

Bảng 3.1: Bảng ký hiệu

Symbol	Description
C	number of classes
A'	the transpose of matrix A
$\bar{\boldsymbol{x}}_i$	the class mean of the i^{th} class
$\bar{\boldsymbol{x}}$	the overall mean
R	the set of selected features
$\bar{\boldsymbol{x}}_{iRf}$	the mean of the i^{th} class when the selected features are $\{R, f\}$
$\bar{\boldsymbol{x}}_{Rf}$	the overall mean when the selected features are $\{R, f\}$
n_i	number of observations in the i^{th} class
x_{ij}	the j^{th} sample of the i^{th} class
S_R	The value of S_w when R is the set of selected features
t_R	$trace(S_w^{-1}S_b)$ when R is the set of selected features

3.1 Tiêu chí vết

Chương này sẽ trình bày chi tiết cơ sở của việc sử dụng tiêu chí vết cho việc lựa chọn đặc trưng.

Tiêu chí vết là một thước đo hữu ích để phân lớp hữu ích để lựa chọn đặc trưng cho lớp bài toán toán phân loại (chi tiết hơn trong [14, 24]). Có nhiều phiên bản định nghĩa tiêu chí vết, chúng là tương đương nhau. Tuy nhiên, để phù hợp với lớp bài toán phân loại, chúng tôi sẽ giới thiệu định nghĩa sau

Định nghĩa 3.1.1. Cho một tập dữ liệu gồm C lớp, và có n_i quan sát cho lớp thứ i, và đặt $\mathbf{x}_{ij}$ là mẫu thứ j của lớp thứ i. Khi đó, ta có định nghĩa tiêu chí vết là

$$trace(S_w^{-1}S_b), \tag{3.1}$$

trong đó,

$$S_b = \sum_{i=1}^{C} n_i(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})', \tag{3.2}$$

và

$$S_w = \sum_{i=1}^{C}\sum_{j=1}^{n_i}(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)'. \tag{3.3}$$

S_b còn được gọi là ma trận phân tán giữa các lớp (between-class scatter matrix) và S_w là ma trận tán xạ trong lớp.

Ở đây A' là ma trận chuyển vị của ma trận A, và $\bar{\boldsymbol{x}}_i$ là giá trị trung bình của lớp thứ i, và $\bar{\boldsymbol{x}}$ là giá trị trung bình của toàn bộ dữ liệu. Nghĩa là,

$$\bar{\boldsymbol{x}}_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(\bar{\boldsymbol{x}}_{ij} - \bar{\boldsymbol{x}}_i)$$

$$\bar{\boldsymbol{x}} = \frac{1}{\sum_{i=1}^{C} n_i}\sum_{i=1}^{C}\sum_{j=1}^{n_i}(\bar{\boldsymbol{x}}_{ij} - \bar{\boldsymbol{x}}_i).$$

Từ đây, ta có kết quả sau

Định lý 3.1.2.

$$trace(S_w^{-1}S_b) = \sum_{i=1}^{C} n_i(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'S_w^{-1}(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}) \tag{3.4}$$

Chứng minh.

$$\begin{aligned}
trace(S_w^{-1}S_b) &= trace\left(S_w^{-1}\sum_{i=1}^{C} n_i(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'\right) \\
&= trace\left(\sum_{i=1}^{C} n_i S_w^{-1}(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'\right) \\
&= \sum_{i=1}^{C} n_i\, trace\left(S_w^{-1}(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'\right) \\
&= \sum_{i=1}^{C} n_i\, trace\left((\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'S_w^{-1}(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})\right)
\end{aligned}$$

$$= \sum_{i=1}^{C} n_i(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})' S_w^{-1}(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})$$

□

Từ định lý trên, ta có thể thấy rằng tiêu chí vết 3.4 có thể được xem là tổng bình phương khoảng cách Mahalanobis từ trung bình của mỗi class đến trung bình của toàn bộ dữ liệu. Hơn nữa khi số lượng lớp là hai, tiêu chí 3.4 có thể được xem là ước lượng thực nghiệm của phân kỳ Kullback–Leibler cho hai phân phối chuẩn có cùng ma trận hiệp phương sai [26].

Vì tiêu chí này đo khả năng tách biệt của các lớp, nên chúng tôi sẽ định hướng vào việc tối đa nó. Để đơn giản ký hiệu, chúng tôi sẽ viết $\{R, f\}$ thay cho $R \cup \{f\}$, và $t_{R,f}$ thay cho $t_{\{R,f\}}$. Ta có định lý sau

Định lý 3.1.3. *Với R là tập hợp các đặc trưng được chọn, và f là một đặc trưng tùy ý mà không nằm trong R. Đặt S_{Rf} là giá trị của S_w khi $\{R, f\}$ là tập hợp các đặc trưng được chọn, và S_R là giá trị của S_w khi R là tập hợp các đặc trưng được chọn*

$$S_{Rf} = \begin{pmatrix} S_R & v \\ v' & u \end{pmatrix}, \tag{3.5}$$

với $u \in \mathbb{R}^+, v \in \mathbb{R}^{|R|\times 1}$, $|R|$ số lượng phần tử trong R.
Nếu $u - v'S_R^{-1}v > 0$ thì

$$t_{R,f} \geq t_R. \tag{3.6}$$

Ngược lại, nếu $u - v'S_R^{-1}v < 0$ thì

$$t_{R,f} < t_R. \tag{3.7}$$

Chứng minh. Đặt R là tập hợp các đặc trưng đưa vào mô hình. Đặt $\bar{\boldsymbol{x}}_{iR}, \bar{\boldsymbol{x}}_R$ lần lượt là trung bình của lớp thứ i, và trung bình toàn bộ dữ liệu của tập hợp R. Tương tự, đặt $\bar{\boldsymbol{x}}_{iRf}, \bar{\boldsymbol{x}}_{Rf}$ lần lượt là trung bình của lớp thứ i và trung bình của toàn bộ dữ liệu toàn bộ dữ liệu của tập hợp $\{R, f\}$. Khi đó, ta có

$$\bar{\boldsymbol{x}}_{iRf} = \begin{pmatrix} \bar{\boldsymbol{x}}_{iR} \\ \bar{\boldsymbol{x}}_{if} \end{pmatrix}, \quad \bar{\boldsymbol{x}}_{Rf} = \begin{pmatrix} \bar{\boldsymbol{x}}_R \\ \bar{\boldsymbol{x}}_f \end{pmatrix}, \tag{3.8}$$

với $\bar{\boldsymbol{x}}_{if}, \bar{\boldsymbol{x}}_f$ lần lượt là trung bình lớp thứ i và trung bình của toàn bộ dữ liệu của đặc trưng f.

Nếu $u \neq v'S_R^{-1}v$ thì $M = (u - v'S_R^{-1}v)^{-1}$ tồn tại. Do đó,

$$S_{Rf}^{-1} = \begin{pmatrix} S_R^{-1} + S_R^{-1}vMv'S_R^{-1} & -S_R^{-1}vM \\ -Mv'S_R^{-1} & M \end{pmatrix} \tag{3.9}$$

Tiếp theo, đặt

$$\begin{aligned}\eta_{iRf} &= (\bar{\boldsymbol{x}}_{iRf} - \bar{\boldsymbol{x}}_{Rf})' S_{Rf}^{-1} (\bar{\boldsymbol{x}}_{iRf} - \bar{\boldsymbol{x}}_{Rf}) \\ &= \eta_{iR} + (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v M v' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R) \\ &- 2(\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v M (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f) \\ &+ (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f)' M (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f),\end{aligned}$$

với $\eta_{iR} = (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)$. Để ý rằng $M \in \mathbb{R}, M \neq 0$. Do đó,

$$\begin{aligned}\frac{\eta_{iRf} - \eta_{iR}}{M} &= (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v v' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R) \\ &- 2(\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f) \\ &+ (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f)' (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f).\end{aligned}$$

Áp dụng bất đẳng thức Cauchy-Schwarz, $||a||^2 + ||b||^2 \geq 2\langle a, b\rangle$, với $a = (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R) S_R^{-1} v$, và $b = (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f)$, ta thấy rằng

$$\frac{\eta_{iRf} - \eta_{iR}}{M} \geq 0.$$

Vì $t_{R,f} - t_R = \sum_{i=1}^{C} (\eta_{iRf} - \eta_{iR})$, nên nếu $u - v' S_R^{-1} v > 0$ thì $t_{R,f} \geq t_R$, ngược lại nếu $u - v' S_R^{-1} v < 0$ thì $t_{R,f} < t_R$. □

Ý nghĩa của định lý 3.1.3 là việc cải thiến hay suy thoái của một mô hình sau khi thêm một đặc trưng không chỉ phụ thuộc vào bản thân các đặc trưng mà còn phụ thuộc vào sự tương quan giữa đặc trưng đó với các đặc trưng khác đã có trong mô hình. Hơn nữa, từ phương trình 3.7, ta có thể thấy rằng thêm một đặc trưng có thể làm giảm đi khả năng phân lớp của một mô hình. Vì thế, lựa chọn đặc trưng là cần phải loại bỏ các đặc trưng mà đóng góp của chúng cho hiệu suất của mô hình là không đáng kể. Thuộc tính này là mong muốn và không phải tất cả các tiêu chí lựa chọn đặc trưng đều có thể đáp ứng được. Ví dụ, với trung bình bình phương sai số (mean square error - MSE), việc thêm vào các đặc trưng, dù có dư thừa hay không cũng không làm tăng MSE. Định lý dưới đây sẽ làm rõ điều này.

Định lý 3.1.4. *Đặt MSE_X là giá trị trung bình bình phương sai số khi ta hồi quy Y dựa trên X. Giả sử T chứa các đặc trưng tùy ý được thêm vào mô hình và đặt*

$$U = (X \quad T). \tag{3.10}$$

Đặt MSE_U là giá trị trung bình bình phương sai số khi ta hồi quy Y dựa trên U. Khi đó, nếu $M = (T'T - T'X(X'X)^{-1}X'T)^{-1}$ tồn tại thì $MSE_U \leq MSE_X$.

Chứng minh. Đặt $H = X(X'X)^{-1}X'$, khi đó ta có $H^2 = H$, nghĩa là H là ma trận lũy đẳng, do đó $I - H$ cũng là ma trận lũy đẳng, thật vậy

$$(I-H)^2 = I - 2H + H^2 = I - 2H + H = I - H.$$

Do đó, tổng bình phương sai số khi hôi quy Y trên X là

$$SSE_X = (Y - \hat{Y}_X)'(Y - \hat{Y}_X) = Y'(I-H)Y$$

với $\hat{Y}_X$ là vector giá trị dự đoán của mô hình khi hồi quy Y trên X.

