Học Máy

(Machine Learning)

Thân Quang Khoát

khoattq@soict.hust.edu.vn

Viện Công nghệ thông tin và Truyền thông
Trường Đại học Bách Khoa Hà Nội

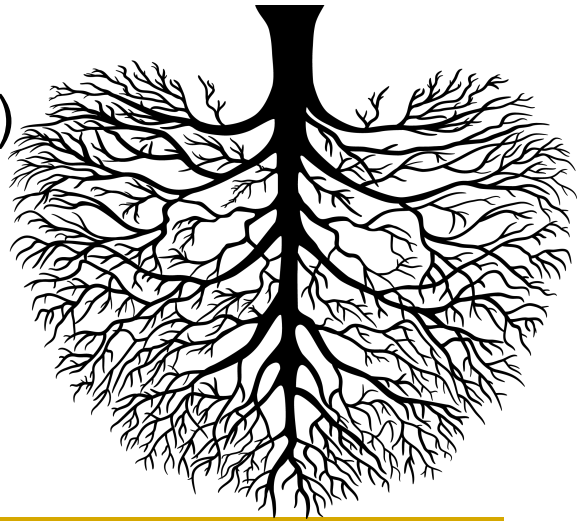
Năm 2015

Nội dung môn học:

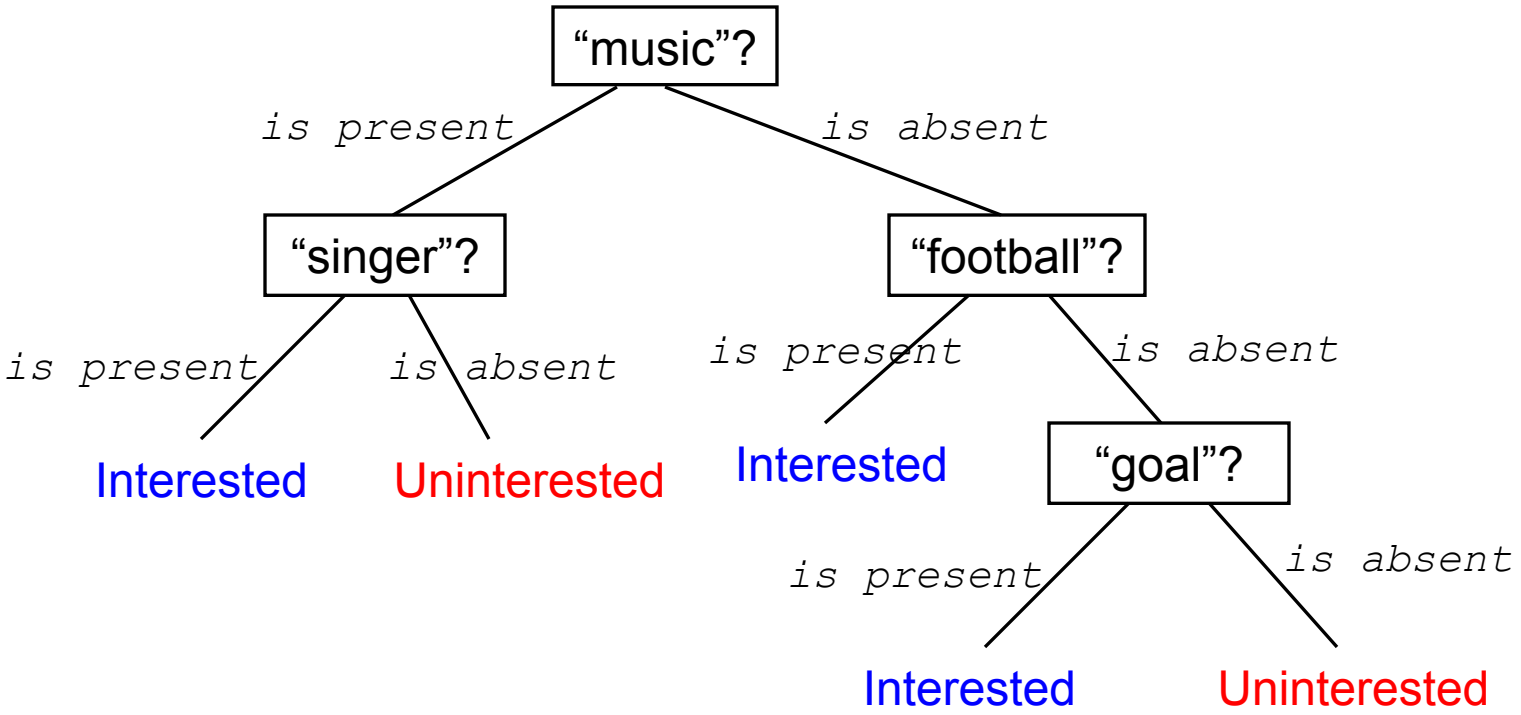
- Giới thiệu chung
- Các phương pháp học không giám sát
- **Các phương pháp học có giám sát**
  - **Cây quyết định và Rừng ngẫu nhiên (Decision tree & Random forest)**
- Đánh giá hiệu năng hệ thống học máy

1. Cây quyết định

- **Cây quyết định** (Decision tree)
  - Để học (xấp xỉ) một hàm mục tiêu có giá trị rời rạc (*discrete-valued target function*): hàm phân lớp
  - Hàm phân lớp được biểu diễn bởi một cây quyết định
- Một cây quyết định có thể được biểu diễn (diễn giải) bằng một tập các luật IF-THEN (dễ đọc và dễ hiểu)
- Học cây quyết định có thể thực hiện ngay cả với các dữ liệu có chứa nhiễu/lỗi (noisy data)
- Được áp dụng thành công trong rất nhiều các bài toán ứng dụng thực tế

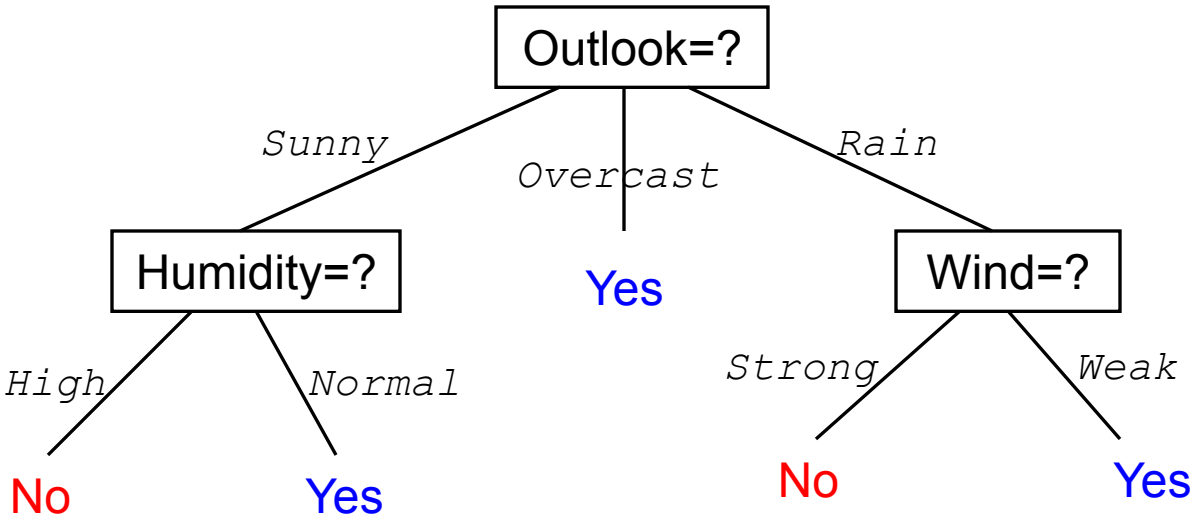


Ví dụ về DT: Những tin tức nào mà tôi quan tâm?



