HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

TỪ MINH PHƯƠNG

BÀI GIẢNG

Nhập môn trí tuệ nhân tạo

Hà nội 2010

LỜI NÓI ĐẦU

Trí tuệ nhân tạo là một nhánh của khoa học máy tính với mục tiêu nghiên cứu xây dựng và ứng dụng các hệ thống thông minh nhân tạo. Đây là một trong những lĩnh vực được quan tâm nghiên cứu nhiều nhất của khoa học máy tính hiện nay với nhiều kết quả được ứng dụng rộng rãi.

Môn học Nhập môn trí tuệ nhân tạo là môn học mang tính chuyên ngành trong chương trình đào tạo công nghệ thông tin hệ đại học. Mục tiêu của môn học nhằm giúp sinh viên làm quen với khái niệm trí tuệ nhân tạo thông qua việc giới thiệu một số kỹ thuật và ứng dụng cụ thể. Với việc học về trí tuệ nhân tạo, một mặt, sinh viên sẽ được làm quen với những phương pháp, cách giải quyết vấn đề không thuộc lĩnh vực toán rời rạc hoặc giải thuật truyền thống, chẳng hạn các phương pháp dựa trên heuristics, các phương pháp dựa trên tri thức, dữ liệu. Mặt khác, sinh viên sẽ được làm quen với khả năng ứng dụng tiềm tàng các kỹ thuật trí tuệ nhân tạo trong nhiều bài toán thực tế.

Do trí tuệ nhân tạo hiện đã phát triển thành một lĩnh vực rộng với khá nhiều lĩnh vực chuyên sâu, việc lựa chọn các nội dung để giới thiệu cho sinh viên là vấn đề không đơn giản. Trong tài liệu này, các nội dung được lựa chọn hoặc là những nội dung có tính tiêu biểu, kinh điển của trí tuệ nhân tạo như nội dung về biểu diễn tri thức bằng logic, các phương pháp tìm kiếm, hoặc là những nội dung có tính ứng dụng và đang có tính thời sự hiện nay, tiêu biểu là phương pháp suy diễn xác suất và các kỹ thuật học máy.

Do khuôn khổ có hạn của tài liệu với tính chất là bài giảng, phần giới thiệu về việc sử dụng kỹ thuật trí tuệ nhân tạo trong ứng dụng cụ thể không được trình bày nhiều. Chúng tôi dành phần lựa chọn ứng dụng cụ thể cho giảng viên trong quá trình lên lớp và hướng dẫn sinh viên. Tùy điều kiện, giảng viên có thể lựa chọn trong danh mục ứng dụng rất phong phú để giới thiệu và minh họa cho các nội dung của tài liệu.

Nội dung tài liệu được trình bày thành năm chương.

Chương 1 là phần giới thiệu tổng quan về trí tuệ nhân tạo bao gồm khái niệm, lịch sử hình thành, sơ lược về những kỹ thuật và ứng dụng tiêu biểu. Nội dung chương không đi quá sâu vào việc định nghĩa chính xác trí tuệ nhân tạo là gì, thay vào đó, sinh viên được giới thiệu về những lĩnh vực nghiên cứu chuyên sâu và lịch sử phát triển, trước khi làm quen với nội dung cụ thể trong các chương sau.

Chương 2 trình bày cách giải quyết vấn đề bằng phương pháp tìm kiếm. Các phương pháp tìm kiếm bao gồm: tìm kiếm mù, tìm kiếm có thông tin, và tìm kiếm cục bộ. Khác với một số tài liệu khác về trí tuệ nhân tạo, nội dung về tìm kiếm có đối thủ không được đề cập đến trong tài liệu này.

Chương 3 tóm tắt về vấn đề sử dụng, biểu diễn tri thức và suy diễn, trước khi đi sâu trình bày về biểu diễn tri thức và suy diễn với logic. Trong hai hệ thống logic được trình bày là logic mệnh đề và logic vị từ, nội dung chương được dành nhiều hơn cho logic vị từ. Do nội dung về lập trình logic hiện không còn ứng dụng nhiều, chúng tôi không giới thiệu về vấn đề lập trình và xây dựng ứng dụng cụ thể ở đây.

Chương 4 là mở rộng của biểu diễn tri thức và suy diễn với việc sử dụng nguyên tắc suy diễn xác suất và mạng Bayes. Sau khi trình bày về sự cần thiết của lập luận trong điều kiện

không rõ ràng cùng với nguyên tắc suy diễn xác suất, phần chính của chương tập trung vào khái niệm cùng với ứng dụng mạng Bayes trong biểu diễn tri thức và suy diễn.

Chương 5 là chương nhập môn về học máy. Trong chương này, sinh viên được làm quen với khái niệm, nguyên tắc và ứng dụng của học máy. Trong phạm vi chương cũng trình bày ba kỹ thuật học máy dùng cho phân loại là cây quyết định, phân loại Bayes và phân loại dựa trên ví dụ. Đây là những phương pháp đơn giản, dễ giới thiệu, đồng thời là những phương pháp tiêu biểu và có nguyên lý khác nhau, thuận tiện để trình bày với tính chất nhập môn.

Tài liệu được biên soạn từ kinh nghiệm giảng dạy học phần Nhập môn trí tuệ nhân tạo của tác giả tại Học viện Công nghệ bưu chính viễn thông, trên cơ sở tiếp thu phản hồi từ sinh viên và đồng nghiệp. Tài liệu có thể sử dụng làm tài liệu học tập cho sinh viên đại học ngành công nghệ thông tin và các ngành liên quan, ngoài ra có thể sử dụng với mục đích tham khảo nhanh cho những người quan tâm tới trí tuệ nhân tạo.

Trong quá trình biên soạn tài liệu, mặc dù tác giả đã có nhiều cố gắng song không thể tránh khỏi những thiếu sót. Ngoài ra, trí tuệ nhân tạo một lĩnh vực rộng, đang tiến bộ rất nhanh của khoa học máy tính đòi hỏi tài liệu phải được cập nhật thường xuyên. Tác giả rất mong muốn nhận được ý kiến phản hồi, góp ý cho các thiếu sót cũng như ý kiến về việc cập nhật, hoàn thiện nội dung của tài liệu.

Hà nội 2010

Tác giả

Mục lục

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1. KHÁI NIỆM TRÍ TUỆ NHÂN TẠO

Trí tuệ nhân tạo là một lĩnh vực nghiên cứu của khoa học máy tính và khoa học tính toán nói chung. Có nhiều quan điểm khác nhau về trí tuệ nhân tạo và do vậy có nhiều định nghĩa khác nhau về lĩnh vực khoa học này.

Mục đích của trí tuệ nhân tạo là xây dựng các *thực thể thông minh*. Tuy nhiên, do rất khó định nghĩa thế nào là thực thể thông minh nên cũng khó thống nhất định nghĩa trí tuệ nhân tạo. Theo một tài liệu được sử dụng rộng rãi trong giảng dạy trí tuệ nhân tạo hiện nay, các định nghĩa có thể nhóm thành bốn nhóm khác nhau, theo đó, trí tuệ nhân tạo là lĩnh vực nghiên cứu việc xây dựng các hệ thống có đặc điểm sau:

1) Hệ thống hành động như người.
2) Hệ thống có thể suy nghĩ như người
3) Hệ thống có thể suy nghĩ hợp lý
4) Hệ thống hành động hợp lý

Trong số các định nghĩa trên, nhóm thứ hai và ba quan tâm tới quá trình suy nghĩ và tư duy, trong khi nhóm thứ nhất và thứ tư quan tâm chủ yếu tới hành vi. Ngoài ra, hai nhóm định nghĩa đầu xác định mức độ thông minh hay mức độ trí tuệ bằng cách so sánh với khả năng suy nghĩ và hành động của con người, trong khi hai nhóm định nghĩa sau dựa trên khái niệm suy nghĩ hợp lý và hành động hợp lý. Trong phần phân tích bên dưới ta sẽ thấy suy nghĩ và hành động hợp lý khác với suy nghĩ và hành động như người thế nào.

Sau đây ta sẽ xem xét cụ thể các nhóm định nghĩa trên.

1) Hành động như người

Theo cách định nghĩa này, trí tuệ nhân tạo nhằm tạo ra các hệ thống có khả năng thực hiện những công việc đòi hỏi có trí tuệ của con người.

Phép thử Turing (Turing test): Vào năm 1950, Alan Turing – nhà toán học người Anh có nhiều đóng góp cho khoa học máy tính – đã xây dựng thủ tục cho phép định nghĩa trí tuệ. Thủ tục này sau đó được gọi là phép thử Turing (Turing test), và được thực hiện như sau. Hệ thống được gọi là thông minh, hay có trí tuệ nếu hệ thống có thể hành động tương tự con người trong các công việc đòi hỏi trí tuệ. Trong quá trình thử, một người kiểm tra sẽ đặt các câu hỏi (dưới dạng văn bản) và nhận câu trả lời cũng dưới dạng văn bản từ hệ thống. Nếu người kiểm tra không phân biệt được câu trả lời là do người thật trả lời hay do máy sinh ra thì hệ thống qua được phép thử và được gọi là có trí tuệ.

Cần lưu ý rằng, phép thử Turing nguyên bản không đòi hỏi có sự tiếp xúc vật lý trực tiếp giữa người kiểm tra và hệ thống bị kiểm tra, do việc tạo ra hệ thống người nhân tạo một cách vật lý được coi là không liên quan tới trí tuệ.

Để qua được phép thử Turing, hệ thống cần có những khả năng sau:

- *Xử lý ngôn ngữ tự nhiên*: để có thể phân tích, hiểu câu hỏi và tổng hợp câu trả lời trên một ngôn ngữ giao tiếp thông thường như tiếng Việt hay tiếng Anh.
- *Biểu diễn tri thức*: phục vụ việc lưu tri thức và thông tin trong hệ thống.
- *Suy diễn*: sử dụng tri thức để trả lời câu hỏi.
- *Học máy*: để có thể thích nghi với hoàn cảnh và học những mẫu trả lời.

Trong lịch sử trí tuệ nhân tạo đã có những hệ thống như ELIZA được xây dựng nhằm mục đích vượt qua phép thử Turing mà không cần đầy đủ tới cả bốn khả năng trên.

2) Suy nghĩ như người

Những nghiên cứu theo hướng này dựa trên việc nghiên cứu quá trình nhận thức và tư duy của con người, từ đây mô hình hóa và tạo ra những hệ thống có mô hình nhận thức, tư duy tương tự. Việc tìm hiểu quá trình nhận thức, tư duy của người có thể thực hiện bằng cách thực hiện thí nghiệm tâm lý hoặc theo dõi quá trình sinh ra ý nghĩ ở người.

Một hệ thống trí tuệ nhân tạo dạng này là hệ thống GPS, viết tắt của General Problem Solver do Newell và Simon trình diễn năm 1961. GPS là chương trình máy tính cho phép giải quyết các bài toán bằng cách mô phỏng chuỗi suy nghĩ của con người khi giải quyết những bài toán như vậy.

Hiện nay, hướng nghiên cứu này được thực hiện trong khuôn khổ khoa học nhận thức (cognitive science) và được quan tâm nhiều hơn trong khuôn khổ tâm lý học.

3) Suy nghĩ hợp lý

Một cách tiếp cận khác là xây dựng những hệ thống có khả năng suy diễn dựa trên việc sử dụng các hệ thống hình thức như lô gic. Tiền thân của cách tiếp cận này có gốc rễ từ triết học Hy lạp do Aristot khởi xướng. Cơ sở chủ yếu là sử dụng lô gic để biểu diễn bài toán và giải quyết bằng suy diễn lô gic.

Khó khăn chủ yếu của cách tiếp cận này là việc mô tả hay biểu diện bài toán dưới dạng các cấu trúc lô gic để có thể giải quyết được. Trên thực tế, tri thức và thông tin về bài toán thường có yếu tố không đầy đủ, không chính xác. Ngoài ra, việc suy diễn lô gic đòi hỏi khối lượng tính toán lớn khi sử dụng trong thực tế.

