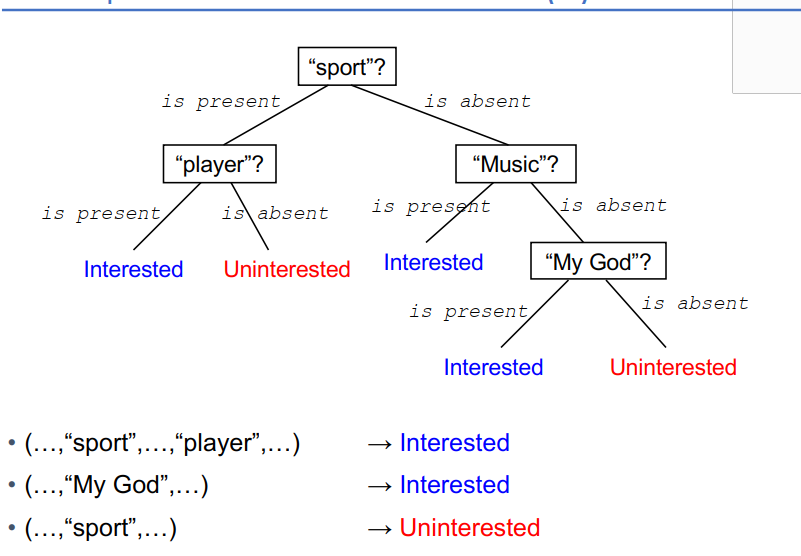
Decision tree: Dùng để biểu diễn 1 func bằng cách sử dụng Tree

+Mỗi cây quyết định có thể hiểu là 1 tập của các IF-Then

* Cây quyết định được sử dụng nhiều trong các ứng dụng thực tế

***Ex:*** ******

Diagram

Description automatically generated

Mỗi nút bên trong đại diện cho một thuộc tính để kiểm tra dữ liệu đến.

Mỗi nhánh/cây con của một nút tương ứng với một giá trị của thuộc tính của nút đó.

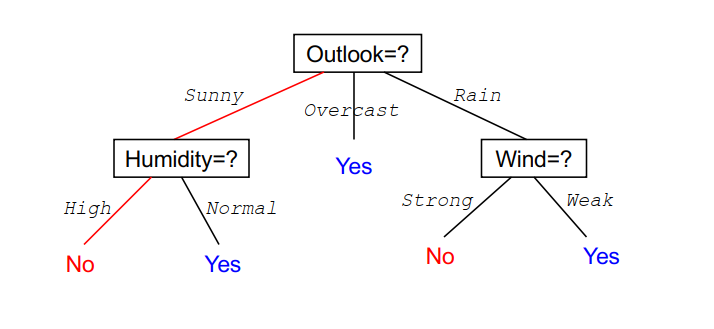
Mỗi nút lá đại diện cho một nhãn lớp.

Khi một cây đã được học, chúng ta có thể dự đoán nhãn cho một phiên bản mới bằng cách sử dụng các thuộc tính của nó để di chuyển từ gốc xuống một chiếc lá.

¨ Nhãn của lá sẽ được sử dụng để gán cho thể hiện mới.

Mỗi đường dẫn từ gốc đến lá là sự kết hợp/VÀ của các bài kiểm tra thuộc tính.

Bản thân cây quyết định là sự phân tách/HOẶC của những liên từ.



Diagram

Description automatically generated

**Học = ID3**

ID3 (Iterative Dichotomiser 3) là một thuật toán tham lam mà được đề xuất bởi Ross Quinlan vào năm 1986.

Nó sử dụng sơ đồ từ trên xuống.

Tại mỗi nút N, chọn thuộc tính kiểm tra A có thể giúp chúng tôi tốt nhất là phân loại cho dữ liệu trong N.

¨ Tạo một nhánh cho mỗi giá trị của A, sau đó tách các dữ liệu vào các nhánh của nó cho phù hợp.

Phát triển cây cho đến khi:

¨ Nó phân loại chính xác tất cả dữ liệu huấn luyện; hoặc là

¨ Tất cả các thuộc tính được sử dụng.

*Lưu ý:* mỗi thuộc tính chỉ được xuất hiện nhiều nhất một lần trong bất kỳ

đường đi của cây.