- (…,"music",…,"singer",…) → Interested
- (…,"goal",…) → Interested
- (…,"music",…) → Uninterested

Ví dụ về DT: Một người có chơi tennis không?



- (Outlook=`Overcast`, Temperature=`Hot`, Humidity=`High`, Wind=`Weak`) → Yes
- (Outlook=`Rain`, Temperature=`Mild`, Humidity=`High`, Wind=`Strong`) → No
- (Outlook=`Sunny`, Temperature=`Hot`, Humidity=`High`, Wind=`Strong`) → No

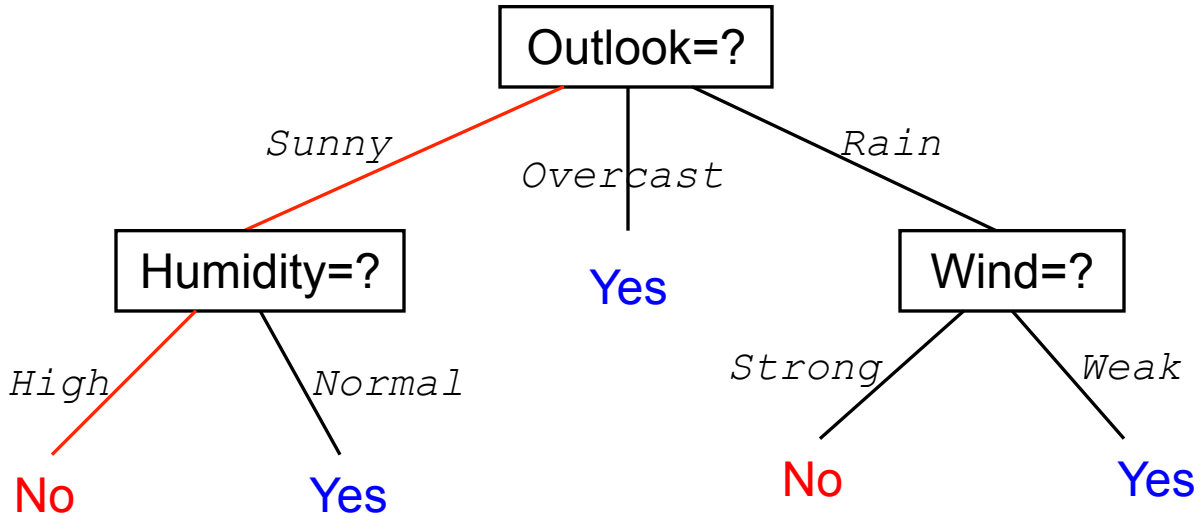
Biểu diễn cây quyết định (1)

- Mỗi nút trong (*internal node)* biểu diễn một thuộc tính cần kiểm tra giá trị đối với các ví dụ.
- Mỗi nhánh (*branch*) từ một nút sẽ tương ứng với một giá trị có thể của thuộc tính gắn với nút đó.
- Mỗi nút lá (*leaf node*) biểu diễn một lớp.
- Một cây quyết định học được sẽ phân lớp đối với một ví dụ, bằng cách duyệt cây từ nút gốc đến một nút lá

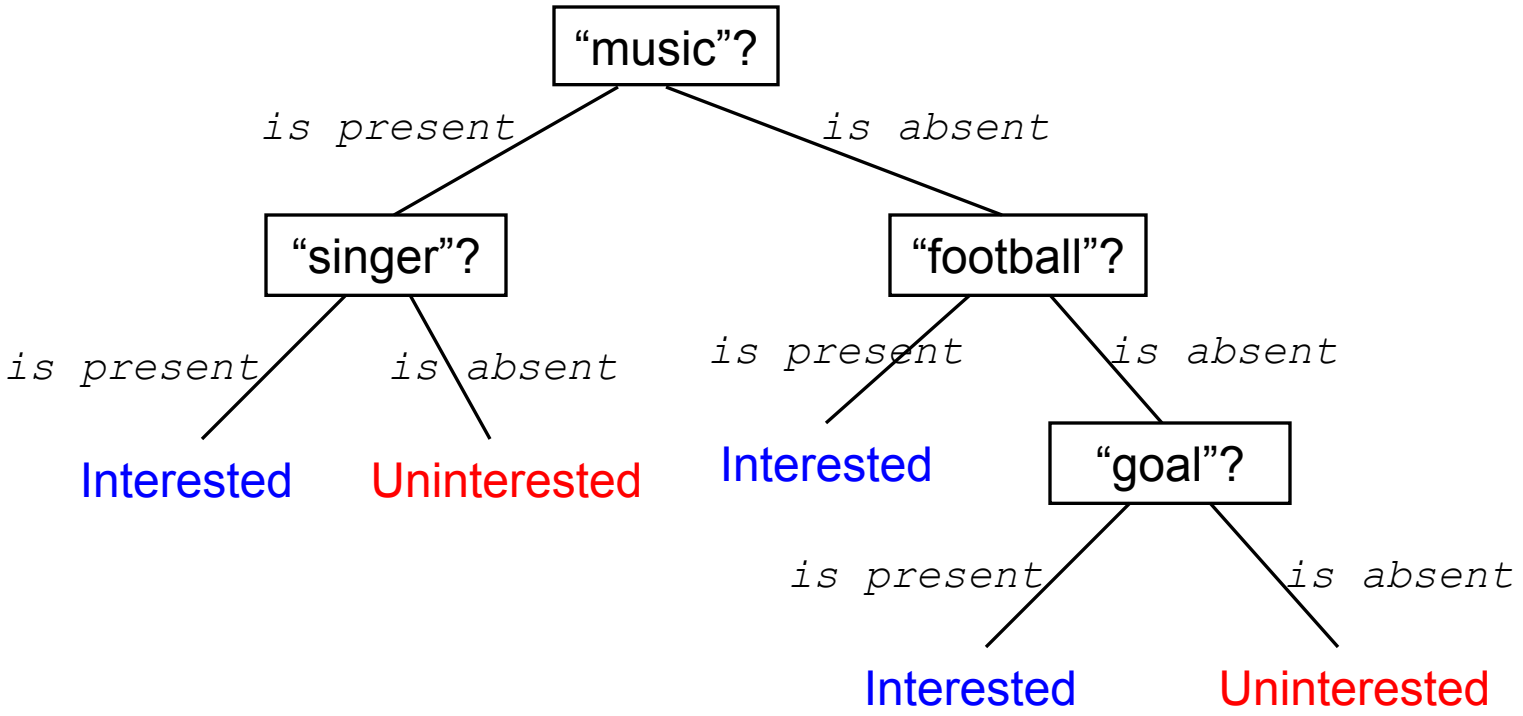
  → Nhãn lớp gắn với nút lá đó sẽ được gán cho ví dụ cần phân lớp.

