Contents

[**1. SVM** 2](#_Toc181260464)

[**2. Cây Quyết Định (Decision Tree)** 5](#_Toc181260465)

[Thuật toán ID3 6](#_Toc181260466)

[Thuật toán Cart 8](#_Toc181260467)

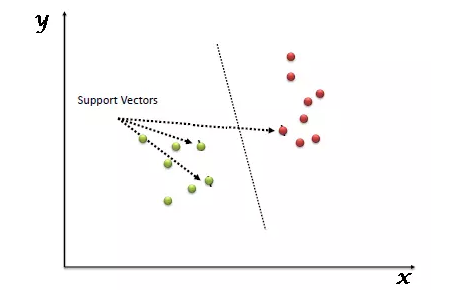
**Họ tên : Lường Văn Chung**

**Msv: 20211484**

**Lớp: DC.CNTT.12.10.6**

# **1. SVM**

**SVM** là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồi thị dữ liệu là các điểm trong n chiều ( ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" (*hyper-plane*) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



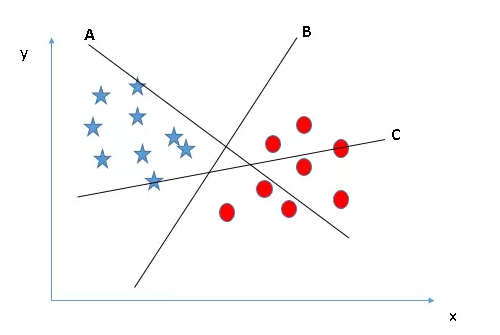
*Support Vectors* hiểu một cách đơn giản là các đối tượng trên đồ thị tọa độ quan sát, *Support Vector Machine* là một biên giới để chia hai lớp tốt nhất.

**2. SVM làm việc như thế nào**

Ở trên, chúng ta đã thấy được việc chia hyper-plane. Bấy giờ làm thế nào chúng ta có thể xác định "Làm sao để vẽ-xác định đúng hyper-plane". Chúng ta sẽ theo các tiêu chí sau:

* **Identify the right hyper-plane (Scenario-1):**

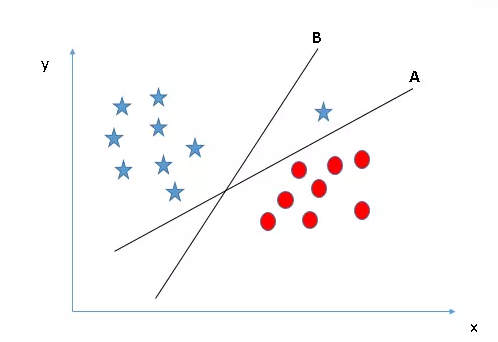
Ở đây, có 3 đường hyper-lane (A,B and C). Bây giờ đường nào là hyper-lane đúng cho nhóm ngôi sao và hình tròn



Quy tắc thứ hai chính là xác định khoảng cách lớn nhất từ điểu gần nhất của một lớp nào đó đến đường hyper-plane. Khoảng cách này được gọi là "Margin", Hãy nhìn hình bên dưới, trong đấy có thể nhìn thấy khoảng cách margin lớn nhất đấy là đường C. Cần nhớ nếu chọn lầm hyper-lane có margin thấp hơn thì sau này khi dữ liệu tăng lên thì sẽ sinh ra nguy cơ cao về việc xác định nhầm lớp cho dữ liệu.

* **Identify the right hyper-plane (Scenario-3):**

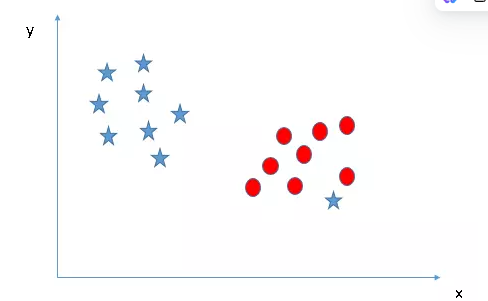
Sử dụng các nguyên tắc đã nêu trên để chọn ra hyper-plane cho trường hợp sau:



