|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” | Mẫu 2 |

BÁO CÁO CHUYÊN ĐỀ SỐ 4.3.2

“Các phương pháp chọn lọc đặc trưng”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

Hà Nội - 2023

|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” |  |

BÁO CÁO CHUYÊN ĐỀ SỐ 4.3.2

“Các phương pháp chọn lọc đặc trưng”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

|  |  |
| --- | --- |
| **Người thực hiện chuyên đề** | **Cơ quan chủ trì** |
| *(Họ tên và chữ ký)* | *(Họ tên và chữ ký)* |

Hà Nội - 2023

MỤC LỤC

[MỤC LỤC 1](#_Toc129698230)

[DANH MỤC HÌNH ẢNH 2](#_Toc129698231)

[CÁC PHƯƠNG PHÁP CHỌN LỌC ĐẶC TRƯNG 3](#_Toc129698232)

[1.1. Thuật toán IG 3](#_Toc129698233)

[1.2. Thuật toán Chi-Square 3](#_Toc129698234)

[1.3. Thuật toán TF-IDF 5](#_Toc129698235)

[1.3.1. Khái niệm 5](#_Toc129698236)

[1.3.2. Cơ chế hình thày TF-IDF 7](#_Toc129698237)

[1.4. Thuật toán tự đề xuất 10](#_Toc129698238)

[1.4.1. Ý tưởng và mô hình tổng thể bài toán 10](#_Toc129698239)

[1.4.2. Xây dựng các độ đo thành phần 11](#_Toc129698240)

[1.4.3. Phương pháp tối ưu đa mục tiêu Pareto 12](#_Toc129698241)

[1.4.4. Xây dựng hàm chọn lọc và lựa chọn đặc trưng 13](#_Toc129698242)

[1.4.2. Code 14](#_Toc129698243)

# DANH MỤC HÌNH ẢNH

[Hình 1: Đánh giá độ quan trọng bằng cách sử dụng IF-IDF 6](#_Toc129698225)

[Hình 2: Mô hình bài toán tổng thể 11](file:///D:\DT_Ban%202022\bản%20word\Báo%20cáo%20tiến%20độ\Nội%20dung%204\BC_CĐ%204.3.2.%20Các%20phương%20pháp%20chọn%20lọc%20đặc%20trưng.docx#_Toc129698226)

# CÁC PHƯƠNG PHÁP CHỌN LỌC ĐẶC TRƯNG

## Thuật toán IG

Information Gain dựa trên sự giảm của hàm Entropy khi tập dữ liệu được phân chia trên một thuộc tính. Để xây dựng một cây quyết định, ta phải tìm tất cả thuộc tính trả về Infomation gain cao nhất.

Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Infomation Gain tại mỗi nút theo trình tự sau:

* Bước 1: Tính toán hệ số Entropy của biến mục tiêu S có N phần tử với Nc phần tử thuộc lớp c cho trước:



* Bước 2: Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node S1, S2, …, SK với số điểm trong mỗi child node lần lượt là m1, m2 ,…, mK , ta có:



* Bước 3: Chỉ số Gain Information được tính bằng:

**G(x, S) = H(S) – H(x,S)**

## Thuật toán Chi-Square

Để hạn chế số chiều của véc tơ đặc trưng, tránh làm gia tăng độ phức tạp tính toán và thời gian thực hiện quá mức chấp nhận được, một tập đặc trưng quá lớn cần được lọc để giữ lại một tập con chỉ bao gồm các đặc trưng thỏa mãn một số tiêu chí cho trước. Các đặc trưng được chọn giữ nguyên ngữ nghĩa ban đầu. Việc rút gọn tập đặc trưng tương đương với hướng quá trình học và xử lý dữ liệu vào những thông tin có ý nghĩa nhất đối với mục tiêu phân lớp, giúp phân biệt dữ liệu thuộc các lớp khác nhau một cách hiệu quả nhất.

Tùy vào việc sử dụng dữ liệu có hay không được gán nhãn, quá trình lọc thông tin đặc trưng có thể được giám sát hay không được giám sát. Các phương pháp được giám sát xác định mối tương quan giữa các đặc trưng và nhãn lớp dựa trên các độ đo khoảng cách, độ phụ thuộc thông tin, và tính nhất. Ngược lại, các phương pháp không giám sát lọc thông tin chỉ dựa trên mối tương quan giữa các đặc trưng với nhau, mà không sử dụng gì đến các thông tin về nhãn. Các nghiên cứu tổng quan và chuyên sâu hơn về lý thuyết lọc thông tin đặc trưng được đề cập đến trong các bài báo.

Thủ tục lọc thông tin đặc trưng sẽ được chúng tôi áp dụng ở giai đoạn học máy khi các dữ liệu đầu vào đều được gán nhãn. Vì vậy, trong khuôn khổ công trình này, chúng tôi chỉ quan tâm đến các phương pháp lọc thông tin có giám sát.

Như chúng tôi đã đề cập trước đó, phát hiện xâm nhập dựa trên bất thường sử dụng đặc trưng là số lần xuất hiện của n-gram cũng tương đương như bài toán phân loại văn bản. Vì vậy, các phương pháp chọn đặc trưng tốt cho bài toán phân loại văn bản cũng sẽ tốt cho bài toán phát hiện bất thường.

