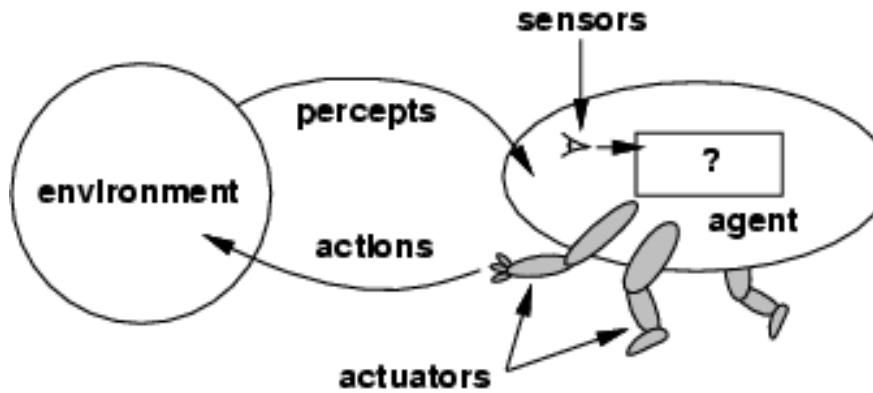


Tác tử - Định nghĩa

- *Tác tử* là bất cứ cái gì (con người, người máy, software robots, các bộ ổn nhiệt,...) có khả năng *cảm nhận* (*nhận biết*) *môi trường* xung quanh nó thông qua các *bộ phận cảm biến* (sensors) và *hành động* phù hợp theo môi trường đó thông qua các *bộ phận hoạt động* (actuators)
- Tác tử con người
 - Các bộ phận cảm biến: mắt, tai, và một số bộ phận cơ thể khác
 - Các bộ phận hoạt động: tay, chân, miệng, và một số bộ phận cơ thể khác
- Tác tử người máy
 - Các bộ phận cảm biến: các máy quay (cameras), các bộ truy tìm tín hiệu hồng ngoại
 - Các bộ phận hoạt động: các loại động cơ (motors)

Tác tử và Môi trường



- Hàm tác tử: là hàm ánh xạ từ lịch sử nhận thức tới các hành động:
$$f: \mathcal{P}^{\star} \rightarrow \mathcal{A}$$
- Chương trình tác tử: hoạt động (chạy) dựa trên kiến trúc thực tế của hàm f
- Tác tử = Kiến trúc + Chương trình

Tác tử máy hút bụi

Bảng hành động của tác tử máy hút bụi

Chuỗi các nhận thức	Hành động
[A, Sạch]	Di chuyển sang phải
[A, Bẩn]	Hút bụi
[B, Sạch]	Di chuyển sang trái
[B, Bẩn]	Hút bụi
[A, Sạch], [A, Sạch]	Di chuyển sang phải
[A, Sạch], [A, Bẩn]	Hút bụi
...	

```
function Reflex-Vacuum-Agent( [location, status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

Tác tử hợp lý (1)

- Tác tử cần phấn đấu để “làm đúng việc cần làm”, dựa trên **những gì nó nhận thức (nhận biết) được** và dựa trên **các hành động mà nó có thể thực hiện**
- Một **hành động đúng (hợp lý)** là hành động giúp cho tác tử đạt được thành công cao nhất đối với mục tiêu đặt ra
- **Đánh giá hiệu quả hoạt động:** là tiêu chuẩn để đánh giá mức độ thành công trong hoạt động của một tác tử
 - Ví dụ: Tiêu chí đánh giá hiệu quả hoạt động của một tác tử máy hút bụi có thể là: *mức độ làm sạch, thời gian hút bụi, mức độ điện năng tiêu tốn, mức độ tiếng ồn gây ra, ...*

Tác tử hợp lý (2)

■ Tác tử hợp lý

- ❑ Với mỗi *chuỗi nhận thức* có được,
- ❑ Một tác tử hợp lý cần phải *lựa chọn một hành động* giúp cực đại hóa tiêu chí đánh giá hiệu quả hoạt động của tác tử đó,
- ❑ Dựa trên các *thông tin* được cung cấp bởi chuỗi nhận thức và các *tri thức* được sở hữu bởi tác tử đó

Tác tử hợp lý (3)

- **Sự hợp lý \neq Sự thông suốt mọi thứ**
 - Sự thông suốt mọi thứ = Biết tất cả mọi thứ, với tri thức vô hạn
 - Vì các nhận thức có thể không cung cấp tất cả các thông tin liên quan
- Các tác tử có thể thực hiện các hành động nhằm thay đổi các nhận thức trong tương lai, với mục đích thu được các thông tin hữu ích (ví dụ: thu thập thông tin, khám phá tri thức)
- **Tác tử *tự trị*** (autonomous agent) là một tác tử mà các hành động của nó được quyết định bởi chính kinh nghiệm của tác tử đó (cùng với khả năng *học* và *thích nghi*)

Môi trường công việc – PEAS (1)

■ PEAS

- Performance measure: Tiêu chí đánh giá hiệu quả hoạt động
 - Environment: Môi trường xung quanh
 - Actuators: Các bộ phận hành động
 - Sensors: Các bộ phận cảm biến
- Để thiết kế một tác tử thông minh (hợp lý), trước tiên cần phải xác định (thiết lập) các giá trị của các thành phần của PEAS

Môi trường công việc – PEAS (2)

■ Ví dụ: Thiết kế một tác tử lái xe taxi tự động

- ❑ Đánh giá hiệu quả hoạt động (P): an toàn, nhạy, đúng luật giao thông, mức độ hài lòng của khách hàng, tối ưu lợi nhuận, ...
- ❑ Môi trường xung quanh (E): các con đường (phố), các phương tiện khác cùng tham gia giao thông, những người đi bộ, các khách hàng, ...
- ❑ Các bộ phận hành động (A): bánh lái, chân ga, phanh, đèn tín hiệu, còi xe, ...
- ❑ Các bộ phận cảm biến (S): máy quay (cameras), đồng hồ tốc độ, GPS, đồng hồ đo khoảng cách quãng đường, các bộ cảm biến động cơ, ...

Các kiểu môi trường (1)

- **Có thể quan sát được hoàn toàn** (hay có thể quan sát được một phần)?
 - Các bộ cảm biến của một tác tử cho phép nó truy cập tới *trạng thái đầy đủ* của môi trường tại mỗi thời điểm
- **Xác định** (hay ngẫu nhiên)?
 - Trạng thái tiếp theo của môi trường được xác định hoàn toàn dựa trên trạng thái hiện tại và hành động của tác tử (tại trạng thái hiện tại này)
 - Nếu một môi trường là xác định, ngoại trừ đối với các hành động của các tác tử khác, thì gọi là *môi trường chiến lược*

Các kiểu môi trường (2)

■ Phân đoạn (hay liên tiếp)?

- ❑ Lịch sử kinh nghiệm của tác tử được chia thành *các giai đoạn* (chương/hồi)
- ❑ Mỗi giai đoạn bao gồm việc nhận thức của tác tử và hành động mà nó thực hiện
- ❑ Ở mỗi giai đoạn, việc lựa chọn hành động để thực hiện chỉ phụ thuộc vào giai đoạn đó (không phụ thuộc vào các giai đoạn khác)

■ Tĩnh (hay động)?

- ❑ Môi trường không thay đổi trong khi tác tử cân nhắc (xem nên đưa ra hành động nào)
- ❑ Môi trường *bán động* (semi-dynamic) là môi trường mà khi thời gian trôi qua thì nó (môi trường) không thay đổi, nhưng hiệu quả hoạt động của tác tử thì thay đổi
 - Ví dụ: Các chương trình trò chơi có tính giờ

Các kiểu môi trường (2)

■ Rời rạc (hay liên tục)?

- ❑ Tập các nhận thức và các hành động là hữu hạn, được định nghĩa phân biệt rõ ràng

■ Tác tử đơn lẻ (hay đa tác tử)?

- ❑ Một tác tử hoạt động độc lập (không phụ thuộc / liên hệ với các tác tử khác) trong một môi trường

Các kiểu môi trường – Ví dụ

	Chơi cờ tính giờ	Chơi cờ không tính giờ	Lái xe taxi
Quan sát đầy đủ?	có	có	không
Xác định?	chiến lược	chiến lược	không
Phân đoạn?	không	không	không
Tĩnh?	bán động	có	không
Rời rạc?	có	có	không
Tác tử đơn?	không	không	không

- Kiểu của môi trường có ảnh hưởng quyết định đối với việc thiết kế tác tử
- Môi trường trong thực tế thường có các đặc điểm: chỉ có thể quan sát được một phần, ngẫu nhiên, liên tiếp, thay đổi (động), liên tục, đa tác tử

Các kiểu tác tử

- 4 kiểu tác tử cơ bản
 - ❑ Tác tử phản xạ đơn giản (simple reflex agents)
 - ❑ Tác tử phản xạ dựa trên mô hình (model-based reflex agents)
 - ❑ Tác tử dựa trên mục tiêu (goal-based agents)
 - ❑ Tác tử dựa trên lợi ích (utility-based agents)

Tác tử phản xạ đơn giản (1)

Tác tử phản xạ đơn giản:

→ Hành động theo một quy tắc (luật) có điều kiện phù hợp với trạng thái hiện thời (của môi trường)

function SIMPLE-REFLEX-AGENT(*percept*)

static: *rules* (tập các luật có dạng: điều kiện-hành động)

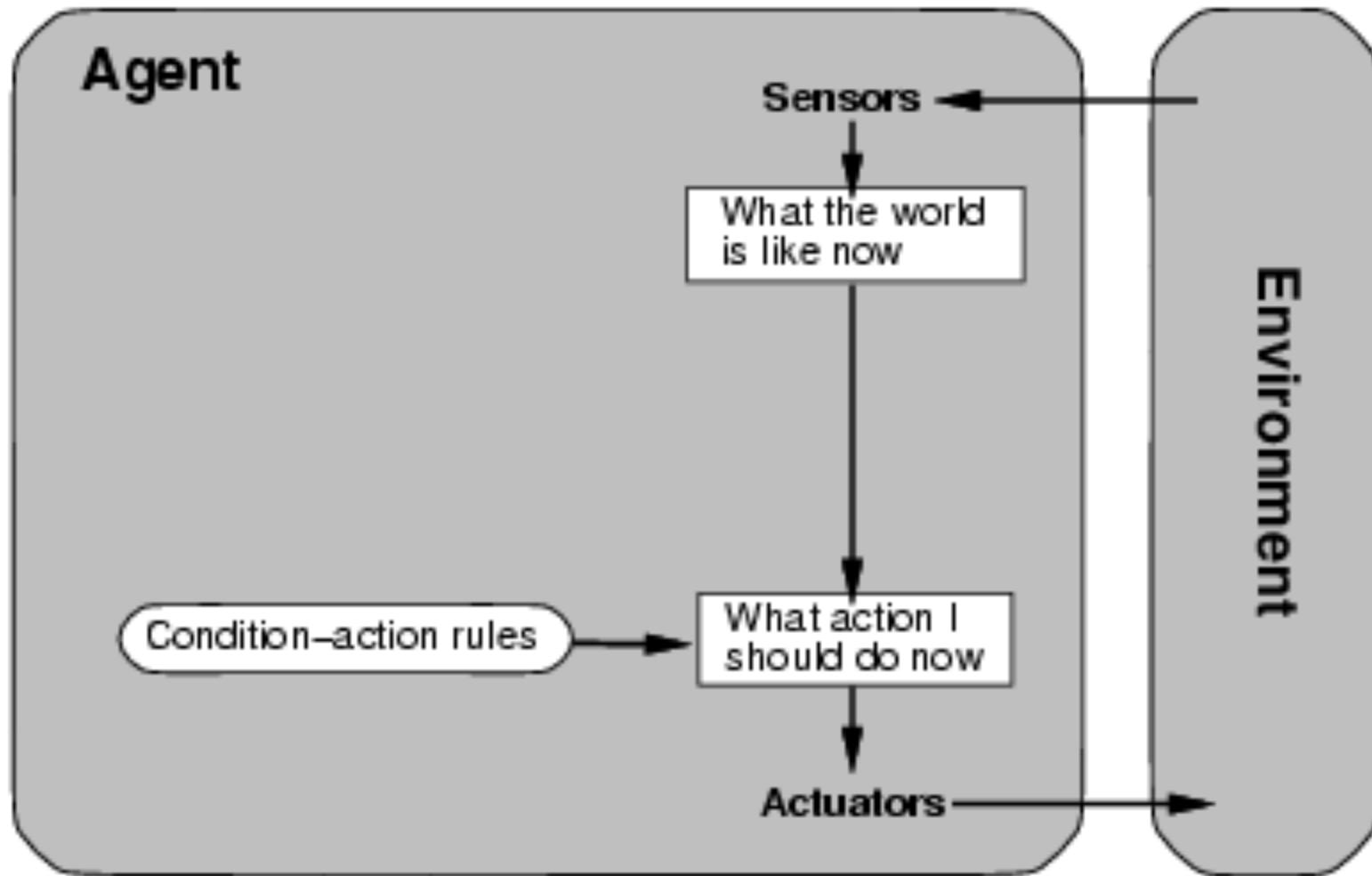
state \leftarrow INTERPRET-INPUT(*percept*)

rule \leftarrow RULE-MATCH(*state*, *rules*)

action \leftarrow RULE-ACTION[*rule*]

return *action*

Tác tử phản xạ đơn giản (2)



Tác tử phản xạ dựa trên mô hình (1)

Tác tử phản xạ dựa trên mô hình:

- ❑ Sử dụng một mô hình nội bộ để giám sát trạng thái hiện tại của môi trường
- ❑ Lựa chọn hành động: giống như đối với tác tử phản xạ đơn giản

function REFLEX-AGENT-WITH-STATE(*percept*)

static: *state* (mô tả trạng thái hiện tại của môi trường)

rules (tập các luật có dạng: điều kiện-hành động)

action (hành động gần nhất)

state \leftarrow UPDATE-STATE(*state*, *action*, *percept*)

rule \leftarrow RULE-MATCH(*state*, *rules*)

action \leftarrow RULE-ACTION[*rule*]

return *action*

Tác tử dựa trên mục tiêu (1)

- Biết về trạng thái hiện tại của môi trường: chưa đủ → Cần biết thêm thông tin về mục tiêu
 - ❑ Trạng thái hiện tại của môi trường: Ở một ngã tư, xe taxi có thể rẽ trái, rẽ phải, hoặc đi thẳng
 - ❑ Thông tin về mục tiêu: xe taxi cần đi tới đích đến của hành khách
- Tác tử dựa trên mục tiêu
 - ❑ Theo dõi trạng thái hiện tại của môi trường
 - ❑ Lưu giữ một tập các mục tiêu (cần đạt được)
 - ❑ Chọn hành động cho phép (rốt cuộc) sẽ đạt đến các mục tiêu

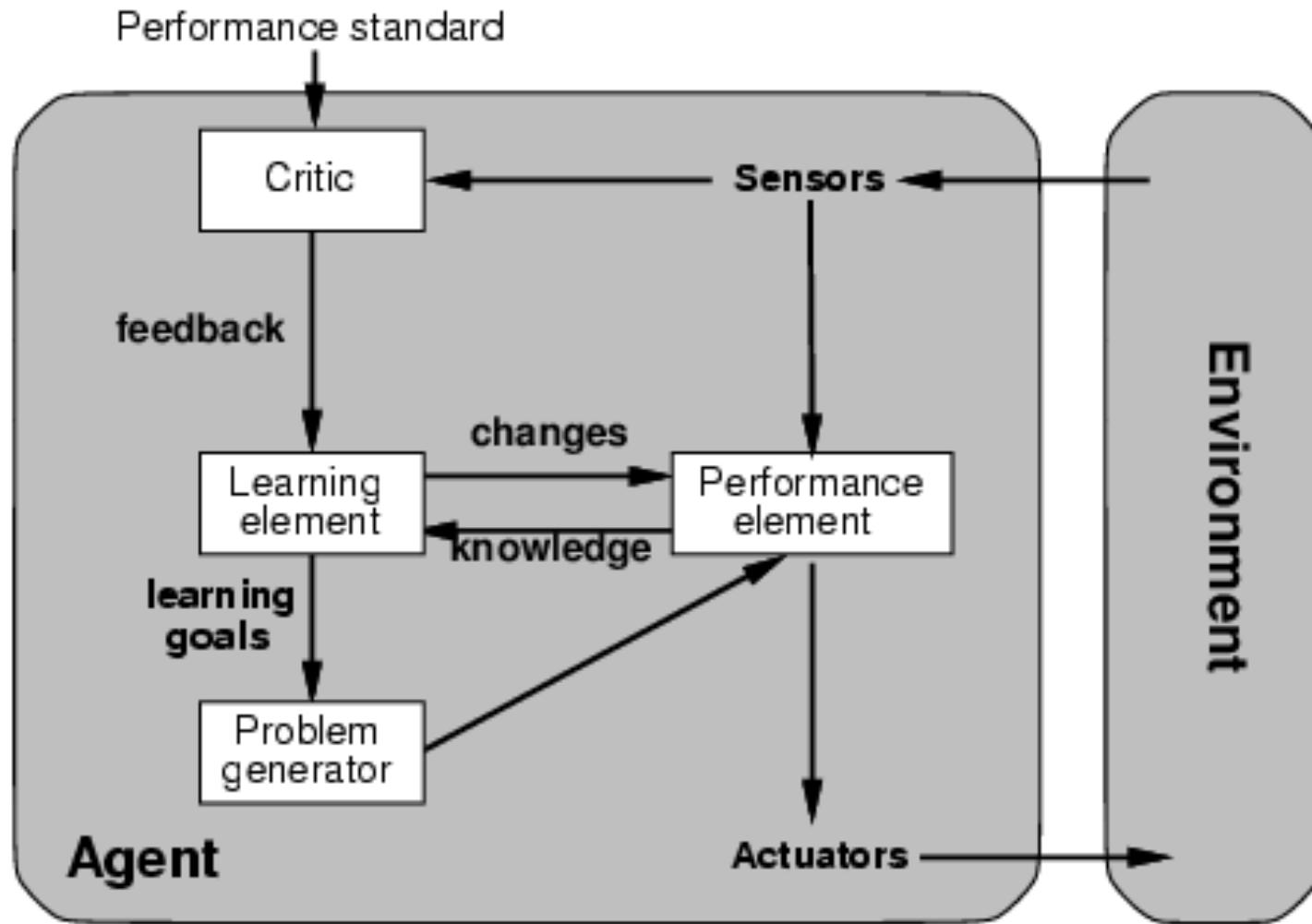
Tác tử dựa trên lợi ích (1)

- Trong nhiều môi trường, thông tin về các mục tiêu không đủ để đánh giá hiệu quả của các hành động
 - ❑ Có rất nhiều chuỗi các hành động cho phép taxi đi đến đích (tức là đạt đến mục tiêu)
 - ❑ Nhưng: chuỗi hành động nào nhanh hơn, an toàn hơn, đáng tin cậy hơn, chi phí thấp hơn?
- Cần sự đánh giá lợi ích đối với tác tử
- Hàm lợi ích (utility function)
 - ❑ Ánh xạ từ **chuỗi** các trạng thái của môi trường tới một giá trị số thực (thể hiện mức lợi ích đối với tác tử)

Tác tử có khả năng học (1)

- Khả năng học cho phép tác tử cải thiện hiệu quả hoạt động của nó
- 4 thành phần tạo nên một tác tử có khả năng học
 - Thành phần *hành động*: đảm nhiệm việc lựa chọn các hành động
 - Thành phần đánh giá (bình luận): đánh giá hiệu quả hoạt động
 - Thành phần *học*: giúp cải thiện hiệu quả hoạt động - dựa trên các đánh giá, để thay đổi (cải thiện) thành phần hành động
 - Thành phần *sản sinh kinh nghiệm*: có nhiệm vụ đề xuất các hành động giúp sản sinh ra (dẫn đến) các kinh nghiệm mới

Tác tử có khả năng học (2)



Cơ sở tri thức của tác tử

- Một cơ sở tri thức (a knowledge base) là một tập các mệnh đề (phát biểu) được biểu diễn trong một ngôn ngữ hình thức, cung cấp tri thức (hiểu biết) cho một tác tử
- Tác tử khai thác cơ sở tri thức (mà nó sở hữu) trong quá trình đưa ra các hành động
- Tác tử cần có khả năng
 - Thu thập (cập nhật) các tri thức mới
 - Cập nhật việc biểu diễn (bên trong tác tử) đối với môi trường xung quanh
 - Suy diễn ra các thuộc tính ẩn của môi trường xung quanh
 - Suy luận để đưa ra các hành động hợp lý

Đa tác tử (1)

- Môi trường hoạt động: **Cộng tác (hợp tác)** hay là **Cạnh tranh (đối kháng)**?
- Trong nhiều bài toán thực tế, môi trường hoạt động luôn thay đổi (biến động) → tác tử cần cập nhật
- Cần một mô hình biểu diễn kế hoạch của các tác tử khác
- **Các tác tử cộng tác**
 - ❑ Cùng chia sẻ các mục tiêu hoặc các kế hoạch
 - ❑ Ví dụ: Lập kế hoạch (cho hoạt động nhóm) trong trò chơi tennis đánh đôi
 - ❑ Các cơ chế cộng tác: Phân tách và phân phối các nhiệm vụ cho mỗi tác tử

Đa tác tử (2)

■ Các tác tử cạnh tranh

- ❑ Ví dụ: chơi cờ
- ❑ Mỗi tác tử phải nhận biết được sự tồn tại (và hoạt động) của các tác tử khác
- ❑ Mỗi tác tử tính toán (dự đoán) được các kế hoạch của (một số) các tác tử khác
- ❑ Mỗi tác tử tính toán (dự đoán) được ảnh hưởng của các kế hoạch của các tác tử khác đối với kế hoạch của bản thân nó
- ❑ Mỗi tác tử quyết định hành động tối ưu đối với dự đoán ảnh hưởng này

Tổng kết

- Tác tử tương tác với môi trường thông qua các bộ phận cảm biến và các bộ phận hành động
- Một tác tử hợp lý sẽ cực đại hóa hiệu quả hoạt động của nó
- Hàm tác tử mô tả các hành động mà tác tử thực hiện trong các tình huống
- Các chương trình tác tử cài đặt (thực hiện) các hàm tác tử
- Các mô tả PEAS xác định môi trường công việc
- Các môi trường được phân loại dựa theo các tiêu chí: Có thể quan sát được? Xác định? Phân đoạn? Tĩnh? Rời rạc? Tác tử đơn lẻ?
- Các kiến trúc tác tử cơ bản: Phản xạ đơn giản, Dựa trên mô hình, Dựa trên mục tiêu, Dựa trên lợi ích

Giải quyết vấn đề bằng tìm kiếm

- Giải quyết vấn đề bằng tìm kiếm
 - Tìm chuỗi các hành động cho phép đạt đến (các) trạng thái mong muốn
- Các bước chính
 - Xác định **mục tiêu** cần đạt đến (goal formulation)
 - Là một tập hợp của các trạng thái (đích)
 - Dựa trên: trạng thái hiện tại (của môi trường) và đánh giá hiệu quả hành động (của tác tử)
 - Phát biểu **bài toán** (problem formulation)
 - Với một mục tiêu, xác định các *hành động* và *trạng thái* cần xem xét
 - Quá trình **tìm kiếm** (search process)
 - Xem xét các chuỗi hành động có thể
 - Chọn chuỗi hành động tốt nhất
- Giải thuật tìm kiếm
 - Đầu vào: một bài toán (cần giải quyết)
 - Đầu ra: một giải pháp, dưới dạng một chuỗi các hành động cần thực hiện

Tác tử giải quyết vấn đề

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

    state  $\leftarrow$  UPDATE-STATE(state, percept)
    if seq is empty then do
        goal  $\leftarrow$  FORMULATE-GOAL(state)
        problem  $\leftarrow$  FORMULATE-PROBLEM(state, goal)
        seq  $\leftarrow$  SEARCH(problem)
    action  $\leftarrow$  FIRST(seq)
    seq  $\leftarrow$  REST(seq)
    return action
```

Giải quyết vấn đề bằng tìm kiếm

■ Một người du lịch đang trong chuyến đi du lịch ở Rumani

- ❑ Anh ta hiện thời đang ở Arad
- ❑ Ngày mai, anh ta có chuyến bay khởi hành từ Bucharest
- ❑ Nay, anh ta cần di chuyển (lái xe) từ Arad đến Bucharest

■ Phát biểu mục tiêu:

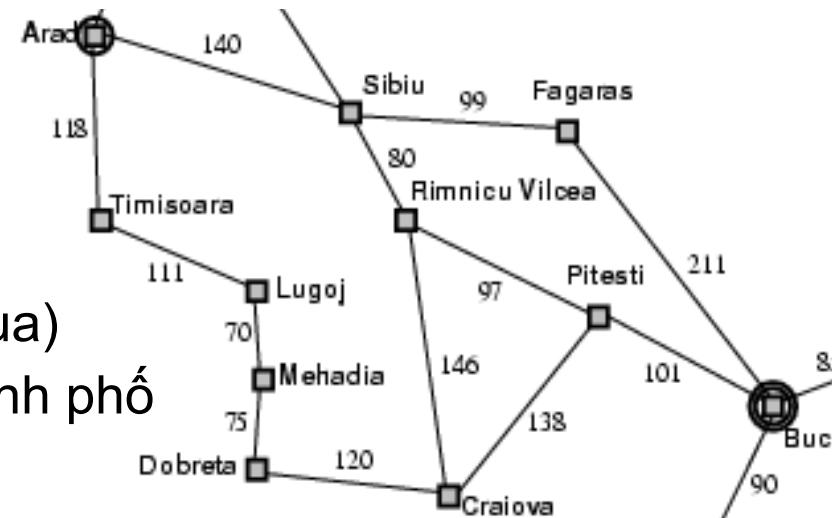
- ❑ Cần phải có mặt ở Bucharest

■ Phát biểu bài toán:

- ❑ Các *trạng thái*: các thành phố (đi qua)
- ❑ Các *hành động*: lái xe giữa các thành phố

■ Tìm kiếm giải pháp:

- ❑ Chuỗi các thành phố cần đi qua, ví dụ: Arad, Sibiu, Fagaras, Bucharest



Các kiểu bài toán

- **Xác định, có thể quan sát hoàn toàn** → Bài toán trạng thái đơn
 - ❑ Tác tử biết chính xác trạng thái tiếp theo mà nó sẽ chuyển qua
 - ❑ Giải pháp của bài toán: một chuỗi hành động
- **Không quan sát được** → Bài toán thiếu cảm nhận
 - ❑ Tác tử có thể không biết là nó đang ở trạng thái nào
 - ❑ Giải pháp của bài toán: một chuỗi hành động
- **Không xác định và/hoặc có thể quan sát một phần** → Bài toán có sự kiện ngẫu nhiên
 - ❑ Các nhận thức cung cấp các thông tin mới về trạng thái hiện tại
 - ❑ Giải pháp của bài toán: một kế hoạch (chính sách)
 - ❑ Thường kết hợp đan xen giữa: tìm kiếm và thực hiện
- **Không biết về không gian trạng thái** → Bài toán thăm dò

Phát biểu bài toán trạng thái đơn

Bài toán được định nghĩa bởi 4 thành phần:

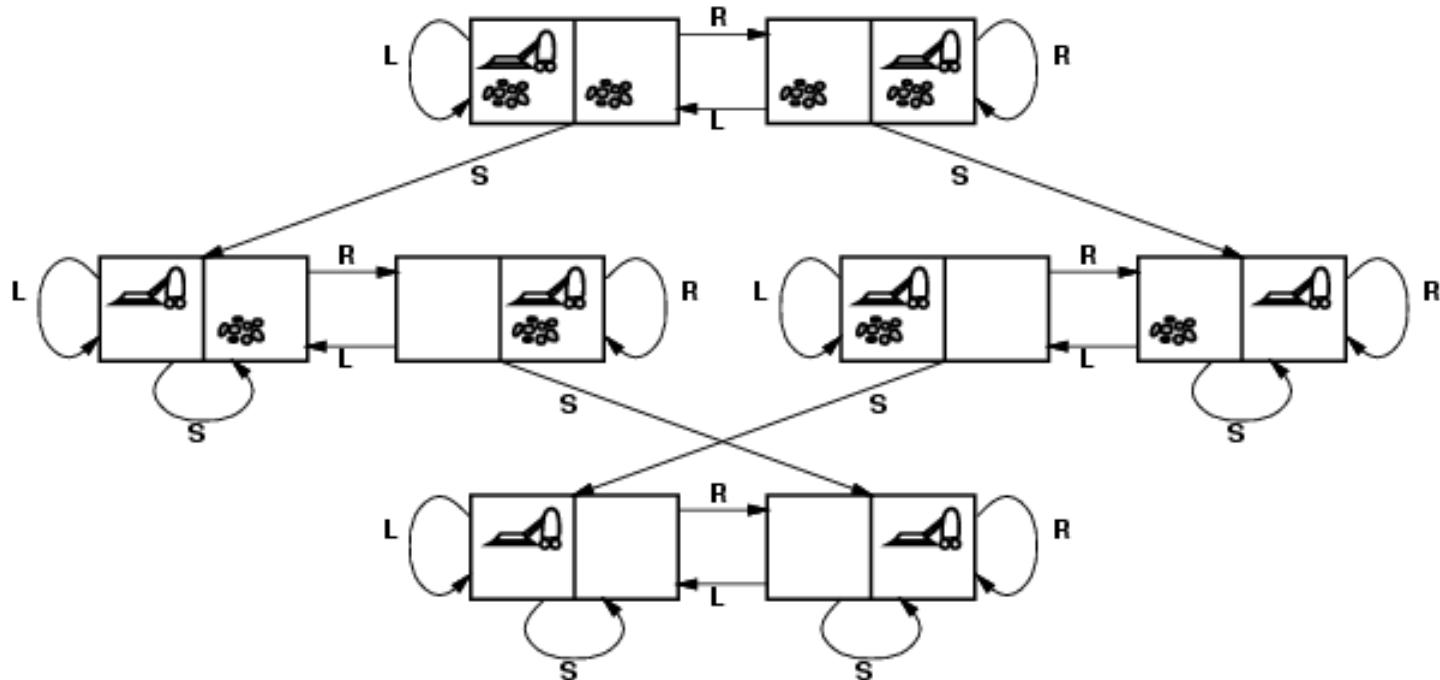
- Trạng thái đầu
 - Ví dụ: “đang ở thành phố Arad”
- Các hành động – Xác định bởi hàm chuyển trạng thái:
 $S(\text{trạng_thái_hiện_thời}) = \text{tập các cặp } <\text{hành_động}, \text{trạng_thái_tiếp_theo}>$
 - Ví dụ: $S(Arad) = \{<Arad \rightarrow Zerind, Zerind>, \dots\}$
- Kiểm tra mục tiêu, có thể là
 - Trực tiếp – ví dụ: Trạng thái hiện thời x = “Đang ở thành phố Bucharest”
 - Gián tiếp – ví dụ: $HếtCờ(x)$, $Sạch(x)$, ...
- Chi phí đường đi (giải pháp)
 - Ví dụ: Tổng các khoảng cách, Số lượng các hành động phải thực hiện,...
 - $c(x,a,y) \geq 0$ là chi phí bước (bộ phận) – chi phí cho việc áp dụng hành động a để chuyển từ trạng thái x sang trạng thái y
- Một giải pháp: Một chuỗi các hành động cho phép dẫn từ trạng thái đầu đến trạng thái đích

Xác định không gian trạng thái

- Các bài toán thực tế thường được mô tả phức tạp
 - Không gian trạng thái cần được khái quát (abstracted) để phục vụ cho việc giải quyết bài toán
- **Trạng thái** (khái quát) = Một tập các trạng thái thực tế
- **Hành động** (khái quát) = Một kết hợp phức tạp của các hành động thực tế
 - Ví dụ: Hành động "Arad → Zerind" biểu diễn một tập kết hợp các đường, đường vòng, chỗ nghỉ, ...
- Để đảm bảo việc thực hiện (quá trình tìm kiếm), bất kỳ trạng thái thực tế nào cũng phải có thể đạt đến được từ trạng thái thực tế khác
- **Giải pháp** (khái quát) = Một tập các đường đi giải pháp trong thực tế

Đồ thị không gian trạng thái (2)

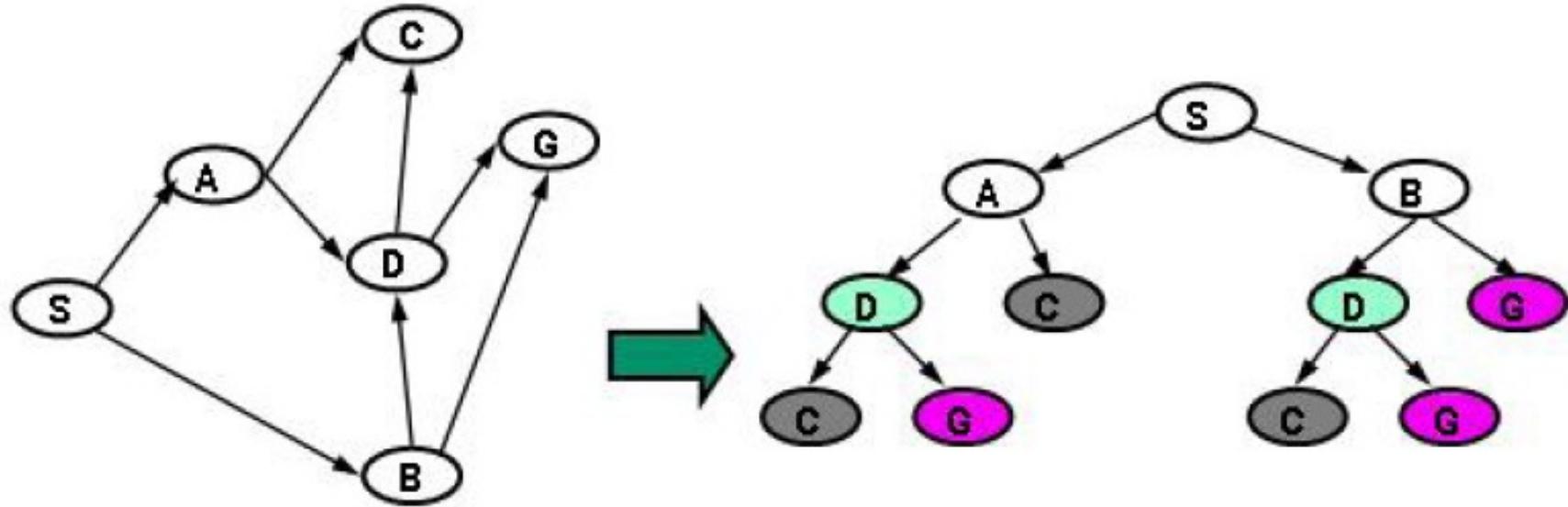
Bài toán máy hút bụi



- Các trạng thái?
- Các hành động?
- Kiểm tra mục tiêu?
- Chi phí đường đi?

*Chỗ bẩn và vị trí máy hút bụi
Sang trái, sang phải, hút bụi, không làm gì
Không còn chỗ (vị trí) nào bẩn
1 (mỗi hành động), 0 (không làm gì cả)*

Đồ thị tìm kiếm → Cây tìm kiếm



- Các bài toán tìm kiếm trên đồ thị có thể được chuyển thành các bài toán tìm kiếm trên cây
 - Thay thế mỗi liên kết (cạnh) vô hướng bằng 2 liên kết (cạnh) có hướng
 - Loại bỏ các vòng lặp tồn tại trong đồ thị (để tránh không duyệt 2 lần đối với một nút trong bất kỳ đường đi nào)

Các chiến lược tìm kiếm cơ bản

- Các chiến lược tìm kiếm cơ bản (uninformed search strategies) **chỉ sử dụng các thông tin chưa trong định nghĩa của bài toán**
 - Tìm kiếm theo chiều rộng (Breadth-first search)
 - Tìm kiếm với chi phí cực tiểu (Uniform-cost search)
 - Tìm kiếm theo chiều sâu (Depth-first search)
 - Tìm kiếm giới hạn độ sâu (Depth-limited search)
 - Tìm kiếm sâu dần (Iterative deepening search)

Tìm kiếm theo chiều rộng – BFS

- Phát triển các nút chưa xét theo chiều rộng – Các nút được xét theo thứ tự độ sâu tăng dần
- Cài đặt giải thuật BFS
 - *fringe* là một cấu trúc kiểu hàng đợi FIFO (các nút mới được bổ sung vào cuối của *fringe*)
- Các ký hiệu được sử dụng trong giải thuật BFS
 - *fringe*: Cấu trúc kiểu hàng đợi (queue) lưu giữ các nút (trạng thái) **sẽ** được duyệt
 - *closed*: Cấu trúc kiểu hàng đợi (queue) lưu giữ các nút (trạng thái) **đã** được duyệt
 - $G=(N,A)$: Cây biểu diễn không gian trạng thái của bài toán
 - n_0 : Trạng thái đầu của bài toán (nút gốc của cây)
 - \mathcal{DICH} : Tập các trạng thái đích của bài toán
 - $\Gamma(n)$: Tập các trạng thái (nút) con của trạng thái (nút) đang xét n

BFS – Giải thuật

BFS (N, A, n_0 , ĐICH)

{

fringe $\leftarrow n_0$;

closed $\leftarrow \emptyset$;

while ($\text{fringe} \neq \emptyset$) do

{ n $\leftarrow \text{GET_FIRST}(\text{fringe})$; // lấy phần tử đầu tiên của fringe

 closed $\leftarrow \text{closed} \oplus n$;

 if ($n \in \text{ĐICH}$) then return $\text{SOLUTION}(n)$;

 if ($\Gamma(n) \neq \emptyset$) then $\text{fringe} \leftarrow \text{fringe} \oplus \Gamma(n)$;

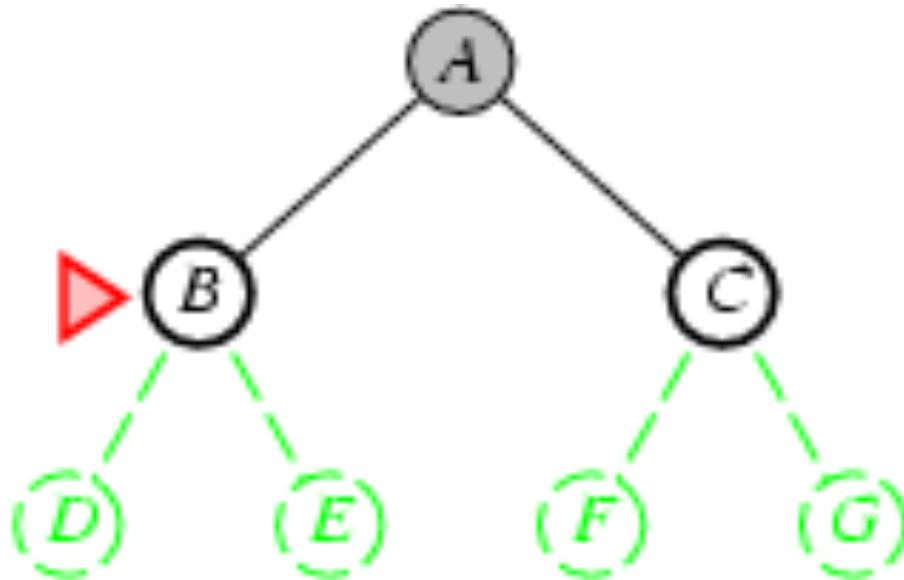
}

return (“No solution”);

}

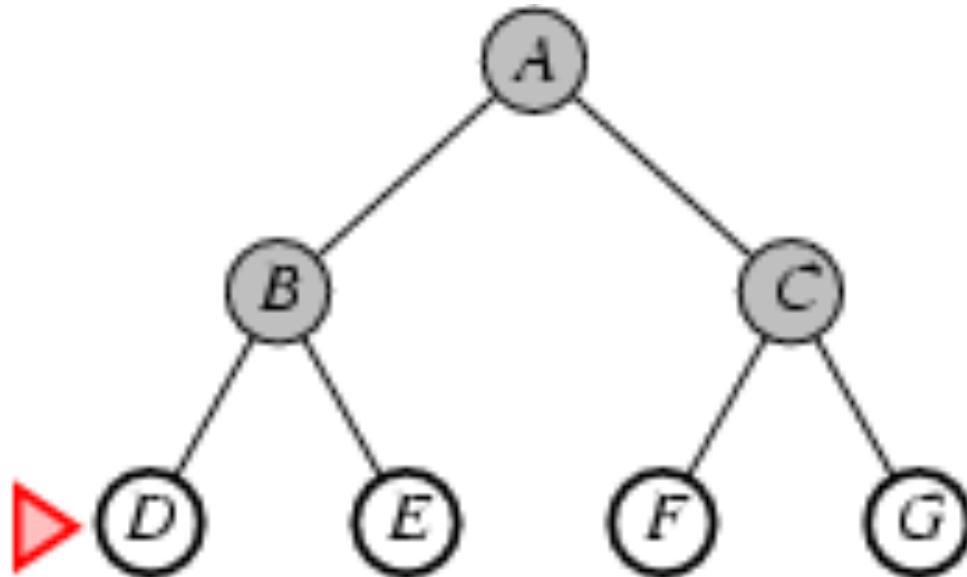
BFS – Ví dụ (2)

- Phát triển các nút chưa xét theo chiều rộng – Các nút được xét theo thứ tự độ sâu tăng dần



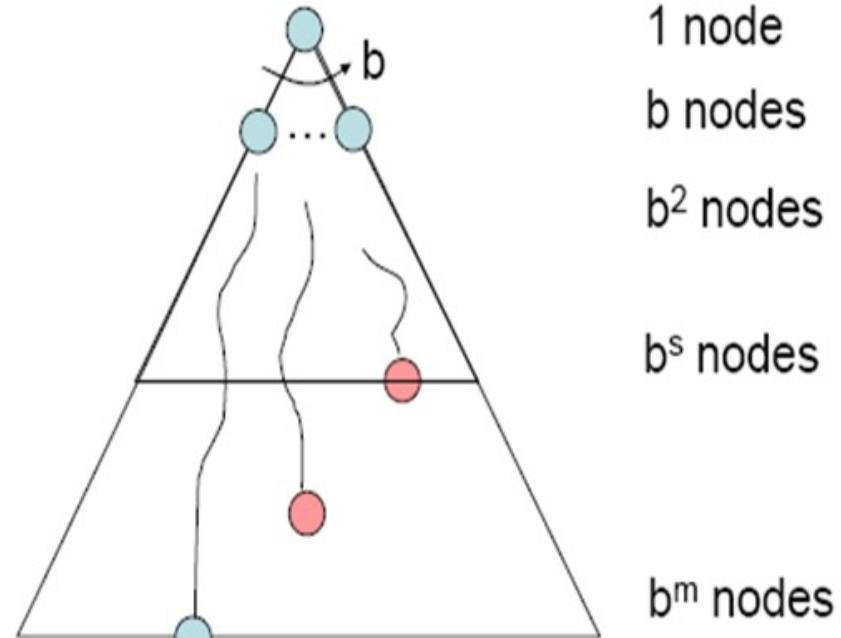
BFS – Ví dụ (4)

- Phát triển các nút chưa xét theo chiều rộng – Các nút được xét theo thứ tự độ sâu tăng dần



BFS – Các đặc điểm

- Tính hoàn chỉnh?
 - Có (nếu b là hữu hạn)
- Độ phức tạp về thời gian?
 - $1+b+b^2+b^3+\dots+b^d = O(b^{d+1})$
- Độ phức tạp về bộ nhớ?
 - $O(b^{d+1})$ – Lưu tất cả các nút trong bộ nhớ)
- Tính tối ưu?
 - Có (nếu chi phí =1 cho mỗi bước)



Tìm kiếm với chi phí cực tiểu (UCS)

- Phát triển các nút chưa xét có chi phí thấp nhất – Các nút được xét theo thứ tự chi phí (từ nút gốc đến nút đang xét) tăng dần
- Cài đặt:
 - *fringe* là một cấu trúc hàng đợi, trong đó các phần tử được sắp xếp theo chi phí đường đi
- Trở thành phương pháp tìm kiếm theo chiều rộng, nếu các chi phí ở mỗi bước (mỗi cạnh của cây tìm kiếm) là như nhau