**ID3\_alg**(*Training\_Set, Class\_Labels, Attributes*)  
Generate the Root of the tree  
If all of *Training\_Set* belong to class c, then Return Root as leaf with label c  
If *Attributes* is empty, then  
Return Root as leaf with label c = **Majority\_Class\_Label**(*Training\_Set*)  
A ← a set of *Attributes* that are best discriminative for *Training\_Set*Let A be the test attributes of Root  
For each value v of A  
Generate a branch of Root which corresponds with v.  
Determine Training\_Setv = { x in *Training\_Set* | xA = v}  
If (Training\_Setv is empty) Then  
Generate a leaf with class label c = **Majority\_Class\_Label**(*Training\_Set*)  
Else  
Generate a subtree by **ID3\_alg**(Training\_Setv, Class\_Labels, Attributes \{A})  
Return Root

Tại mỗi nút, làm thế nào chúng ta có thể chọn một tập hợp các thuộc tính kiểm tra?

¨ Các thuộc tính này phải có tính phân biệt, nghĩa là có thể giúp chúng ta phân loại tốt dữ liệu bên trong nút đó.

Làm thế nào để biết một thuộc tính là phân biệt đối xử?

Ví dụ: giả sử có 2 lớp trong dữ liệu, lớp nào là A1 và A2 nên được chọn làm thuộc tính kiểm tra?

Diagram

Description automatically generated

Lớp A2 sẽ tốt hơn , tại vì dữ liệu A1 chênh lệch giữa c1 và c2 quá ít.và A2 ít nhánh hơn và tối ưu sẽ ok hơn.

***Information gain*** can help.

Information gain: entropy

Entropy đo lường tạp chất/tính không đồng nhất của một tập hợp.

Text, letter

Description automatically generated

Đối với 2 lớp: entropy(S) = - p1log2p1 - p2log2p2

Ý nghĩa của entropy trong Lý thuyết thông tin:

¨ Entropy cho biết số bit trung bình để mã hóa một lớp S.

¨ Entropy của một tin nhắn đo lượng thông tin trung bình

có trong thông điệp đó.

¨ Entropy của một biến ngẫu nhiên x đo độ khó dự đoán của x.

S gồm 14 ví dụ trong đó 9 thuộc lớp c1 và 5 thuộc lớp c2.

¡ Vậy entropy của S là:

Entropy(S)

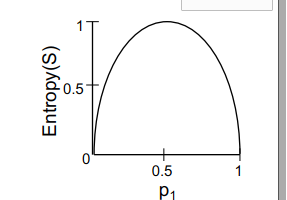
= -(14/9).log2(14/9) -(14/5).log2(14/5)

≈ 0,94

¡ Entropy = 0 nếu tất cả các ví dụ trong S có cùng nhãn.

¡ Entropy = 1 nếu hai lớp trong S có kích thước bằng nhau.

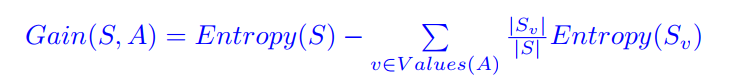
¡ Mặt khác, entropy sẽ luôn thuộc về (0, 1)



Độ lợi thông tin của một thuộc tính trong S:

¨ Đo mức giảm entropy nếu chúng ta chia S thành các tập con theo thuộc tính đó.

Độ lợi thông tin của thuộc tính A trong S được định nghĩa là:



Where Values(A) is the set of all values of A, and  
Sv= {**x** | **x** in S, and xA = v}

Thuật ngữ thứ hai trong Gain(S,A) đo lường thông tin còn lại khi S được chia thành các tập con theo các giá trị của A

Ý nghĩa của Gain(S,A): lượng thông tin trung bình s mất đi khi chia S theo A .

Một tập hợp S các quan sát về một người chơi quần vợt.

