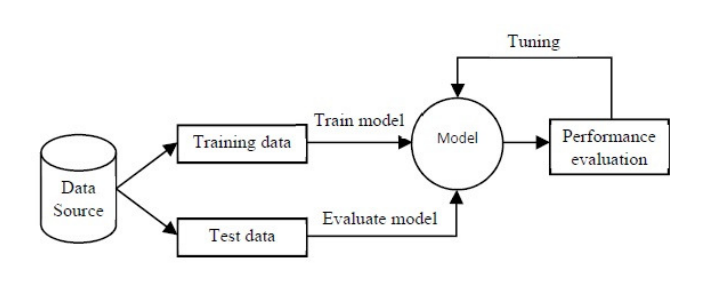
Thông tin sinh viên

Đoàn Văn Sơn – 20211512

---------------------------------------------------------------------

Học có giám sát được thực hiện bằng cách huấn luyện một mô hình có thể tổng quát hóa cho tất cả các dữ liệu liên quan khác. Các bộ phân loại có giám sát được huấn luyện bằng cách sử dụng dữ liệu có gán nhãn, và đây là một phương pháp rất quan trọng trong nhận dạng mẫu và học máy. Phương pháp cơ bản của học có giám sát được thể hiện trong Hình 5.8.



HÌNH 5.8: Phương pháp học tập có giám sát.

Phân đoạn ảnh cũng có thể được thực hiện bằng các phương pháp phân cụm có giám sát. Ví dụ, để phân đoạn ảnh nội soi, cần cô lập các vùng polyp khỏi nền. Việc này có thể được xem như một bài toán phân loại, vì cần gán nhãn riêng biệt cho các điểm ảnh thuộc vùng polyp và các điểm ảnh nền. Các đặc trưng phân biệt rõ nhất cho hai lớp này sẽ được dùng để huấn luyện mô hình.

Giả sử chúng ta có một tập dữ liệu D. Dữ liệu huấn luyện S được biểu diễn bằng một vector đặc trưng x. Nhãn y cho mỗi điểm dữ liệu huấn luyện là đã biết.

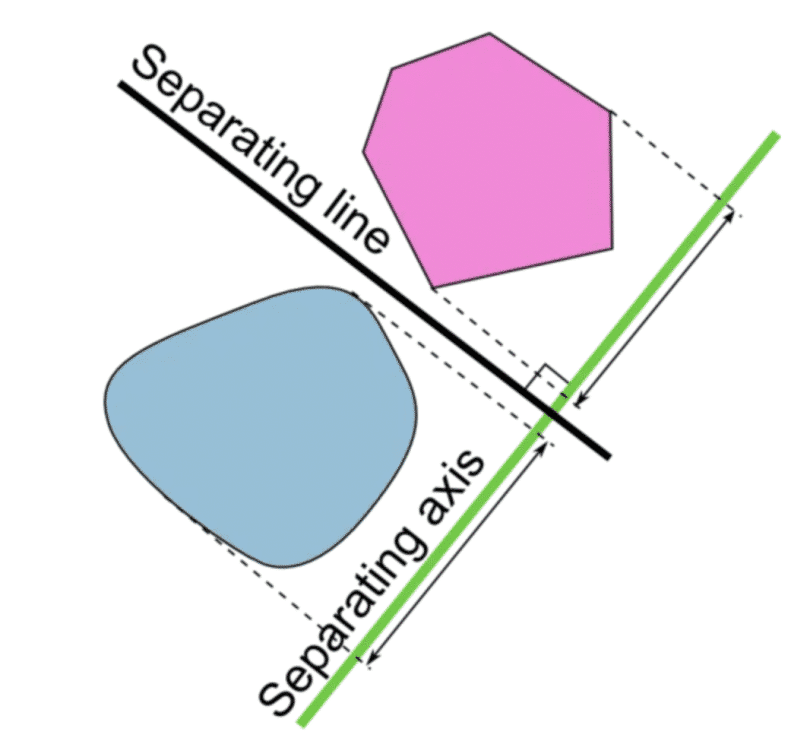
Vậy cặp (xi, yi): i = 1, 2, ..., m) sẽ tạo thành một tập học Sl, trong đó mỗi điểm đặc trưng xi € ℝn và y € (1, 2, ..., C), với C là tổng số lớp. Mục tiêu của việc huấn luyện là tìm các tham số của bộ phân loại sao cho có thể ánh xạ đặc trưng x vào lớp y có khả năng nhất. Sau khi huấn luyện, mô hình có thể được sử dụng để phân loại các dữ liệu thử nghiệm chưa biết.