Biểu diễn cây quyết định (2)

- Mỗi đường đi (path) từ nút gốc đến một nút lá sẽ tương ứng với một kết hợp (conjunction) của các kiểm tra giá trị thuộc tính (attribute tests).
- Cây quyết định (bản thân nó) chính là một phép tuyển (disjunction) của các kết hợp (conjunctions) này.



Những tin tức nào mà tôi quan tâm?

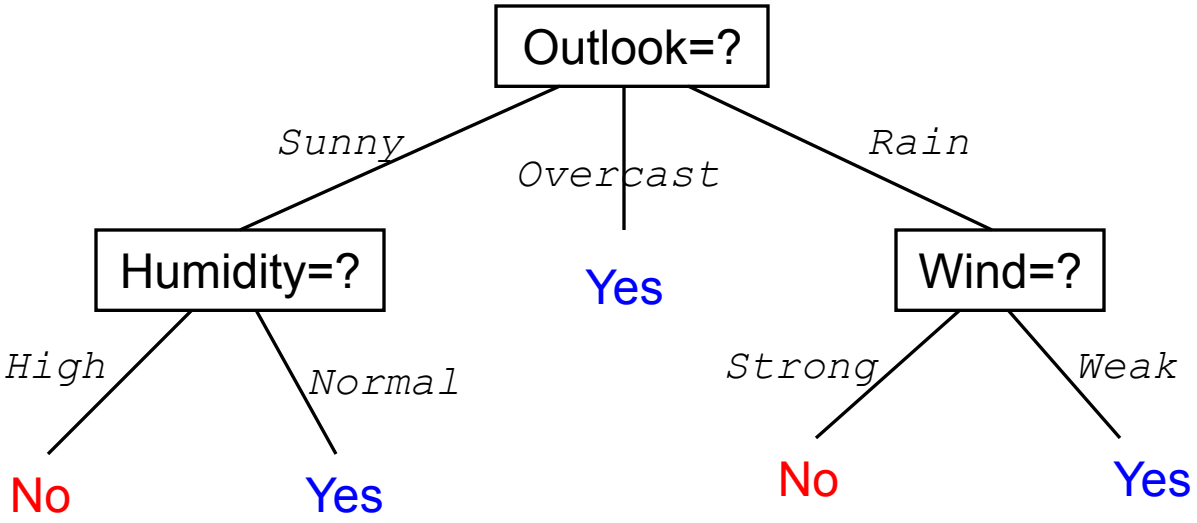


[("music" is present) ∧ ("singer" is present)] ∨

[("music" is absent) ∧ ("football" is present)] ∨

[("music" is absent) ∧ ("football" is absent) ∧ ("goal" is present)]

Một người có chơi tennis không?



[(Outlook=Sunny) ∧ (Humidity=Normal)] ∨

(Outlook=Overcast) ∨

[(Outlook=Rain) ∧ (Wind=Weak)]

2. Học cây quyết định bằng ID3

- **ID3** (Iterative Dichotomiser 3) thực hiện tìm kiếm tham lam trên không gian các cây quyết định (do Ross Quinlan đề xuất năm 1986).
- Xây dựng (học) một cây quyết định theo chiến lược top-down, bắt đầu từ nút gốc.
- *Ở mỗi nút, chọn thuộc tính kiểm tra (là thuộc tính có khả năng phân loại tốt nhất đối với các ví dụ học gắn với nút đó).*
- *Tạo mới một cây con của nút hiện tại cho mỗi giá trị của thuộc tính kiểm tra, và tập học sẽ được tách ra (thành các tập con) tương ứng với cây con vừa tạo.*
- *Quá trình phát triển cây quyết định sẽ tiếp tục cho đến khi:*
  - *Cây quyết định phân loại hoàn toàn các ví dụ học, hoặc*
  - *Tất cả các thuộc tính đã được sử dụng*
- Chú ý: Mỗi thuộc tính chỉ được phép xuất hiện tối đa 1 lần đối với bất kỳ một đường đi nào trong cây.

Giải thuật ID3

ID3_alg(`Training_Set`, `Class_Labels`, `Attributes`)

Tạo nút `Root` của cây quyết định

If tất cả các ví dụ của `Training_Set` thuộc cùng lớp `c`, Return Cây quyết định có nút `Root` được gắn với (có nhãn) lớp `c`

If Tập thuộc tính `Attributes` là rỗng, Return Cây quyết định có nút `Root` được gắn với nhãn lớp ≡ **Majority_Class_Label**(`Training_Set`)

`A` ← Thuộc tính trong tập `Attributes` có khả năng phân loại "tốt nhất" đối với `Training_Set`

Thuộc tính kiểm tra cho nút `Root` ← `A`

For each Giá trị có thể `v` của thuộc tính `A`

Bổ sung một nhánh cây mới dưới nút `Root`, tương ứng với trường hợp: "Giá trị của `A` là `v`"

Xác định $\texttt{Training\_Set}_v$ = {ví dụ `x` | `x` ⊆ `Training_Set`, x_A=v}

If ($\texttt{Training\_Set}_v$ là rỗng) Then

Tạo một nút lá với nhãn lớp ≡ **Majority_Class_Label**(`Training_Set`)

Gắn nút lá này vào nhánh cây mới vừa tạo

Else Gắn vào nhánh cây mới vừa tạo một cây con sinh ra bởi **ID3_alg**($\texttt{Training\_Set}_v$, `Class_Labels`, {`Attributes` \ `A`})

