Chương IX

HỌC MÁY (MACHINE LEARNING)

Nội dung chính: Trong chương này, chúng ta sẽ tìm hiểu về một nhánh nghiên cứu hiện đại của Trí Tuệ Nhân Tạo, đó là học máy bao gồm giải thuật ID3, mạng neuron, và giải thuật di truyền.

Mục tiêu cần đạt : Sau chương này, sinh viên có thể :

- Hiểu được mục tiêu của lĩnh vực 'máy hoc'.
- Biết 3 tiếp cận học của lĩnh vực học máy.
- Vận dụng giải thuật ID3 vào các bài toán thực tế
- Hiểu khái niệm mạng neuron và các vấn đề có liên quan
- Hiểu giải thuật di truyền và ứng dụng của nó vào các bài toán thực tế.

Kiến thức tiên quyết: Biểu diễn tri thức ở dạng luật, tìm kiếm trong không gian trạng thái, khái niệm Entropy trong Lý thuyết thông tin.

Tài liệu tham khảo :

[1] Geogre F. Luger – *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*– Addison – Wesley Publishing Company, Inc – 2002 ***(trang 349 – 381, 417 – 438, 469 – 480)***

[2] Tom M. Mitchell – *Machine Learning* – McGraw Hill, Inc ***(trang 52 – 65)***

[3] Wikipedia – Bách khoa toàn thư mở - *Học máy:* http://en.wikipedia.org/wiki/Machine_learning

[4] Ender ÖZCAN, Murat ERENTÜRK – *A Brief Review Of Memetic Algorithms For Solving TSP :* http://physics.yeditepe.edu.tr/merenturk/tainn04.ppt

I GIỚI THIỆU:

Khi được hỏi về những kỹ năng thông minh nào là cơ bản nhất đồng thời khó tự động hóa nhất của con người ngoài các hoạt động sáng tạo nghệ thuật, hành động ra quyết định mang

tính đạo đức, trách nhiệm xã hội thì người ta thường đề cập đến vấn đề ngôn ngữ và học. Trãi qua nhiều năm, hai lĩnh vực này vẫn là mục tiêu, thách thức của khoa học TTNT.

Tầm quan trọng của việc học thì không cần phải tranh cãi, vì khả năng học chính là một trong những thành tố quan trọng của hành vi thông minh. Mặc dù tiếp cận hệ chuyên gia đã phát triển được nhiều năm, song số lượng các hệ chuyên vẫn còn hạn chế. Một trong những nguyên nhân chủ yếu là do quá trình tích lũy tri thức phức tạp, chi phí phát triển các hệ chuyên gia rất cao, nhưng chúng không có khả năng học, khả năng tự thích nghi khi môi trường thay đổi. Các chiến lược giải quyết vấn đề của chúng cứng nhắc và khi có nhu cầu thay đổi, thì việc sửa đổi một lượng lớn mã chương trình là rất khó khăn. Một giải pháp hiển nhiên là các chương trình tự học lấy cách giải quyết vấn đề từ kinh nghiệm, từ sự giống nhau, từ các ví dụ hay từ những 'chỉ dẫn', 'lời khuyên',...

Mặc dù học vẫn còn là một vấn đề khó, nhưng sự thành công của một số chương trình học máy thuyết phục rằng có thể tồn tại một tập hợp các nguyên tắc học tổng quát cho phép xây dựng nên các chương trình có khả năng học trong nhiều lĩnh vực thực tế.

Chương này sẽ giới thiệu sơ lược về lĩnh vực nghiên cứu này, đồng thời đi vào chi tiết một số giải thuật học quan trọng.

I.1 Định nghĩa 'học'

Theo Herbert Simon: 'Học được định nghĩa như là bất cứ sự thay đổi nào trong một hệ thống cho phép nó tiến hành tốt hơn trong lần thứ hai khi lặp lại cùng một nhiệm vụ hoặc với một nhiệm vụ khác rút ra từ cùng một quần thể các nhiệm vụ đó'