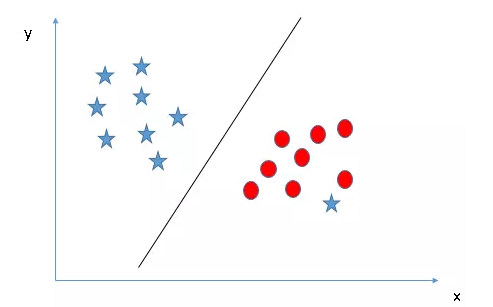
Có thể có một vài người sẽ chọn đường B bởi vì nó có margin cao hơn đường A, nhưng đấy sẽ không đúng bởi vì nguyên tắc đầu tiên sẽ là nguyên tắc số 1, chúng ta cần chọn hyper-plane để phân chia các lớp thành riêng biệt. Vì vậy đường A mới là lựa chọn chính xác.

* **Can we classify two classes (Scenario-4)?**

Tiếp the hãy xem hình bên dưới, không thể chia thành hai lớp riêng biệt với 1 đường thẳng, để tạo 1 phần chỉ có các ngôi sao và một vùng chỉ chứa các điểm tròn.

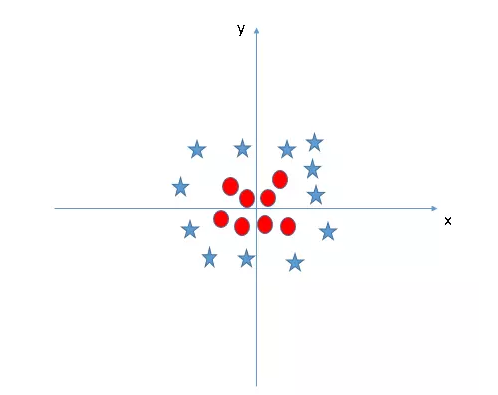


Ở đây sẽ chấp nhận, một ngôi sao ở bên ngoài cuối được xem như một ngôi sao phía ngoài hơn, SVM có tính năng cho phép bỏ qua các ngoại lệ và tìm ra hyper-plane có biên giới tối đa . Do đó có thể nói, SVM có khả năng mạnh trong việc chấp nhận ngoại lệ.

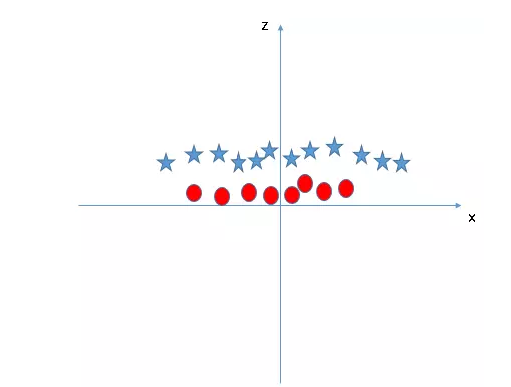


* **Find the hyper-plane to segregate to classes (Scenario-5)**

Trong trường hợp dưới đây, không thể tìm ra 1 đường hyper-plane tương đối để chia các lớp, vậy làm thế nào để SVM phân tách dữ liệu thành hai lớp riêng biệt? Cho đến bây giờ chúng ta chỉ nhìn vào các đường tuyến tính hyper-plane.

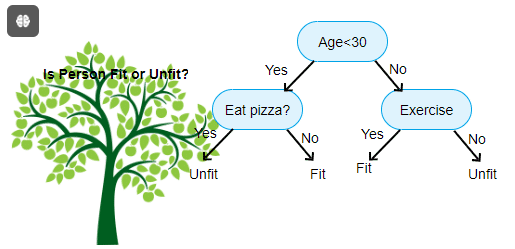


SVM có thể giải quyết vấn đề này, Khá đơn giản, nó sẽ được giải quyết bằng việc thêm một tính năng, Ở đây chúng ta sẽ thêm tính năng z = x^2+ y^2. Bây giờ dữ liệu sẽ được biến đổi theo trục x và z như sau