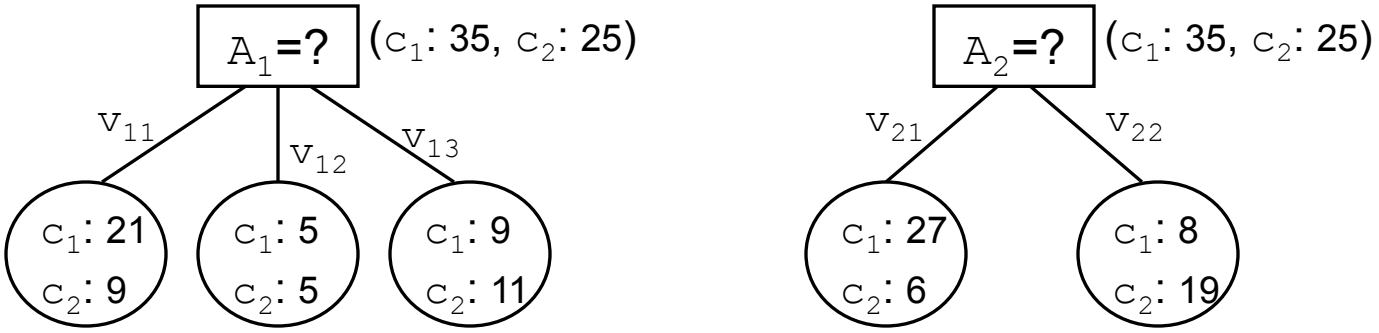
Return `Root`

Lựa chọn thuộc tính kiểm tra

- Tại mỗi nút, chọn thuộc tính kiểm tra như thế nào?
- Chọn thuộc tính quan trọng nhất cho việc phân lớp các ví dụ học gắn với nút đó
- Làm thế nào để đánh giá khả năng của một thuộc tính đối với việc phân tách các ví dụ học theo nhãn lớp của chúng?

  → Sử dụng một đánh giá thống kê – *Information Gain*
- Ví dụ: Xét bài toán phân lớp có 2 lớp (c_1, c_2)

  → Thuộc tính nào, A_1 hay A_2, nên được chọn là thuộc tính kiểm tra?



Information gain: Entropy

- Entropy đo mức độ hỗn tạp (impurity/inhomogeneity) của một tập
- Entropy của tập S đối với việc phân lớp có c lớp

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

trong đó p_i là tỷ lệ các ví dụ trong tập S thuộc vào lớp i, và $0 \log_2 0 = 0$

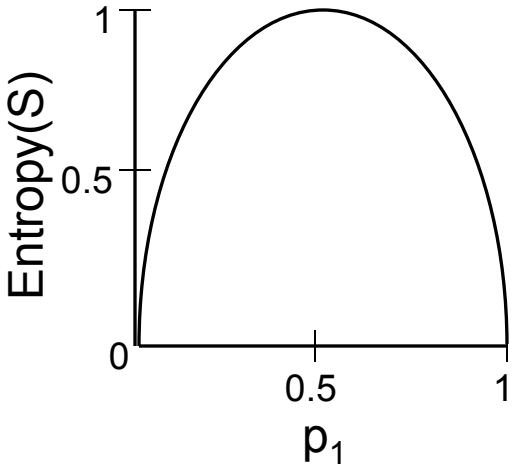
- Entropy của tập S đối với việc phân lớp có 2 lớp

$$Entropy(S) = -p_1.\log_2 p_1 - p_2.\log_2 p_2$$

- Ý nghĩa của entropy trong lĩnh vực Information Theory
  - → Entropy chỉ ra số bits trung bình cần thiết để mã hóa một lớp trong S.
  - → Entropy của một message đo giá trị trung bình của lượng thông tin chứa trong message đó.
  - → Entropy của một biến ngẫu nhiên x đo mức độ không đoán được x.

Information gain: Ví dụ về Entropy

- S gồm 14 ví dụ, trong đó 9 ví dụ thuộc về lớp c_1 và 5 ví dụ thuộc về lớp c_2
- Entropy của tập S đối với phân lớp có 2 lớp:
  Entropy(S)
  $= -(9/14).\log_2(9/14) -(5/14).\log_2(5/14)$
  ≈ 0.94



- Entropy =0, nếu tất cả các ví dụ thuộc cùng một lớp (c_1 hoặc c_2)
- Entropy =1, số lượng các ví dụ thuộc về lớp c_1 bằng số lượng các ví dụ thuộc về lớp c_2
- Entropy thuộc (0,1), nếu như số lượng các ví dụ thuộc về lớp c_1 khác với số lượng các ví dụ thuộc về lớp c_2

Information gain

- Information Gain của một thuộc tính đối với một tập S:
  - Đo mức độ giảm Entropy nếu chia S theo các giá trị của thuộc tính đó.
- Information Gain của thuộc tính A đối với tập S

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

trong đó Values(A) là tập các giá trị có thể của thuộc tính A, và
S_v = {**x** | **x** thuộc S, và x_a = v}

- Trong công thức trên, thành phần thứ 2 thể hiện giá trị Entropy sau khi tập S được phân chia bởi các giá trị của thuộc tính A.
- Ý nghĩa của Gain(A): *Lượng thông tin (trung bình) giữ lại được nếu chia S theo thuộc tính A.*

Tập các ví dụ học

Xét tập dữ liệu S ghi lại những ngày mà một người chơi (không chơi) tennis:

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

[Mitchell, 1997]

Information Gain: Ví dụ

- Hãy tính giá trị Information Gain của thuộc tính `Wind` đối với tập học `S` – `Gain(S,Wind)`?
- Thuộc tính `Wind` có 2 giá trị có thể: `Weak` và `Strong`
- `S` = {9 ví dụ lớp `Yes` và 5 ví dụ lớp `No`}
- S_{weak} = {6 ví dụ lớp `Yes` và 2 ví dụ lớp `No` có giá trị `Wind=Weak`}
- S_{strong} = {3 ví dụ lớp `Yes` và 3 ví dụ lớp `No` có giá trị `Wind=Strong`}

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - \frac{8}{14} Entropy(S_{Weak}) - \frac{6}{14} Entropy(S_{Strong})$$

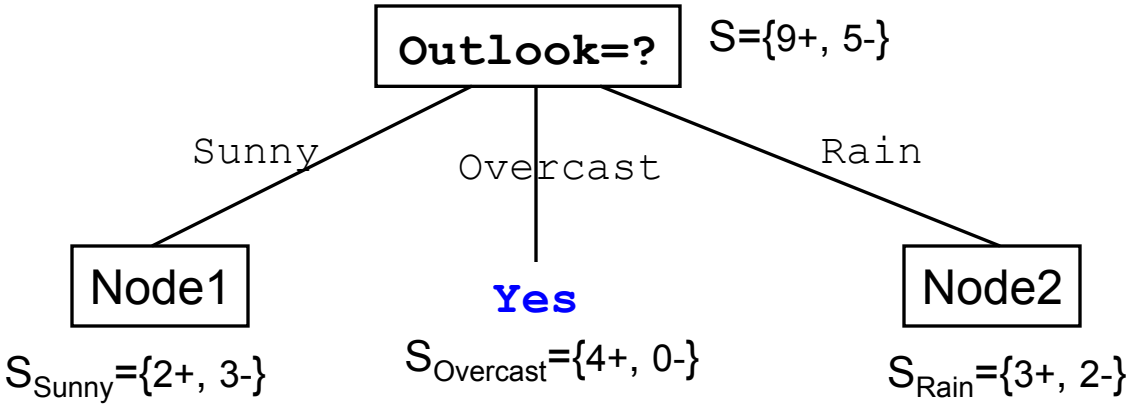
$$= 0.94 - \frac{8}{14} * 0.81 - \frac{6}{14} * 1 = 0.048$$

Học cây quyết định: Ví dụ.