4) Hành động hợp lý

Cách tiếp cận này tập trung vào việc xây dựng các tác tử (agent) có khả năng hành động hợp lý, tức là hành động để đem lại kết quả tốt nhất hoặc kết quả kỳ vọng tốt nhất khi có yếu tố không chắc chắn. Cần lưu ý rằng, hành động hợp lý có thể khác với hành động giống con người: con người không phải lúc nào cũng hành động hợp lý do bị chi phối bởi các yếu tố chủ quan.

Một đặc điểm quan trọng của hành động hợp lý là hành động kiểu này có thể dựa trên việc suy nghĩ (suy diễn) hợp lý hoặc không. Trong nhiều tình huống, việc hành động theo phản xạ, chẳng hạn khi gặp nguy hiểm, không đòi hỏi suy diễn phức tạp, nhưng lại cho kết quả tốt hơn.

1.2. LỊCH SỬ HÌNH THÀNH VÀ PHÁT TRIỂN

Lịch sử hình thành và phát triển TTNT có thể chia thành một số giai đoạn sau (các giai đoạn được chia theo mức độ phát triển và có thể giao nhau về thời gian):

a. Giai đoạn tiền khởi đầu (1943-1955)

Mặc dù chưa có khái niệm về TTNT, giai đoạn này ghi nhận một số kết quả có liên quan trực tiếp tới nghiên cứu về TTNT sau này:

- Năm 1943, Warren McCulloch và Walter Pitts mô tả mô hình mạng nơ ron nhân tạo đầu tiên, và cho thấy mạng nơ ron nhân tạo có khả năng biểu diễn nhiều hàm số toán học.
- Năm 1950, Alan Turing công bố bài báo nhắc tới trí tuệ máy, trong đó lần đầu tiên mô tả khái niệm phép thử Turing, học máy, thuật toán di truyền, và học tăng cường.
- Năm 1956 được coi là năm chính thức ra đời của khái niệm trí tuệ nhân tạo. Mười nhà nghiên cứu trẻ đã tổ chức một cuộc hội thảo kéo dài hai tháng tại trường đạt học Dartmouth với mục đích đặt nền móng đầu tiên cùng với tên gọi chính thức của TTNT.

b. Giai đoạn khởi đầu (1952-1969)

Đây là giai đoạn với nhiều thành tích nhất định của các nghiên cứu TTNT, được thể hiện qua một số ví dụ sau:

- Các chương trình Logic Theorist và sau đó là General Problem Solver của Newell và Simon, có khả năng chứng minh định lý toán học theo cách tương tự tư duy của con người.
- Năm 1952, Arthur Samuel xây dựng một số chương trình chơi checkers. Chương trình có khả năng học và đánh thắng các đối thủ nghiệp dư.
- Năm 1958, John McCarthy đề xuất ngôn ngữ Lisp, sau này trở thành một trong hai ngôn ngữ thông dụng nhất của TTNT.
- Mạng nơ ron nhân tạo tiếp tục tiếp tục được phát triển với một số phát minh mới như mạng adalines (1962), Perceptron của Rosenblatt (1962), cho phép giải quyết nhiều bài toán học máy.

c. Hệ thống dựa trên trí thức (1969-1979)

Các chương trình trí tuệ nhân tạo xây dựng trong giai đoạn trước có một số nhất định do không có tri thức về lĩnh vực liên quan, và do vậy không thể giải quyết những bài toán khó, đòi hỏi khối lượng tính toán lớn hoặc nhiều tri thức chuyên sâu. Để khắc phục, giai đoạn này chú trọng tới việc sử dụng nhiều tri thức, thông tin đặc thù cho lĩnh vực hẹp của vấn đề cần giải quyết. Sau đây là một số hệ thống như vậy:

- DENDRAL là chương trình hệ chuyên gia xây dựng tại trường Stanford, cho phép dự đoán cấu trúc phân tử. Chương trình làm việc dựa trên các luật do chuyên gia hóa cung cấp. Một trong các tác giả của DENDRAL, sau đó đã cùng với cộng sự xây dựng MYCIN, hệ chuyên gia cho phép chẩn đoán bệnh nhiễm trùng máy. Với khoảng 450 quy

tắc do chuyên gia cung cấp, hệ thống có chất lượng chẩn đoán tương đương chuyên gia trong lĩnh vực này.

- Việc sử dụng tri thức cũng được sử dụng để giải quyết vấn đề hiểu ngôn ngữ tự nhiên, ví dụ trong hệ thống dịch tự động.

d. TTNT có sản phẩm thương mại (1980 đến nay)

Sau thành công của những hệ chuyên gia đầu tiên, việc xây dựng hệ chuyên gia được thương mại hóa từ năm 1980 và đặc biệt phát triển cho tới 1988. Sau giai đoạn này, do một số hạn chế của hệ chuyên gia, TTNT rơi vào một giai đoạn trì trệ, không có những bước tiến đáng kể.

Giai đoạn này cũng đánh dấu sự trở lại của mạng nơ ron nhân tạo sau một thời gian không có các phát minh và ứng dụng đáng kể. Cho đến hiện này, mạng nơ ron nhân tạo vẫn được sử dụng tương đối nhiều cho học máy và như các chương trình nhận dạng, phân loại tự động.

e. TTNT chính thức trở thành ngành khoa học (1987 đến nay)

Bắt đầu từ giai đoạn này, TTNT đã có phương pháp nghiên cứu riêng của mình, tuân theo các yêu cầu chung đối với phương pháp nghiên cứu khoa học. Chẳng hạn, kết quả cần chứng minh bằng thực nghiệm, và được phân tích kỹ lưỡng bằng khoa học thống kê.

Nhiều phát minh trước đây của TTNT như mạng nơ ron nhân tạo được phân tích và so sánh kỹ càng với những kỹ thuật khác của thống kê, nhận dạng, và học máy, do vậy các phương pháp không còn mang tính kinh nghiệm thuần túy mà đều dựa trên các cơ sở lý thuyết rõ ràng hơn.

1.3. CÁC LĨNH VỰC NGHIÊN CỨU VÀ ỨNG DỤNG CHÍNH

1.3.1. Các lĩnh vực nghiên cứu

Trí tuệ nhân tạo được chia thành một số lĩnh vực nghiên cứu nhỏ hơn nhằm giải quyết những vấn đề khác nhau khi xây dựng một hệ thống trí tuệ nhân tạo. Thông thường, một hệ thống trí tuệ nhân tạo hoàn chỉnh, làm việc trong việc một môi trường nào đó cần có khả năng: *cảm nhận* (perception), *suy diễn* (reasoning), và *hành động* (action). Lĩnh vực nghiên cứu của trí tuệ nhân tạo cũng được phân chia theo ba thành phần này.

a. Cảm nhận

Hệ thống cần có cơ chế thu nhận thông tin liên quan tới hoạt động từ môi trường bên ngoài. Đó có thể là camera, cảm ứng âm thanh, các cảm biến khác. Đó cũng có thể đơn giản hơn là thông tin do người dùng nhập vào chương trình nhập vào bằng tay. Để biến đổi thông tin nhận được về dạng có thể hiểu được, thông tin cần được xử lý nhờ những kỹ thuật thuộc các lĩnh vực sau:

- **Thị giác máy** (computer vision): nghiên cứu về việc thu nhận, xử lý, nhận dạng thông tin hình ảnh (chẳng hạn từ camera) thành biểu diễn mức cao hơn như các đối tượng xung quanh để máy tính sau đó có thể hiểu được.

- **Xử lý ngôn ngữ tự nhiên** (natural language processing): phân tích thông tin, dữ liệu nhận được dưới dạng âm thanh hoặc văn bản và được trình bày dưới dạng ngôn ngữ tự nhiên của con người.

b. Suy diễn

Là quá trình đưa ra kết luận về hành động trên cơ sở cảm nhận và tri thức bên trong. Thành phần này được xây dựng dựa trên kỹ thuật từ những lĩnh vực nghiên cứu sau:

- **Biểu diễn tri thức** (knowledge representation): sự kiện, thông tin, tri thức về thế giới xung quanh cần được biểu diễn dưới dạng máy tính có thể "hiểu" được, chẳng hạn dưới dạng lô gic hoặc ngôn ngữ TTNT nào đó.
- **Tìm kiếm** (search): nhiều bài toán hoặc vấn đề có thể phát biểu và giải quyết như bài toán tìm kiếm trong không gian trạng thái. TTNT nghiên cứu cách tìm kiếm khi số trạng thái trong không gian quá lớn.
- **Suy diễn** (inference hay reasoning): là quá trình sinh ra kết luận hoặc sự kiện mới từ những sự kiện và thông tin đã có.
- **Học máy** (machine learning): làm tăng hiệu quả giải quyết vấn đề nhờ trên dữ liệu và kinh nghiệm đã có.
- **Lập kế hoạch** (planning): là quá trình sinh ra các bước hành động cần thực hiện để thực hiện một mục tiêu nào đó dựa trên thông tin về môi trường, về hiệu quả từng hành động, về tình huống hiện thời và mục tiêu cần đạt.

c. Hành động

Cho phép hệ thống tác động vào môi trường xung quanh hoặc đơn giản là đưa ra thông tin về kết luận của mình. Thành phần này được xây dựng dựa trên những kỹ thuật sau:

- **Kỹ thuật rôbốt** (robotics): là kỹ thuật xây dựng các cơ quan chấp hành như cánh tay người máy, tổng hợp tiếng nói, tổng hợp ngôn ngữ tự nhiên.

1.3.2. Một số ứng dụng

a. Các chương trình trò chơi

Việc xây dựng chương trình có khả năng chơi những trò chơi trí tuệ là một lĩnh vực có nhiều thành tựu cúa TTNT. Tiêu biểu là chương trình cờ vua Deep Blue đã từng thắng vô địch cờ vua thế giới Gary Kasparov.

b. Nhận dạng tiếng nói

Cho phép biến đổi từ âm thanh thu được thành các từ. Một trong những hệ thống nhận dạng tiếng nói thương mại đầu tiên là PEGASUS được dùng tại American Airlines cho phép nhận thông tin đặt chỗ tự động qua điện thoại. Phổ biến hơn là những chương trình nhận dạng cho phép quay số di động bằng giọng nói. Nhìn chung, các hệ thống nhận dạng tiếng nói hiện nay có độ chính xác tương đối hạn chế.

c. Thị giác máy tính

Tiêu biểu là các hệ thống nhận dạng chữ in với độ chính xác gần như tuyệt đối, hệ thống nhận dạng tròng mắt, vân tay, mặt người. Những hệ thống dạng này được sử dụng rộng rãi trong sản xuất để kiểm tra sản phẩm, trong hệ thống camera an ninh. Nhiều nghiên cứu thuộc vùng ứng dụng này đang được thực hiện như nghiên cứu nhận dạng chữ viết tay.

d. Hệ chuyên gia

Là các hệ thống làm việc dựa trên kinh nghiệm và tri thức của chuyên gia trong một lĩnh vực tương đối hẹp nào đó để đưa ra khuyến cáo, kết luận, chuẩn đoán một cách tự động. Các ví dụ gồm:

- MYCIN: hệ chuyên gian đầu tiên chẩn đoán bệnh về nhiễm trùng máu và cách điều trị.
- XCON của DEC: hỗ trợ chọn cấu hình máy tính tự động.

e. Xử lý, hiểu ngôn ngữ tự nhiên

Tiêu biểu là các hệ thống dịch tự động như hệ thống dịch của Google, các hệ thống tóm tắt nội dung văn bản tự động. Những hệ thống này sử dụng những thành phần đơn giản hơn như các phân hệ phân tích hình thái, cú pháp, ngữ nghĩa.

f. Lập kế hoạch, lập thời khóa biểu

Kỹ thuật TTNT được sử dụng nhiều trong bài toán lập thời khóa biểu cho trường học, xí nghiệp, các bài toán lập kế hoạch khác. Một ví dụ lập kế hoạch thành công với quy mô lớn là kế hoạch đảm bảo hậu cần cho quân đội Mỹ trong chiến dịch Cơn bão sa mạc tại Iraq đã được thực hiện gần như hoàn toàn dựa trên kỹ thuật TTNT.