Chúng ta sẽ tìm hiểu về thuật toán SVM!

**1.Support Vector Machines**

Support Vector Machines (có tài liệu dịch là Máy véctơ hỗ trợ) là một trong số những thuật toán phổ biến và được sử dụng nhiều nhất trong học máy trước khi mạng [nơ ron](https://trituenhantao.io/tu-dien-thuat-ngu/no-ron/) nhân tạo trở lại với các mô hình [deep learning](https://trituenhantao.io/tu-dien-thuat-ngu/deep-learning/). Nó được biết đến rộng rãi ngay từ khi mới được phát triển vào những năm 1990.

Mục tiêu của [SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/) là tìm ra một [siêu phẳng](https://vi.wikipedia.org/wiki/Si%C3%AAu_ph%E1%BA%B3ng) trong không gian N chiều (ứng với N [đặc trưng](https://trituenhantao.io/tu-dien-thuat-ngu/dac-trung/)) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Nói theo ngôn ngữ của đại số tuyển tính, siêu phẳng này phải có lề cực đại và phân chia hai [bao lồi](https://vi.wikipedia.org/wiki/Bao_l%E1%BB%93i) và [cách đều](https://en.wikipedia.org/wiki/Equidistant) chúng.



Trong không gian N chiều, một siêu phẳng là một không gian con có kích thước N-1 chiều. Một cách trực quan, trong một mặt phẳng (2 chiều) thì siêu phẳng là một đường thẳng, trong một không gian 3 chiều thì siêu phẳng là một mặt phẳng.

Siêu phẳng tạo ra một biên giới phân chia 2 lớp của dữ liệu.

Ảnh có chứa văn bản, biểu đồ, hàng

Mô tả được tạo tự động

[SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/) trong không gian hai chiều và ba chiều

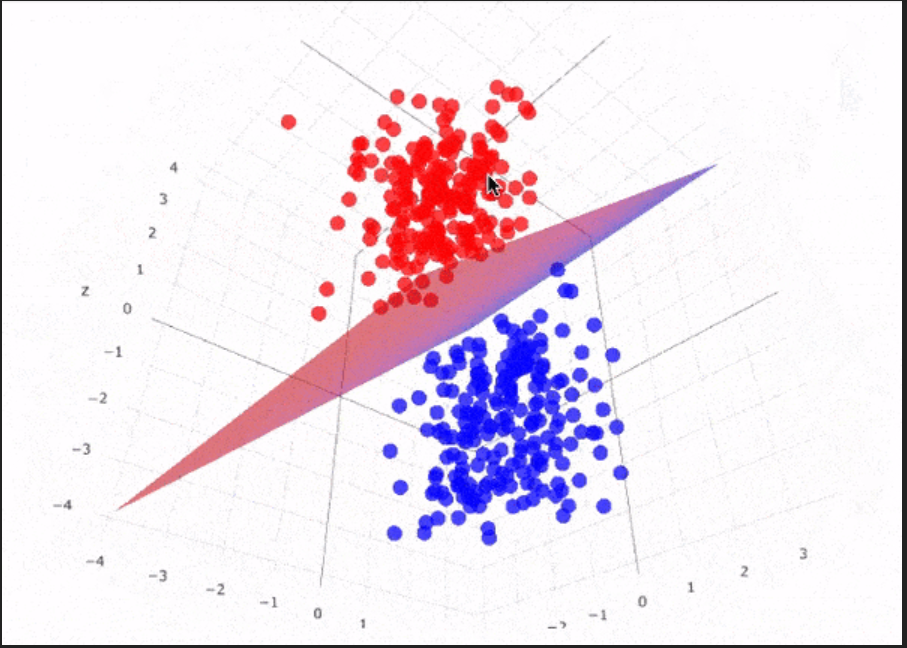
Để phân chia hai lớp dữ liệu, rõ ràng là có rất nhiều siêu phẳng có thể làm được điều này. Mặc dù vậy, mục tiêu của chúng ta là tìm ra siêu phẳng có lề rộng nhất tức là có khoảng cách tới các điểm của hai lớp là lớn nhất. Hình dưới đây là một ví dụ trực quan về điều đó.