- Tại nút gốc, thuộc tính nào trong số {Outlook, Temperature, Humidity, Wind} nên được chọn là thuộc tính kiểm tra?
  - Gain(S, **Outlook**) = ... = **0.246**
  - Gain(S, Temperature) = ... = 0.029
  - Gain(S, Humidity) = ... = 0.151
  - Gain(S, Wind) = ... = 0.048

← Có giá trị IG cao nhất

→Vì vậy, Outlook được chọn là thuộc tính kiểm tra cho nút gốc!



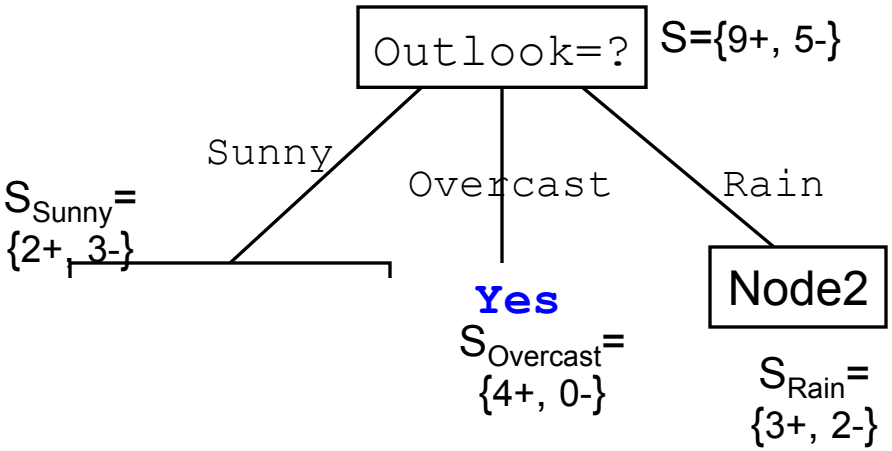
Học cây quyết định: Ví dụ..

- Tại nút `Node1`, thuộc tính nào trong số {`Temperature`, `Humidity`, `Wind`} nên được chọn là thuộc tính kiểm tra?

  Lưu ý! Thuộc tính `Outlook` bị loại ra, bởi vì nó đã được sử dụng bởi cha của nút `Node1` (là nút gốc)

  - Gain(S_{Sunny}, `Temperature`) =...= 0.57
  - Gain(S_{Sunny}, **`Humidity`**) = ... = **0.97**
  - Gain(S_{Sunny}, `Wind`) = ... = 0.019

→Vì vậy, `Humidity` được chọn là thuộc tính kiểm tra cho nút `Node1`!



Học cây quyết định: Chiến lược tìm kiếm.

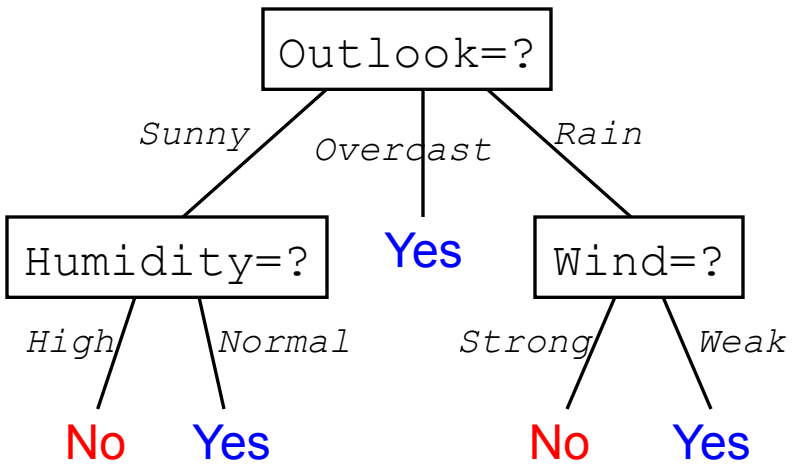
- ID3 tìm kiếm một cây quyết định phù hợp (fits) các ví dụ học, trong không gian các cây quyết định.
- ID3 thực hiện chiến lược tìm kiếm từ đơn giản đến phức tạp, bắt đầu với cây rỗng (empty tree).
- Quá trình tìm kiếm của ID3 được điều khiển bởi độ đo đánh giá Information Gain.
- ID3 chỉ tìm kiếm một (chứ không phải tất cả các) cây quyết định phù hợp với các ví dụ học.

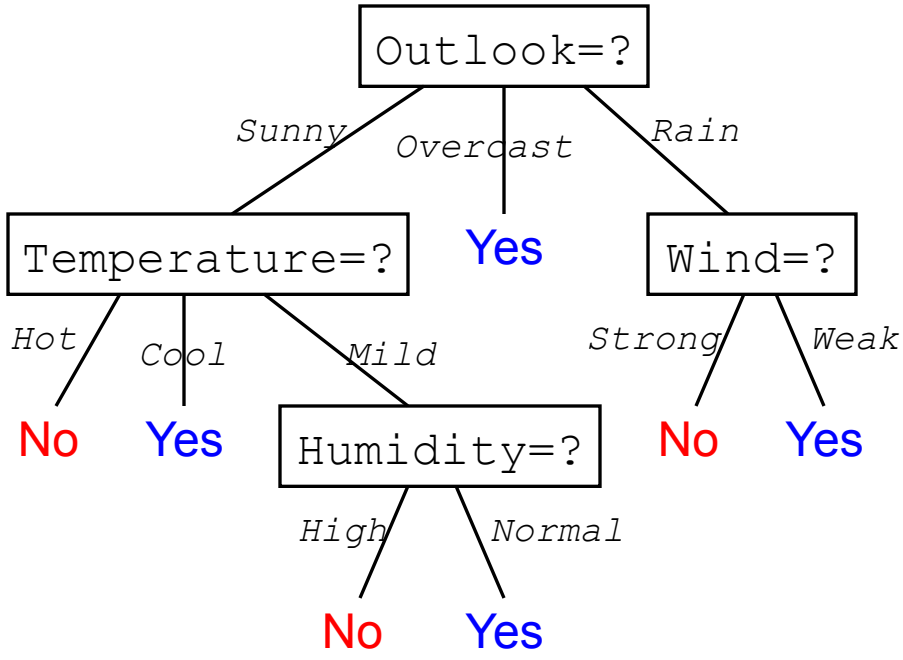
Học cây quyết định: Chiến lược tìm kiếm..