Trong thống kê, phương pháp Chi-square được dùng để kiểm tra độ liên quan giữa 2 sự kiện A và B. Trong việc giảm đặc trưng của bài toán này, 2 sự kiện là sự xuất hiện của các đặc trưng và của nhãn lớp. Các đặc trưng sẽ được xếp hạng theo công thức sau:



Trong đó, et = 1 khi mà phần nội dung gói tin D chứa đặc trưng t, ngược lại et = 0; ec = 1 khi D thuộc lớp c, ngược lại ec = 0, N là tần số quan sát được của D và E là tần số kỳ vọng. Ví dụ, E11 là tần số kỳ vọng của sự kiện đặc trưng t xuất hiện trong phần nội dung gói tin thuộc lớp c. Tương tự, N11 là tần số quan sát được của sự kiện đặc trưng *t* xuất hiện trong phần nội dung gói tin thuộc lớp *c*. Ta giả thiết sự xuất hiện của một đặc trưng và sự xuất hiện của một nhãn lớp là độc lập với nhau.

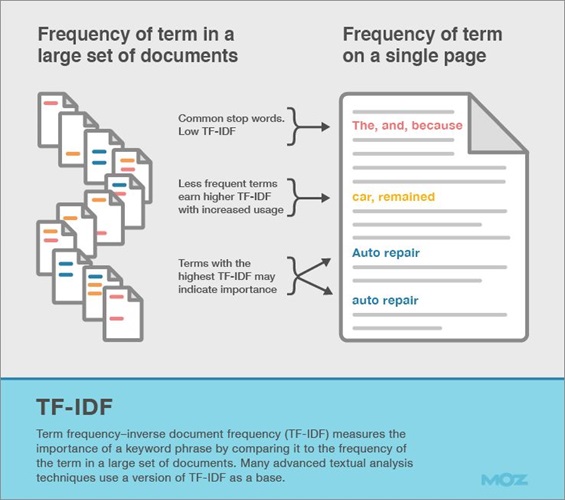
## Thuật toán TF-IDF

### Khái niệm

TF-IDF (viết tắt của term frequency – inverse document frequency) là một phương thức thống kê thường được sử dụng trong mảng truy xuất thông tin (information retrieval) và khai phá dữ liệu văn bản (text mining) để đánh giá mức độ quan trọng của một cụm từ đối với một tài liệu cụ thể trong một tập hợp bao gồm nhiều tài liệu. Khái niệm này đã xuất hiện từ rất sớm trong các lĩnh vực nghiên cứu khác nhau, chẳng hạn như ngôn ngữ học (linguistics) và cấu trúc thông tin (information architecture), nhờ vào khả năng hỗ trợ xử lý nhiều tập tài liệu với số lượng lớn trong một khoảng thời gian ngắn.

Các máy tìm kiếm thường sử dụng các biến số khác nhau của thuật toán TF-IDF như là một phần trong cơ chế xếp hạng. Bằng cách gán cho các tài liệu một mức điểm số về độ liên quan (relevance score), chúng có thể đưa ra các kết quả tìm kiếm thích hợp chỉ trong phần triệu giây.

Ví dụ, TF-IDF từ lâu đã là một phần trong cơ chế xếp hạng của Google. Google sử dụng TF-IDF để xác định xem những cụm từ nào có liên quan (hoặc không liên quan) về mặt chủ đề bằng cách phân tích tần suất một cụm từ xuất hiện trên một trang (term frequency – TF) và tần suất ước tính xuất hiện trên một trang trung bình, trong một tập hợp lớn hơn bao gồm nhiều tài liệu (inverse document frequency – IDF).



Hình 1: Đánh giá độ quan trọng bằng cách sử dụng IF-IDF

Để xác định xem một trang cụ thể có mức độ liên quan thế nào, Google sẽ phân tích các trang có trong danh sách chỉ mục của nó dựa trên một số thuộc tính cụ thể (features) mà nó cho là liên quan đến truy vấn.

Bởi vì hầu hết nội dung online đều là văn bản, nên phần lớn các thuộc tính này có thể là sự xuất hiện (presence) hoặc không xuất hiện (absence) của những từ hay cụm từ nhất định trên trang đó. Ngoài ra, hệ thống cũng xem xét đến sự nổi bật (prominence) của những từ đó trên trang so với các trang khác trên website.

Và đây là lúc thuật toán TF-IDF trở nên cần thiết. Nó sẽ tính toán tần suất sử dụng trung bình đối với một cụm từ cụ thể trên toàn website cũng như đặt ra một mức tiêu chuẩn (benchmark) cho các từ stop word (là những từ được xem là quá phổ biến, quá tổng quát và không mang ý nghĩa cụ thể nếu đứng độc lập) để mang đến một kết quả chính xác hơn.