Chú ý rằng M là ma trận đối xứng, khi đó đặt $\hat{Y}_U$ là vector giá trị dự đoán của mô hình khi hồi quy Y trên U. Do đó,

$$\begin{aligned}
&(U'U)^{-1}U'Y \\
&= HY + HTMT'HY - HTMT'Y - TMT'HY + TMT'Y \\
&= HY - (I-H)TMT' + (I-H)TMT'Y.
\end{aligned}$$

Suy ra

$$\begin{aligned}
&Y - \hat{Y}_U \\
&= Y - \hat{Y}_X + (I-H)TMT'HY - (I-H)TMT'Y \\
&= Y - \hat{Y}_X - (I-H)TMT'(I-H)Y
\end{aligned}$$

Nên tổng bình phương sai số khi hồi quy Y trên U là

$$\begin{aligned}
SSE_U &= (Y - \hat{Y}_U)'(Y - \hat{Y}_U) \\
&= SSE_X - 2Y'(I-H)TMT'(I-H)(Y - \hat{Y}_X) \\
&\quad + Y'(I-H)TMT'(I-H)TMT'(I-H)Y
\end{aligned}$$

Hơn nữa, ta cũng có $(I-H)TMT'$ là ma trận lũy đẳng, và $Y'(I-H)TMT'(I-H)Y \geq 0$. Vì thế,

$$\begin{aligned}
&Y'(I-H)TMT'(I-H)TMT'(I-H)Y \\
&= Y'(I-H)TMT'(I-H)Y \\
&\leq 2Y'(I-H)TMT'(I-H)Y.
\end{aligned}$$

Suy ra, $SSE_U \leq SSE_X$. Lại có $U = (XT)$ có nghĩa là số lượng mẫu trên X bằng với số lượng mẫu trên T, nên $MSE_U \leq MSE_X$. □

3.2 Thuật toán lựa chọn đặc trưng song song dựa trên tiêu chí vết

Trong mục này, chúng tôi sẽ trình bày chi tiết về thuật toán lựa chọn đặc trưng song song dựa trên tiêu chí vết (PFST). Hơn nữa, để có được các thuộc tính

mong muốn của tiêu chí vết như đã đề cập ở trên, phương pháp này sẽ dựa trên các nhận xét sau.

3.2.1 Cơ sở sử dụng tiêu chí vết

Đầu tiên, như đã trình bày trong phần 2.4, việc lựa chọn chuyển tiến nhanh hơn lựa chọn chuyển lùi bởi vì nó liên tục thêm các đặc trưng phù hợp nhất vào mô hình. Tuy nhiên, ta vẫn phải chịu một chi phí tính toán tương đối lớn. Một cách tiềm năng để tăng tốc độ xử lý của quá trình này là áp dụng thuật toán Early Dropping (Early Dropping heuristic) [7]. Thuật toán này sẽ loại bỏ khỏi trong các lần lặp lại tiếp theo các đặc trưng được cho là không có khả năng làm tăng hiệu suất của mô hình. Trong bài báo [7], nhóm tác giả đã đánh giá khả năng đó bằng trị số p ($p-$value). Tuy nhiên, việc tính toán trị số p đòi hỏi sử dụng bootstrap rất tốn kém chi phí tính toán. Chúng tôi sẽ tiếp cận theo một hướng khác, chúng tôi sử dụng thuật toán Early Dropping và dùng tiêu chí vết để làm công cụ đánh giá. Điều này không yêu cầu phải lặp lại nhiều lần trong cả dữ liệu cho mỗi bước tiến như việc sử dụng trị số p.

Thứ hai, lựa chọn đặc trưng từng bước dường như là một biện pháp khắc phục sai số trong lựa chọn tiến. Lựa chọn đặc trưng từng bước thêm các đặc trưng vào mô hình một cách tuần tự như trong lựa chọn tiến. Hơn nữa, sau khi thêm vào một đặc trưng mới, cách tiếp cận này sẽ loại bỏ khỏi mô hình các đặc trưng mà không còn quan trọng dựa theo tiêu chí lựa chọn đặc trưng đang được sử dụng. Tuy nhiên, điều này sẽ làm tăng chi phí tính toán khi kết hợp các bước lùi để loại bỏ các đặc trưng không cần thiết. Trong bài báo [50], nhóm tác giả có đề xuất một thuật toán là chỉ lùi một bước khi bình phương sai số (squared error) giảm không quá một nửa mức giảm bình phương sai số trong các bước tiến trước đó. Thực tế, bài báo [36] đã chỉ ra một ví dụ mô phỏng để cho thấy rằng việc lựa chọn đặc trưng từng bước hiếm khi thực hiện một bước lùi. Vì vậy, trong cách tiếp cận của chúng tôi, chúng tôi sẽ thêm các đặc trưng đến khi đạt được mô hình thỏa mãn, và sau đó sử dụng lựa chọn lùi để loại bỏ các đặc trưng dư thừa.

Thứ ba, với tập dữ liệu chỉ có một đặc trưng, thì $trace(S_w^{-1}S_b)$ có thể được rút gọn thành

$$t = \frac{\sum_{i=1}^{C} n_i(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})^2}{\sum_{i=1}^{C}\sum_{j=1}^{n_i}(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)^2}, \tag{3.11}$$

còn với tập dữ liệu có những đặc trưng đã được chuẩn hóa thì S_b có thể được rút gọn thành $S_b = \sum_{i=1}^{C} n_i \bar{x}_i \bar{x}_i'$. Đặt

$$t_R = trace(S_w^{-1}S_b) \tag{3.12}$$

với R là tập hợp các đặc trưng được chọn trong mô hình.

3.2.2 Thuật toán

Thuật toán 4. Thuật toán tiến-lùi song song với loại bỏ sớm
(Parallel forward-backward algorithm with early dropping)

Input: Tập hợp A gồm tất cả các đặc trưng. Các feature được chia vào các khối $F_1, ..., F_B$, và nhãn Y tương ứng, tham số tiến là α, tham số lùi β, tham số loại bỏ sớm là γ, số bước tối đa sau khi bắt đầu lại là maxRef.
Output: Tập hợp R các đặc trưng được chọn.

Quy trình:

$f_b \leftarrow \arg\max_f \{t_{\{f\}} : f \in F_b\}, b = 1, ..., B$ chạy song song trên B khối.
$R \leftarrow \{f_1, ..., f_B\}$.
 # Forward with forward-dropping stage:
while $\bigcup_{1 \le b \le B} F_b \neq \emptyset$ **do**
    $f_b, F_b \leftarrow OneForwardDropping(F_b, Y, \alpha, \gamma)$ (chạy song song trên B khối).
    $R \leftarrow R \cup \{f_1, ..., f_B\}$
end while
 # Re-forward stage:
Bắt đầu lại với tập hợp $A \setminus R$ và lại chia chúng vào B khối $F_1, ..., F_B$.
runs $\leftarrow 0$
while runs $<$ maxRef & $\bigcup_{1 \le b \le B} F_b \neq \emptyset$ **do**
    $f_b, F_b \leftarrow \text{OneReforward}(F_b, Y, \alpha)$ (chạy song song trên B khối)
    $R \leftarrow R \cup \{f_1, ..., f_B\}$
    runs $\leftarrow$ runs $+1$
end while
 # Backward stage:
Chia R thành B khối.
$f_b \leftarrow \arg\max_{f \in F_b} t_{R \setminus \{f\}}, b = 1, ...B$ chạy song song trên B khối
$f_r \leftarrow \arg\max_{f \in \{f_1, ..., f_B\}} t_{R \setminus \{f\}}$
if $t_R - t_{R \setminus \{f_r\}} < \beta$ **then**
    $R \leftarrow R \setminus \{f_r\}$
else
    break
end if
return R

Thuật toán chúng tôi đề xuất là thuật toán PFST được trình bày trong thuật toán 4. Chúng tôi trước tiên chia các đặc trưng thành B khối để chạy song song B khối. Bắt đầu với tập hợp các đặc trưng được chọn R là tập rỗng. Do đó, chúng tôi tính $\arg\max_f \{t_f : f \in F_b\}, b = 1, ..., B$ song song trên B khối và

sinh ra một tập hợp R đầu tiên.

Sau đó, chúng tôi sử dụng thuật toán 5 để lần lượt thêm vào các đặc trưng tốt với mô hình và loại bỏ các đặc trưng không quan trọng khỏi các lần lặp lại tiếp theo (ta gọi bước này là *forward-dropping*). Cụ thể hơn, thuật toán loại bỏ tất các đặc trưng không thể cái thiện được tiêu chí vết một lượng lớn hơn β trong một khối. Điều này giúp giảm được chi phí tính toán của việc quét lại các đặc trưng mà dường như chúng không thể cải thiện được mô hình khi so sánh với các đặc trưng khác.

Tuy nhiên, một nhược điểm của cách làm này là ta vô tình loại bỏ đi một đặc trưng tưởng chừng là không quan trọng. Nghĩa là, ở bước này nó không quan trọng, nhưng khi kết hợp được với một số đặc trưng khác thì nó sẽ giúp cải thiện tốt hơn cho mô hình. Vì thế, sau khi những bước này kết thúc, chúng tôi sử dụng thuật toán 6 để chạy lại từ đầu với tập hợp các đặc trưng là $P = A \setminus R$ và lại chia chúng vào B khối. Sau đó tiến hành lựa chọn lần nữa và lần này không loại bỏ những đặc trưng sớm (ta gọi bước này là *re-forward*). Vì hầu hết các đặc trưng quan trọng đã có trong mô hình, nên giai đoạn này cũng sẽ không tốn quá nhiều chi phí tính toán như việc chạy song song lựa chọn tiến. Thực nghiệm cũng cho thấy, thuật toán PFST mất ít thời gian hơn so với các kĩ thuật, thuật toán khác.

Cuối cùng, việc đưa vào các đặc trưng mới, có thể làm cho những đặc trưng được đưa vào trước đó trở nên dư thừa, lý do giải thích đơn giản nhất có thể là do tính tương quan cao của các đặc trưng. Do đó, giai đoạn cuối cùng của thuật toán PFST sẽ cố gắng loại bỏ các đặc trưng dư thừa nếu có. Trong giai đoạn này, chúng tôi tiến hành lựa chọn lùi để loại bỏ các đặc trưng dư thừa còn lại trong mô hình (ta gọi bước này là *backward*). Nhắc lại, điều này giúp tránh phát sinh chi phí tính toán cho việc kiểm tra lùi sau mỗi lần đưa vào một đặc trưng mới. Điều này khác với các bước lùi trong lựa chọn từng bước.