- ID3 thực hiện tìm kiếm tham lam
  - → Chỉ đảm bảo tìm được lời giải tối ưu cục bộ (locally optimal solution) – chứ không đảm bảo tìm được lời giải tối ưu tổng thể (globally optimal solution).
  - → Một khi một thuộc tính được chọn là thuộc tính kiểm tra cho một nút, thì ID3 không bao giờ cân nhắc lại lựa chọn này.
- Ở mỗi bước trong quá trình tìm kiếm, ID3 sử dụng một đánh giá thống kê (Information Gain) để cải thiện giả thiết hiện tại
  - → Nhờ vậy, quá trình tìm kiếm (lời giải) ít bị ảnh hưởng bởi các lỗi/ nhiễu (nếu có) của một số ít ví dụ học (tại sao?)

Ưu tiên trong học cây quyết định.

- Cả 2 cây quyết định dưới đây đều phù hợp với tập học đã cho
- Vậy thì, cây quyết định nào sẽ được ưu tiên (được học) bởi giải thuật ID3?





Ưu tiên trong học cây quyết định..

- Đối với một tập các ví dụ học, có thể tồn tại nhiều (hơn 1) cây quyết định phù hợp với các ví dụ học này
- Cây quyết định nào (trong số đó) được chọn?
- ID3 chọn cây quyết định phù hợp đầu tiên tìm thấy trong quá trình tìm kiếm của nó
  - →Lưu ý là trong quá trình tìm kiếm, giải thuật ID3 không bao giờ cân nhắc lại các lựa chọn trước đó (without backtracking)
- Chiến lược tìm kiếm của giải thuật ID3
  - Ưu tiên các cây quyết định đơn giản (chiều cao cây thấp)
  - Ưu tiên các cây quyết định trong đó một thuộc tính có giá trị *Information Gain càng lớn* thì sẽ là thuộc tính kiểm tra của một nút *càng gần nút gốc*

3. Vài vấn đề trong ID3

- Cây quyết định học được quá phù hợp (over-fit) với các ví dụ học.
- Xử lý các thuộc tính có kiểu giá trị liên tục (kiểu số thực)?
- Các đánh giá phù hợp hơn (tốt hơn Information Gain) đối với việc xác định thuộc tính kiểm tra?
- Xử lý các ví dụ học thiếu giá trị thuộc tính (missing-value attributes)?
- Xử lý các thuộc tính có chi phí (cost) khác nhau?

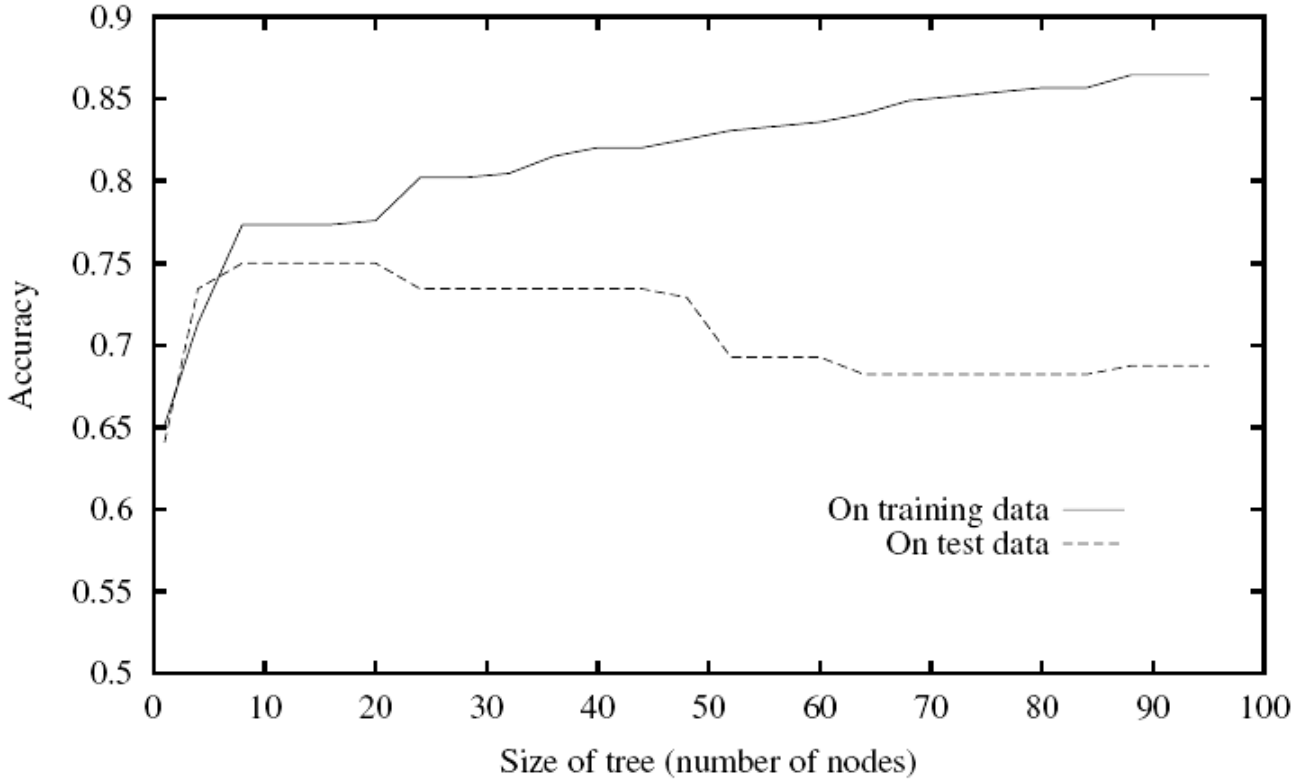
→ Cải tiến của giải thuật ID3 với tất cả các vấn đề nêu trên được giải quyết: giải thuật C4.5

Over-fitting (1)

- Một cây quyết định phù hợp hoàn hảo đối với tập huấn luyện có phải là giải pháp tối ưu?

Over-fitting (2)

Tiếp tục quá trình học cây quyết định sẽ làm giảm độ chính xác đối với tập thử nghiệm mặc dù tăng độ chính xác đối với tập học



[*Mitchell, 1997*]

Over-fitting: Cách giải quyết.

- 2 chiến lược
  - Ngừng học sớm hơn, trước khi nó đạt tới cấu trúc cây cho phép phân loại hoàn hảo tập huấn luyện.
  - Học (phát triển) cây đầy đủ (tương ứng với cấu trúc cây hoàn toàn phù hợp đối với tập huấn luyện), và sau đó thực hiện quá trình tỉa (to post-prune) cây.
- Chiến lược tỉa cây đầy đủ (Post-pruning over-fit trees) thường cho hiệu quả tốt hơn trong thực tế.

  → Lý do: Chiến lược "ngừng sớm" việc học cây cần phải đánh giá chính xác được *khi nào nên ngừng việc học* (phát triển) cây – Khó xác định!