Định nghĩa này mặc dù ngắn nhưng đưa ra nhiều vấn đề liên quan đến việc phát triển một chương trình có khả năng học. Học liên quan đến việc khái quát hóa từ kinh nghiệm: hiệu quả thực hiện của chương trình không chỉ cải thiện với 'việc lặp lại cùng một nhiệm vụ' mà còn với các nhiệm vụ tương tự. Vì những lĩnh vực đáng chú ý thường có khuynh hướng là to lớn, nên các *chương trình học* – CTH (learner) chỉ có thể khảo sát một phần nhỏ trong toàn bộ các ví dụ có thể; từ kinh nghiệm hạn chế này, CTH vẫn phải khái quát hóa được một cách đúng đắn những ví dụ chưa từng gặp trong lĩnh vực đó. Đây chính là bài toán *quy nạp* (induction), và nó chính là trung tâm của việc học. Trong hầu hết các bài toán học, dữ liệu luyện tập sẵn có thường không đủ để đảm bảo đưa ra được một khái quát hóa tối ưu, cho dù CTH sử dụng giải thuật nào. Vì vậy, các giải thuật học phải khái quát hóa theo phương pháp heuristic, nghĩa là chúng sẽ *chọn* một số khía cạnh nào đó mà theo kinh nghiệm là cho hiệu quả trong tương lai để khái quát. Các tiêu chuẩn lựa chọn này gọi là *thiên lệch quy nạp* (inductive bias).

Có nhiều nhiệm vụ học (learning task) khác nhau. Ở đây chỉ trình bày nhiệm vụ *học quy nạp* (inductive learning), đây là một trong những nhiệm vụ học cơ bản. Nhiệm vụ của CTH là học một *khái quát* (generalization) từ một tập hợp các ví dụ. *Học khái niệm* (concept learning) là một bài toán học quy nạp tiêu biểu: cho trước một số ví dụ của khái niệm, chúng ta phải suy ra một định nghĩa cho phép người dùng nhận biết một cách đúng đắn những thể hiện của khái niệm đó trong tương lai.

I.2 Các tiếp cận học:

Có ba tiếp cận học: *tiếp cận ký hiệu* (symbol-based learning), *tiếp cận mạng neuron hay kết nối* (neural or connectionist networks) và tiếp cận nổi trội (emergent) hay *di truyền và tiến hóa* (genetic and evolutionary learning).

Các CTH thuộc tiếp cận dựa trên ký hiệu biểu diễn vấn đề dưới dạng các *ký hiệu* (symbol), các giải thuật học sẽ tìm cách suy ra các *khái quát* mới, hợp lệ, hữu dụng và được biểu diễn bằng các ký hiệu này. Có nhiều giải thuật được đưa ra theo tiếp cận học này, tuy nhiên phần II của chương này chỉ trình bày một giải thuật được sử dụng rộng rãi trong số đó, đó là giải thuật quy nạp cây quyết định ID3.

Ngược lại với tiếp cận ký hiệu, tiếp cận kết nối không học bằng cách tích lũy các câu trong một ngôn ngữ ký hiệu. Giống như bộ não động vật chứa một số lượng lớn các tế bào thần kinh liên hệ với nhau, mạng neuron là những hệ thống gồm các neuron nhân tạo liên hệ với nhau. Tri thức của chương trình là ngầm định trong tổ chức và tương tác của các neuron này. Phần III sẽ đi vào chi tiết của tiếp cận này.

Tiếp cận thứ ba là tiếp cận nổi trội mô phỏng cách thức các hệ sinh học tiến hóa trong tự nhiên, nên còn được gọi là tiếp cận di truyền và tiến hóa. Phần IV sẽ đi vào chi tiết của tiếp cận này.

II TIẾP CẬN KÝ HIỆU: gIẢI THUẬT QUY NẠP CÂY QUYẾT ĐỊNH ID3

II.1 Giới thiệu

Giải thuật quy nạp cây ID3 (gọi tắt là ID3) là một giải thuật học đơn giản nhưng tỏ ra thành công trong nhiều lĩnh vực. ID3 là một giải thuật hay vì cách biểu diễn tri thức học được của nó, tiếp cận của nó trong việc quản lý tính phức tạp, heuristic của nó dùng cho việc chọn lựa các khái niệm ứng viên, và tiềm năng của nó đối với việc xử lý dữ liệu nhiễu.

ID3 biểu diễn các *khái niệm* (concept) ở dạng các cây quyết định (decision tree). Biểu diễn này cho phép chúng ta xác định phân loại của một đối tượng bằng cách kiểm tra các giá trị của nó trên một số thuộc tính nào đó.