Chú ý rằng một giá trị γ cao thì sẽ phải loại bỏ nhiều đặc trưng hơn và quét lại ít hơn trong quá trình forward-dropping. Tuy nhiên, phụ thuộc vào dữ liệu và tiêu chí chọn, ta có thể ưu tiên sử dụng một giá trị γ thấp hơn cho dữ liệu nhiều chiều. Lý do là một giá trị γ cao có khả năng loại bỏ quá nhiều đặc trưng, dẫn đến ít đặc trưng có cơ hội được thêm vào mô hình trong quá trình forward-dropping. Điều này làm cho quá trình forward-dropping sẽ kết thúc sớm, và khi đó sẽ còn một lượng lớn đặc trưng cần quét lại trong quá trình re-forward.

Một điều quan trọng cần chú ý là sau bước forward-dropping là re-foward. Do đó, sau các bước forward-dropping, thuật toán sẽ xếp hạng các đặc trưng theo mức độ quan trọng. Có thể chỉ định số lượng tối đa các đặc trưng được đưa vào mô hình nếu người dùng muốn bộ bộ đặc trưng có số lượng nhỏ

hơn những gì thật sự thu được. Do đó, chúng tôi có thể chỉ định số lượng đặc trưng tối đa được đưa vào mô hình cho gia đoạn re-foward.

Điều cuối cùng nhưng cũng không kém phần quan trọng, nếu số lượng các đặc trưng được chọn sau giai đoạn re-forward là không đáng kể, người dùng nên sử dụng tính năng lùi an toàn thay vì lùi song song. Điều này là do khi số lượng đặc trưng trong mô hình ít, sẽ dẫn đến việc tính toán song song làm thuật toán chạy chậm hơn.

Thuật toán 5. OneForwardDropping

Input: Một khối các đặc trưng F_b và nhãn Y tương ứng, tham số loại bỏ sớm là γ.
Output: Một đặc trưng tốt f_b được chọn, một khối các đặc trưng còn lại F_b.

$f_b \leftarrow \arg\max_{f \in F_b} t_{R,f}$
if $t_{R,f_b} - t_R < \alpha$ **then**
    # if there is no more relevant features, drop block by setting it to $\emptyset$
    $F_b \leftarrow \emptyset$
else
    # Early dropping:
    $D \leftarrow \{f : t_{\{R,f\}} - t_R < \gamma\}$
    $F_b \leftarrow F_b \setminus \{D, f_b\}$
end if
return f_b, F_b

Thuật toán 6. OneReforward

Input: Một khối các đặc trưng F_b và nhãn Y tương ứng, tham số tiến là α.
Output: Một đặc trưng tốt f_b được chọn, một khối các đặc trưng còn lại F_b.

$f_b \leftarrow \arg\max_{f \in F_b} t_{R,f}$
if $t_{R,f_b} - t_R < \alpha$ **then**
    # if there is no more relevant features, drop block by setting it to $\emptyset$:
    $F_b \leftarrow \emptyset$
else
    $F_b \leftarrow F_b \setminus \{f_b\}$
end if
return f_b, F_b

Chương 4

Thực nghiệm

Để mô tả hiệu suất của PFST, chúng tôi so sánh PFST với một số kĩ thuật gồm

- Parallel Sequential Forward Selection (PSFS) [39],
- Parallel Sequential Backward Selection (PSBS) [39],
- Parallel Support Vector Machine Feature Selection based on Recursive Feature Elimination with Cross-Validation (PSVMR) [20],
- Parallel Mutual Information-based Feature Selection (PMI) [5].

Dưới đây là chúng tôi xin trình bày tóm tắt các thuật toán lựa chọn đặc trưng mà chúng tôi sử dụng để so sánh với thuật toán của chúng tôi.

4.1 Một vài thuật toán lựa chọn đặc trưng

4.1.1 Thuật toán lựa chọn tuần tự song song [39]

Thuật toán lựa chọn tuần tự song song có hai phiên bản

- Thuật toán Sequential Forward Selection (SFS): được xây dựng trên cơ sở toán học của bài toán tối ưu hóa. Mục tiêu của SFS là tìm kiếm một tập con tối ưu của các đặc trưng để cải thiện hiệu suất của một mô hình dự đoán. Cụ thể, SFS bắt đầu với một tập con rỗng của các đặc trưng và tìm kiếm lần lượt các tập con tốt hơn bằng cách thêm một đặc trưng mới vào mỗi lần. SFS sẽ kiểm tra tất cả các tập con con của tập con hiện tại, thêm đặc trưng nào cải thiện hiệu suất của mô hình dự đoán nhất, và sau đó thêm đặc trưng đó vào tập con mới. Thuật toán tiếp tục lặp lại quá trình

này cho đến khi không thể tìm thấy thêm đặc trưng nào có thể cải thiện hiệu suất của mô hình.

Để đánh giá hiệu suất của mô hình trên mỗi tập con, SFS sử dụng một hàm mục tiêu, thường là độ chính xác (accuracy) hoặc F1 score. Mục tiêu của SFS là tối đa hóa hàm mục tiêu này trên tất cả các tập con có thể có.

Ngoài ra, việc tính toán độ đo đánh giá của một tập con đặc trưng độc lập với các tập con khác nên ta có thể áp dụng tính toán song song để tính toán các giá trị đánh giá tập con đặc trưng cho các bước chuyển tiến. Tuy nhiên, việc sử dụng song song cho SFS có thể khó khăn vì mỗi bước chuyển tiến cần phải sử dụng kết quả của bước trước đó.

- Thuật toán Sequential Backward Selection (SBS): là một thuật toán tiếp cận từng bước để chọn lọc đặc trưng, trong đó bắt đầu với toàn bộ tập dữ liệu và lần lượt loại bỏ các đặc trưng ít quan trọng nhất cho đến khi đạt được số lượng đặc trưng mong muốn.

  Đầu tiên, tập dữ liệu được đưa vào mô hình học máy để đánh giá hiệu suất của từng đặc trưng. Sau đó, đặc trưng ít quan trọng nhất được loại bỏ khỏi tập dữ liệu và mô hình lại được đánh giá hiệu suất với các đặc trưng còn lại. Quá trình này được lặp lại cho đến khi số lượng đặc trưng mong muốn được đạt được.

  Để đánh giá độ quan trọng của từng đặc trưng, SBS sử dụng một giá trị đánh giá hiệu suất của mô hình, chẳng hạn như độ chính xác hoặc sai số, và đo lường hiệu suất của mô hình trên tập dữ liệu huấn luyện sau khi loại bỏ một số lượng đặc trưng.

  SBS có thể giúp giảm chiều của tập dữ liệu và cải thiện hiệu suất của mô hình bằng cách loại bỏ những đặc trưng ít quan trọng nhất. Tuy nhiên, nó có thể mất thời gian để chạy trên tập dữ liệu lớn và có số lượng đặc trưng lớn.

  Việc chạt song song thuật toán Sequential Backward Selection (SBS) cần phải chia tập dữ liệu và tính toán các mẫu con độc lập. Tuy nhiên, điều này đòi hỏi một số điều kiện nhất định, chẳng hạn như phải có thể xác định được các tính năng có thể bị loại bỏ độc lập và cùng phân phối trên các mẫu con. Vì vậy, khả năng chạy song song của SBS phụ thuộc vào đặc tính của tập dữ liệu và thuật toán. Trong nhiều trường hợp, việc thực hiện SBS song song có thể không hiệu quả và không cải thiện thời gian chạy.

4.1.2 Thuật toán Máy vectơ hỗ trợ lựa chọn đặc trưng song song dựa trên việc loại bỏ các đặc trưng đệ quy bằng xác thực chéo (Parallel Support Vector Machine Feature Selection based on Recursive Feature Elimination with Cross-Validation - PSVMR) [20]

Thuật toán Máy vectơ hỗ trợ lựa chọn đặc trưng song song dựa trên việc loại bỏ các đặc trưng đệ quy bằng xác thực chéo là một phương pháp chọn đặc trưng cho bài toán phân loại, kết hợp việc sử dụng mô hình Support Vector Machine (SVM) và phương pháp RFE-CV. PSVMR sử dụng kỹ thuật đệ quy để loại bỏ các thuộc tính dư thừa, và sử dụng kiểm tra chéo để đánh giá hiệu suất của mô hình và chọn ra các thuộc tính quan trọng nhất.

Thuật toán bắt đầu bằng việc huấn luyện một mô hình SVM với toàn bộ các đặc trưng của tập dữ liệu. Sau đó, các đặc trưng được xếp hạng dựa trên giá trị hệ số của SVM. Các đặc trưng có hệ số nhỏ nhất sẽ được loại bỏ khỏi tập dữ liệu.

Tiếp theo, một mô hình SVM mới được huấn luyện trên tập dữ liệu đã loại bỏ các đặc trưng dư thừa. Quá trình này được lặp lại đến khi đạt được số lượng đặc trưng cần chọn. Thuật toán cũng sử dụng tính toán song song để tăng tốc độ thực thi của thuật toán trên các bộ dữ liệu lớn.

4.1.3 Thuật toán lựa chọn đặc trưng song song dựa trên thông tin lẫn nhau (Parallel Mutual Information-based Feature Selection - PMI) [5]

Định nghĩa 4.1.1. Thông tin lẫn nhau (mutual infomation - MI) giữa hai biến ngẫu nhiên X và Y là

$$I(X;Y) = H(X) - H(X|Y),$$

trong đó $H(X) = -\sum_{x\in X} p(x)\log p(x)$ là hàm mất mát entropy và $H(X|Y) = -\sum_{y\in Y} p(y)\sum_{x\in X} p(x|y)\log p(x|y)$.

Thuật toán lựa chọn đặc trưng song song dựa trên thông tin lẫn nhau là một phương pháp lựa chọn đặc trưng dựa trên thông tin lẫn nhau được thực hiện song song để tăng tốc độ xử lý. Mục tiêu của PMI là giảm số lượng đặc trưng cần thiết để đạt được hiệu suất phân loại (hoặc hồi quy) tốt nhất, đồng thời giảm chi phí tính toán.

Các bước chính của thuật toán PMIFS:

- Tính toán MI giữa mỗi đặc trưng và nhãn của bài toán (hoặc giá trị hồi quy). Lưu trữ kết quả vào một danh sách.
- Sắp xếp danh sách đặc trưng theo giá trị MI giảm dần.
- Chia danh sách đặc trưng thành các phần nhỏ để thực hiện song song trên nhiều CPU hoặc GPU.
- Trong mỗi phần, xác định các cặp đặc trưng có MI cao (tức là có mối tương quan mạnh) và loại bỏ một trong hai đặc trưng trong mỗi cặp.
- Kết thúc khi tất cả các phần đã được xử lý. Kết hợp các phần lại để tạo thành tập đặc trưng được lựa chọn.

PMI giúp tăng tốc quá trình chọn đặc trưng bằng cách phân chia công việc và thực hiện song song trên nhiều thiết bị xử lý. Điều này làm giảm thời gian tính toán và cho phép xử lý bộ dữ liệu lớn hơn trong thời gian ngắn hơn.