Over-fitting: Cách giải quyết..

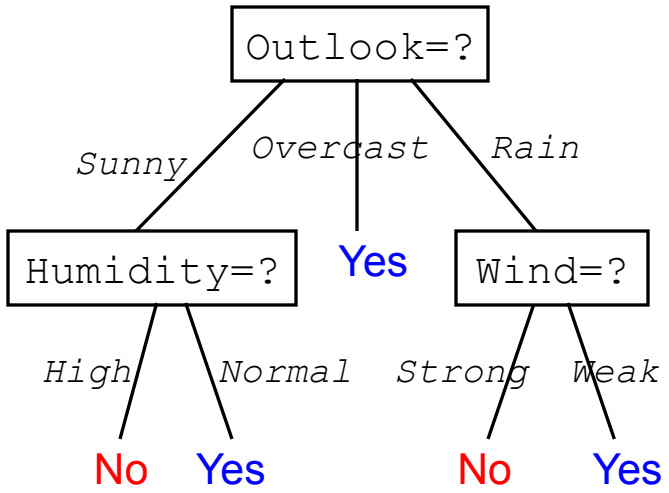
- Làm sao để chọn kích thước "phù hợp" của cây quyết định? Cắt tỉa đến bao giờ?
  - Đánh giá hiệu năng phân loại đối với một tập tối ưu (validation set)
    - Đây là phương pháp thường được sử dụng nhất
    - 2 f.f. chính: *reduced-error pruning* and *rule post-pruning*
  - Áp dụng một thí nghiệm thống kê (vd: chi-square test) để đánh giá xem việc mở rộng (hay cắt tỉa) một nút có giúp cải thiện hiệu năng đối với tập huấn luyện
  - Đánh giá độ phức tạp của việc mã hóa các ví dụ học và cây quyết định, và ngừng việc học (phát triển) cây quyết định khi kích thước của việc mã hóa này là tối thiểu.
    - Dựa trên nguyên lý Minimum Description Length (MDL)
    - Cần cực tiểu hóa: `size`(tree) + `size`(misclassifications(tree))

Reduced-error pruning

- Mỗi nút của cây (khớp hoàn toàn) được kiểm tra để cắt tỉa
- Một nút sẽ bị cắt tỉa nếu cây (sau khi cắt tỉa nút đó) đạt được hiệu năng không tồi hơn cây ban đầu đối với tập tối ưu (validation set)
- Cắt tỉa một nút bao gồm các việc:
  - Loại bỏ toàn bộ cây con (sub-tree) gắn với nút bị cắt tỉa
  - Chuyển nút bị cắt tỉa thành một nút lá (nhãn phân lớp)
  - Gắn với nút lá này (nút bị cắt tỉa) nhãn lớp chiếm số đông trong tập huấn luyện gắn với nút đó
- Lặp lại việc cắt tỉa các nút
  - Luôn lựa chọn một nút mà việc cắt tỉa nút đó tối đa hóa khả năng phân loại của cây quyết định đối với tập tối ưu (validation set)
  - Kết thúc, khi việc cắt tỉa thêm nút làm giảm khả năng phân loại của cây quyết định đối với tập tối ưu (validation set)

Rule post-pruning

- Học (phát triển) cây quyết định hoàn toàn phù hợp với tập huấn luyện
- Chuyển biểu diễn cây quyết định học được thành một tập các luật tương ứng (tạo một luật cho mỗi đường đi từ nút gốc đến một nút lá)
- Rút gọn (tổng quát hóa) mỗi luật (độc lập với các luật khác), bằng cách loại bỏ bất kỳ điều kiện nào giúp mang lại sự cải thiện về hiệu quả phân loại của luật đó
- Sắp xếp các luật đã rút gọn theo khả năng (hiệu quả) phân loại, và sử dụng thứ tự này cho việc phân loại các ví dụ trong tương lai



IF (Outlook=Sunny) ∧
(Humidity=Normal)

THEN (PlayTennis=**Yes**)

Các thuộc tính có giá trị liên tục

- Cần xác định (chuyển đổi thành) các thuộc tính có giá trị rời rạc, bằng cách chia khoảng giá trị liên tục thành một tập các khoảng (intervals) không giao nhau
- Đối với thuộc tính (có giá trị liên tục) A, tạo một thuộc tính mới kiểu nhị phân A_v sao cho: A_v là đúng nếu $A > v$, và là sai nếu ngược lại
- Làm thế nào để xác định giá trị ngưỡng v "tốt nhất"?

  → Chọn giá trị ngưỡng v giúp sinh ra giá trị *Information Gain* cao nhất
- Ví dụ:
  - Sắp xếp các ví dụ học theo giá trị tăng dần đối với thuộc tính $\texttt{Temperature}$
  - Xác định các ví dụ học liền kề nhưng khác phân lớp
  - Có 2 giá trị ngưỡng có thể: $\texttt{Temperature}_{54}$ và $\texttt{Temperature}_{85}$
  - Thuộc tính mới kiểu nhị phân $\texttt{Temperature}_{54}$ được chọn, bởi vì Gain($\texttt{S}$,$\texttt{Temperature}_{54}$) > Gain($\texttt{S}$,$\texttt{Temperature}_{85}$)

Temperature	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

Lựa chọn thuộc tính

- Xu hướng của đánh giá *Information Gain*

  → Ưu tiên các thuộc tính có nhiều giá trị hơn các thuộc tính có ít giá trị

  Vd: Thuộc tính `Date` có số lượng rất lớn các giá trị có thể
  - Thuộc tính này sẽ có giá trị Information Gain cao nhất
  - Một mình thuộc tính này có thể phân loại hoàn hảo toàn bộ tập huấn luyện (thuộc tính này phân chia các ví dụ học thành rất nhiều các tập con có kích thước rất nhỏ)
  - Thuộc tính này được chọn là thuộc tính kiểm tra ở nút gốc (của cây quyết định chỉ có mức độ sâu bằng 1, nhưng rất rộng, rất nhiều phân nhánh)

- Một đánh giá khác: Gain Ratio

  →Giảm ảnh hưởng của các thuộc tính có (rất) nhiều giá trị

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInformation(S,A)},$$

$$SplitInformation(S,A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

(trong đó `Values(A)` là tập các giá trị có thể của thuộc tính `A`, và $S_v = \{x \mid x \in S, x_A = v\}$)

Xử lý các thuộc tính thiếu giá trị

- Giả sử thuộc tính A là một ứng cử cho thuộc tính kiểm tra ở nút n
- Xử lý thế nào với ví dụ x không có (thiếu) giá trị đối với thuộc tính A (tức là: x_A là không xác định)?
- Gọi S_n là tập các ví dụ học gắn với nút n có giá trị đối với thuộc tính A

→Giải pháp 1: x_A là giá trị phổ biến nhất đối với thuộc tính A trong số các ví dụ thuộc tập S_n

→Giải pháp 2: x_A là giá trị phổ biến nhất đối với thuộc tính A trong số các ví dụ thuộc tập S_n có cùng phân lớp với x

Cây quyết định: dùng khi nào?