Như vậy, nhiệm vụ của giải thuật ID3 là học cây quyết định từ một tập các ví dụ rèn luyện (training example) hay còn gọi là dữ liệu rèn luyện (training data). Hay nói khác hơn, giải thuật có:

- **Đầu vào:** Một tập hợp các ví dụ. Mỗi ví dụ bao gồm các thuộc tính mô tả một tình huống, hay một đối tượng nào đó, và một giá trị phân loại của nó.
- **Đầu ra:** Cây quyết định có khả năng phân loại đúng đắn các ví dụ trong tập dữ liệu rèn luyện, và hy vọng là phân loại đúng cho cả các ví dụ chưa gặp trong tương lai.

Ví dụ, chúng ta hãy xét bài toán phân loại xem ta 'có đi chơi tennis' ứng với thời tiết nào đó không. Giải thuật ID3 sẽ học cây quyết định từ tập hợp các ví dụ sau:

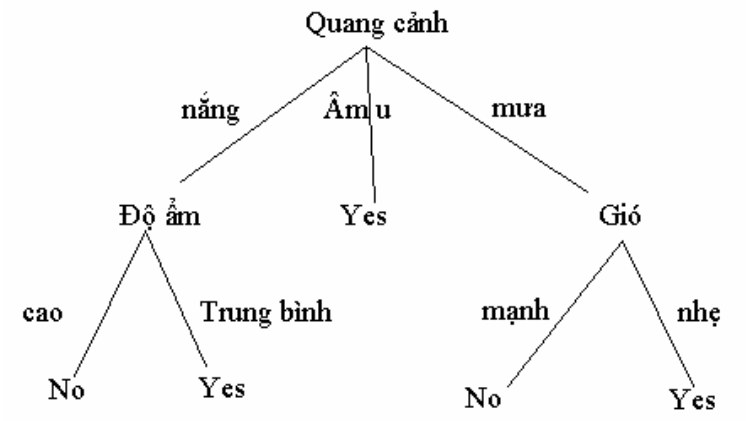
Ngày	Quang cảnh	Nhiệt độ	Độ ẩm	Gió	Chơi Tennis
D1	Nắng	Nóng	Cao	nhẹ	Không
D2	Nắng	Nóng	Cao	Mạnh	Không
D3	Âm u	Nóng	Cao	Nhẹ	Có
D4	Mưa	ấm áp	Cao	nhẹ	Có
D5	Mưa	Mát	TB	nhẹ	Có
D6	Mưa	Mát	TB	Mạnh	Không
D7	Âm u	Mát	TB	Mạnh	Có
D8	Nắng	ấm áp	Cao	nhẹ	Không
D9	Nắng	Mát	TB	nhẹ	Có
D10	Mưa	ấm áp	TB	nhẹ	Có
D11	Nắng	ấm áp	TB	Mạnh	Có
D12	Âm u	ấm áp	Cao	Mạnh	Có
D13	Âm u	Nóng	TB	nhẹ	Có
D14	Mưa	ấm áp	Cao	Mạnh	không

Bảng 9.1 *- Tập dữ liệu rèn luyện cho khái niệm 'Có đi chơi tennis không?'*

Tập dữ liệu này bao gồm 14 ví dụ. Mỗi ví dụ biểu diễn cho tình trạng thời tiết gồm các thuộc tính quang cảnh, nhiệt độ, độ ẩm và gió; và đều có một thuộc tính phân loại 'chơi Tennis' (có, không). 'Không' nghĩa là không đi chơi tennis ứng với thời tiết đó, 'Có' nghĩa là ngược lại. Giá trị phân loại ở đây chỉ có hai loại (có, không), hay còn ta nói phân loại của tập ví dụ của khái niệm này thành hai *lớp* (classes). Thuộc tính 'Chơi tennis' còn được gọi là thuộc tính đích (target attribute).

Mỗi thuộc tính đều có một tập các giá trị hữu hạn. Thuộc tính quang cảnh có ba giá trị (âm u, mưa, nắng), nhiệt độ có ba giá trị (nóng, mát, ấm áp), độ ẩm có hai giá trị (cao, TB) và gió có hai giá trị (mạnh, nhẹ). Các giá trị này chính là *ký hiệu* (symbol) dùng để biểu diễn bài toán.