Ngoài ra, thuật toán này có ba phương pháp để áp dụng tiêu chí MI vào lựa chọn đặc trưng. Trong luận văn này, chúng tôi đã sử dụng phương pháp Joint Mutual Infomation (JMI) được tính như sau

Định nghĩa 4.1.2. Cho ba biến ngẫu nhiên rời rạc X, Y, Z, khi đó MI điều kiện (conditional mutual information) được định nghĩa:

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = I(X;Y,Z) - I(X;Z),$$

với $I(X;Y,Z)$ là joint mutual information.

4.2 Thông tin về các tập dữ liệu và các thiết lập

Bảng 4.1: Các tập dữ liệu được sử dụng trong thực nghiệm

Tập dữ liệu	# Số lớp	# Số đặc trưng	# Kích thước mẫu
Breast cancer	2	30	569
Parkinson	2	754	756
Mutants	2	5408	31419
Gene	5	20531	801
Micromass	10	1087	360

Các thực nghiệm đã được hoàn thành trên các tập dữ liệu lấy từ thư viện Scikit-learn [39] và kho dữ liệu máy học UCI [12]. Thông tin số liệu của các tập dữ liệu này được trình bày trong bảng 4.1.

Chúng tôi đã cộng một lượng nhỏ nhiễu vào tập dữ liệu Gene và tập dữ liệu Mutants để tránh lỗi không tìm được nghịch đảo của các ma trận đơn khi chạy thuật toán PSFT. Ngoài ra, đối với tập dữ liệu Mutants, chúng tôi đã loại bỏ một dòng mà tất cả các giá trị trong dòng đều là rỗng.

Về cấu hình, chúng tôi chạy thực nghiệm trên một CPU là AMD Ryzen 7 3700X với 8 nhân và 16 luồng, 3.6GHz và 16GB ram. Sau khi chọn được các đặc trưng, chúng tôi thực hiện bài toán phân loại bằng việc sử dụng mô hình phân tích biệt thức tuyến tính (linear discriminant analysis - LDA) và trình bày kết quả của 5-fold misclassification rate trong bảng 4.2. Ngoài ra, chúng tôi cũng trình bày thời gian chạy và số lượng đặc trưng được chọn từ tất cả đặc trưng ban đầu trong bảng 4.3

Chúng tôi sẽ bỏ những trường hợp nếu không nhận được kết quả sau 5 giờ hoặc khi có vấn đề về việc tràn RAM, và ký hiệu NA trong bảng 4.2 và 4.3) để chỉ những trường hợp như vậy.

Với thuật toán PSFT, các tham số được sử dụng là $\alpha = \gamma = 0.05$, $\beta = 0.01$. Với thuật toán PSBS và PSFS, chúng tôi đã sử dụng thuật toán KNN với $K = 3$ làm hàm ước lượng. Với thuật toán PSVMR, chúng tôi sử dụng kernel tuyến tính và sử dụng kernel "JMI" cho thuật toán PMI. Để cho việc so sánh được công bằng, chúng tôi đã set số lượng đặc trưng cần được chọn từ các kĩ thuật khác bằng với số lượng đặc trưng mà PSFT chọn được.

4.3 Độ đo đánh giá

Ta sử dụng k-fold misclassification rate là một sự kết hợp của phương pháp kiểm tra chéo (cross-validation) với k phần (k-fold cross-validation) và tỷ lệ phân loại sai (misclassification rate). Cụ thể,

- k-fold cross-validation là một kỹ thuật kiểm định chéo được sử dụng trong học máy và thống kê để đánh giá hiệu suất của mô hình trên một tập dữ liệu chưa được sử dụng. Tập dữ liệu được chia thành k phần (folds) cân bằng, trong đó k-1 phần được sử dụng để huấn luyện mô hình và phần còn lại được sử dụng để kiểm tra. Quá trình này được lặp lại k lần, sao cho mỗi phần đều được sử dụng làm tập kiểm tra đúng một lần. Kết quả cuối cùng là trung bình của kết quả từ các lần kiểm tra chéo.
- Misclassification rate: Là một phép đo hiệu suất của mô hình phân loại, đo lường tỷ lệ mẫu bị phân loại sai. Nó được tính bằng cách lấy tổng số mẫu bị phân loại sai chia cho tổng số mẫu trong tập dữ liệu.

Để tính toán chỉ số này, ta cần thực hiện quá trình kiểm tra chéo k-fold và

tính tỷ lệ phân loại sai cho mỗi lần kiểm tra, sau đó lấy trung bình của các tỷ lệ đó để thu được k-fold misclassification rate.

Trong thực nghiệm, chúng tôi sử dụng 5-fold misclassification rate để đánh giá hiếu suất của mô hình LDA tương ứng với mỗi tập con các đặc trưng được chọn của mỗi phương pháp.

4.4 Kết quả và thảo luận

Bảng 4.2: 5-fold misclassification rate

Datasets	# Selected Features	PFST (our)	PSFS	PSBS	PSVMR	PMI	Full Features
Breast cancer	3	**0.042**	0.111	0.074	0.051	0.076	0.042
Parkinson	11	**0.112**	0.234	NA	NA	0.181	0.362
Mutants	6	**0.008**	NA	NA	NA	NA	0.010
Gene	12	**0.006**	0.009	NA	NA	0.007	0.042
Micromass	19	0.115	0.310	0.218	**0.096**	0.228	0.129

Bảng 4.3: Thời gian chạy và số lượng đặc trưng chọn được

Datasets	# selected features	# Features	Running Time (s)				
			PFST (our)	PSFS	PSBS	PSVMR	PMI
Breast cancer	3	30	**0.095**	1.403	8.268	12.470	0.646
Parkinson	11	754	**3.219**	163.32	NA	NA	77.269
Mutants	6	5408	**674.702**	NA	NA	NA	NA
Gene	12	20531	**172.386**	5350.14	NA	NA	2706.45
Micromass	19	1087	**16.1**	252.9	10561.3	46.909	144.684

Từ bảng 4.2 và 4.3, ta có thể thấy rằng phương pháp lựa chọn đặc trưng PFST không chỉ có hiệu suất tốt về mặt tốc độ mà con đạt độ chính xác cao cho bài toán phân loại, Ví dụ, với tập dữ liệu "Breast cancer", PFST đã chọn ra tập hợp gồm 3 đặc trưng từ 30 đặc trưng ban đầu với chỉ 0.095 giây, trong khi PSBS cần 8.268 giây và PSVMR cần 12.470 giây. Không chỉ có thời gian chạy tốt nhất, phương pháp PFST còn đạt được kết quả phân loại tốt nhất với tỉ lệ phân loại sai chỉ 0.042, đây cũng là tỉ lệ phân loại sai khi sử dụng tất cả đặc trưng. Rõ ràng, kết quả này đã cho thấy, PFST lựa chọn được các đặc trưng tốt, ảnh hưởng thật sự đến kết quả phân loại. Ngoài ra, ta cũng thấy rằng hiệu suất của PFST tốt hơn PSBS - một phiên bản lựa chọn đặc trưng lùi.

Với tập dữ liệu Parkinson, PFST đã lựa chọn ra 11 trong tổng số 754 đặc trưng. Kết quả thu được là tỉ lệ phân loại sai vẫn là thấp nhất với chỉ 0.112, khoảng 33% tỉ lệ phân loại sai khi sử dụng toàn bộ đặc trưng (0.362), và 66.87% tỉ lệ phân loại sai của phương pháp tốt thứ hai là PMI (0.181). Điều này cho thấy PFST đã loại bỏ rất hiệu quả các đặc trưng dư thừa và đẩy được hiệu suất

lên đáng kể. Về mặt thời gian chạy, PFST chỉ mất 3.219 giây để chọn 11 đặc trưng từ 754 đặc trưng, trong khi PMI cần đến 77.269 giây, PSFS cần 163.32 giây. PSBS và PSVMR không thể thu được kết quả khi thời gian chạy vượt quá 5 giờ.

Trong tập dữ liệu nhiều đặc trưng nhất, tập dữ liệu Gene, PFST đã rút gọn 20531 đặc trưng xuống còn 12 đặc trưng trong chưa đầy 3 phút và đạt được tỉ lệ phân loại sai tốt nhất với chỉ 0.006. Mặc dùng PSFS và PMI cũng đạt được kết quả phân loại tốt với tỉ lệ phân loại sai lần lượt là 0.009 và 0.007, nhưng hai phương pháp này chạy lâu hơn PFST, còn đối với PSBS và PSVMR đã không thể thu được kết quả trong vòng 5 giờ chạy.

Mặc dù PFST là thuật toán nhanh nhất trong số các thuật toán được sử dụng trong phần thực nghiệm này, nhưng phương pháp này cũng làm tốt hơn các phương pháp khác về tỉ lệ phân loại sai với bốn trong năm tập dữ liệu. Với tập dữ liệu Micromass, hiệu suất tốt nhất về kết quả phân loại thuộc về phương pháp PSVMR, và tốt thứ hai là PFST. Tuy nhiên, PSVMR chỉ tốt hơn PFST rất ít, không đáng kể (chỉ thấp hơn 1.9% tỉ lệ phân loại sai), nhưng thời gian chạy lại gần gấp 3 lần PSFT (46.909 giây so với 16.1 giây).

Kết luận

Qua khóa luận này, chúng tôi đã trình bày phương pháp PFST, một cách tiếp cận mới về việc lựa chọn song song các đặc trưng từ các tập dữ liệu lớn trong bài toán phân loại. Phương pháp của chúng tôi đã sử dụng tiêu chí vết, là tiêu chí có các thuộc tính để đánh giá các đặc trưng tốt. Chúng tôi ước lượng các tiếp cận này thông qua nhiều thực nghiệm và các tập dữ liệu khác nhau. Chúng tôi sử dụng LDA làm mô hình phân loại trên các đặc trưng được chọn. Thực nghiệm cho thấy phương pháp của chúng tôi có thể chọn ra các đặc trưng tốt với thời gian ngắn hơn khi so với các phương pháp tiếp cận khác. Hơn nữa, phân loại dựa trên các đặc trưng chọn được bằng PFST cũng thu được độ chính xác tốt hơn các phương pháp khác (tốt hơn bốn trong năm tập dữ liệu) và tốt hơn khi chạy mô hình phân loại với toàn bộ đặc trưng. Tuy nhiên, một trong những nhược điểm của PFST là tiêu chí vết chỉ có thể sử dụng cho các đặc trưng liên tục. Do đó, trong tương lại, chúng tôi vẫn sẽ nỗ lực làm việc, và hi vọng sẽ khám phá ra cách mở rộng cho các trường hợp còn lại trong bài toán phân loại.