Table

Description automatically generated

Text, letter

Description automatically generated

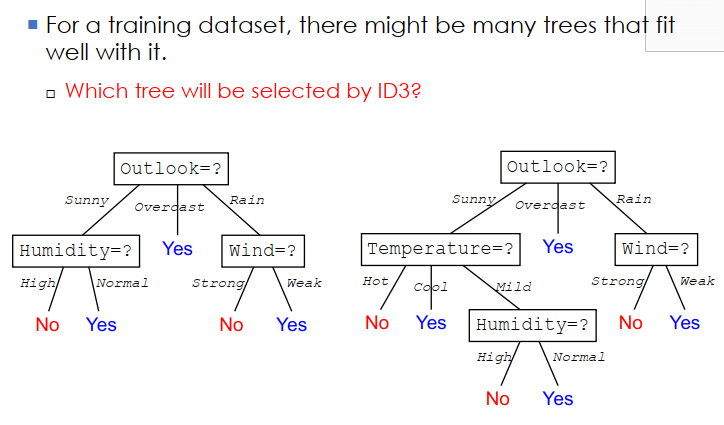
Diagram

Description automatically generated

Diagram

Description automatically generated

Cây quyết định là cây thấp hơn ,bởi vì nó được tối ưu thuật toán tốt và cho ra kết quả đúng ngay cả khi bỏ qua 1 attribute nào đó,…



ID3: lược đồ tìm kiếm(searching scheme)

ID3 chọn cây đầu tiên phù hợp với dữ liệu huấn luyện,

¨Bởi vì nó không bao giờ xem xét lại các lựa chọn của mình khi trồng cây.

Vậy sơ đồ tìm kiếm của ID3:

¨ Thích những cây đơn giản.

¨ Thích những cây trong đó các thuộc tính có mức tăng thông tin cao hơn

sẽ được đặt gần rễ hơn.

***Một số vấn đề với ID3:***

Các cây đã học có thể khớp quá mức với dữ liệu huấn luyện.

* Nên dẫn đến overfitting ở ID3

Làm thế nào để làm việc với các thuộc tính thực?

¨ Nhiều ứng dụng có đầu vào thực.

Có biện pháp nào tốt hơn là thu được thông tin không?

Làm thế nào để xử lý các giá trị bị thiếu?

¨ Thiếu giá trị là một vấn đề cố hữu trong nhiều các ứng dụng.

Làm cách nào để đính kèm chi phí của các thuộc tính trong ID3?

***Vd*** về overfitting, hoạt động tốt trên tập dữ liệu train nhưng lại tệ trên tập thực tế.

Chart, line chart

Description automatically generated

***2 giải pháp:***

¨ Ngừng học sớm: ngăn cây trước khi nó phù hợp với việc đào tạo dữ liệu một cách hoàn hảo.

¨ Cắt tỉa toàn bộ cây: phát triển cây đến kích thước đầy đủ của nó, sau đó đăng tỉa cây.

Thật khó để quyết định khi nào nên ngừng học.

Sau khi cắt tỉa cây theo kinh nghiệm cho kết quả tốt hơn hiệu suất. Nhưng mà

¨ Làm thế nào để quyết định kích thước tốt của cây?

¨ Khi nào ngừng cắt tỉa?

Chúng ta có thể sử dụng bộ xác thực để thực hiện cắt bớt, chẳng hạn như, cắt tỉa giảm lỗi và cắt tỉa theo quy tắc

***Attribute selection***

Information gain:

¨ Ưu tiên thuộc tính có nhiều giá trị độc đáo hơn.

¨ Các thuộc tính có nhiều giá trị độc đáo hơn sẽ được đặt gần hơn với

root hơn thuộc tính khác.

Chúng ta có thể sử dụng một số biện pháp khác, chẳng hạn như Tỷ lệ tăng (Gain Ratio)

Text

Description automatically generated

***Mất hoặc thiếu giá trị thực tế:***

Làm thế nào để làm việc với các thuộc tính thực?

¨ Các thuộc tính/đặc trưng thực phổ biến trong thực tế.

¨ Một cách là rời rạc hóa, tức là chuyển đổi một thuộc tính thực

thành một cái rời rạc bằng cách chia miền của thuộc tính đó thành một tập hợp các khoảng.

Ví dụ: [0, 1] à { [0, 0,25); [0,25; 0,5); [0,5; 0,75); [0,75, 1]}

Làm thế nào để đối phó với các giá trị bị thiếu?

¨ Các giá trị còn thiếu vốn có trong các ứng dụng thực tế.

¨ Một quan sát x có thể không có giá trị xA.

¨ Cách 1: điền xA là giá trị phổ biến nhất của A trong tập huấn luyện dữ liệu.

¨ Cách 2: điền xA là giá trị phổ biến nhất của A trong tập huấn luyện dữ liệu thuộc cùng một lớp với x.

***Random Forest:***

Rừng ngẫu nhiên (RF) là một phương pháp của Leo Breiman (2001) để cả phân loại và hồi quy (classification and regression).