Từ tập dữ liệu rèn luyện này, giải thuật ID3 sẽ học một cây quyết định có khả năng phân loại đúng đắn các ví dụ trong tập này, đồng thời hy vọng trong tương lai, nó cũng sẽ phân loại đúng các ví dụ không nằm trong tập này. Một cây quyết định ví dụ mà giải thuật ID3 có thể quy nạp được là:



Hình 9.1 *- Cây quyết định cho khái niệm 'Có đi chơi tennis không?'*

Các nút trong cây quyết định biểu diễn cho một sự kiểm tra trên một thuộc tính nào đó, mỗi giá trị có thể có của thuộc tính đó tương ứng với một nhánh của cây. Các nút lá thể hiện sự phân loại của các ví dụ thuộc nhánh đó, hay chính là giá trị của thuộc tính phân loại.

Sau khi giải thuật đã quy nạp được cây quyết định, thì cây này sẽ được sử dụng để phân loại tất cả các ví dụ hay thể hiện (instance) trong tương lai. Và cây quyết định sẽ không thay đổi cho đến khi ta cho thực hiện lại giải thuật ID3 trên một tập dữ liệu rèn luyện khác.

Ứng với một tập dữ liệu rèn luyện sẽ có nhiều cây quyết định có thể phân loại đúng tất cả các ví dụ trong tập dữ liệu rèn luyện. Kích cỡ của các cây quyết định khác nhau tùy thuộc vào thứ tự của các kiểm tra trên thuộc tính.

Vậy làm sao để học được cây quyết định có thể phân loại đúng tất cả các ví dụ trong tập rèn luyện? Một cách tiếp cận đơn giản là học thuộc lòng tất cả các ví dụ bằng cách xây dựng một cây mà có một lá cho mỗi ví dụ. Với cách tiếp cận này thì có thể cây quyết định sẽ không phân loại đúng cho các ví dụ chưa gặp trong tương lai. Vì phương pháp này cũng giống như hình thức 'học vẹt', mà cây không hề học được một khái quát nào của khái niệm cần học. Vậy, ta nên học một cây quyết định như thế nào là tốt?

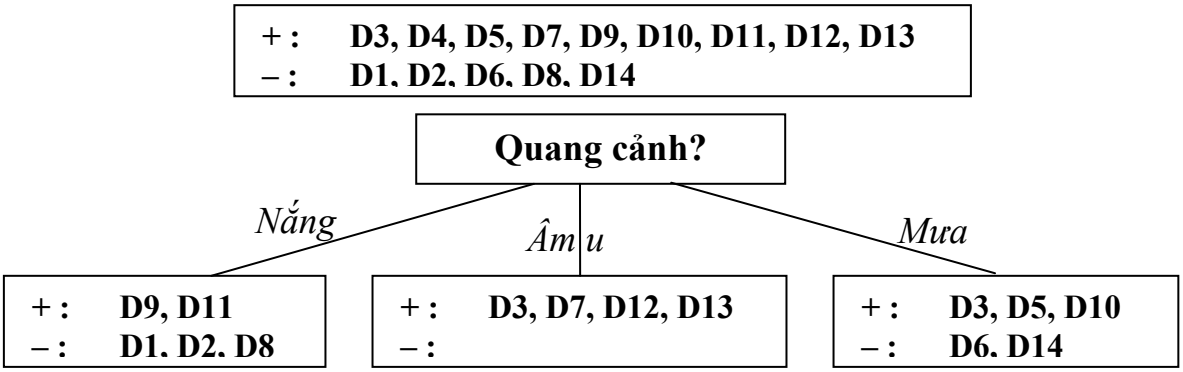
Occam's razor và một số lập luận khác đều cho rằng 'giả thuyết có khả năng nhất là giả thuyết đơn giản nhất thống nhất với tất cả các quan sát', ta nên luôn luôn chấp nhận những câu trả lời đơn giản nhất đáp ứng một cách đúng đắn dữ liệu của chúng ta. Trong trường hợp này là các giải thuật học cố gắng tạo ra cây quyết định nhỏ nhất phân loại một cách đúng đắn tất cả các ví dụ đã cho. Trong phần kế tiếp, chúng ta sẽ đi vào giải thuật ID3, là một giải thuật quy nạp cây quyết định đơn giản thỏa mãn các vấn đề vừa nêu.