Ảnh có chứa biểu đồ, hàng, Sơ đồ, Song song

Mô tả được tạo tự động

Siêu phẳng tối ưu có lề cực đại

Siêu phẳng có lề cực đại trong không gian 3 chiều:



**Lưu ý: Số chiều của siêu phẳng phụ thuộc vào số**[**đặc trưng**](https://trituenhantao.io/tu-dien-thuat-ngu/dac-trung/)

**Các véc tơ hỗ trợ**

Một điểm trong không gian véc tơ có thể được coi là một véc tơ từ gốc tọa độ tới điểm đó. Các điểm dữ liệu nằm trên hoặc gần nhất với siêu phẳng được gọi là véc tơ hỗ trợ, chúng ảnh hưởng đến vị trí và hướng của siêu phẳng. Các véc tơ này được sử dụng để tối ưu hóa lề và nếu xóa các điểm này, vị trí của siêu phẳng sẽ thay đổi. Một điểm lưu ý nữa đó là các véc tơ hỗ trợ phải cách đều siêu phẳng.

Ảnh có chứa biểu đồ, hàng, Sơ đồ

Mô tả được tạo tự động

[SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/) chỉ có thể hoạt động trên dữ liệu có thể phân chia tuyến tính.

Câu hỏi đặt ra **Nếu dữ liệu không thể phân chia tuyến tính thì sao?**

Ví dụ: nhìn vào hình ảnh bên dưới, dữ liệu được phân tách phi tuyến tính, rõ ràng, chúng ta không thể vẽ một đường thẳng để phân loại các điểm dữ liệu đỏ và xanh. Để giải quyết vấn đề này, có hai giải pháp:

1. Lề mềm (Soft margin)
2. Thủ thuật [Kernel](https://trituenhantao.io/tu-dien-thuat-ngu/kernel/) ([Kernel](https://trituenhantao.io/tu-dien-thuat-ngu/kernel/) tricks)

Ảnh có chứa ảnh chụp màn hình, văn bản, vòng tròn

Mô tả được tạo tự động

**Lề mềm**

Thuật toán này cho phép [SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/) mắc một số lỗi nhất định và giữ cho lề càng rộng càng tốt để các điểm khác vẫn có thể được phân loại chính xác. Nói một cách khác, nó cân bằng giữa việc phân loại sai và tối đa hóa lề.

Có hai kiểu phân loại sai có thể xảy ra:

1. Dữ liệu nằm ở đúng bên nhưng phạm vào lề
2. Dữ liệu nằm ở sai bên

***Mức độ chấp nhận lỗi***  
Mức độ chấp nhận lỗi là một [siêu tham số](https://trituenhantao.io/tu-dien-thuat-ngu/sieu-tham-so/) quan trọng trong [SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/). Khi lập trình với [sklearn](https://scikit-learn.org/), mức độ chấp nhận lỗi được coi như một tham số phạt (C). Hình dưới thể hiện [SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/) với các giá trị C khác nhau.

Ảnh có chứa ảnh chụp màn hình, Nhiều màu sắc

Mô tả được tạo tự động

C càng lớn có nghĩa là [SVM](https://trituenhantao.io/tu-dien-thuat-ngu/svm/) càng bị phạt nặng khi thực hiện phân loại sai. Do đó, lề càng hẹp và càng ít vectơ hỗ trợ được sử dụng.