UCS: Các đặc điểm

- Tính hoàn chỉnh?
 - Có (nếu chi phí ở mỗi bước $\geq \varepsilon > 0$)
- Độ phức tạp về thời gian?
 - Phụ thuộc vào tổng số các nút có chi phí \leq chi phí của lời giải tối ưu: $O(b^{\lceil C^*/\varepsilon \rceil})$, trong đó C^* là chi phí của lời giải tối ưu
- Độ phức tạp về bộ nhớ?
 - Phụ thuộc vào tổng số các nút có chi phí \leq chi phí của lời giải tối ưu: $O(b^{\lceil C^*/\varepsilon \rceil})$
- Tính tối ưu?
 - Có, nếu các nút được xét theo thứ tự tăng dần về chi phí $g(n)$

Tìm kiếm theo chiều sâu (DFS)

- Phát triển các nút chưa xét theo chiều sâu – Các nút được xét theo thứ tự độ sâu giảm dần
- Cài đặt:
 - *fringe* là một cấu trúc kiểu ngăn xếp LIFO (Các nút mới được bổ sung vào đầu của *fringe*)

DFS: Giải thuật

DFS (N, A, n_0 , ĐICH)

{

fringe $\leftarrow n_0$;

closed $\leftarrow \emptyset$;

while ($\text{fringe} \neq \emptyset$) do

{ n $\leftarrow \text{GET_FIRST}(\text{fringe})$; // lấy phần tử đầu tiên của fringe

 closed $\leftarrow \text{closed} \oplus n$;

 if ($n \in \text{ĐICH}$) then return $\text{SOLUTION}(n)$;

 if ($\Gamma(n) \neq \emptyset$) then $\text{fringe} \leftarrow \Gamma(n) \oplus \text{fringe}$;

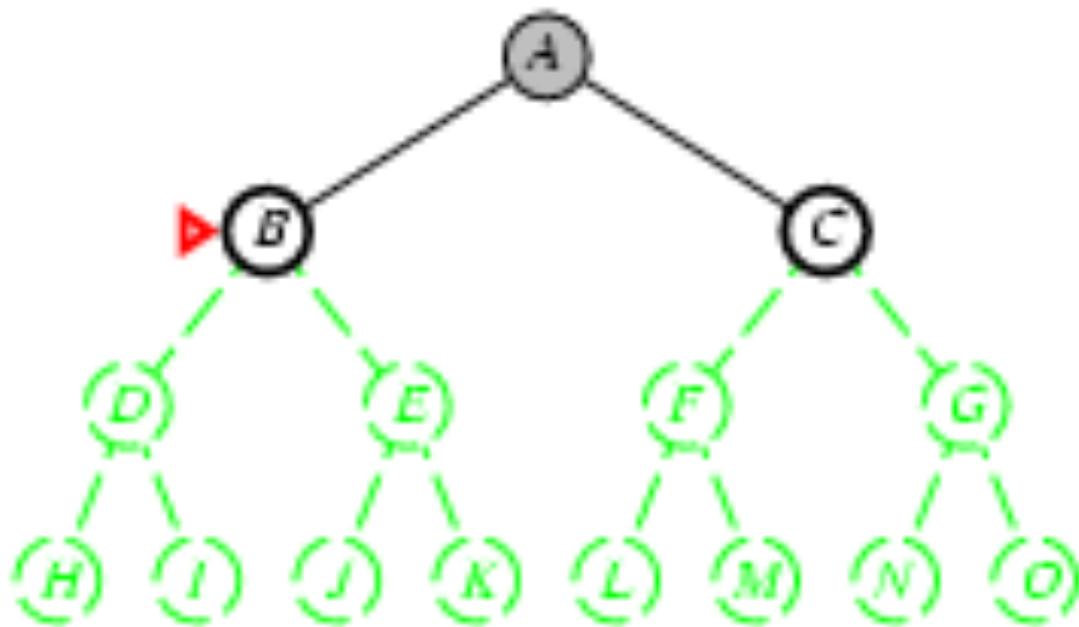
}

return (“No solution”);

}

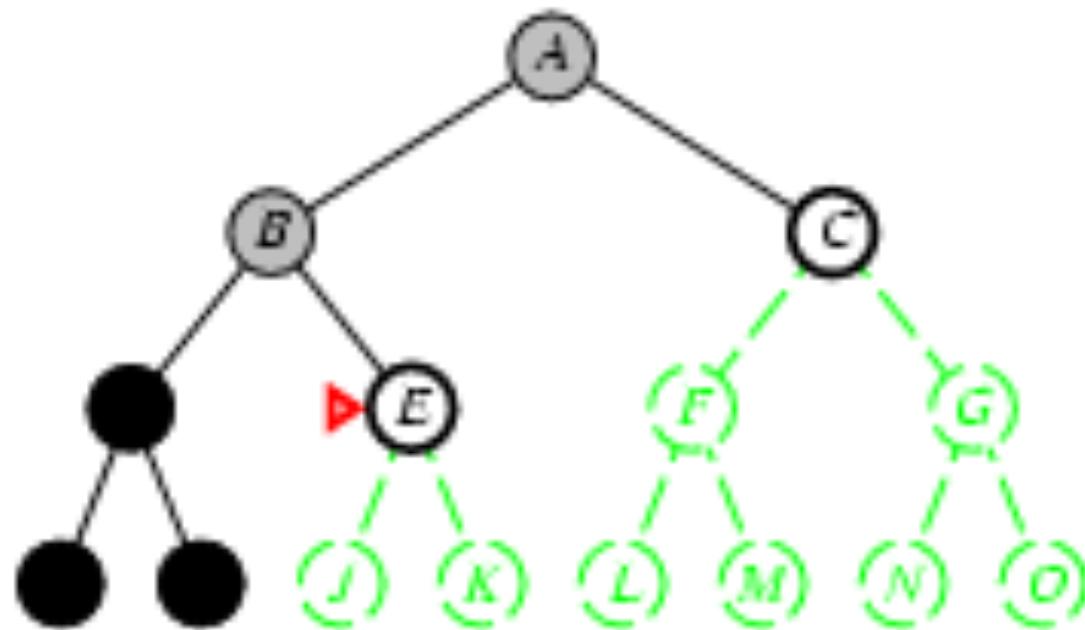
DFS: Ví dụ (2)

- Phát triển các nút chưa xét theo chiều sâu – Các nút được xét theo thứ tự độ sâu giảm dần



DFS: Ví dụ (6)

- Phát triển các nút chưa xét theo chiều sâu – Các nút được xét theo thứ tự độ sâu giảm dần



DFS: Các đặc điểm

- Tính hoàn chỉnh?
 - Không – Thất bại (không tìm được lời giải) nếu không gian trạng thái có độ sâu vô hạn, hoặc nếu không gian trạng thái chứa các vòng lặp giữa các trạng thái
 - Đề cử: Sửa đổi để tránh việc một trạng thái nào đó bị lặp lại (bị xét lại) theo một đường đi tìm kiếm
 - → Đạt tính hoàn chỉnh đối với không gian trạng thái hữu hạn
- Độ phức tạp về thời gian?
 - $O(b^m)$: rất lớn, nếu m lớn hơn nhiều so với d
- Độ phức tạp về bộ nhớ?
 - $O(bm)$ - độ phức tạp tuyến tính
- Tính tối ưu?
 - Không

Tìm kiếm giới hạn độ sâu (DLS)

= Phương pháp tìm kiếm theo chiều sâu (DFS) + Sử dụng giới hạn về độ sâu / trong quá trình tìm kiếm
→ các nút ở độ sâu / không có nút con

```
function DEPTH-LIMITED-SEARCH( problem, limit) returns soln/fail/cutoff
    RECURSIVE-DLS(MAKE-NODE(INITIAL-STATE[problem]), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns soln/fail/cutoff
    cutoff-occurred? ← false
    if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
    else if DEPTH[node] = limit then return cutoff
    else for each successor in EXPAND(node, problem) do
        result ← RECURSIVE-DLS(successor, problem, limit)
        if result = cutoff then cutoff-occurred? ← true
        else if result ≠ failure then return result
    if cutoff-occurred? then return cutoff else return failure
```

Tìm kiếm sâu dần (IDS)

- Vấn đề với giải thuật tìm kiếm với giới hạn độ sâu (DLS)
 - Nếu tất cả các lời giải (các nút đích) nằm ở độ sâu lớn hơn giới hạn độ sâu l , thì giải thuật DLS thất bại (không tìm được lời giải)
- Giải thuật tìm kiếm sâu dần
 - Áp dụng giải thuật DFS đối với các đường đi (trong cây) có độ dài ≤ 1
 - Nếu thất bại (không tìm được lời giải), tiếp tục áp dụng giải thuật DFS đối với các đường đi có độ dài ≤ 2
 - Nếu thất bại (không tìm được lời giải), tiếp tục áp dụng giải thuật DFS đối với các đường đi có độ dài ≤ 3
 - ... (tiếp tục như trên, cho đến khi: 1) tìm được lời giải, hoặc 2) toàn bộ cây đã được xét mà không tìm được lời giải)

DLS vs. IDS

- Với độ sâu d và hệ số phân nhánh b , thì số lượng các nút được sinh ra trong giải thuật tìm kiếm giới hạn độ sâu là:

$$N_{DLS} = b^0 + b^1 + b^2 + \dots + b^{d-2} + b^{d-1} + b^d$$

- Số lượng các nút được sinh ra trong giải thuật tìm kiếm sâu dần là:

$$N_{IDS} = (d+1).b^0 + d.b^1 + (d-1).b^2 + \dots + 3.b^{d-2} + 2.b^{d-1} + 1.b^d$$

- Ví dụ với $b = 10$, $d = 5$:

- $N_{DLS} = 1 + 10 + 100 + 1,000 + 10,000 + 100,000 = 111,111$
 - $N_{IDS} = 6 + 50 + 400 + 3,000 + 20,000 + 100,000 = 123,456$
 - Lãng phí = $(123,456 - 111,111)/111,111 = 11\%$

IDS: Các đặc điểm

- Tính hoàn chỉnh?
 - Có
- Độ phức tạp về thời gian?
 - $(d+1)b^0 + d b^1 + (d-1)b^2 + \dots + b^d = O(b^{d+1})$
- Độ phức tạp về bộ nhớ?
 - $O(bd)$
- Tính tối ưu?
 - Có, nếu chi phí cho mỗi bước (mỗi cạnh của cây tìm kiếm) = 1

So sánh giữa các giải thuật tìm kiếm cơ bản

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^{d+1})$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes (some cases)	Yes (some cases)	No	No	Yes (some cases)

Tổng kết

- Việc phát biểu bài toán thường yêu cầu việc khái quát hóa các chi tiết của bài toán thực tế, để có thể định nghĩa không gian trạng thái sao cho việc xét (khám phá) các trạng thái trong quá trình tìm kiếm được thuận tiện
- Có nhiều chiến lược tìm kiếm cơ bản
 - ❑ Tìm kiếm theo chiều rộng (BFS)
 - ❑ Tìm kiếm theo chiều sâu (DFS)
 - ❑ Tìm kiếm với chi phí cực tiểu (UCS)
 - ❑ Tìm kiếm giới hạn độ sâu (DLS)
 - ❑ Tìm kiếm sâu dần (IDS)
- Phương pháp tìm kiếm sâu dần (IDS)
 - ❑ Chi phí về bộ nhớ ở mức hàm tuyến tính
 - ❑ Chi phí về thời gian chỉ nhiều hơn một chút so với các phương pháp tìm kiếm cơ bản khác