II.2 Giải thuật ID3 xây dựng cây quyết định từ trên–xuống

ID3 xây dựng cây quyết định (cây QĐ) theo cách từ trên xuống. Lưu ý rằng đối với bất kỳ thuộc tính nào, chúng ta cũng có thể phân vùng tập hợp các ví dụ rèn luyện thành những tập con tách rời, mà ở đó mọi ví dụ trong một phân vùng (partition) có một giá trị chung cho thuộc tính đó. ID3 chọn một thuộc tính để kiểm tra tại nút hiện tại của cây và dùng trắc nghiệm này để phân vùng tập hợp các ví dụ; thuật toán khi đó xây dựng theo cách đệ quy một cây con cho từng phân vùng. Việc này tiếp tục cho đến khi mọi thành viên của phân vùng đều nằm trong cùng một lớp; lớp đó trở thành nút lá của cây.

Vì thứ tự của các trắc nghiệm là rất quan trọng đối với việc xây dựng một cây QĐ đơn giản, ID3 phụ thuộc rất nhiều vào tiêu chuẩn chọn lựa trắc nghiệm để làm gốc của cây. Để đơn giản, phần này chỉ mô tả giải thuật dùng để xây dựng cây QĐ, với việc giả định một hàm chọn trắc nghiệm thích hợp. Phần kế tiếp sẽ trình bày heuristic chọn lựa của ID3.

Ví dụ, hãy xem xét cách xây dựng cây QĐ của ID3 trong hình sau từ tập ví dụ rèn luyện trong bảng 9.1.



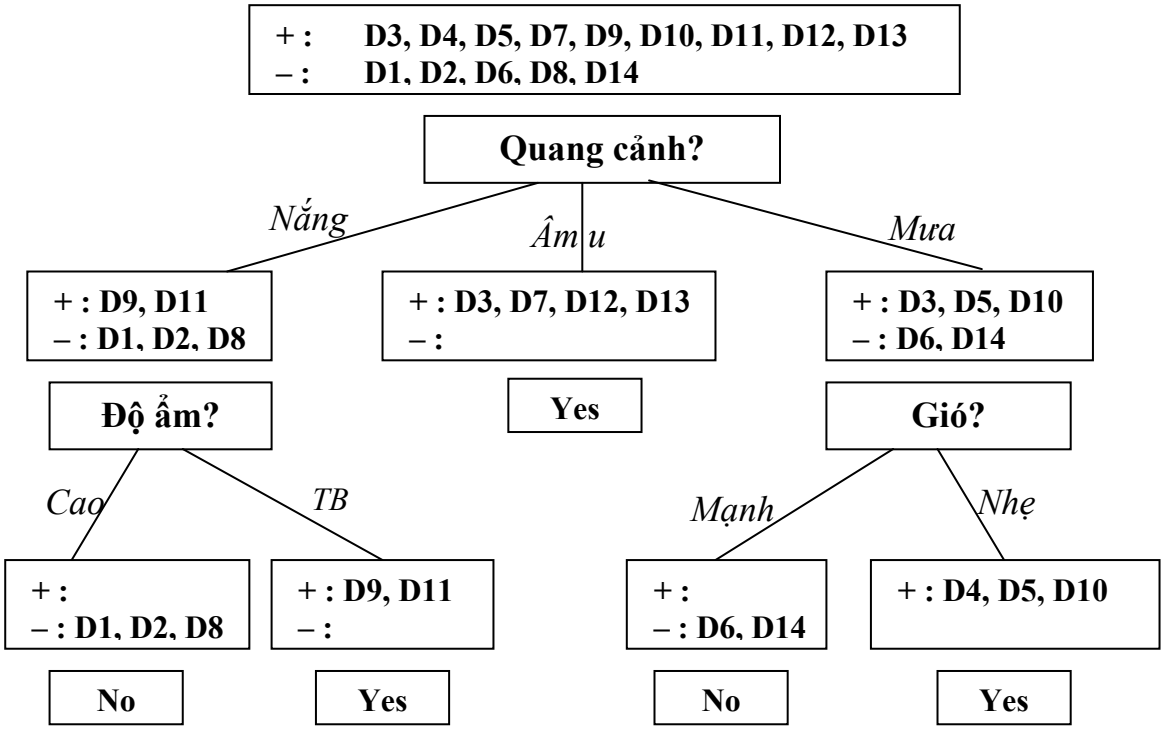
***Hình 9.2** - Một phần cây quyết định xây dựng được*

Bắt đầu với bảng đầy đủ gồm 14 ví dụ rèn luyện, ID3 chọn thuộc tính quang cảnh để làm thuộc tính gốc sử dụng hàm chọn lựa thuộc tính mô tả trong phần kế tiếp. Trắc nghiệm này phân chia tập ví dụ như cho thấy trong hình 9.2 với phần tử của mỗi phân vùng được liệt kê bởi số thứ tự của chúng trong bảng.