Trong sơ đồ trên, các điểm cần xem xét là: • Tất cả dữ liệu trên trục z sẽ là số dương vì nó là tổng bình phương x và y • Trên biểu đồ các điểm tròn đỏ xuất hiện gần trục x và y hơn vì thế z sẽ nhỏ hơn => nằm gần trục x hơn trong đồ thị (z,x) Trong SVM, rất dễ dàng để có một siêu phẳng tuyến tính (linear hyper-plane) để chia thành hai lớp, Nhưng một câu hỏi sẽ nảy sinh đấy là, chúng ta có cần phải thêm một tính năng phân chia này bằng tay hay không. Không, bởi vì SVM có một kỹ thuật được gọi là kernel trick ( kỹ thuật hạt nhân), đây là tính năng có không gian đầu vào có chiều sâu thấm và biến đổi nó thành không gian có chiều cao hơn, tức là nó không phân chia các vấn đề thành các vấn đề riêng biệt, các tính năng này được gọi là kernel. Nói một cách đơn giản nó thực hiện một số biết đổi dữ liệu phức tạp, sau đó tìm ra quá trình tách dữ liệu dựa trên các nhãn hoặc đầu ra mà chúng ra đã xác định trước.

# **2. Cây Quyết Định (Decision Tree)**



*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.

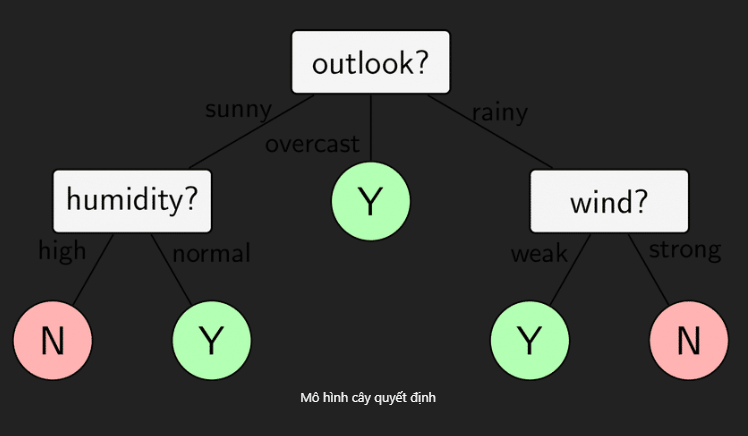
Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

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:



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.

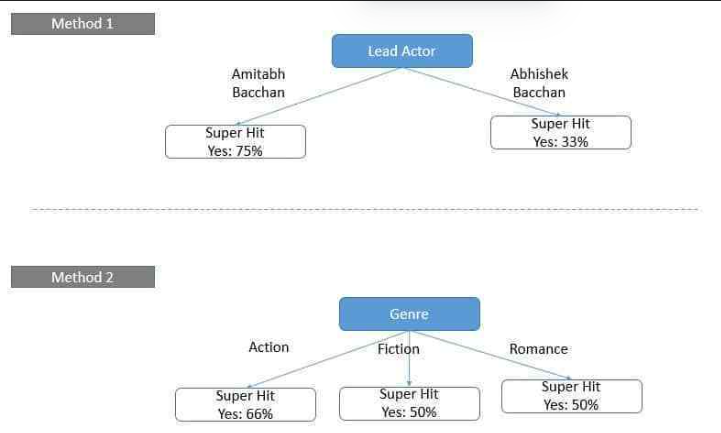
ID3 (J. R. Quinlan 1993) sử dụng phương pháp tham lam tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng [Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/entropy/) và Information Gain để xây dựng một cây quyết định.

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:



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.

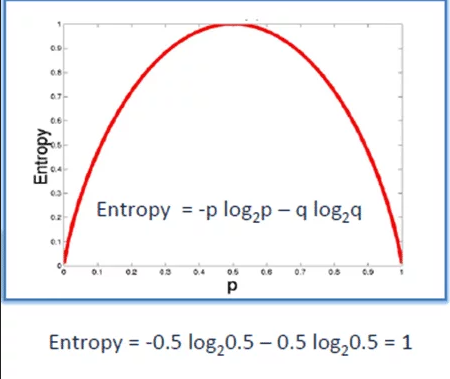
Giả sử rằng xác suất để x nhận các giá trị này là pi=p(x=xi).