Nhắc lại: Tìm kiếm theo cấu trúc cây

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if GOAL-TEST[problem] applied to STATE(node) succeeds return node
    fringe  $\leftarrow$  INSERTALL(EXPAND(node, problem), fringe)
```

- Một chiến lược (phương pháp) tìm kiếm = Một cách xác định thứ tự xét các nút của cây

Tìm kiếm với tri thức bổ sung

- Các chiến lược tìm kiếm cơ bản (uninformed search strategies) chỉ sử dụng các thông tin chưa trong định nghĩa của bài toán
 - Không phù hợp với nhiều bài toán thực tế (do đòi hỏi chi phí quá cao về thời gian và bộ nhớ)
- Các chiến lược tìm kiếm với tri thức bổ sung (informed search strategies) sử dụng các *tri thức cụ thể* của bài toán → Quá trình tìm kiếm hiệu quả hơn
 - Các giải thuật tìm kiếm best-first (Greedy best-first, A*)
 - Các giải thuật tìm kiếm cục bộ (Hill-climbing, Simulated annealing, Local beam, Genetic algorithms)
 - Các giải thuật tìm kiếm đối kháng (MiniMax, Alpha-beta pruning)

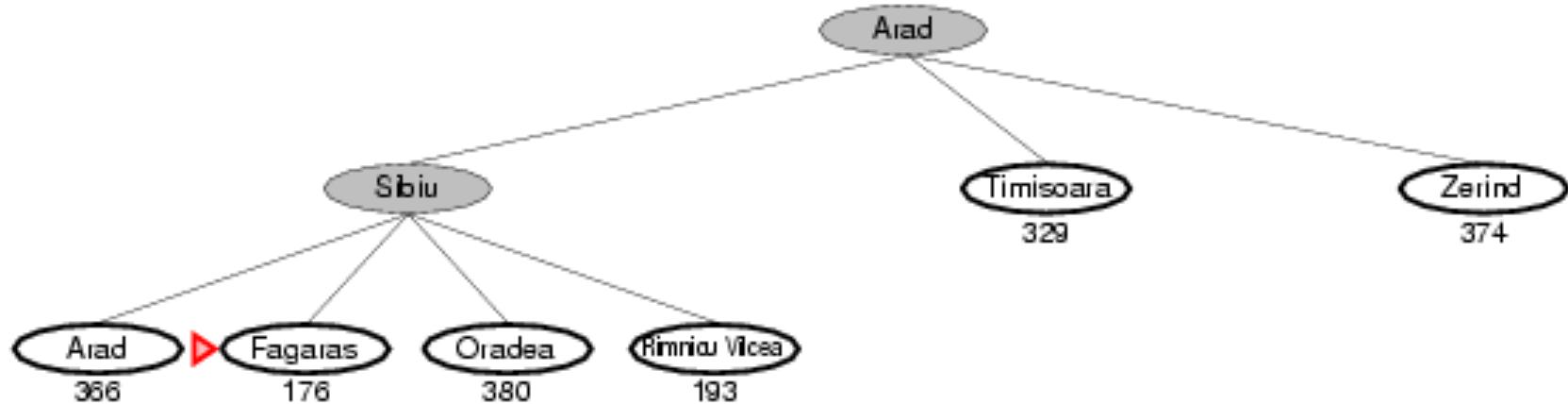
Best-first search

- **Ý tưởng:** Sử dụng một *hàm đánh giá* $f(n)$ cho mỗi nút của cây tìm kiếm
 - Để đánh giá mức độ “phù hợp” của nút đó
→ Trong quá trình tìm kiếm, ưu tiên xét các nút có mức độ phù hợp cao nhất
- Cài đặt giải thuật
 - Sắp thứ tự các nút trong cấu trúc fringe theo trật tự giảm dần về mức độ phù hợp
- Các trường hợp đặc biệt của giải thuật Best-first search
 - Greedy best-first search
 - A* search

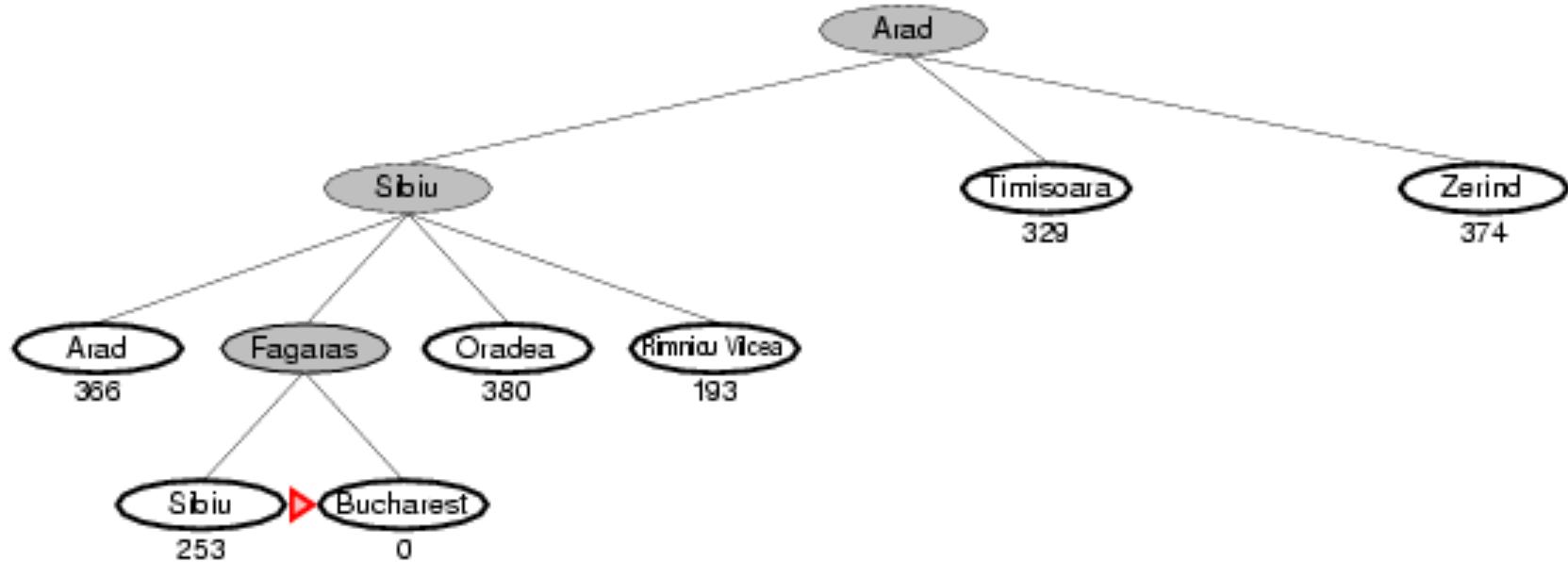
Greedy best-first search

- Hàm đánh giá $f(n)$ là hàm *heuristic* $h(n)$
- Hàm heuristic $h(n)$ đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu)
- Ví dụ: Trong bài toán tìm đường đi từ Arad đến Bucharest, sử dụng: $h_{SLD}(n) = \text{Ước lượng khoảng cách đường thẳng ("chim bay") từ thành phố hiện tại } n \text{ đến Bucharest}$
- Phương pháp tìm kiếm Greedy best-first search sẽ xét (phát triển) nút “có vẻ” gần với nút đích (mục tiêu) nhất

Greedy best-first search – Ví dụ (4)



Greedy best-first search – Ví dụ (5)



Greedy best-first search – Các đặc điểm

■ Tính hoàn chỉnh?

- ❑ Không – Vì có thể vướng (chết tắc) trong các vòng lặp kiểu như:
Iasi → Neamt → Iasi → Neamt →...

■ Độ phức tạp về thời gian?

- ❑ $O(b^m)$
- ❑ Một hàm heuristic tốt có thể mang lại cải thiện lớn

■ Độ phức tạp về bộ nhớ?

- ❑ $O(b^m)$ – Lưu giữ tất cả các nút trong bộ nhớ

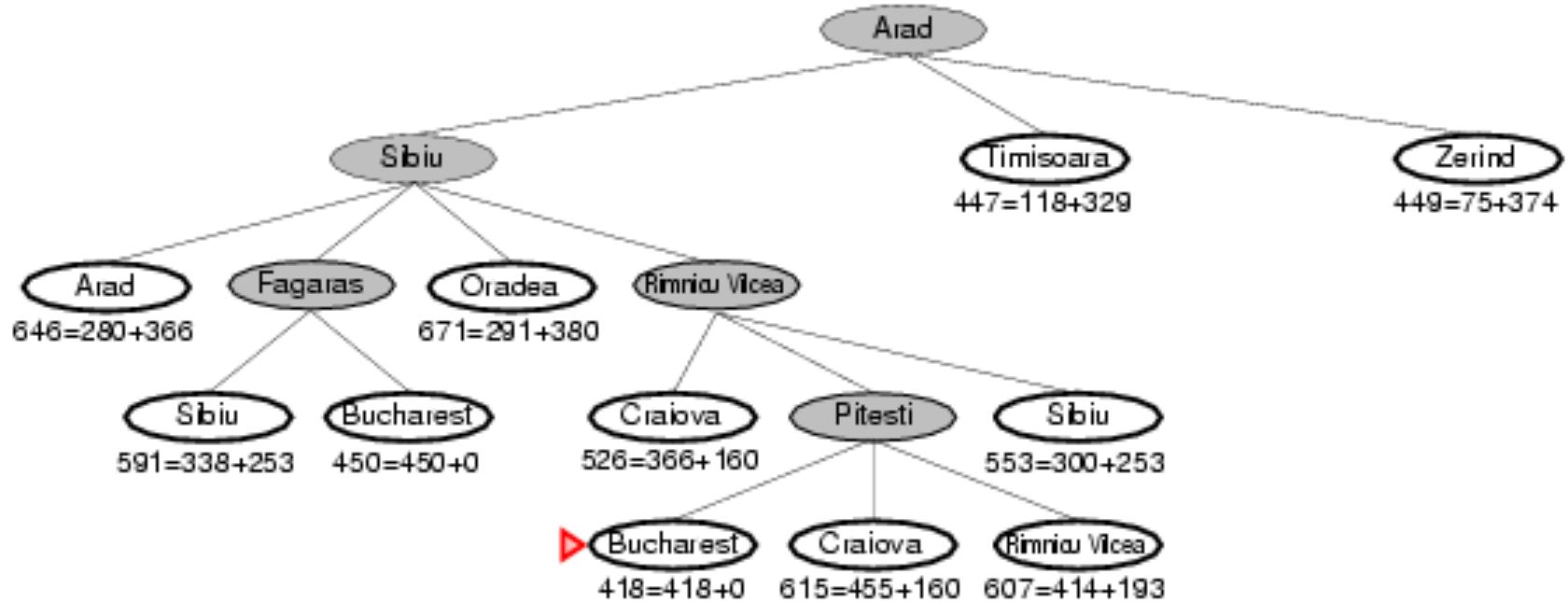
■ Tính tối ưu?

- ❑ Không

A* search

- **Ý tưởng:** Tránh việc xét (phát triển) các nhánh tìm kiếm đã xác định (cho đến thời điểm hiện tại) là có chi phí cao
- Sử dụng hàm đánh giá $f(n) = g(n) + h(n)$
 - $g(n)$ = chi phí từ nút gốc cho đến nút hiện tại n
 - $h(n)$ = chi phí ước lượng từ nút hiện tại n tới đích
 - $f(n)$ = chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích

A* search – Ví dụ (6)



A* search: các đặc điểm

- Nếu không gian các trạng thái là hữu hạn và có giải pháp để tránh việc xét (lặp) lại các trạng thái, thì giải thuật A* là hoàn chỉnh (tìm được lời giải) – nhưng không đảm bảo là tối ưu
- Nếu không gian các trạng thái là hữu hạn và không có giải pháp để tránh việc xét (lặp) lại các trạng thái, thì giải thuật A* là không hoàn chỉnh
- Nếu không gian các trạng thái là vô hạn, thì giải thuật A* là không hoàn chỉnh
- Khi nào thì A* tối ưu?

Các ước lượng chấp nhận được

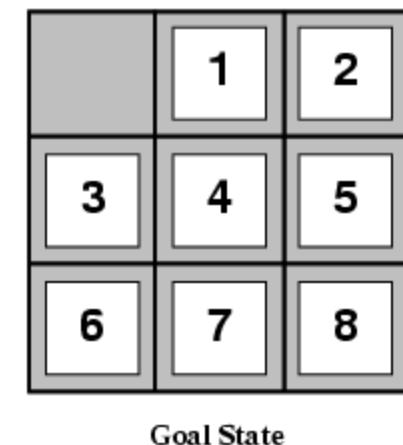
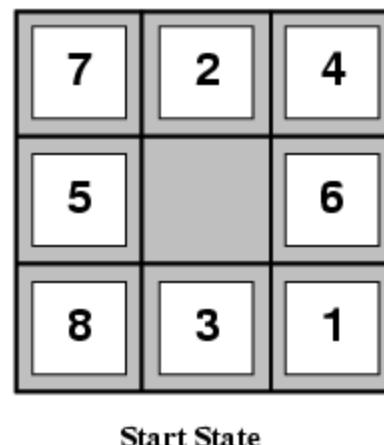
- Một ước lượng $h(n)$ được xem là chấp nhận được nếu đối với mọi nút n : $0 \leq h(n) \leq h^*(n)$, trong đó $h^*(n)$ là chi phí thật (thực tế) để đi từ nút n đến đích
- Một ước lượng chấp nhận được không bao giờ đánh giá quá cao (overestimate) đối với chi phí để đi tới đích
 - Thực chất, ước lượng chấp nhận được có xu hướng đánh giá “lạc quan”
- Ví dụ: Ước lượng $h_{SLD}(n)$ đánh giá thấp hơn khoảng cách đường đi thực tế
- **Định lý:** Nếu $h(n)$ là đánh giá chấp nhận được, thì phương pháp tìm kiếm A* sử dụng giải thuật TREE-SEARCH là tối ưu

Các ước lượng chấp nhận được (2)

Ví dụ đối với trò chơi ô chữ 8 số:

- $h_1(n) =$ số các ô chữ nằm ở sai vị trí (so với vị trí của ô chữ đấy ở trạng thái đích)
- $h_2(n) =$ khoảng cách dịch chuyển ($\leftarrow, \rightarrow, \uparrow, \downarrow$) ngắn nhất để dịch chuyển các ô chữ nằm sai vị trí về vị trí đúng

- $h_1(S) = 8$
- $h_2(S) = 3+1+$
 $2+2+$
 $2+3+$
 $3+2 = 18$



Ước lượng ưu thế

- Ước lượng h_2 được gọi là **ưu thế hơn / trội hơn** (dominate) ước lượng h_1 nếu:
 - $h^*(n) \geq h_2(n) \geq h_1(n)$ đối với tất cả các nút n
- Nếu ước lượng h_2 ưu thế hơn ước lượng h_1 , thì h_2 tốt hơn (nên được sử dụng hơn) cho quá trình tìm kiếm
- Trong ví dụ (ô chữ 8 số) ở trên: Chi phí tìm kiếm = Số lượng trung bình của các nút phải xét:
 - Với độ sâu $d = 12$
 - IDS (Tìm kiếm sâu dần): 3.644.035 nút phải xét
 - A*(sử dụng ước lượng h_1): 227 nút phải xét
 - A*(sử dụng ước lượng h_2): 73 nút phải xét
 - Với độ sâu $d = 24$
 - IDS (Tìm kiếm sâu dần): Quá nhiều nút phải xét
 - A*(sử dụng ước lượng h_1): 39.135 nút phải xét
 - A*(sử dụng ước lượng h_2): 1.641 nút phải xét

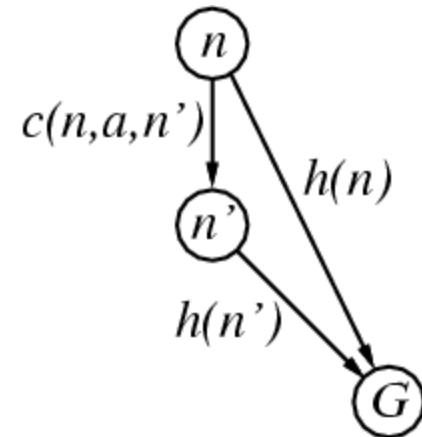
Các ước lượng kiên định

- Một ước lượng h được xem là **kiên định (consistent)**, nếu với mọi nút n và với mọi nút tiếp theo n' của n (được sinh ra bởi hành động a):

$$h(n) \leq c(n,a,n') + h(n')$$

- Nếu ước lượng h là kiên định, ta có:

$$\begin{aligned}f(n') &= g(n') + h(n') \\&= g(n) + c(n,a,n') + h(n') \\&\geq g(n) + h(n) = f(n)\end{aligned}$$



Nghĩa là: $f(n)$ không giảm trong bất kỳ đường đi (tìm kiếm) nào đi qua n

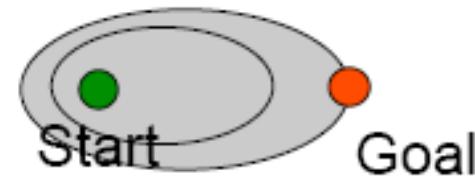
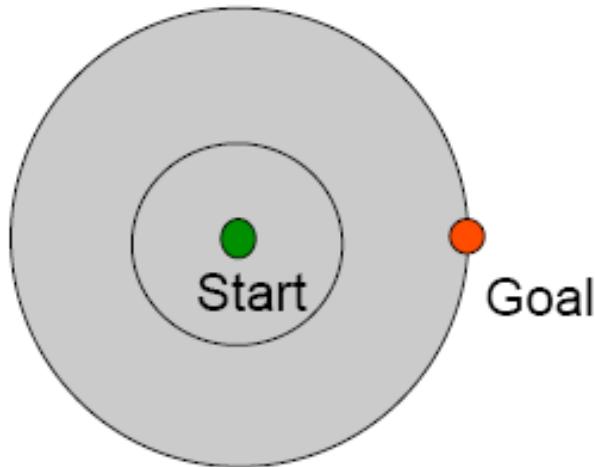
- Định lý:** Nếu $h(n)$ là kiên định, thì phương pháp tìm kiếm A* sử dụng giải thuật GRAPH-SEARCH là tối ưu

Các đặc điểm của A*

- Tính hoàn chỉnh?
 - Có (trừ khi có rất nhiều các nút có chi phí $f \leq f(G)$)
- Độ phức tạp về thời gian?
 - Độ của hàm mũ – Số lượng các nút được xét là hàm mũ của độ dài đường đi của lời giải
- Độ phức tạp về bộ nhớ?
 - Lưu giữ tất cả các nút trong bộ nhớ
- Tính tối ưu?
 - Có (đối với điều kiện đặc biệt)

A* vs. UCS

- Tìm kiếm với chi phí cực tiểu (UCS) phát triển theo mọi hướng
- Tìm kiếm A* phát triển chủ yếu theo hướng tới đích, nhưng đảm bảo tính tối ưu

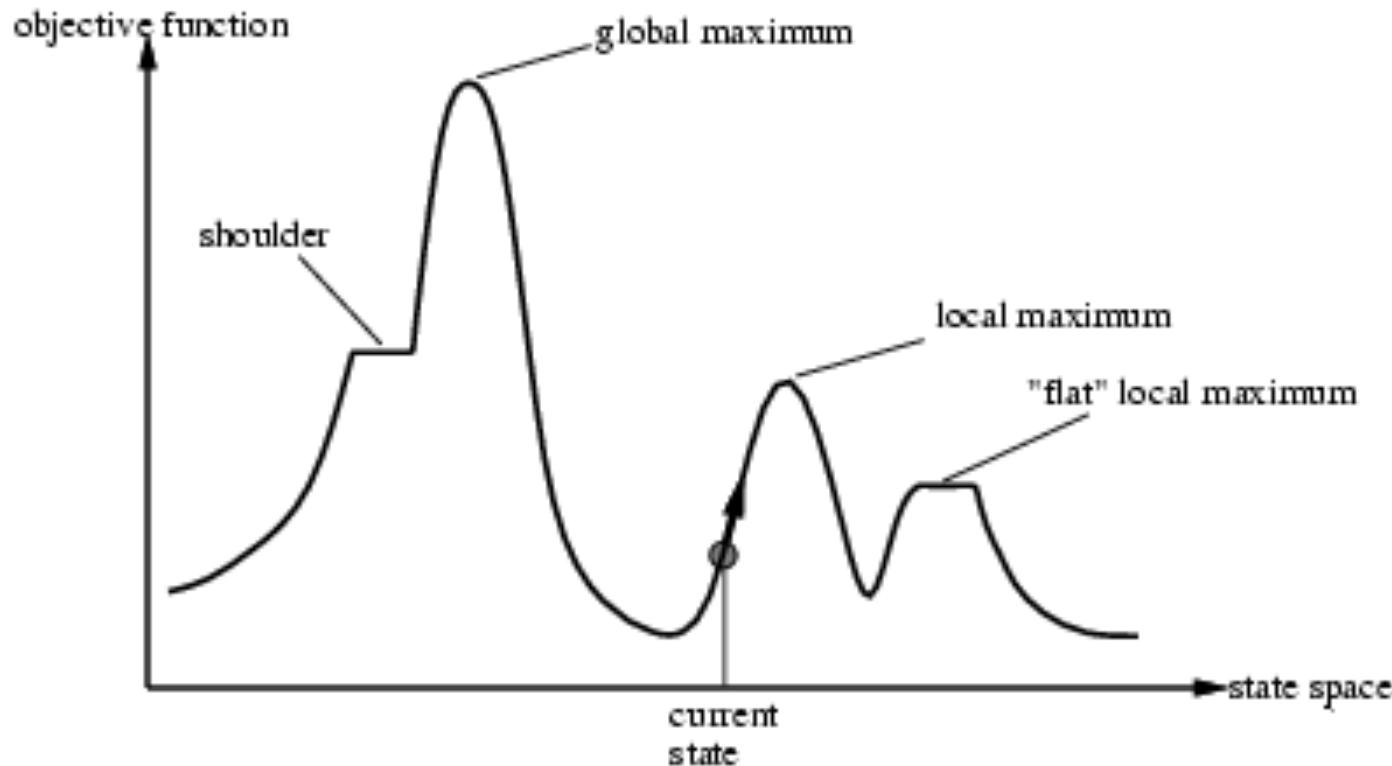


Các giải thuật tìm kiếm cục bộ

- Trong nhiều bài toán tối ưu, các đặc điểm thường phức tạp và ta không thể tìm được lời giải tối ưu.
 - Trạng thái đích = Lời giải của bài toán phải thoả mãn ràng buộc nào đó.
 - Ví dụ: Bài toán n quân hậu (bố trí n quân hậu trên một bàn cờ kích thước $n \times n$, sao cho các quân hậu không ăn nhau)
 - Bài toán tối ưu có nhiều điểm cực trị địa phương.
- Trong những bài toán như thế, chúng ta có thể sử dụng các giải thuật tìm kiếm cục bộ
- Tại mỗi thời điểm, chỉ lưu một trạng thái "hiện thời" duy nhất. Mục tiêu: cố gắng "cải thiện" trạng thái (cấu hình) hiện thời này đối với một tiêu chí nào đó (định trước)

Tìm kiếm leo đồi: minh họa

- Nhược điểm: Tùy vào trạng thái đầu, giải thuật tìm kiếm leo đồi có thể “tắc” ở các điểm cực đại cục bộ (local maxima)
 - Không tìm được lời giải tối ưu toàn cục (global optimal solution)



Simulated annealing search

- Dựa trên quá trình tinh luyện (annealing process): Kim loại nguội đi và lạnh cứng lại thành cấu trúc kết tinh
- Phương pháp tìm kiếm Simulated Annealing có thể tránh được các điểm tối ưu cục bộ (local optima)
- Phương pháp tìm kiếm Simulated Annealing sử dụng chiến lược tìm kiếm ngẫu nhiên, trong đó chấp nhận các thay đổi làm tăng giá trị hàm mục tiêu (i.e., cần tối ưu) và *cũng chấp nhận (có hạn chế) các thay đổi làm giảm*
- Phương pháp tìm kiếm Simulated Annealing sử dụng một tham số điều khiển T (như trong các hệ thống nhiệt độ)
 - Bắt đầu thì T nhận giá trị cao, và giảm dần về 0

Local beam search

- Ở mỗi thời điểm (trong quá trình tìm kiếm), luôn lưu giữ k – thay vì chỉ 1 – trạng thái tốt nhất
- Bắt đầu giải thuật: Chọn k trạng thái ngẫu nhiên
- Ở mỗi bước tìm kiếm, sinh ra tất cả các trạng thái kế tiếp của k trạng thái này
- Nếu một trong số các trạng thái là trạng thái đích, thì giải thuật kết thúc (thành công); nếu không, thì chọn k trạng thái tiếp theo tốt nhất (từ tập các trạng thái tiếp theo), và lặp lại bước trên

Giải thuật di truyền: mô tả

- Xây dựng (khởi tạo) **quần thể (population)** ban đầu
 - Tạo nên một số các giả thiết (khả năng của lời giải) ban đầu
 - Mỗi giả thiết khác các giả thiết khác (vd: khác nhau đối với các giá trị của một số tham số nào đó của bài toán)
- Đánh giá quần thể
 - Đánh giá (cho điểm) mỗi giả thiết (vd: bằng cách kiểm tra độ chính xác của hệ thống trên một tập dữ liệu kiểm thử)
 - Trong lĩnh vực sinh học, điểm đánh giá này của mỗi giả thiết được gọi là **độ phù hợp (fitness)** của giả thiết đó
 - Xếp hạng các giả thiết theo mức độ phù hợp của chúng, và chỉ giữ lại các giả thiết tốt nhất (gọi là **các giả thiết phù hợp nhất – survival of the fittest**)
- Sản sinh ra **thế hệ tiếp theo (next generation)**
 - Thay đổi ngẫu nhiên các giả thiết để sản sinh ra thế hệ tiếp theo (gọi là **các con cháu – offspring**)
- Lặp lại quá trình trên cho đến khi ở một thế hệ nào đó có giả thiết tốt nhất có độ phù hợp cao hơn giá trị phù hợp mong muốn (định trước)

Tìm kiếm có đối thủ

- Các thủ tục tìm kiếm sâu dần (IDS) và tìm kiếm A* hữu dụng đối với các bài toán (tìm kiếm) liên quan đến một tác tử
- Thủ tục tìm kiếm cho các bài toán liên quan đến 2 tác tử có mục tiêu đối nghịch nhau (xung đột với nhau)?
 - Tìm kiếm có đối thủ (Adversarial search)
- Phương pháp tìm kiếm có đối thủ được áp dụng phổ biến trong các trò chơi (games)

Các vấn đề của tìm kiếm trong trò chơi

- Khó dự đoán trước được phản ứng của đối thủ
 - Cần xác định (xét) một nước đi phù hợp đối với mỗi phản ứng (nước đi) có thể của đối thủ
- Giới hạn về thời gian (trò chơi có tính giờ)
 - Thường khó (hoặc không thể) tìm được giải pháp tối ưu → Xấp xỉ
- Tìm kiếm có đối thủ đòi hỏi tính hiệu quả (giữa chất lượng của nước đi và chi phí thời gian)
Đây là một yêu cầu khó khăn
- Nguyên tắc trong nhiều trò chơi đối kháng
 - Một người chơi thắng = Người chơi kia thua
 - Mức độ (tổng điểm) thắng của một người chơi = Mức độ (tổng điểm) thua của người chơi kia

Trò chơi cờ ca-rô

- Trò chơi cờ ca-rô là một ví dụ phổ biến trong AI để minh họa về tìm kiếm có đối thủ
 - Vd: <http://www.ourvirtualmall.com/tictac.htm>
- Là trò chơi đối kháng giữa 2 người (gọi là MAX và MIN)
 - Thay phiên nhau đi các nước (moves)
 - Kết thúc trò chơi: Người thắng được thưởng (điểm), người thua bị phạt (điểm)

Biểu diễn bài toán trò chơi đối kháng

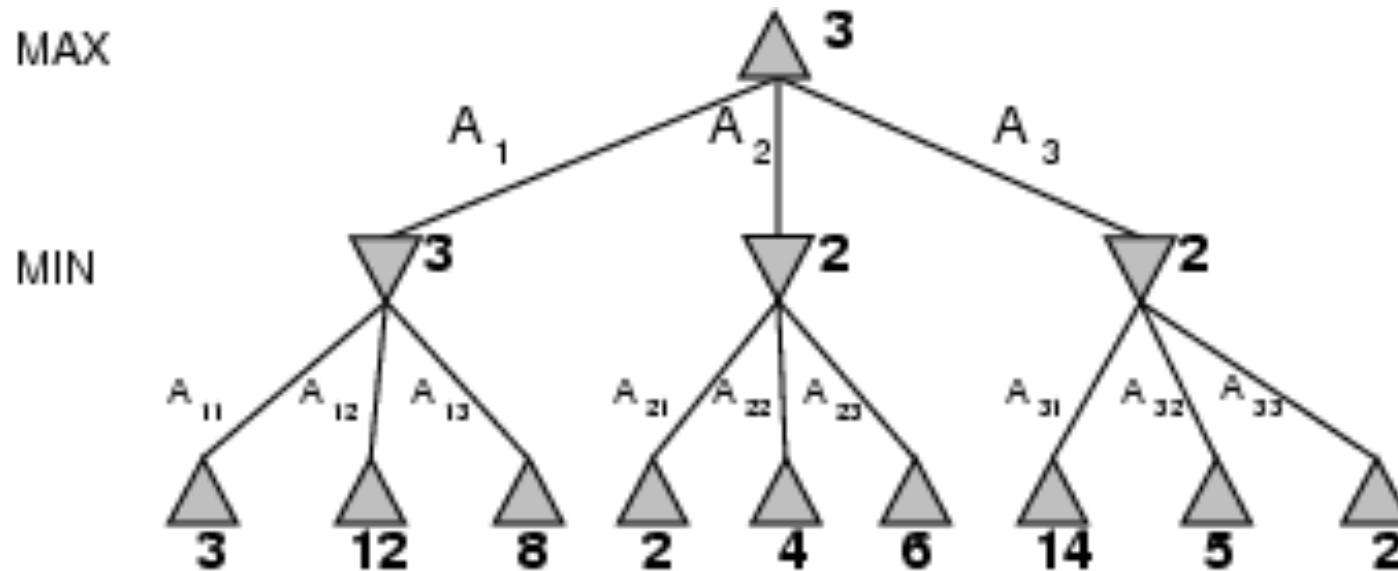
- Trò chơi bao gồm các thông tin
 - Trạng thái bắt đầu (Initial state): Trạng thái của trò chơi + Người chơi nào được đi nước đầu tiên
 - Hàm chuyển trạng thái (Successor function): Trả về thông tin gồm (nước đi, trạng thái)
 - Tất cả các nước đi hợp lệ từ trạng thái hiện tại
 - Trạng thái mới (là trạng thái chuyển đến sau nước đi)
 - Kiểm tra kết thúc trò chơi (Terminal test)
 - Hàm tiện ích (Utility function) để đánh giá các trạng thái
- Trạng thái bắt đầu + Các nước đi hợp lệ = Cây biểu diễn trò chơi (Game tree)

Các chiến lược tối ưu

- Một chiến lược tối ưu là một chuỗi các nước đi giúp đưa đến trạng thái đích mong muốn (vd: chiến thắng)
- Chiến lược của MAX bị ảnh hưởng (phụ thuộc) vào các nước đi của MIN – và ngược lại
- MAX cần chọn một chiến lược giúp cực đại hóa giá trị hàm mục tiêu – với giả sử là MIN đi các nước đi tối ưu
 - MIN cần chọn một chiến lược giúp cực tiểu hóa giá trị hàm mục tiêu
- Chiến lược này được xác định bằng việc xét giá trị MINIMAX đối với mỗi nút trong cây biểu diễn trò chơi
 - Chiến lược tối ưu đối với các trò chơi có không gian trạng thái xác định (deterministic states)

Giá trị MINIMAX

- MAX chọn nước đi ứng với giá trị MINIMAX cực đại (để đạt được giá trị cực đại của hàm mục tiêu)
- Ngược lại, MIN chọn nước đi ứng với giá trị MINIMAX cực tiểu



Giải thuật MINIMAX: các đặc điểm

- Tính hoàn chỉnh
 - Có (nếu cây biểu diễn trò chơi là hữu hạn)
- Tính tối ưu
 - Có (đối với một đối thủ luôn chọn nước đi tối ưu)
- Độ phức tạp về thời gian
 - $O(b^m)$
- Độ phức tạp về bộ nhớ
 - $O(bm)$ (khám phá theo chiến lược tìm kiếm theo chiều sâu)
- Đối với trò chơi cờ vua, hệ số phân nhánh $b \approx 35$ và hệ số mức độ sâu của cây biểu diễn $m \approx 100$
 - Chi phí quá cao – Không thể tìm kiếm chính xác nước đi tối ưu

Cắt tỉa tìm kiếm

- Vấn đề: Giải thuật tìm kiếm MINIMAX vấp phải vấn đề bùng nổ (mức hàm mũ) các khả năng nước đi cần phải xét → không phù hợp với nhiều bài toán trò chơi thực tế
- Chúng ta có thể cắt tỉa (bỏ đi – không xét đến) một số nhánh tìm kiếm trong cây biểu diễn trò chơi
- Phương pháp cắt tỉa α - β (Alpha-beta pruning)
 - Ý tưởng: Nếu một nhánh tìm kiếm nào đó không thể cải thiện đối với giá trị (hàm tiện ích) mà chúng ta đã có, thì không cần xét đến nhánh tìm kiếm đó nữa!
 - Việc cắt tỉa các nhánh tìm kiếm (“tòi”) không ảnh hưởng đến kết quả cuối cùng

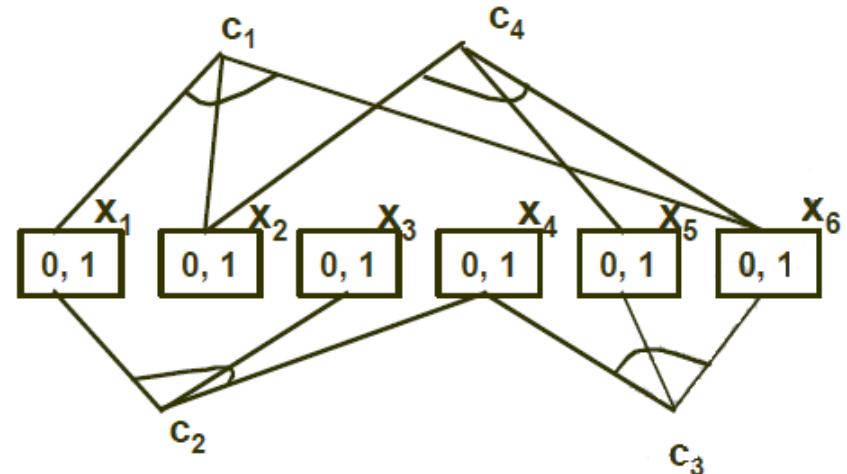
Ràng buộc

- Một **ràng buộc** (constraint) là một quan hệ trên một tập các biến
 - Mỗi biến có (gắn với) một tập các giá trị có thể nhận – gọi là miền giá trị (domain)
 - Trong môn học này, chúng ta chỉ xét các miền hữu hạn các giá trị rời rạc
- Một ràng buộc có thể được biểu diễn bằng
 - Một biểu thức (toán học / logic)
 - Một bảng liệt kê các phép gán giá trị phù hợp cho các biến
- Ví dụ về ràng buộc
 - Tổng các góc trong một tam giác là 180°
 - Độ dài của từ W là 10 ký tự
 - X nhỏ hơn Y
 - Tuấn có thể tham dự buổi seminar vào thứ 3 sau 14h
 - ...



Bài toán thỏa mãn ràng buộc

- Một bài toán thỏa mãn ràng buộc (Constraint Satisfaction Problem – CSP) bao gồm:
 - Một tập hữu hạn các biến X
 - Miền giá trị (một tập hữu hạn các giá trị) cho mỗi biến D
 - Một tập hữu hạn các ràng buộc C
- Một lời giải (solution) của bài toán thỏa mãn ràng buộc là một phép gán đầy đủ các giá trị của các biến sao cho thỏa mãn tất cả các ràng buộc
- Một bài toán thỏa mãn ràng buộc có thể được biểu diễn bằng một đồ thị (graph)



Ví dụ:

Các biến x_1, \dots, x_6 .

Miền giá trị $\{0, 1\}$.

Các ràng buộc:

$$\bullet x_1 + x_2 + x_6 = 1$$

$$\bullet x_1 - x_3 + x_4 = 1$$

$$\bullet x_4 + x_5 - x_6 > 0$$

$$\bullet x_2 + x_5 - x_6 = 0$$

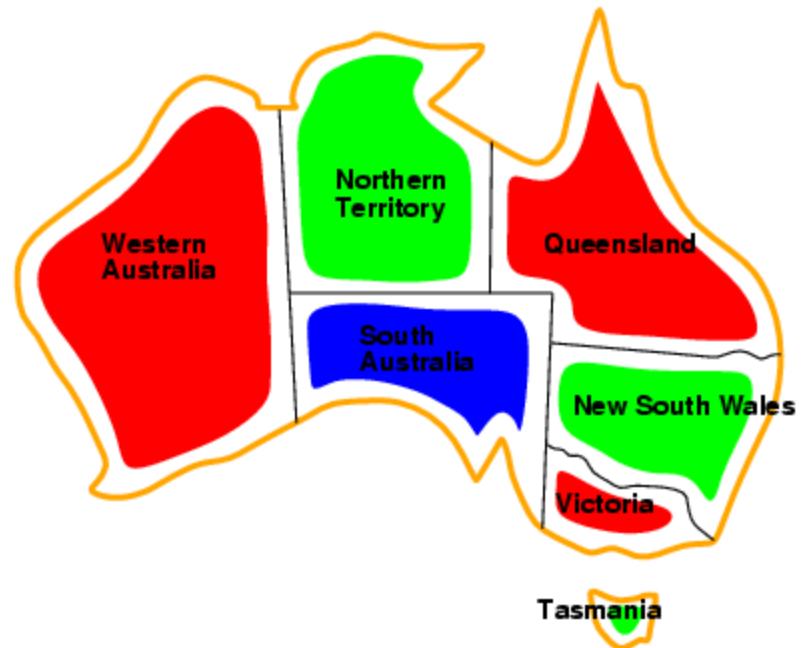
Ví dụ: Bài toán tô màu bản đồ (1)

- Các biển: WA, NT, Q, NSW, V, SA, T
- Các miền giá trị: $D = \{\text{red}, \text{green}, \text{blue}\}$
- Các ràng buộc: Các vùng liền kề nhau phải có màu khác nhau
- Ví dụ:
 - $WA \neq NT$
 - $(WA, NT) = \{(\text{red}, \text{green}), (\text{red}, \text{blue}), (\text{green}, \text{red}), (\text{green}, \text{blue}), (\text{blue}, \text{red}), (\text{blue}, \text{green})\}$



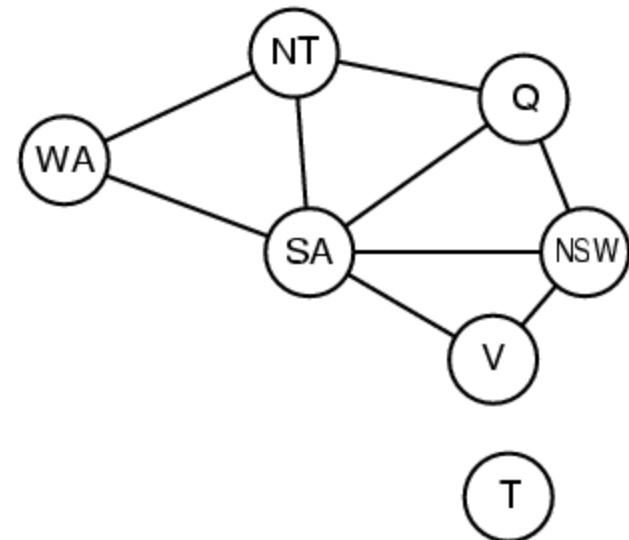
Ví dụ: Bài toán tô màu bản đồ (2)

- Các lời giải là các phép gán **đầy đủ** và **chính xác** (thỏa mãn tất cả các ràng buộc)
- Ví dụ: WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green



Đồ thị các ràng buộc

- Đối với bài toán thỏa mãn ràng buộc nhị phân (binary CSP): Mỗi ràng buộc chỉ liên quan đến 2 biến
- Đồ thị các ràng buộc (constraint graph)
 - ❑ Các nút biểu diễn các biến
 - ❑ Các cạnh biểu diễn các ràng buộc



Các kiểu bài toán thỏa mãn ràng buộc

■ Các biến rời rạc

❑ Các miền giá trị hữu hạn

- Với n biến và kích thước miền giá trị d , thì số lượng các phép gán đầy đủ giá trị cần xét là $O(d^n)$
- Ví dụ: Các bài toán thỏa mãn ràng buộc nhị phân (Boolean CSPs)

❑ Các miền giá trị vô hạn

- Miền giá trị các số nguyên, các chuỗi, ...
- Ví dụ: Trong bài toán xếp lịch công việc, các biến là các ngày bắt đầu và kết thúc đối với mỗi công việc
- Cần một ngôn ngữ biểu diễn ràng buộc (constraint language), ví dụ:
 $StartJob_1 + 5 \leq StartJob_3$

■ Các biến liên tục

- ❑ Ví dụ: Các mốc thời gian bắt đầu và kết thúc đối với các quan sát bằng kính viễn vọng không gian Hubble
- ❑ Bài toán với các ràng buộc tuyến tính có thể giải quyết được ở mức chi phí thời gian đa thức.

Các kiểu ràng buộc

- **Ràng buộc đơn** (unary constraint) chỉ liên quan đến 1 biến
 - ❑ Ví dụ: SA ≠ green
- **Ràng buộc nhị phân** (binary constraint) liên quan đến 2 biến
 - ❑ Ví dụ: SA ≠ WA
- **Ràng buộc bậc cao** (higher-order constraint) liên quan đến *nhiều hơn 2 biến*
 - ❑ Ví dụ: Các ràng buộc trong bài toán mật mã số học (trình bày ở slide tiếp theo)

Vài bài toán CSP trong thực tế

- Các bài toán giao nhiệm vụ
 - Ví dụ: Giáo viên nào dạy lớp nào?
- Các bài toán lập thời khóa (gian) biểu
 - Ví dụ: Lớp học nào được dạy vào thời gian nào và ở đâu?
- Các bài toán lập lịch vận tải (giao hàng) của các công ty
- Các bài toán lập lịch sản xuất của các nhà máy
- Lưu ý: Nhiều bài toán thực tế liên quan đến các biến có giá trị thực (liên tục)

Tìm kiếm quay lui (1)

- **Tìm kiếm quay lui (backtracking)** là giải thuật tìm kiếm được sử dụng phổ biến nhất trong CSP
 - Dựa trên giải thuật tìm kiếm theo chiều sâu (depth-first search)
 - Mỗi lần gán, chỉ làm việc (gán giá trị) cho một biến
 - (Tìm kiếm bằng kiểm thử: mỗi lần gán xác định các giá trị cho tất cả các biến)
- Phương pháp tìm kiếm quay lui đối với bài toán CSP
 - Gán giá trị lần lượt cho các biến – Việc gán giá trị của biến này chỉ được làm sau khi đã hoàn thành việc gán giá trị của biến khác
 - Sau mỗi phép gán giá trị cho một biến nào đó, kiểm tra các ràng buộc có được thỏa mãn bởi tất cả các biến đã được gán giá trị cho đến thời điểm hiện tại – Quay lui (backtrack) nếu có lỗi (không thỏa mãn các ràng buộc)

Tìm kiếm quay lui (2)

- Các yếu tố ảnh hưởng đến phương pháp tìm kiếm quay lui
 - Thứ tự được xét của các biến?
 - Ưu tiên các biến quan trọng hơn (được định nghĩa tùy vào bài toán cụ thể)
 - Ưu tiên xét trước các biến có ít giá trị (miền giá trị nhỏ)
 - Ưu tiên xét trước các biến tham gia vào nhiều ràng buộc
 - Với mỗi biến, thứ tự được xét của các giá trị?
 - Thứ tự ưu tiên của các giá trị với mỗi biến được định nghĩa tùy thuộc vào bài toán cụ thể

Tìm kiếm quay lui: Các vấn đề (1)

- Lặp đi lặp lại lỗi
 - Lý do: Bỏ qua (không khai thác) lý do của mâu thuẫn
 - Ví dụ:
 - Các biến A,B,C,D,E lấy các giá trị trong miền 1..10
 - Ràng buộc: A>E
 - Phương pháp tìm kiếm quay lui thử tất cả các khả năng gán giá trị cho các biến B,C,D cho đến khi phát hiện ra rằng $A \neq 1$
- Giải pháp: Phương pháp Backjumping (chuyển đến xét từ chối sinh ra lỗi)

Tìm kiếm quay lui: Các vấn đề (2)

- Các thao tác (kiểm tra) không cần thiết
 - Lặp lại các kiểm tra ràng buộc không cần thiết
 - Ví dụ:
 - Các biến A,B,C,D,E lấy các giá trị trong miền 1..10
 - Các ràng buộc: $B+8 < D$; $C=5*E$
 - Khi gán giá trị cho các biến C,E, thì các giá trị 1..9 được kiểm tra (lặp đi lặp lại) đối với biến D
- Giải pháp: Phương pháp Backtracking (lưu giữ / ghi nhớ các phép gán tốt và không tốt)

Tìm kiếm quay lui: Các vấn đề (3)

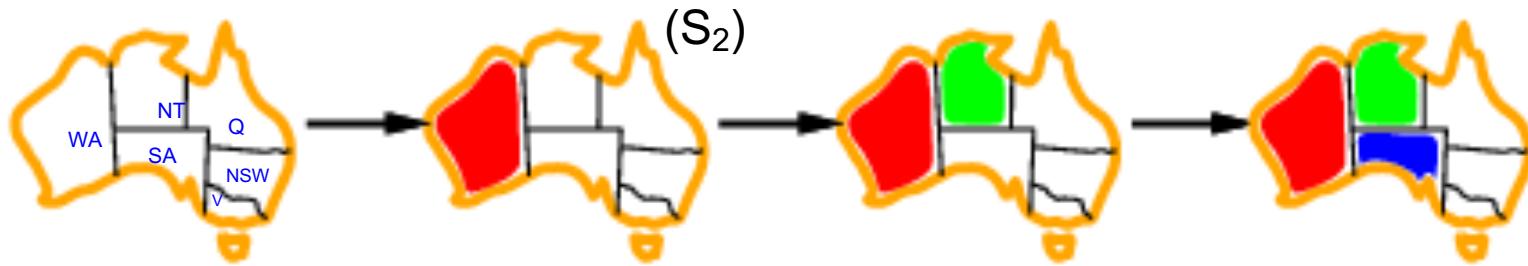
- Phát hiện muộn các mâu thuẫn (vi phạm ràng buộc)
 - Các vi phạm ràng buộc chỉ được phát hiện sau khi các giá trị được gán
 - Ví dụ:
 - Các biến A,B,C,D,E lấy các giá trị trong miền 1..10
 - Ràng buộc: $A=3*E$
 - Chỉ đến khi gán giá trị cho biến E thì mới phát hiện ra rằng $A>2$
- Giải pháp: Phương pháp Forward checking (kiểm tra trước các ràng buộc)

Tìm kiếm quay lui: Cải thiện

- Hiệu quả của phương pháp tìm kiếm quay lui trong CSP có thể được cải thiện bằng
 - Thứ tự xét các biến (để gán giá trị)
 - Thứ tự xét (gán) các giá trị đối với mỗi biến
 - Phát hiện sớm các lỗi (vi phạm ràng buộc) sẽ xảy ra

Biến bị ràng buộc nhiều nhất

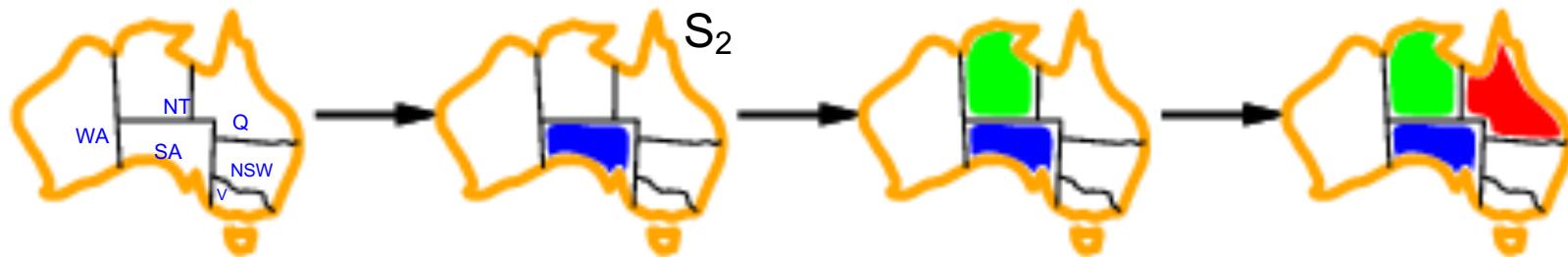
- Quy tắc lựa chọn thứ tự xét các biến: Ưu tiên biến bị ràng buộc nhiều nhất (most constrained variable)
 - Chọn biến có số lượng các giá trị hợp lệ ít nhất
 - Ví dụ: Tại bước S_2 , biến NT được chọn vì nó có số lượng các giá trị hợp lệ ít nhất (2)



- Còn được gọi là quy tắc ưu tiên các biến có tập giá trị hợp lệ nhỏ nhất (Minimum Remaining Values – MRV)

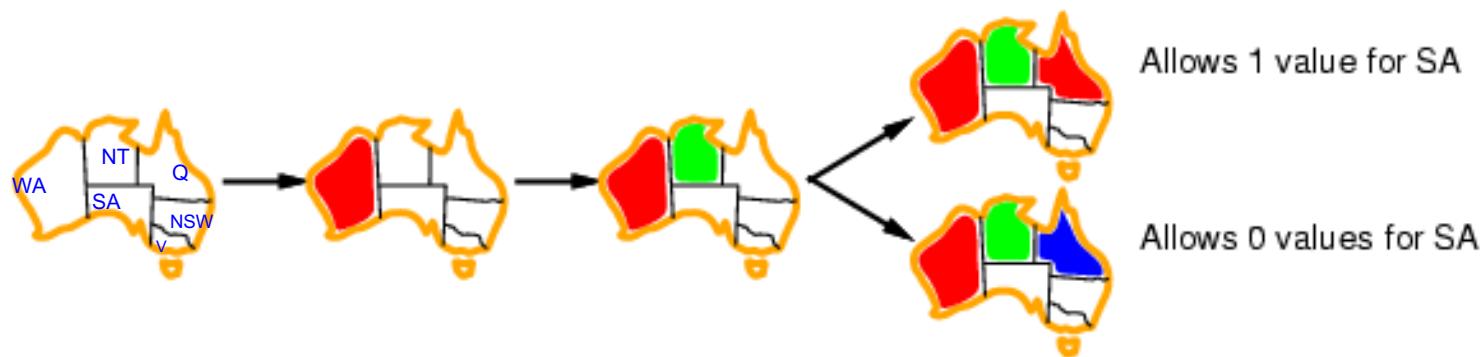
Biến ràng buộc các biến khác nhiều nhất

- Khi có ≥ 2 biến có cùng số lượng giá trị hợp lệ ít nhất, thì chọn biến nào?
 - Ví dụ: Trong ví dụ trước, 2 biến NT và SA có cùng số lượng giá trị hợp lệ ít nhất (2)
- Chọn biến ràng buộc (khống chế) các biến khác (chưa được gán giá trị) nhiều nhất
 - Ví dụ: Tại bước S_2 , tuy cùng mức độ bị ràng buộc, nhưng biến SA nên được xét trước biến NT – vì SA ràng buộc 5 biến khác, còn NT chỉ ràng buộc 3 biến khác



Giá trị ràng buộc các biến khác ít nhất

- Đối với một biến, các giá trị được xét (để gán) theo thứ tự nào?
- Chọn giá trị ràng buộc (không chế) các biến khác (chưa được gán giá trị) ít nhất
 - Giá trị này gây ra hạn chế tối thiểu đối với các khả năng gán giá trị của các biến khác



Kiểm tra tiến (Forward checking)

- Mục đích: Tránh các thất bại, bằng kiểm tra trước các ràng buộc
- Kiểm tra tiến đảm bảo sự phù hợp (consistency) giữa biến đang được xét gán giá trị và các biến khác **có liên quan (ràng buộc) trực tiếp** với nó
- Ý tưởng:
 - ❑ Ở mỗi bước gán giá trị, theo dõi các giá trị hợp lệ (có thể được gán) đối với các biến chưa được gán giá trị
 - ❑ Loại bỏ (dừng) hướng tìm kiếm hiện tại khi có bất kỳ một biến (chưa được gán giá trị) nào đó không còn giá trị hợp lệ

Tìm kiếm cục bộ cho CSP (1)

- Mục đích: Để sử dụng các phương pháp tìm kiếm cục bộ (ví dụ: hill-climbing, simulated annealing) cho bài toán thỏa mãn ràng buộc
- Mỗi trạng thái (của không gian tìm kiếm) ứng với một phép gán *đầy đủ* giá trị cho *tất cả* các biến
 - Không gian tìm kiếm bao gồm cả các trạng thái trong đó các ràng buộc bị vi phạm
 - Dịch chuyển trạng thái = Gán giá trị mới cho các biến
- Trạng thái đích = Trạng thái trong đó tất cả các ràng buộc được thỏa mãn

Tìm kiếm cục bộ cho CSP (2)

- Quá trình tìm kiếm
 - ❑ Lựa chọn biến để gán giá trị mới? → Chọn *ngẫu nhiên* một biến mà giá trị của nó vi phạm các ràng buộc
 - ❑ Đối với một biến, lựa chọn giá trị mới? → Dựa theo chiến lược **min-conflicts**: chọn giá trị mà nó vi phạm ít nhất các ràng buộc
- Ví dụ: Áp dụng phương pháp tìm kiếm cục bộ Hill-climbing, với hàm ước lượng $h(n) =$ tổng số các ràng buộc bị vi phạm
 - ❑ Trạng thái (lân cận) tiếp theo chuyển đến (được xét) là trạng thái ứng với giá trị hàm $h(n)$ tốt hơn (=ít ràng buộc bị vi phạm hơn)

Thỏa mãn ràng buộc: Tổng kết

- Trong một bài toán thỏa mãn ràng buộc (CSP) :
 - Mỗi trạng thái tương ứng với một phép gán giá trị cho các biến
 - Kiểm tra trạng thái đích = Kiểm tra tập các ràng buộc đối với các giá trị của các biến
- Phương pháp quay lui (Backtracking) = Tìm kiếm theo chiều sâu (Depth-first search) với mỗi nút tương ứng với một phép gán giá trị cho một biến
- Các chiến lược chọn thứ tự xét các biến và thứ tự xét các giá trị đối với một biến sẽ ảnh hưởng quan trọng đến hiệu quả của quá trình tìm lời giải
- Phương pháp tìm kiếm tiến (Forward checking) cho phép ngăn chặn các phép gán giá trị đưa đến các thất bại sau đó
- Lan truyền ràng buộc (ví dụ: phương pháp phù hợp cạnh – Arc consistency) cho phép giới hạn hơn nữa các giá trị hợp lệ và cho phép phát hiện các mâu thuẫn
- Phương pháp tìm kiếm cục bộ sử dụng chiến lược Min-conflicts thường hiệu quả trong nhiều bài toán thực tế

Giới thiệu về logic

- **Logic** là ngôn ngữ hình thức cho phép (giúp) biểu diễn thông tin dưới dạng các kết luận có thể được đưa ra
 - Logic = Syntax + Semantics
- **Cú pháp (syntax)**: để xác định các mệnh đề (sentences) trong một ngôn ngữ.
- **Ngữ nghĩa (semantics)**: để xác định “ý nghĩa” của các mệnh đề trong một ngôn ngữ
 - Tức là, xác định sự đúng đắn của một mệnh đề
- Ví dụ: Trong ngôn ngữ của toán học
 - $(x+2 \geq y)$ là một mệnh đề; $(x+y > \{ \})$ không phải là một mệnh đề
 - $(x+2 \geq y)$ là đúng nếu và chỉ nếu giá trị $(x+2)$ không nhỏ hơn giá trị y
 - $(x+2 \geq y)$ là đúng khi $x = 7, y = 1$
 - $(x+2 \geq y)$ là sai khi $x = 0, y = 6$

Cú pháp (syntax)

- Cú pháp = Ngôn ngữ + Lý thuyết chứng minh
- **Ngôn ngữ (Language)**
 - Các ký hiệu (symbols), biểu thức (expressions), thuật ngữ (terms), công thức (formulas) hợp lệ
 - Ví dụ: *one plus one equal two*
- **Lý thuyết chứng minh (Proof theory)**
 - Tập hợp các luật suy diễn cho phép chứng minh (suy luận ra) các biểu thức
 - Ví dụ: Luật suy diễn *any plus zero \vdash any*
- Một **định lý (theorem)** là một mệnh đề logic cần chứng minh
- Việc chứng minh một định lý không cần phải xác định ngữ nghĩa (interpretation) của các ký hiệu!

Ngữ nghĩa (semantics)

- Ngữ nghĩa = Ý nghĩa (diễn giải) của các ký hiệu
- Ví dụ
 - ❑ $I(one)$ nghĩa là **1** ($\in \mathbb{N}$)
 - ❑ $I(two)$ nghĩa là **2** ($\in \mathbb{N}$)
 - ❑ $I(plus)$ nghĩa là phép cộng $+ : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$
 - ❑ $I(equal)$ nghĩa là phép so sánh bằng $= : \mathbb{N} \times \mathbb{N} \rightarrow \{\text{true}, \text{false}\}$
 - ❑ $I(one plus one equal two)$ nghĩa là *true*
- Nếu diễn giải của một biểu thức là đúng (*true*), chúng ta nói rằng phép diễn giải này là một **mô hình (model)** của biểu thức
- Một biểu thức đúng đối với bất kỳ phép diễn giải nào thì được gọi là một biểu thức **đúng đắn (valid)**
 - ❑ Ví dụ: A OR NOT A

Tính bao hàm

- Tính bao hàm có nghĩa là một cái gì đó tuân theo (bị hàm chứa ý nghĩa bởi, được suy ra từ) một cái gì khác:

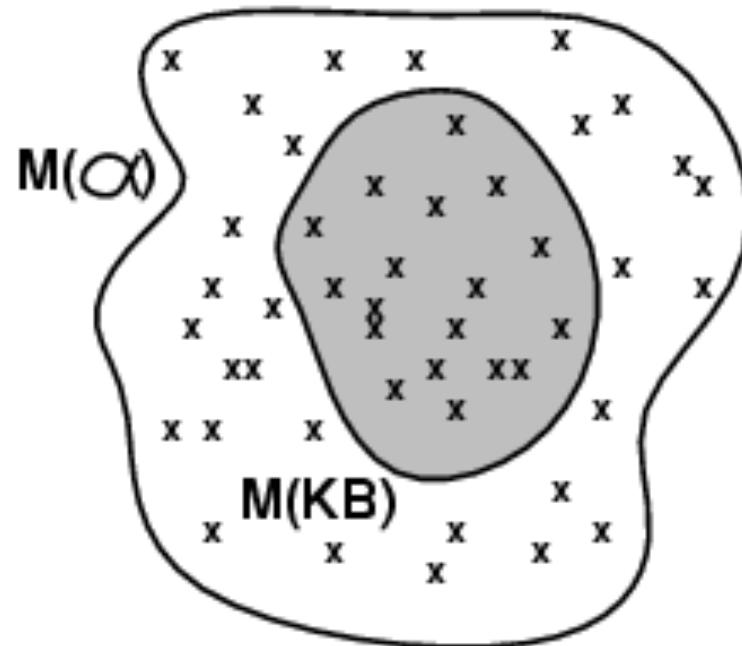
$$KB \models \alpha$$

- Một cơ sở tri thức ***KB bao hàm*** (hàm chứa) mệnh đề α nếu và chỉ nếu α là đúng ***trong mọi mô hình*** (thế giới) mà trong đó *KB* là đúng. Tức là: nếu *KB* đúng, thì α cũng phải đúng

- Ví dụ: Nếu một cơ sở tri thức *KB* chứa các mệnh đề “Đội bóng A đã thắng” và “Đội bóng B đã thắng”, thì *KB* bao hàm mệnh đề “Đội bóng A hoặc đội bóng B đã thắng”
 - Ví dụ: Mệnh đề $(x+y = 4)$ bao hàm mệnh đề $(4 = x+y)$

Các mô hình

- Các nhà logic học thường hay xem xét các sự việc theo các mô hình (models)
- **Định nghĩa:** m là một mô hình của mệnh đề α nếu α là đúng trong m
- $M(\alpha)$ là tập hợp tất cả các mô hình của α
- $KB \models \alpha$ nếu và chỉ nếu $M(KB) \subseteq M(\alpha)$
 - Ví dụ: $KB =$ “Đội bóng A đã thắng và đội bóng B đã thắng”, $\alpha =$ “Đội bóng A đã thắng”



Suy diễn logic (1)

- $KB \vdash_i \alpha$
 - Mệnh đề α **được suy ra** từ KB bằng cách áp dụng thủ tục (suy diễn) i
 - (Nói cách khác) Thủ tục i **suy ra** mệnh đề α từ KB
- **Tính đúng đắn (soundness)**
 - Một thủ tục suy diễn i được gọi là **đúng đắn (sound)**, nếu thủ tục i suy ra **chỉ** các mệnh đề được bao hàm (entailed sentences)
 - Thủ tục i là đúng đắn, nếu bất cứ khi nào $KB \vdash_i \alpha$, thì cũng đúng đối với $KB \models \alpha$
 - Nếu thủ tục i suy ra mệnh đề α , mà α không được bao hàm trong KB , thì thủ tục i là không đúng đắn (unsound)

Suy diễn logic (2)

- **Tính hoàn chỉnh (completeness)**
 - ❑ Một thủ tục suy diễn i được gọi là **hoàn chỉnh (complete)**, nếu thủ tục i có thể suy ra **mọi** mệnh đề được bao hàm (entailed sentences)
 - ❑ Thủ tục i là hoàn chỉnh, nếu bất cứ khi nào $KB \models \alpha$, thì cũng đúng đối với $KB \models_i \alpha$
- (Trong phần tiếp theo của bài giảng) chúng ta sẽ xét đến logic vị từ bậc 1 (first-order logic)
 - ❑ Có khả năng biểu diễn (diễn đạt) hầu hết các phát biểu logic
 - ❑ Với logic vị từ bậc 1, tồn tại một thủ tục suy diễn **đúng đắn** và **hoàn chỉnh**

Suy diễn logic (3)

- Logic là một cách để biểu diễn hình thức và suy diễn tự động
- Việc suy diễn (reasoning) có thể được thực hiện ở mức cú pháp (bằng các chứng minh): **suy diễn演绎 (deductive reasoning)**
- Việc suy diễn có thể được thực hiện ở mức ngữ nghĩa (bằng các mô hình): **suy diễn dựa trên mô hình (model-based reasoning)**

Suy diễn logic (4)

- Suy diễn ngữ nghĩa ở mức của một phép diễn giải (mô hình):
 - Với một biểu thức, có tồn tại một mô hình không? **có thể thỏa mãn được (satisfiability)?**
 - Với một biểu thức và một phép diễn giải, kiểm tra xem phép diễn giải có phải là một mô hình của biểu thức không?: **kiểm tra mô hình (model checking)**
- Suy diễn ngữ nghĩa ở mức của tất cả các phép diễn giải có thể: **kiểm tra tính đúng đắn (validity checking)**

Logic định đề: Cú pháp (1)

- Logic định đề (propositional logic) là loại logic đơn giản nhất
- **Biểu thức định đề (propositional formula)**
 - Một ký hiệu định đề (S_1, S_2, \dots) là một biểu thức (định đề)
 - Các giá trị hằng logic **đúng (true)** và **sai (false)** là các biểu thức
 - Nếu S_1 là một biểu thức, thì $(\neg S_1)$ cũng là một biểu thức (Phép phủ định)

Logic định đề: Cú pháp (2)

■ Biểu thức định đề (propositional formula)...

- Nếu S_1 và S_2 là các biểu thức, thì $(S_1 \wedge S_2)$ cũng là một biểu thức (Phép kết hợp / và)
- Nếu S_1 và S_2 là các biểu thức, thì $(S_1 \vee S_2)$ cũng là một biểu thức (Phép tuyễn / hoặc)
- Nếu S_1 và S_2 là các biểu thức, thì $(S_1 \Rightarrow S_2)$ cũng là một biểu thức (Phép suy ra / kéo theo)
- Nếu S_1 và S_2 là các biểu thức, thì $(S_1 \Leftrightarrow S_2)$ cũng là một biểu thức (Phép tương đương)
- Không gì khác (các dạng trên) là một biểu thức

Logic định đề: Ngữ nghĩa (1)