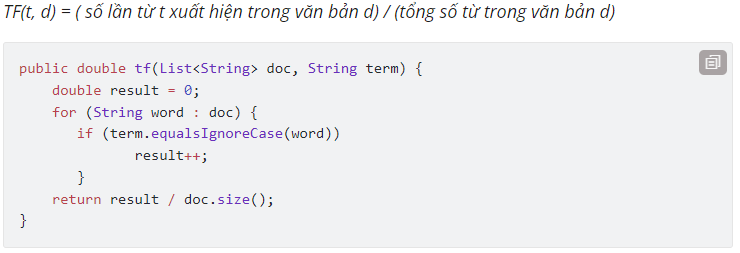
### Cơ chế hình thày TF-IDF

Đầu tiên, cần biết rằng chỉ số TF-IDF có thể được tính toán dựa theo công thức: TF-IDF = TF x IDF; trong đó đại lượng TF thể hiện số lần xuất hiện của một cụm từ trong một tài liệu nào đó, còn đại lượng IDF sẽ là một đại lượng được tính bằng hàm log. Mọi công việc tính toán sẽ được các máy tính thực hiện. Tuy nhiên, nên hiểu rằng giá trị TF-IDF không phải chỉ được hình thành dựa trên mật độ từ khóa. Sau đây là công thức tính của các chỉ số này:

**Term Frequency (TF):** Tần suất xuất hiện của một từ trong một văn bản

Hình thức đơn giản nhất của TF chính là đếm số lần sử dụng một từ khóa trên một page hoặc một tài liệu nào đó. Tuy nhiên, lúc này sẽ có một vấn đề xảy ra: giả sử có một trang sử dụng một từ khóa 10 lần thì sẽ được xem là có giá trị về độ liên quan hơn là một trang chỉ sử dụng từ khóa 1 lần. Điều này trên thực tế lại không đúng. Và vì lý do đó, chúng ta sẽ hạn chế sự sai lệch này lại, về mặt toán học thì giải pháp chính là sử dụng một hàm dưới tuyến tính (sublinear function) để phản ánh chính xác hơn, cụ thể ở đây là hàm lô-ga-rít (log).





Khi tìm kiếm trên Internet, có thể thấy có nhiều công thức tính TF khác nhau, nhưng mỗi biến thể đều được xây dựng dựa trên cơ sở: Không phải một từ xuất hiện càng nhiều trong một tệp thì mới có càng nhiều độ liên quan, và sự đóng góp của TF đến độ liên quan của một tài liệu về bản chất là một hàm dưới tuyến tính.

Chẳng hạn nếu TF của từ khóa “máy tính” trong tài liệu thứ nhất là 10 và trong tài liệu thứ 2 là 20, thì ta có thể nói tài liệu 2 có sự liên quan với từ khóa “máy tính” hơn tài liệu 1. Tuy nhiên, trong trường hợp TF của từ này không phải là 10 và 20 như trên mà là 1.000.000 và 2.000.000, thì lúc này sẽ không có nhiều sự khác biệt về sự liên quan nữa do cả hai đều chứa một số lượng lớn từ khóa “máy tính”. Và lúc này, hàm log được áp dụng để tinh gọn lại độ quan trọng của cụm từ có tần suất lớn. Chẳng hạn nếu sử dụng hàm log cơ số 2, thì giá trị 1.000.000 đó sẽ được giảm xuống chỉ còn 19.9.

Ngoài ra, trong công thức trên, ta còn cộng thêm 1 vào, vì đôi khi có trường hợp từ khóa chỉ xuất hiện 1 lần, và giá trị của log(1)=0. Nếu không cộng 1 thì TF lúc này sẽ bằng 0. Trong khi về bản chất TF sẽ là 0 khi không có từ khóa nào xuất hiện trong văn bản.

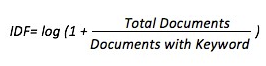
Ví dụ, đối với công thức TF ở trên, nếu bạn có một trang có 1,000 từ và từ khóa của bạn xuất hiện 10 lần, thì giá trị TF của nó sẽ là 4.32/9.97=0.43 (nếu bạn sử dụng hàm số lô-ga-rít cơ số 2).

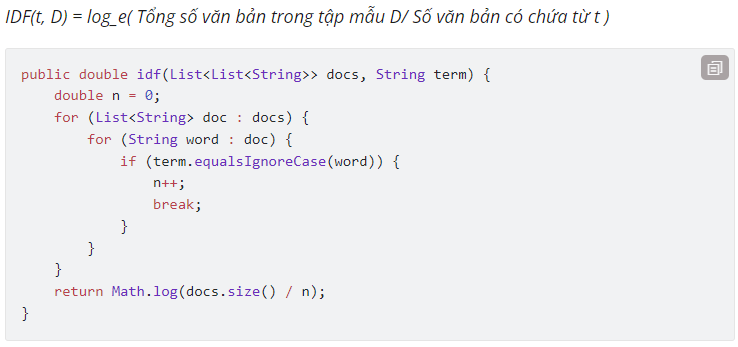
Nếu từ khóa của bạn xuất hiện nhiều hơn gấp hai lần trong cùng một tài liệu đó, thì giá trị TF của nó không thay đổi nhiều, lúc này sẽ là 5.32/9.97=0.53 (trong trường hợp vẫn sử dụng hàm log cơ số 2).

Tần suất xuất hiện của một cụm từ (TF) phản ánh xem bạn có đang sử dụng một từ khóa quá thường xuyên hoặc quá hiếm khi dùng hay không. Tuy nhiên về bản chất thì nó không mang lại nhiều giá trị vì bạn cần đo lường độ quan trọng của cụm từ, không chỉ là tần suất về số lần sử dụng nó. Nếu không thì các từ ngữ chức năng (là những từ nối, giới từ, đại từ…) sẽ thống trị hệ thống tìm kiếm. Để tránh việc đó, chúng ta cần đến chỉ số IDF.