Ngoài ra, chúng tôi cũng đã xây dựng thuật toán như một thư viện trong python. Người dùng có thể tìm thấy và cài đặt trong github sau: `https://github.com/pthnhan/PFST`

Tài liệu tham khảo

[1] C. Aggarwal and P. Yu. Outlier detection for high dimensional data, in 'acm sigmod international conference on management of data', 2001.

[2] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(5):971–989, 2015.

[3] H. Arai, C. Maung, K. Xu, and H. Schweitzer. Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[4] F. Azmandian, A. Yilmazer, J. G. Dy, J. A. Aslam, and D. R. Kaeli. Gpu-accelerated feature selection for outlier detection using the local kernel density ratio. In *2012 IEEE 12th International Conference on Data Mining*, pages 51–60. IEEE, 2012.

[5] M. Bennasar, Y. Hicks, and R. Setchi. Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42(22):8520–8532, 2015.

[6] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos. *Feature selection for high-dimensional data*. Springer, 2015.

[7] G. Borboudakis and I. Tsamardinos. Forward-backward selection with early dropping. *The Journal of Machine Learning Research*, 20(1):276–314, 2019.

[8] Y.-W. Chen and C.-J. Lin. Combining svms with various feature selection strategies. *Feature extraction: foundations and applications*, pages 315–324, 2006.

[9] J. T. de Souza, S. Matwin, and N. Japkowicz. Parallelizing feature selection. *Algorithmica*, 45(3):433–456, 2006.

[10] P. J. Denning and W. F. Tichy. Highly parallel computation. *Science*, 250(4985):1217–1222, 1990.

[11] D. L. Donoho, M. Elad, and V. N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on information theory*, 52(1):6–18, 2005.

[12] D. Dua and C. Graff. UCI machine learning repository, 2017.

[13] A. K. Farahat, A. Ghodsi, and M. S. Kamel. An efficient greedy method for unsupervised feature selection. In *2011 IEEE 11th International Conference on Data Mining*, pages 161–170. IEEE, 2011.

[14] K. Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[15] D. J. Garcia, L. O. Hall, D. B. Goldgof, and K. Kramer. A parallel feature selection algorithm from random subsets. In *Proceedings of the international workshop on parallel data mining*, volume 18, pages 64–75. Citeseer, 2006.

[16] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. 1988.

[17] A. Guillén, A. Sorjamaa, Y. Miche, A. Lendasse, and I. Rojas. Efficient parallel feature selection for steganography problems. In *International Work-Conference on Artificial Neural Networks*, pages 1224–1231. Springer, 2009.

[18] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[19] I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr. Competitive baseline methods set new standards for the nips 2003 feature selection benchmark. *Pattern recognition letters*, 28(12):1438–1444, 2007.

[20] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[21] H. Hao, C.-L. Liu, and H. Sako. Comparison of genetic algorithm and sequential search methods for classifier subset selection. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, volume 3, pages 765–765. IEEE Computer Society, 2003.

[22] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22:85–126, 2004.

[23] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[24] R. A. Johnson, D. W. Wichern, et al. *Applied multivariate statistical analysis*, volume 5. Prentice hall Upper Saddle River, NJ, 2002.

[25] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[26] D. Kong and C. Ding. Pairwise-covariance linear discriminant analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[27] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[28] V. Kumar and S. Minz. Feature selection: a literature review. *SmartCR*, 4(3):211–229, 2014.

[29] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[30] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[31] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, 17(4):491–502, 2005.

[32] F. G. López, M. G. Torres, B. M. Batista, J. A. M. Pérez, and J. M. Moreno-Vega. Solving feature subset selection problem by a parallel scatter search. *European Journal of Operational Research*, 169(2):477–489, 2006.

[33] M. Masaeli, Y. Yan, Y. Cui, G. Fung, and J. G. Dy. Convex principal feature selection. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 619–628. SIAM, 2010.

[34] N. Melab, S. Cahon, and E.-G. Talbi. Grid computing for parallel bioinspired algorithms. *Journal of parallel and Distributed Computing*, 66(8):1052–1061, 2006.

[35] L. C. Molina, L. Belanche, and À. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 306–313. IEEE, 2002.

[36] T. Nguyen. Faster feature selection with a dropping forward-backward algorithm. *arXiv preprint arXiv:1910.08007*, 2019.

[37] X. V. Nguyen, J. Chan, S. Romano, and J. Bailey. Effective global approaches for mutual information based feature selection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 512–521, 2014.

[38] J. Ni, J. Li, and J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

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

[40] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[41] A. Shishkin, A. Bezzubtseva, A. Drutsa, I. Shishkov, E. Gladkikh, G. Gusev, and P. Serdyukov. Efficient high-order interaction-aware feature selection based on conditional mutual information. In *Advances in neural information processing systems*, pages 4637–4645, 2016.

[42] K. P. Sinaga, I. Hussain, and M.-S. Yang. Entropy k-means clustering with feature reduction under unknown number of clusters. *IEEE Access*, 9:67736–67751, 2021.

[43] S. Singh, J. Kubica, S. Larsen, and D. Sorokina. Parallel large scale feature selection for logistic regression. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 1172–1183. SIAM, 2009.

[44] P. Somol, P. Pudil, and J. Kittler. Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(7):900–912, 2004.

[45] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10):2231–2242, 2004.

[46] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, and V. Christophides. A greedy feature selection algorithm for big data of high dimensionality. *Machine learning*, 108(2):149–202, 2019.

[47] A. Vehtari and J. Lampinen. Bayesian input variable selection using posterior probabilities and expected utilities. *Report B31*, 2002.

[48] J. R. Vergara and P. A. Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24:175–186, 2014.

[49] S. C. Yusta. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, 30(5):525–534, 2009.

[50] T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE transactions on information theory*, 57(7):4689–4708, 2011.

[51] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle. Massively parallel feature selection: an approach based on variance preservation. *Machine learning*, 92(1):195–220, 2013.

Phụ lục

1. Parallel feature selection based on the trace ratio criterion, Thu Nguyen; Nhan Phan; Nhuong Nguyen; Binh T. Nguyen; Pål Halvorsen; Michael A. Riegler, 2022 International Joint Conference on Neural Networks (IJCNN). https://ieeexplore.ieee.org/abstract/document/9892181

Parallel feature selection based on the trace ratio criterion

Thu Nguyen*[1], Thanh Nhan Phan*[3,4,5], Nhuong Nguyen[2], Thanh Binh Nguyen[3,4,5], Pål Halvorsen[1], and Michael Riegler[1]

[1]SimulaMet, Norway
[2]University of Connecticut, USA
[3]AISIA Research Lab, Vietnam
[4]University of Science, Vietnam
[5]Vietnam National University, Vietnam

***Abstract*—The growth of data today poses a challenge in management and inference. While feature extraction methods are capable of reducing the size of the data for inference, they do not help in minimizing the cost of data storage. On the other hand, feature selection helps to remove the redundant features and therefore is helpful not only in inference but also in reducing management costs. This work presents a novel parallel feature selection approach for classification, namely Parallel Feature Selection using Trace criterion (PFST), which scales up to very large datasets. Our method uses trace criterion, a measure of class separability used in Fisher's Discriminant Analysis, to evaluate feature usefulness. We analyzed the criterion's desirable properties theoretically. Based on the criterion, PFST rapidly finds important features out of a set of features for big datasets by first making a forward selection with early removal of seemingly redundant features parallelly. After the most important features are included in the model, we check back their contribution for possible interaction that may improve the fit. Lastly, we make a backward selection to check back possible redundant added by the forward steps. We evaluate our methods via various experiments using Linear Discriminant Analysis as the classifier on selected features. The experiments show that our method can produce a small set of features in a fraction of the amount of time by the other methods under comparison. In addition, the classifier trained on the features selected by PFST not only achieves better accuracy than the ones chosen by other approaches, but can also achieve better accuracy than the classification on all available features.**

***Keywords*—feature selection, classification, trace ratio**

I. Introduction

In this era of big data, the growth of data poses challenges for effective data management and inference. For example, a gene dataset can contain hundred thousands of features [1]. Hence, directly handling such datasets may face the curse of dimensionality. Moreover, the presence of redundant features can derail the learning performance of classification algorithms. For dealing with this issue, many dimension reduction techniques have been developed [2]–[6], and they are categorized into either feature extraction or feature selection methods [7], [8].

Feature extraction techniques (e.g., Principal Component Analysis [9], Linear Discriminant Analysis [9]) involve projecting the data into a new feature space with lower dimensionality via some linear or nonlinear transformation of the original features. However, this creates sets of new features that can not be directly interpreted. Moreover, since those approaches use all the features available during feature extraction, it does not help reduce the cost of data storage and the cost of collecting data in the future.

On the other hand, feature selection methods (forward selection [10], backward selection [10], etc.) select only a subset of useful features for model construction. Therefore, it maintains the original features' meanings while reducing the cost of storage and collecting data in the future by removing irrelevant features.

However, data from various fields, such as text mining, business analytics, and biology, are often measured in gigabytes or terabytes with millions of features [11], [12]. For example, the Amazon Review dataset [13] is a 34 gigabytes dataset. In such cases, the performance of the most current feature selection techniques may be jeopardized [12]. This is due to the search space for a set of relevant features increases significantly. One way to deal with this issue is to go for parallelization, which allows better use of computers' computational resources by data partitioning and running features selection on multiple cores at the same time.

In this work, we develop a novel parallel feature selection method for classification that can help remove redundant features and improve the model's performance without sacrificing the running time for feature selection. This is achieved by using the trace criterion in Linear Discriminant Analysis [9] as the evaluation criteria for the measure of separability between classes and therefore help evaluate if adding a feature helps to improve the model. Using this criterion, we propose a parallel feature selection approach, namely Parallel Feature Selection based on Trace criterion (PFST), which consists of three stages. First, in a forward-dropping stage, we do parallel forward selection on multiple workers and discard seemingly redundant features from subsequent steps of the stages. At the second stage, after the most important features are already included in the model, we check back their contribution after including many essential features for possible interaction that

* denotes equal contribution

may improve the fit. Finally, we conduct backward selection in the last stage to check back possible irrelevant features added by the forward moves.

The rest of the article is organized as follows: First, Section II gives a review of related works in feature selection. Second, Section III reviews some basic approaches in feature selection, and Section IV details the trace criterion and some of its desirable properties as a measure of class separability for feature selection. Next, Section V describes our methodology, and Section VI illustrates the power of our method via experiments on various datasets. Lastly, we summarize our approach and discuss future works in Section VII.

II. Related Works

Due to the benefits of maintaining original features while reducing the storage cost of feature selection, there have been many efforts in the field of feature selection to develop new techniques for the growing data size challenges. Aside from hybrid approaches that combine different feature selection strategies [14], [15], most feature selection methods can be classified into three categories.