* ID3 xây dựng cây quyết định theo giải thuật sau:

Function induce_tree(tập_ví_dụ, tập_thuộc_tính)
begin
    if mọi ví dụ trong tập_ví_dụ đều nằm trong cùng một lớp ***then***
        return một nút lá được gán nhãn bởi lớp đó
    else if tập_thuộc_tính là rỗng ***then***
        return nút lá được gán nhãn bởi tuyển của tất cả các lớp trong tập_ví_dụ
    else begin
        chọn một thuộc tính P, lấy nó làm gốc cho cây hiện tại;
        xóa P ra khỏi tập_thuộc_tính;
        với mỗi giá trị V của P
        begin
            tạo một nhánh của cây gán nhãn V;
            Đặt vào phân_vùng$_V$ các ví dụ trong tập_ví_dụ có giá trị V tại thuộc tính P;
            Gọi induce_tree(phân_vùng$_V$, tập_thuộc_tính), gắn kết quả vào nhánh V
        end
    end
end

ID3 áp dụng hàm *induce_tree* một cách đệ quy cho từng phân vùng. Ví dụ, phân vùng của nhánh "Âm u" có các ví dụ toàn dương, hay thuộc lớp 'Có', nên ID3 tạo một nút lá với nhãn là lớp 'Có'. Còn phân vùng của hai nhánh còn lại vừa có ví dụ âm, vừa có ví dụ dương. Nên tiếp tục chọn thuộc tính "Độ ẩm" để làm trắc nghiệm cho nhánh Nắng, và thuộc tính Gió cho nhánh Mưa, vì các ví dụ trong các phân vùng con của các nhánh cây này đều thuộc cùng một lớp, nên giải thuật ID3 kết thúc và ta có được cây QĐ như hình 9.3.



***Hình 9.3** - Cây quyết định đã xây dựng xong.*

Lưu ý, để phân loại một ví dụ, có khi cây QĐ không cần sử dụng tất cả các thuộc tính đã cho, mặc dù nó vẫn phân loại đúng tất cả các ví dụ.

* Các khả năng có thể có của các phân vùng (partition):

Trong quá trình xây dựng cây QĐ, phân vùng của một nhánh mới có thể có các dạng sau:

1. Có các ví dụ thuộc các lớp khác nhau, chẳng hạn như có cả ví dụ âm và dương như phân vùng "Quang cảnh = Nắng" của ví dụ trên => giải thuật phải tiếp tục tách một lần nữa.

2. Tất cả các ví dụ đều thuộc cùng một lớp, chẳng hạn như toàn âm hoặc toàn dương như phân vùng "Quang cảnh = Âm u" của ví dụ trên => giải thuật trả về nút lá với nhãn là lớp đó.

3. Không còn ví dụ nào => giải thuật trả về mặc nhiên

4. Không còn thuộc tính nào => nghĩa là dữ liệu bị nhiễu, khi đó giải thuật phải sử dụng một luật nào đó để xử lý, chẳng hạn như luật đa số (lớp nào có nhiều ví dụ hơn sẽ được dùng để gán nhãn cho nút lá trả về).

Từ các nhận xét này, ta thấy rằng để có một cây QĐ đơn giản, hay một cây có chiều cao là thấp, ta nên chọn một thuộc tính sao cho tạo ra càng nhiều các phân vùng chỉ chứa các ví dụ thuộc cùng một lớp càng tốt. Một phân vùng chỉ có ví dụ thuộc cùng một lớp, ta nói phân vùng đó có tính thuần nhất. Vậy, để chọn thuộc tính kiểm tra có thể giảm thiểu chiều sâu của cây QĐ, ta cần một phép đo để đo tính thuần nhất của các phân vùng, và chọn thuộc tính kiểm tra tạo ra càng nhiều phân vùng thuần nhất càng tốt. ID3 sử dụng lý thuyết thông tin để thực hiện điều này.