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:



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





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.

## Thuật toán Cart

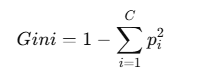
CART xây dựng cây quyết định dựa trên việc chia tập dữ liệu thành các tập con, dựa vào các thuộc tính khác nhau và giá trị ngưỡng để tối ưu hóa tiêu chí phân chia. Đối với cây phân loại, tiêu chí phổ biến nhất được sử dụng là **Gini Index** hoặc **Entropy** (mặc dù CART chủ yếu sử dụng Gini Index).

CART hoạt động theo các bước chính sau:

1. **Chọn thuộc tính và giá trị phân chia**: CART duyệt qua từng thuộc tính và giá trị ngưỡng để tìm cách phân chia dữ liệu thành hai nhóm (nhánh trái và phải) sao cho tiêu chí phân chia (Gini hoặc Entropy) được tối ưu.
2. **Tạo các nhánh**: Sau khi chọn thuộc tính và giá trị ngưỡng tối ưu, dữ liệu được chia thành hai nhánh. Với mỗi nhánh con, quy trình này được lặp lại trên dữ liệu con cho đến khi đạt các điều kiện dừng.
3. **Điều kiện dừng**:
   * **Độ sâu tối đa của cây**: Để tránh cây phát triển quá sâu dẫn đến quá khớp (overfitting), người ta thường giới hạn độ sâu của cây.
   * **Kích thước nhóm nhỏ nhất**: Nếu số lượng phần tử trong một nhóm nhỏ hơn giá trị quy định, cây sẽ không phân chia tiếp.
   * **Gini hoặc Entropy đạt giá trị nhỏ**: Nếu một nhóm đạt độ thuần nhất cao (các phần tử đều thuộc một lớp), cây có thể dừng chia nhánh.
4. **Xử lý nút lá**: Khi đạt đến một nút lá (kết thúc phân chia), thuật toán sẽ gán nhãn lớp cho nút đó (phân loại) hoặc gán giá trị trung bình (hồi quy).

**2. Các tiêu chí phân chia**

* **Gini Index**: Đo lường độ thuần nhất của dữ liệu sau khi chia. Giá trị Gini càng nhỏ, dữ liệu càng thuần nhất. Gini của một tập dữ liệu tính như sau:



* trong đó pip\_ipi​ là xác suất của lớp iii trong tập dữ liệu và CCC là số lớp.
* **Entropy (tùy chọn)**: Đo lường lượng thông tin của dữ liệu. Entropy càng nhỏ thì dữ liệu càng thuần nhất. Tuy nhiên, CART mặc định sử dụng Gini Index thay vì Entropy.

**3. Ưu điểm và Nhược điểm của CART**

**Ưu điểm:**

* **Dễ hiểu và dễ diễn giải**: Cây quyết định tạo ra các điều kiện phân chia rõ ràng, dễ hiểu và có thể được biểu diễn dưới dạng đồ thị.
* **Hỗ trợ cả phân loại và hồi quy**: CART có thể áp dụng cho nhiều loại bài toán.
* **Ít bị ảnh hưởng bởi dữ liệu nhiễu**: Đặc biệt khi cây quyết định được cắt tỉa hợp lý.

**Nhược điểm:**

* **Dễ bị quá khớp**: Nếu không có điều kiện dừng hợp lý, cây có thể trở nên phức tạp và không tổng quát.
* **Nhạy cảm với dữ liệu mất cân bằng**: CART có thể thiên vị nếu một lớp có số lượng mẫu vượt trội.
* **Không hiệu quả cho các bài toán phi tuyến tính phức tạp**: CART đơn thuần khó nắm bắt các mối quan hệ phi tuyến tính phức tạp mà cần kết hợp nhiều cây (như Random Forest) để tăng độ chính xác.