- Với một mô hình (model) cụ thể, nó sẽ xác định giá trị đúng/sai cho mỗi ký hiệu định đề
 - Ví dụ: Với 3 ký hiệu S_1 , S_2 và S_3 , thì có thể lấy ví dụ một mô hình m_1 xác định như sau:
$$m_1 \equiv (S_1 = \text{sai}, S_2 = \text{đúng}, S_3 = \text{sai})$$
- Với 3 ký hiệu định đề như ví dụ trên, có thể chỉ ra 8 mô hình có thể

Logic định đề: Ngữ nghĩa (2)

- Ngữ nghĩa của một mô hình m = Các quy tắc để đánh giá giá trị chân lý (đúng/sai) của các mệnh đề trong mô hình m đó
 - $\neg S_1$ là đúng, khi và chỉ khi S_1 là sai
 - $S_1 \wedge S_2$ là đúng, khi và chỉ khi S_1 là đúng và S_2 là đúng
 - $S_1 \vee S_2$ là đúng, khi và chỉ khi S_1 là đúng hoặc S_2 là đúng
 - $S_1 \Rightarrow S_2$ là đúng, khi và chỉ khi S_1 là sai **hoặc** S_2 là đúng;
là sai, khi và chỉ khi S_1 là đúng **và** S_2 là sai
 - $S_1 \Leftrightarrow S_2$ là đúng, khi và chỉ khi $S_1 \Rightarrow S_2$ là đúng và $S_2 \Rightarrow S_1$ là đúng
- Ví dụ: Với mô hình m_1 , như trong ví dụ trước, thì giá trị của biểu thức logic định đề sau sẽ là:
 $\neg S_1 \wedge (S_2 \vee S_3) = \text{đúng} \wedge (\text{đúng} \vee \text{sai}) = \text{đúng} \wedge \text{đúng} = \text{đúng}$

Bảng chân lý đối với các toán tử logic

S_1	S_2	$\neg S_1$	$S_1 \wedge S_2$	$S_1 \vee S_2$	$S_1 \Rightarrow S_2$	$S_1 \Leftrightarrow S_2$
sai	sai	đúng	sai	sai	đúng	đúng
sai	đúng	đúng	sai	đúng	đúng	sai
đúng	sai	sai	sai	đúng	sai	sai
đúng	đúng	sai	đúng	đúng	đúng	đúng

Tương đương logic

- Hai mệnh đề được gọi là tương đương logic khi và chỉ khi hai mệnh đề này luôn đúng trong cùng mô hình:

$$\alpha \equiv \beta \text{ khi và chỉ khi } \alpha \models \beta \text{ và } \beta \models \alpha$$

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \text{ commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \text{ commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \text{ associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \text{ associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \text{ double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \text{ contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \text{ implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \text{ biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \text{ de Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \text{ de Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \text{ distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \text{ distributivity of } \vee \text{ over } \wedge$$

Biểu diễn bằng logic định đề: Ví dụ

■ Giả sử chúng ta có các định đề sau

- $p \equiv$ “Chiều nay trời nắng”
- $q \equiv$ “Thời tiết lạnh hơn hôm qua”
- $r \equiv$ “Tôi sẽ đi bơi”
- $s \equiv$ “Tôi sẽ đi đá bóng”
- $t \equiv$ “Tôi sẽ về đến nhà vào buổi tối”

■ Biểu diễn các phát biểu trong ngôn ngữ tự nhiên

- “Chiều nay trời *không* nắng và thời tiết lạnh hơn hôm qua”: $\neg p \wedge q$
- “Tôi sẽ đi bơi *nếu như* chiều nay trời nắng”: $p \rightarrow r$
- “*Nếu* tôi (sẽ) *không* đi bơi *thì* tôi sẽ đi đá bóng”: $\neg r \rightarrow s$
- “*Nếu* tôi (sẽ) đi đá bóng *thì* tôi sẽ về nhà vào buổi tối”: $s \rightarrow t$

Mâu thuẫn và Tautology

- Một biểu thức logic định đê luôn có giá trị sai (false) trong mọi phép diễn giải (mọi mô hình) thì được gọi là một **mâu thuẫn (contradiction)**
 - Ví dụ: $(p \wedge \neg p)$
- Một biểu thức logic định đê luôn có giá trị đúng (true) trong mọi phép diễn giải (mọi mô hình) thì được gọi là một **tautology**
 - Ví dụ: $(p \vee \neg p)$
 $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$
 $\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q)$

Tính thỏa mãn được và Tính đúng đắn

- Một biểu thức logic định đề là **thỏa mãn được (satisfiable)**, nếu biểu thức đó đúng trong *một mô hình* nào đó
 - Ví dụ: $A \vee B$, $A \wedge B$
- Một biểu thức là **không thể thỏa mãn được (unsatisfiable)**, nếu *không tồn tại bất kỳ mô hình nào* mà trong đó biểu thức là đúng
 - Ví dụ: $A \wedge \neg A$
- Một biểu thức là **đúng đắn (valid)**, nếu biểu thức đúng trong *mọi mô hình*
 - Ví dụ: *đúng*; $A \vee \neg A$; $A \Rightarrow A$; $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Bài toán chứng minh logic

- Với một cơ sở tri trực (một tập các mệnh đề) KB và một mệnh đề α cần chứng minh (gọi là một định lý)
 - Cơ sở tri thức KB có bao hàm (về mặt ngũ nghĩa) α hay không: $KB \models \alpha$?
 - Nói cách khác, α có thể được suy ra (được chứng minh) từ cơ sở tri thức KB hay không?
 - **Câu hỏi đặt ra:** Liệu có tồn tại một thủ tục (suy diễn) có thể giải quyết được bài toán chứng minh logic, trong một số hữu hạn các bước?
 - Đối với logic định đề, câu trả lời là có!

Giải quyết bài toán chứng minh logic

- Mục đích: để trả lời câu hỏi $KB \models \alpha ?$
- Có 3 phương pháp (chứng minh) phổ biến:
 - Sử dụng bảng chân lý (Truth-table)
 - Áp dụng các luật suy diễn (Inference rules)
 - Chuyển về bài toán chứng minh thỏa mãn (SAT)
 - Phương pháp chứng minh bằng phản chứng (Refutation)

Chứng minh dựa trên bảng chân lý (1)

- Bài toán chứng minh: $\text{KB} \models \alpha$?
- Kiểm tra tất cả các phép diễn giải có thể (tất cả các mô hình có thể) mà trong đó KB là đúng, để xem α đúng hay sai
- Bảng chân lý: Liệt kê các giá trị chân lý (đúng/sai) của các mệnh đề, đối với tất cả các phép diễn giải có thể
 - Các phép gán giá trị đúng/sai đối với các ký hiệu định đề

		KB		α
p	q	$p \vee q$	$p \leftrightarrow q$	$(p \vee \neg q) \wedge q$
đúng	đúng	đúng	đúng	đúng
đúng	sai	đúng	sai	sai
sai	đúng	đúng	sai	sai
sai	sai	sai	đúng	sai

← chứng minh

Chứng minh dựa trên bảng chân lý (2)

- $KB = (p \vee r) \wedge (q \vee \neg r)$
- $\alpha = (p \vee q)$
- $KB \models \alpha ?$

p	q	r	$p \vee r$	$q \vee \neg r$	KB	α
đúng	đúng	đúng	đúng	đúng	đúng	đúng
đúng	đúng	sai	đúng	đúng	đúng	đúng
đúng	sai	đúng	đúng	sai	sai	đúng
đúng	sai	sai	đúng	đúng	đúng	đúng
sai	đúng	đúng	đúng	đúng	đúng	đúng
sai	đúng	sai	sai	đúng	sai	đúng
sai	sai	đúng	đúng	sai	sai	sai
sai	sai	sai	sai	đúng	sai	sai

Chứng minh dựa trên bảng chân lý (3)

- Đối với logic định đề, phương pháp chứng minh dựa trên bảng chân lý có tính *đúng đắn (sound)* và *hoàn chỉnh (complete)*
- Độ phức tạp tính toán của phương pháp chứng minh dựa trên bảng chân lý
 - ❑ Hàm mũ đối với số lượng (n) các ký hiệu định đề: 2^n
 - ❑ Nhưng chỉ có một tập con (nhỏ) của tập các khả năng gán giá trị chân lý, mà trong đó KB và α là đúng

Chứng minh bằng các luật suy diễn (2)

- Luật suy diễn loại bỏ phủ định hai lần (**Elimination of Double Negation**)

$$\neg\neg p \quad \underline{\hspace{2cm}}$$

$$p$$

- Luật suy diễn hợp giải (**Resolution**)

$$p \vee q, \quad \neg q \vee r \quad \underline{\hspace{2cm}}$$

$$p \vee r$$

- Luật suy diễn hợp giải đơn (**Unit Resolution**)

$$p \vee q, \quad \neg q \quad \underline{\hspace{2cm}}$$

$$p$$

- Tất cả các luật suy diễn trên đều có tính *đúng đắn* (*sound*)!

Chứng minh bằng luật suy diễn: Ví dụ (1)

■ Giả sử có tập giả thiết KB

- 1) $p \wedge q$
- 2) $p \rightarrow r$
- 3) $(q \wedge r) \rightarrow s$

■ Cần chứng minh định lý s

■ Từ 1) và sử dụng luật And-Elimination, ta có:

- 4) p

■ Từ 2), 4), và sử dụng luật Modus Ponens, ta có:

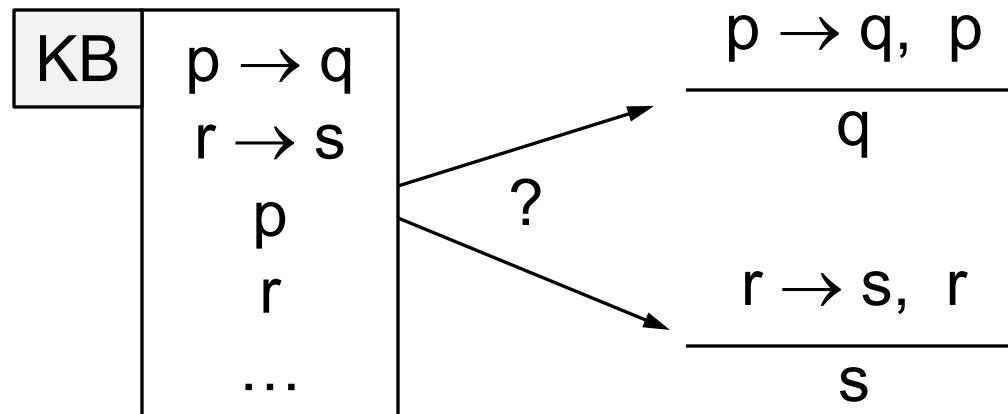
- 5) r

Chứng minh bằng luật suy diễn: Ví dụ (2)

- ...
- Từ 1), và sử dụng luật And-Elimination, ta có:
 - 6) q
- Từ 5), 6), và sử dụng luật And-Introduction, ta có:
 - 7) $(q \wedge r)$
- Từ 7), 3), và sử dụng luật Modus-Ponens, ta có:
 - 8) s
- Vậy định lý (biểu thức logic) s được chứng minh là đúng!

Suy diễn logic và Tìm kiếm

- Để chứng minh định lý α là đúng đối với tập giả thiết KB , cần áp dụng một chuỗi các luật suy diễn đúng đắn
- **Vấn đề:** Ở mỗi bước suy diễn, có nhiều luật có thể áp dụng được
 - Chọn luật nào để áp dụng tiếp theo?
- Đây là vấn đề của bài toán tìm kiếm (search)



Chuyển đổi các biểu thức logic

- Trong logic định đề
 - ❑ Một biểu thức có thể bao gồm nhiều liên kết: \neg , \wedge , \vee , \rightarrow , \leftrightarrow
 - ❑ Một biểu thức có thể bao gồm nhiều biểu thức con (lồng) khác
- Chúng ta có cần sử dụng tất cả các liên kết logic để biểu diễn một biểu thức phức tạp?
 - ❑ Không.
 - ❑ Chúng ta có thể viết lại (chuyển đổi) một biểu thức logic định đề thành một biểu thức tương đương *chỉ chứa các liên kết \neg , \wedge , \vee*

Các dạng chuẩn

- Các biểu thức trong logic định đề có thể được chuyển đổi về một trong các dạng chuẩn (Normal forms)
 - Giúp đơn giản hóa quá trình suy diễn
- **Dạng chuẩn kết hợp** (Conjunctive normal form – CNF)
 - Là kết hợp (liên kết VÀ) của các mệnh đề (clauses)
 - Mỗi mệnh đề (clause) là một liên kết HOẶC của các ký hiệu định đề đơn
 - Ví dụ: $(p \vee q) \wedge (\neg q \vee \neg r \vee s)$
- **Dạng chuẩn tuyễn** (Disjunctive normal form – DNF)
 - Là liên kết HOẶC của các mệnh đề (clauses)
 - Mỗi mệnh đề (clause) là một liên kết VÀ của các ký hiệu định đề đơn
 - Ví dụ: $(p \wedge \neg q) \vee (\neg p \wedge r) \vee (r \wedge \neg s)$

Chuyển đổi về dạng chuẩn CNF: Ví dụ

Chuyển đổi về dạng chuẩn CNF: $\neg(p \rightarrow q) \vee (r \rightarrow p)$

1. Loại bỏ các liên kết $\rightarrow, \leftrightarrow$

$$\neg(\neg p \vee q) \vee (\neg r \vee p)$$

2. Sử dụng các phép biến đổi tương đương (vd: luật De Morgan và phép phủ định 2 lần)

$$(p \wedge \neg q) \vee (\neg r \vee p)$$

3. Sử dụng các luật kết hợp (associative rules) và phân bố (distributive rules)

$$(p \vee \neg r \vee p) \wedge (\neg q \vee \neg r \vee p)$$

$$(p \vee \neg r) \wedge (\neg q \vee \neg r \vee p)$$

Giải quyết bài toán SAT

■ Phương pháp Backtracking

- ❑ Áp dụng chiến lược tìm kiếm theo chiều sâu (Depth-first search)
- ❑ Xét một biến (một định đề đơn), xét các khả năng gán giá trị (đúng/sai) cho biến đó
- ❑ Lặp lại, cho đến khi tất cả các biến được gán giá trị, hoặc việc gán giá trị cho tập con của tập tất cả các biến, làm cho **biểu thức là sai**

■ Các phương pháp tối ưu hóa lặp (Iterative optimization methods)

- ❑ Bắt đầu với một phép gán ngẫu nhiên các giá trị đúng/sai cho các ký hiệu định đề
- ❑ Đổi giá trị (*đúng thành sai / sai thành đúng*) đối với một biến
- ❑ Heuristic: ưu tiên các phép gán giá trị làm cho nhiều mệnh đề (hơn) đúng
- ❑ Sử dụng các phương pháp tìm kiếm cục bộ: Simulated Annealing, Walk-SAT

Bài toán suy diễn vs. Bài toán thỏa mãn được

■ Bài toán suy diễn logic

- ❑ Cần chứng minh: biểu thức logic (định lý) α được bao hàm bởi tập các mệnh đề KB
- ❑ Nói cách khác: với mọi phép diễn giải mà trong đó KB đúng, thì α có đúng?

■ Bài toán thỏa mãn được (SAT)

- ❑ Có tồn tại một phép gán giá trị đúng/sai cho các ký hiệu định đề (một phép diễn giải) sao cho biểu thức α là đúng?

■ Mối quan hệ:

$KB \models \alpha$

nếu và chỉ nếu:

$(KB \wedge \neg\alpha)$

là **không thể thỏa mãn được**
(unsatisfiable)

Luật suy diễn hợp giải (1)

■ Luật suy diễn hợp giải (Resolution)

$$p \vee q, \quad \neg q \vee \underline{r} \\ p \vee r$$

- Luật suy diễn hợp giải áp dụng được đối với các biểu thức logic ở dạng chuẩn CNF
- Luật suy diễn hợp giải có tính *đúng đắn* (*sound*), nhưng *không có tính hoàn chỉnh* (*incomplete*)
 - ❑ Tập giả thiết (cơ sở tri thức) KB chứa biểu thức $(p \wedge q)$
 - ❑ Cần chứng minh: $(p \vee q) ?$
 - ❑ Luật suy diễn hợp giải không thể suy ra được biểu thức cần chứng minh!

Luật suy diễn hợp giải (2)

- Chuyển bài toán chứng minh logic về bài toán SAT
 - Phương pháp chứng minh bằng phản chứng
 - Việc chứng minh sự mâu thuẫn của: $(KB \wedge \neg\alpha)$
 - Tương đương việc chứng minh sự bao hàm: $KB \models \alpha$
- Luật suy diễn hợp giải (Resolution rule)
 - Nếu các biểu thức trong tập KB và biểu thức $\neg\alpha$ đều ở dạng CNF, thì áp dụng luật suy diễn hợp giải sẽ xác định tính (không) thỏa mãn được của $(KB \wedge \neg\alpha)$

Giải thuật hợp giải

- Chuyển đổi tất cả các biểu thức trong KB về dạng chuẩn CNF
- Áp dụng liên tiếp luật suy diễn hợp giải (Resolution rule) bắt đầu từ: $(KB \wedge \neg\alpha)$
 - KB là kết hợp của các biểu thức ở dạng chuẩn CNF
 - Do đó, $(KB \wedge \neg\alpha)$ cũng là một biểu thức ở dạng chuẩn CNF!
- Quá trình áp dụng luật suy diễn hợp giải dừng lại khi:
 - Có mâu thuẫn xảy ra
 - Sau khi hợp giải, thu được (suy ra) biểu thức rỗng (mâu thuẫn)
$$\frac{p, \quad \neg p}{\emptyset}$$
 - Không có biểu thức mới nào được sinh ra nữa

Kết luận sao?

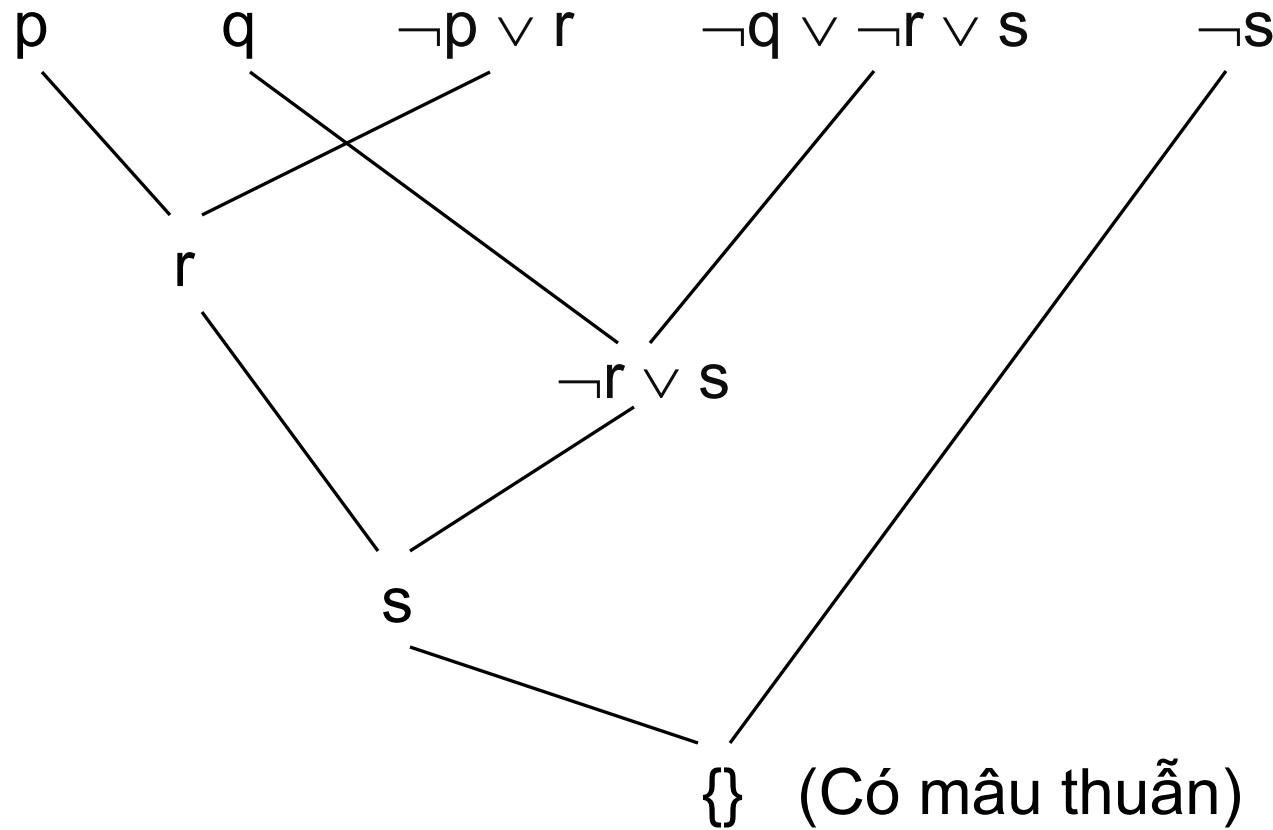
Chứng minh bằng hợp giải: Ví dụ (1)

- Giả sử có tập giả thiết KB
 - $p \wedge q$
 - $p \rightarrow r$
 - $(q \wedge r) \rightarrow s$
- Cần chứng minh định lý s
- Bước 1. Chuyển đổi KB về dạng chuẩn CNF
 - $(p \rightarrow r)$ được chuyển thành $(\neg p \vee r)$
 - $((q \wedge r) \rightarrow s)$ được chuyển thành $(\neg q \vee \neg r \vee s)$
- Bước 2. Phủ định biểu thức cần chứng minh
 - $\neg s$
- Bước 3. Áp dụng liên tiếp luật hợp giải đối với $(KB \wedge \neg s)$:
 $\{p, q, \neg p \vee r, \neg q \vee \neg r \vee s, \neg s\}$

Chứng minh bằng hợp giải: Ví dụ (2)

- Bắt đầu quá trình hợp giải, ta có tập các mệnh đề:
 - 1) p
 - 2) q
 - 3) $\neg p \vee r$
 - 4) $\neg q \vee \neg r \vee s$
 - 5) $\neg s$
- Hợp giải 1) và 3), ta thu được
 - 6) r
- Hợp giải 2) và 4), ta thu được
 - 7) $\neg r \vee s$
- Hợp giải 6) và 7), ta thu được
 - 8) s
- Hợp giải 8) và 5), ta thu được mâu thuẫn ($\{\}$)
- Tức là biểu thức ban đầu (s) được chứng minh là đúng

Chứng minh bằng hợp giải: Ví dụ (3)



Dạng chuẩn Horn

- Một biểu thức logic ở dạng chuẩn Horn nếu:
 - Biểu thức đó là một liên kết VÀ của các mệnh đề
 - Mỗi mệnh đề là một liên kết HOẶC các ký hiệu (literals), và **có tối đa là 1 ký hiệu khẳng định** (positive literal)
 - Ví dụ: $(p \vee \neg q) \wedge (\neg p \vee \neg r \vee s)$
- Không phải mọi biểu thức logic định đê đều có thể được chuyển về dạng chuẩn Horn!
- Biểu diễn tập giả thiết KB ở dạng chuẩn Horn
 - **Các luật (Rules)**
 - $(\neg p_1 \vee \neg p_2 \vee \dots \vee \neg p_n \vee q)$
 - Tương đương với luật: $(p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow q)$
 - **Các sự kiện (Facts)**
 - p, q
 - **Các ràng buộc toàn vẹn (Integrity constraints)**
 - $(\neg p_1 \vee \neg p_2 \vee \dots \vee \neg p_n)$
 - Tương đương với luật: $(p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow sai)$

Luật suy diễn Modus Ponens tổng quát

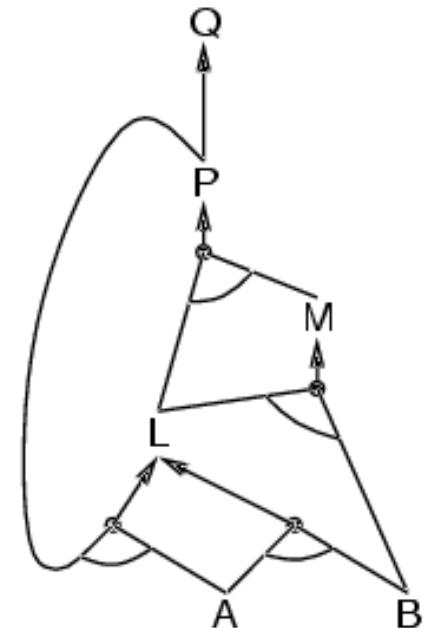
$$\frac{(p_1 \wedge p_2 \wedge \dots \wedge p_n \rightarrow q), \ p_1, \ p_2, \ \dots, \ p_n}{q}$$

- Luật suy diễn Modus Ponens có tính *đúng đắn* (*sound*) và *hoàn chỉnh* (*complete*), đối với các ký hiệu định đề và đối với tập các biểu thức KB ở dạng chuẩn Horn
- Luật suy diễn Modus Ponens có thể được sử dụng với cả 2 chiến lược suy diễn (suy diễn tiến và suy diễn lùi)

Suy diễn tiến (forward chaining)

- Với một tập các mệnh đề giả thiết (cơ sở tri thức) KB , cần suy ra mệnh đề kết luận Q
- Ý tưởng: Lặp lại 2 bước sau cho đến khi suy ra được kết luận
 - Áp dụng các luật có mệnh đề giả thiết được thỏa mãn trong KB
 - Bổ sung kết luận của các luật đó vào KB

$$\begin{aligned} P &\Rightarrow Q \\ L \wedge M &\Rightarrow P \\ B \wedge L &\Rightarrow M \\ A \wedge P &\Rightarrow L \\ A \wedge B &\Rightarrow L \\ A \\ B \end{aligned}$$



Suy diễn tiên: Ví dụ (7)

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

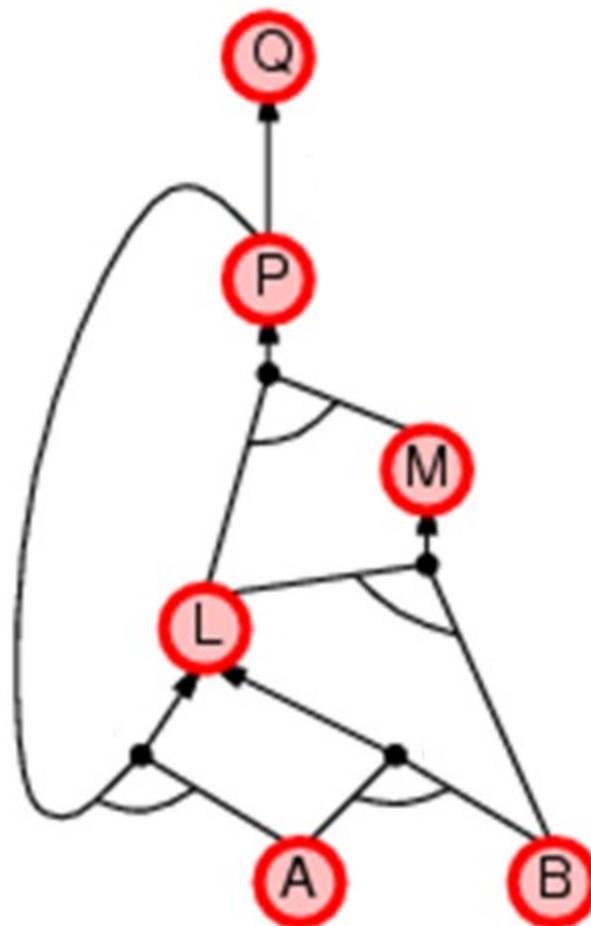
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$$A$$

$$B$$



Suy diễn lùi (backward chaining)

- Ý tưởng: Quá trình suy diễn bắt đầu từ mệnh đề kết luận Q
- Để chứng minh Q bằng tập mệnh đề (cơ sở tri thức) KB
 - Kiểm tra xem Q đã được chứng minh (trong KB) chưa,
 - Nếu chưa, tiếp tục chứng minh tất cả các mệnh đề giả thiết của một luật nào đó (trong KB) có mệnh đề kết luận là Q
- Tránh các vòng lặp
 - Kiểm tra xem các mệnh đề mới đã có trong danh sách các mệnh đề cần chứng minh chưa? – Nếu rồi, thi không bổ sung (lại) nữa!
- Tránh việc chứng minh lặp lại đối với 1 mệnh đề
 - Đã được chứng minh (trước đó) là đúng
 - Đã được chứng minh (trước đó) là không thể thỏa mãn được (sai) trong KB

Suy diễn tiến hay Suy diễn lùi?

- Suy diễn tiến là quá trình dựa trên dữ liệu (data-driven)
 - Ví dụ: việc nhận dạng đối tượng, việc đưa ra quyết định
- Suy diễn tiến có thể thực hiện nhiều bước suy diễn dự thừa – chẳng liên quan tới (cần thiết cho) mục tiêu cần chứng minh
- Suy diễn lùi là quá trình hướng tới mục tiêu (goal-driven), phù hợp cho việc giải quyết vấn đề

Logic định đề: Ưu và nhược điểm

- (+) Logic định đề cho phép dễ dàng phát biểu (biểu diễn) cơ sở tri thức bằng tập các mệnh đề
- (+) Logic định đề cho phép làm việc với các thông tin ở dạng phủ định, dạng tuyễn (disjunctive)
- (+) Logic định đề có tính cấu tạo (kết cấu)
 - Ngữ nghĩa của mệnh đề $(S_1 \wedge S_2)$ được suy ra từ ngữ nghĩa của S_1 và ngữ nghĩa của S_2
- (+) Ngữ nghĩa trong logic định đề không phụ thuộc ngữ cảnh (context-independent)
 - Không như trong ngôn ngữ tự nhiên (ngữ nghĩa phụ thuộc vào ngữ cảnh của các câu nói)
- (-) Khả năng diễn đạt (biểu diễn) của logic định đề là rất hạn chế
 - Logic định đề không thể diễn đạt được (như trong ngôn ngữ tự nhiên): “Nếu X là cha của Y, thì Y là con của X”
 - Logic định đề phải liệt kê (xét) mọi khả năng gán giá trị chân lý (đúng/sai) cho X và Y

Dữ liệu, Thông tin, Tri thức (1)

- **Dữ liệu (data)** thường được định nghĩa là các sự kiện (facts) hoặc các ký hiệu (symbols)
- **Thông tin (information)** thường được định nghĩa là dữ liệu đã được xử lý hoặc chuyển đổi thành những dạng hoặc cấu trúc phù hợp cho việc sử dụng của con người
 - Thu được sau khi loại bỏ những thứ dư thừa và giữ lại phần cốt lõi.
- **Tri thức (knowledge)** thường được định nghĩa là sự hiểu biết (nhận thức) về thông tin
 - Thu được sau quá trình nhận thức, phát hiện, hoặc học tập

Dữ liệu, Thông tin, Tri thức (2)

- Dữ liệu
 - ❑ Nhiệt độ ngoài trời là 5 độ C
- Thông tin
 - ❑ Ngoài trời thời tiết lạnh
- Tri thức
 - ❑ Nếu ngoài trời thời tiết lạnh thì bạn nên mặc áo choàng ấm (khi đi ra ngoài)

Biểu diễn tri thức

- Biểu diễn tri thức (Knowledge representation) là một bài toán quan trọng của Trí tuệ nhân tạo
 - các phương pháp, cách thức biểu diễn tri thức và các công cụ hỗ trợ việc biểu diễn tri thức
- Tồn tại nhiều phương pháp biểu diễn tri thức
 - Luật sản xuất (Production rules)
 - Khung (Frames)
 - Mạng ngữ nghĩa (Semantic networks)
 - Ontology
 - Các mô hình xác suất (probabilistic models)
 - Học sâu (deep learning)
 - ...

Biểu diễn tri thức: vấn đề

- Tính hoàn chỉnh (Completeness)
 - ❑ Phương pháp biểu diễn có hỗ trợ việc thu thập và thể hiện mọi khía cạnh của tri thức (của một lĩnh vực cụ thể)?
- Tính ngắn gọn (Conciseness)
 - ❑ Phương pháp biểu diễn có cho phép việc thu thập tri thức một cách hiệu quả?
 - ❑ Phương pháp biểu diễn có cho phép việc lưu trữ và truy nhập dễ dàng tri thức không?
- Tính hiệu quả về tính toán (Computational efficiency)
- Tính rõ ràng, dễ hiểu (Transparency)
 - ❑ Phương pháp biểu diễn có cho phép diễn giải (để người dùng hiểu) về các hoạt động và các kết luận của hệ thống?

Biểu diễn tri thức: bằng luật (1)

- Biểu diễn tri thức bằng các luật (rules) là cách biểu diễn phổ biến trong các hệ cơ sở tri thức
 - Một luật chứa đựng (biểu diễn) tri thức về việc giải quyết một vấn đề nào đó
 - Các luật được tạo nên khá dễ dàng, và dễ hiểu
- Một luật được biểu diễn ở dạng:

IF A₁ AND A₂ AND ... AND A_n THEN B

- A_i
 - Là các **điều kiện** (**conditions, antecedents, premises**)
- B
 - Là **kết luận** (**conclusion, consequence, action**)

Biểu diễn tri thức: bảng luật (2)

■ Mệnh đề điều kiện của một luật

- Không cần sử dụng toán tử logic OR
- Một luật với toán tử logic OR trong mệnh đề điều kiện, thì sẽ được chuyển thành một tập các luật tương ứng không chứa OR
- Ví dụ: Luật ($\text{IF } A_1 \vee A_2 \text{ THEN } B$) được chuyển thành 2 luật ($\text{IF } A_1 \text{ THEN } B$) và ($\text{IF } A_2 \text{ THEN } B$)

■ Mệnh đề kết luận của một luật

- Không cần sử dụng toán tử logic AND
- Một luật với toán tử logic AND trong mệnh đề kết luận, thì sẽ được chuyển thành một tập các luật tương ứng không chứa AND
- Ví dụ: Luật ($\text{IF } \dots \text{ THEN } B_1 \wedge B_2$) được chuyển thành 2 luật ($\text{IF } \dots \text{ THEN } B_1$) and ($\text{IF } \dots \text{ THEN } B_2$)
- **Không cho phép sử dụng toán tử OR!**

Các kiểu luật

- Các kiểu luật khác nhau để biểu diễn các kiểu tri thức khác nhau
- Quan hệ liên kết
 - IF addressAt (x, Hospital) THEN healthIs (x, Bad)
- Quan hệ nguyên nhân (kết quả)
 - IF diseaseType (x, Infection) THEN tempIs (x, High)
- Tình huống và hành động (gợi ý)
 - IF diseaseType (x, Infection) THEN takeMedicine (x, Antibiotic)
- Quan hệ logic
 - IF tempGreater (x, 37) THEN isFever (x)

Vài vấn đề với biểu diễn luật

- Các luật có chứa các vòng lặp
 - IF A THEN A
 - {IF A THEN B, IF B THEN C, IF C THEN A}
- Các luật có chứa mâu thuẫn
 - {IF A THEN B, IF B THEN C, IF A AND D THEN \neg C}
- Các kết luận không thể suy ra được (từ các luật hiện có)
- Khó khăn trong việc thay đổi (cập nhật) cơ sở tri thức
 - Cơ sở tri thức cũ: {IF A_1 THEN B_1 , IF A_2 THEN B_2 , ..., IF A_n THEN B_n }
 - Cần bổ sung thêm điều kiện C vào tất cả các luật
 - Cơ sở tri thức mới: {IF A_1 AND C THEN B_1 , IF A_2 AND C THEN B_2 , ..., IF A_n AND C THEN B_n }

Sử dụng các luật trong suy diễn

■ So khớp mẫu (Pattern matching)

- ❑ Để kiểm tra một luật có thể được sử dụng (áp dụng) hay không
- ❑ Ví dụ: Nếu cơ sở tri thức chứa đựng tập các luật {IF A_1 THEN B_1 , IF A_1 AND A_2 THEN B_2 , IF A_2 AND A_3 THEN B_3 } và các sự kiện (được lưu trong bộ nhớ làm việc) bao gồm A_1 và A_2 , thì 2 luật đầu tiên có thể được sử dụng

■ Chuỗi suy diễn (chuỗi áp dụng các luật)

- ❑ Xác định trật tự áp dụng các luật trong quá trình suy diễn
- ❑ Với một tập các luật và một tập các sự kiện (các giả thiết), các luật nào nên được sử dụng, và theo trật tự nào, để đạt tới (suy ra) một kết luật cần chứng minh?
- ❑ 2 chiến lược suy diễn: tiến (forward) vs. lùi (backward)
- ❑ 2 chiến lược suy diễn này đã được trình bày trong bài trước!

Giải quyết xung đột

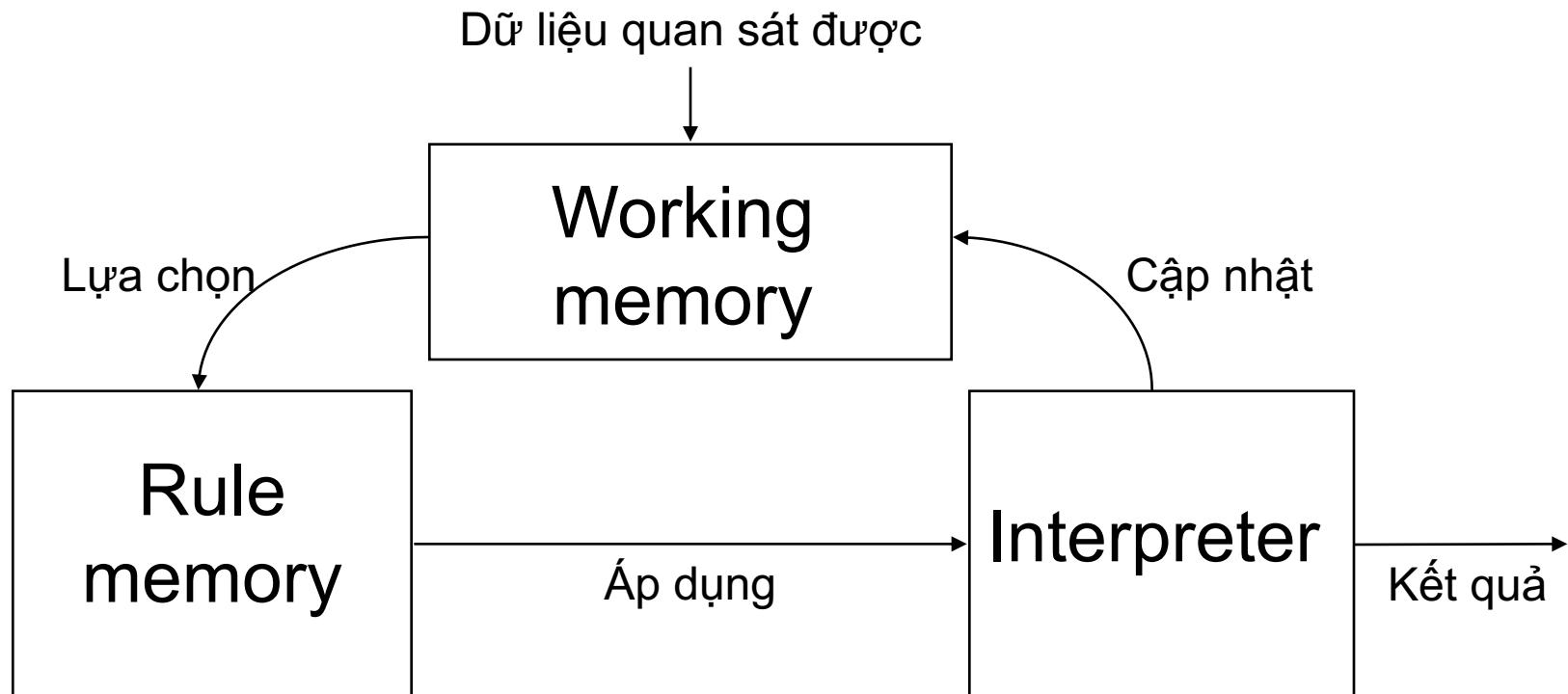
- Một xung đột (conflict) xảy ra khi có nhiều hơn một luật có thể áp dụng được (phù hợp với các sự kiện trong bộ nhớ làm việc)
 - Lưu ý, một xung đột không phải là một mâu thuẫn của tập luật
- Trong trường hợp xảy ra xung đột, cần một chiến lược giải quyết xung đột (conflict resolution strategy - CRS) để quyết định luật nào được (ưu tiên) áp dụng
- Sự lựa chọn thích hợp một chiến lược giải quyết xung đột có thể mang lại những cải thiện đáng kể đối với quá trình suy diễn của hệ thống

Chiến lược giải quyết xung đột

- Áp dụng luật **xuất hiện đầu tiên** (theo thứ tự) trong cơ sở tri thức
- Không áp dụng các luật sinh ra các kết quả (sự kiện) đã có trong bộ nhớ làm việc
- Áp dụng **luật cụ thể nhất** (luật có nhiều điều kiện nhất)
- Áp dụng các luật phù hợp với các sự kiện được đưa vào trong bộ nhớ làm việc **gần thời điểm hiện tại nhất**
- Không áp dụng lại một luật, nếu nó vẫn sinh ra cùng một tập các sự kiện (giống như lần áp dụng trước của nó)
- Áp dụng **luật có độ tin cậy** (chắc chắn) cao nhất
- ...
- *Kết hợp* của các chiến lược trên

Hệ thống suy diễn dựa trên luật (1)

Kiến trúc điển hình của một hệ thống suy diễn dựa trên luật (Rule-based system – RBS)



(<http://www.cwa.mdx.ac.uk/bis2040/johnlect.html>)

Hệ thống suy diễn dựa trên luật (2)

■ Bộ nhớ làm việc (Working memory)

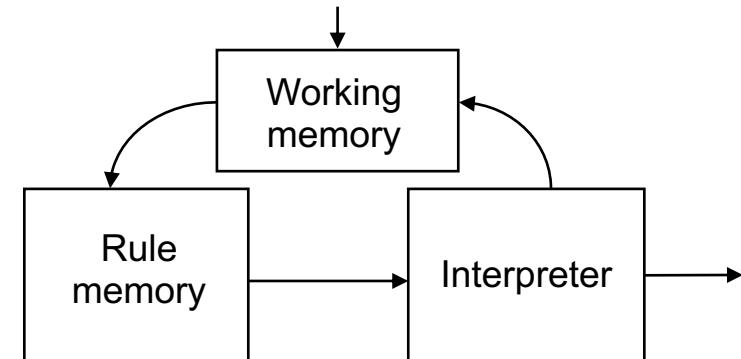
- ❑ Lưu giữ các sự kiện (các giả thiết đúng, đã được chứng minh)
- ❑ Các sự kiện này sẽ quyết định những luật nào được áp dụng (bởi thành phần Interpreter)

■ Bộ nhớ các luật (Rule memory)

- ❑ Chính là cơ sở tri thức của hệ thống
- ❑ Lưu giữ các luật có thể áp dụng

■ Bộ diễn dịch (Interpreter)

- ❑ Hệ thống bắt đầu bằng việc đưa một sự kiện (dữ liệu) phù hợp vào bộ nhớ làm việc
- ❑ Khi sự kiện (dữ liệu) trong bộ nhớ làm việc phù hợp với các điều kiện của một luật trong bộ nhớ các luật, thì luật đó sẽ được áp dụng



RBS: Ưu điểm (1)

- Cách biểu diễn (diễn đạt) phù hợp
 - Rất gần với cách diễn đạt trong ngôn ngữ tự nhiên
 - Rất dễ dàng để diễn đạt nhiều tri thức bởi các luật
- Dễ hiểu
 - Các luật dạng IF-THEN rất dễ hiểu đối với người sử dụng
 - Trong một lĩnh vực (bài toán) cụ thể, cách biểu diễn bằng luật giúp các chuyên gia trong lĩnh vực này có thể đánh giá và cải tiến các luật

RBS: Ưu điểm (2)

■ Một cách biểu diễn tri thức theo kiểu khai báo (declarative)

- ❑ Thu thập các tri thức (ở dạng các luật IF-THEN) về một lĩnh vực cụ thể, và đưa chúng vào trong một cơ sở các luật (rule base)
- ❑ (Có thể) không cần phải quan tâm đến khi nào, làm thế nào, và theo trật tự nào mà các luật được sử dụng – Hệ thống sẽ tự động đảm nhận các nhiệm vụ này

RBS: Nhược điểm

- Khả năng biểu diễn (điễn đạt) bị giới hạn
 - Trong nhiều lĩnh vực bài toán thực tế, tri thức của lĩnh vực bài toán đó không phù hợp với cách biểu diễn dạng (IF-THEN)
- Sự tương tác giữa các luật và trật tự của các luật trong cơ sở luật có thể gây ra các hiệu ứng không mong muốn
 - Trong quá trình thiết kế (design) và bảo trì (maintenance) một cơ sở luật, mỗi luật mới được đưa vào cần phải được cân nhắc (kiểm tra) với các luật đã có từ trước
 - Rất khó khăn và chi phí tốn kém để xem xét tất cả các tương tác (interactions) có thể giữa các luật
- Quá tốn kém chi phí: thời gian, công sức, tiền bạc

Biểu diễn tri thức: bằng khung (1)

- Làm thế nào để biểu diễn tri thức “Xe buýt màu blue”?
- Giải pháp thứ 1. Blue (bus)
 - ❑ Câu hỏi “Cái gì là màu blue?” có thể trả lời được
 - ❑ Nhưng câu hỏi “Màu của xe buýt là gì?” thì không thể trả lời được
- Giải pháp thứ 2. Color (bus, blue)
 - ❑ Câu hỏi “Cái gì có màu blue?” có thể trả lời được
 - ❑ Câu hỏi “Màu của xe buýt là gì?” có thể trả lời được
 - ❑ Nhưng câu hỏi “Thuộc tính nào của xe buýt có giá trị là màu blue?” thì không thể trả lời được
- Giải pháp thứ 3. Prop (bus, color, blue)
 - ❑ Tất cả 3 câu hỏi trên đều có thể trả lời được

Biểu diễn tri thức: bằng khung (2)

- Một đối tượng được biểu diễn bởi:
(Object, Property, Value)
 - Được gọi là cách biểu diễn bằng bộ ba
đối tượng-thuộc tính-giá trị (*object-property-value*)

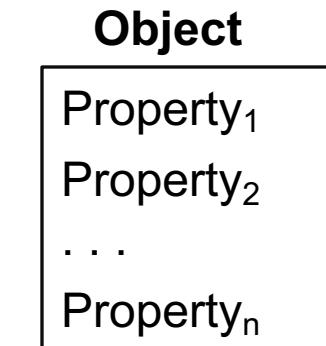
- Nếu chúng ta gộp nhiều thuộc tính của cùng một kiểu đối tượng thành một cấu trúc, thì chúng ta có cách biểu diễn hướng đối tượng (object-centered representation)

$Prop(Object, Property_1, Value_1)$

$Prop(Object, Property_2, Value_2)$

...

$Prop(Object, Property_n, Value_n)$



Biểu diễn hướng đối tượng

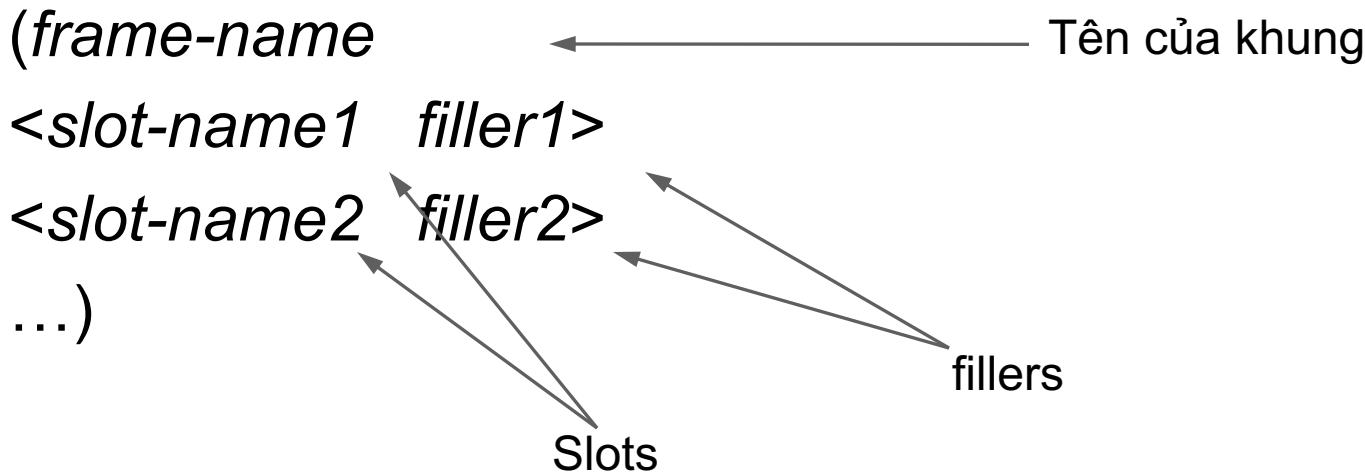
- Cách biểu diễn bằng bộ ba *object-property-value* là một cách tự nhiên để biểu diễn các đối tượng
- Các đối tượng cụ thể (Physical objects)
 - Ví dụ: Một *cái bàn* có các thuộc tính chất liệu bề mặt, số ngăn kéo, độ rộng, độ dài, độ cao, màu sắc, ...
- Các tình huống (Situations)
 - Ví dụ: Một *lớp học* có các thuộc tính mã số phòng học, danh sách sinh viên tham dự, giáo viên, ngày học, thời gian học, ...
 - Ví dụ: Một *chuyến đi nghỉ mát* có các thuộc tính nơi khởi hành, nơi đến, phương tiện di chuyển, phòng nghỉ, ...

Khung (Frame)

- Có 2 kiểu khung: cụ thể (individual) và tổng quát (generic)
- **Khung cụ thể (Individual frames).** Để biểu diễn một đối tượng cụ thể, chẳng hạn như một người cụ thể, một chuyến đi nghỉ mát cụ thể, ...
- **Khung tổng quát (Generic frames).** Để biểu diễn một lớp (loại) các đối tượng, chẳng hạn như các sinh viên, các chuyến đi nghỉ mát, ...
- Ví dụ
 - Khung tổng quát: European_City
 - Khung cụ thể: City_Paris

Biểu diễn của một khung

- Một khung được biểu diễn bằng một danh sách định danh các thuộc tính được gọi là các **slots**
- Giá trị gán cho một thuộc tính được gọi là **filler of the slot**



Khung đơn

- Một khung đơn (individual frame) có một thuộc tính đặc biệt có tên là **INSTANCE-OF**, và giá trị của thuộc tính đặc biệt này là tên của một khung tổng quát (generic frame)

- Ví dụ

(toronto % sử dụng chữ thường cho các khung đơn
<:INSTANCE-OF CanadianCity>
<:Province ontario>
<:Population 4.5M>
...)

Khung tổng quát

- Một khung tổng quát (generic frames) có thể có một thuộc tính đặc biệt là **IS-A** mà giá trị của thuộc tính đặc biệt này là tên của một khung tổng quát khác
- Ví dụ

(CanadianCity % sử dụng tên bắt đầu chữ hoa cho khung tổng quát
<:IS-A City>
<:Province CanadianProvince>
<:Country canada>
...)

Suy diễn với khung (1)

- Các thuộc tính (slots) trong các khung tổng quát có thể được gắn (liên kết) với các thủ tục để thực hiện và điều khiển việc suy diễn
- Có 2 kiểu thủ tục: **IF-NEEDED** và **IF-ADDED**
- Thủ tục **IF-NEEDED**
 - Được thực hiện khi không có giá trị cần thiết được gán cho một thuộc tính (no slot filler)
 - Ví dụ: (Table
 - <:Clearance [**IF-NEEDED** computeClearance]>
 - ...)

computeClearance là một thủ tục để tính toán (xác định) mức độ sạch sẽ của cái bàn

Suy diễn với khung (2)

■ Thủ tục **IF-ADDED**

- ❑ Được thực hiện khi một thuộc tính được gán giá trị, để cho phép *lan truyền ảnh hưởng của việc gán giá trị của thuộc tính đó đối với các khung khác* (ví dụ, để đảm bảo các ràng buộc trong bài toán)

- ❑ Ví dụ: (Lecture