CHƯƠNG 2: GIẢI QUYẾT VẤN ĐỀ BẰNG TÌM KIẾM

2.1. GIẢI QUYẾT VẤN ĐỀ VÀ KHOA HỌC TTNT

Tại sao phải tìm kiếm

Tìm kiếm là một trong những hướng nghiên cứu quan trọng của TTNT. Mục đích của TTNT là tìm cách giải quyết vấn đề hay bài toán một cách hợp lý, trong khi đó, một lớp lớn các bài toán có thể phát biểu và giải quyết dưới dạng tìm kiếm. Sau đây là một số ví dụ các bài toán như vậy.

- Trò chơi: nhiều trò chơi, ví dụ cờ vua, thực chất là quá trình tìm kiếm nước đi của các bên trong số những nước mà luật chơi cho phép, để giành lấy ưu thế cho bên mình.
- Lập thời khóa biểu: lập thời khóa biểu là lựa chọn thứ tự, thời gian, tài nguyên (máy móc, địa điểm, con người) thỏa mãn một số tiêu chí nào đó. Như vậy, lập thời khóa biểu có thể coi như quá trình tìm kiếm trong số tổ hợp phương án sắp xếp phương án đáp ứng yêu cầu đề ra.
- Tìm đường đi: trong số những đường đi, lựa chọn đường đi tới đích, có thể thỏa mãn một số tiêu chí nào đó như tiêu chí tối ưu về độ dài, thời gian, giá thành.v.v.
- Lập kế hoạch: là lựa chọn chuỗi hành động cơ sở cho phép đạt mục tiêu đề ra đồng thời thỏa mãn các yêu cầu phụ.

Sự phổ biến của các vấn đề có tính chất tìm kiếm dẫn tới yêu cầu phát biểu bài toán tìm kiếm một cách tổng quát, đồng thời xây dựng phương pháp giải bài toán tìm kiếm sao cho hiệu quả, thuận lợi. Do vậy, tìm kiếm đã được nghiên cứu trong khuôn khổ toán rời rạc, lý thuyết thuật toán. Trong khuôn khổ TTNT, tìm kiếm được đặc biệt quan tâm từ khía cạnh xây dựng phương pháp cho phép tìm ra kết quả trong trường hợp không gian tìm kiếm có kích thước lớn khiến cho những phương pháp truyền thống gặp khó khăn.

Ngoài việc đứng độc lập như chủ đề nghiên cứu riêng, tìm kiếm còn là cơ sở cho rất nhiều nhánh nghiên cứu khác của TTNT như lập kế hoạch, học máy, xử lý ngôn ngữ tự nhiên, suy diễn.

2.2. BÀI TOÁN TÌM KIẾM TRONG KHÔNG GIAN TRẠNG THÁI

2.2.1. Phát biểu bài toán tìm kiếm

Một cách tổng quát, một vấn đề có thể giải quyết thông qua tím kiếm bằng cách xác định tập hợp tất cả các phương án, đối tượng, hay trạng thái liên quan, gọi chung là *không gian trạng thái*. Thủ tục tìm kiếm sau đó sẽ khảo sát không gian trạng thái theo một cách nào đó để tìm ra lời giải cho vấn đề. Tùy vào cách thức khảo sát không gian trạng thái cụ thể, ta sẽ có những phương pháp tìm kiếm khác nhau.

Để có thể khảo sát không gian trạng thái, thuật toán tìm kiếm bắt đầu từ một trạng thái xuất phát nào đó, sau đó sử dụng những phép biến đổi trạng thái để nhận biết và chuyển sang trạng thái khác. Quá trình tìm kiếm kết thúc khi tìm ra lời giải, tức là khi đạt tới *trạng thái đích*.

Bài toán tìm kiếm cơ bản có thể phát biểu thông qua bốn thành phần chính sau:

- Tập các trạng thái Q. Đây chính là không gian trạng thái của bài toán.
- Tập (không rỗng) các trạng thái xuất phát S ($S \subseteq Q$). Thuật toán tìm kiếm sẽ xuất phát từ một trong những trạng thái này để khảo sát không gian tìm kiếm.
- Tập (không rỗng) các trạng thái đích G ($G \subseteq Q$). Trạng thái đích có thể được cho một cách tường minh, tức là chỉ ra cụ thể đó là trạng thái nào, hoặc không tường minh. Trong trường hợp sau, thay vì trạng thái cụ thể, bài toán sẽ quy định một số điều kiện mà trạng thái đích cần thỏa mãn. Ví dụ, khi chơi cờ vua, thay vì chỉ ra vị trí cụ thể quân cờ, ta chỉ có quy tắc cho biết trạng thái chiếu hết.
- Các *toán tử*, còn gọi là *hành động* hay *chuyển động*. Thực chất đây là một ánh xạ P: $Q \rightarrow Q$, cho phép chuyển từ trạng thái hiện thời sang các trạng thái khác. Với mỗi trạng thái n, $P(n)$ là tập các trạng thái được sinh ra khi áp dụng toán tử hay chuyển động P.
- Giá thành c: $Q \times Q \rightarrow R$. Trong một số trường hợp, quá trình tìm kiếm cần quan tâm tới giá thành đường đi. Giá thành để di chuyển từ nút x tới nút hàng xóm y được cho dưới dạng số dương $c(x, y)$.

2.2.2. Một số ví dụ

Các thành phần của bài toán tìm kiếm được minh họa trên một số ví dụ sau. Đây là những ví dụ không có ứng dụng thực tế nhưng đơn giản, dễ sử dụng cho mục đích minh họa.

Bài toán đố tám ô

Cho hình chữ nhật được chia thành chín ô như trên hình dưới, trong đó tám ô được đánh số từ 1 đến 8 và một ô trống. Có thể thay đổi vị trí các số bằng cách di chuyển ô trống. Mỗi lần di chuyển, ô trống có thể đổi chỗ với một ô số ở ngay phía trên, phía dưới, bên phải hoặc bên trái.

3	1	6
5		8
2	7	4

Trạng thái xuất phát

	1	2
3	4	5
6	7	8

Trạng thái đích

Hình 2.1. Trò đố 8 ô

Yêu cầu của bài toán là thực hiện các di chuyển để chuyển từ một sắp xếp các ô (ví dụ trên hình bên trái) sang một cách sắp xếp khác (hình bên phải). Ngoài ra có thể có yêu cầu phụ, ví dụ cần di chuyển sao cho số lần đổi chỗ ô trống là tối thiểu.

Trò đố 8 ô có thể phát biểu như bài toán tìm kiếm với các thành phần sau.

- Trạng thái (state): mỗi trạng thái là một sắp xếp cụ thể vị trí các ô
- Hành động (action): mỗi hành động tương ứng với một di chuyển ô trống trái, phải, lên, xuống
- Mục tiêu và kiểm tra (goal & test): so sánh với trạng thái đích
- Giá thành (cost): bằng tổng số lần dịch chuyển ô trống.

Bài toán n con hậu

Cho một bàn cờ vua kích thước n x n. Cần xếp n con hậu lên bàn cờ sao cho không có đôi hậu nào đe dọa nhau.

Đây cũng là bài toán tìm kiếm với các thành phần cụ thể như sau:

- Trạng thái: sắp xếp của cả n con hậu trên bàn cờ, hoặc sắp xếp của 0 đến n con hậu trên bàn cờ.
- Trạng thái xuất phát:
- Trạng thái đích: sắp xếp n con hậu trên bàn cờ sao cho không có đôi hậu nào đe dọa nhau.
- Chuyển động: Có nhiều cách khác nhau: đổi chỗ 2 con hậu, di chuyển một con hậu sang ô khác (cùng cột, khác cột).

2.2.3. Các tiêu chuẩn đánh giá thuật toán tìm kiếm

Với bài toán tìm kiếm được phát biểu như trên, nhiều thuật toán tìm kiếm có thể sử dụng để khảo sát không gian và tìm ra lời giải. Thuật toán tìm kiếm được đánh giá theo bốn tiêu chuẩn sau:

- *Độ phức tạp tính toán*: được xác định bằng khối lượng tính toán cần thực hiện để tìm ra lời giải. Thông thường, khối lượng tính toán được xác định bằng số lượng trạng thái cần xem xét trước khi tìm ra lời giải.
- *Yêu cầu bộ nhớ*: được xác định bằng số lượng trạng thái cần lưu trữ đồng thời trong bộ nhớ khi thực hiện thuật toán.
- *Tính đầy đủ*: nếu bài toán có lời giải thì thuật toán có khả năng tìm ra lời giải đó không? Nếu có, ta nói rằng thuật toán có tính đầy đủ, trong trường hợp ngược lại ta nói thuật toán không có tính đầy đủ.

- *Tính tối ưu*: nếu bài toán có nhiều lời giải thì thuật toán có cho phép tìm ra lời giải tốt nhất không?

2.2.4. Thuật toán tìm kiếm tổng quát và cây tìm kiếm

Thuật toán tổng quát dựa trên nguyên lý chung như sau:

Ý tưởng chung: xem xét trạng thái, sử dụng các hàm biến đổi trạng thái để di chuyển trong không gian trạng thái cho tới khi đạt đến trạng thái mong muốn.

Ý tưởng này được thể hiện qua thuật toán tìm kiếm tổng quát trên hình 2.2.

Search(Q, S, G, P)

(Q: không gian trạng thái, S: trạng thái bắt đầu, G: đích, P: hành động)

Đầu vào: bài toán tìm kiếm với 4 thành phần như trên

Đầu ra: trạng thái đích

Khởi tạo: $O \leftarrow S$ (*O: danh sách các nút mở, bước này làm nhiệm vụ gán S cho O*)

While($O \neq \emptyset$) do

1. chọn nút $n \in O$ và xóa n khỏi O
2. If $n \in G$, return (đường đi tới n)
3. Thêm $P(n)$ vào O

Return: Không có lời giải

Hình 2.2. Thuật toán tìm kiếm tổng quát

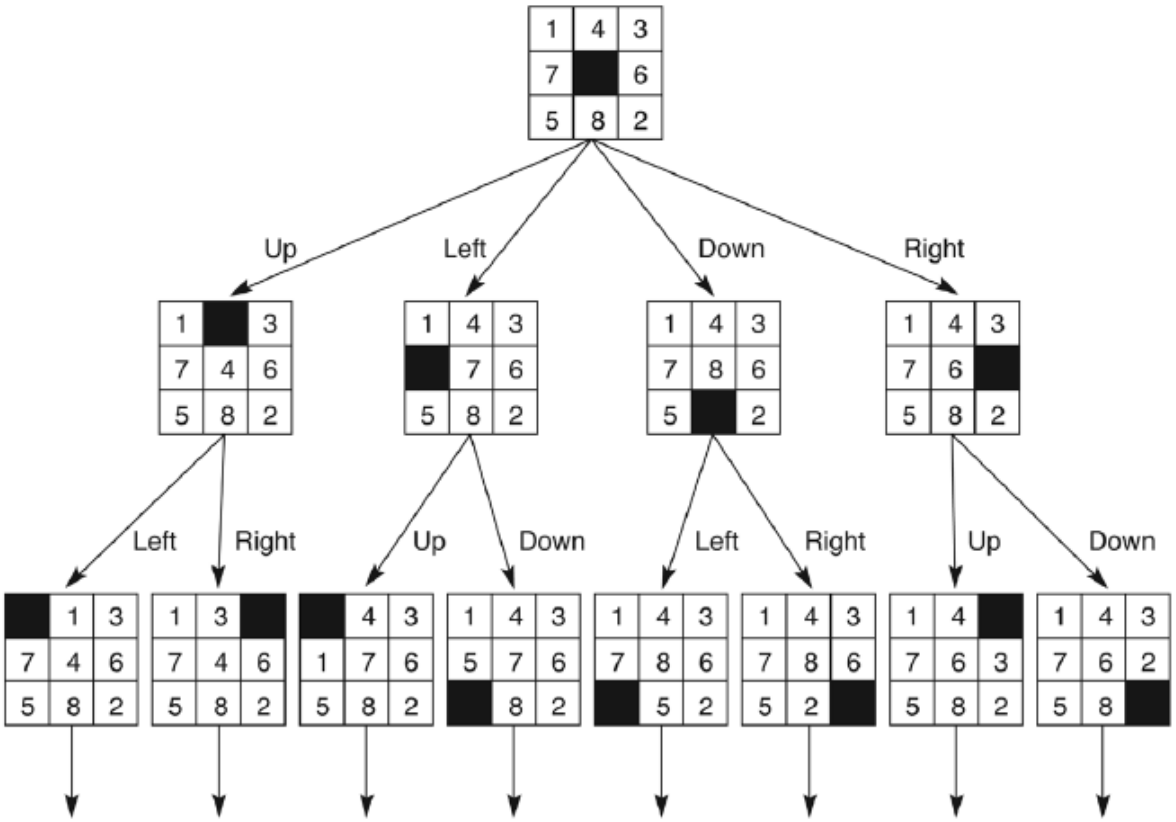
Thuật toán tìm kiếm tổng quát sinh ra một *cây tìm kiếm*, trong đó mỗi trạng thái tương ứng với một nút trên cây. Trạng thái xuất phát tương ứng với gốc cây, những trạng thái được mở rộng tạo thành các nút thế hệ tiếp theo. Trên hình 2.3 là ví dụ một cây tìm kiếm sinh ra cho bài toán đố 8 ô.