Ý chính: dự đoán dựa trên sự kết hợp của nhiều cây quyết định, bằng cách lấy giá trị trung bình của tất cả các cây quyết định phỏng đoán.

Mỗi cây trong RF đơn giản nhưng ngẫu nhiên.

Mỗi cây được trồng khác nhau, tùy thuộc vào sự lựa chọn của thuộc tính và dữ liệu huấn luyện.

RF hiện là một trong những công cụ phổ biến và chính xác nhất

phương pháp [Fernández-Delgado và cộng sự, 2014]

¨ Nó cũng rất chung chung.

RF có thể được triển khai dễ dàng và hiệu quả.

Nó có thể hoạt động với các bài toán có kích thước rất cao mà không có overfitting.

Tuy nhiên, người ta biết rất ít về các đặc tính lý thuyết của nó

A group of trees

Description automatically generated with medium confidence

***Three basic ingredients(3 thành phần cơ bản):***

**Ngẫu nhiên và không cắt tỉa (Randomization and no pruning):**

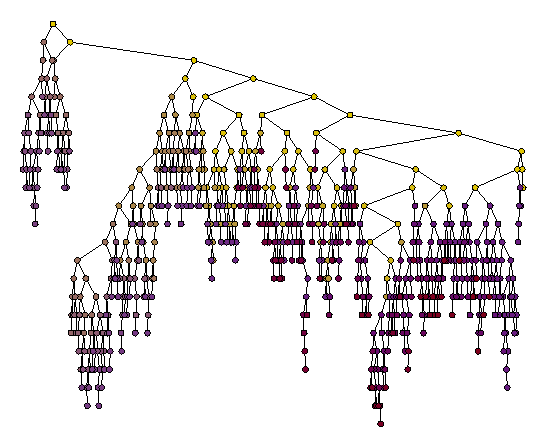
¨ Đối với mỗi cây và tại mỗi nút, chúng tôi chọn ngẫu nhiên một tập hợp con của các thuộc tính.

¨ Tìm cách tách tốt nhất, sau đó phát triển các cây con thích hợp.

¨ Mỗi cây sẽ phát triển đến kích thước lớn nhất mà không cần cắt tỉa.

**Kết hợp (Combination):** mỗi dự đoán sau này được thực hiện bằng cách lấy trung bình của tất cả các dự đoán của từng cây.

**Đóng bao (Bagging):** tập huấn luyện cho mỗi cây được tạo bởi lấy mẫu (có thay thế) từ dữ liệu gốc.



***Algorithm (Thuật toán)***

***Đầu vào (Input):*** dữ liệu huấn luyện D

***Học (Learn):*** trồng cây K như sau

¨ Tạo tập huấn luyện Di bằng cách lấy mẫu với sự thay thế từ D.

¨ Tìm hiểu cây thứ i từ Di:

¨ Tại mỗi nút:

Chọn ngẫu nhiên một tập con S các thuộc tính.

Chia nút thành các cây con theo S.

¨ Phát triển cây này đến kích thước lớn nhất mà không cần cắt tỉa.

***Dự đoán (Prediction):*** lấy giá trị trung bình của tất cả các dự đoán từ cây riêng lẻ.

***practical performance (Hiệu suất thực tế):***

RF được so sánh rộng rãi với các phương pháp khác

¨ Bởi Fernández-Delgado et al. (2014).

¨ Sử dụng 55 bài toán khác nhau.

¨ Sử dụng độ chính xác trung bình (µP) làm thước đo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Classifier | *µP* | No. | Classifier | *µP* |
| 1  2  3  4  5  6  7  8  9  10 | rf t  parRF t  svm C  RRF t  RRFglobal t  LibSVM w  RotationForest w  C5.0 t  rforest R  treebag t | 91.1  91.1  90.7  90.6  90.6  90.6  90.5  90.5  90.3  90.2 | 11  12  13  14  15  16  17  18  19  20 | Bagging LibSVM w  RandomCommittee w  Bagging RandomTree w  MultiBoostAB RandomTree w  MultiBoostAB LibSVM w  MultiBoostAB PART w  Bagging PART w  AdaBoostM1 J48 w  Bagging REPTree w  MultiBoostAB J48 w | 89.9 89.9 89.8 89.8 89.8 89.7 89.7 89.5 89.5 89.4 |