```
<:DayOfWeek WeekDay>
```

```
<:Date [IF-ADDED computeDayOfWeek]>
```

```
...)
```

Giá trị của thuộc tính *:DayOfWeek* sẽ được tính toán (lại) khi thuộc tính *:Date* được gán giá trị

Khung: Giá trị mặc định (1)

- Hãy xét khung tổng quát (generic frame) sau đây

(CanadianCity

<:IS-A City>

<:Province CanadianProvince>

<:Country canada>

...)

- Hãy xét khung cụ thể (individual frame) sau đây

(city134

<:INSTANCE-OF CanadianCity>

...)

- Đối với khung city134, giá trị (mặc định) cho thuộc tính :Country là canada

Khung: Giá trị mặc định (2)

- Hãy xét khung cụ thể (individual frame) sau đây
(city135
<:INSTANCE-OF CanadianCity>
<:Country holland>
...)
- Đối với khung city135, thì giá trị cho thuộc tính :Country là holland (chứ không phải là giá trị mặc định canada)

Khung: Suy diễn

- Quá trình suy diễn trong phương pháp biểu diễn bằng khung sẽ diễn ra như sau
 - 1. Người dùng khởi tạo một khung (tương đương với việc khai báo sự tồn tại của một đối tượng hay một tình huống)
 - 2. Các giá trị của các thuộc tính sẽ được kế thừa (từ các khung tổng quát hơn)
 - 3. Các thủ tục IF-ADDED sẽ được thực hiện. Việc này có thể sẽ dẫn đến việc khởi tạo của các khung khác, và việc gán giá trị của các thuộc tính
- Nếu người dùng hoặc một thủ tục yêu cầu việc gán giá trị cho một thuộc tính, thì:
 - ❑ Nếu có giá trị cho thuộc tính, thì giá trị đó sẽ được gán
 - ❑ Nếu không, thủ tục IF-NEEDED sẽ được thực hiện

Biểu diễn bằng khung: Ưu điểm

- Kết hợp được cả tri thức khai báo (declarative knowledge) và tri thức thủ tục (procedural knowledge) trong cùng một phương pháp biểu diễn
- Các khung được tổ chức có cấu trúc phân cấp, cho phép dễ dàng phân loại (phân lớp) tri thức
- Cấu trúc phân cấp các khung cho phép giảm bớt sự phức tạp (và chi phí) trong quá trình xây dựng cơ sở tri thức
- Cho phép thiết lập các ràng buộc đối với các giá trị được gán cho các thuộc tính (ví dụ: ràng buộc giá trị nhập vào phải nằm trong một khoảng giá trị cụ thể)
- Cho phép lưu giữ các giá trị mặc định (sử dụng thuộc tính đặc biệt IS-A, các giá trị của các thuộc tính của một khung tổng quát hơn được sử dụng để gán cho các thuộc tính của một khung cụ thể hơn)

Biểu diễn bằng khung: Nhược điểm

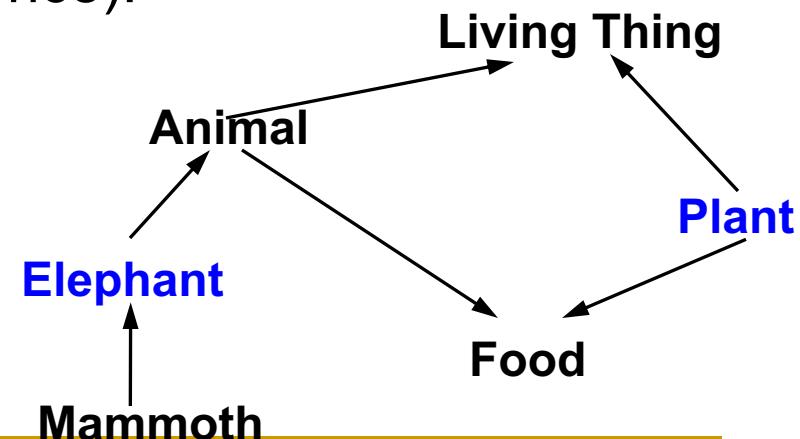
- Trong quá trình thiết kế cấu trúc phân cấp của các khung, cần rất để ý đến sự hợp lý của việc phân loại (các khung)
- Có thể gặp vấn đề chi phí cao cho việc thiết kế các thủ tục (IF-ADDED và IF-NEEDED) – Quá nhiều công sức dành cho việc thiết kế các thủ tục phù hợp, thay vì tập trung vào việc kiểm tra cấu trúc và nội dung của các khung
- Quá trình khai thác các khung có thể không hiệu quả, vì không có phương pháp hiệu quả để lưu trữ dữ liệu (của các khung) trong máy tính

Mạng ngữ nghĩa (1)

- **Mạng ngữ nghĩa (Semantic Network)** được đề cử bởi Quillian vào năm 1966 như là một mô hình biểu diễn bộ nhớ của con người
- Các động cơ thúc đẩy sự phát triển của mạng ngữ nghĩa
 - Để hiểu về cấu trúc của bộ nhớ con người, và việc sử dụng cấu trúc bộ nhớ này trong xử lý (hiểu) ngôn ngữ
 - Kiểu biểu diễn nào cho phép lưu giữ các ý nghĩa (meanings) của các từ để có thể sử dụng lại các ngữ nghĩa này (như trong bộ nhớ con người)?
 - Các chứng minh về tâm lý học đã chỉ ra rằng bộ nhớ của con người sử dụng các liên kết trong việc xử lý (hiểu) các từ
- **Yêu cầu:** cần biểu diễn định nghĩa từ điển của các từ, để
 - So sánh và phân biệt ý nghĩa của 2 từ
 - Sinh ra các câu “tựa” (gần giống) tiếng Anh để mô tả sự so sánh này

Mạng ngữ nghĩa (2)

- Mạng ngữ nghĩa (Semantic Network) là phương pháp biểu diễn ngữ nghĩa dựa trên đồ thị (graph-based representation)
- Một mạng ngữ nghĩa bao gồm một tập **các nút (nodes)** và **các liên kết (links)**
 - Các nút biểu diễn các khái niệm
 - Các liên kết biểu diễn các mối quan hệ giữa các khái niệm
- Quá trình suy diễn (reasoning/inference):
được thực hiện thông qua cơ chế
lai truyền
 - Tác động (Activation)
 - Kế thừa (Inheritance)

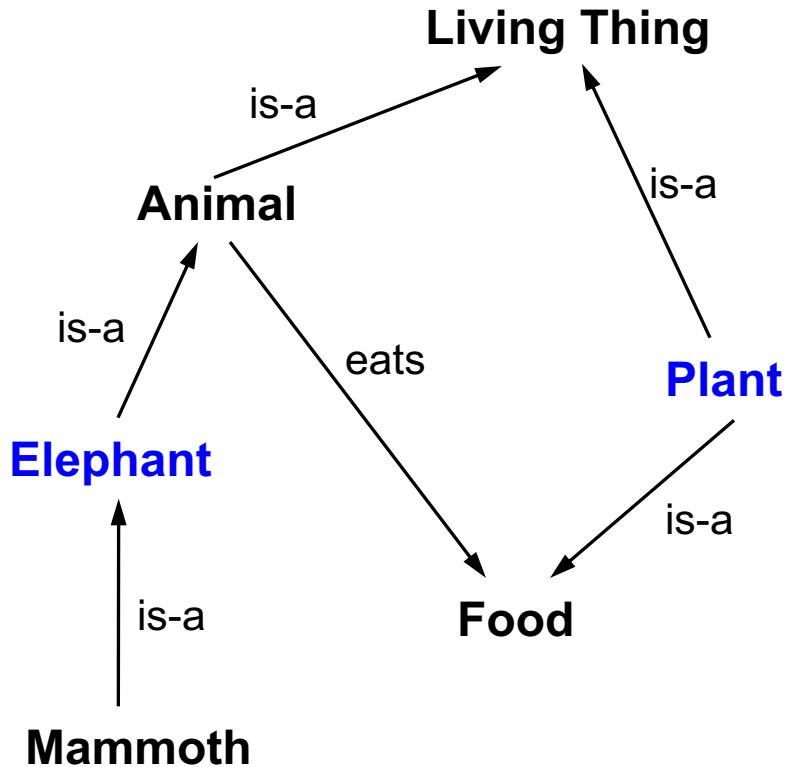


Mạng ngữ nghĩa: Cú pháp

- **Các nút (nodes)** biểu diễn các *khái niệm (concepts)*, *hành động (actions)* hoặc *đối tượng (objects)* trong lĩnh vực bài toán đang xét
- **Các liên kết (links)** là các quan hệ được gán nhãn và có chiều (directional and labeled) giữa các nút
- Hai kiểu liên kết: **kế thừa** và **cụ thể**
- **Liên kết kế thừa (Inheritance-oriented link)** biểu diễn:
 - Nút A là **một lớp (loại) con** của nút B (vd: liên kết *IS-A*)
 - Nút A là **một ví dụ (instance)** của nút B (vd: liên kết *INSTANCE-OF*)
- **Liên kết cụ thể (Domain-specific link)** biểu diễn:
 - Nút A liên quan tới (có quan hệ với) nút B
 - Ví dụ: *HAS, CAN, HAS-PART, CAUSES, HAS-COLOR, ...*

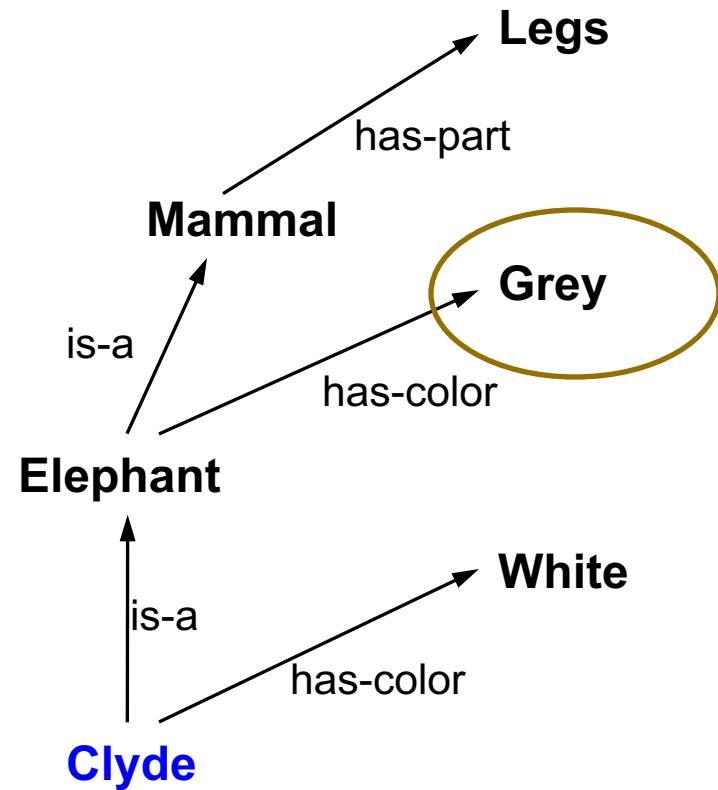
SN: Lan truyền tác động

- Đối với 2 khái niệm, việc lan truyền tác động (spreading activation) sẽ kích hoạt tác động từ khái niệm này tới khái niệm kia, hoặc theo cả 2 hướng
- Cho phép xác định các khái niệm “nằm giữa” liên quan đến cả 2 khái niệm đó
 - Ví dụ: Xét việc lan truyền tác động giữa 2 khái niệm “Elephant” và “Plant”



SN: Tính kế thừa

- Các thuộc tính (properties) của lớp (loại) cha được kế thừa cho các lớp (loại) con
- **Kế thừa toàn bộ (Universal inheritance):** Tất cả các quan hệ được kế thừa
- **Kế thừa mặc định (Default inheritance):** Các quan hệ được kế thừa, trừ khi có các thông tin mâu thuẫn (với nút cha) ở một nút con



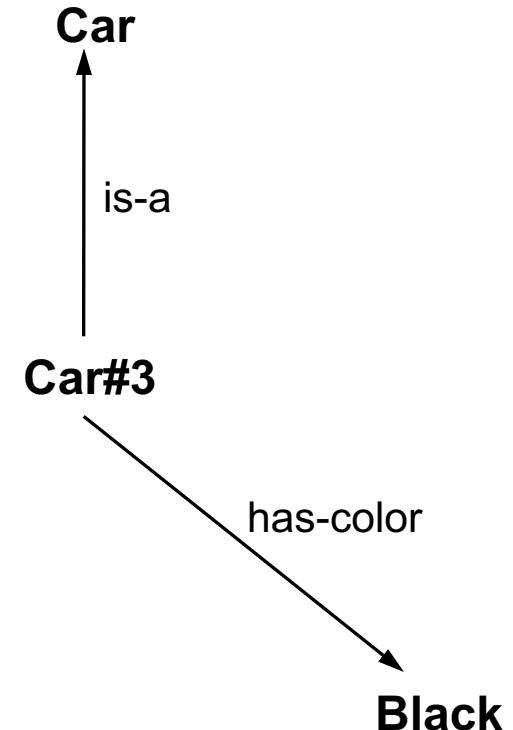
SN: Ngữ nghĩa (1)

- Biểu diễn bằng mạng ngữ nghĩa rất trực quan, và gần với nhận thức (cách biểu diễn) của con người
- Nhưng: Đối với *cùng một đồ thị (mạng) ngữ nghĩa*, các hệ thống khác nhau có thể có *các cách diễn giải (interpretations) khác nhau*
- Ngữ nghĩa (semantics) của các mạng ngữ nghĩa?
 - "Since the semantics of any given language is dependent of the interpretation of the primitive elements....., the well-definedness of a network language rests heavily on the set of node and link types that it provides" (Brachman, p204, Readings in KR)

SN: Ngữ nghĩa (2)

■ Ý nghĩa của mạng ngữ nghĩa sau là gì?

- Thể hiện định nghĩa của một cái ô-tô màu đen (**Thông tin định nghĩa**)
- Thể hiện rằng tồn tại một ô-tô màu đen (**Thông tin xác nhận**)
- Thể hiện rằng một cái ô-tô cụ thể (Car#3) là màu đen (**Xác nhận sự tồn tại**)



SN: Ngữ nghĩa (3)

- Các liên kết có thể là
 - **Liên kết xác nhận (Assertional links)**
 - Lưu giữ các thông tin về không gian bài toán đang xét
 - *Có thể thay đổi khi không gian bài toán thay đổi*
 - Ví dụ: “John hit Mary” (một sự kiện cụ thể đã xảy ra)
 - **Liên kết định nghĩa (Definitional links)**
 - Lưu giữ các ý nghĩa của các khái niệm
 - *Không thay đổi khi không gian bài toán thay đổi*
 - Ví dụ: “apple is-a fruit”, “apple has-color red”

Mạng ngữ nghĩa: Ưu điểm

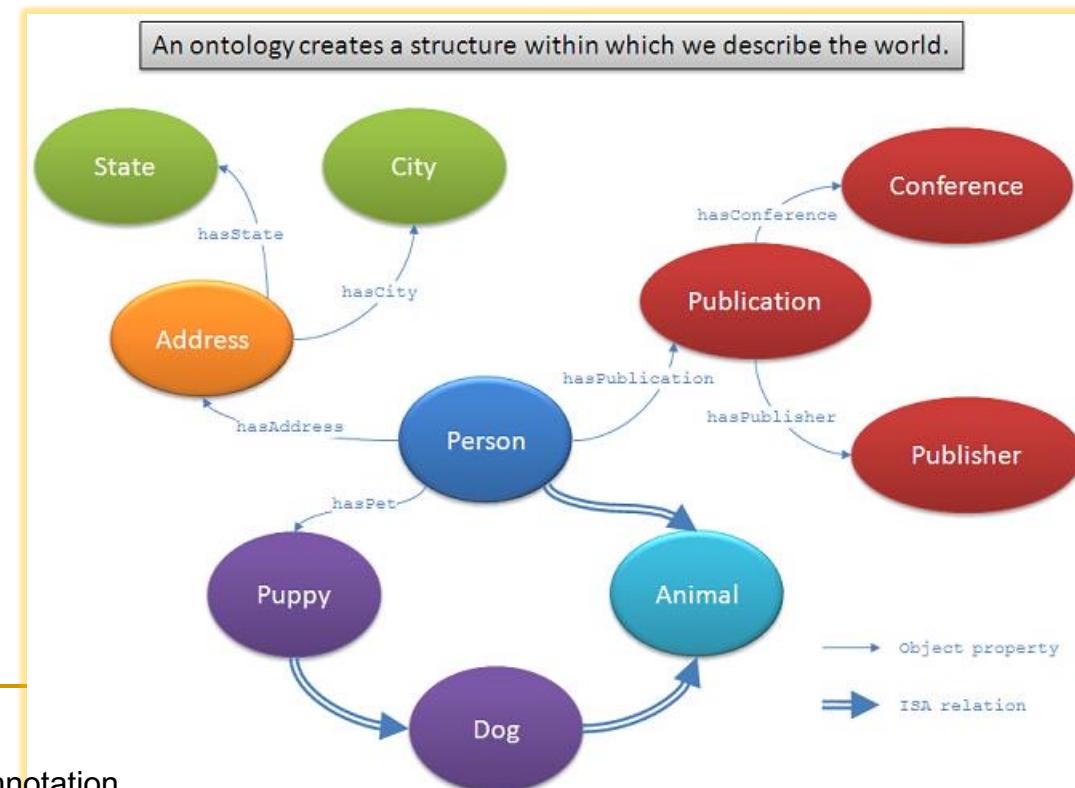
- Rõ ràng (trực quan) trong hiển thị, dễ hiểu đối với người dùng
 - SNs thường được sử dụng như là một công cụ trao đổi (làm việc) giữa các kỹ sư tri thức (knowledge engineers) và các chuyên gia (experts) trong giai đoạn thu thập tri thức
- SNs là rất phù hợp đối với các bài toán biểu diễn tri thức ở dạng phân cấp các khái niệm
 - Tri thức được phân loại (phân lớp) thành một cấu trúc phân cấp
- Cơ chế biểu diễn phân cấp của SNs hỗ trợ quá trình suy diễn nhanh chóng
- Tập trung vào các thành phần chính của tri thức và liên kết giữa chúng

Mạng ngữ nghĩa: Nhược điểm

- Không tồn tại cách diễn dịch (interpretation) chung (chuẩn) – Ngữ nghĩa của các mạng ngữ nghĩa không được định nghĩa một cách chuẩn tắc
- Gặp vấn đề trong việc biểu diễn các thông tin phủ định (negation) và tuyễn (disjunction)
 - Vd: “John **does not** go fishing”, “John eats pizza **or** fish and chips”
- Khó khăn trong việc chọn các thành phần cơ bản (primitives) phù hợp
- Khả năng suy diễn hạn chế
- Tốn kém (công sức, thời gian, tiền bạc) khi xây dựng

Ontology

- Một **ontology** là một đặc tả (biểu diễn) hình thức và rõ ràng về các khái niệm
- Một **ontology** là một từ vựng dùng chung, được dùng để biểu diễn (mô hình) một lĩnh vực cụ thể
 - ❑ Các đối tượng và/hoặc các khái niệm
 - ❑ Các thuộc tính và các quan hệ của chúng



Ontology (2)

- Một ontology có thể được xem như là một cơ sở tri thức
- Một ontology có thể phục vụ các mục đích khác nhau
 - Ví dụ: Mô hình cơ sở dữ liệu (database schema) là một ontology
- Cơ sở tri thức chứa đựng các tri thức cụ thể cần thiết cho việc giải quyết vấn đề của một lĩnh vực

Ontology: Các động cơ thúc đẩy

- Các động cơ thúc đẩy về mặt công nghệ
 - Rất nhiều các hệ thống dựa trên tri thức sử dụng một ontology mô tả các tri thức của lĩnh vực liên quan
 - Việc xây dựng ontology là một nhiệm vụ rất tốn kém (về thời gian và công sức) trong quá trình phát triển một hệ thống dựa trên tri thức
 - Tại sao không giảm chi phí bằng cách chia sẻ các ontology?
 - Ví dụ: Các ontology “cơ bản” mô tả không gian, thời gian, số lượng, ...
- Động cơ thúc đẩy về mặt khoa học
 - Để hiểu được các vấn đề nền tảng trong quá trình khái niệm hóa (nhận thức các khái niệm) của con người

Các khía cạnh của một ontology (1)

■ Nội dung của ontology

- ❑ Các kiểu đối tượng, các kiểu quan hệ
- ❑ Ví dụ: Bài toán sắp xếp các khối (blocks)
 - Các lớp đối tượng: Blocks, Robot Hands
 - Các thuộc tính: shapes of blocks, color of blocks
 - Các quan hệ: On, Above, Below, Grasp
 - Các quá trình: thiết kế hoặc xây nên một tòa tháp

■ Kiểu của ontology

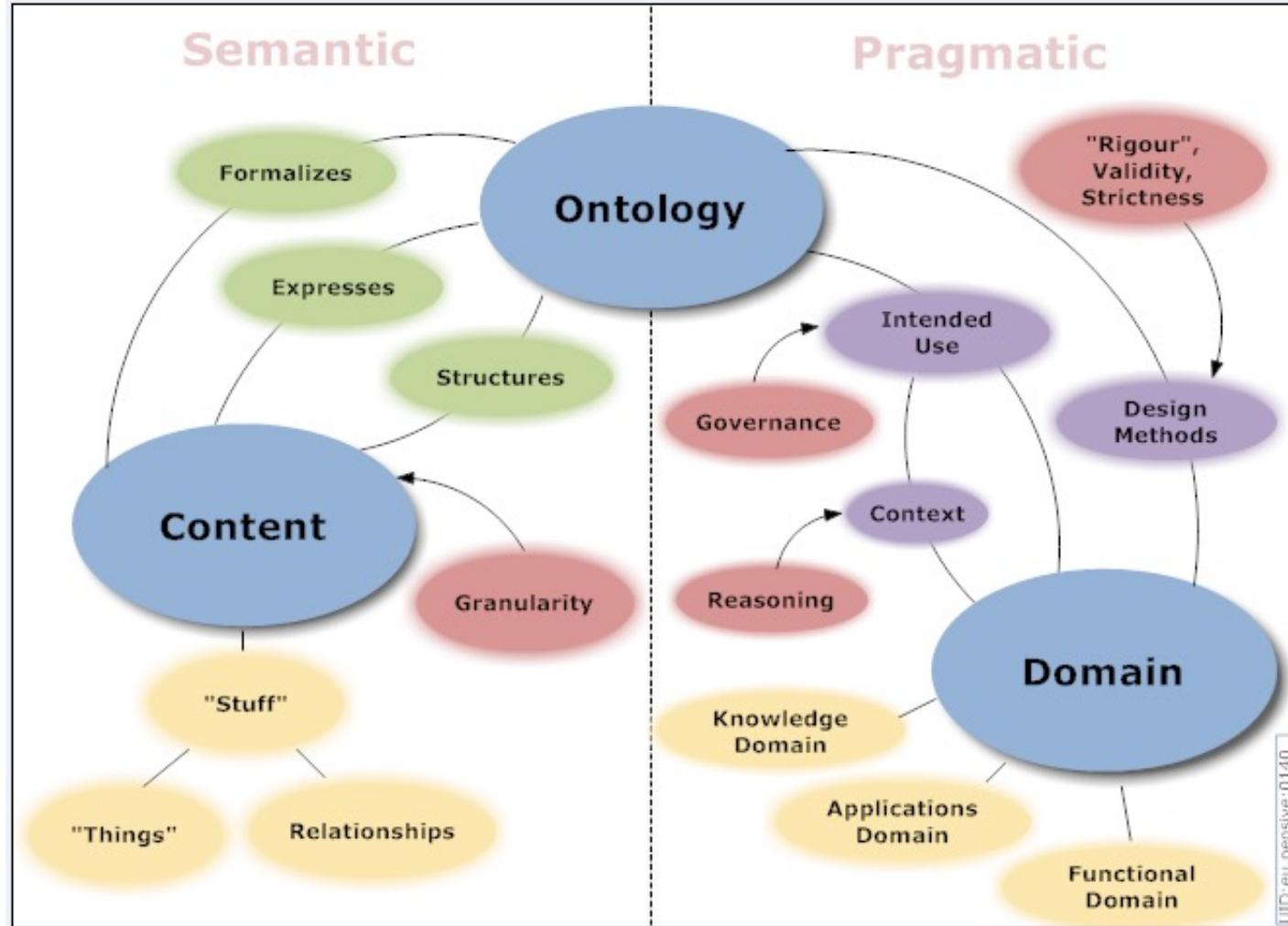
- ❑ Các quan hệ phân loại cơ bản (vd: instance-of, subclass) là gì?
- ❑ Các định nghĩa của các khái niệm (và các ràng buộc đối với chúng)?
- ❑ Khả năng diễn đạt của ngôn ngữ định nghĩa các khái niệm?
- ❑ Hướng quá trình (process-centric) hay hướng đối tượng (object-centric)?

Các khía cạnh của một ontology (2)

- Mục đích sử dụng của ontology
 - Chia sẻ tri thức
 - Ví dụ: Giữa những người sử dụng, giữa các hệ thống, ...
 - Sử dụng lại tri thức
 - Ví dụ: Sử dụng lại (một phần) tri thức khi các mô hình hoặc hệ thống thay đổi
- Việc xây dựng ontology
 - Ontology sẽ được thu thập hay xây dựng?
 - Nếu được thu thập, thì cần kiểm tra:
 - Chất lượng của tri thức?
 - Sự khác biệt về nội dung?
 - Sự tin cậy của tri thức?
 - Các khả năng sử dụng trong tương lai?

Ontology

(Lý
thuyết
cần
xem
xét)



Source: http://www.emiliosanfilippo.it/?page_id=1172

Xây dựng ontology: các vấn đề

- Xác định giới hạn (phạm vi) và mục đích sử dụng
- Cân nhắc việc sử dụng lại các ontology đã có và liên quan đến lĩnh vực đang xét
- Liệt kê các khái niệm
- Định nghĩa sự phân loại
- Định nghĩa các thuộc tính
- Định nghĩa các khía cạnh (các ràng buộc)
- Xác định các ví dụ cụ thể (instances)
- Kiểm tra các bất thường (anomalies)

Phạm vi và mục đích sử dụng

- Không tồn tại một ontology “chuẩn” cho một lĩnh vực cụ thể
 - Một ontology là một sự khái quát hóa về một lĩnh vực cụ thể, và luôn tồn tại nhiều ontologies phù hợp
- Việc khái quát hóa này nên tính đến:
 - Việc sử dụng trong tương lai của ontology (mục đích sử dụng)
 - Các khả năng mở rộng (của ontology) có thể dự đoán trước
- Các câu hỏi cần phải được trả lời ở giai đoạn này:
 - Ontology này được dùng cho lĩnh vực nào?
 - Ontology này sẽ được dùng để làm gì?
 - Những kiểu câu hỏi nào mà ontology này có thể đưa ra câu trả lời?
 - Ai sẽ dùng và bảo trì ontology này?

Cân nhắc sử dụng lại các ontology

- Với sự phát triển rất nhanh chóng của Internet và của lĩnh vực Semantic Web, sẽ tồn tại (có) rất nhiều ontologies có thể khai thác

Liệt kê các khái niệm

- Xác định (theo một danh sách) tất cả các khái niệm liên quan sẽ xuất hiện trong ontology
 - Các danh từ (nouns) thường là cơ sở để xác định các tên lớp (class names)
 - Các động từ (verbs) hoặc cụm động từ (verb phrases) thường là cơ sở để xác định các tên thuộc tính (property names)
- Nên sử dụng các công cụ xây dựng tri thức, để thu được
 - Tập các khái niệm
 - Cấu trúc (phân cấp) ban đầu của các khái niệm này

Định nghĩa sự phân loại

- Các khái niệm liên quan cần được tổ chức lại với nhau thành một cấu trúc phân cấp phân loại (a taxonomic hierarchy)
 - Hai chiến lược thường được dùng: top-down vs. bottom-up
 - Cần cân nhắc đến hai yếu tố: sự tin cậy (chính xác) và hiệu quả (suy diễn)
- Đảm bảo rằng phân cấp thu được thực sự là một phân loại khái niệm
 - Nếu A là một lớp con của B, thì mọi ví dụ của A cũng phải là ví dụ của B

Định nghĩa các thuộc tính

- Thường được thực hiện kết hợp với bước trước (định nghĩa sự phân loại)
- Yêu cầu: Nếu A là một lớp con của B , thì mọi thuộc tính của B đều phải được áp dụng cho A
 - Nên gắn các thuộc tính với lớp cao nhất (phù hợp) trong cấu trúc phân cấp
- Khi xác định một thuộc tính cho một lớp, cần ngay lập tức xác định các thông tin về miền (data type) và giá trị (value domain) của thuộc tính đó

Định nghĩa các ràng buộc

- Các ràng buộc về số chiều (số phần tử của một tập hợp)
- Các giá trị có thể được gán cho
- Các đặc điểm của quan hệ
 - Tính đối xứng, bắc cầu, đảo, ...

Template Slots				
Name	Type	Cardinality	Other Facets	
S body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}	
S color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}	
S flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}	
S grape	Instance	multiple	classes={Wine grape}	
S maker	Instance	single	classes={Winery}	
S name	String	single		
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}	

(protege.stanford.edu/amia2003/AMIA2003Tutorial.ppt)

Xác định các ví dụ

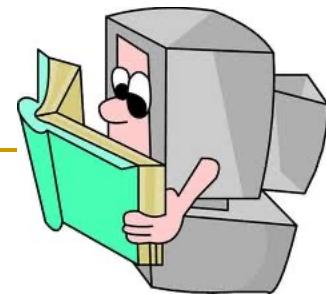
- Bổ sung vào ontology các ví dụ (instances) cụ thể
- Số lượng các ví dụ thường lớn hơn (nhiều) số lượng các lớp (classes)
- Việc bổ sung các ví dụ vào một ontology thường được làm thủ công (manually)
 - Lấy từ các nguồn dữ liệu sẵn có (vd: từ các cơ sở dữ liệu)
 - Trích tự động từ một tập các dữ liệu văn bản

Kiểm tra các bất thường

- Để kiểm tra các mâu thuẫn có thể có trong một ontology, (hoặc có trong một ontology đi kèm với các ví dụ)
- Các ví dụ của các mâu thuẫn
 - Không tương thích về miền, về giá trị, về các đặc tính bắc cầu, đối xứng, đảo,...
 - Không tương thích với ràng buộc về số chiều của một thuộc tính

Một máy học

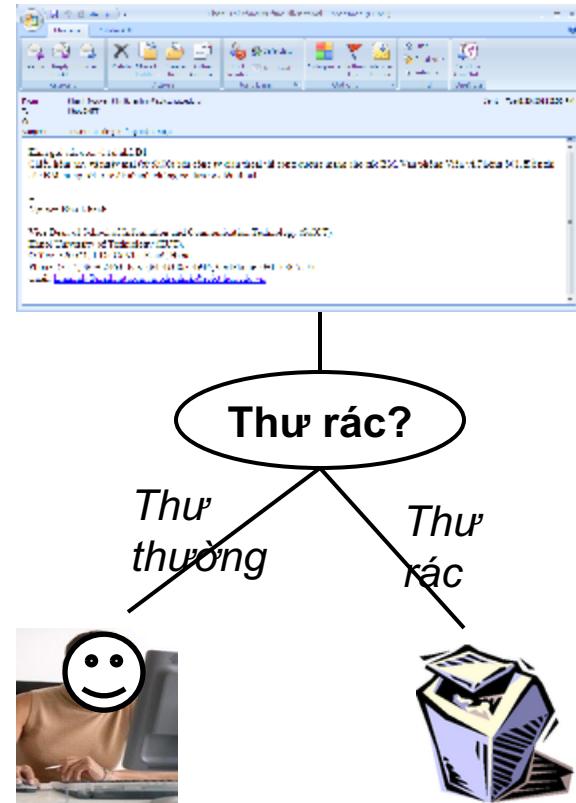
- Ta nói một máy tính *có khả năng học* nếu nó tự cải thiện hiệu suất hoạt động P cho một công việc T cụ thể, dựa vào kinh nghiệm E của nó.
- Như vậy *một bài toán học máy* có thể biểu diễn bằng 1 bộ (T , P , E)
 - T : một công việc (nhiệm vụ)
 - P : tiêu chí đánh giá hiệu năng
 - E : kinh nghiệm



Ví dụ bài toán học máy (1)

Lọc thư rác (email spam filtering)

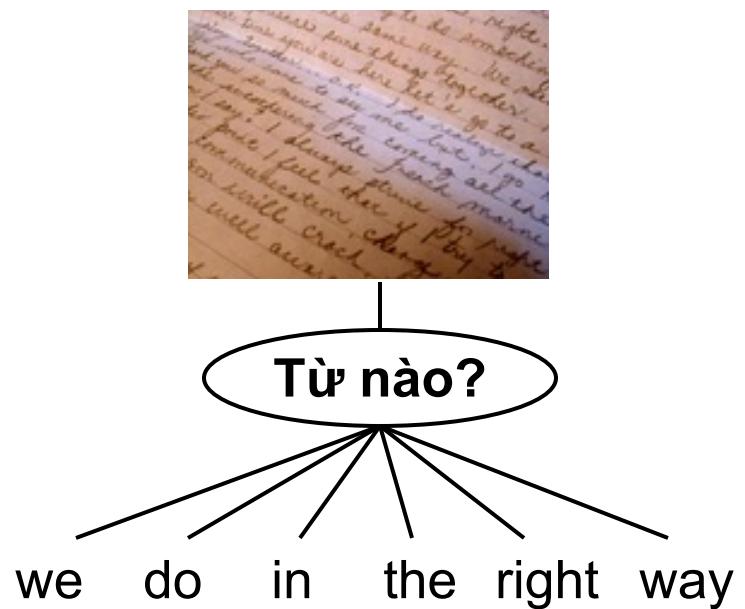
- T : Dự đoán (để lọc) những thư điện tử nào là thư rác (spam email)
- P : số lượng thư điện tử gửi đến được phân loại chính xác
- E : Một tập các thư điện tử (emails) mẫu, mỗi thư điện tử được biểu diễn bằng một tập thuộc tính (vd: tập từ khóa) và nhãn lớp (thư thường/thư rác) tương ứng



Ví dụ bài toán học máy (2)

Nhận dạng chữ viết tay

- **T**: Nhận dạng và phân loại các từ trong các ảnh chữ viết
- **P**: Tỷ lệ (%) các từ được nhận dạng và phân loại đúng
- **E**: Một tập các ảnh chữ viết, trong đó mỗi ảnh được gắn với một định danh của một từ



Ví dụ bài toán học máy (3)

Gán nhãn ảnh

- **T:** đưa ra một vài mô tả ý nghĩa của 1 bức ảnh
- **P:** ?
- **E:** Một tập các bức ảnh, trong đó mỗi ảnh đã được gán một tập các từ mô tả ý nghĩa của chúng



lychee-inspired spherical chair



a girl giving cat a gentle hug



a small hedgehog holding a piece of watermelon

Máy học (1)

■ Học một ánh xạ (hàm):

$$y^*: x \mapsto y$$

- x : quan sát (dữ liệu), kinh nghiệm
 - y : phán đoán, tri thức mới, kinh nghiệm mới, ...
- **Hồi quy** (regression): nếu y là một số thực
 - **Phân loại** (classification): nếu y thuộc một tập rời rạc (tập nhãn lớp)

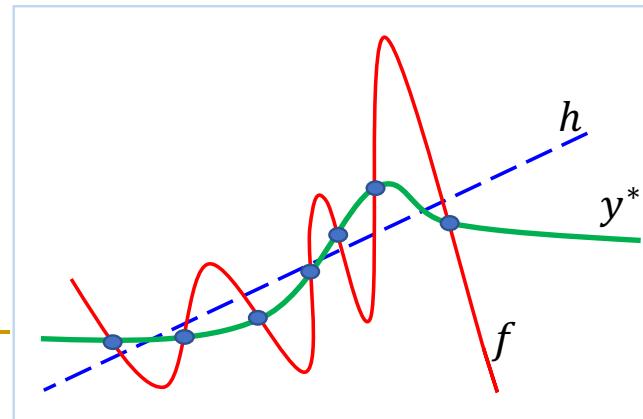
Máy học (2)

■ Học từ đâu?

- ❑ Từ các quan sát trong quá khứ (**tập học**).
 $\{x_1, x_2, \dots, x_N\}; \{y_1, y_2, \dots, y_M\}$

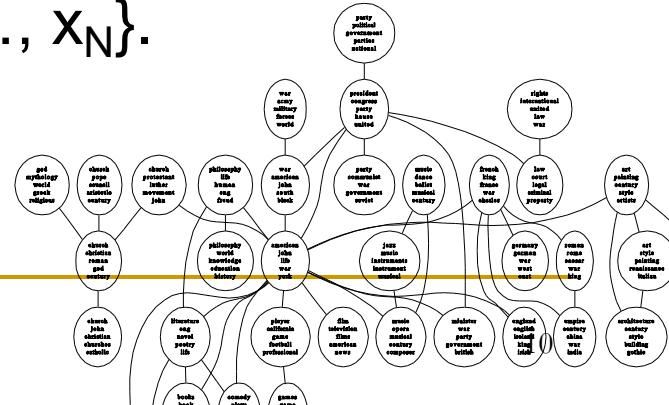
■ Sau khi đã học:

- ❑ Thu được một mô hình (f), kinh nghiệm, tri thức mới.
- ❑ Dùng nó để **suy diễn (phán đoán)** cho quan sát trong tương lai.
 $Y = f(x)$

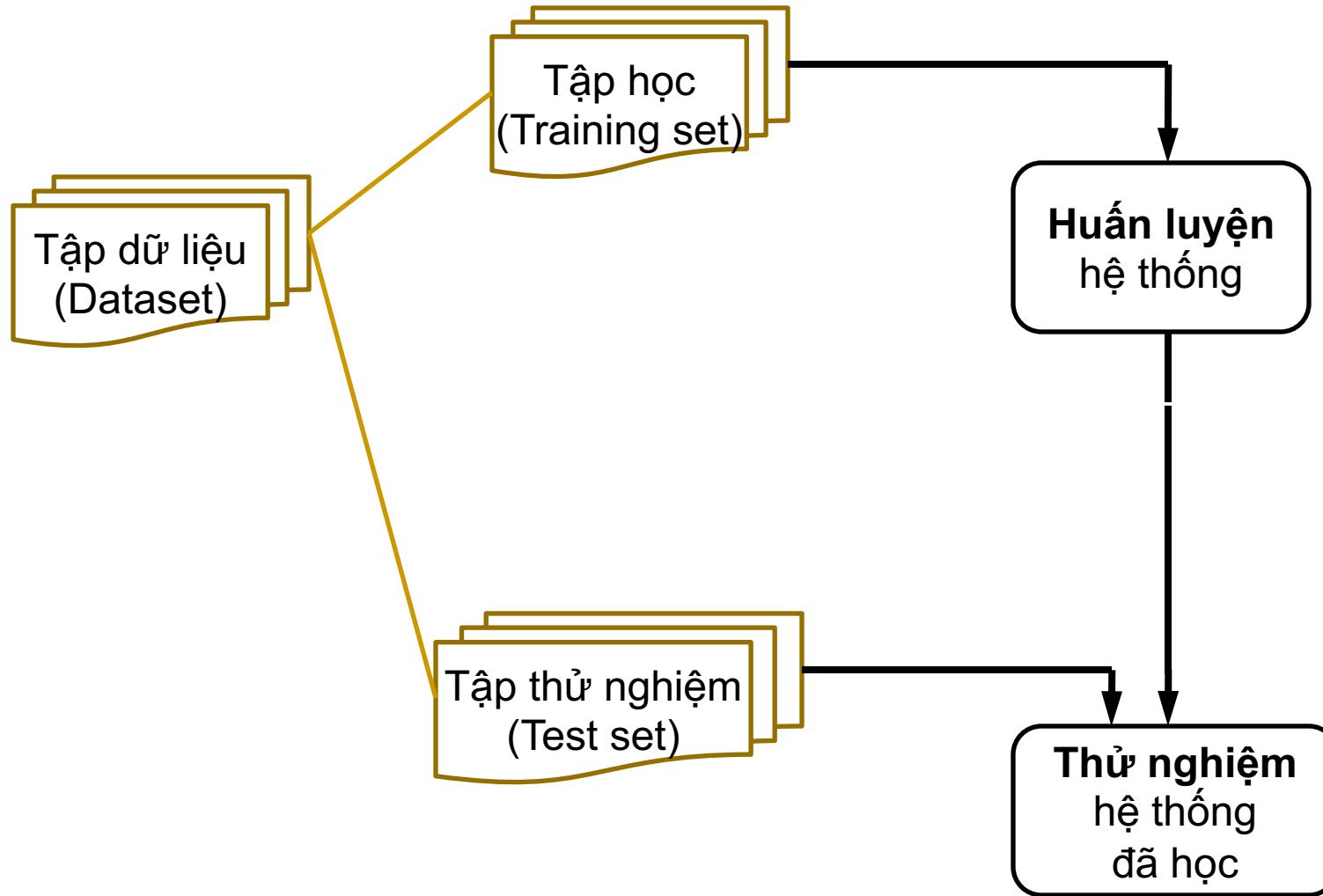


Hai bài toán học cơ bản

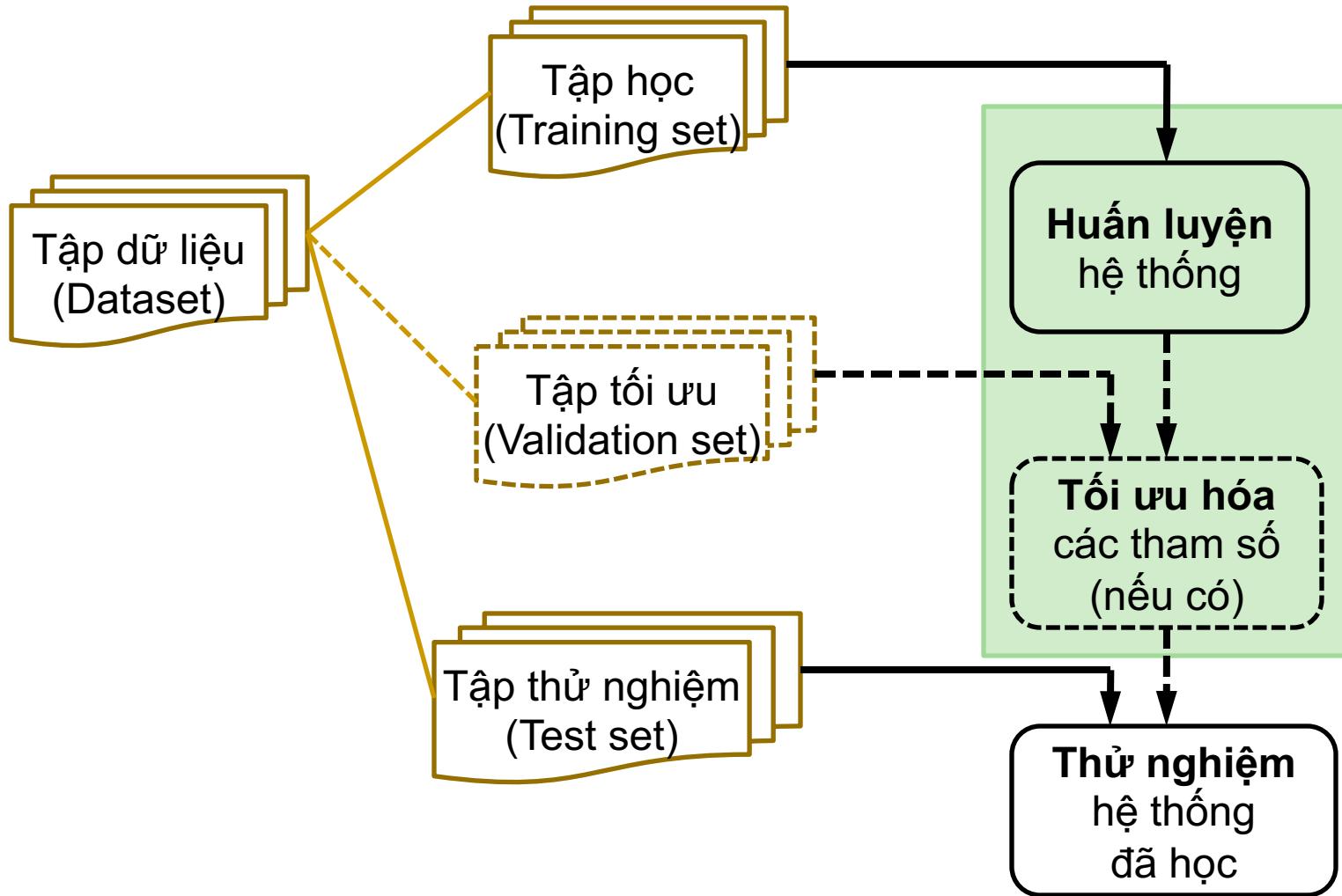
- Có một hàm y^* , kết nối mỗi đầu vào x tới một đầu ra y
 - ❑ Trong thực tế, ta có thể thu thập được các cặp (x_i, y_i) , với $y_i = y^*(x_i)$
- **Học có giám sát (supervised learning):** cần học một hàm y^* từ tập học $\{x_1, x_2, \dots, x_N; y_1, y_2, \dots, y_N\}$, trong đó $y_i \cong y^*(x_i)$.
 - ❑ *Phân loại* (phân lớp): nếu y chỉ nhận giá trị từ một tập rời rạc, chẳng hạn {cá, cây, quả, mèo}
 - ❑ *Hồi quy*: nếu y nhận giá trị số thực
- **Học không giám sát (unsupervised learning):** cần học một hàm y^* từ tập học cho trước $\{x_1, x_2, \dots, x_N\}$.
 - ❑ Y có thể là các cụm dữ liệu.
 - ❑ Y có thể là các cấu trúc ẩn.



Quá trình học máy: cơ bản



Quá trình học máy: toàn diện



Thiết kế một hệ thống học (1)

■ Xác định hàm mục tiêu (giả thiết, khái niệm) cần học

- $y^*: X \rightarrow \{0,1\}$
- $y^*: X \rightarrow$ Một tập các nhãn lớp
- $y^*: X \rightarrow \mathbb{R}^+$ (miền các giá trị số thực dương)
- ...

■ Lựa chọn các ví dụ học (training/learning examples)

- Các thông tin hướng dẫn quá trình học (training feedback) được chứa ngay trong các ví dụ học, hay là được cung cấp gián tiếp (vd: từ môi trường hoạt động)
- Các ví dụ học theo kiểu có giám sát (supervised) hay không có giám sát (unsupervised)
- Các ví dụ học nên tương thích với (đại diện cho) các ví dụ sẽ được làm việc bởi hệ thống trong tương lai (future test examples)

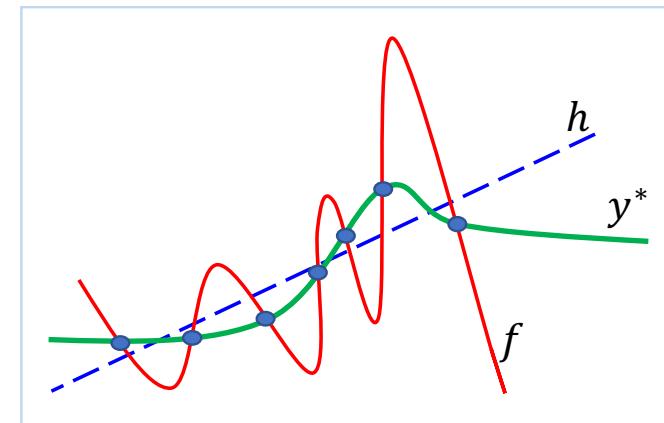
Thiết kế một hệ thống học (2)

■ Lựa chọn dạng hàm (mô hình) f để xấp xỉ hàm y^*

- Hàm đa thức (a polynomial function)
- Một cây quyết định (a decision tree)
- Một mạng nơ-ron nhân tạo
- ...

■ Lựa chọn một giải thuật học có thể tìm hàm f xấp xỉ y^* tốt

- Phương pháp học hàm hồi quy
- Phương pháp học cây quyết định (ID3 hoặc C4.5)
- Phương pháp học lan truyền ngược (Back-propagation)
- ...



Các vấn đề trong Học máy (1)

- Giải thuật học máy (Learning algorithm)
 - Những giải thuật học máy nào có thể học (xấp xỉ) một hàm mục tiêu cần học?
 - Với những điều kiện nào, một giải thuật học máy đã chọn sẽ hội tụ (tiệm cận) hàm mục tiêu cần học?
 - Đối với một lĩnh vực bài toán cụ thể và đối với một cách biểu diễn các ví dụ (đối tượng) cụ thể, giải thuật học máy nào thực hiện tốt nhất?

Các vấn đề trong Học máy (2)

■ Các ví dụ học (Training examples)

- Bao nhiêu ví dụ học là đủ?
- Kích thước của tập học (tập huấn luyện) ảnh hưởng thế nào đối với độ chính xác của hàm mục tiêu học được?
- Các ví dụ lỗi (nhiều) và/hoặc các ví dụ thiếu giá trị thuộc tính (missing-value) ảnh hưởng thế nào đối với độ chính xác?

Các vấn đề trong Học máy (3)

■ Quá trình học (Learning process)

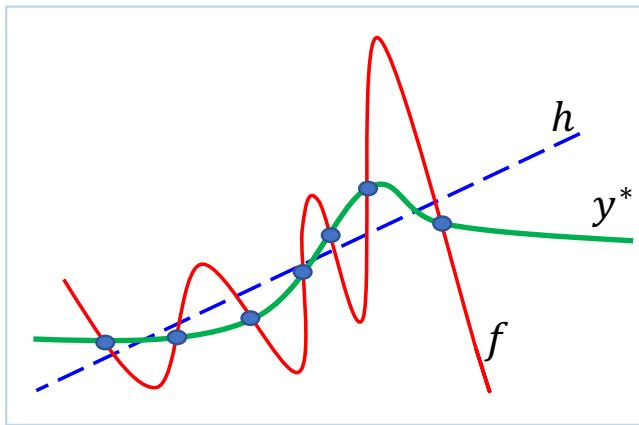
- Chiến lược tối ưu cho việc lựa chọn thứ tự sử dụng (khai thác) các ví dụ học?
- Các chiến lược lựa chọn này làm thay đổi mức độ phức tạp của bài toán học máy như thế nào?
- Các tri thức cụ thể của bài toán (ngoài các ví dụ học) có thể đóng góp thế nào đối với quá trình học?

Các vấn đề trong Học máy (4)

- Khả năng/giới hạn học (Learnability)
 - Mô hình nào?
 - Biểu diễn hàm mục tiêu: Khả năng biểu diễn của f (vd: hàm tuyến tính / hàm phi tuyến) vs. Độ phức tạp của giải thuật và quá trình học
 - Các giới hạn (trên lý thuyết) đối với khả năng học của các giải thuật học máy?
 - Khả năng khái quát hóa (**generalization**) của hệ thống?
 - Để tránh vấn đề “over-fitting” (đạt độ chính xác cao trên tập học, nhưng đạt độ chính xác thấp trên tập thử nghiệm)
 - Khả năng hệ thống tự động thay đổi (thích nghi) biểu diễn (cấu trúc) bên trong của nó?
 - Để cải thiện khả năng (của hệ thống đối với việc) biểu diễn và học hàm mục tiêu

Overfitting (quá khớp)

- Sử dụng ít hàng xóm có thể làm k-NN có thể *giảm lỗi phán đoán cho tập học*, nhưng lại làm **tăng lỗi phán đoán cho những mẫu dữ liệu mới**.

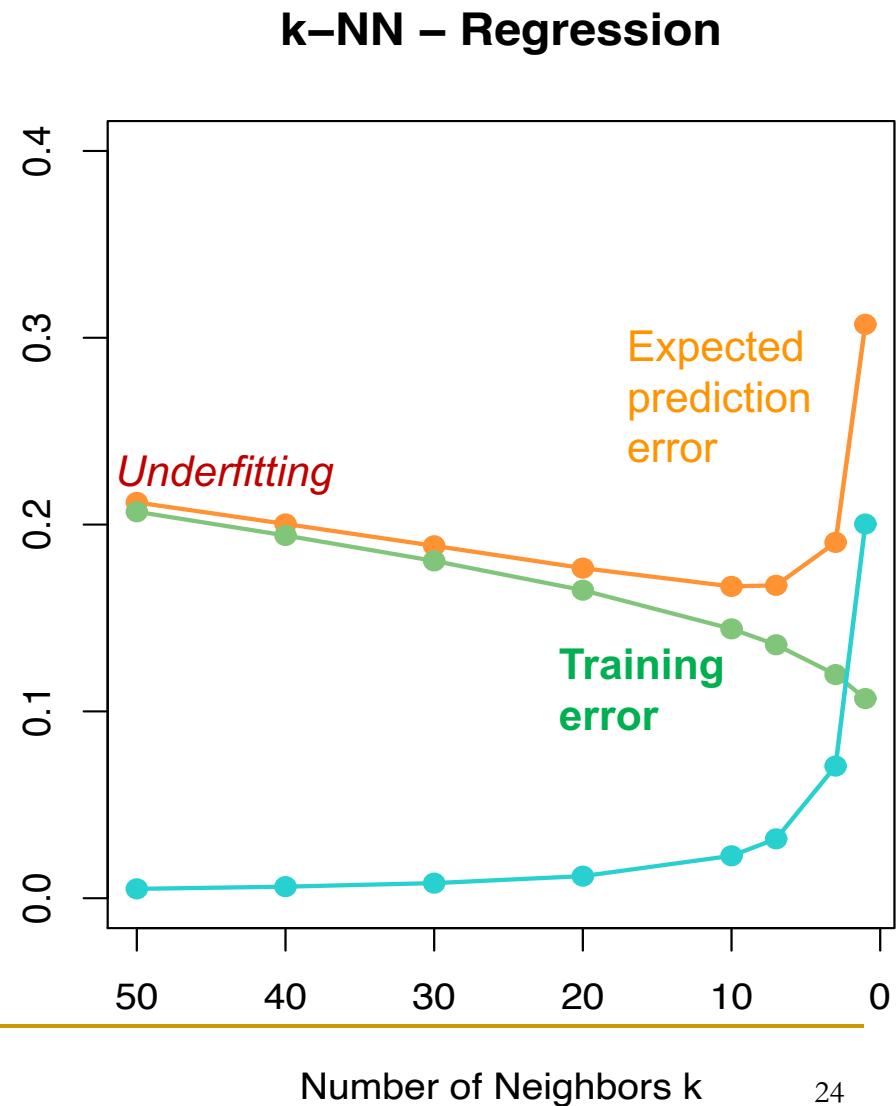


k-NN – Regression