Thuật toán trên cũng giả sử rằng mỗi nút đã được duyệt sẽ không được duyệt lại lần nữa, do vậy không cần lưu trữ danh sách những nút đã duyệt. Trên thực tế, trong nhiều trường hợp, việc di chuyển trong không gian trạng thái sẽ dẫn tới những nút đã duyệt qua và tạo thành vòng lặp. Trong trường hợp như vậy cây tìm kiếm có thể là vô tận và cần có cách kiểm tra để không xem xét lại nút đã duyệt.

Sau đây là một số thuật ngữ sử dụng khi trình bày về thuật toán tìm kiếm:

- Các nút mở (nút biên): là các nút đang chờ để được mở rộng tiếp
- Sau khi nút được mở rộng, ta xóa nó khỏi danh sách các nút mở và nút khi đó gọi là nút đóng.

Cần lưu ý là trong thuật toán tìm kiếm tổng quát ở trên không quy định rõ nút nào trong số các nút biên (nằm trong tập O) được chọn để mở rộng. Việc lựa chọn nút cụ thể phụ thuộc vào từng phương pháp tìm kiếm và được trình bày trong những phần tiếp theo.



Hình 2.3. Cây tìm kiếm cho bài toán 8 ô

2.3. TÌM KIẾM KHÔNG CÓ THÔNG TIN (TÌM KIẾM MÙ)

Định nghĩa: Tìm kiếm không có thông tin, còn gọi là tìm kiếm mù (blind, uninformed search) là phương pháp duyệt không gian trạng thái chỉ sử dụng các thông tin theo phát biểu của bài toán tìm kiếm tổng quát trong quá trình tìm kiếm.

Tìm kiếm không có thông tin bao gồm một số thuật toán khác nhau. Điểm khác nhau căn bản của các thuật toán là ở thứ tự mở rộng các nút biên. Sau đây ta sẽ xem xét các thuật toán tìm theo chiều rộng, tìm theo chiều sâu, tìm kiếm sâu dần và một số biến thể của những thuật toán này.

2.3.1. Tìm kiếm theo chiều rộng (Breadth-first search – BFS)

Nguyên tắc của tìm kiếm theo chiều rộng là trong số những nút biên lựa chọn nút nông nhất (gần nút gốc nhất) để mở rộng.

Khi mở rộng một nút ta cần sử dụng con trỏ ngược để ghi lại nút cha của nút vừa được mở ra. Con trỏ này được sử dụng để tìm ngược lại đường đi về trạng thái xuất phát khi tìm được trạng thái đích.

Có thể nhận thấy, để thực hiện nguyên tắc tìm kiếm theo chiều rộng, ta cần lựa chọn nút được thêm vào sớm hơn trong danh sách nút biên O để mở rộng. Điều này có thể thực hiện dễ dàng bằng cách dùng một hàng đợi để lưu các nút biên.

Thuật toán tìm theo chiều rộng được thể hiện trên hình sau:

BFS (Q, S, G, P)

Đầu vào: bài toán tìm kiếm

Đầu ra: trạng thái đích

Khởi tạo: $O \leftarrow S$

While(O không rỗng) do

1. Chọn nút đầu tiên n từ O và xóa n khỏi O
2. If $n \in G$, return (đường đi tới n)
3. Thêm $P(n)$ vào cuối O

Return: Không có lời giải

Hình 2.4. Thuật toán tìm kiếm theo chiều rộng

Tính chất của tìm theo chiều rộng:

Đối chiếu với các tiêu chuẩn ở trên, tìm kiếm theo chiều rộng có những tính chất sau:

- Thuật toán có tính đầy đủ, tức là nếu bài toán có lời giải, tìm kiếm theo chiều rộng đảm bảo tìm ra lời giải.
- Có tính tối ưu. Đảm bảo tìm ra lời giải nằm gần nút gốc nhất. Tuy nhiên, trong trường hợp giá thành đường đi giữa các nút không bằng nhau thì điều này chưa đảm bảo tìm ra đường đi ngắn nhất.
- Độ phức tạp của thuật toán lớn (giả sử mỗi bước có b node được mở rộng trên b nhánh và có d mức, khi đó độ phức tạp của thuật toán là O(b^d)).

Giả sử rằng, mỗi trạng thái khi được phát triển sẽ sinh ra b trạng thái kề. Ta sẽ gọi b là ***nhân tố nhánh***. Giả sử rằng, nghiệm của bài toán là đường đi có độ dài d. Bởi nhiều nghiệm có thể được tìm ra tại một đỉnh bất kỳ ở mức d của cây tìm kiếm, do đó số đỉnh cần xem xét để tìm ra nghiệm là:

$$\mathbf{1 + b + b^2 + ... + b^{d-1} + k}$$

Trong đó k có thể là 1, 2, ..., b^d. Do đó số lớn nhất các đỉnh cần xem xét là:

$$\mathbf{1 + b + b^2 + ... + b^d}$$

Như vậy, độ phức tạp thời gian của thuật toán tìm kiếm theo bề rộng là $O(b^d)$. Độ phức tạp không gian cũng là $O(b^d)$, bởi vì ta cần lưu vào danh sách L tất cả các đỉnh của cây tìm kiếm ở mức d, số các đỉnh này là b^d.

2.3.2. Tìm kiếm theo giá thành thống nhất (Uniform-Cost-Search)

Trong trường hợp giá thành di chuyển giữa hai nút là không bằng nhau giữa các cặp nút, tìm theo chiều rộng không cho tìm ra lời giải có giá thành nhỏ nhất và do vậy không tối ưu. Để tìm ra đường đi ngắn nhất trong trường hợp này cần sử dụng một biến thể của tìm theo chiều rộng có tên gọi là tìm kiếm theo giá thành duy nhất.

Phương pháp: chọn node mở rộng có giá thành nhỏ nhất để mở rộng trước thay vì chọn nút nông nhất như trong tìm theo chiều rộng.

Thuật toán: được biến đổi từ tìm kiếm theo chiều rộng bằng cách thay ba bước trong vòng lặp While như sau:

1. Chọn node n có giá thành nhỏ nhất thuộc O và xóa n khỏi O
2. If $n \in G$, return (đường đi tới n)
3. Thêm P(n) và giá thành đường đi tới n vào O

Thuật toán tìm kiếm theo giá thành thống nhất còn được gọi là thuật toán Dijkstra

2.3.3. Tìm kiếm theo chiều sâu (Depth-First-Search: DFS)

Nguyên tắc của tìm kiếm theo chiều sâu là trong số những nút biên lựa chọn nút sâu nhất (xa nút gốc nhất) để mở rộng.

Để thực hiện nguyên tắc trên, ta cần lựa chọn nút được thêm vào sau cùng trong tập nút biên O để mở rộng. Điều này có thể thực hiện dễ dàng bằng cách dùng một ngăn xếp để lưu các nút biên, các nút được thêm vào và lấy ra theo nguyên lý LIFO.

Thuật toán tìm kiếm theo chiều sâu được thể hiện trên hình 2.5.

Tính chất thuật toán tìm theo chiều sâu:

- Thuật toán không đầy đủ trong trường hợp số trạng thái là vô hạn (cứ đi theo nhánh không đúng mãi mà không chuyển sang nhánh khác được).
- Thuật toán không tối ưu: thuật toán có thể mở rộng những nhánh dẫn tới lời giải không tối ưu trước, đặc biệt trong trường hợp có nhiều lời giải.
- Độ phức tạp của thuật toán ở trường hợp xấu nhất là $O(b^m)$ với m là độ sâu tối đa. Trên thực tế DFS tìm ra lời giải nhanh hơn BFS, đặc biệt nếu tồn tại nhiều lời giải.

- Bộ nhớ cần nhớ tối đa b*m (mỗi mức chỉ nhớ b node, với m mức). Để đánh giá độ phức tạp không gian của tìm kiếm theo độ sâu ta có nhận xét rằng, khi ta phát triển một đỉnh u trên cây tìm kiếm theo độ sâu, ta chỉ cần lưu các đỉnh chưa được phát triển mà chúng là các đỉnh con của các đỉnh nằm trên đường đi từ gốc tới đỉnh u. Như vậy đối với cây tìm kiếm có nhân tố nhánh b và độ sâu lớn nhất là m, ta chỉ cần lưu ít hơn b*m đỉnh. Ưu cầu bộ nhớ so với tìm theo chiều rộng là ưu điểm nổi bật nhất của tìm theo chiều sâu.

DFS(Q, S, G, P)

Đầu vào: bài toán tìm kiếm

Đầu ra: (đường đi tới) trạng thái đích

Khởi tạo: O←S

While(O ≠ Ø) do

1. Chọn node đầu tiên n ∈ O và xóa n khỏi O
2. If n ∈ G, return (đường đi tới n)
3. Thêm P(n) vào đầu O

Return: Không có lời giải

Hình 2.5. Thuật toán tìm kiếm theo chiều sâu

Tránh các nút lặp:

Trong nhiều trường hợp có thể có nhiều đường đi cùng dẫn tới một nút. Khi đó cây tìm kiếm sẽ trở thành đồ thị tìm kiếm, tức là có nhiều hơn một đường đi giữa hai nút. Thuật toán tìm kiếm khi đó sẽ mở rộng cùng một nút nhiều lần làm tăng khối lượng tính toán không cần thiết. Thậm chí có thể dẫn tới vòng lặp vô hạn.

Để tránh mở rộng những nút đã mở rộng rồi cần ghi lại những nút đã được mở rộng. Đối với tìm kiếm theo chiều sâu có hai cách kiểm tra:

- Chỉ ghi nhớ các nút đã mở rộng của nhánh hiện thời, nếu nút chuẩn bị mở rộng đã có trong số này thì không cần mở rộng nữa. Phương pháp này yêu cầu nhớ ít nút, thời gian kiểm tra cũng nhanh, tuy nhiên chỉ cho phép tránh vòng lặp vô hạn và không cho phép giảm khối lượng tính toán trong trường hợp nhiều nhánh của đồ thị tìm kiếm có những phần trùng nhau.
- Ghi nhớ toàn bộ những nút đã được mở rộng. Những nút này được lưu trong một danh sách gọi là danh sách nút đóng. Khi chuẩn bị mở rộng một nút, thuật toán kiểm tra xem nút đó đã có trong danh sách nút đóng chưa, nếu chưa, thuật toán sẽ mở rộng sau

đó thêm nút vào danh sách nút đóng. Trong trường hợp nút đã có trong danh sách nút đóng rồi, nút sẽ không được mở rộng lại lần thứ hai.

Việc không mở rộng những nút đã có trong danh sách nút đóng sẽ ảnh hưởng tới tính tối ưu của lời giải do thuật toán chỉ xem xét một đường đi tới một nút, trong khi đó có thể là đường đi không tối ưu.

3.4. Tìm kiếm sâu dần (Iterative Depending Search - IDS)

Mặc dù có ưu điểm rất lớn là không yêu cầu nhiều bộ nhớ như tìm theo chiều rộng, tìm theo chiều sâu có thể rất chậm hoặc bế tắc nếu mở rộng những nhánh sâu (vô tận) không chứa lời giải. Để khắc phục, có thể sử dụng kỹ thuật *tìm kiếm với độ sâu hữu hạn*: tìm kiếm theo phương pháp sâu dần nhưng không tiếp tục phát triển một nhánh khi đã đạt tới một độ sâu nào đó, thay vào đó, thuật toán chuyển sang phát triển nhánh khác.