II.3 Thuộc tính nào là thuộc tính dùng để phân loại tốt nhất?

Quinlan (1983) là người đầu tiên đề xuất việc sử dụng lý thuyết thông tin để tạo ra các cây quyết định và công trình của ông là cơ sở cho phần trình bày ở đây. Lý thuyết thông tin của Shannon (1948) cung cấp khái niệm entropy để đo tính thuần nhất (hay ngược lại là độ pha trộn) của một tập hợp. Một tập hợp là thuần nhất nếu như tất cả các phần tử của tập hợp đều thuộc cùng một loại, và khi đó ta nói tập hợp này có độ pha trộn là thấp nhất. Trong trường hợp của tập ví dụ, thì tập ví dụ là thuần nhất nếu như tất cả các ví dụ đều có cùng giá trị phân loại.

Khi tập ví dụ là thuần nhất thì có thể nói: ta biết chắc chắn về giá trị phân loại của một ví dụ thuộc tập này, hay ta có lượng thông tin về tập đó là cao nhất. Khi tập ví dụ có độ pha trộn cao nhất, nghĩa là số lượng các ví dụ có cùng giá trị phân loại cho mỗi loại là tương đương nhau, thì khi đó ta không thể đoán chính xác được một ví dụ có thể có giá trị phân loại gì, hay nói khác hơn, lượng thông tin ta có được về tập này là ít nhất. Vậy, điều ta mong muốn ở đây là làm sao chọn thuộc tính để hỏi sao cho có thể chia tập ví dụ ban đầu thành các tập ví dụ thuần nhất càng nhanh càng tốt. Vậy trước hết, ta cần có một phép đo để đo độ thuần nhất của một tập hợp, từ đó mới có thể so sánh tập ví dụ nào thì tốt hơn. Phần kế tiếp sẽ trình bày công thức tính entropy của một tập hợp.

II.3.1 Entropy đo tính thuần nhất của tập ví dụ

Khái niệm entropy của một tập S được định nghĩa trong Lý thuyết thông tin là số lượng mong đợi các bít cần thiết để mã hóa thông tin về lớp của một thành viên rút ra một cách ngẫu nhiên từ tập S. Trong trường hợp tối ưu, mã có độ dài ngắn nhất. Theo lý thuyết thông tin, mã có độ dài tối ưu là mã gán $-log_2 p$ bits cho thông điệp có xác suất là p.

Trong trường hợp S là tập ví dụ, thì thành viên của S là một ví dụ, mỗi ví dụ thuộc một lớp hay có một giá trị phân loại.

Entropy có giá trị nằm trong khoảng [0..1],

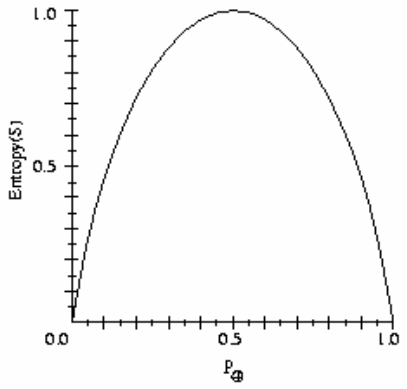
Entropy(S) = 0 ⇔ tập ví dụ S chỉ toàn ví dụ thuộc cùng một loại, hay S là thuần nhất.

Entropy(S) = 1 ⇔ tập ví dụ S có các ví dụ thuộc các loại khác nhau với độ pha trộn là cao nhất.

0 < Entropy(S) < 1 ⇔ tập ví dụ S có số lượng ví dụ thuộc các loại khác nhau là không bằng nhau.

Để đơn giản ta xét trường hợp các ví dụ của S chỉ thuộc loại âm (-) hoặc dương (+).

Hình 9.4 minh họa sự phụ thuộc của giá trị entropy vào xác suất xuất hiện của ví dụ dương.



***Hình 9.4** - Entropy(S)*

Cho trước:

- Tập S là tập dữ liệu rèn luyện, trong đó thuộc tính phân loại có hai giá trị, giả sử là âm (-) và dương (+)
- p_+ là phần các ví dụ dương trong tập S.
- p_ là phần các ví dụ âm trong tập S.