Underfitting (kém khớp)

- Sử dụng quá nhiều hàng xóm có thể làm k-NN giảm hiệu quả phán đoán về các mẫu dữ liệu học và mẫu mới.



Học dựa trên các láng giềng gần nhất

■ **K-nearest neighbors** (k-NN) là một trong số các phương pháp phổ biến trong học máy. Vài tên gọi khác như:

- Instance-based learning
- Lazy learning
- Memory-based learning

■ **Ý tưởng của phương pháp**

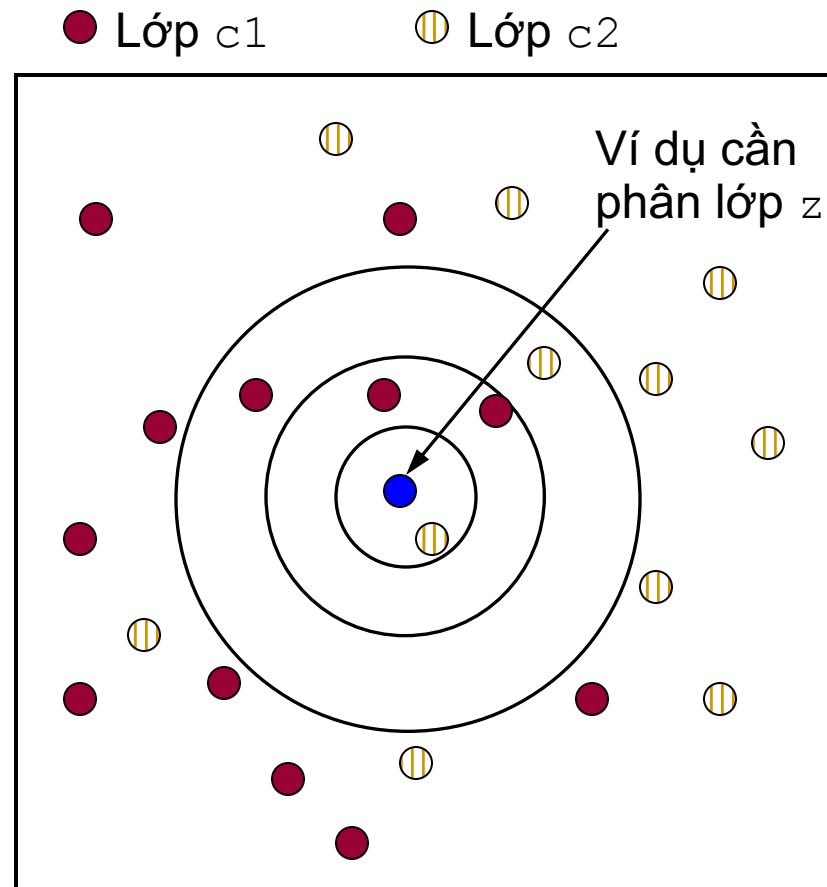
- Không xây dựng một mô hình (mô tả) rõ ràng cho hàm mục tiêu cần học.
- Quá trình học chỉ lưu lại các dữ liệu huấn luyện.
- Việc dự đoán cho một quan sát mới sẽ dựa vào các hàng xóm gần nhất trong tập học.

k-NN

- Hai thành phần chính:
 - Độ đo tương đồng (similarity measure/distance) giữa các đối tượng.
 - Các hàng xóm sẽ dùng vào việc phán đoán.
- *Trong một số điều kiện thì k-NN có thể đạt mức lỗi tối ưu Bayes (mức lỗi mong muốn của bất kỳ phương pháp nào)* [Gyuader and Hengartner, JMLR 2013]
 - Thậm chí khi chỉ dùng 1 hàng xóm gần nhất thì nó cũng có thể đạt đến mức lỗi tối ưu Bayes. [Kontorovich & Weiss, AISTATS 2015]

Ví dụ: bài toán phân lớp

- Xét 1 láng giềng gần nhất
→ Gán z vào lớp c_2
- Xét 3 láng giềng gần nhất
→ Gán z vào lớp c_1
- Xét 5 láng giềng gần nhất
→ Gán z vào lớp c_1



Giải thuật k-NN cho phân lớp

- Mỗi ví dụ học x được biểu diễn bởi 2 thành phần:
 - Mô tả của ví dụ: $x = (x_1, x_2, \dots, x_n)$, trong đó $x_i \in R$
 - Nhãn lớp: $c \in C$, với C là tập các nhãn lớp được xác định trước
- Giai đoạn học
 - Đơn giản là lưu lại các ví dụ học trong tập học: D
- Giai đoạn phân lớp: Để phân lớp cho một ví dụ (mới) z
 - Với mỗi ví dụ học $x \in D$, tính khoảng cách giữa x và z
 - Xác định tập $NB(z)$ – các láng giềng gần nhất của z
 - Gồm k ví dụ học trong D gần nhất với z tính theo một hàm khoảng cách d
 - **Phân z vào lớp chiếm số đông** (the majority class) trong số các lớp của các ví dụ trong $NB(z)$

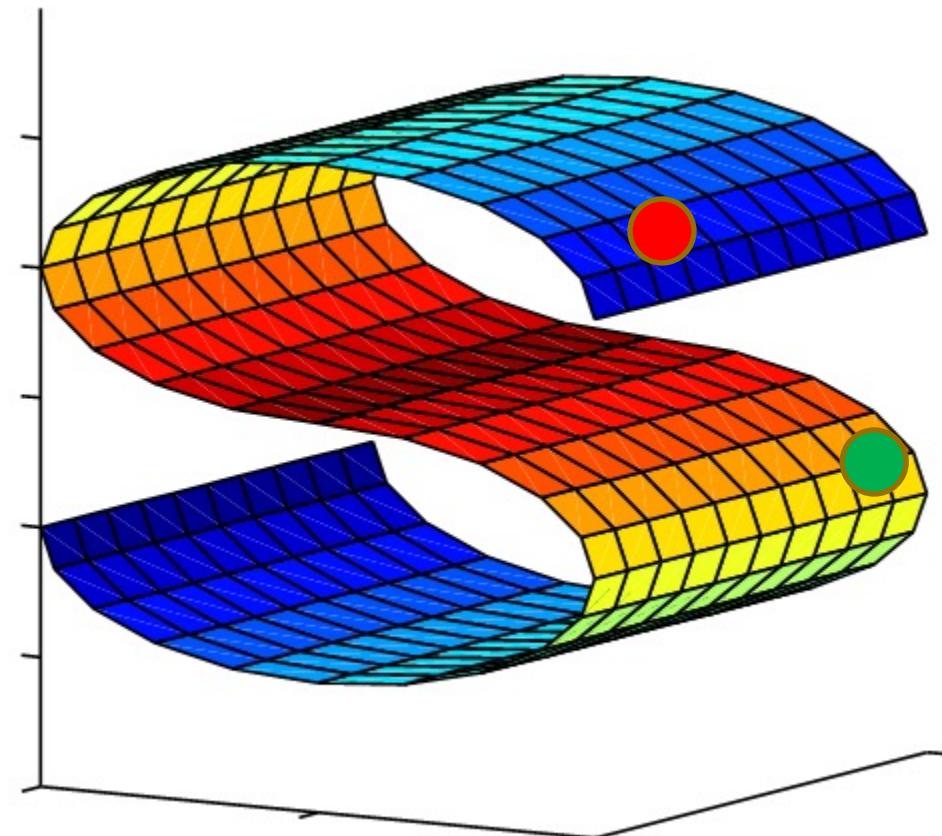
Giải thuật k-NN cho hồi quy

- Mỗi ví dụ học x được biểu diễn bởi 2 thành phần:
 - Mô tả của ví dụ: $x = (x_1, x_2, \dots, x_n)$, trong đó $x_i \in R$
 - Giá trị đầu ra mong muốn: $y_x \in R$ (là một số thực)
- Giai đoạn học
 - Đơn giản là lưu lại các ví dụ học trong tập học D
- Giai đoạn dự đoán: Để dự đoán giá trị đầu ra cho ví dụ z
 - Đối với mỗi ví dụ học $x \in D$, tính khoảng cách giữa x và z
 - Xác định tập $NB(z)$ – các láng giềng gần nhất của z
 - Gồm k ví dụ học trong D gần nhất với z tính theo một hàm khoảng cách d
 - Dự đoán giá trị đầu ra đối với z :
$$y_z = \frac{1}{k} \sum_{x \in NB(z)} y_x$$

k-NN: Các vấn đề cốt lõi

■ Hàm khoảng cách

- ❑ Mỗi hàm sẽ tương ứng với một cách nhìn về dữ liệu.
- ❑ Vô hạn hàm!!!
- ❑ Chọn hàm nào?



k-NN: một hay nhiều láng giềng?

- Về lý thuyết thì 1-NN cũng có thể là một trong số các phương pháp tối ưu.
- k-NN là một phương pháp tối ưu Bayes nếu gặp một số điều kiện, chẳng hạn: y bị chặn, cỡ M của tập học lớn, hàm hồi quy liên tục, và

$$k \rightarrow \infty, (k/M) \rightarrow 0, (k/\log M) \rightarrow +\infty$$

- Trong thực tiễn ta nên lấy nhiều hàng xóm ($k > 1$) khi cần phân lớp/dự đoán, nhưng không quá nhiều. Lý do:
 - Tránh ảnh hưởng của lỗi/nhiễu nếu chỉ dùng 1 hàng xóm.
 - Nếu quá nhiều hàng xóm thì sẽ phá vỡ cấu trúc tiềm ẩn trong dữ liệu.

Hàm tính khoảng cách (2)

■ Các hàm tính khoảng cách hình học (Geometry distance functions)

- Hàm Minkowski (p -norm):

$$d(x, z) = \left(\sum_{i=1}^n |x_i - z_i|^p \right)^{1/p}$$

- Hàm Manhattan ($p = 1$):

$$d(x, z) = \sum_{i=1}^n |x_i - z_i|$$

- Hàm Euclid ($p = 2$):

$$d(x, z) = \sqrt{\sum_{i=1}^n (x_i - z_i)^2}$$

- Hàm Chebyshev ($p = \infty$):

$$d(x, z) = \lim_{p \rightarrow \infty} \left(\sum_{i=1}^n |x_i - z_i|^p \right)^{1/p}$$

$$= \max_i |x_i - z_i|$$

Hàm tính khoảng cách (3)

■ Hàm khoảng cách Hamming

- Đối với các thuộc tính đầu vào là kiểu nhị phân

$$d(x, z) = \sum_{i=1}^n Difference(x_i, z_i)$$

$$Difference(a, b) = \begin{cases} 1, & \text{if } (a \neq b) \\ 0, & \text{if } (a = b) \end{cases}$$

k-NN: Ưu nhược điểm

■ Các ưu điểm

- Chi phí thấp cho quá trình huấn luyện (chỉ việc lưu lại các ví dụ học)
- Hoạt động tốt với các bài toán phân loại gồm nhiều lớp
 - Không cần phải học c bộ phân loại cho c lớp
- Phương pháp học k-NN ($k>>1$) có khả năng xử lý nhiễu cao
 - Phân loại/dự đoán được thực hiện dựa trên k láng giềng gần nhất
- Rất linh động trong việc chọn hàm khoảng cách.
 - Có thể dùng độ tương tự (similarity): cosine
 - Có thể dùng độ đo khác, chẳng hạn Kullback-Leibler divergence, Bregman divergence, ...

■ Các nhược điểm

- Phải lựa chọn hàm tính khoảng cách (sự khác biệt) thích hợp với bài toán
- Chi phí tính toán (thời gian, bộ nhớ) cao tại thời điểm phân loại/dự đoán
- Có thể cho kết quả kém/sai với các thuộc tính không liên quan

Phân lớp Naïve Bayes

- Là các phương pháp học phân lớp có giám sát và dựa trên xác suất
- Dựa trên một mô hình (hàm) xác suất
- Việc phân loại dựa trên các giá trị xác suất của các khả năng xảy ra của các giả thiết
- Là một trong các phương pháp học máy thường được sử dụng trong các bài toán thực tế
- Dựa trên định lý Bayes (Bayes theorem)

Định lý Bayes

$$P(h | D) = \frac{P(D | h).P(h)}{P(D)}$$

- $P(h)$: Xác suất trước (tiên nghiệm) của giả thiết h
- $P(D)$: Xác suất trước (tiên nghiệm) của việc quan sát được dữ liệu D
- $P(D | h)$: Xác suất (có điều kiện) của việc quan sát được dữ liệu D , nếu biết giả thiết h là đúng. (likelihood)
- $P(h | D)$: Xác suất (hậu nghiệm) của giả thiết h là đúng, nếu quan sát được dữ liệu D
 - **Nhiều phương pháp phân loại dựa trên xác suất sẽ sử dụng xác suất hậu nghiệm (*posterior probability*) này!**

Xác suất hậu nghiệm cực đại (MAP)

- Với một tập các giả thiết (các phân lớp) có thể H , hệ thống học sẽ tìm **giả thiết có thể xảy ra nhất (the most probable hypothesis)** $h \in H$ đối với các dữ liệu quan sát được D
- Giả thiết h này được gọi là giả thiết có xác suất hậu nghiệm cực đại (**Maximum a posteriori – MAP**)

$$h_{MAP} = \arg \max_{h \in H} P(h | D)$$

$$h_{MAP} = \arg \max_{h \in H} \frac{P(D | h) \cdot P(h)}{P(D)} \quad (\text{bởi định lý Bayes})$$

$$h_{MAP} = \arg \max_{h \in H} P(D | h) \cdot P(h)$$

($P(D)$ là như nhau đối với các giả thiết h)

Đánh giá khả năng có thể nhất (MLE)

- Phương pháp MAP: Với một tập các giả thiết có thể H , cần tìm một giả thiết cực đại hóa giá trị: $P(D|h) \cdot P(h)$
- Giả sử (assumption) trong phương pháp **đánh giá khả năng có thể nhất (Maximum likelihood estimation – MLE)**: Tất cả các giả thiết đều có giá trị xác suất trước nhau: $P(h_i) = P(h_j)$, $\forall h_i, h_j \in H$
- **Phương pháp MLE** tìm giả thiết cực đại hóa giá trị $P(D|h)$; trong đó $P(D|h)$ được gọi là *khả năng có thể* (*likelihood*) của dữ liệu D đối với h
- Giả thiết có khả năng nhất (maximum likelihood hypothesis)

$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

MLE: Ví dụ

■ Tập H bao gồm 2 giả thiết có thể

- h_1 : Anh ta chơi tennis
- h_2 : Anh ta không chơi tennis

D: Tập dữ liệu (các ngày) mà trong đó thuộc tính *Outlook* có giá trị *Sunny* và thuộc tính *Wind* có giá trị *Strong*

■ Tính 2 giá trị khả năng xảy ra (likelihood values) của dữ liệu D đối với 2 giả thiết: $P(D|h_1)$ và $P(D|h_2)$

- $P(\text{Outlook}=\text{Sunny}, \text{Wind}=\text{Strong} | h_1) = 1/8$
- $P(\text{Outlook}=\text{Sunny}, \text{Wind}=\text{Strong} | h_2) = 1/4$

■ Giả thiết MLE $h_{\text{MLE}}=h_1$ nếu $P(D|h_1) \geq P(D|h_2)$; và ngược lại thì $h_{\text{MLE}}=h_2$

→ Bởi vì $P(\text{Outlook}=\text{Sunny}, \text{Wind}=\text{Strong} | h_1) < P(\text{Outlook}=\text{Sunny}, \text{Wind}=\text{Strong} | h_2)$, hệ thống kết luận rằng:
Anh ta sẽ không chơi tennis!

Phân loại Naïve Bayes (1)

■ Biểu diễn bài toán phân loại (classification problem)

- Một tập học D_{train} , trong đó mỗi ví dụ học x được biểu diễn là một vectơ n chiều: (x_1, x_2, \dots, x_n)
- Một tập xác định các nhãn lớp: $C = \{c_1, c_2, \dots, c_m\}$
- Với một ví dụ (mới) z , thì z sẽ được phân vào lớp nào?

■ Mục tiêu: Xác định phân lớp có thể (phù hợp) nhất đối với z

$$c_{MAP} = \arg \max_{c_i \in C} P(c_i | z)$$

$$c_{MAP} = \arg \max_{c_i \in C} P(c_i | z_1, z_2, \dots, z_n)$$

$$c_{MAP} = \arg \max_{c_i \in C} \frac{P(z_1, z_2, \dots, z_n | c_i).P(c_i)}{P(z_1, z_2, \dots, z_n)} \quad (\text{bởi định lý Bayes})$$

Phân loại Naïve Bayes (2)

- Để tìm được phân lớp có thể nhất đối với z ...

$$c_{MAP} = \arg \max_{c_i \in C} P(z_1, z_2, \dots, z_n | c_i) \cdot P(c_i) \quad (\text{P}(z_1, z_2, \dots, z_n) \text{ là nhau nhau với các lớp})$$

- **Giả thuyết (assumption) trong phương pháp phân loại Naïve Bayes:** Các thuộc tính là *độc lập* có *điều kiện* (*conditionally independent*) đối với các lớp

$$P(z_1, z_2, \dots, z_n | c_i) = \prod_{j=1}^n P(z_j | c_i)$$

- Phân loại Naïve Bayes tìm phân lớp có thể nhất đối với z

$$c_{NB} = \arg \max_{c_i \in C} P(c_i) \cdot \prod_{j=1}^n P(z_j | c_i)$$

Phân loại Naïve Bayes: Ví dụ (2)

■ Biểu diễn bài toán phân loại

- $z = (\text{Age}=\text{Young}, \text{Income}=\text{Medium}, \text{Student}=\text{Yes}, \text{Credit_Rating}=\text{Fair})$
- Có 2 phân lớp có thể: c_1 ("Mua máy tính") và c_2 ("Không mua máy tính")

■ Tính giá trị xác suất trước cho mỗi phân lớp

- $P(c_1) = 9/14$
- $P(c_2) = 5/14$

■ Tính giá trị xác suất của mỗi giá trị thuộc tính đối với mỗi phân lớp

- | | |
|---|--|
| • $P(\text{Age}=\text{Young} c_1) = 2/9;$ | $P(\text{Age}=\text{Young} c_2) = 3/5$ |
| • $P(\text{Income}=\text{Medium} c_1) = 4/9;$ | $P(\text{Income}=\text{Medium} c_2) = 2/5$ |
| • $P(\text{Student}=\text{Yes} c_1) = 6/9;$ | $P(\text{Student}=\text{Yes} c_2) = 1/5$ |
| • $P(\text{Credit_Rating}=\text{Fair} c_1) = 6/9;$ | $P(\text{Credit_Rating}=\text{Fair} c_2) = 2/5$ |

Phân loại văn bản bằng NB (1)

■ Biểu diễn bài toán phân loại văn bản

- Tập học D , trong đó mỗi ví dụ học là một biểu diễn văn bản gắn với một nhãn lớp: $D = \{(d_k, c_i)\}$
- Một tập các nhãn lớp xác định: $C = \{c_i\}$

■ Giai đoạn học

- Từ tập các văn bản trong D , trích ra tập các từ khóa $T = \{t_j\}$
- Gọi D_{c_i} là tập các văn bản trong D có nhãn lớp c_i
- Đối với mỗi phân lớp c_i
 - Tính giá trị xác suất trước của phân lớp c_i : $P(c_i) = \frac{|D_{c_i}|}{|D|}$
 - Đối với mỗi từ khóa t_j , tính xác suất từ khóa t_j xuất hiện đối với lớp c_i

$$P(t_j | c_i) = \frac{\left(\sum_{d_k \in D_{c_i}} n(d_k, t_j) \right) + 1}{\left(\sum_{d_k \in D_{c_i}} \sum_{t_m \in T} n(d_k, t_m) \right) + |T|} \quad (n(d_k, t_j) : \text{số lần xuất hiện của từ khóa } t_j \text{ trong văn bản } d_k)$$

Phân loại văn bản bằng NB (2)

■ Giai đoạn phân lớp đối với một văn bản mới d

- Từ văn bản d , trích ra tập T_d gồm các từ khóa (keywords) t_j đã được định nghĩa trong tập T
- **Giả sử (assumption).** Xác suất từ khóa t_j xuất hiện đối với lớp c_i là độc lập đối với vị trí của từ khóa đó trong văn bản

$$P(t_j \text{ ở vị trí } k | c_i) = P(t_j \text{ ở vị trí } m | c_i), \forall k, m$$

- Đối với mỗi phân lớp c_i , tính xác suất hậu nghiệm của văn bản d đối với c_i :

$$P(c_i) \cdot \prod_{t_j \in T_d} P(t_j | c_i)$$

- Phân lớp văn bản d thuộc vào lớp c^*

$$c^* = \arg \max_{c_i \in C} P(c_i) \cdot \prod_{t_j \in T_d} P(t_j | c_i)$$

Nhắc lại: Học có giám sát

- **Học có giám sát (Supervised learning)**
 - Tập dữ liệu học (*training data*) bao gồm các quan sát (*examples, observations*), mà mỗi quan sát được *gắn kèm với một giá trị đầu ra mong muốn*.
 - Mục đích là học một hàm (vd: một phân lớp, một hàm hồi quy,...) phù hợp với tập dữ liệu hiện có và khả năng tổng quát hoá cao.
 - Hàm học được sau đó sẽ được dùng để dự đoán cho các quan sát mới.
 - *Phân loại (classification)*: nếu đầu ra (*output – y*) thuộc tập rời rạc và hữu hạn.

Phân loại

- **Multi-class classification (phân loại nhiều lớp):** mỗi quan sát x chỉ nhận 1 nhãn trong tập nhãn lớp $\{c_1, c_2, \dots, c_L\}$
 - Lọc Spam: y thuộc {spam, normal}
 - Đánh giá nguy cơ tín dụng: y thuộc {high, normal}
 - Phán đoán tấn công mạng: ?



1. Đánh giá hiệu năng hệ thống học máy..

- **Đánh giá lý thuyết (theoretical evaluation)**: nghiên cứu các khía cạnh lý thuyết của một hệ thống mà có thể chứng minh được.
 - Tốc độ học, thời gian học,
 - Bao nhiêu ví dụ học là đủ?
 - Độ chính xác trung bình của hệ thống,
 - Khả năng chống nhiễu,...
- **Đánh giá thực nghiệm (experimental evaluation)**: quan sát hệ thống làm việc trong thực tế, sử dụng một hoặc nhiều tập dữ liệu và các tiêu chí đánh giá. Tổng hợp đánh giá từ các quan sát đó.
- Chúng ta sẽ nghiên cứu cách đánh giá thực nghiệm.

1. Đánh giá hiệu năng hệ thống học máy...

- **Bài toán đánh giá** (model assessment): cần đánh giá hiệu năng của phương pháp (model) học máy A, chỉ dựa trên bộ dữ liệu đã quan sát D.
- Việc đánh giá hiệu năng của hệ thống
 - Thực hiện một cách tự động, sử dụng một tập dữ liệu.
 - Không cần sự tham gia (can thiệp) của người dùng.
- **Chiến lược đánh giá** (evaluation strategies)
 - Làm sao có được một đánh giá đáng tin cậy về hiệu năng của hệ thống?
- Các **tiêu chí đánh giá** (evaluation metrics)
 - Làm sao để đo hiệu năng của hệ thống?

2. Các phương pháp đánh giá

- Hold-out (chia đôi)
- Stratified sampling (lấy mẫu phân tầng)
- Repeated hold-out (chi đôi nhiều lần)
- Cross-validation (đánh giá chéo)
 - k -fold
 - Leave-one-out
- Bootstrap sampling

Hold-out (Splitting)

- Toàn bộ tập ví dụ D được chia thành 2 tập con **không giao nhau**
 - Tập huấn luyện D_{train} – để huấn luyện hệ thống
 - Tập kiểm thử D_{test} – để đánh giá hiệu năng của hệ thống đã học $\rightarrow D = D_{train} \cup D_{test}$, và thường là $|D_{train}| \gg |D_{test}|$
- Các yêu cầu:
 - Bất kỳ ví dụ nào thuộc vào tập kiểm thử D_{test} đều không được sử dụng trong quá trình huấn luyện hệ thống
 - Bất kỳ ví dụ nào được sử dụng trong giai đoạn huấn luyện hệ thống (i.e., thuộc vào D_{train}) đều không được sử dụng trong giai đoạn đánh giá hệ thống
 - Các ví dụ kiểm thử trong D_{test} cho phép một đánh giá không thiên vị đối với hiệu năng của hệ thống
- Các lựa chọn thường gặp: $|D_{train}| = (2/3).|D|$, $|D_{test}| = (1/3).|D|$
- Phù hợp khi ta có tập ví dụ D có kích thước lớn

Stratified sampling

- Đối với các tập ví dụ có kích thước nhỏ hoặc không cân xứng (unbalanced datasets), các ví dụ trong tập huấn luyện và thử nghiệm có thể không phải là đại diện
 - Ví dụ: Có (rất) ít ví dụ đối với một số lớp
- Mục tiêu: Phân bố lớp (class distribution) trong tập huấn luyện và tập kiểm thử phải xấp xỉ như trong tập toàn bộ các ví dụ (D)
- Lấy mẫu phân tầng (Stratified sampling)
 - Là một phương pháp để cân xứng (về phân bố lớp)
 - Đảm bảo tỷ lệ phân bố lớp (tỷ lệ các ví dụ giữa các lớp) trong tập huấn luyện và tập kiểm thử là xấp xỉ nhau
- Phương pháp lấy mẫu phân tầng không áp dụng được cho bài toán hồi quy (vì giá trị đầu ra của hệ thống là một giá trị số thực, không phải là một nhãn lớp)

Repeated hold-out

- Áp dụng phương pháp đánh giá Hold-out nhiều lần, để sinh ra (sử dụng) các tập huấn luyện và thử nghiệm khác nhau
 - Trong mỗi bước lặp, một tỷ lệ nhất định của tập D **được lựa chọn ngẫu nhiên** để tạo nên tập huấn luyện (có thể sử dụng kết hợp với phương pháp lấy mẫu phân tầng – stratified sampling)
 - Các giá trị lỗi (hoặc các giá trị đối với các tiêu chí đánh giá khác) ghi nhận được trong các bước lặp này được *lấy trung bình cộng (averaged)* để xác định giá trị lỗi tổng thể
- Phương pháp này vẫn không hoàn hảo
 - Mỗi bước lặp sử dụng một tập kiểm thử khác nhau
 - Có một số ví dụ trùng lặp (được sử dụng lại nhiều lần) trong các tập kiểm thử này

Cross-validation

- Để tránh việc trùng lặp giữa các tập kiểm thử (một số ví dụ cùng xuất hiện trong các tập kiểm thử khác nhau)
- k -fold cross-validation
 - Tập toàn bộ các ví dụ D được chia thành k tập con **không giao nhau** (gọi là “*fold*”) có kích thước xấp xỉ nhau
 - Mỗi lần (trong số k lần) lặp, một tập con được sử dụng làm tập kiểm thử, và $(k-1)$ tập con còn lại được dùng làm tập huấn luyện
 - k giá trị lỗi (mỗi giá trị tương ứng với một *fold*) được tính trung bình cộng để thu được giá trị lỗi tổng thể
- Các lựa chọn thông thường của k : 10, hoặc 5
- Thông thường, mỗi tập con (*fold*) được lấy mẫu phân tầng (xấp xỉ phân bố lớp) trước khi áp dụng quá trình đánh giá Cross-validation
- Phù hợp khi ta có tập ví dụ D vừa và nhỏ

Leave-one-out cross-validation

- Một trường hợp (kiểu) của phương pháp Cross-validation
 - Số lượng các nhóm (folds) bằng kích thước của tập dữ liệu ($k=|D|$)
 - Mỗi nhóm (fold) chỉ bao gồm một ví dụ
- Khai thác tối đa (triệt để) tập ví dụ ban đầu
- Không hề có bước lấy mẫu ngẫu nhiên (no random subsampling)
- Áp dụng lấy mẫu phân tầng (stratification) không phù hợp
→ Vì ở mỗi bước lặp, tập thử nghiệm chỉ gồm có một ví dụ
- Chi phí tính toán (rất) cao
- Phù hợp khi ta có một tập ví dụ D (rất) nhỏ

3. Lựa chọn tham số

- Nhiều phương pháp học máy thường có (tập) tham số (hyperparameters), buộc người dùng phải đưa giá trị vào.
 - Ridge regression: λ
 - Linear SVM: C
- Làm sao để lựa chọn giá trị tốt nhất cho các tham số?
→ *model selection*
- **Model selection:** *từ một tập học D, cần lựa chọn bộ tham số (model) trong phương pháp học A sao cho hệ thống được huấn luyện tốt nhất từ D.*
- Tập tối ưu (validation set) được sử dụng để tối ưu giá trị các tham số trong giải thuật học máy (thường lấy từ tập D)
 - Đối với một tham số, giá trị tối ưu là giá trị giúp sinh ra *hiệu năng cực đại đối với tập tối ưu*

Lựa chọn tham số: sử dụng Hold-out

- Cho trước tập quan sát D, ta lựa chọn tham số λ cho phương pháp học A như sau:
 - Chọn tập hữu hạn S mà chứa các giá trị tiềm năng cho λ .
 - Chọn độ đo P để đánh giá hiệu năng.
 - Chia D thành 2 tập rời nhau: D_{train} và $T_{validation}$
 - Với mỗi giá trị $\lambda \in S$:
 - Học A từ tập học D_{train} với tham số đầu vào λ . Đo hiệu năng trên tập $T_{validation}$ → thu được P_λ
 - Chọn λ^* mà có P_λ tốt nhất.
- Có thể học lại A từ D với tham số λ^* để hệ thống thu được kết quả tốt.
- Có thể thay Hold-out bằng kỹ thuật khác (e.g., sampling, cross-validation).

4. Đánh giá và lựa chọn mô hình

- Cho trước tập quan sát D , ta cần lựa chọn tham số λ (model selection) cho phương pháp học A và đánh giá (assessment) chất lượng tổng thể của A .
 - Chọn tập hữu hạn S mà chứa các giá trị tiềm năng cho λ .
 - Chọn độ đo P để đánh giá hiệu năng.
 - Chia tập D thành 3 tập rời nhau: D_{train} , $T_{\text{validation}}$, và T_{test}
 - Với mỗi giá trị $\lambda \in S$:
 - Học A từ tập học D_{train} với tham số đầu vào λ . Đo hiệu năng trên tập $T_{\text{validation}}$ → thu được P_λ
 - Chọn λ^* mà có P_λ tốt nhất.
 - Huấn luyện A trên tập $D_{\text{train}} \cup T_{\text{validation}}$, với tham số đầu vào λ^* .
 - Đo hiệu năng của hệ thống trên tập T_{test} .
 - Có thể thay Hold-out bằng kỹ thuật khác (cross-validation).

5. Các tiêu chí đánh giá.

- **Tính chính xác (Accuracy)**

→ Mức độ dự đoán (phân lớp) chính xác của hệ thống (đã được huấn luyện) đối với các ví dụ kiểm chứng (test instances)

- **Tính hiệu quả (Efficiency)**

→ Chi phí về thời gian và tài nguyên (bộ nhớ) cần thiết cho việc huấn luyện và kiểm thử hệ thống

- **Khả năng xử lý nhiễu (Robustness)**

→ Khả năng xử lý (chịu được) của hệ thống đối với các ví dụ nhiễu (lỗi) hoặc thiếu giá trị

5. Các tiêu chí đánh giá..

- Khả năng mở rộng (Scalability)
→ Hiệu năng của hệ thống (vd: tốc độ học/phân loại) thay đổi như thế nào đối với kích thước của tập dữ liệu
- Khả năng diễn giải (Interpretability)
→ Mức độ dễ hiểu (đối với người sử dụng) của các kết quả và hoạt động của hệ thống
- Mức độ phức tạp (Complexity)
→ Mức độ phức tạp của mô hình hệ thống (hàm mục tiêu) học được

Tính chính xác (Accuracy)

- Đối với bài toán phân loại

$$\text{Accuracy} = \frac{\text{Số phán đoán chính xác}}{\text{Tổng số phán đoán}}$$

- Đối với bài toán hồi quy: MAE (mean absolute error)

$$MAE = \frac{1}{|D_{test}|} \sum_{x \in D_{test}} |o(x) - y(x)|$$

- $o(x)$: Giá trị đầu ra (dự đoán) bởi hệ thống đối với ví dụ x
- $d(x)$: Giá trị đầu ra thực sự (đúng) đối với ví dụ x

Ma trận nhầm lẫn (Confusion matrix)