Kỹ thuật này có thể sử dụng trong trường hợp có thể dự đoán được độ sâu của lời giải bằng cách dựa trên đặc điểm bài toán cụ thể. Chẳng hạn, nếu ta biết rằng đi từ Hà nội vào Vinh không đi qua quá 4 thành phố khác thì có thể dùng 4 làm giới hạn chiều sâu khi tìm kiếm đường đi. Một số bài toán khác cũng có thể dự đoán trước giới hạn độ sâu như vậy.

Tuy nhiên, trong trường hợp chung, ta thường không có trước thông tin về độ sâu của lời giải. Trong trường hợp như vậy có thể sử dụng phương pháp tìm kiếm sâu dần. Thực chất tìm kiếm sâu dần là tìm kiếm với độ sâu hữu hạn, trong đó giới hạn độ sâu được khởi đầu bằng một giá trị nhỏ, sau đó tăng dần cho tới khi tìm được lời giải.

Phương pháp: Tìm theo DFS những không bao giờ mở rộng các node có độ sâu quá một giới hạn nào đó. Giới hạn độ sâu sẽ được tăng dần cho đến khi tìm được lời giải (VD: nếu giới hạn là 2 mà không tìm được thì sẽ tăng lên 3).

Thuật toán tìm kiếm sâu dần thể hiện trên hình 2.6, trong đó tìm kiếm sâu được lặp lại, tại mỗi bước lặp, độ sâu được giới hạn bởi biến C.

IDS(Q, S, G, P)

Đầu vào: thuật toán tìm kiếm

Đầu ra: trạng thái đích

Khởi tạo: O←S (*O: danh sách các node mở, bước này làm nhiệm vụ gán S cho O*)

C ← 0 (C là giới hạn độ sâu tìm kiếm)

While(O ≠ Ø) do

1. Thực hiện 3 bước
   - Lấy node n đầu tiên ra khỏi O
   - If n ∈ G, return (đường đi tới) n

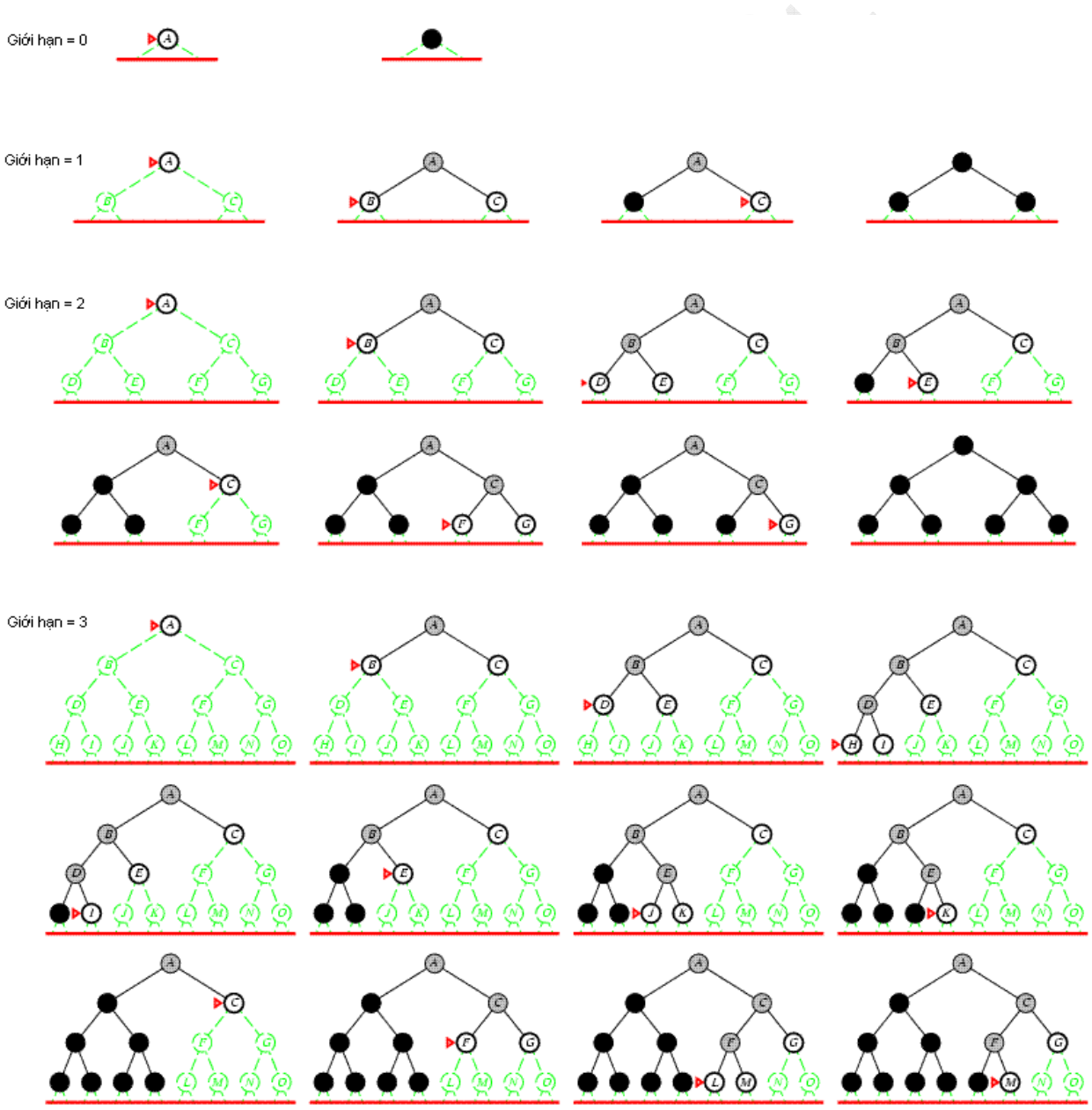
- If độ sâu (n) nhỏ hơn hoặc bằng C, then thêm P(n) vào đầu O

2. C++, O←S

Return: Không có lời giải

Hình 2.6. Thuật toán tìm kiếm sâu dần

Cây tìm kiếm trong trường hợp tìm sâu dẫn được minh họa trên hình 2.7.



Hình 2.7. Cây tìm kiếm theo thuật toán tìm kiếm sâu dần

Tính chất:

- Thuật toán đầy đủ
- Thuật toán tối ưu (nếu có nhiều lời giải, có thể tìm ra được lời giải gần gốc nhất)
- Yêu cầu bộ nhớ nhỏ (b*d) do tại mỗi bước lặp, thuật toán thực hiện tìm kiếm sâu dần
- Độ phức tạp tính toán O(b^d).Trong tìm kiếm sâu lặp, ta phải phát triển lặp lại nhiều lần cùng một trạng thái. Điều đó làm cho ta có cảm giác rằng, tìm kiếm sâu lặp lãng phí nhiều thời gian. Thực ra thời gian tiêu tốn cho phát triển lặp lại các trạng thái là không đáng kể so với thời gian tìm kiếm theo bề rộng. Thật vậy, mỗi lần gọi thủ tục tìm kiếm sâu hạn chế tới mức d, nếu cây tìm kiếm có nhân tố nhánh là b, thì số đỉnh cần phát triển là:

  $$\mathbf{1 + b + b^2 + ... + b^d}$$

  Nếu nghiệm ở độ sâu d, thì trong tìm kiếm sâu lặp, ta phải gọi thủ tục tìm kiếm sâu hạn chế với độ sâu lần lượt là 0, 1, 2, ..., d. Do đó các đỉnh ở mức 1 phải phát triển lặp d lần, các đỉnh ở mức 2 lặp d-1 lần, ..., các đỉnh ở mức d lặp 1 lần. Như vậy tổng số đỉnh cần phát triển trong tìm kiếm sâu lặp là:

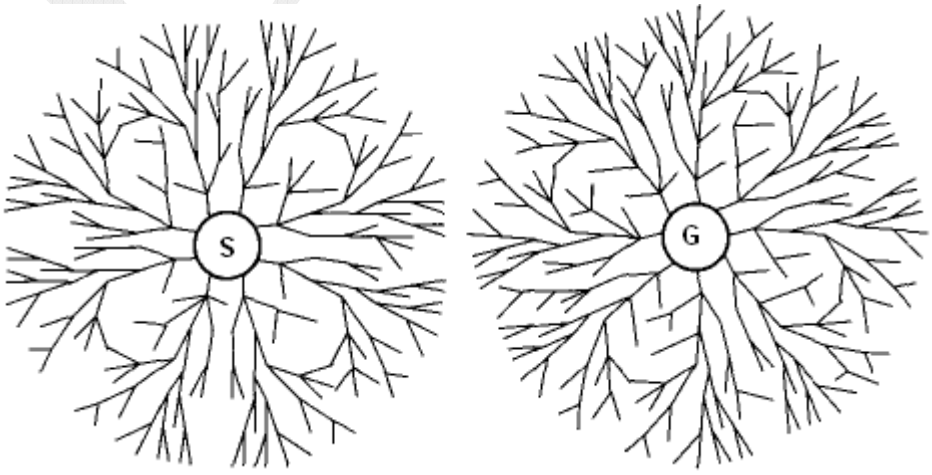
  $$\mathbf{(d+1)1 + db + (d-1)b^2 + ... + 2b^{d-1} + 1b^d}$$

Do đó thời gian tìm kiếm sâu dần là O(b^d).

2.3.4. Tìm theo hai hướng (Bidirectional Search)

Trong các phương pháp tìm kiếm ở trên, quá trình tìm kiếm bắt đầu từ nút xuất phát và kết thúc khi đạt tới nút đích. Do tính chất đối xứng của đường đi, quá trình tìm kiếm cũng có thể bắt đầu từ nút đích và tìm tới nút xuất phát. Ngoài ra, quá trình tìm kiếm có thể xuất phát đồng thời từ cả nút xuất phát và nút đích, xây dựng đồng thời hai cây tìm kiếm. Quá trình tìm kiếm kết thúc khi hai cây tìm kiếm có một nút chung (hình 2.8).

Phương pháp: tìm kiếm bắt nguồn từ nút xuất phát và nút đích. Vì vậy, sẽ tồn tại hai cây tìm kiếm, một cây có gốc là nút xuất phát và một cây có gốc là nút đích. Tìm kiếm kết thúc khi có lá của cây này trùng với lá của cây kia.



Hình 2.8: Cây tìm kiếm trong trường hợp tìm theo hai hướng

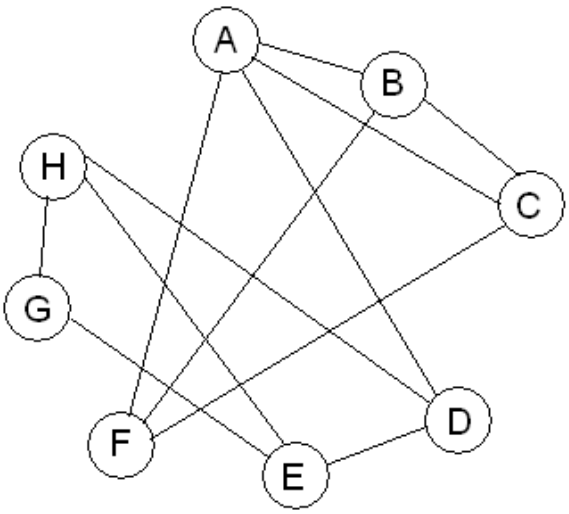
Chú ý:

- Khi tìm theo hai hướng cần sử dụng tìm theo chiều rộng. Việc tìm theo chiều sâu có thể không cho phép tìm ra lời giải nếu hai cây tìm kiếm phát triển theo hai nhánh không gặp nhau.
- Để tìm theo hai hướng cần viết thêm một hàm chuyển động ngược là P(x): tập các node con của x và D(x): tập các node cha của x. Node càng gần node xuất phát càng đươc coi là tổ tiên

Tính chất:

- Việc kiểm tra xem node lá này có trùng với node lá kia gây tốn thời gian
- Độ phức tạp tính toán: do gặp nhau ở giữa nên chiều sâu mỗi cây là d/2. Theo tính toán đối với tìm theo chiều rộng, độ phức tạp tính toán khi đó là O($b^{d/2}$). Như vậy mặc dù việc kiểm tra các nút trùng nhau gây tốn thời gian nhưng số lượng nút cần mở rộng của cả hai cây giảm đáng kể so với tìm theo một chiều.