- Các ví dụ học được biểu diễn bằng các cặp (thuộc tính,giá trị)
  - Phù hợp với các thuộc tính có giá trị rời rạc
  - Đối với các thuộc tính có giá trị liên tục, phải rời rạc hóa
- Hàm mục tiêu có giá trị đầu ra là các giá trị rời rạc
  - Ví dụ: Phân loại các ví dụ vào lớp phù hợp
- Tập huấn luyện có thể chứa nhiễu/lỗi
  - Lỗi trong phân loại (nhãn lớp) của các ví dụ học
  - Lỗi trong giá trị thuộc tính biểu diễn các ví dụ học

4. Cây quyết định cho hồi quy

- Có thể dễ dàng thiết kế cây quyết định cho bài toán hồi quy.
- Thay đổi từ ID3:
  - Thay thế độ đo Information gain, để tìm được một tập các thuộc tính khi muốn phân chia tại mỗi node.
  - Tại mỗi node lá, lưu tất cả các ví dụ đã được phân vào node đó.
- Phán đoán cho quan sát mới **z**:
  - Duyệt cây từ gốc cho tới nút lá, theo các giá trị thuộc tính của **z**.
  - Gọi L là nút lá mà **z** duyệt đến. D(L) là tập các quan sát chứa trong nút lá L, và k là tổng số quan sát trong L.
  - Dự đoán giá trị đầu ra cho **z**:

$$y_z = \frac{1}{k} \sum_{x \in D(L)} y_x$$

5. Rừng ngẫu nhiên (Random forests)

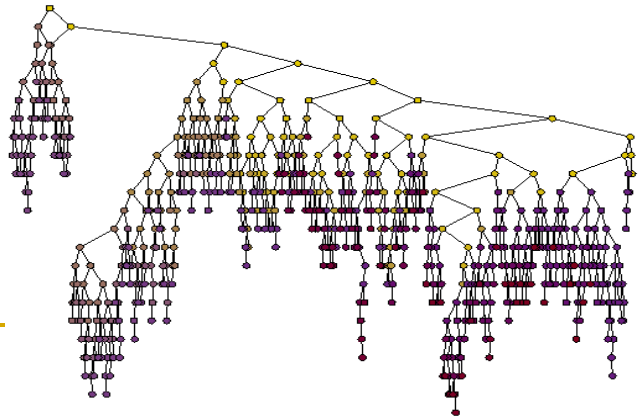
- **Random forests (RF)** là một phương pháp dành cho phân lớp và hồi quy. Được đề xuất bởi Leo Breiman (2001).
- *Ý tưởng:* là một sự kết hợp của các cây quyết định, bằng cách lấy trung bình các phán đoán của các cây.
- Mỗi cây trong đó là 1 cây đơn giản nhưng ngẫu nhiên.
- Mỗi cây được sinh ra phụ thuộc vào cách lựa chọn các thuộc tính trong quá trình học.

5. Rừng ngẫu nhiên (Random forests)

- RF là một trong những phương pháp được dùng phổ biến nhất và mạnh nhất hiện tại [Fernández-Delgado et al., 2014]
- RF có thể cài đặt dễ dàng và nhanh, nhưng có thể tạo ra các phán đoán chính xác và có thể làm việc tốt với những bài toán rất nhiều chiều mà không bị overfitting. ☺
- Tuy nhiên, vẫn khó phân tích các tính chất toán học của RF. ☹

5. RF: 3 thành phần cơ bản

- **Ngẫu nhiên hoá và không cắt tỉa:**
  - Đối với mỗi cây, tại mỗi nút ta chọn ngẫu nhiên một nhóm nhỏ các thuộc tính để chia.
  - Tính mốc chia tốt nhất, và phân nhánh cây.
  - Cây đó sẽ được sinh ra với cỡ lớn nhất, mà không dùng cắt tỉa.
- **Tổng hợp:** mỗi phán đoán về sau thu được bằng cách lấy trung bình các phán đoán từ tất cả các cây.
- **Bagging:** tập học dành cho mỗi cây được sinh ra bằng cách lấy ngẫu nhiên (có trùng lặp) từ tập học ban đầu.



Nguồn: Gerard Biau (2012)

5. RF: thuật toán

- **Đầu vào:** tập học *D*
- Tạo *K* cây, mỗi cây được sinh ra như sau:
  - Xây dựng một tập con D_i bằng cách lấy ngẫu nhiên (có trùng lặp) từ *D*.
  - Học cây thứ *i* từ D_i như sau:
    - Tại mỗi đỉnh (nút):
      - chọn ngẫu nhiên một tập con các thuộc tính
      - phân nhánh cây dựa trên tập thuộc tính đó.
    - Cây này sẽ được sinh ra với cỡ lớn nhất, không dùng cắt tỉa.
- Mỗi phán đoán về sau thu được bằng cách lấy trung bình các phán đoán từ tất cả các cây.

5. RF: so sánh với các phương pháp

- So sánh RF với các phương pháp phân loại khác
  - Được thực hiện bởi [Fernández-Delgado et al., 2014].
  - Sử dụng 55 bài toán khác nhau.
  - Độ chính xác trung bình (μ^P) được dùng để so sánh.

No.	Classifier	μ^P	No.	Classifier	μ^P
1	rf_t	91.1	11	Bagging_LibSVM_w	89.9
2	parRF_t	91.1	12	RandomCommittee_w	89.9
3	svm_C	90.7	13	Bagging_RandomTree_w	89.8
4	RRF_t	90.6	14	MultiBoostAB_RandomTree_w	89.8
5	RRFglobal_t	90.6	15	MultiBoostAB_LibSVM_w	89.8
6	LibSVM_w	90.6	16	MultiBoostAB_PART_w	89.7
7	RotationForest_w	90.5	17	Bagging_PART_w	89.7
8	C5.0_t	90.5	18	AdaBoostM1_J48_w	89.5
9	rforest_R	90.3	19	Bagging_REPTree_w	89.5
10	treebag_t	90.2	20	MultiBoostAB_J48_w	89.4

Tài liệu tham khảo

- L. Breiman. *Random forests*. Machine learning, 45(1), 5-32, 2001.
- Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim. *Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?* Journal of Machine Learning Research, 15(Oct):3133-3181, 2014.
- T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- M. Nunez. *The use of background knowledge in decision tree induction*. Machine Learning, 6(3): 231-250, 1991.
- M. Tan and J. C. Schlimmer. *Two case studies in cost-sensitive concept acquisition*. In Proceedings of the 8th National Conference on Artificial Intelligence, AAAI-90, pp.854-860, 1990.
- Quinlan, J. R. *Induction of Decision Trees*. Mach. Learn. 1, 1 (Mar. 1986), 81-106, 1986