Inverse Document Frequency (IDF): tần suất nghịch của một cụm từ trong một tập hợp gồm nhiều tài liệu

Chỉ số này thể hiện giá trị thực của một từ khóa cụ thể. Nó đo lường tỷ lệ giữa tổng số lượng tài liệu với số lượng các tài liệu có chứa từ khóa đó. Công thức để tính IDF như sau (tương tự như TF, bạn cũng có thể tìm thấy một vài cách tính biến thể khác nhau của IDF, nhưng đều được xây dựng dựa trên bản chất của nó):

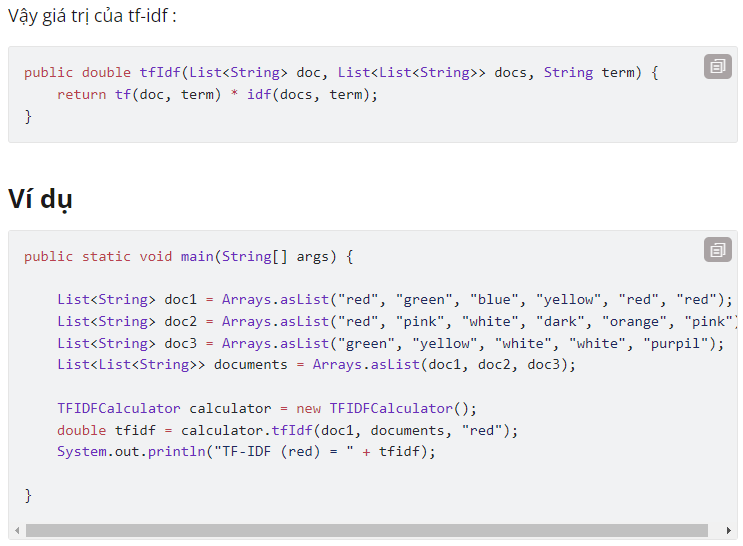




Cũng giống như công thức TF ở trên, đối với IDF cũng vậy. Một hàm IDF tuyến tính có thể đẩy điểm số của tài liệu lên quá cao khi có những cụm từ có điểm IDF cao (có thể là những cụm từ hiếm gặp, những cụm từ sai chính tả…), trong trường hợp đó thì một hàm dưới tuyến tính sẽ phản ánh chính xác giá trị hơn.

Theo công thức trên, thì nếu từ khóa là một từ phổ biến, thì khả năng rất cao là nó sẽ được sử dụng trong rất nhiều tài liệu. Do vậy, giá trị IDF của nó sẽ rất nhỏ, và nếu ta nhân nó với chỉ số TF, thì giá trị tổng thể sẽ không thay đổi nhiều. Ngược lại, nếu cụm từ đó chỉ được tìm thấy trong một vài tài liệu, thì giá trị IDF của nó sẽ cao hơn từ đó dẫn đến điểm số TF-IDF cao hơn.

Như vậy có thể thấy, không giống như đại lượng mật độ từ khóa chỉ phản ánh mức độ “nhồi nhét” một từ khóa cụ thể vào trong văn bản, TF-IDF thể hiện vai trò là một chỉ số toàn diện và nâng cao hơn, phản ánh mức độ quan trọng của một từ khóa cụ thể đối với một trang cụ thể. Nó giúp làm giảm đi sự lấn át của những từ và cụm từ không quan trọng, trong khi những cụm từ có ý nghĩa và xuất hiện không nhiều sẽ được nâng cao mức độ quan trọng hơn.