First, the wrapper approaches rely on the performance of a specified learning algorithm to evaluate the importance of selected features. A typical wrapper method will first search for a subset of features for a given learning algorithm and then evaluate them. These steps are repeated until some stopping criteria are satisfied. Methods in this category are usually computational expensive as subset evaluation requires multiple iterations. Even though many search strategies, such as best-first search [16] and the genetic algorithm [17], have been proposed, using these strategies for high dimensional data is still computational inefficient.

On the other hand, the filter approaches consist of techniques that evaluate feature subsets via ranking with some criteria such as information criteria [18], [19], reconstruction ability [20], [21]. These approaches choose features independently from learning algorithms and are typically more computationally efficient than wrapper methods [12]. However, since the are not optimized for any target learning algorithm, they may not be optimal for a specific learning algorithm.

Meanwhile, the embedded approaches use independent criteria to find optimal subsets for a known cardinality. After that, a learning algorithm is used to select the final optimal subset among the optimal ones across different cardinalities. Therefore, they are more computationally efficient than wrapper methods since they do not evaluate feature subsets iteratively. In addition, they are also guided by the learning algorithm. Therefore, they can be seen as a trade-off between the filter strategy and wrapper strategy [12].

Despite efforts in the field of feature selection so far, data from various fields may be too abundant even for filter methods to be computationally efficient. This motivates many types of research in parallel feature selection. For example, the methods proposed in [2]–[6] use parallel processing to evaluate multiple features or feature subsets simultaneously, and therefore speed up the feature selection process. Yet, these algorithms require access to the whole data. On the other hand, in [22], the authors propose a parallel feature selection algorithm for logistic regression based on the MapReduce framework, and features are evaluated via the objective function of the logistic regression model. Meanwhile, the authors in [23] propose a *Parallel, Forward–Backward with Pruning algorithm* (PFBP) for feature selection by Early Dropping [23] some features from consideration in subsequent iterations, Early Stopping [23] of consideration of features within the same iteration, and Early Return of the winner in each iteration. However, this approach requires bootstrap computations of p-value, which is computationally expensive. Zhao et al. [24] introduce a parallel feature selection algorithm that selects features based on their abilities to explain the variance of the data. Yet, in their approach, determining the number of features in the model is based on transforming the categorical labels to numerical values and using the sum of squared errors. Getting the sum of squared errors requires fitting the model and, therefore, the algorithm is still computationally expensive.

III. Preliminaries

Various feature selection algorithms have been developed up to now. However, forward, backward, and stepwise feature selection remains widely used. This section summarizes these three techniques, and the details are given in Algorithms 1, 2, and 3.

Forward feature selection (Algorithm 1) has been used widely due to its computational efficiency, along with the ability to deal with problems where the number of features highly exceeds the number of observations efficiently. However, some features included by forwarding steps may appear redundant after including some other features. About the sufficient conditions for forward feature selection to recover the original model and its stability, we refer to [25] and [26] for further readings.

Algorithm 1. Forward Feature Selection

Input: A dataset of p features $f_1, f_2, ..., f_p$, threshold α.
Output: A set R of relevant features.

$R \leftarrow \emptyset$
$S \leftarrow \{f_1, f_2, \ldots, f_p\}$
while True **do**
  $f_j \leftarrow$ the most useful feature in S
  if the model improves more than an amount of α after including f_j **then**
    $R \leftarrow R \cup \{f_j\}$
    $S \leftarrow S \setminus \{f_j\}$
  else
    return R
  end if
end while

Backward feature selection [10], as shown in Algorithm 2, starts with the full model containing all the predictors and then iteratively removes the least useful one, one-at-a-time. It ensures that only redundant features are removed from the

model. However, backward feature selection is slow compared to forwarding selection.

Algorithm 2. Backward Feature Selection

Input: A dataset of p features $f_1, f_2, ..., f_p$, threshold β.
Output: A set R of relevant features.

$R \leftarrow \{f_1, f_2, \ldots, f_p\}$
while True, **do**
  $f_j \leftarrow$ the least useful feature in R
  if the amount of loss by excluding f_j is less than β **then**
    $R \leftarrow R \setminus \{f_j\}$
  else
    return R
  end if
end while

As another alternative, a hybrid version of both forward and backward feature selection is stepwise selection, detailed in Algorithm 3. In this approach, the features are added to the model sequentially as in the forwarding feature selection. However, after adding each new feature, the method may remove any feature that no longer seems applicable.

Algorithm 3. Stepwise Feature Selection

Input: A dataset of p features $f_1, f_2, ..., f_p$, forward threshold α, backward threshold β.
Output: A set R of relevant features.

$R \leftarrow \emptyset$
$S \leftarrow \{f_1, f_2, \ldots, f_p\}$
while True, **do**
  $f_j \leftarrow$ the most useful feature in S
  if the model improves more than an amount of α after including f_j **then**
    $R \leftarrow R \cup \{f_j\}$
    $S \leftarrow S \setminus \{f_j\}$
    while True **do**
      $f_k \leftarrow$ the least useful feature in R
      if the model performance decreases with an amount less than β after excluding f_k **then**
        $R \leftarrow R \setminus \{f_k\}$
      else break
      end if
    end while
  else
    return R
  end if
end while

IV. Trace criterion as a feature selection criterion

Trace criterion is a useful class separability measure for feature selection in classification tasks (more details in [9], [27]). There are many equivalent versions. However, suppose that we have C classes, and there are n_i observation for the i^{th} class, and let $\mathbf{x}_{ij}$ be the j^{th} sample of the i^{th} class. Then, one way to define the criterion is

$$trace(S_w^{-1} S_b), \tag{1}$$

where

$$S_b = \sum_{i=1}^{C} n_i (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})' \tag{2}$$

and

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)' \tag{3}$$

are the *between-class scatter matrix* and *within-class scatter matrix*, respectively. Here, A' is the transpose of a matrix A, $\bar{\boldsymbol{x}}_i$ is the class mean of the i^{th} class, and $\bar{\boldsymbol{x}}$ is the overall mean, i.e.,

$$\bar{\boldsymbol{x}}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} (\bar{\boldsymbol{x}}_{ij} - \bar{\boldsymbol{x}}_i)$$

$$\bar{\boldsymbol{x}} = \frac{1}{\sum_{i=1}^{C} n_i} \sum_{i=1}^{C} \sum_{j=1}^{n_i} (\bar{\boldsymbol{x}}_{ij} - \bar{\boldsymbol{x}}_i).$$

This leads to the following result:
Theorem 1:

$$trace(S_w^{-1} S_b) = \sum_{i=1}^{C} (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})' S_w^{-1} (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}) \tag{4}$$

The proof of this statement is similar to the (population) mean and covariance matrix version in [9]. From this theorem, we can see that the trace criterion 4 can be considered as the sum of squared Mahalanobis distances from the class means to the overall means. In addition, when the number of classes is two, the criterion 4 can be considered as the empirical estimation of KL-divergence of two multivariate normal distributions that have the same covariance matrix [28].

Since this criterion measures the separability of classes, we aim to maximize it. For notational simplicity, we write $\{R, f\}$ instead of $R \cup \{f\}$, and $t_{R,f}$ instead of $t_{\{R,f\}}$. We then have the following theorem:

Theorem 2: Let R be the set of selected features, and f be an arbitrary feature that does not belong to R.

Let S_{Rf} be the value of S_w when $\{R, f\}$ is the set of selected features, and S_R be the value of S_w when R is the set of selected features. Partition

$$S_{Rf} = \begin{pmatrix} S_R & v \\ v' & u \end{pmatrix}, \tag{5}$$

where $u \in \mathbb{R}^+, v \in \mathbb{R}^{|R| \times 1}$, $|R|$ is the cardinality of R.

If $u - v' S_R^{-1} v > 0$ then

$$t_{R,f} \geq t_R. \tag{6}$$

Else, if $u - v' S_R^{-1} v < 0$ then

$$t_{R,f} < t_R. \tag{7}$$

The proof of this theorem is given in Appendix A. This theorem implies that the improvement or degradation of the

model after adding a feature depends not only on the features themselves but also on the interaction between that feature and the features already in the model. In addition, from Equation 7, we can see that adding a feature may also reduce the class separability of a model. Hence, feature selection is needed to remove these features and the features that can only improve the model performance insignificantly. This property is desirable and not all feature selection criteria satisfy it. For example, for the mean square error (MSE), adding extra features, regardless of whether that feature is redundant or not, never increase MSE, as stated in the following theorem:

Theorem 3: Let MSE_X be the resulting mean squared error when we regress Y based on X. Suppose T contains arbitrary features to be added to the model and

$$U = (X \quad T). \tag{8}$$

Let MSE_U be the resulting mean squared error when we regress Y based on U. If $M = (T'T - T'X(X'X)^{-1}X'T)^{-1}$ exists then, $MSE_U \leq MSE_X$.
The proof of this theorem is given in Appendix B.

V. Parallel Feature Selection using Trace criterion (PFST Algorithm)

This section details the PFST algorithm for parallel feature selection. In addition to the desirable properties of the trace criterion as discussed in the previous section, the method is based on the following observations:

First, it is well known that forward feature selection is faster than backward feature selection because it sequentially adds the best relevant feature into the model. However, it is still computationally expensive. A potential way to speed up the process is to adapt the Early Dropping heuristic [29], in which features that are deemed unlikely to increase the performance of the model are discarded in subsequent iterations. In the original paper [29], the authors evaluate such possibilities by $p-$values. Yet, computing $p-$value using bootstrap is expensive. In our approach, we use the Early Dropping heuristic with the trace criterion as the evaluation tool instead. This does not require multiple iterations through the data for a forward step as using $p-$value for evaluation.

Second, stepwise feature selection appears to be a remedy to the forward error in the forward selection. It adds features to the model sequentially as in the forward feature selection. In addition, after adding a new feature, this approach removes from the model feature(s) that is no longer important according to the feature selection criterion being used. However, there is a computational cost associated with the backward steps that remove unnecessary features. [30] proposes an algorithm that takes a backward step only when the squared error is no more than half of the squared error decrease in the earlier forward steps. Yet, for a large-scale dataset, there is computational cost with checking a backward step, while there may not exist a redundant feature after a forward at all. In fact, [31] provides a simulation example to show that a backward step is rarely taken in stepwise feature selection. Hence, in our approach, we add features until reaching a satisfactory model and then use backward feature selection to remove those redundant features.

Algorithm 4. Parallel forward-backward algorithm with early dropping

Input: Set A of all features, partitioned into feature blocks $F_1, ..., F_b$, response Y, forward threshold α, backward threshold β, early dropping threshold γ, maximum number of re-forward steps maxRef.