- Còn được gọi là Contingency Table
- Chỉ được sử dụng đối với bài toán phân loại
 - Không thể áp dụng cho bài toán hồi quy (dự đoán)
- TP_i (true positive): Số lượng các ví dụ thuộc lớp c_i được phân loại chính xác vào lớp c_i ;
- FP_i (false positive): Số lượng các ví dụ không thuộc lớp c_i bị phân loại nhầm vào lớp c_i ;
- TN_i (true negative): Số lượng các ví dụ không thuộc lớp c_i được phân loại (chính xác)
- FN_i (false negative): Số lượng các ví dụ thuộc lớp c_i bị phân loại nhầm (vào các lớp khác c_j)

Lớp c_i		Được phân lớp bởi hệ thống	
		Thuộc	Ko thuộc
Phân lớp thực sự (đúng)	Thuộc	TP_i	FN_i
	Ko thuộc	FP_i	TN_i

Precision and Recall (1)

- Rất hay được sử dụng để đánh giá các hệ thống phân loại văn bản
- **Precision** đối với lớp C_i
 - Tổng số các ví dụ thuộc lớp C_i được phân loại chính xác chia cho tổng số các ví dụ được phân loại vào lớp C_i
- **Recall** đối với lớp C_i
 - Tổng số các ví dụ thuộc lớp C_i được phân loại chính xác chia cho tổng số các ví dụ thuộc lớp C_i

$$Precision(c_i) = \frac{TP_i}{TP_i + FP_i}$$

$$Recall(c_i) = \frac{TP_i}{TP_i + FN_i}$$

Precision and Recall (2)

- Làm thế nào để tính toán được giá trị Precision và Recall (một cách tổng thể) cho toàn bộ các lớp $C=\{C_i\}$?
- Trung bình vi mô (Micro-averaging)

$$Precision = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)}$$

$$Recall = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}$$

- Trung bình vĩ mô (Macro-averaging)

$$Precision = \frac{\sum_{i=1}^{|C|} Precision(c_i)}{|C|}$$

$$Recall = \frac{\sum_{i=1}^{|C|} Recall(c_i)}{|C|}$$

F_1

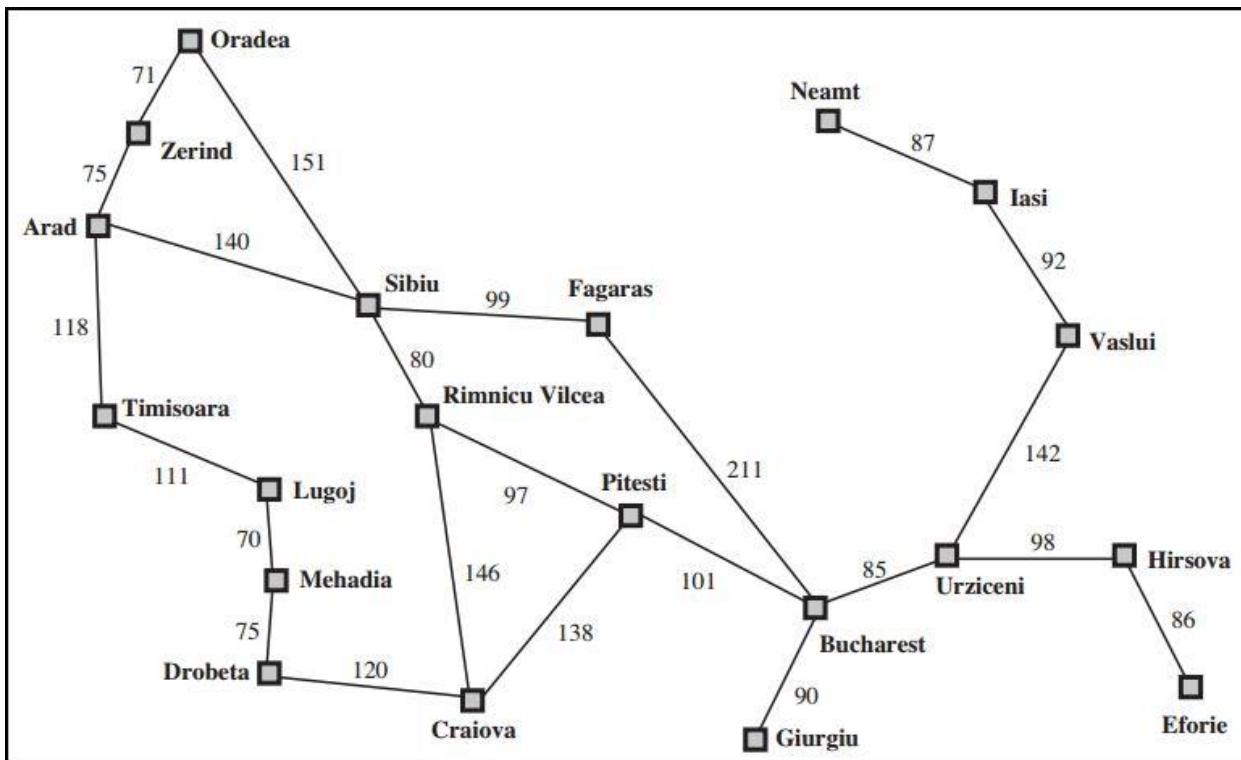
- Tiêu chí đánh giá F_1 là sự kết hợp của 2 tiêu chí đánh giá *Precision* và *Recall*

$$F_1 = \frac{2.Precision \cdot Recall}{Precision + Recall} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

- F_1 là một **trung bình điều hòa (harmonic mean)** của các tiêu chí *Precision* và *Recall*
 - F_1 có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị *Precision* và *Recall*
 - F_1 có giá trị lớn nếu cả 2 giá trị *Precision* và *Recall* đều lớn

Phần I: Giải quyết vấn đề

Bài 1: Cho bản đồ của Romania



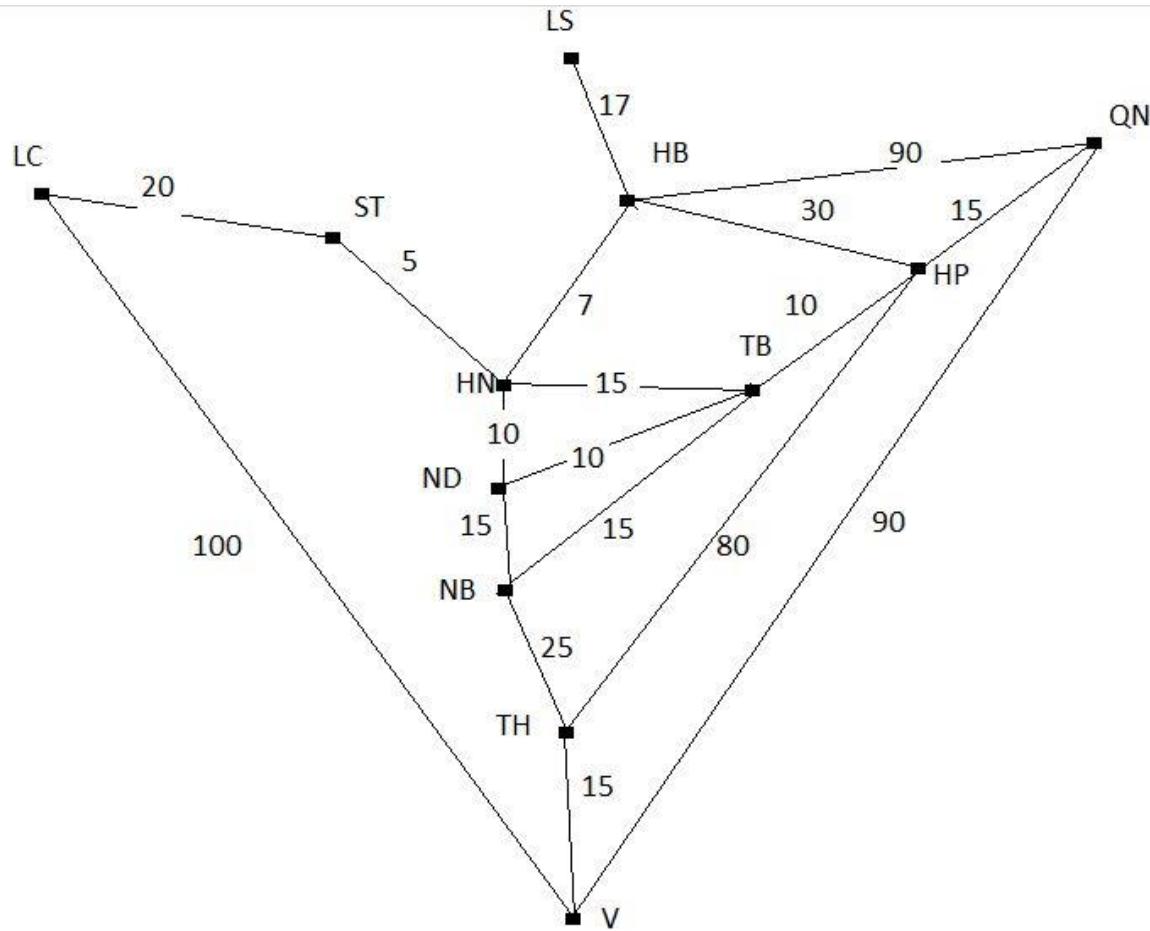
- 1.1. Thực hiện các giải thuật tìm kiếm theo chiều rộng (BFS), tìm kiếm với chi phí cực tiểu (UCS), tìm kiếm theo chiều sau (DFS) và tìm kiếm sâu dần (IDS) từ thành phố Lugoj đến Bucharest
- 1.2. Cho hàm heuristic $h(n)$ là hàm ước lượng khoảng cách theo đường chim bay từ thành phố n đến Bucharest

n	Arad	Bucharest	Craiova	Drobeta	Eforie	Fagaras	Giurgiu	Hirsova	Iasi	Lugoj
$h(n)$	366	0	160	242	161	176	77	151	226	244

n	Mehadia	Neamt	Oradea	Pitesti	Rimnicu Vilcea	Sibiu	Timisoara	Urziceni	Vaslui	Zerind
$h(n)$	241	234	380	10	193	253	329	80	199	374

Thực hiện tìm kiếm theo giải thuật tham lam (GBFS) và giải thuật A* từ thành phố Lugoj đến Bucharest

Bài 2: Cho bản đồ Việt Nam



Cùng với hàm ước lượng $h(n)$ là khoảng cách theo đường chim bay từ tỉnh (thành phố) n đến thành phố Vinh (V)

n	HN	ST	LC	HB	LS	HP	QN	TB	ND	NB	TH	V
$h(n)$	50	60	75	65	70	80	80	55	45	20	15	0

Thực hiện tìm kiếm đường đi theo các giải thuật BFS, UCS, DFS, IDS, GBFS, và A* từ thành phố Hà Nội (HN) đến thành phố Vinh (V)

Bài 3: Xét một không gian trạng thái trong đó trạng thái bắt đầu là số 1. Mỗi trạng thái k có 2 trạng thái con là $2k$ và $2k + 1$ (biểu diễn theo cấu trúc cây)

- Vẽ cây biểu diễn cho không gian trạng thái các số từ 1 đến 15
- Ta muốn đến 1 trạng thái mục tiêu x nào đó ($x \in [1, 15]$). Hãy phát biểu bài toán trên (trạng thái bắt đầu, hàm chuyển trạng thái, trạng thái mục tiêu và chi phí bước đi)

- c. Giả sử trạng thái mục tiêu là 11. Hãy liệt kê theo thứ tự các node sẽ được thăm bằng giải thuật tìm kiếm theo chiều rộng (BFS), giải thuật tìm kiếm độ sâu giới hạn (DLS) với giới hạn là 3 và giải thuật tìm kiếm sâu dần (IDS)
- d. Có thể sửa đổi phát biểu bài toán mà cho phép ta giải quyết được vấn đề di chuyển từ trạng thái 1 đến trạng thái mục tiêu x mà hầu như không cần tìm kiếm gì không?
- e. Theo như phát biểu bài toán trong phần b, gọi hành động đi từ node k đến node 2k là **Left** và hành động đi từ node k đến node 2k+1 là **Right**. Hãy thử đề xuất 1 giải thuật tìm lời giải cho bài toán này mà không cần tìm kiếm?

Bài 4: Giả sử có 2 người bạn sống ở 2 thành phố khác nhau trên một bản đồ (ví dụ như bản đồ Romania chẳng hạn). Tại mỗi lần, chúng ta có thể đồng thời di chuyển 2 người bạn tới 2 thành phố hàng xóm của 2 thành phố tương ứng với người đó đang đứng hiện tại. Lượng thời gian cần thiết để di chuyển từ thành phố i tới thành phố hàng xóm j bằng với khoảng cách $d(i,j)$ giữa 2 thành phố. Ở mỗi lượt đi, người bạn mà đến trước phải đợi cho đến khi người bạn còn lại đến được đích trước khi lượt đi tiếp theo bắt đầu. Chúng ta muốn 2 người bạn gặp nhau nhanh nhất có thể (sau một hữu hạn các bước đi, 2 người dừng chân tại cùng 1 thành phố)

- a. Phát biểu bài toán trên (4 thành phần: trạng thái đầu, các hành động-hàm chuyển trạng thái, kiểm tra mục tiêu, chi phí đường đi). Không gian trạng thái của bài toán là gì?
- b. Gọi $D(i, j)$ là khoảng cách theo đường chim bay từ thành phố i đến thành phố j. Hàm ước lượng heuristic nào sau đây là phù hợp? (i) $D(i,j)$ (ii) $2*D(i,j)$ (iii) $D(i,j) / 2$
- c. Hỏi có tồn tại một bản đồ (đô thị) nào không mà không cho lời giải cho bài toán trên?
- d. Có tồn tại bản đồ nào trong đó tất cả các lời giải phải yêu cầu 1 người bạn thăm cùng 1 thành phố ít nhất 2 lần

Bài 5 (lập trình - phần c, d): Xem xét bài toán tìm đường đi ngắn nhất giữa điểm trong mặt phẳng mà có chướng ngại vật là đa giác lồi như sau:

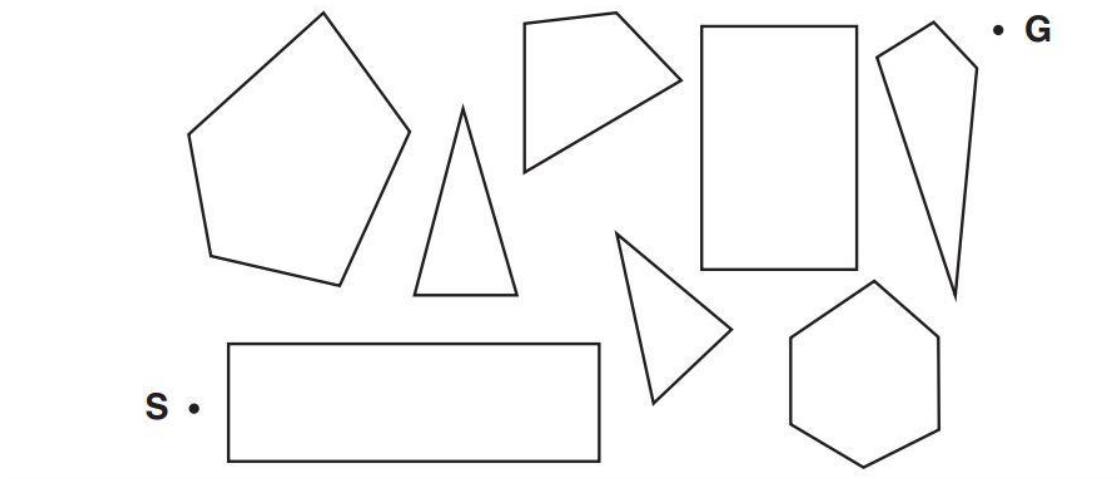


Figure 3.31 A scene with polygonal obstacles. *S* and *G* are the start and goal states.

Đây cũng có thể mô phỏng cho bài toán mà một robot cần phải giải quyết để tránh va chạm với vật cản trong quá trình di chuyển đến đích

- Giả sử không gian trạng thái bao gồm tất cả các vị trí (x, y) trong mặt phẳng. Có bao nhiêu trạng thái? Có bao nhiêu đường đi có thể tới đích?
- Giải thích một cách ngắn gọn tại sao đường đi ngắn nhất từ 1 đỉnh của 1 đa giác đến một đỉnh khác nào đó (có thể của đa giác khác) là một đường gồm nhiều đoạn thẳng, trong đó mỗi đoạn thẳng là đoạn thẳng nối 2 đỉnh tồn tại trên mặt phẳng (có thể cùng 1 đa giác hoặc 2 đa giác khác nhau). Sau đó, đề xuất một không gian trạng thái tốt cho bài toán này. Lực lượng của không gian này là bao nhiêu?
- Định nghĩa các hàm cần thiết để cài đặt bài toán này, bao gồm một hàm ACTIONS sẽ nhận đầu vào là tọa độ 1 đỉnh và trả lại 1 tập các vectors, trong đó mỗi vector sẽ là vector giữa đỉnh đầu vào và đỉnh mà có thể đến được từ đỉnh đầu vào theo đường thẳng. Dùng thuật toán A* với hàm heuristic là khoảng cách theo đường thẳng từ 1 điểm tới đích *G* (tọa độ *S*, *G* và các đỉnh của các đa giác nhập tùy chọn)
- Thử áp dụng các giải thuật tìm kiếm khác đã học và đánh giá hiệu năng của chúng

Bài 6: Xét một trò chơi như sau:

Có 2 người chơi X và Y. Tại mọi thời điểm, trạng thái của game là 1 số nguyên *N*. Khi đến lượt của X đi, X có thể chọn giữa 2 nước đi có thể là:

Nước đi A: $N := N + (N \bmod 23) - 11$

Nước đi B: $N := N + (N \bmod 7) - 4$

Khi đến lượt đi của Y. Y có thể chọn 1 trong 2 nước đi có thể là:

Nước đi C: $N := N + 2 * (N \bmod 17) - 16$

Nước đi D: $N := N + ((N \bmod 11) - 5) * (N \bmod 17)$

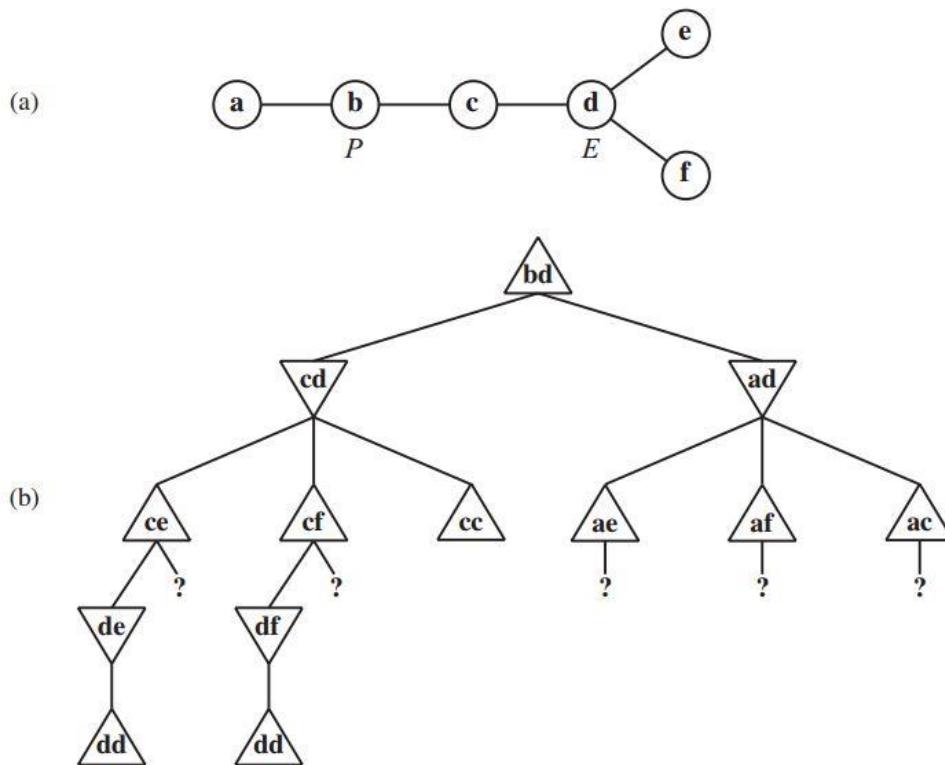
Tại trạng thái bắt đầu, $N = 100$ và là lượt đi của X. Trò chơi được chơi trong 4 lượt: X đi, Y đi, sau đó X đi và Y đi nốt lượt cuối, trò chơi kết thúc

Giả sử rằng cây tìm kiếm được tạo theo chiến lược deep-first từ trái qua phải (A, B, C, D cũng được áp dụng theo thứ tự này)

- Vẽ cây được tạo và thực hiện thuật toán cắt tỉa alpha-beta trên cây đó
- Tính giá trị của node gốc của cây
- Hướng đi của X là gì?

Bài 7: Cho 1 đồ thị vô hướng G, ta xét 1 trò chơi pursuit-evasion như sau:

Có người chơi P và E đứng trên 2 đỉnh của đồ thị (hình a), người chơi luân phiên nhau đi lượt của mình. Mỗi lượt đi, người chơi tương ứng đi từ đỉnh hiện tại sang 1 đỉnh hàng xóm nào đó, trong đó P cố gắng bắt E còn E tìm cách tránh P. Trò chơi kết thúc khi E bị P bắt (tức là 2 người chơi đứng trên cùng 1 đỉnh của đồ thị)



P được khởi tạo tại đỉnh b còn E được khởi tạo tại đỉnh d. Hình b là cây biểu diễn trò chơi, trong đó mỗi node được gán nhãn là vị trí tương ứng của P và E hiện tại. P là người đi trước.

Giá trị của các node cuối của cây (terminals, tại đó P thắng) = - (số bước mà P đã đi)

- a. Hãy điền giá trị cho các node terminal trong cây trên
- b. Hãy điền giá trị cho các node bên trên (có thể là 1 số hoặc miền giá trị, bỏ qua các dấu ?)
- c. Đánh nhãn cho các node ở dấu “?”
- d. Hãy suy luận để chặn miền giá trị cho các node vừa điền trong phần c. (Gợi ý: Dựa vào độ dài đường đi ngắn nhất giữa 2 người chơi tại thời điểm đang xét, và lưu ý là chi phí mà đi từ node gốc tới node terminal cũng chính là chi phí để P thắng)
- e. Sau khi đã hoàn thành phần d, ta đã biết tất cả các giá trị / miền giá trị các nodes của cây. xét thứ tự đánh giá từ trái qua phải. Hãy cắt tỉa những nhánh mà không cần xét đến
- f. Có thể biết được rằng ai sẽ thắng nếu đồ thị là một cây hay không? (hình a là 1 ví dụ điển hình)

Bài 8: Xét bài toán đặt k quân mã trên bàn cờ vua n x n sao cho không có 2 quân nào ăn nhau, trong đó k đã biết và k $\leq n$

- a. Phát biểu một CSP. Các biến được dùng ở đây là gì?
- b. Giá trị có thể của mỗi biến là gì?
- c. Tập các biến được ràng buộc là gì? Và ràng buộc như thế nào?
- d. Bây giờ xét bài toán đặt nhiều quân mã nhất có thể lên bàn cờ mà không có 2 quân mã nào ăn nhau. Giải thích cách giải bài toán này với local search bằng cách định nghĩa các hàm ACTIONS và RESULT phù hợp

Bài 9: Giải bài toán mập mã số học (slide 10 – Lecture 5) bằng tay, dùng chiến lược backtracking với các kỹ thuật forward checking và Minimum Remaining Values (MRV) cùng với giá trị ràng buộc ít nhất (least-constraining-values)

Bài 10: Xét đồ thị với 8 nodes A1, A2, A3, A4, H, T, F1, F2. Ai được nối với Ai+1 với mọi i, mỗi Ai được nối với H, H được nối với T và T được nối với mỗi Fi. Hãy giải bài toán tô màu đồ thị này bằng 3 màu bằng tay theo các chiến lược: backtracking với conflict-directed backjumping, thứ tự các biến là A1, H, A4, F1, A2, F2, A3, T và thứ tự giá trị màu là R, G, B

Bài 11: Dùng giải thuật AC-3 để chỉ ra rằng phù hợp cạnh có thể tìm ra những điểm không phù hợp trong phép gán {WA = green, V=red} cho bài toán tô màu bản đồ (slide 5, lecture 5)

Bài 12: Giải thuật AC-3 đặt ngược trở lại vào hàng đợi mỗi cạnh (X_k, X_i) khi một giá trị bất kì được xóa trong miền của X_i , cho dù mỗi giá trị của X_k là phù hợp với các giá trị còn lại trong X_i . Giả sử rằng, với mỗi cạnh (X_k, X_i), chúng ta vẫn giữ lại số các giá trị còn lại của X_i mà phù hợp với mỗi giá trị của X_k . Giải thích cách update những số này một cách hiệu quả và vì vậy chỉ ra phù hợp cạnh có thể được cải thiện với thời gian tính là $O(n^2d^2)$

Phần II: Suy diễn logic

Bài 13: Các trường hợp nào sau đây là đúng:

- c. $(A \wedge B) \models (A \Leftrightarrow B)$.
- d. $A \Leftrightarrow B \models A \vee B$.
- e. $A \Leftrightarrow B \models \neg A \vee B$.
- f. $(A \wedge B) \Rightarrow C \models (A \Rightarrow C) \vee (B \Rightarrow C)$.
- g. $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$.
- h. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B)$.
- i. $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B) \wedge (\neg D \vee E)$.
- j. $(A \vee B) \wedge \neg(A \Rightarrow B)$ is satisfiable.
- k. $(A \Leftrightarrow B) \wedge (\neg A \vee B)$ is satisfiable.

Bài 14: Kiểm tra xem các câu (sentence) sau đây là đúng đắn, không thỏa mãn được hoặc không rơi vào 2 trường hợp này. Dùng bảng chân lý hoặc các luật tương đương logic

- a. $\text{Smoke} \Rightarrow \text{Smoke}$
- b. $\text{Smoke} \Rightarrow \text{Fire}$
- c. $(\text{Smoke} \Rightarrow \text{Fire}) \Rightarrow (\neg \text{Smoke} \Rightarrow \neg \text{Fire})$
- d. $\text{Smoke} \vee \text{Fire} \vee \neg \text{Fire}$
- e. $((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire}) \Leftrightarrow ((\text{Smoke} \Rightarrow \text{Fire}) \vee (\text{Heat} \Rightarrow \text{Fire}))$
- f. $(\text{Smoke} \Rightarrow \text{Fire}) \Rightarrow ((\text{Smoke} \wedge \text{Heat}) \Rightarrow \text{Fire})$
- g. $\text{Big} \vee \text{Dumb} \vee (\text{Big} \Rightarrow \text{Dumb})$

Bài 15: Cho tập các câu sau:

$$S1: A \Leftrightarrow (B \vee E).$$

$$S2: E \Rightarrow D.$$

$$S3: C \wedge F \Rightarrow \neg B.$$

$$S4: E \Rightarrow B.$$

$$S5: B \Rightarrow F.$$

$$S6: B \Rightarrow C$$

Chứng minh mệnh đề $\neg A \wedge B$ dùng hợp giải

Bài 16: Cho phát biểu sau: “Một người mà ở trong đảng (R) có thể được bầu cử (E) nếu như người đó đáng tin cậy (C). Ngược lại, người đó sẽ không được bầu cử”

a. Câu nào dưới đây biểu diễn đúng phát biểu trên

- (i) $(R \wedge E) \iff C$
- (ii) $R \Rightarrow (E \iff C)$
- (iii) $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

b. Câu nào trong (a) có thể được biểu diễn dưới dạng chuẩn Horn

Bài 17: Xét câu sau:

$$[(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party)] \Rightarrow [(Food \wedge Drinks) \Rightarrow Party]$$

- a. Dùng bảng chân lý để xác định xem liệu sentence này là đúng đắn, thỏa mãn (nhưng không đúng đắn) hay không thỏa mãn được
- b. Chuyển đổi về bên trái và về bên phải phép kéo theo về dạng chuẩn CNF, viết đầy đủ mỗi bước và giải thích kết quả để xác nhận kết quả câu a
- c. Dùng hợp giải để chứng minh kết luận trong câu a

Bài 18:

18.1 Cho: (1) a (2) $a \rightarrow b$ (3) $b \rightarrow (c \rightarrow d)$ (4) c Chứng minh d	18.2 Cho: (1) $a \wedge b \rightarrow c$ (2) $b \wedge c \rightarrow d$ (3) a (4) b Chứng minh d
18.3 Cho: (1) p (2) $p \rightarrow q$ (3) $q \wedge r \wedge s \rightarrow t$ (4) $p \rightarrow u$ (5) $v \rightarrow w$ (6) $u \rightarrow v$ (7) $v \rightarrow t$ (8) r (9) s Chứng minh: t	18.4 Cho: (1) $(a \vee b) \wedge c \rightarrow (c \wedge d)$ (2) $a \wedge m \wedge d \rightarrow f$ (3) $m \rightarrow b \wedge c$ (4) $a \rightarrow c$ (5) $(a \wedge f) \rightarrow (\neg e \vee g)$ (6) $(m \wedge f) \rightarrow g$ (7) a Chứng minh g

Bài 19: Cho các câu sau

1. $a_1 \vee a_2 \Rightarrow a_3 \vee a_4$
2. $a_1 \Rightarrow a_5$
3. $a_2 \wedge a_3 \Rightarrow a_5$
4. $a_2 \wedge a_4 \Rightarrow a_6 \wedge a_7$
5. $a_5 \Rightarrow a_7$
6. $a_1 \wedge a_3 \Rightarrow a_6 \vee a_7$

Cho các mệnh đề a_1, a_2 đúng

- a. Đưa các biểu thức trên về dạng chuẩn CNF
- b. Dùng hợp giải chứng minh a_7 đúng

Bài 20: Suy diễn tiến

- a. Cho tập giả thiết $GT = \{a, b, m_a\}$. Tìm $KL = \{h_c\}$. Cho tập các luật sau:

- | | |
|------------------------------|---------------------------|
| 1. $a, b, m_a \rightarrow c$ | 6. $a, B \rightarrow h_c$ |
| 2. $a, b, c \rightarrow A$ | 7. $A, B \rightarrow C$ |
| 3. $b, A \rightarrow h_c$ | 8. $B, C \rightarrow A$ |
| 4. $a, b, c \rightarrow B$ | 9. $A, C \rightarrow B$ |
| 5. $a, b, c \rightarrow C$ | |

- b. Cho $GT = \{a\}$ và tập luật: 1. $a \rightarrow b$ 2. $b \rightarrow c$ 3. $c \rightarrow d$ 4. $a \rightarrow u$
Tìm $KL = \{u\}$

Bài 21: Suy diễn lùi

- a. Chứng minh bài 20 phần a bằng suy diễn lùi

- | | |
|------------------------------|----------------------|
| b. Cho $GT = \{a, b, m_a\}$ | c. Cho $GT = \{a\}$ |
| 1. $a, b, m_a \rightarrow c$ | 1. $a \rightarrow b$ |
| 2. $a, b, C \rightarrow s$ | 2. $d \rightarrow c$ |
| 3. $a, s \rightarrow h_a$ | 3. $c \rightarrow u$ |
| 4. $b, s \rightarrow h_b$ | 4. $a \rightarrow m$ |
| 5. $c, s \rightarrow h_c$ | 5. $b \rightarrow n$ |
| 6. $a, B \rightarrow h_c$ | 6. $m \rightarrow p$ |
| 7. $a, b, c \rightarrow B$ | 7. $p \rightarrow q$ |
| Tìm $KL = \{h_c\}$ | 8. $q \rightarrow u$ |

Tìm $KL = \{u\}$

Phần I: Giải quyết vấn đề

Bài 1: Cho bản đồ của Romania

1.1. Thực hiện các giải thuật tìm kiếm

1. **Breadth-First Search (BFS)**: This algorithm explores all nodes at the present depth before moving on to nodes at the next depth level.
2. **Uniform-Cost Search (UCS)**: This algorithm expands the node with the lowest path cost.
3. **Depth-First Search (DFS)**: This algorithm explores as far as possible along each branch before backtracking.
4. **Iterative Deepening Search (IDS)**: This algorithm repeatedly applies depth-limited search with increasing limits.

1.2. Thực hiện tìm kiếm theo giải thuật tham lam (GBFS) và giải thuật A*

1. **Greedy Best-First Search (GBFS)**: This algorithm expands the node that appears to be closest to the goal, using the heuristic function $h(n)$.
2. **A***: This algorithm uses both the actual cost to reach the node and the estimated cost to get from the node to the goal.

Bài 2: Cho bản đồ Việt Nam

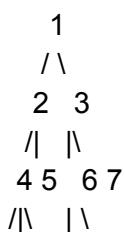
Thực hiện các giải thuật tìm kiếm từ Hà Nội (HN) đến Vinh (V):

1. **Breadth-First Search (BFS)**
2. **Uniform-Cost Search (UCS)**
3. **Depth-First Search (DFS)**
4. **Iterative Deepening Search (IDS)**
5. **Greedy Best-First Search (GBFS)**
6. **A***

Bài 3: Cây trạng thái với các số từ 1 đến 15

a. Vẽ cây biểu diễn cho không gian trạng thái

...



8 9 10 11 12

...

****b. Phát biểu bài toán****

- Trạng thái bắt đầu: 1
- Hàm chuyển trạng thái: $k \rightarrow \{2k, 2k+1\}$
- Trạng thái mục tiêu: x (giả sử $x = 11$)
- Chi phí bước đi: 1

****c. Liệt kê theo thứ tự các node sẽ được thăm****

- **BFS**: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
- **DLS (giới hạn 3)**: 1, 2, 4, 8, 9, 5, 10, 3, 6, 11
- **IDS**: (depth 0) 1, (depth 1) 1, 2, 3, (depth 2) 1, 2, 4, 5, 3, 6, 7, (depth 3) 1, 2, 4, 8, 9, 5, 10, 3, 6, 11

****d. Sửa đổi phát biểu bài toán****

One could introduce direct links from the start state to the goal state, significantly reducing the need for search.

****e. Đề xuất giải thuật không cần tìm kiếm****

Use a mathematical approach based on binary representation. For instance, for target state 11, convert to binary (1011), navigate accordingly.

Bài 4: Hai người bạn trên bản đồ

****a. Phát biểu bài toán****

- Trạng thái đầu: (i, j)
- Hành động-hàm chuyển trạng thái: Move each friend to a neighboring city.
- Kiểm tra mục tiêu: Both friends arrive at the same city.
- Chi phí đường đi: Sum of distances traveled.

****b. Hàm ước lượng heuristic****

(i) $\sqrt{D(i, j)}$

****c. Bản đồ không có lời giải****

Yes, if the cities are arranged such that friends cannot meet in a finite number of moves.

****d. Bản đồ yêu cầu thăm 1 thành phố ít nhất 2 lần****

Yes, this could happen if there are cycles or redundant paths in the graph.

Bài 5: Tìm đường đi ngắn nhất với chướng ngại vật

a. Trạng thái và đường đi

- Số lượng trạng thái: Infinite (all points in the plane).
- Số lượng đường đi: Infinite (all possible paths).

b. Không gian trạng thái tốt

A state space consisting of the vertices of the polygons. The state space size is the number of vertices.

c. Định nghĩa các hàm cần thiết

Define an ACTIONS function that returns feasible moves from each vertex and use A* with straight-line distance heuristic.

d. Đánh giá hiệu năng các giải thuật khác

Compare efficiency and optimality of BFS, DFS, and UCS.

This is a basic outline of how to approach these problems. For detailed algorithm implementations and results, additional steps such as coding and testing the algorithms are required. Let me know if you need further details on a specific problem or the complete solution for any of these exercises.

Let's continue solving the remaining problems from the document.

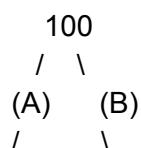
Bài 6: Trò chơi giữa hai người chơi X và Y

Phát biểu bài toán

- Trạng thái bắt đầu: $\{ N = 100 \}$
- Các hành động của X:
 - A: $\{ N := N + (N \bmod 23) - 11 \}$
 - B: $\{ N := N + (N \bmod 7) - 4 \}$
- Các hành động của Y:
 - C: $\{ N := N + 2*(N \bmod 17) - 16 \}$
 - D: $\{ N := N + ((N \bmod 11) - 5) * (N \bmod 17) \}$
- Trò chơi diễn ra trong 4 lượt: X, Y, X, Y.

a. Vẽ cây tìm kiếm và thực hiện thuật toán cắt tỉa alpha-beta

...