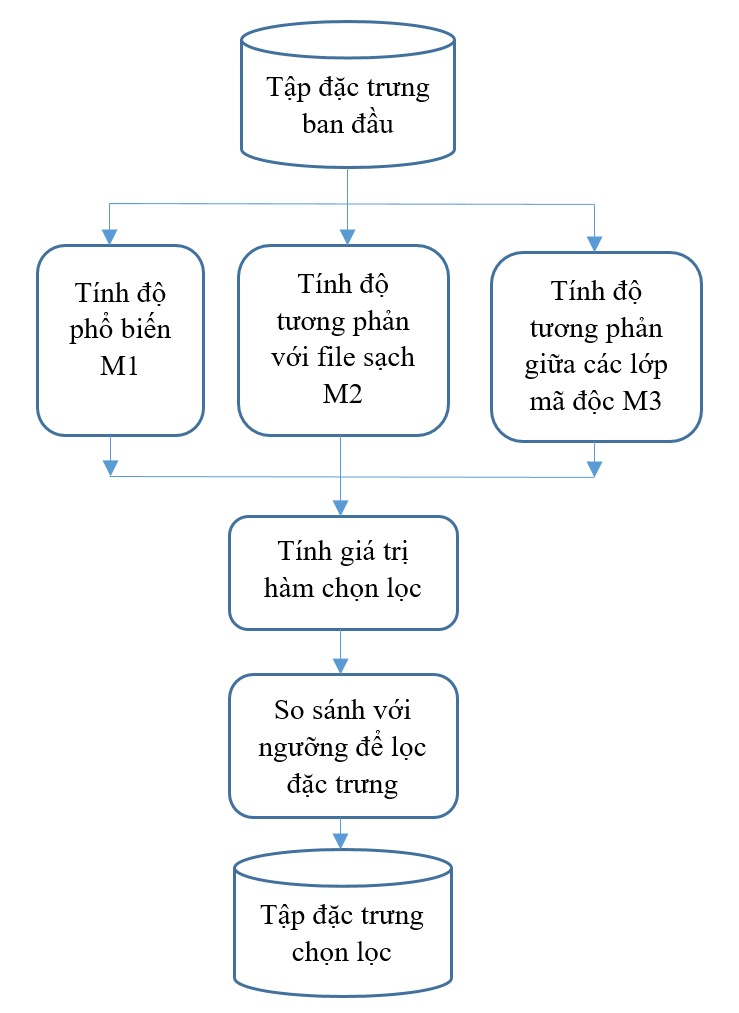


## Thuật toán tự đề xuất

### Ý tưởng và mô hình tổng thể bài toán

Ý tưởng chính của phương pháp đề xuất là sử dụng phương pháp tối ưu đa mục tiêu Pareto để xây dựng hàm chọn lọc (hàm toàn cục) dựa trên ba độ đo thành phần là độ phổ biến, độ tương phản với mã sạch và độ tương phản giữa các mã độc.

Mô hình tổng thể của phương pháp được mô tả trong Hình 2. Đầu tiên, chúng tôi xây dựng các độ đo thành phần bao gồm: độ phổ biến M1, độ tương phản với file mã sạch M2 và độ tương phản giữa các lớp M3. Hàm chọn lọc chính là hàm tối ưu toàn cục - được xây dựng dựa trên ba độ đo này theo cách tiếp cận cân bằng giữa các độ đo thành phần. Với mỗi đặc trưng trong tập đặc trưng ban đầu, chúng tôi tính giá trị của hàm chọn lọc dựa trên các độ đo thành phần. Chỉ những đặc trưng có giá trị hàm chọn lọc lớn hơn hoặc bằng ngưỡng mới được chọn.



Hình 2: Mô hình bài toán tổng thể

### Xây dựng các độ đo thành phần

Trong phần này, chúng tôi xây dựng các độ đo thành phần dựa trên giá trị của mỗi đặc trưng trong tập dữ liệu. Mỗi độ đo thành phần thể hiện một đặc tính chất lượng của đặc trưng. Các độ đo thành phần được sử dụng để xây dựng hàm chọn lọc - đánh giá đặc trưng toàn cục.

**Định nghĩa 1 – Độ phổ biến**

Độ phổ biến của mỗi đặc trưng là độ đo thể hiện mức độ phổ biến của mỗi đặc trưng. Trong trường hợp này được xây dựng dựa trên tần suất của đặc trưng. Độ phổ biến được ký hiệu M1 và được tính theo Công thức (1).

|  |  |
| --- | --- |
|  | (1) |

**Định nghĩa 2 –** **Độ tương phản với mã sạch**

Độ tương phản với mã sạch là độ đo đánh giá sự tương phản giá trị của đặc trưng trong các mẫu mã sạch và mã độc. Độ đo này càng lớn thì sự tương phản giá trị giữa mã độc và mã sạch càng cao nên càng tốt cho phân lớp. Độ tương phản với mã sạch được ký hiệu là M2 và được tính theo Công thức (2).

|  |  |
| --- | --- |
|  | (2) |

**Định nghĩa 2 –** **Độ tương phản giữa các lớp mã độc**

Độ tương phản giữa các lớp mã độc là độ đo đánh giá sự tương phản giữa các lớp mã độc với nhau. Độ tương phản giữa các lớp mã độc được ký hiệu là M3 và được tính theo công thức (3).

|  |  |
| --- | --- |
|  | (3) |

Từ (1) - (3) có,

* s là một file mẫu
* B là tập mã sạch
* V là tập mã độc
* Vj là tập mã độc có nhãn thứ j
* fi là đặc trưng thứ i

### Phương pháp tối ưu đa mục tiêu Pareto

Tối ưu Pareto là phương pháp then chốt trong tối ưu đa mục tiêu. Trong phương pháp này, gọi *X\** là nghiệm cần tìm thì *X\** phải có các tính chất sau:

* *X\** phải thuộc điểm các phương án khả thi của bài toán tức là thoả mãn các ràng buộc X\* ∈ D.
* Mọi phương án khả thi khác X ∈ D mà có một mục tiêu nào đó tốt hơn (fi(X) ≥ fi(X\*) thì cũng phải có ít nhất một mục tiêu khác xấu hơn (fj(X)<fj(X\*)) với i ≠ j.

Nghiệm *X\** này còn được gọi là nghiệm hiệu quả. Nghĩa là, nghiệm hiệu quả là các nghiệm thoả mãn: ∄X ∈ D mà có thể: với mọi i mà fi(X)>= fi(X\*) và phải tồn tại một j mà fj(X)>fj(X\*). Trên tổng thể, không có một X nào có thể trội hơn X\*.