**Thủ thuật**[**Kernel**](https://trituenhantao.io/tu-dien-thuat-ngu/kernel/)

Một [kernel](https://trituenhantao.io/tu-dien-thuat-ngu/kernel/) là một hàm ánh xạ dữ liệu từ không gian ít nhiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu. Một cách trực quan, kỹ thuật này giống như việc bạn gập tờ giấy lại để có thể dùng kéo cắt một lỗ tròn trên nó.

Biểu diễn trực quan của thủ thuật [kernel](https://trituenhantao.io/tu-dien-thuat-ngu/kernel/) :

Ảnh có chứa hàng, biểu đồ, Sơ đồ, số

Mô tả được tạo tự độngVí dụ Φ(x) = x^2+x

Ảnh có chứa biểu đồ, văn bản, nghệ thuật gấp giấy origami, thiết kế

Mô tả được tạo tự động

Các kiểu [Kernel](https://trituenhantao.io/tu-dien-thuat-ngu/kernel/):

1. Tuyến tính
2. Đa thức
3. RBF
4. Sigmoid

**2.Cây quyết định (Decision Tree)**

Ảnh có chứa văn bản, biểu đồ, thiết kế

Mô tả được tạo tự động

Lấy ví dụ 1 bài toán: bạn đến siêu thị mua sữa cho cả gia đình. Câu đầu tiên trong đầu bạn sẽ là: Bạn cần mua bao nhiêu sữa?

Bạn sẽ xác định: Nếu là ngày thường thì gia đình bạn sẽ sử dụng hết 1 lít sữa, còn cuối tuần thì sẽ là 1,5 lít. Như vậy, dựa theo ngày, bạn sẽ quyết định lượng thực phẩm cần mua cho gia đình bạn.

Đó chính là một dạng của cây quyết định nhị phân.

*Khái niệm Cây quyết định* (Decision Tree)

Cây quyết định ([Decision Tree](https://trituenhantao.io/tu-dien-thuat-ngu/decision-tree/)) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

Ta hãy xét một ví dụ 1 kinh điển khác về cây quyết định. Giả sử dựa theo thời tiết mà các bạn nam sẽ quyết định đi đá bóng hay không?

Những đặc điểm ban đầu là:

* Thời tiết
* Độ ẩm
* Gió

Dựa vào những thông tin trên, bạn có thể xây dựng được mô hình như sau:

Ảnh có chứa văn bản, ảnh chụp màn hình, vòng tròn, Phông chữ

Mô tả được tạo tự động

Dựa theo mô hình trên, ta thấy:

Nếu trời nắng, độ ẩm bình thường thì khả năng các bạn nam đi chơi bóng sẽ cao. Còn nếu trời nắng, độ ẩm cao thì khả năng các bạn nam sẽ không đi chơi bóng.

Thuật toán Cây quyết định (Decision Tree)

Thuật toán ID3

Giờ chúng ta hãy cùng tìm hiểu cách thức hoạt động của thuật toán cây quyết định thông qua thuật toán đơn giản ID3.

ID3 (J. R. Quinlan 1993) sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) và Information Gain để xây dựng một cây quyết định.

Ta xét ví dụ 2:

Bạn muốn xem xét sự thành công của một bộ phim thông qua hai yếu tố: diễn viên chính của phim và thể loại phim:

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Mô tả được tạo tự động

| **Lead Actor** | **Genre** | **Hit(Y/N)** |
| --- | --- | --- |
| Amitabh Bacchan | Action | Yes |
| Amitabh Bacchan | Fiction | Yes |
| Amitabh Bacchan | Romance | No |
| Amitabh Bacchan | Action | Yes |
| *Abhishek Bacchan* | *Action* | *No* |
| *Abhishek Bacchan* | *Fiction* | *No* |
| *Abhishek Bacchan* | *Romance* | *Yes* |

Giả sử, bạn muốn xác định độ thành công của bộ phim chỉ trên 1 yếu tố, bạn sẽ có hai cách thực hiện sau: qua diễn viên chính của phim và qua thể loại phim.