6. Bài tập áp dụng



Tìm chu trình Hamilton xuất phát từ A

2.4. TÌM KIẾM CÓ THÔNG TIN (INFORMED SEARCH)

Đối với tìm kiếm mù, việc mở rộng các nút thuân theo một quy luật và không dựa vào thông tin hỗ trợ của bài toán. Kết quả của việc tìm kiếm như vậy là việc di chuyển trong không gian tìm kiếm không có định hướng, dẫn tới phải xem xét nhiều trạng thái. Đối với những bài toán thực tế có không gian trạng thái lớn, tìm kiếm mù thường không thực tế do có độ phức tạp tính toán và yêu cầu bộ nhớ lớn.

Để giải quyết vấn đề trên, chiến lược tìm kiếm có thông tin sử dụng thêm những thông tin phụ từ bài toán để định hướng tìm kiếm, cụ thể là lựa chọn thứ tự mở rộng nút theo hướng mau dẫn tới đích hơn.

Nguyên tắc chung của tìm kiếm có thông tin là sử dụng một hàm $f(n)$ để đánh giá độ "tốt" tiềm năng của nút n, từ đó chọn nút n có hàm f tốt nhất để mở rộng trước.

Trên thực tế, việc xây dựng hàm $f(n)$ phản ánh chính xác độ tốt của nút thường không thực hiện được, thay vào đó ta sử dụng các giá trị ước lượng cho $f(n)$.

Trong phần này ta sẽ xem xét hai chiến lược tìm kiếm có thông tin, đó là *tìm kiếm tham lam* và *tìm kiếm A**.

2.4.1. Tìm kiếm tham lam (Greedy Search)

Phương pháp: xem xét mở rộng nút có giá thành đường đi tới đích nhỏ nhất trước.

Trong phương pháp này, để đánh giá độ tốt của một nút, ta sử dụng hàm đo giá thành đường đi từ nút đó tới đích. Tuy nhiên, do không biết được chính xác giá thành đường đi từ một nút tới đích, ta chỉ có thể ước lượng giá trị này. Hàm ước lượng độ tốt, hay giá thành đường đi từ một nút n tới địch gọi là *hàm heuristic* và ký hiệu $h(n)$. Như vậy, đối với thuật toán tham lam, ta có $f(n) = h(n)$.

Do hàm $h(n)$ chỉ là hàm ước lượng giá thành đường đi tới đích nên có thể nói rằng tìm kiếm tham lam mở rộng nút trông có vẻ gần đích nhất trước các nút khác. Thuật toán được gọi là "tham lam" do thuật toán chỉ quan tâm tới việc lựa chọn nút trông có vẻ tốt nhất ở mỗi bước mà không quan tâm tới việc trong tương lai việc dùng nút đó có thể không phải là tối ưu.

Điều kiện của hàm $h(n)$: $h(n) \geq 0$

Nhận xét: tìm kiếm heuristic tương tự DFS nhưng cho phép quay lui khi gặp bế tắc

Ví dụ hàm heuristic $h(n)$. Khi tìm đường trên bản đồ, hàm heuristic cho một thành phố có thể tính bằng khoảng cách theo đường chim bay giữa thành phố đó với thành phố đích cần đến.

Đặc điểm:

- Không có tính đầy đủ do có khả năng tạo thành vòng lặp ở một số nút.
- Độ phức tạp về thời gian: $O(b^m)$ (thuật toán nhanh hơn BFS, và có thể cũng nhanh hơn DFS nếu tồn tại heuristic tốt). Tuy nhiên trong trường hợp heuristic không tốt thì thuật toán sẽ đi sai hướng và do vậy gần giống với tìm sâu.

- Độ phức tạp về không gian lưu trữ: O(b^m) (lưu tất cả các nút trong bộ nhớ)
- Thuật toán không tối ưu

2.4.2. Thuật toán A*

Một trong những nhược điểm của tìm kiếm tham lam là không cho lời giải ngắn nhất. Lý do tìm kiếm tham lam không đảm bảo tìm ra đường đi ngắn nhất là do thuật toán chỉ quan tâm tới khoảng cách ước lượng từ một nút tới đích mà không quan tâm tới đường đi từ nút xuất phát tới nút đó. Trong trường hợp khoảng cách từ nút xuất phát tới nút đang xét lớn sẽ làm tổng độ dài đường đi từ xuất phát tới đích qua nút hiện thời lớn lên.

Để khắc phục nhược điểm này, thuật toán A* sử dụng hàm đánh giá f(n) với hai thành phần, thành phần thứ nhất là đường đi từ nút đang xét tới nút xuất phát, thành phần thứ hai là khoảng cách ước lượng tới đích.

Phương pháp: khắc phục nhược điểm của thuật toán tham lam, thuật toán A* sẽ tính f(n) = g(n) + h(n). Trong đó:

- g(n) là giá thành đường đi từ nút xuất phát đến nút n
- h(n) là giá thành ước lượng đường đi từ nút n đến nút đích, h(n) là hàm heuristic.

Thuật toán A* yêu cầu hàm h(n) là hàm *chấp nhận được* (admissible).

Định nghĩa*: Hàm h(n) được gọi là chấp nhận được nếu h(n) không lớn hơn khoảng cách thực tế từ n tới nút đích.*

Thuật toán: *A**(*Q*, *S*, *G*, *P*, *c*, *h*)

- Đầu vào: bài toán tìm kiếm, hàm heuristic *h*
- Đầu ra: đường đi ngắn nhất từ nút xuất phát đến nút đích
- Khởi tạo: tập các nút biên (nút mở) $O \leftarrow S$

While(*O* không rỗng) do

1. Lấy nút *n* có *f*(*n*) nhỏ nhất ra khỏi *O*
2. Nếu *n* thuộc *G*, return(đường đi tới *n*)
3. Với mọi $m \in P(n)$
   i. $g(m) = g(n) + c(m, n)$
   ii. $f(m) = g(m) + h(m)$
   iii. Thêm *m* vào *O* cùng giá trị *f*(*m*)

Return: không tìm được đường đi

Hình 2.9. Thuật toán A*

Nhận xét:

- Thuật toán cho kết quả tối ưu nếu hàm heuristic h là hàm *chấp nhận được*.
- Thuật toán đầy đủ trừ trường hợp có vô số các node với hàm f có giá trị rất nhỏ nằm giữa node xuất phát và node đích.
- Thời gian: O(b^m)
- Trong tất cả các thuật toán tìm kiếm tối ưu thì thuật toán A* hiệu quả nhất với độ phức tạp của thuật toán là nhỏ nhất

2.4.3. Các hàm heuristic

Các hàm heuristic h(n) được xây dựng tùy thuộc vào bài toán cụ thể. Cùng một loại bài toán chúng ta có thể có rất nhiều hàm heuristic khác nhau. Chất lượng hàm heuristic ảnh hưởng rất nhiều đến quá trình tìm kiếm.

Hàm heuristic h(n) được gọi là chấp nhận được khi:

$h(n) \leq h^*(n)$

trong đó h*(n) là giá thành đường đi thực tế từ n đến node đích. Lưu ý rằng hàm h(n)=0 với mọi n, là hàm chấp nhận được.

Ví dụ:

Đường chim bay như nhắc tới ở trên là một ví dụ của hàm heuristic chấp nhận được.

Ngoài ra, ta sẽ xem xét một số hàm heuristic cho bài toán trò đố 8 ô.

3	1	6
5		4
2	7	8

	1	2
3	4	5
6	7	8

Ta có thể sử dụng hai hàm heuristic sau.

- $h_1(n)$: số ô đặt sai chỗ. Chẳng hạn nếu hình bên phải là trạng thái đích và hình bên trái là trạng thái u thì trạng thái bên trái có $h_1(u) = 5$ do có 5 ô là các ô 3, 6, 4, 5, 2 nằm sai vị trí. Có thể nhận thấy h_1 là hàm chấp nhận được do muốn di chuyển từ trạng thái bên trái sang trạng thái đích ta phải chuyển vị trí tất cả những ô đứng sai, trong khi để di chuyển mỗi ô sai, ta cần ít nhất một nước đi. Như vậy độ dài đường đi luôn lớn hơn hoặc bằng h1.
- $h_2(n)$: tổng khoảng cách Manhattan giữa vị trí hiện thời của mỗi ô tới vị trí đúng của ô đó. Khoảng cách Manhattan được tính bằng số ít nhất các dịch chuyển theo hàng hoặc

cột để đưa một quân tới vị trí của nó trong trạng thái đích. Ví dụ, để đưa quân 2 tới vị trí đích ta cần 4 dịch chuyển và do vậy khoảng cách Manhattan của 2 tới đích là 4. Giá trị h_2 của trạng thái u trên hình bên trái sẽ bằng $h_2(u) = 1 + 4 + 1 + 2 + 1$. Hàm h_2 cũng là hàm chấp nhận được. Thật vậy, để di chuyển một ô tới vị trí đích, ta cần ít nhất số nước đi bằng khoảng cách Manhattan từ ô đó tới đích. Như vậy, số nước để di chuyển toàn bộ các ô đứng sai sẽ lớn hơn hoặc bằng tổng khoảng cách Manhattan như cách tính h_2.

Hàm heuristic trội

Ví dụ trên cho thấy, với cùng một bài toán, ta có thể xây dựng đồng thời nhiều hàm heuristic chấp nhận được. Vấn đề đặt ra khi đó là nên dùng hàm nào trong thuật toán tìm kiếm, hàm được chọn phải là hàm tốt hơn, tức là hàm nhanh dẫn tới kết quả hơn.

Giả sử có hai hàm heuristic chấp nhận được $h_1(n)$ và $h_2(n)$. Nếu $h_1(n) \le h_2(n)$ với mọi n thì ta nói rằng $h_2(n)$ trội hơn so với $h_1(n)$. Rõ ràng hàm $h_2(n)$ mang nhiều thông tin hơn và do vậy là hàm tốt hơn do dẫn tới kết quả nhanh hơn.

Trong trường hợp trong hai hàm $h_1(n)$ và $h_2(n)$ không có hàm trội hơn thì ta có thể tạo ra hàm $h'(n) = \max(h_1(n), h_2(n))$ với mọi n. Rõ ràng, $h'(n)$ là hàm trội hơn hai hàm ban đầu.

Cách xây dựng hàm heuristic

Việc lựa chọn hàm heuristic phụ thuộc vào bài toán cụ thể. Nguyên tắc chung để xây dựng hàm heuristic cho một bài toán là nới lỏng các ràng buộc của bài toán đó. Ví dụ, đối với hàm $h_1(n)$ trong trò đố 8 ô, ta đã bỏ ràng buộc về việc các ô phải di chuyển từng bước để đến được vị trí đích. Hay đối với heuristic đường chim bay, ta đã nới rỏng ràng buộc về việc di chuyển trên đường bộ và giả sử có thể di chuyển thẳng từ một điểm tới đích.

2.4.4. Thuật toán IDA* (thuật toán A* sâu dần)

Thuật toán A* có những ưu điểm quan trọng như tính tối ưu và tính đầy đủ. Tuy nhiên, nếu hàm heuristic không tốt, thuật toán sẽ phải xem xét nhiều trạng thái và có yêu cầu bộ nhớ trong trường hợp xấu nhất là $O(b^m)$. Yêu cầu bộ nhớ theo hàm mũ như vậy làm giảm khả năng sử dụng A*. Để giải quyết vấn đề này có thể sử dụng phiên bản của A* được gọi là A* sâu dần (Iterative Deepening A*) có yêu cầu bộ nhớ tỷ lệ tuyến tính với độ sâu.

Phương pháp: Nguyên tắc của A* sâu dần là lặp lại việc tìm kiếm theo chiều sâu trên các cây tìm kiếm con có giá trị hàm $f(n)$ không lớn hơn các ngưỡng 0, α, 2α, 3α, .v.v..