### Xây dựng hàm chọn lọc và lựa chọn đặc trưng

**Xây dựng hàm chọn lọc**

Phương pháp tối ưu đa mục tiêu không hướng đến việc tối ưu một thành phần cụ thể mà hướng đến cân bằng, phù hợp giữa các mục tiêu tối ưu. Theo cách tiếp cận này, mỗi độ đo chính là một hàm mục tiêu thành phần; mỗi độ đo này thể hiện một khía cạnh đánh giá chất lượng đặc trưng – đây chính là một mục tiêu thành phần. Tuy nhiên, việc tối ưu đồng thời nhiều mục tiêu thành phần là không thể; thậm chí, cải tiến mục tiêu thành phần này lại làm mục tiêu thành phần khác kém đi.

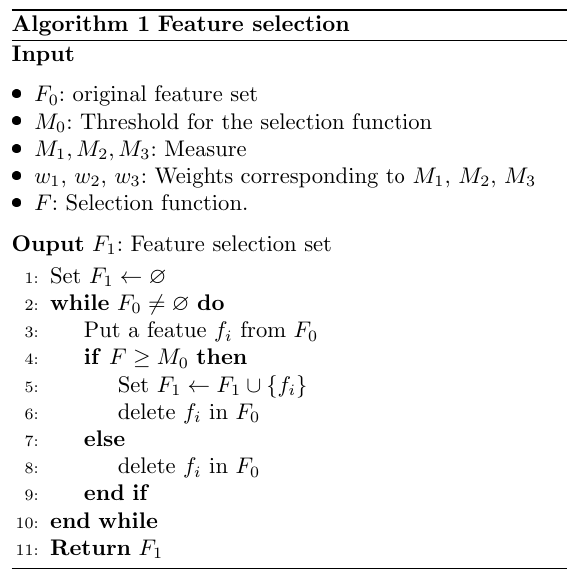
Hàm chọn lọc được xây dựng chính là hàm mục tiêu toàn cục, không hướng đến tối ưu một thành phần cụ thể mà hướng đến tối ưu toàn cục, cân bằng giữa các mục tiêu thành phần. Hàm chọn lọc được ký hiệu là F, được xây dựng dựa trên các độ đo thành phần và trọng số tương ứng như trong Công thức (4).

F = w1\*M1 + w2 \* M2 + w3 \* M3 (4)

Trong đó,

* w1, w2, w3: là các trọng số tương ứng với mỗi độ đo. Tùy theo bài toán và tùy theo mục tiêu tối ưu, độ quan trọng của mỗi độ đo mà giá trị các trọng số được thiết lập cho phù hợp.
* w1 + w2 + w3 = 1.
* F là hàm chọn lọc, F có giá trị càng lớn, càng tốt.

**Chọn lọc đặc trưng**

Hàm chọn lọc được sử dụng để lựa chọn các đặc trưng phù hợp. Mục tiêu của hàm chọn lọc hướng đến sự phù hợp, cân bằng nhất giữa các độ đo thành phần, nghĩa là nó hướng đến mục tiêu toàn cục và chất lượng tổng thể của đặc trưng. Tùy theo bài toán và số lượng đặc trưng cần chọn lọc mà ta đưa vào một giá trị ngưỡng M0 phù hợp. Đặc trưng có giá trị F ≥ M0 sẽ được chọn. Việc chọn lọc đặc trưng được thực hiện theo Thuật toán 1.

## 1.4.2. Code

import pandas as pd

import csv

import keras

from keras.layers import Input

from keras.models import Model

from keras.layers import Dense

from keras.layers import Flatten

from keras.layers.convolutional import Conv2D

from keras.layers.pooling import MaxPooling2D

from keras.layers.merge import concatenate

from tensorflow.keras.utils import to\_categorical

from keras.callbacks import TensorBoard, Callback

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import numpy as np

import time

from google.colab import drive

drive.mount('/content/drive')

np.set\_printoptions(precision=3, suppress=True)

PER\_range = 1

PER\_range\_raw = PER\_range

API\_range = 877

API\_range\_raw = API\_range

loopcount = 299

Acc\_test = 0 #lưu giá trị trung bình

train\_df = pd.read\_csv(r"/content/drive/MyDrive/Colab Notebooks/Chạy code Feature selection 05052022/Dữ liệu mới đã đánh trọng số/AMD\_Benign/F3\_AMD\_Bening\_sort/Train/train-0.csv")#, nrows=10000)

val\_df = pd.read\_csv(r"/content/drive/MyDrive/Colab Notebooks/Chạy code Feature selection 05052022/Dữ liệu mới đã đánh trọng số/AMD\_Benign/F3\_AMD\_Bening\_sort/File/file-0.csv")

test\_df = pd.read\_csv(r"/content/drive/MyDrive/Colab Notebooks/Chạy code Feature selection 05052022/Dữ liệu mới đã đánh trọng số/AMD\_Benign/F3\_AMD\_Bening\_sort/File/file-1.csv")

print(train\_df)

train\_df = train\_df.iloc[:,:877]

nhan\_train = train\_df.iloc[:,0]

val\_df = val\_df.iloc[:,:877]

nhan\_val = val\_df.iloc[:,0]

test\_df = test\_df.iloc[:,:877]

nhan\_test = test\_df.iloc[:,0]

print("train\_df ban đầu: ", train\_df)

print("nhan\_train\_df ban đầu: ", nhan\_train)

class Metrics(Callback):

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y if (y.ndim == 1 or y.shape[1] == 1) else np.argmax(y, axis=1)

self.reports = []

def on\_epoch\_end(self, epoch, logs={}):

y\_hat = np.asarray(self.model.predict(self.x))

y\_hat = np.where(y\_hat > 0.5, 1, 0) if (y\_hat.ndim == 1 or y\_hat.shape[1] == 1) else np.argmax(y\_hat, axis=1)

report = classification\_report(self.y,y\_hat,output\_dict=True)

self.reports.append(report)

return

# Utility method

def get(self, metrics, of\_class):

return [report[str(of\_class)][metrics] for report in self.reports]

'''

!WARNING: Khi chạy đoạn này, file cũ sẽ mất hết và thay vào 1 file csv raw chỉ có tên cột.