Khi đó, entropy đo độ pha trộn của tập S theo công thức sau:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Một cách tổng quát hơn, nếu các ví dụ của tập S thuộc nhiều hơn hai loại, giả sử là có c giá trị phân loại thì công thức entropy tổng quát là:

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

II.3.2 Lượng thông tin thu được đo mức độ giảm entropy mong đợi

Entropy là một số đo đo độ pha trộn của một tập ví dụ, bây giờ chúng ta sẽ định nghĩa một phép đo hiệu suất phân loại các ví dụ của một thuộc tính. Phép đo này gọi là lượng thông tin

thu được, nó đơn giản là lượng giảm entropy mong đợi gây ra bởi việc phân chia các ví dụ theo thuộc tính này.
Một cách chính xác hơn, Gain(S,A) của thuộc tính A, trên tập S, được định nghĩa như sau:

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

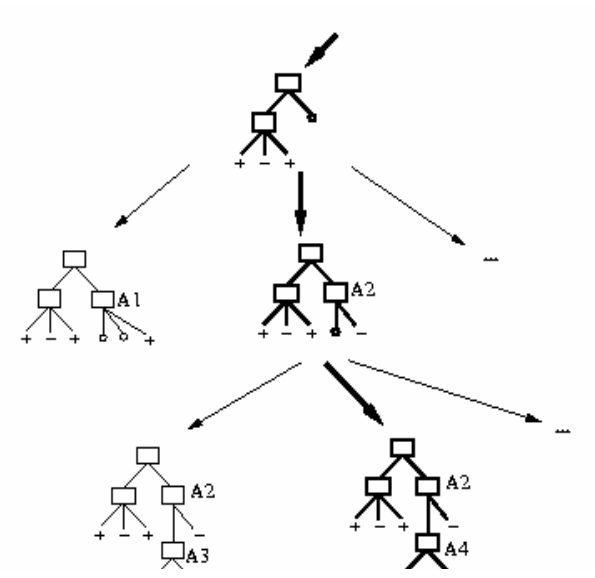
trong đó Values(A) là tập hợp có thể có các giá trị của thuộc tính A, và S_v là tập con của S chứa các ví dụ có thuộc tính A mang giá trị v.

Câu hỏi :

Áp dụng giải thuật ID3 kết hợp với công thức entropy và gain để tạo ra cây quyết định
đơn giản nhất cho bài toán phân loại xem ta 'có đi chơi tennis' từ tập dữ liệu rèn luyện
đã cho trong bảng 9.1.

II.4 Tìm kiếm không gian giả thuyết trong ID3

Cũng như các phương pháp học quy nạp khác, ID3 cũng tìm kiếm trong một không gian các giả thuyết một giả thuyết phù hợp với tập dữ liệu rèn luyện. Không gian giả thuyết mà ID3 tìm kiếm là một tập hợp các cây quyết định có thể có. ID3 thực hiện một phép tìm kiếm từ đơn giản đến phức tạp, theo giải thuật leo-núi (hill climbing), bắt đầu từ cây rỗng, sau đó dần dần xem xét các giả thuyết phức tạp hơn mà có thể phân loại đúng các ví dụ rèn luyện. Hàm đánh giá được dùng để hướng dẫn tìm kiếm leo núi ở đây là phép đo lượng thông tin thu được.



Hình 9.5 - Không gian tìm kiếm của ID3.

Từ cách nhìn ID3 như là một giải thuật tìm kiếm trong không gian các giả thuyết, ta có một số nhận xét như sau:

- Không gian giả thuyết các cây quyết định của ID3 là một không gian đầy đủ các cây quyết định trên các thuộc tính đã cho trong tập rèn luyện. Điều này có nghĩa là không gian mà ID3 tìm kiếm chắc chắn có chứa cây quyết định cần tìm.
- Trong khi tìm kiếm, ID3 chỉ duy trì một giả thuyết hiện tại. Vì vậy, giải thuật này không có khả năng biểu diễn được tất cả các cây quyết định khác nhau có khả năng phân loại đúng dữ liệu hiện có.
- Giải thuật thuần ID3 không có khả năng quay lui trong khi tìm kiếm. Vì vậy, nó có thể gặp phải những hạn chế giống như giải thuật leo núi, đó là hội tụ về cực tiểu địa phương.