Qua sơ đồ, ta có thể thấy rõ ràng ràng, với phương pháp thứ nhất, ta phân loại được rõ ràng, trong khi phương pháp thứ hai, ta có một kết quả lộn xộn hơn. Và tương tự, cây quyết định sẽ thực hiện như trên khi thực hiện việc chọn các biến.

Có rất nhiều hệ số khác nhau mà phương pháp cây quyết định sử dụng để phân chia. Dưới đây, tôi sẽ đưa ra hai hệ số phổ biến là ***Information Gain*** và **Gain Ratio** (ngoài ra còn hệ số Gini).

***Entropy trong Cây quyết định (Decision Tree)***

[Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) là thuật ngữ thuộc Nhiệt động lực học, là thước đo của sự biến đổi, hỗn loạn hoặc ngẫu nhiên. Năm 1948, Shannon đã mở rộng khái niệm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) sang lĩnh vực nghiên cứu, thống kê với công thức như sau:

Với một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau x1,x2,…,xn.

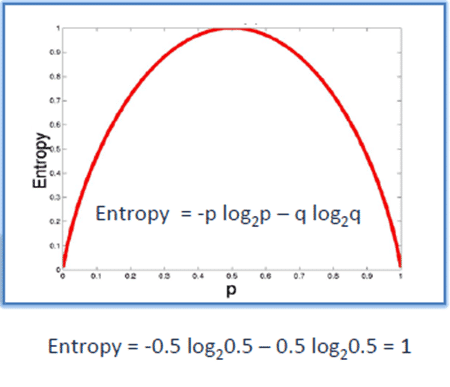
Giả sử rằng xác suất để x nhận các giá trị này là pi=p(x=xi).

Ký hiệu phân phối này là p=(p1 ,p2 ,…,pn). [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) của phân phối này được định nghĩa là:

H(p)=  – ∑nn=1 pi log(pi)

Giả sử bạn tung một đồng xu, [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) sẽ được tính như sau:

H = -[0.5 ln(0.5) + 0.5 ln(0.5)]

Hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)

Hình vẽ trên biểu diễn sự thay đổi của hàm [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/). Ta có thể thấy rằng, [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) đạt tối đa khi xác suất xảy ra của hai lớp bằng nhau.

* P tinh khiết: pi = 0 hoặc pi = 1
* P vẩn đục: pi = 0.5, khi đó hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) đạt đỉnh cao nhất

***Information Gain* *trong Cây quyết định (Decision Tree)***

Information Gain dựa trên sự giảm của hàm [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/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](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) của biến mục tiêu S có N phần tử với Nc phần tử thuộc lớp c cho trước:

H(S)=  – ∑cc=1 (Nc/N) log(Nc/N)

•**Bước 2**: Tính hàm số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/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:

H(x, S) = ∑Kk=1 (mk / N) \* H(Sk )

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

**G(x, S) = H(S) – H(x,S)**

Với ví dụ 2 trên, ta tính được hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) như sau:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)*Parent= -(0.57\*ln(0.57) + 0.43\*ln(0.43))* = 0.68

Hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) theo phương pháp chia thứ nhất:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)*left= -(.75\*ln(0.75) + 0.25\*ln(0.25))  = 0.56*[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)*right= -(.33\*ln(0.33) + 0.67\*ln(0.67)) = 0.63*

Ta có thể tính hệ số ***Information Gain*** như sau:

*Information Gain* = *0.68 – (4\*0.56 + 3\*0.63)/7 = 0.09*

Hệ số [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) với phương pháp chia thứ hai như sau:

[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)*left= -(.67\*ln(0.67) + 0.33\*ln(0.33))  = 0.63*[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)*middle= -(.5\*ln(0.5) + 0.5\*ln(0.5))  = 0.69*[*Entropy*](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/)*right= -(.5\*ln(0.5) + 0.5\*ln(0.5))  = 0.69*

Hệ số ***Information Gain***:

*Information Gain = 0.68 – (3\*0.63 + 2\*0.69 + 2\*0.69)/7 = 0.02*

So sánh kết quả, ta thấy nếu chia theo phương pháp 1 thì ta được giá trị hệ số Information Gain lớn hơn gấp 4 lần so với phương pháp 2. Như vậy, giá trị thông tin ta thu được theo phương pháp 1 cũng nhiều hơn phương pháp 2.