'''

def createRawCsv():

columns = [

"Lần lặp",

"Thời gian huấn luyện",

"Acc",

"Precision",

"Recall",

"F1\_score"

]

delimiter = "|"

saveName = "/content/drive/MyDrive/Colab Notebooks/Chạy code Feature selection 05052022/Mô hình học máy học sâu/F3\_Amd\_Benign\_sort\_chuan\_left\_to\_right\_per\_begin300.csv"

try:

file = open(saveName, "w")

file.write(delimiter.join(columns) + "\n")

file.close()

print("Da tao csv")

except Exception as exc:

print (exc)

createRawCsv()

import pathlib

pathlib.Path("/content/drive/MyDrive/Colab Notebooks/Chạy code Feature selection 05052022/Mô hình học máy học sâu/F3\_Amd\_Benign\_sort\_chuan\_left\_to\_right\_per\_begin300.csv").parent.resolve()

'''

Gọi hàm này khi đã có file raw CSV bên trên, nếu không thì sẽ tạo file csv không có tên cột!

Ghi luôn sau mỗi vòng lặp

'''

def writeDataToCSV(number, thoiGian, acc, precision, recall, f1):

path = "/content/drive/MyDrive/Colab Notebooks/Chạy code Feature selection 05052022/Mô hình học máy học sâu/F3\_Amd\_Benign\_sort\_chuan\_left\_to\_right\_per\_begin300.csv"

delimiter = "|"

#loop\_range = len(acc\_list) - 1

try:

file = open(path, "a")

file.write(number)

file.write(delimiter)

file.write(thoiGian)

file.write(delimiter)

file.write(acc)

file.write(delimiter)

file.write(precision)

file.write(delimiter)

file.write(recall)

file.write(delimiter)

file.write(f1)

#file.write(delimiter)

file.write("\n")

file.close()

except Exception as exc:

print(exc)

#writeDataToCSV("1", "2", "3", "4", "5", "6", "7")

# test ok

def readData(JMP\_STEP):

global loopcount, API\_range#, PER\_range

global Acc\_test

PER\_range\_use = PER\_range

#API\_range\_use = API\_range

PER\_range\_use += JMP\_STEP \* loopcount

#API\_range\_use += JMP\_STEP \* loopcount

Acc\_test = 0

loopcount += 10

try:

print("vào try")

train\_df\_2 = train\_df.drop([str(i) for i in range(PER\_range, PER\_range\_use)], axis = 1)

#train\_df\_2 = train\_df\_2.drop([str(i) for i in range(API\_range,API\_range\_use)], axis = 1)

print("traindf2: ",train\_df\_2)

val\_df\_2 = val\_df.drop([str(i) for i in range(PER\_range, PER\_range\_use)], axis = 1)

#val\_df\_2 = val\_df\_2.drop([str(i) for i in range(API\_range,API\_range\_use)], axis = 1)

test\_df\_2 = test\_df.drop([str(i) for i in range(PER\_range, PER\_range\_use)], axis = 1)

#test\_df\_2 = test\_df\_2.drop([str(i) for i in range(API\_range,API\_range\_use)], axis = 1)

#print(train\_df\_2)

train\_y = np.array(train\_df\_2.iloc[1:,0]) #lấy cột 0 là cột nhãn

train\_x = np.array(train\_df\_2.iloc[1:,1:]) #lấy cột 1 làm đặc trưng

unique, counts = np.unique(train\_y, return\_counts=True)

print("LABEL TRAIN: ", len(unique))

val\_y = np.array(val\_df\_2.iloc[1:,0]) #lấy cột 0 là cột nhãn

val\_x = np.array(val\_df\_2.iloc[1:,1:]) #lấy cột 1 làm đặc trưng

unique\_v, counts\_ = np.unique(val\_y, return\_counts=True)

#print("LABEL VAL: ", len(unique\_v))

test\_y = np.array(test\_df\_2.iloc[1:,0]) #lấy cột 0 là cột nhãn

test\_x = np.array(test\_df\_2.iloc[1:,1:]) #lấy cột 1 làm đặc trưng

unique\_t, counts\_t = np.unique(test\_y, return\_counts=True)

print("LABEL TEST: ", len(unique\_t))

SIZE = (len(train\_df\_2.columns) - 1)

print("size = ", SIZE)