Cụ thể:

1. Tìm kiếm sâu dần (DFS), không mở rộng nút có hàm f > 0, nếu tìm được đích thì dứng lại.
2. Tìm kiếm sâu dần (DFS), không mở rộng nút có hàm f > α, nếu tìm được đích thì dứng lại.
2. Tìm kiếm sâu dần , không mở rộng nút có hàm f > 2α, nếu tìm được đích thì dứng lại.

...

Ở đây, α là giá trị được thêm vào ngưỡng sau mỗi vòng lặp. Để mỗi vòng lặp có thể xét thêm các nút mới α cần có giá trị lớn hơn hoặc bằng giá thành nhỏ nhất để di chuyển giữa hai trạng thái trong không gian tìm kiếm. Ở đây cần lưu ý cách chọn α. Nếu α nhỏ, sau mỗi lần tăng ngưỡng, cây tìm kiếm mới sẽ thêm được ít nút do với cây tìm kiếm cũ và do vậy cần lặp lại quá trình tìm sâu nhiều lần, dẫn tới tăng độ phức tạp tính toán. Ví dụ, trong trường hợp đặc biệt, khi giá trị của f(n) trên mọi nút đều khác nhau, mỗi bước lặp sẽ chỉ xem xét thêm được một nút so với bước trước. Khi đó, nếu A* cần mở rộng N nút, thì A* sâu dần sẽ phải mở rộng 1+2+…+N = $O(N^2)$.

Giải pháp cho vấn đề độ phức tạp tính toán là sử dụng mức độ tăng ngưỡng $\beta > \alpha$, sao cho tại mỗi bước lặp sẽ mở rộng cây tìm kiếm thêm một số nút mới. Giá trị β như vậy cho phép giảm thời gian tìm kiếm, tuy nhiên chỉ trả lại lời giải β-tối ưu trong trường hợp xấu nhất, tức là nếu thuật toán tìm được lời giải m^* thì ta có $g(m^*) < g^* + \beta$.

Thuật toán A* sâu dần chi tiết được thể hiện trên hình dưới đây:

Thuật toán: A*(Q, S, G, P, c, h)

- Đầu vào: bài toán tìm kiếm, hàm heuristic h
- Đầu ra: đường đi ngắn nhất từ nút xuất phát đến nút đích
- Khởi tạo: danh sách các nút biên (nút mở) O ← S

  giá trị $i = 0$ là ngưỡng cho hàm f

While(1) do

1. while (O không rỗng) do
   a) Lấy nút n từ đầu O
   b) Nếu n thuộc G, return(đường đi tới n)
   c) Với mọi m ∈ P(n)
      i. g(m) = g(n) + c(m, n)
      ii. f(m) = g(m) + h(m)
      iii. If f(m) ≤ i then Thêm m vào đầu O
2. i ← i + β, $O \leftarrow S$

Hình 2.10. Thuật toán A* sâu dần

Tính chất:

- Thuật toán đầy đủ và β-tối ưu

- Yêu cầu bộ nhớ tuyến tính (chỉ nhớ nhánh, tương đương sâu dần)
- Độ phức tạp tính toán lớn hơn của thuật toán A*
- Không phụ thuộc vào h(n) (sau mỗi lần tăng lại làm lại từ đầu → không sử dụng h(n) để thu hẹp nhánh cần tìm).

2.5. TÌM KIẾM CỤC BỘ

Các thuật toán tìm kiếm ở trên đều dựa trên việc khảo sát không gian tìm kiếm một cách hệ thống bằng cách ghi lại những đường đi đã qua cùng với thông tin về phương án đã hoặc chưa được xem xét tại mỗi trạng thái trên đường đi. Vấn đề của thuật toán như vậy là việc sử dụng đường đi để khảo sát không gian tìm kiếm một cách hệ thống làm tăng số lượng trạng thái cần xem xét đồng thời đòi hỏi ghi nhớ nhiều trạng thái và do vậy không thích hợp với bài toán có không gian trạng thái lớn..

Trong phần này ta sẽ xem xét các thuật toán *tìm kiếm cục bộ* (local search), còn được gọi là tìm kiếm *cải thiện dần* (iterative improvement). Tìm kiếm cục bộ thường được sử dụng cho những bài toán tối ưu hóa tổ hợp hoặc tối ưu hóa rời rạc, tức là những bài toán trong đó cần tìm trạng thái tối ưu hoặc tổ hợp tối ưu trong không gian rời rạc các trạng thái, và không quan tâm tới đường đi dẫn tới trạng thái đó.

Bài toán tối ưu hóa tổ hợp (tối ưu hóa rời rạc) có những đặc điểm sau.

- Tìm trạng thái tối ưu cực đại hóa hoặc cực tiểu hóa hàm mục tiêu. Không quan tâm tới đường đi.
- Không gian trạng thái rất lớn.
- Không thể sử dụng các phương pháp tìm kiếm trước để xem xét tất cả không gian trạng thái
- Thuật toán cho phép tìm lời giải tốt nhất với độ phức tạp tính toán nhỏ. Thuật toán cũng chấp nhận lời giải tương đối tốt.

Ví dụ: tối ưu hóa tổ hợp là lớp bài toán có nhiều ứng dụng trên thực tế. Có thể kể ra một số ví dụ sau: bài toán lập lịch, bảng biểu, thiết kế transitor, triệu con hậu,…

Tìm kiếm cục bộ được thiết kế cho bài toán tìm kiếm với không gian trạng thái rất lớn và cho phép tìm kiếm trạng thái tương đối tốt với thời gian tìm kiếm chấp nhận được.

Ý tưởng: nguyên tắc chung của tìm kiếm cục bộ

- Chỉ quan tâm đến trạng thái đích, không quan tâm đến đường đi.
- Mỗi trạng thái tương ứng với một lời giải (chưa tối ưu) → cải thiện dần bằng cách chỉ quan tâm tới một trạng thái hiện thời, sau đó xem xét để để chuyển sang trạng thái hàm xóm của trạng thái hiện thời (thường là trạng thái có hàm mục tiêu tốt hơn).

- Thay đổi trạng thái bằng cách thực hiện các chuyển động (trạng thái nhận được từ trạng thái n bằng cách thực hiện các chuyển động được gọi là hàng xóm của n).

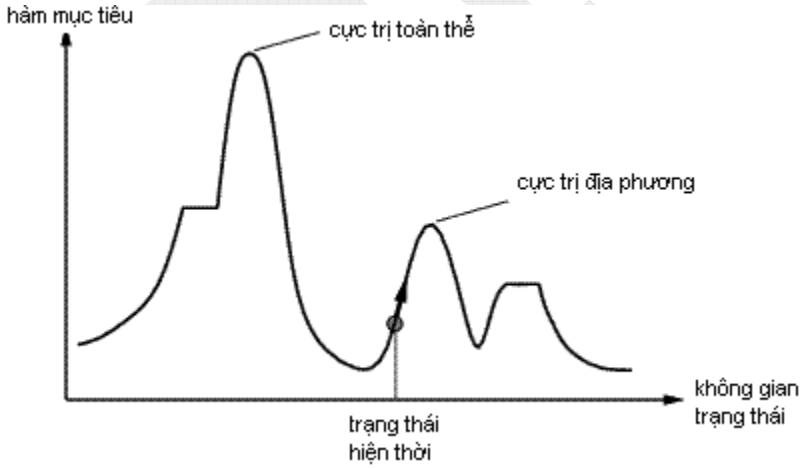
Do tìm kiếm cục bộ chỉ quan tâm tới trạng thái hiện thời và hàng xóm nên cần ít bộ nhớ hơn nhiều so với các phương pháp tìm kiếm hệ thống ở trên. Tìm kiếm cục bộ thường cho phép tìm được lời giải chấp nhận được kể cả khi bài toán lớn đến mức không dùng được những phương pháp tìm kiếm hệ thống.

Phát biểu bài toán:

Bài toán tìm kiếm cục bộ được cho bởi những thành phần sau:

- Không gian trạng thái X
- Tập chuyển động để sinh ra hàng xóm
- Hàm mục tiêu Obj: X → R
- Yêu cầu: Tìm trạng thái X* sao cho Obj(X*) là min hoặc max.

Có thể minh họa bài toán tìm kiếm cục bộ bằng cách xem xét hình 2.11. Trục hoành trên hình vẽ thể hiện không gian các trạng thái (để cho đơn giản, không gian trạng thái ở đây được thể hiện trong không gian một chiều dưới dạng các điểm trên trục hoành), trục tung là độ lớn của hàm mục tiêu. Yêu cầu bài toán tối ưu hóa tổ hợp là tìm được trạng thái (điểm trên trục hoành) có hàm mục tiêu lớn nhất. Lưu ý, hình vẽ minh họa trường hợp cần tìm trạng thái với hàm mục tiêu lớn nhất, tuy nhiên trong bài toán khác có thể yêu cầu tìm trạng thái với hàm mục tiêu nhỏ nhất.



Hình 2.11. Bài toán tìm kiếm cục bộ với không gian trạng thái và hàm mục tiêu

2.5.1. Thuật toán leo đồi (Hill climbing)

Leo đồi là tên chung để chỉ một họ các thuật toán. Thuật toán thực hiện bằng cách tạo ra hàng xóm cho trạng thái hiện thời và di chuyển sang hàng xóm có hàm mục tiêu tốt hơn, tức là di chuyển lên cao đối với trường hợp cần cực đại hóa hàm mục tiêu. Thuật toán dừng lại khi đạt tới

một đỉnh của đồ thị hàm mục tiêu, tương ứng với trạng thái không có hàng xóm nào tốt hơn. Đỉnh này có thể là đỉnh cao nhất, hoặc cũng là những đỉnh thấp hơn (hình 2.11). Trong trường hợp thứ nhất, thuật toán tìm được giá trị cực trị, trong trường hợp thứ hai thuật toán chỉ tìm được cực trị địa phương. Thuật toán leo đồi không lưu lại những trạng thái đã qua, đồng thời không nhìn xa hơn hàng xóm của trạng thái hiện thời.

a) Di chuyển sang trạng thái tốt nhất

Có nhiều phiên bản khác nhau của thuật toán leo đồi. Một trong những phiên bản thông dụng nhất có tên là leo đồi *di chuyển sang trạng thái tốt nhất* (best-improvement hill climbing). Phiên bản này của leo đồi lựa chọn trong số hàng xóm hiện thời hàng xóm có hàm mục tiêu tốt nhất. Nếu hàng xóm đó tốt hơn trạng thái hiện thời thì di chuyển sang hàm xóm đó. Nếu ngược lại thì kết thúc và trả về trạng thái hiện thời. Thuật toán đầy đủ được thể hiện trên hình 2.12.

Đầu vào: bài toán tối ưu tổ hợp

Đầu ra: trạng thái với hàm mục tiêu lớn nhất (hoặc cực đại địa phương)

1. Chọn ngẫu nhiên trạng thái x
2. Gọi Y là tập các trạng thái hàng xóm của x
3. Nếu $\forall yi \in Y$: Obj (yi) < Obj (x) thì

Kết thúc và trả lại x là kết quả

1. $x \leftarrow yi$, trong đó i = argmax$i$ (Obj (yi))
2. Go to 2

Hình 2.12. Thuật toán leo đồi di chuyển sang trạng thái tốt nhất

Đặc điểm của leo đồi

- Đơn giản,dễ lập trình, không tốn bộ nhớ do không phải lưu lại bất kỳ thứ gì, chỉ lưu lại trạng thái tạm thời và các hàng xóm.
- Dễ bị lời giải tối ưu cục bộ (cực trị địa phương) tương ứng với đỉnh các "đồi" thấp trong hình 2.11. Để khắc phục phần nào vấn đề này, thuật toán được thực hiện nhiều lần, mỗi lần sử dụng một trạng thái xuất phát sinh ngẫu nhiên khác với trạng thái xuất phát trong những lần trước đó.