Output: List R of relevant features.

Procedure:

$f_b \leftarrow \arg\max_f \{t_{\{f\}} : f \in F_b\}, b = 1, ..., B$ parallelly on B workers.
$R \leftarrow \{f_1, ..., f_B\}$.
 # Forward with forward-dropping stage:
while $\bigcup_{1 \leq b \leq B} F_b \neq \emptyset$ **do**
    $f_b, F_b \leftarrow OneForwardDropping(F_b, Y, \alpha, \gamma)$ (run parallel on B workers).
    $R \leftarrow R \cup \{f_1, ..., f_B\}$
end while
 # Re-forward stage:
Reset the selection pool to $A \setminus R$ and partition it into $F_1, ..., F_B$.
runs $\leftarrow 0$
while runs $<$ maxRef & $\bigcup_{1 \leq b \leq B} F_b \neq \emptyset$ **do**
    $f_b, F_b \leftarrow$ OneReforward(F_b, Y, α) (run parallel on B workers)
    $R \leftarrow R \cup \{f_1, ..., f_B\}$
    runs $\leftarrow$ runs $+1$
end while
 # Backward stage:
Partition R into B blocks of features.
$f_b \leftarrow \arg\max_{f \in F_b} t_{R \setminus \{f\}}, b = 1, ...B$ parallelly on B workers.
$f_r \leftarrow \arg\max_{f \in \{f_1, ..., f_B\}} t_{R \setminus \{f\}}$
if $t_R - t_{R \setminus \{f_r\}} < \beta$ **then**
    $R \leftarrow R \setminus \{f_r\}$
else
    break
end if
return R

Next, note that for single feature, $trace(S_w^{-1} S_b)$ reduces to

$$t = \frac{\sum_{i=1}^{C} n_i (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})^2}{\sum_{i=1}^{C} \sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)^2} \tag{9}$$

and for normalized features, S_b reduced to $S_b = \sum_{i=1}^{C} n_i \bar{x}_i \bar{x}_i'$. Let

$$t_R = trace(S_w^{-1} S_b) \tag{10}$$

when only the set of features R is included in the model.

For the sake of clarity, we added an overview of the descriptions for the used notations in Table I. Finally, with all these heuristics and observations, the proposed PFST algorithm is presented in Algorithm 4.

TABLE I: Table of notations

Symbol	Description
C	number of classes
A'	the transpose of matrix A
$\bar{\boldsymbol{x}}_i$	the class mean of the i^{th} class
$\bar{\boldsymbol{x}}$	the overall mean
R	the set of selected features
$\bar{\boldsymbol{x}}_{iRf}$	the mean of the i^{th} class when the selected features are $\{R, f\}$
$\bar{\boldsymbol{x}}_{Rf}$	the overall mean when the selected features are $\{R, f\}$
n_i	number of observations in the i^{th} class
x_{ij}	the j^{th} sample of the i^{th} class
S_R	The value of S_w when R is the set of selected features
t_R	$trace(S_w^{-1}S_b)$ when R is the set of selected features

We firstly partition the features into B blocks to assign to B workers. At first, the set of selected features R is empty. Therefore, we compute $\arg\max_f\{t_f : f \in F_b\}, b = 1, ..., B$ parallel on B workers to produce an initial set R.

Next, we use the OneForwardDropping algorithm (Algorithm 5) to sequentially add relevant features to the model and discard seemingly unimportant ones from subsequent iterations. To be more specific, the algorithm removes all the features in the pool that can not improve the trace criterion more than any amount of β. This helps reduce the computational cost of re-scanning through the features that temporarily do not seem to improve the model a lot compared to other features. However, a major drawback of the forward with early dropping stage is that a seemingly useless feature at a step may be helpful later when combined with other features. Therefore, after these steps are finished, we run the re-forward stage, we reset the features pool to $P = A \setminus R$ and re-partition into B blocks, and conduct forward selection without early dropping. Since most of the important features are already in the model, this stage is not as computationally expensive as running forward selection parallelly when combined with the forward stage. We illustrate in the experiments that our method runs only in a fraction of time compared to the other techniques.

Algorithm 5. OneForwardDropping

Input: Feature block F_b, response Y, forward threshold α, early dropping threshold γ.

Output: a relevant feature f_b and the remaining features in the selection pool F_B for worker B.

$f_b \leftarrow \arg\max_{f \in F_b} t_{R,f}$
if $t_{R,f_b} - t_R < \alpha$ **then**
    # if there is no more relevant features, drop block by setting it to $\emptyset$
    $F_b \leftarrow \emptyset$
else
    # Early dropping:
    $D \leftarrow \{f : t_{\{R,f\}} - t_R < \gamma\}$
    $F_b \leftarrow F_b \setminus \{D, f_b\}$
end if
return f_b, F_b

Finally, note that the inclusion of new features may lead to some features that are included before becoming redundant, possibly due to feature interaction. Therefore, the last stage of PFST is the Backward Stage, which tries to remove the redundant features that forward-dropping and re-forward stages may have made before. In this stage, we conduct backward feature selection steps to remove the redundant features that remained in the model. Again, this is to avoid the computational cost of checking for a backward move after every inclusion of a new feature. This is different from how backward steps are conducted in stepwise selection.

Algorithm 6. OneReforward

Input: Feature block F_b, response Y, forward threshold α.

Output: A relevant feature f_b and the remaining features in the selection pool F_B for worker B.

$f_b \leftarrow \arg\max_{f \in F_b} t_{R,f}$
if $t_{R,f_b} - t_R < \alpha$ **then**
    # if there is no more relevant features, drop block by setting it to $\emptyset$:
    $F_b \leftarrow \emptyset$
else
    $F_b \leftarrow F_b \setminus \{f_b\}$
end if
return f_b, F_b

Note that a higher γ will result in more feature dropping and less re-scanning during forward-dropping moves. However, depending on the data and the chosen criterion, we may prefer to use a lower γ for high dimensional data. The reason is that a higher γ may result in too many feature droppings, which implies that much fewer features have chances to get into the model during the forward-dropping moves. This causes the forward-dropping moves to terminate early, and we have to re-scan a large number of features during the re-forward stage.

It is also important to point out that after the forward-dropping steps are the re-forward steps. Therefore, after the forward-dropping steps, the algorithm ranks the importance of the features. It is possible to specify the maximum number of features to be included in the model if one wishes for a smaller set of features than what the thresholds may produce. Hence, we can specify the maximum number of features to be included in the model for the re-forward stage.

Last but not least, if the number of features selected after the re-forward stage is not significantly large, one should use the regular backward feature selection instead of parallel backward feature selection. This is because when the number of features in the model is small, parallelism makes the process run slower.

VI. Experiments

To illustrate the performance of our approach, we compare our method with Parallel Sequential Forward Selection (PSFS) [32], Parallel Sequential Backward Selection (PSBS) [32], Parallel Support Vector Machine Feature Selection based on Recursive Feature Elimination with Cross-Validation (PSVMR)

TABLE II: The description of datasets that are used in our experiments

Dataset	# Classes	# Features	# Samples
Breast cancer	2	30	569
Parkinson	2	754	756
Mutants	2	5408	31419
Gene	5	20531	801
Micromass	10	1087	360

[33], and Parallel Mutual Information-based Feature Selection (PMI) [34].

A. Datasets & Settings

The experiments are done on the datasets from the Scikit-learn library [32] and UCI Machine Learning repository [35]. The details of these datasets are given in Table II.

For the p53 mutants dataset, we eliminated a row of all null values. In addition, we added a small amount of noise to the Gene expression cancer RNA-Seq dataset and the p53 mutants dataset to avoid inversion error resulting from matrix singularity when running PSFT.

For PSFT, the thresholds used are $\alpha = \gamma = 0.05, \beta = 0.01$. For PSBS and PSFS, we used the K-nearest neighbors algorithm with $K = 3$ as the estimator. For PSVMR, we used linear kernel for PSVMR and "JMI" for PMI. For a fair comparison, we force other feature selection techniques to select the same number of features as PSFT.

We run the experiments on an AMD Ryzen 7 3700X CPU with 8 Cores and 16 processing threads, 3.6GHz and 16GB RAM. After selecting the relevant features, we classify the samples using *linear discriminant analysis (LDA)* and report the 5-fold misclassification rate in Table III. In addition, we present the running time and the number of selected features compared to the total number of features in Table IV.

We terminate an experiment if no result is produced after five hours of running or when having the out-of-memory issue, and denote this as NA in the result Tables III and IV).

B. Results and discussion

From the Tables III and IV, we can see that our PFST feature selection method has excellent performance not only in terms of speed but also the accuracy after doing classification. For example, in the breast cancer dataset, PFST was able to produce a set of 3 relevant features out of 30 features in only 0.095 seconds (s), while PSBS and PSVMR need 8.268s and 12.470s, respectively. Even though being the fastest feature selection method, it can achieve the best classification result with a misclassification rate of only 0.042, which is as good as classification using all available features. Moreover, we can observe that the performance of PFST is better than PSBS, which is the parallel version of backward feature selection.

Interestingly, for the Parkinson dataset, PFST selects only 11 out of 754 features. Still, the misclassification rate is the lowest (0.112), which is only 33% of the misclassification rate of classification using all available features (0.362), and 66.87% of the misclassification rate of the next best performer PMI (0.181). This implies that removing redundant features may boost the performance of the classifier significantly. For the running time, note that PFST takes only 3.219s to get 11 features from 754 features, while PMI takes 77.269s, PSFS takes 163.32s. PSBS and PSVMR can not get the results when running for more than 5 hours.

In the dataset with the most features, the Gene Expression Cancer RNA-Seq dataset, PFST reduced 20531 features to 12 features within 3 minutes and got the best misclassification rate of only 0.006. Although PSFS and PMI have also gotten good performance with a misclassification rate of only 0.009 and 0.007, respectively, they need much longer than PFST, while PSBS and PSVMR cannot get the result within 5 hours.

Even though PFST is the fastest algorithm among the ones used in the experiments, it outperforms other approaches in four out of five datasets in terms of classification results. For the Micromass dataset, the best performer for the classification is PSVMR, and the next best is PFST. However, PSVMR is only slightly better than PFST in terms of classification result (1.9% lower misclassification rate), but its running time is almost three times longer than that of PFST (46.909s and 16.1s, respectively).

VII. Conclusion

This paper presents a novel parallel feature selection approach for classification, called PFST, for feature selection on large scale datasets. The method uses the trace criterion, i.e., a criterion with many desirable properties, to evaluate feature usefulness. We evaluate the approach via various experiments and datasets using Linear Discriminant Analysis as the classifier on selected features. The experiments show that our PFST method can select relevant features in a fraction amount of time compared to other compared state of the art approaches. In addition, the classifier trained on the features selected by PFST achieves better accuracy than those selected by other methods in four out of five cases and better than the model that is fitted on all available features. However, the trace criterion can only be used for continuous features. Therefore, in the future, it would be desirable to explore how to extend this work to the case where there are categorical features in the model as well.