BATCH\_SIZE = 32

#SIZE = API\_range\_use - 1 - JMP\_STEP\*2

SIZE2= 1

N\_CLASSES = 228

LR = 0.001

N\_EPOCHS = 10

print("số lượng đặc trưng trong train: ",train\_x.shape)

print("số lượng đặc trưng trong val: ", val\_x.shape)

print("số lượng đặc trưng trong test: ", test\_x.shape)

train\_x = train\_x.reshape(train\_x.shape[0], SIZE2, SIZE, 1)

val\_x = val\_x.reshape(val\_x.shape[0], SIZE2, SIZE, 1)

test\_x = test\_x.reshape(test\_x.shape[0], SIZE2, SIZE, 1)

LABELS = np.unique(train\_y)

original\_test\_y = test\_y

train\_y = to\_categorical(train\_y, N\_CLASSES)

val\_y = to\_categorical(val\_y, N\_CLASSES)

test\_y = to\_categorical(test\_y, N\_CLASSES)

print(LABELS)

input0 = Input(shape=(SIZE2,SIZE,1))

conv1 = Conv2D(32, kernel\_size=2, activation='relu', padding="same" ,input\_shape=(SIZE2, SIZE, 1))(input0)

pool1 = MaxPooling2D((1, 2), padding = 'same')(conv1)

conv2 = Conv2D(32, kernel\_size=2, activation='relu', padding="same")(pool1)

pool2 = MaxPooling2D((1, 2), padding = 'same')(conv2)

conv3 = Conv2D(64, kernel\_size=2, activation='relu', padding="same")(pool2)

pool3 = MaxPooling2D((1, 2), padding = 'same')(conv3)

'''

#conv4 = Conv2D(64, kernel\_size=2, activation='relu', padding="same")(pool3)

#pool4 = MaxPooling2D((2, 2), padding = 'same')(conv4)

'''

flatten\_per = Flatten()(pool3)

hidden1 = Dense(1024, activation='relu')(flatten\_per)

output = Dense(N\_CLASSES, activation='softmax')(hidden1)

model = Model(inputs=input0, outputs=output)

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

metrics\_multiclass = Metrics(train\_x, train\_y)

first\_train = time.time()

history = model.fit(train\_x, train\_y, epochs=N\_EPOCHS, batch\_size = BATCH\_SIZE ,validation\_data=(val\_x, val\_y), callbacks=[metrics\_multiclass])

second\_train = time.time()

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss=history.history['loss']

val\_loss=history.history['val\_loss']

epochs\_range = range(N\_EPOCHS)

plt.figure(figsize=(8, 8))

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

plt.savefig('/content/drive/My Drive/test.png')

plt.close('all')

tmp = list(dict.fromkeys(original\_test\_y))

y\_pred1 = model.predict(test\_x)

y\_pred = np.argmax(y\_pred1, axis=1)

# Print f1, precision, and recall scores

'''

cnf\_matrix=confusion\_matrix(original\_test\_y, y\_pred, labels=tmp, normalize='true')

plt.close('all')

def plot\_confusion\_matrix(cm, classes,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.figure(figsize=(20, 20))

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

#plt.figure()

#plot\_confusion\_matrix(cnf\_matrix, classes=tmp, title='Normalized confusion matrix')

#plt.show()

#plt.close('all')

'''

Acc\_test = accuracy\_score(original\_test\_y, y\_pred)

#print("số lượng đặc trưng trong train\_per: ",train\_x.shape)

print("loopcount: ", loopcount)

print("Thoi gian tính train riêng: ", round(second\_train - first\_train, 2), "s") #chỉ đo thời gian huấn luyện

#print("Acc\_Acc: ", Acc\_test)

print("Acc: ",accuracy\_score(original\_test\_y, y\_pred))

print("Precision: ",precision\_score(original\_test\_y, y\_pred , average="macro"))

print("recall: ",recall\_score(original\_test\_y, y\_pred , average="macro"))

print("F1\_score: ", f1\_score(original\_test\_y, y\_pred , average="macro"))

# /content/drive/My Drive/

writeDataToCSV(str(loopcount),str(round(second\_train-first\_train,2)),str(accuracy\_score(original\_test\_y,y\_pred)),str(precision\_score(original\_test\_y,y\_pred,average="macro")),str(recall\_score(original\_test\_y,y\_pred,average="macro")),str(f1\_score(original\_test\_y,y\_pred,average="macro")))

report = classification\_report(original\_test\_y, y\_pred, output\_dict=True)

df = pd.DataFrame(report).transpose()

print(df)

print(classification\_report(original\_test\_y, y\_pred, output\_dict=True))

except Exception as exc:

raise exc

i = 1

JMP\_STEP = 1

while(PER\_range\_raw < 878):

if(PER\_range\_raw < 878):

print("lan loop thu ", i)

i += 1

print("PER\_range\_raw = ", PER\_range\_raw)

#print("API\_range\_raw = ", API\_range\_raw)

first = time.time()

readData(JMP\_STEP)

second = time.time()

print("Thoi gian chay: ", round(second - first, 2), "s")

print("Acc so sánh: ", Acc\_test)

#if(Acc\_test<0.955):

# print("giá trị trung bình nhỏ hơn ngưỡng cho phép")

# break

# print("khong chạy được tới đây")

#else:

# print("giá trị vẫn lớn hơn ngưỡng nên vẫn chạy tiếp")

PER\_range\_raw += JMP\_STEP

#API\_range\_raw += JMP\_STEP

print("Done")