Thuật toán C4.5

Thuật toán C4.5 là thuật toán cải tiến của ID3.

Trong thuật toán ID3, Information Gain được sử dụng làm độ đo. Tuy nhiên, phương pháp này lại ưu tiên những thuộc tính có số lượng lớn các giá trị mà ít xét tới những giá trị nhỏ hơn. Do vậy, để khắc phục nhược điểm trên, ta sử dụng độ đo Gain Ratio (trong thuật toán C4.5) như sau:

Đầu tiên, ta chuẩn hoá information gain với trị thông tin phân tách (split information):



Trong đó: Split Info được tính như sau:

Ảnh có chứa văn bản, Phông chữ, màu trắng, Đồ họa

Mô tả được tạo tự động

Giả sử chúng ta phân chia biến thành n nút cón và Di đại diện cho số lượng bản ghi thuộc nút đó. Do đó, hệ số Gain Ratio sẽ xem xét được xu hướng phân phối khi chia cây.

Áp dụng cho ví dụ trên và với cách chia thứ nhất, ta có

*Split Info = – ((4/7)\*log2(4/7)) – ((3/7)\*log2(3/7)) = 0.98*

*Gain Ratio = 0.09/0.98 = 0.092*

**Tiêu chuẩn dừng**

Trong các thuật toán [Decision tree](https://trituenhantao.io/tu-dien-thuat-ngu/decision-tree/), với phương pháp chia trên, ta sẽ chia mãi các node nếu nó chưa tinh khiết. Như vậy, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, cây có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng [overfitting](https://trituenhantao.io/tu-dien-thuat-ngu/overfitting/) sẽ xảy ra.

Để tránh trường họp này, ta có thể dừng cây theo một số phương pháp sau đây:

* nếu node đó có [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) bằng 0, tức mọi điểm trong node đều thuộc một class.
* nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh [overfitting](https://trituenhantao.io/tu-dien-thuat-ngu/overfitting/). Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
* nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế *chiều sâu của tree* này làm giảm độ phức tạp của tree và phần nào giúp tránh [overfitting](https://trituenhantao.io/tu-dien-thuat-ngu/overfitting/).
* nếu tổng số leaf node vượt quá một ngưỡng nào đó.
* nếu việc phân chia node đó không làm giảm [entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

Ngoài ra, ta còn có phương pháp [cắt tỉa cây](https://en.wikipedia.org/wiki/Decision_tree_pruning).

Một số thuật toán khác

Ngoài ID3, C4.5, ta còn một số thuật toán khác như:

* Thuật toán CHAID: tạo cây quyết định bằng cách sử dụng thống kê chi-square để xác định các phân tách tối ưu. Các biến mục tiêu đầu vào có thể là số (liên tục) hoặc phân loại.
* Thuật toán C&R: sử dụng phân vùng đệ quy để chia cây. Tham biến mục tiêu có thể dạng số hoặc phân loại.
* MARS
* Conditional Inference Trees

**Ưu/nhược điểm của thuật toán** cây quyết định

**Ưu điểm**

Cây quyết định là một thuật toán đơn giản và phổ biến. Thuật toán này được sử dụng rộng rãi bới những lợi ích của nó:

* Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.
* Dữ liệu đầu vào có thể là là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả
* Có thể làm việc với cả dữ liệu số và dữ liệu phân loại
* Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê
* Có khả năng là việc với dữ liệu lớn

**Nhược điểm**

Kèm với đó, cây quyết định cũng có những nhược điểm cụ thể:

* Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thạm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.
* Cây quyết định hay gặp vấn đề [overfitting](https://trituenhantao.io/tu-dien-thuat-ngu/overfitting/)