References

[1] I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr, "Competitive baseline methods set new standards for the nips 2003 feature selection benchmark," *Pattern recognition letters*, vol. 28, no. 12, pp. 1438–1444, 2007.

[2] N. Melab, S. Cahon, and E.-G. Talbi, "Grid computing for parallel bioinspired algorithms," *Journal of parallel and Distributed Computing*, vol. 66, no. 8, pp. 1052–1061, 2006.

[3] J. T. de Souza, S. Matwin, and N. Japkowicz, "Parallelizing feature selection," *Algorithmica*, vol. 45, no. 3, pp. 433–456, 2006.

[4] D. J. Garcia, L. O. Hall, D. B. Goldgof, and K. Kramer, "A parallel feature selection algorithm from random subsets," in *Proceedings of the international workshop on parallel data mining*, vol. 18. Citeseer, 2006, pp. 64–75.

TABLE III: 5-fold misclassification rate

Datasets	# Selected Features	PFST (our)	PSFS	PSBS	PSVMR	PMI	Full Features
Breast cancer	3	**0.042**	0.111	0.074	0.051	0.076	0.042
Parkinson	11	**0.112**	0.234	NA	NA	0.181	0.362
Mutants	6	**0.008**	NA	NA	NA	NA	0.010
Gene	12	**0.006**	0.009	NA	NA	0.007	0.042
Micromass	19	0.115	0.310	0.218	**0.096**	0.228	0.129

TABLE IV: Running time and number of selected features

Datasets	# selected features	# Features	Running Time (s)				
			PFST (our)	PSFS	PSBS	PSVMR	PMI
Breast cancer	3	30	**0.095**	1.403	8.268	12.470	0.646
Parkinson	11	754	**3.219**	163.32	NA	NA	77.269
Mutants	6	5408	**674.702**	NA	NA	NA	NA
Gene	12	20531	**172.386**	5350.14	NA	NA	2706.45
Micromass	19	1087	**16.1**	252.9	10561.3	46.909	144.684

[5] A. Guillén, A. Sorjamaa, Y. Miche, A. Lendasse, and I. Rojas, "Efficient parallel feature selection for steganography problems," in *International Work-Conference on Artificial Neural Networks*. Springer, 2009, pp. 1224–1231.

[6] F. G. López, M. G. Torres, B. M. Batista, J. A. M. Pérez, and J. M. Moreno-Vega, "Solving feature subset selection problem by a parallel scatter search," *European Journal of Operational Research*, vol. 169, no. 2, pp. 477–489, 2006.

[7] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[8] V. Kumar and S. Minz, "Feature selection: a literature review," *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014.

[9] R. A. Johnson, D. W. Wichern *et al.*, *Applied multivariate statistical analysis*. Prentice hall Upper Saddle River, NJ, 2002, vol. 5, no. 8.

[10] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[11] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, *Feature selection for high-dimensional data*. Springer, 2015.

[12] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[13] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[14] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[15] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 5, pp. 971–989, 2015.

[16] H. Arai, C. Maung, K. Xu, and H. Schweitzer, "Unsupervised feature selection by heuristic search with provable bounds on suboptimality," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[17] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.

[18] X. V. Nguyen, J. Chan, S. Romano, and J. Bailey, "Effective global approaches for mutual information based feature selection," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 512–521.

[19] A. Shishkin, A. Bezzubtseva, A. Drutsa, I. Shishkov, E. Gladkikh, G. Gusev, and P. Serdyukov, "Efficient high-order interaction-aware feature selection based on conditional mutual information," in *Advances in neural information processing systems*, 2016, pp. 4637–4645.

[20] A. K. Farahat, A. Ghodsi, and M. S. Kamel, "An efficient greedy method for unsupervised feature selection," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, 2011, pp. 161–170.

[21] M. Masaeli, Y. Yan, Y. Cui, G. Fung, and J. G. Dy, "Convex principal feature selection," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 619–628.

[22] S. Singh, J. Kubica, S. Larsen, and D. Sorokina, "Parallel large scale feature selection for logistic regression," in *Proceedings of the 2009 SIAM international conference on data mining*. SIAM, 2009, pp. 1172–1183.

[23] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, and V. Christophides, "A greedy feature selection algorithm for big data of high dimensionality," *Machine learning*, vol. 108, no. 2, pp. 149–202, 2019.

[24] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle, "Massively parallel feature selection: an approach based on variance preservation," *Machine learning*, vol. 92, no. 1, pp. 195–220, 2013.

[25] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[26] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on information theory*, vol. 52, no. 1, pp. 6–18, 2005.

[27] K. Fukunaga, *Introduction to statistical pattern recognition*. Elsevier, 2013.

[28] D. Kong and C. Ding, "Pairwise-covariance linear discriminant analysis," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[29] G. Borboudakis and I. Tsamardinos, "Forward-backward selection with early dropping," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 276–314, 2019.

[30] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *IEEE transactions on information theory*, vol. 57, no. 7, pp. 4689–4708, 2011.

[31] T. Nguyen, "Faster feature selection with a dropping forward-backward algorithm," *arXiv preprint arXiv:1910.08007*, 2019.

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

[33] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1, pp. 389–422, 2002.

[34] M. Bennasar, Y. Hicks, and R. Setchi, "Feature selection using joint mutual information maximisation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8520–8532, 2015.

[35] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[36] M. H. Kutner, C. J. Nachtsheim, J. Neter, W. Li *et al.*, *Applied linear statistical models*. McGraw-Hill Irwin New York, 2005, vol. 5.

APPENDIX

A. Proof of Theorem 2

Let $\bar{\boldsymbol{x}}_{iR}, \bar{\boldsymbol{x}}_R$ be the mean of the i^{th} class, and the overall mean when a set R of features are included in the model, respectively. Similarly, let $\bar{\boldsymbol{x}}_{iRf}, \bar{\boldsymbol{x}}_{Rf}$ be the mean of the i^{th} class, and the overall mean when a set $\{R, f\}$ of features are included in the model, respectively. Next, note that

$$\bar{\boldsymbol{x}}_{iRf} = \begin{pmatrix} \bar{\boldsymbol{x}}_{iR} \\ \bar{\boldsymbol{x}}_{if} \end{pmatrix}, \tag{11}$$

and

$$\bar{\boldsymbol{x}}_{Rf} = \begin{pmatrix} \bar{\boldsymbol{x}}_{R} \\ \bar{\boldsymbol{x}}_{f} \end{pmatrix}, \tag{12}$$

where $\bar{\boldsymbol{x}}_{if}, \bar{\boldsymbol{x}}_f$ is the i^{th} class mean, and the overall mean of the feature f, respectively.

If $u \neq v' S_R^{-1} v$ then $M = (u - v' S_R^{-1} v)^{-1}$ exists. Hence,

$$S_{Rf}^{-1} = \begin{pmatrix} S_R^{-1} + S_R^{-1} v M v' S_R^{-1} & -S_R^{-1} v M \\ -M v' S_R^{-1} & M \end{pmatrix} \tag{13}$$

Next, let

$$\begin{aligned} \eta_{iRf} &= (\bar{\boldsymbol{x}}_{iRf} - \bar{\boldsymbol{x}}_{Rf})' S_{Rf}^{-1} (\bar{\boldsymbol{x}}_{iRf} - \bar{\boldsymbol{x}}_{Rf}) \\ &= \eta_{iR} + (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v M v' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R) \\ &- 2(\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v M (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f) \\ &+ (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f)' M (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f), \end{aligned}$$

where

$$\eta_{iR} = (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R).$$

Note that $M \in \mathbb{R}, M \neq 0$. Hence,

$$\begin{aligned} \frac{\eta_{iRf} - \eta_{iR}}{M} &= (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v v' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R) \\ &- 2(\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} v (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f) \\ &+ (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f)' (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f). \end{aligned}$$

Applying Cauchy-Schwarz inequality $||a||^2 + ||b||^2 \geq 2\langle a, b \rangle$, with $a = (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R) S_R^{-1} v$, and $b = (\bar{\boldsymbol{x}}_{if} - \bar{\boldsymbol{x}}_f)$, we see that

$$\frac{\eta_{iRf} - \eta_{iR}}{M} \geq 0.$$

Since,

$$t_{R,f} - t_R = \sum_{i=1}^{C} (\eta_{iRf} - \eta_{iR}),$$

it follows that if $u - v' S_R^{-1} v > 0$ then

$$t_{R,f} \geq t_R,$$

and if $u - v' S_R^{-1} v < 0$ then

$$t_{R,f} < t_R.$$

B. Proof of Theorem 3

Let $H = X(X'X)^{-1}X'$ then it is known from [36] that $I - H$ is idempotent, i.e.,

$$(I - H)^2 = I - H \tag{14}$$

Hence, the sum of square when regressing Y over X is

$$\begin{aligned} SSE_X &= (Y - \hat{Y}_X)'(Y - \hat{Y}_X) \\ &= Y'(I - H)Y \end{aligned}$$

where $\hat{Y}_X$ is the predicted outcome when regressing Y over X.

Note that M is symmetric, and let $\hat{Y}_U$ is the predicted outcome when regressing Y over U. Hence,

$$\begin{aligned} &(U'U)^{-1}U'Y \\ &= HY + HTMT'HY - HTMT'Y - TMT'HY + TMT'Y \\ &= HY - (I - H)TMT' + (I - H)TMT'Y. \end{aligned}$$

This implies

$$\begin{aligned} &Y - \hat{Y}_U \\ &= Y - \hat{Y}_X + (I - H)TMT'HY - (I - H)TMT'Y \\ &= Y - \hat{Y}_X - (I - H)TMT'(I - H)Y \end{aligned}$$

Hence, the sum of square error when regressing Y over U is

$$\begin{aligned} SSE_U &= (Y - \hat{Y}_U)'(Y - \hat{Y}_U) \\ &= SSE_X - 2Y'(I - H)TMT'(I - H)(Y - \hat{Y}_X) \\ &\quad + Y'(I - H)TMT'(I - H)TMT'(I - H)Y \end{aligned}$$

In addition, note that $(I - H)TMT'$ is idempotent, and that $Y'(I - H)TMT'(I - H)Y \geq 0$. Therefore,

$$\begin{aligned} &Y'(I - H)TMT'(I - H)TMT'(I - H)Y \\ &= Y'(I - H)TMT'(I - H)Y \\ &\leq 2Y'(I - H)TMT'(I - H)Y. \end{aligned}$$

Hence

$$SSE_U \leq SSE_X$$

and the proof follows.