```
89      96  
/\      /\  
... ...  ... ...  
...  
...
```

The tree will be generated according to the moves by X and Y, alternating.

****b. Tính giá trị của node gốc của cây****

Evaluate each leaf node and backpropagate the values using alpha-beta pruning to find the optimal moves.

****c. Hướng đi của X là gì?****

After applying alpha-beta pruning, choose the move (A or B) that maximizes X's utility based on the tree evaluation.

Bài 7: Trò chơi pursuit-evasion

Phát biểu bài toán

- P (pursuer) khởi tạo tại đỉnh b
- E (evader) khởi tạo tại đỉnh d
- P cố gắng bắt E, còn E cố gắng tránh P
- Trò chơi kết thúc khi P và E ở cùng một đỉnh

****a. Điền giá trị cho các node terminal****

The terminal nodes have values equal to the negative number of steps P has taken to catch E.

****b. Đиền giá trị cho các node bên trên****

Propagate the terminal values up the tree to assign values to higher-level nodes based on min-max strategy.

****c. Đиền nhãn cho các node ở dấu "?"****

Follow the same propagation method to fill in the missing values.

****d. Chặn miền giá trị cho các node vừa điền trong phần c****

Use the shortest path between P and E to determine the range of values.

****e. Cắt tỉa các nhánh không cần xét đến****

Apply alpha-beta pruning to eliminate branches that won't affect the outcome.

****f. Ai sẽ thắng nếu đồ thị là một cây?****

If the graph is a tree, P can always catch E given enough time, because there are no cycles for E to exploit.

Bài 8: Đặt k quân mã trên bàn cờ vua n x n

Phát biểu bài toán

****a. CSP cho bài toán****

- Biến: Vị trí của mỗi quân mã
- Giá trị: Các ô trên bàn cờ

****b. Giá trị của mỗi biến****

- Các tọa độ (i, j) trên bàn cờ n x n

****c. Tập các biến ràng buộc****

- Không có hai quân mã nào ăn nhau, tức là không được cách nhau theo bước đi của mã.

****d. Đặt nhiều quân mã nhất có thể****

- Use local search with actions being placing or moving knights and result being a new configuration.

Bài 9: Giải bài toán mập mã số học

Use backtracking with forward checking and Minimum Remaining Values (MRV) to solve a cryptarithmetic puzzle. This involves systematically assigning values to variables while pruning inconsistent paths.

Bài 10: Tô màu đồ thị

Phát biểu bài toán

****a. Giải bài toán tô màu đồ thị với 3 màu****

- Sử dụng backtracking với conflict-directed backjumping
- Thứ tự các biến: A1, H, A4, F1, A2, F2, A3, T
- Thứ tự giá trị màu: R, G, B

Color each node while ensuring no two adjacent nodes share the same color.

Bài 11: AC-3 thuật toán kiểm tra tính nhất quán cạnh

Show how AC-3 can detect inconsistency in the assignment {WA = green, V = red}.

Bài 12: Cải thiện thời gian thuật toán AC-3

Explain how to efficiently update the counts of remaining values to improve the runtime of the AC-3 algorithm.

Phần II: Suy diễn logic

Bài 13-21

These problems involve logical reasoning, proof techniques, and various inference methods such as forward chaining, backward chaining, and resolution. Detailed solutions would include constructing truth tables, applying inference rules, converting logical expressions to normal forms, and using algorithms like AC-3 and others.

For more detailed solutions, particularly for the algorithmic and logical problems, implementation or extensive manual calculation may be needed. Let me know if you need further elaboration or a detailed solution for a specific problem.

Week 4 – Chapter 3.

1. Executing and understanding :

- Explain the source code of the following algorithms: A*, Uniform Cost Search, Greedy Best First Search.
- Explain why Breath first tree search takes 102 steps while Breath first graph search takes only 21 to find the way from Arad to Bucharest.
- **Detail explains the infinite loops at the Depth-First Tree Search example.**
- Informed Search Strategies - **search.ipynb**

2. Exercises:

3.3 Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.2. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

- a. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)
- b. Let $D(i, j)$ be the straight-line distance between cities i and j . Which of the following heuristic functions are admissible?
 - (i) $D(i, j)$
 - (ii) $2 \cdot D(i, j)$
 - (iii) $D(i, j)/2$.
- c. Are there completely connected maps for which no solution exists?
- d. Are there maps in which all solutions require one friend to visit the same city twice?

3.22 Compare the performance of A* and RBFS on a set of randomly generated problems in the 8-puzzle (with Manhattan distance) and TSP (with MST—see Exercise 3.30) domains. Discuss your results. What happens to the performance of RBFS when a small random number is added to the heuristic values in the 8-puzzle domain?

3.30 The traveling salesperson problem (TSP) can be solved with the minimum-spanningtree (MST) heuristic, which estimates the cost of completing a tour, given that a partial tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

- a.** Show how this heuristic can be derived from a relaxed version of the TSP.
 - b.** Show that the MST heuristic dominates straight-line distance.
 - c.** Write a problem generator for instances of the TSP where cities are represented by random points in the unit square.
 - d.** Find an efficient algorithm in the literature for constructing the MST, and use it with A* graph search to solve instances of the TSP.
- 3. Thay đổi vị trí kiểm tra goal state cho các bài toán search khác anh TA đã chưa . Chụp lại kết quả của 1 trong các thuật toán search.**
- 4. Thay đổi THỜI ĐIỂM thực hiện GOAL TEST cho các giải thuật DFS, BFS và phân tích ĐỘ PHÚC TẠP THỜI GIAN/KHÔNG GIAN của chúng.**
- 5. Giải thích và lấy ví dụ cho từng trường hợp trả về của Depth-limited search.**

TRẮC NGHIỆM

Câu 1: Trong quá trình huấn luyện mô hình phát hiện đối tượng, tác động của kĩ thuật tăng cường dữ liệu:

- a. Giảm hiệu suất của mô hình
- b. Không ảnh hưởng đến hiệu suất của mô hình
- c. Cải thiện hiệu suất của mô hình phát hiện đối tượng
- d. Phụ thuộc vào kĩ thuật tăng cường dữ liệu được sử dụng**

Câu 2: Trong thuật toán Adam, tỉ lệ học được:

- a. Thay đổi theo công thức cho trước**
- b. Thiết lập cố định
- c. Thay đổi ngẫu nhiên
- d. Không đáp án nào đúng

Câu 3: Những cách nào sau đây giúp giảm số phép toán dấu phẩy động (FLOPs) cần thực hiện của mô hình phát hiện đối tượng:

- a. Thay các lớp tích chập giảm chiều dữ liệu bằng các lớp pooling
- b. Giảm kích thước của filter trong lớp tích chập của mô hình
- c. Giảm kích thước đầu vào của mô hình phát hiện đối tượng
- d. Cả 3 đáp án trên**

a. Thay các lớp tích chập giảm chiều dữ liệu bằng các lớp pooling:
 Lớp pooling (như Max Pooling hoặc Average Pooling) thường được sử dụng để giảm kích thước của đầu ra bằng cách lấy giá trị lớn nhất hoặc trung bình trong một vùng nhỏ của đầu vào. Thay vì sử dụng lớp tích chập giảm chiều dữ liệu, ta có thể sử dụng lớp pooling để giảm số lượng phép toán FLOPs trong mô hình.

b. Giảm kích thước của filter trong lớp tích chập của mô hình: Sử dụng kích thước filter nhỏ hơn trong lớp tích chập giúp giảm số lượng phép nhân FLOPs cần thực hiện trong quá trình tính toán tích chập. Tuy nhiên, việc giảm kích thước filter cũng có thể làm giảm khả năng học các đặc trưng phức tạp, vì vậy việc chọn kích thước filter phù hợp là cần thiết.

c. Giảm kích thước đầu vào của mô hình phát hiện đối tượng: Giảm kích thước đầu vào của mô hình có thể giúp giảm số lượng phép toán FLOPs cần thực hiện trong quá trình huấn luyện và dự đoán. Tuy nhiên, việc giảm kích thước đầu vào cũng có thể làm mất mát thông tin quan trọng và ảnh hưởng đến hiệu suất của mô hình, do đó cần cân nhắc khi áp dụng phương pháp này.

Câu 4: Nhược điểm của việc áp dụng các bộ lọc có kích thước lớn (7x7, 11 x 11, 13 x 13 ...) trong lớp tích chập:

- a. Làm tăng số lượng phép toán cần thực hiện
- b. Làm chậm tốc độ xử lý của mô hình
- c. Làm tăng số lượng tham số học của mô hình
- d. **Tất cả đều đúng**
- e. Tất cả đều sai

Nhược điểm của việc áp dụng các bộ lọc có kích thước lớn (7x7, 11x11, 13x13, vv.) trong lớp tích chập trong mô hình tích chập (CNN) bao gồm:

- a. Làm tăng số lượng phép toán cần thực hiện: Kích thước lớn của bộ lọc đòi hỏi thực hiện nhiều phép toán tích chập hơn trên từng vùng của đầu vào. Điều này dẫn đến việc tăng số lượng phép nhân và cộng phải thực hiện, tăng đáng kể tổng số phép toán (FLOPs) trong mô hình. Điều này làm cho mô hình có khả năng tính toán nặng nề và tốn nhiều thời gian.
- b. Làm chậm tốc độ xử lý của mô hình: Vì số lượng phép toán tăng lên đáng kể, việc tính toán bộ lọc lớn sẽ làm chậm tốc độ xử lý của mô hình. Điều này đặc biệt quan trọng trong các ứng dụng thời gian thực hoặc trong việc triển khai mô hình trên các thiết bị có tài nguyên tính toán hạn chế.
- c. Làm tăng số lượng tham số học của mô hình: Bộ lọc lớn yêu cầu nhiều tham số học hơn so với bộ lọc nhỏ. Vì vậy, việc sử dụng các bộ lọc có kích thước lớn làm tăng số lượng tham số trong mô hình. Điều này có thể làm cho mô hình dễ bị overfitting (quá khớp) và yêu cầu nhiều dữ liệu huấn luyện hơn để đạt được hiệu suất tốt.

Câu 5: Vì sao mô hình yolo lại sử dụng cách biểu diễn bounding box theo tâm và tỉ lệ

- a. Giúp quá trình huấn luyện nhanh và dễ hội tụ
- b. Giúp mô hình ít tham số hơn
- c. Giúp mô hình phát hiện chính xác hơn
- d. **Cả 3 đáp án trên**
- e. Không đáp án nào chính xác

Mô hình YOLO (You Only Look Once) sử dụng cách biểu diễn bounding box theo tâm (center) và tỉ lệ (width, height) vì những lý do sau:

- a. Giúp quá trình huấn luyện nhanh và dễ hội tụ: Bằng cách biểu diễn bounding box theo tâm và tỉ lệ, YOLO giúp giảm khối lượng tính toán so với

việc dự đoán các điểm góc của bounding box. Điều này làm cho quá trình huấn luyện nhanh hơn và dễ dàng hội tụ đến kết quả tốt.

b. Giúp mô hình ít tham số hơn: Việc biểu diễn bounding box theo tâm và tỉ lệ giúp giảm số lượng tham số cần học trong mô hình. Điều này giúp giảm thiểu sự phức tạp của mô hình và giúp nó hoạt động hiệu quả trên các tài nguyên tính toán có hạn, đồng thời tránh hiện tượng overfitting.

c. Giúp mô hình phát hiện chính xác hơn: Cách biểu diễn bounding box theo tâm và tỉ lệ giúp mô hình có khả năng dự đoán chính xác hơn vị trí và kích thước của đối tượng. Điều này giúp cải thiện đáng kể chất lượng phát hiện đối tượng và giảm sai số trong việc dự đoán bounding box.

Câu 6: Khi kích thước ảnh đầu vào tăng lên thì số BFLOPs của mô hình CNN tăng lên hay giảm đi:

- a. Tăng
- b. Giảm
- c. Không thay đổi

BFLOPs (billion floating-point operations per second) là một đơn vị đo lường tốn kém tính toán của mô hình CNN (Convolutional Neural Network). Khi kích thước ảnh đầu vào tăng lên, số lượng phép tính dấu phẩy động (FLOPs) của mô hình CNN thường sẽ tăng lên. Điều này xảy ra vì mỗi phần tử trong ma trận đầu vào phải tham gia vào các phép toán tích chập và pooling trong quá trình lan truyền thuận qua mạng CNN. Kích thước lớn hơn của ảnh đầu vào cũng tương đương với việc có nhiều thông tin hơn trong mỗi lần xử lý, do đó, số lượng phép toán tính toán cần thực hiện cũng tăng lên.

Câu 7: Mục đích sử dụng hàm kích hoạt là gì ?

- a. Phá vỡ tính tuyến tính của mạng nơron
- b. Cho phép mạng nơron biểu diễn được các hàm phi tuyến phức tạp
- c. **Cả 2 đáp án trên**
- d. Không đáp án nào chính xác

Mục đích sử dụng hàm kích hoạt trong mạng nơron là cả phá vỡ tính tuyến tính của mạng và cho phép mạng biểu diễn được các hàm phi tuyến phức tạp.

1. Phá vỡ tính tuyến tính: Nếu không sử dụng hàm kích hoạt, mạng nơron chỉ sẽ thực hiện các phép toán tuyến tính (như tích chập và ma trận trọng số) và kết quả của nó cũng chỉ là một phép toán tuyến tính. Điều này giới hạn khả năng của mạng trong việc học các biểu diễn phi tuyến phức tạp và giải quyết các bài toán phức tạp hơn.
2. Biểu diễn hàm phi tuyến phức tạp: Bằng cách sử dụng hàm kích hoạt phi tuyến, như ReLU (Rectified Linear Unit), Sigmoid, Tanh, Leaky ReLU, ELU (Exponential Linear Unit),... mạng nơron có khả năng biểu diễn các hàm phi tuyến phức tạp, từ đó nó có thể học được các biểu diễn non-

linear của dữ liệu và giải quyết các bài toán phức tạp như phân loại, phát hiện đối tượng, và dự đoán chuỗi thời gian.

Câu 8: Vì sao cần chia tập dữ liệu thành tập huấn luyện và kiểm tra

- a. Để đánh giá chính xác và khách quan hiệu quả của mô hình đã huấn luyện
- b. Để giảm thời gian huấn luyện
- c. Để giảm kích thước tập dữ liệu huấn luyện
- d. **Tất cả đáp án trên**

Câu 9: Những cách nào sau đây giúp tăng tốc độ xử lý của mô hình phát hiện đối tượng:

- e. Giảm số lớp của mô hình backbone
 - f. Giảm số filter trong lớp convolution của mô hình backbone
 - g. Giảm kích thước đầu vào của mô hình phát hiện đối tượng
 - h. **Cả 3 đáp án trên**
 - i. Không đáp án nào chính xác
- a. Giảm số lớp của mô hình backbone: Bằng cách giảm số lớp của mô hình backbone, ta giảm đi sự phức tạp của mô hình, từ đó giúp tăng tốc độ xử lý.
- b. Giảm số filter trong lớp convolution của mô hình backbone: Bằng cách giảm số lượng filter trong các lớp convolution, ta giảm số lượng tham số và phép tính được thực hiện trong mô hình, từ đó cũng giúp tăng tốc độ xử lý.
- c. Giảm kích thước đầu vào của mô hình phát hiện đối tượng: Bằng cách giảm kích thước đầu vào, mô hình phải thực hiện ít phép tính hơn, từ đó giúp tăng tốc độ xử lý.
- Tuy nhiên, cần lưu ý rằng việc giảm các yếu tố như số lớp, số filter hay kích thước đầu vào có thể làm giảm độ chính xác và khả năng phát hiện của mô hình, do đó cần cân nhắc kỹ càng trước khi thực hiện.

Câu 10: Những cách nào sau đây giúp tăng độ chính xác của mô hình phát hiện đối tượng:

- a. Tăng cường dữ liệu
- b. Thay đổi mô hình backbone
- c. Thay đổi hàm mất mát
- d. **Cả 3 đáp án trên**
- e. Không đáp án nào chính xác

- a. Tăng cường dữ liệu: Bằng cách tăng cường dữ liệu, ta tạo thêm các biến thể của dữ liệu huấn luyện bằng cách thêm các phép biến đổi như xoay, lật, thu phóng, hay làm mờ ảnh. Điều này giúp mô hình học được các đặc trưng tổng quát hơn và tăng khả năng phát hiện đối tượng trong các tình huống khác nhau.
- b. Thay đổi mô hình backbone: Mô hình backbone là phần cơ bản của mô hình phát hiện đối tượng, có nhiệm vụ trích xuất các đặc trưng của ảnh. Thay đổi mô hình backbone có thể giúp mô hình học được các đặc trưng phức tạp hơn, từ đó tăng độ chính xác của việc phát hiện đối tượng.
- c. Thay đổi hàm mất mát: Hàm mất mát là cách để đo lường sự sai khác giữa dự đoán của mô hình và nhãn thực tế. Thay đổi hàm mất mát có thể giúp tối ưu hóa mô hình cho các trường hợp cụ thể, từ đó tăng độ chính xác của mô hình.

TƯ LUẬN

Câu 1: Tính output size của model, biết rằng input của model là 224x224x3 và convolution layer sử dụng zero-padding = 1 và stride = 1

Type	Filters	Size/Stride
Convolutional	32	3×3
Maxpool		$2 \times 2/2$
Convolutional	64	3×3
Maxpool		$2 \times 2/2$
Convolutional	128	3×3
Convolutional	64	1×1
Convolutional	128	3×3
Maxpool		$2 \times 2/2$
Convolutional	256	3×3
Convolutional	128	1×1
Convolutional	256	3×3
Maxpool		$2 \times 2/2$
Convolutional	512	3×3
Convolutional	256	1×1
Convolutional	512	3×3
Convolutional	256	1×1
Convolutional	512	3×3
Maxpool		$2 \times 2/2$
Convolutional	1024	3×3
Convolutional	512	1×1
Convolutional	1024	3×3

Qua mỗi lớp convolution output size ko đổi ; qua maxpool thì giảm đi 2

Ouput size : 224/32 = 7

Output Shape 7*7*1024

Câu 2: Tính số Flops và số parameter của mô hình ở câu 1

Câu 3: Trong mạng Neural, làm thế nào biết được đặc trưng nào có ý nghĩa và đặc trưng nào không có ý nghĩa khi thực hiện bài toán phân loại

Trong mạng Neural, việc biết được đặc trưng nào có ý nghĩa và đặc trưng nào không có ý nghĩa là một phần quan trọng của quá trình huấn luyện và đánh giá mô hình phân loại. Dưới đây là một số phương pháp và kỹ thuật giúp bạn hiểu về đặc trưng và quyết định đặc trưng nào nên được sử dụng trong bài toán phân loại:

1. Phân tích đặc trưng (Feature analysis): Phân tích sâu các đặc trưng đã được trích xuất từ dữ liệu và hiểu ý nghĩa của chúng trong việc phân loại. Nếu một đặc trưng có xu hướng phân tách các nhóm dữ liệu, nó có thể là một đặc trưng quan trọng.
2. Heatmap và Visualizations: Sử dụng heatmap hoặc biểu đồ tương tác (interaction plots) để kiểm tra sự tương quan giữa các đặc trưng và nhãn phân loại. Điều này giúp xác định xem các đặc trưng có mối quan hệ nào với kết quả phân loại.
3. Quan sát mô hình: Hiểu cách mô hình học và sử dụng các kỹ thuật như mô tả đường cong (ROC curve) và đường cong tự tin (Precision-Recall curve) để đánh giá hiệu suất mô hình với từng đặc trưng riêng lẻ.
4. Đánh giá đặc trưng theo đóng góp (Feature Importance): Sử dụng các thuật toán như Random Forest hoặc Gradient Boosting để tính toán đóng góp của từng đặc trưng trong việc xây dựng mô hình. Các đặc trưng có đóng góp lớn hơn thường có ý nghĩa hơn trong việc phân loại.
5. Kỹ thuật lựa chọn đặc trưng (Feature Selection): Áp dụng các kỹ thuật lựa chọn đặc trưng để chọn ra những đặc trưng quan trọng nhất và loại bỏ các đặc trưng không có ý nghĩa hoặc gây nhiễu. Các kỹ thuật này có thể là Univariate Feature Selection, Recursive Feature Elimination (RFE), hoặc L1 regularization (LASSO).
6. Phân tích ma trận hiệp phương sai (Covariance matrix analysis): Xem xét ma trận hiệp phương sai của dữ liệu để tìm ra các đặc trưng có sự tương quan cao với nhau. Điều này giúp xác định xem có những đặc trưng nào có thể loại bỏ hoặc kết hợp lại vì chúng mang thông tin tương tự.
7. Cross-validation: Sử dụng kỹ thuật cross-validation để đánh giá hiệu suất của mô hình với từng tập hợp đặc trưng khác nhau. Quá trình này sẽ giúp bạn xác định xem việc sử dụng các đặc trưng nào sẽ dẫn đến một mô hình tốt hơn.

Câu 4: Trong các ứng dụng thực tế của bài toán object detection, khi nào chúng ta sử dụng ảnh đầu vào là ảnh xám (grayscale)

Tra ChatGPT

Câu 5: Trong bài toán phát hiện đối tượng, bạn cần làm gì nếu dữ liệu huấn luyện thu thập được ít ?

Trong bài toán phát hiện đối tượng (object detection), dữ liệu huấn luyện đóng vai trò cực kỳ quan trọng để mô hình có thể học và hiểu các đối tượng cần phát hiện. Khi dữ liệu huấn luyện được thu thập ít, bạn cần xem xét và thực hiện một số biện pháp để tối ưu hóa việc huấn luyện mô hình. Dưới đây là một số gợi ý:

1. **Thu thập thêm dữ liệu:** Cố gắng tăng cường dữ liệu huấn luyện bằng cách thu thập thêm ảnh, video hoặc các tập dữ liệu tương tự từ các nguồn khác nhau. Điều này giúp mô hình tiếp cận được nhiều biến thể của đối tượng cần phát hiện.
2. **Augmentation (Tăng cường dữ liệu):** Sử dụng kỹ thuật tăng cường dữ liệu để tạo ra nhiều biến thể của các ảnh huấn luyện hiện có. Các biến thể này có thể là việc xoay, phóng to, thu nhỏ, làm mờ, thay đổi độ sáng, hay lật ảnh. Việc làm này giúp tăng cường và đa dạng hóa dữ liệu huấn luyện mà không phải tốn nhiều chi phí để thu thập dữ liệu mới.
3. **Transfer learning (Học chuyển giao):** Sử dụng mô hình đã được huấn luyện trước đó trên một tác vụ tương tự để làm mô hình khởi đầu cho bài toán phát hiện đối tượng. Bằng cách sử dụng mô hình này làm điểm khởi đầu, bạn có thể cải thiện hiệu suất của mô hình mới trong điều kiện dữ liệu huấn luyện ít.
4. **Fine-tuning (Tinh chỉnh mô hình):** Nếu bạn có một mô hình hoặc một kiến trúc mạng đã được huấn luyện trên dữ liệu lớn khác, hãy tiến hành tinh chỉnh mô hình này trên tập dữ liệu huấn luyện ít của bài toán phát hiện đối tượng cụ thể.
5. **Sử dụng các kiến trúc mô hình thích hợp:** Nếu dữ liệu huấn luyện ít, hãy chọn các kiến trúc mô hình nhẹ và dễ huấn luyện để tránh overfitting và tối ưu hóa hiệu suất mô hình.
6. **Sử dụng mô hình được điều chỉnh trước (Pre-trained models):** Sử dụng các mô hình được huấn luyện sẵn trên tập dữ liệu lớn (như ImageNet) và sau đó điều chỉnh lại cho bài toán phát hiện đối tượng của bạn. Việc này giúp mô hình có khả năng tổng quát hóa tốt hơn, nhất là khi dữ liệu huấn luyện ít.
7. **Regularization (Chính quy hóa):** Sử dụng các kỹ thuật chính quy hóa như L1 hoặc L2 regularization để giảm overfitting và tăng khả năng tổng quát hóa của mô hình.
8. **Kiểm tra các mô hình phức tạp hơn:** Trong trường hợp dữ liệu huấn luyện ít, hạn chế việc sử dụng các mô hình phức tạp và sâu, vì chúng dễ dẫn đến overfitting. Thay vào đó, tập trung vào các mô hình đơn giản và nhẹ hơn có khả năng học tốt với ít dữ liệu.

Câu 6: Vì sao hiện nay các giải thuật one stage object detection được ưa chuộng hơn là two stage object detection

Hiện nay, các giải thuật one-stage object detection đang được ưa chuộng hơn so với two-stage object detection vì nhiều lý do sau đây:

1. **Tốc độ:**

- One-stage object detection thường nhanh hơn two-stage object detection. Với một giai đoạn (one-stage), mô hình dự đoán các bounding box và nhãn trực tiếp từ các vị trí cụ thể trên hình ảnh, điều này giúp tiết kiệm thời gian và tăng tốc độ xử lý.
- Two-stage object detection có thể chậm hơn vì phải thực hiện giai đoạn "tìm kiếm" (search) để đề xuất các vùng chứa đối tượng trước khi dự đoán bằng giai đoạn thứ hai.

2. Đơn giản hóa:

- One-stage object detection đơn giản hơn về kiến trúc so với two-stage object detection. Bằng cách giảm bớt một giai đoạn, mô hình trở nên dễ triển khai và hiểu quả hơn.
- Two-stage object detection có xu hướng sử dụng các thành phần phức tạp như region proposal network (RPN) để đề xuất vùng chứa đối tượng, điều này đòi hỏi thêm tính toán và đồng thời làm tăng phức tạp cho mô hình.

3. Hiệu suất:

- Trong những năm gần đây, các mô hình one-stage object detection đã đạt được hiệu suất rất tốt và tiệm cận với hoặc thậm chí vượt qua two-stage object detection. Các phương pháp one-stage như YOLO (You Only Look Once) và SSD (Single Shot Multibox Detector) đã chứng minh khả năng phát hiện đối tượng hiệu quả và chính xác trong thời gian thực.
- Việc cải thiện hiệu suất của one-stage object detection đã giúp giảm sự cần thiết của two-stage methods.

Tuy nhiên, không phải lúc nào one-stage object detection cũng vượt trội hơn two-stage object detection. Two-stage object detection vẫn có những ưu điểm riêng như khả năng chính xác cao hơn và khả năng tập trung vào các đối tượng nhỏ hơn. Tùy thuộc vào yêu cầu cụ thể của bài toán và tình huống sử dụng, các phương pháp một giai đoạn hoặc hai giai đoạn có thể phù hợp hơn cho mục tiêu cụ thể.

Câu 7: Kỹ thuật label smoothing là gì ?

Kỹ thuật Label Smoothing (làm mờ nhãn) là một phương pháp được sử dụng trong quá trình huấn luyện mô hình máy học, đặc biệt là trong bài toán phân loại (classification). Mục tiêu chính của kỹ thuật này là giúp cải thiện hiệu suất và độ tổng quát của mô hình, đồng thời giảm thiểu hiện tượng overfitting (quá khớp).

Trong quá trình huấn luyện mô hình phân loại, mỗi điểm dữ liệu trong tập huấn luyện sẽ được gán một nhãn (label) thể hiện lớp (class) mà nó thuộc về. Label Smoothing giới thiệu một phép biến đổi nhẹ vào nhãn trước khi sử dụng chúng để huấn luyện mô hình. Thay vì sử dụng nhãn cứng (one-hot label) như [0, 1, 0, 0] cho lớp chính xác, Label Smoothing sẽ sử dụng các giá trị mềm (soft label) gần với 0 và 1, như [0.025, 0.95, 0.025, 0.025].

Ví dụ, giả sử chúng ta có một bài toán phân loại ảnh gồm 3 lớp: "mèo," "chó," và "chim." Trong trường hợp không sử dụng Label Smoothing, mô hình sẽ nhận được nhãn cứng như sau:

- Ảnh chứa mèo: [1, 0, 0]
- Ảnh chứa chó: [0, 1, 0]
- Ảnh chứa chim: [0, 0, 1]

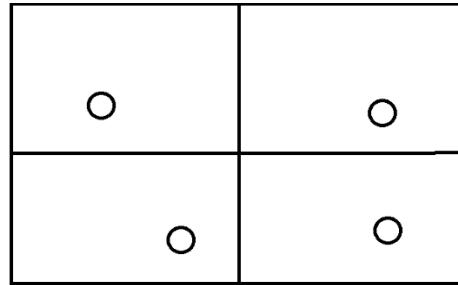
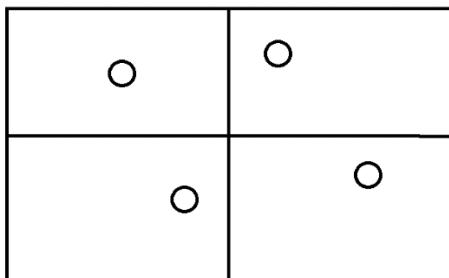
Khi sử dụng Label Smoothing, chúng ta có thể biến đổi các nhãn thành soft label như sau:

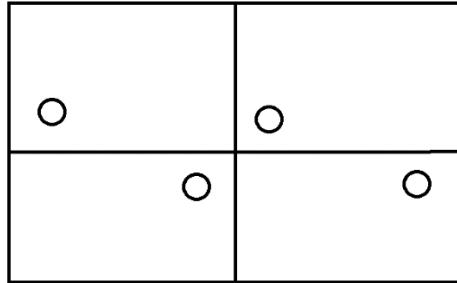
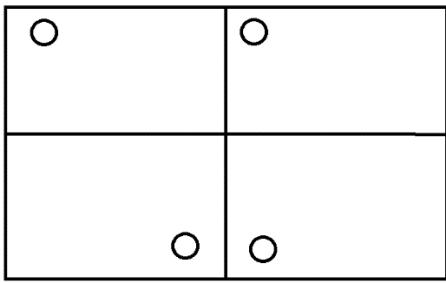
- Ảnh chứa mèo: [0.025, 0.95, 0.025]
- Ảnh chứa chó: [0.025, 0.025, 0.95]
- Ảnh chứa chim: [0.95, 0.025, 0.025]

Lý do để sử dụng Label Smoothing là để làm mờ việc quá tin cậy vào nhãn cứng và giảm thiểu overfitting. Trong nhiều trường hợp, mô hình có thể trở nên quá chính xác và "tự tin" vào các quyết định dựa trên nhãn cứng, điều này có thể dẫn đến hiện tượng overfitting. Bằng cách sử dụng soft label, mô hình sẽ tự tin hơn với các dự đoán và học cách đối phó với sự không chắc chắn trong dữ liệu, từ đó giúp tăng tính tổng quát và hiệu suất của mô hình.

Câu 8: Các thuật toán phát hiện đối tượng dùng anchor box để làm gì ?

Câu 9: Cho một tập ảnh như sau:





Biết rằng ảnh được chia ra làm 4 phần bằng nhau bởi 2 đường kẻ dọc và ngang, tại mỗi phần của ảnh có 1 hình tròn, hình tròn tại các phần là giống nhau hoàn toàn.

A. Hỏi rằng có thể áp dụng mô hình Yolo để xác định vị trí hình tròn trong ảnh và lần lượt đánh nhãn :

- class 1 cho hình tròn trong góc phần tư trên + trái
- class 2 cho hình tròn trong góc phần tư trên + phải
- class 3 cho hình tròn trong góc phần tư dưới + trái
- class 4 cho hình tròn trong góc phần tư dưới + phải

B. Có thể áp dụng kỹ thuật Mosaic để tăng cường dữ liệu trong trường hợp này ko

TỰ LUẬN

Câu 1: Khi thay đổi Stride và padding ở lớp Convolution thì ảnh hưởng đến điều gì? Và với sự thay đổi đó thì có làm thay đổi số parameters của lớp Convolution đó không

Khi thay đổi Stride và Padding ở lớp Convolution, có một số ảnh hưởng quan trọng đến quá trình xử lý và đặc điểm của mô hình CNN:

1. Stride (bước nhảy): Stride quy định khoảng cách giữa các bước trượt của bộ lọc (filter) trên đầu vào (input) khi thực hiện phép tích chập. Nếu Stride được đặt lớn hơn 1, các bước trượt sẽ thực hiện bỏ qua một số vị trí và đi tới vị trí mới, từ đó giảm kích thước đầu ra (output) của lớp Convolution. Stride càng lớn, kích thước đầu ra càng nhỏ.
2. Padding: Padding là việc thêm các giá trị số 0 xung quanh đầu vào, giúp tăng kích thước đầu vào và tránh mất thông tin cạnh (border) khi thực hiện phép tích chập. Nếu không sử dụng Padding, đầu ra của lớp Convolution sẽ giảm kích thước theo số lượng bước trượt (Stride) và số lượng lớp Convolution. Padding càng nhiều, kích thước đầu ra càng lớn.

Khi thay đổi Stride và Padding, số lượng tham số của lớp Convolution sẽ không thay đổi. Số lượng tham số trong lớp Convolution dựa vào kích thước của bộ lọc (filter) và số lượng bộ lọc được sử dụng. Stride và Padding không ảnh hưởng đến số lượng tham số.

Tuy nhiên, khi thay đổi Stride, kích thước đầu ra (output) của lớp Convolution sẽ thay đổi, điều này có thể ảnh hưởng đến kiến trúc tổng thể của mô hình, đặc biệt nếu lớp Convolution là một phần của kiến trúc mạng lớn hơn. Thay đổi kích thước đầu ra cũng có thể ảnh hưởng đến các lớp tiếp theo trong mạng, do đó, cần kiểm tra và điều chỉnh các lớp khác trong mạng để đảm bảo tính thống nhất.

Câu 2: Có thể train model NN bằng việc khởi tạo các bias = 0 ?

Có, bạn có thể khởi tạo các bias của mô hình Neural Network (NN) bằng 0. Trong NN, bias là một tham số được sử dụng để điều chỉnh đầu ra của mỗi neuron. Bias cho phép mô hình học cách phản ứng với các mẫu dữ liệu mà không cần thông qua việc truyền giá trị zero-centered qua mạng.

Khi bạn khởi tạo bias bằng 0, mô hình sẽ có xu hướng xuất ra giá trị gần 0 khi chưa được huấn luyện. Tuy nhiên, trong quá trình huấn luyện, các trọng số của mô hình sẽ được điều chỉnh để phù hợp với dữ liệu đầu vào và đầu ra mong muốn. Điều này cho phép mô hình tìm ra các giá trị bias tối ưu hơn để đạt được hiệu suất tốt hơn trong việc dự đoán.

Việc khởi tạo bias bằng 0 là một trong những phương pháp khởi tạo ban đầu đơn giản và phổ biến. Tuy nhiên, trong một số trường hợp, khởi tạo bias theo các giá trị khác nhau hoặc sử dụng các kỹ thuật khởi tạo ngẫu nhiên có thể giúp mô hình học nhanh hơn hoặc tránh được các vấn đề như đối xứng trong mạng neuron.

Tóm lại, việc khởi tạo bias bằng 0 là một cách tiếp cận hợp lý và bạn có thể huấn luyện mô hình Neural Network thành công với bias được khởi tạo như vậy.

Câu 3: Tại sao đối với bài toán ảnh chúng ta hay dùng các lớp Convolution đầu tiên thay vì sử dụng các lớp Fully Connected

<https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>

Câu 4: Làm thế nào lựa chọn 1 mô hình NN phù hợp cho một bài toán cụ thể ?

Câu 5: Hãy sắp xếp lại các bước của thuật toán Gradient Descent:

- 1. Get values from the output layer by passing the input through the neural network**
- 2. Determine the error between the actual and predicted value.**
- 3. Initialize random weight and bias.**
- 4. Repeat the feedforward and backpropagation to update parameters and stop when you find optimal parameters.**
- 5. Update the weight and bias.**

31254

Câu 6: Làm thế nào để giải quyết vấn đề “Constant validation accuracy” ?

<https://www.tutorialspoint.com/fixing-constant-validation-accuracy-in-cnn-model-training>

Câu 7: Vì sao dropout lại có thể giải quyết tránh được hiện tượng overfitting ?

Dropout là một kỹ thuật được sử dụng để giải quyết vấn đề overfitting trong mô hình Neural Network. Overfitting xảy ra khi mô hình quá tương thích với dữ liệu huấn luyện cụ thể mà không tổng quát hóa tốt cho dữ liệu mới. Dropout có thể giảm hiện tượng overfitting nhờ các đặc điểm sau:

1. **Regularization:** Dropout có tác dụng như một kỹ thuật regularization trong quá trình huấn luyện mô hình. Khi áp dụng dropout, một phần ngẫu nhiên các đơn vị (neuron) trong mạng sẽ bị tắt (bị loại bỏ) trong mỗi lần chạy qua một mẫu dữ liệu. Việc loại bỏ các đơn vị này đồng nghĩa với việc mô hình phải học cách đáp ứng với sự vắng mặt của một số phần tử, từ đó giúp giảm khả năng overfitting.
2. **Tính ngẫu nhiên:** Dropout làm cho mỗi lần chạy qua một mẫu dữ liệu, mô hình được huấn luyện trên một mạng con ngẫu nhiên được tạo ra bằng cách tắt một số đơn vị. Điều này làm cho quá trình huấn luyện trở nên ngẫu nhiên và đa dạng hơn, từ đó giúp tránh việc mô hình nhớ rất chính xác các mẫu dữ liệu huấn luyện và giúp nó tổng quát hóa tốt hơn cho dữ liệu mới.
3. **Tạo ra mô hình ensemble:** Khi sử dụng dropout, mô hình tạo ra một tập hợp các mạng con khác nhau trong quá trình huấn luyện, với mỗi mạng con tương ứng với một lần chạy qua một mẫu dữ liệu. Khi đưa ra dự đoán trên dữ liệu mới, dropout sẽ áp dụng trạng thái dropout tương tự như trong quá trình huấn luyện. Việc này tạo ra một mô hình ensemble, kết hợp dự đoán từ các mạng con khác nhau, từ đó giảm khả năng overfitting và cải thiện khả năng tổng quát hóa.

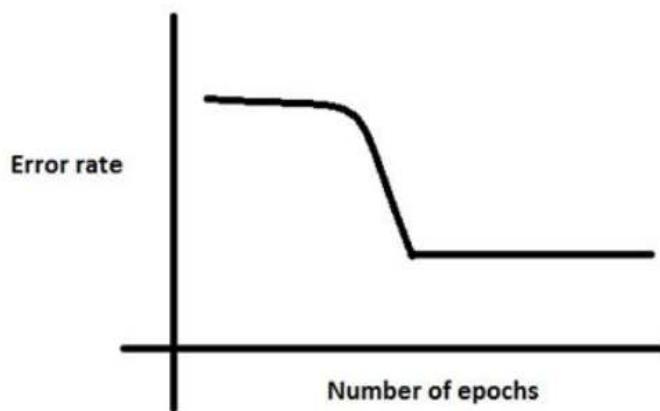
Tổng kết lại, dropout giải quyết hiện tượng overfitting trong mô hình Neural Network bằng cách áp dụng regularization, tạo tính ngẫu nhiên và tạo ra một mô hình ensemble. Qua đó, nó giúp

mô hình học cách tổng quát hóa tốt hơn cho dữ liệu mới và giảm khả năng overfitting khi mô hình quá tương thích với dữ liệu huấn luyện cụ thể.

Câu 8: Trong bài toán phân loại 5 loại bệnh (tiểu đường, béo phì, cao huyết áp, covid, viêm phổi) dựa trên 7 loại thông số đặc trưng sinh, có tổng cộng 1000 mẫu dữ liệu hỏi kích thước của tập dữ liệu là bao nhiêu nếu sử dụng kiểu one-hot encoding và label encoding cho mã hóa các loại bệnh ?

Câu 9: Nhận định sau đúng hay sai : "Convolutional Neural Networks can perform various types of transformation (rotations or scaling) in an input" (ko co trong de thi)

Câu 10: Khi quan sát quá trình huấn luyện mạng Neural Network, nguyên nhân gì có thể dẫn đến hiện tượng dưới đây:



Nghe lại lí do thứ 3

Câu 11: Nhận định nào sau đây là đúng:

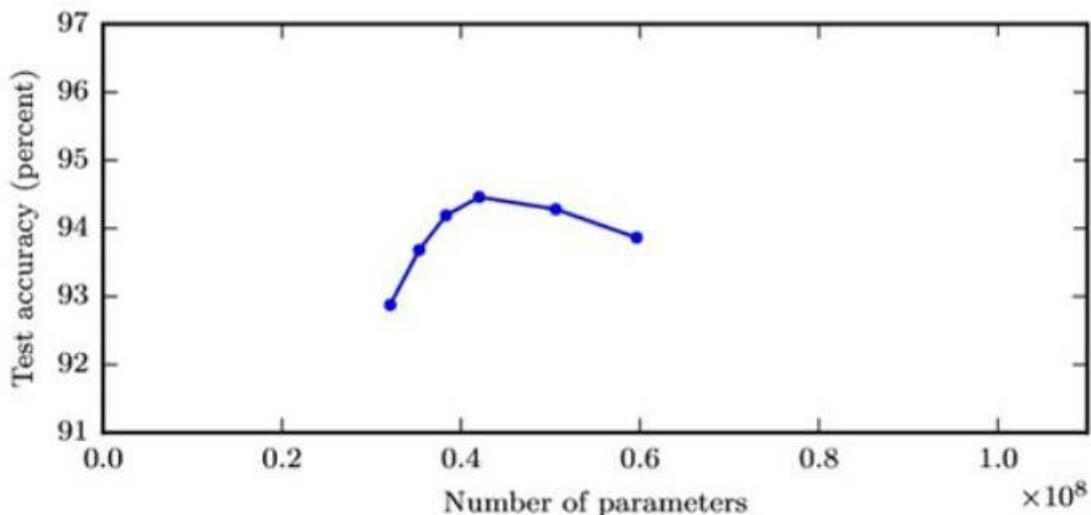
- Khi số lớp ẩn tăng thì kích thước mô hình Neural Network tăng (khi đây hidden node tăng , thì trọng số liên kết tăng)
- Khi tỉ lệ học tăng thì kích thước mô hình Neural Network tăng (nó chả liên quan đến NN)
- Khi tỉ lệ dropout tăng thì kích thước mô hình Neural Network tăng
- Cả 3 đáp án trên
- Không đáp án nào chính xác

Đáp án chính xác là e. Không có đáp án nào chính xác.

1. Khi số lớp ẩn tăng, không nhất thiết là kích thước mô hình Neural Network sẽ tăng. Số lớp ẩn chỉ ảnh hưởng đến độ phức tạp của mô hình, nhưng không đồng nghĩa với việc kích thước mô hình tăng. Kích thước của mô hình được xác định bởi số lượng neuron và trọng số trong mỗi lớp.
2. Khi tỉ lệ học tăng, không đồng nghĩa với việc kích thước mô hình tăng. Tỉ lệ học (learning rate) là tham số quyết định mức độ thay đổi của trọng số trong quá trình huấn luyện. Tuy nhiên, kích thước mô hình không phụ thuộc vào tỉ lệ học.
3. Khi tỉ lệ dropout tăng, không đồng nghĩa với việc kích thước mô hình tăng. Tỉ lệ dropout là tham số quyết định tỷ lệ đơn vị bị tắt trong quá trình huấn luyện. Tuy nhiên, kích thước mô hình không phụ thuộc vào tỉ lệ dropout.

Vì vậy, không có đáp án nào chính xác trong số a, b, c và d.

Câu 12: Cho đồ thị biểu diễn mối quan hệ giữa độ chính xác trên tập test với kích thước của mô hình (qua số parameter). Nguyên nhân vì sao dẫn đến độ chính xác giảm khi số parameter tăng.

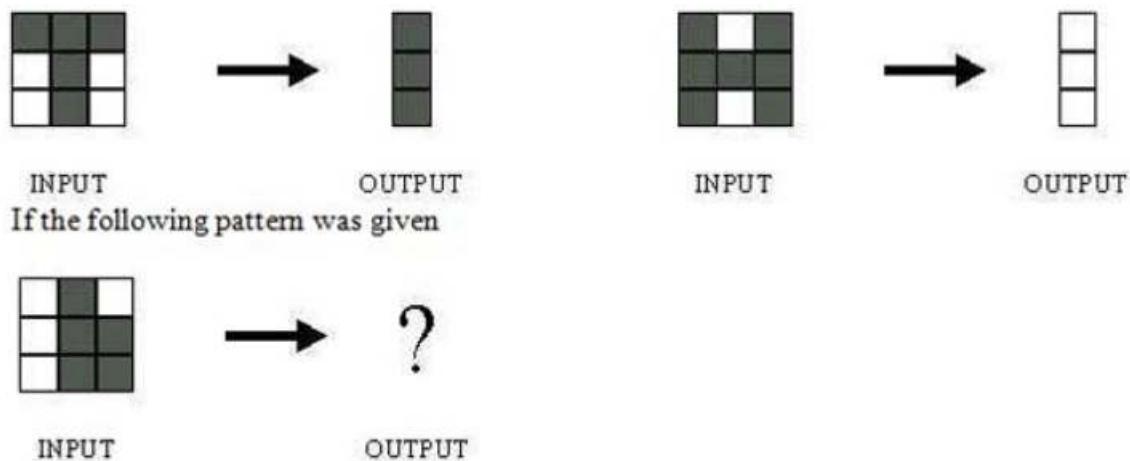


Câu 13: Trong bài toán phân loại ảnh 4 loại hoa quả bao gồm táo, cam, dưa hấu và bưởi. Tại lớp đầu ra có thể sử dụng 2 output node với activation function là sigmoid để dự đoán không.

Câu 14: Mô tả nào sau đây là chính xác nhất về kỹ thuật “Early stopping”

- a. Train the network until a local minimum in the error function is reached
- b. **Simulate the network on a test dataset after every epoch of training. Stop training when the generalization error starts to increase**
- c. Add a momentum term to the weight update in the Generalized Delta Rule, so that training converges more quickly
- d. A faster version of backpropagation, such as the 'Quickprop' algorithm

Câu 15: Cho hình minh họa dưới đây:

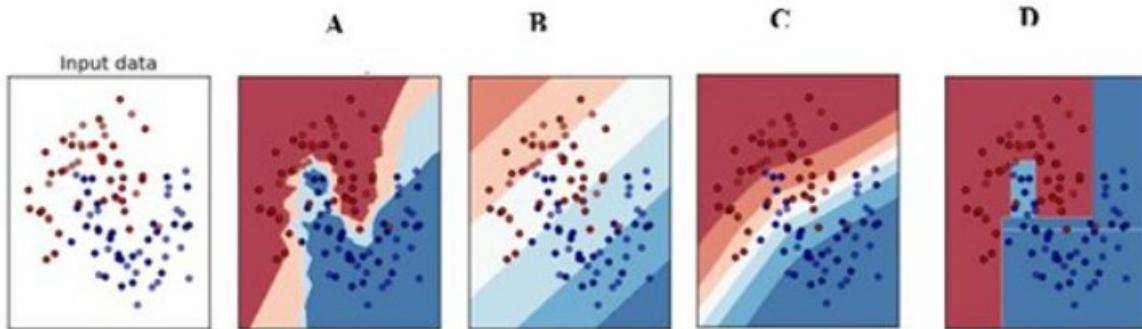


Hỏi kết quả của OUTPUT là gì :

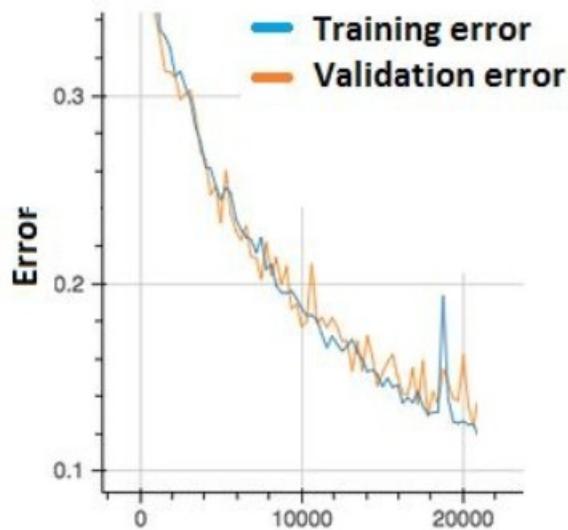
- a.
- b.
- c.
- d. Không phải đáp án trên

Câu 16: Lựa chọn nào sau đây phù hợp với biểu diễn phân lớp sử dụng Neural Network:

(Xuất hiện bài thi)



Câu 17: Trong đồ thị dưới đây, chúng ta quan sát thấy có nhiều thời điểm error tăng và giảm. Điều này có đáng lo ngại ko :

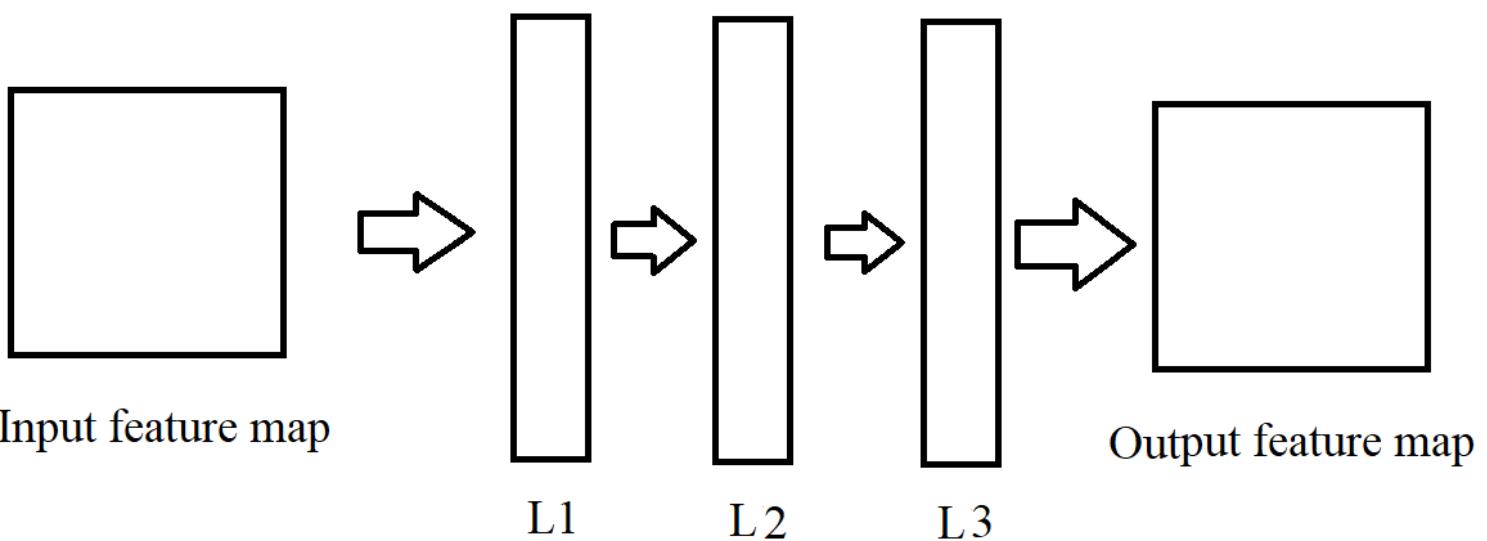


Nhìn chung là đang giảm nên ko đáng lo ngại lắm

Câu 18: Để cải thiện độ chính xác cho một mô hình CNN khi triển khai trên hệ thống nhưng có dung lượng RAM nhỏ, hướng nào sau đây hợp lí nhất:

- a. Cải thiện và tăng cường tập dữ liệu
- b. Tăng kích thước của kernel
- c. Tăng độ sâu của mô hình mạng
- d. Cả 3 đáp án trên

Câu 19: Cho mô hình mạng CNN gồm 4 layer



Biết rằng Input feature map có kích thước 35×35

L1 là lớp Atrous Convolution với 1 kernel = 3×3 , rate = 1, padding = 0, stride = 1

L2 là lớp Atrous Convolution với 1 kernel = 3x3 , rate = 2, padding = 0, stride = 1

L3 là lớp Atrous Convolution với 1 kernel = 3x3 , rate = 3, padding = 0, stride = 1

Hỏi kích thước của Output feature map bằng bao nhiêu ?

Câu 20: Trong bài toán phát hiện đuôi nước dựa trên arm detection, để trích chọn khung xuong của cánh tay thì dùng giải pháp Bottom Up hay **Top Down** hợp lí hơn

(Slide bám đối tượng và ước lượng thực tế)

Độ chính xác và tốc độ xử lý: Giải pháp Top-Down có thể hợp lý hơn nếu yêu cầu độ chính xác cao và tốc độ xử lý nhanh. Top-Down approach bắt đầu bằng việc phát hiện toàn bộ đối tượng chính (đuối nước), sau đó từ đó xác định và trích xuất các cánh tay.

Câu 21: Phân biệt Feature Learning và Feature Extraction

Câu 22: Học chuyển giao là gì ? Có phải lúc nào cũng nên sử dụng học chuyển giao không ?

Câu 23: Cho 1 tập data có 100.000 mẫu dữ liệu. Thực hiện huấn luyện mô hình trên tập data này 100 lần. Cứ 100 mẫu dữ liệu thì sẽ cập nhật trọng số 1 lần. Hỏi cần bao nhiêu iteration để hoàn thành quá trình huấn luyện này.



BK-ELEARNING MANAGEMENT SYSTEM

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Hide sidebars

BL-IT3160-144916 - Nhập môn Trí tuệ nhân tạo

Started on Sunday, 26 November 2023, 12:24 PM

State Finished

Completed on Sunday, 26 November 2023, 12:28 PM

Time taken 3 mins 30 secs

Marks 11.00/11.00

Grade 10.00 out of 10.00 (100%)

Question 1

Complete

Mark 1.00 out of
1.00

Trong học máy, điều gì là quá trình tìm kiếm các tham số tối ưu của một mô hình để giảm thiểu lỗi dự đoán?

Select one:

- a. Lập trình động (Dynamic programming)
- b. Lập trình động (Dynamic programming)
- c. Tìm kiếm thông tin (Information retrieval)
- d. Thuật toán lan truyền ngược (Backpropagation)

Question 2

Complete

Mark 1.00 out of
1.00

Sự khác biệt chính giữa bài toán phân loại và hồi quy là gì?

Select one:

- a. Hồi quy luôn là bài toán học không giám sát, trong khi phân loại là bài toán học có giám sát
- b. Hồi quy luôn là bài toán học có giám sát, trong khi phân loại là bài toán không học giám sát
- c. Đầu ra thường là dạng số thực trong bài toán hồi quy, trong khi có dạng định danh (categorical) trong bài toán phân loại
- d. Đầu ra thường là dạng định danh (categorical) trong bài toán hồi quy, trong khi có dạng số thực trong bài toán phân loại

Question 3

Complete

Mark 1.00 out of
1.00

Một máy học có thể

Select one or more:

- a. Học mà không cần dữ liệu
- b. Phán đoán mà không cần học từ dữ liệu
- c. Học một hàm mà có thể ánh xạ từ một vào tới một đầu ra
- d. Học từ một tập dữ liệu cho trước và sau đó tạo ra các phán đoán cho dữ liệu trong tương lai

Hide sidebars

Question 4

Complete

Mark 1.00 out of
1.00

Kỹ thuật học máy nào thường được sử dụng để phân loại dữ liệu thành hai lớp: "Có" và "Không"?

Select one:

- a. Học máy không giám sát
- b. Học máy không giám sát
- c. Học máy có giám sát
- d. Học máy tăng cường

Question 5

Complete

Mark 1.00 out of
1.00

Để đánh giá hiệu suất của một mô hình học máy, ta thường sử dụng các độ đo nào sau đây?

Select one or more:

- 1. Độ phức tạp tính toán
- 2. Số lần lặp và thời gian thực hiện
- 3. Độ chính xác và độ phủ
- 4. Giá trị trung bình và phương sai

Question 6

Complete

Mark 1.00 out of
1.00

Hãy chọn phát biểu đúng về Kém khớp (Underfitting)

Select one:

- a. Một hàm được gọi là bị kém khớp so với hàm khác nếu nó xấp xỉ kém chính xác hơn trên tập dữ liệu huấn luyện, nhưng chính xác hơn khi phán đoán các dữ liệu trong tương lai
- b. Một hàm được gọi là bị kém khớp so với hàm khác nếu nó xấp xỉ chính xác hơn trên tập dữ liệu huấn luyện, nhưng kém chính xác hơn khi phán đoán các dữ liệu trong tương lai
- c. Một hàm được gọi là bị kém khớp so với hàm khác nếu nó xấp xỉ kém chính xác hơn trên tập dữ liệu huấn luyện và phán đoán các dữ liệu trong tương lai kém chính xác hơn
- d. Tất cả các lựa chọn khác là sai

Question 7

Complete

Mark 1.00 out of
1.00

Học máy (Machine Learning) cung cấp các phương pháp để phân tích dữ liệu, tạo các phán đoán cho các quan sát trong tương lai

Select one:

- a. Đúng
- b. Đúng, nó còn cung cấp các phương pháp tăng tốc tính toán của máy tính
- c. Sai, nó cung cấp các phương pháp tăng tốc tính toán của máy tính

Question 8

Complete

Mark 2.00 out of
2.00

Quá khớp (Overfitting) có thể do

Select one or more:

- a. Nhiều trong dữ liệu
- b. Lựa chọn mô hình không phù hợp
- c. Mô hình đang dùng quá phức tạp
- d. Độ phức tạp của bài toán học
- e. Một lỗi nào đó trong quá trình huấn luyện

Hide sidebars

Question 9

Complete

Mark 1.00 out of
1.00**Sự khác nhau giữa học có giám sát và không giám sát ở đâu?**

Select one:

- a. Ở cách chúng ta huấn luyện một mô hình, học có giám sát đòi hỏi chúng ta phải cung cấp hướng dẫn chi tiết tại mỗi bước để máy có thể học
- b. Ở mục tiêu của thuật toán học, vì học không giám sát thường không thực hiện phán đoán nào cả
- c. Ở kiểu đầu ra, vì nó thường là các số thực trong học có giám sát
- d. Ở tập dữ liệu huấn luyện, học có giám sát thường yêu cầu biết nhãn hoặc đầu ra cho mỗi quan sát cho giai đoạn huấn luyện

Question 10

Complete

Mark 1.00 out of
1.00**Khả năng tổng quát hoá của một hàm phân loại nói về?**

Select one:

- a. Khả năng gán nhiều nhãn lớp cho một mẫu dữ liệu, từ lớp chung nhất cho đến lớp con chi tiết nhất
- b. Khả năng khớp hoàn toàn với tập huấn luyện
- c. Sự hiệu quả khi phân loại những dữ liệu mới (mà không nằm trong tập huấn luyện)
- d. Khả năng thích nghi với những nhiệm vụ mới, chẳng hạn chuyển từ nhiệm vụ phân loại sang nhiệm vụ hồi quy khi cần

[◀ Video tuần 12. Giới t...](#)[Jump to...](#)[5.2. Các chủ đề nâng c...](#)