Khi thiết kế thuật toán leo đồi, việc lựa chọn chuyển động rất quan trọng. Nếu nhiều chuyển động sẽ sinh ra nhiều hàng xóm, do vậy việc chọn ra hàng xóm tốt nhất đòi hỏi nhiều thời gian do phải tính hàm mục tiêu cho tất cả hàng xóm. Ngược lại, nếu sinh ra tập hàng xóm nhỏ sẽ dễ dẫn tới cực trị địa phương do không vượt qua được những "hố" nhỏ trên đường đi.

b) Leo đồi ngẫu nhiên

Leo đồi ngẫu nhiên (stochastic hill climbing) là một phiên bản khác của leo đồi. Thay vì tìm ra hàng xóm tốt nhất, phiên bản này lựa chọn ngẫu nhiên một hàng xóm. Nếu hàng xóm đó tốt hơn trạng thái hiện thời, hàng xóm đó sẽ được chọn làm trạng thái hiện thời và thuật toán lặp lại. Ngược lại, nếu hàng xóm được chọn không tốt hơn, thuật toán sẽ chọn ngẫu nhiên một hàng xóm khác và so sánh. Thuật toán kết thúc và trả lại trạng thái hiện thời khi đã hết "kiên nhẫn". Thông thường, kiên nhẫn được cho bằng số lượng tối đa hàng xóm mà thuật toán xem xét trong mỗi bước lặp hoặc trong toàn bộ thuật toán.

Thuật toán leo đồi ngẫu nhiên được thể hiện trên hình 2.13.

Đầu vào: bài toán tối ưu tổ hợp

Đầu ra: trạng thái với hàm mục tiêu lớn nhất (hoặc cực đại địa phương)

---

1. Chọn ngẫu nhiên trạng thái x
2. Gọi Y là tập các trạng thái hàng xóm của x
3. Chọn ngẫu nhiên $y_i \in Y$
4. Nếu Obj (yi) > Obj (x) thì

   $x \leftarrow yi$
5. Go to 2 nếu chưa hết kiên nhẫn

Hình 2.13. Thuật toán leo đồi ngẫu nhiên

Các nghiên cứu cho thấy, trong một số trường hợp, leo đồi ngẫu nhiên cho kết quả nhanh hơn và có thể tránh được một số cực trị địa phương.

2.5.2. Thuật toán tôi thép (Simulated Annealing)

Một vấn đề lớn với leo đồi là thuật toán không có khả năng "đi xuống" và do vậy không thoát khỏi được cực trị địa phương khi đã rơi vào. Ngược lại, cách di chuyển hoàn toàn ngẫu nhiên (random walk) có thể khảo sát toàn bộ không gian trạng thái nhưng không hiệu quả. *Thuật toán tôi thép* (simulated annealing) là một phương pháp tìm kiếm cục bộ cho phép giải quyết phần nào vấn đề cực trị địa phương một cách tương đối hiệu quả.

Có thể coi tôi thép là phiên bản của thuật toán leo đồi ngẫu nhiên, trong đó thuật toán chấp nhận cả những trạng thái kém hơn trạng thái hiện thời với một xác suất p nào đó. Cụ thể là khi lựa chọn ngẫu nhiên một hàng xóm, nếu hàng xóm đó kém hơn trạng thái hiện thời, thuật toán có thể quyết định di chuyển sang đó với một xác suất p.

Vấn đề quan trọng đối với thuật toán là lựa chọn xác suất *p* thế nào. Nguyên tắc chung là không chọn *p* cố định, giá trị *p* được xác định dựa trên hai yếu tố sau.

- Nếu trạng thái mới kém hơn nhiều so với trạng thái hiện thời thì p phải giảm đi. Có nghĩa là xác suất chấp nhận trạng thái tỷ lệ nghịch với độ kém của trạng thái đó. Gọi $\Delta(x,y) = Obj(x) - Obj(y)$ trong đó x là trạng thái hiện thời, ta cần chọn p tỷ lệ nghịch với $\Delta(x,y)$.
- Theo thời gian, giá trị của *p* phải giảm dần. Ý nghĩa của việc giảm p theo thời gian là do khi mới bắt đầu, thuật toán chưa ở vào vùng trạng thái tốt và do vậy chấp nhận thay đổi lớn. Theo thời gian, thuật toán sẽ chuyển sang vùng trạng thái tốt hơn và do vậy cần hạn chế thay đổi.

Thuật toán tôi thép được thể hiện trên hình 2.14.

SA(X, Obj, N, m, x, C) //Obj càng nhỏ càng tốt

Đầu vào: số bước lặp m

trạng thái bắt đầu x (chọn ngẫu nhiên)

sơ đồ làm lạnh C

Đầu ra: trạng thái tốt nhất x*

Khởi tạo: x* = x

For i = 1 to m

1. chọn ngẫu nhiên $y \in N(x)$

   a) tính $\Delta(x,y) = Obj(y) - Obj(x)$

   b) **if** $\Delta(x,y) < 0$ **then** p = 1

   c) **else** $p = e - \Delta(x,y)/T$

   d) **if** rand[0,1] < p **then** $x \leftarrow y$

      if $Obj(x) < Obj(x^*)$ then $x^* \leftarrow x$

2. giảm T theo sơ đồ C

return x* //x* là trạng thái tốt nhất trong số những trạng thái đã xem xét

Hình 2.14. Thuật toán tôi thép

Thuật toán tôi thép vừa trình bày dựa trên một hiện tượng cơ học là quá trình làm lạnh kim loại để tạo ra cấu trúc tinh thể bền vững. Hàm mục tiêu khi đó được đo bằng độ vững chắc của

cấu trúc tinh thể. Khi còn nóng, mức năng lượng trong kim loại cao, các nguyên tử kim loại có khả năng di chuyển linh động hơn. Khi nhiệt độ giảm xuống, tinh thể dần chuyển tới trạng thái ổn định và tạo ra mạng tinh thể. Bằng cách thay đổi nhiệt độ hợp lý, có thể tạo ra những mạng tinh thể rất rắn chắc.

Chính vì sự tương tự với cách tôi kim loại như vậy nên trong thuật toán, xác suất p giảm theo thời gian dựa vào một công thức gọi là sơ đồ làm lạnh C. Có nhiều dạng sơ đồ làm lạnh khác nhau. Sau đây là ví dụ một sơ đồ làm lạnh.

Sơ đồ làm lạnh C:

$$T_{t+1} = T_0 * \alpha^{t*k}$$

Trong đó:

$T_0 > 0$, α thuộc (0,1), 1 < k < m

t càng tăng → α càng nhỏ → T càng nhỏ

Khi T → ∞: p = 1 với mọi Δ(x, y) → tương đương với chuyển động ngẫu nhiên

Khi T → 0: p = 0 với mọi Δ(x, y) → đưa về trường hợp leo đồi ngẫu nhiên

Việc lựa chọn các tham số cho sơ đồ làm lạnh thường được thực hiện bằng cách thực nghiệm với từng bài toán cụ thể.

Thuật toán tôi thép được dùng nhiều trong việc thiết kế vi mạch có độ tích hợp lớn cũng như giải quyết những bài toán tối ưu hóa tổ hợp có kích thước lớn trên thực tế.

2.5.3. Một số thuật toán tìm kiếm cục bộ khác

Ngoài phương pháp leo đồi và tôi thép, nhiều thuật toán tìm kiếm cục bộ khác được đề xuất và sử dụng trong thực tế, trong đó phải kể đến những thuật toán sau:

Tabu search. Thuật toán lưu một danh sách những trạng thái đã đi qua gọi là tabu list (tạm dịch là danh sách cấm). Khi duyệt tập hàng xóm, những trạng thái thuộc danh sách này sẽ bị loại không được xem xét.

Giải thuật di truyền (genetic algorithm). Giải thuật di truyền có thể xem như một phiên bản leo đồi ngẫu nhiên được thực hiện song song. Thuật toán mã hóa các trạng thái dưới dạng các chuỗi gen, sau đó thực hiện ba thao tác biến đổi chính là: chọn lọc, lai giống và đột biến để sinh ra trạng thái mới. Quá trình đi tới trạng thái tốt được thực hiện dựa trên sự tương tự với quá trình chọn lọc cá thể tốt theo thuyết tiến hóa của Đác Uyn.

CHƯƠNG 3: BIỂU DIỄN TRI THỨC VÀ SUY DIỄN LOGIC

3.1. SỰ CẦN THIẾT SỬ DỤNG TRI THỨC TRONG GIẢI QUYẾT VẤN ĐỀ

Sự cần thiết của tri thức và suy diễn

Một yêu cầu quan trọng đối với hệ thống thông minh là phải có khả năng sử dụng tri thức và suy diễn. Rất khó để đạt được những hành vi thông minh và mềm dẻo mà không có tri thức về thế giới xung quanh và khả năng suy diễn với tri thức đó. Sử dụng tri thức và suy diễn đem lại những lợi ích sau.

- Hệ thống dựa trên tri thức có tính mềm dẻo cao. Việc kết hợp tri thức và suy diễn cho phép tạo ra tri thức khác, giúp hệ thống đạt được những mục tiêu khác nhau, đồng thời có khả năng suy diễn về bản thân mục tiêu. Chương trước đã đề cập tới kỹ thuật giải quyết vấn đề bằng cách tìm kiếm. Những hệ thống tìm kiếm chỉ sử dụng tri thức hạn chế, thể hiện trong việc biểu diễn bài toán và các heuristic. Hệ thống như vậy không có khả năng tự thay đổi mục đích cũng như không có khả năng hành động một cách mềm dẻo, ngoài những gì chứa trong giải thuật và mô tả bài toán. Vì vậy kỹ thuật tìm kiếm là chưa đủ để tạo ra hệ thống thông minh.
- Sử dụng tri thức và suy diễn cho phép hệ thống hoạt động cả trong trường hợp thông tin quan sát về môi trường là không đầy đủ. Hệ thống có thể kết hợp tri thức chung đã có để bổ sung cho thông tin quan sát được khi cần ra quyết định. Ví dụ, khi giao tiếp bằng ngôn ngữ tự nhiên, có thể hiểu một câu ngắn gọn nhờ sử dụng tri thức đã có về ngữ cảnh giao tiếp và nội dung liên quan tới chủ đề.
- Việc sử dụng tri thức thuận lợi cho việc xây dựng hệ thống. Thay vì lập trình lại hoàn toàn hệ thống, có thể thay đổi tri thức trang bị cho hệ thống và mô tả mục đích cần đạt được, đồng thời giữ nguyên thủ tục suy diễn.

Biểu diễn tri thức

Để có thể sử dụng tri thức, tri thức cần được biểu diễn dưới dạng thuận tiện cho việc mô tả và suy diễn. Nhiều ngôn ngữ và mô hình biểu diễn tri thức đã được thiết kế để phục vụ mục đích này. Ngôn ngữ biểu diễn tri thức phải là ngôn ngữ hình thức để tránh tình trạng nhập nhằng như thường gặp trong ngôn ngữ tự nhiên. Một ngôn ngữ biểu diễn tri thức tốt phải có những tính chất sau:

- Ngôn ngữ phải có khả năng biểu diễn tốt, tức là cho phép biểu diễn mọi tri thức cần thiết cho bài toán.
- Cần đơn giản và hiệu quả, tức là cho phép biểu diễn ngắn gọn tri thức, đồng thời cho phép đi đến kết luận với khối lượng tính toán thấp.
- Gần với ngôn ngữ tự nhiên để thuận lợi cho người sử dụng trong việc mô tả tri thức.

Sau khi đã có ngôn ngữ biểu diễn tri thức, tri thức về thế giới của bài toán được biểu diễn dưới dạng tập hợp các câu và tạo thành cơ sở tri thức (ký hiệu KB trong các phần sau). Thủ tục suy diễn được sử dụng để tạo ra những câu mới nhằm trả lời cho các vấn đề của bài toán. Thay vì trực tiếp hành động trong thế giới thực của bài toán, hệ thống có thể suy diễn dựa trên cơ sở tri thức được tạo ra.

Logic

Trong chương này, ta sẽ xem xét lô gic với vai trò là phương tiện để biểu diễn tri thức và suy diễn.

Dạng biểu diễn tri thức cổ điển nhất trong máy tính là logic, với hai dạng phổ biến là logic mệnh đề và logic vị từ. Logic là một ngôn ngữ biểu diễn tri thức trong đó các câu nhận hai giá trị đúng (True) hoặc sai (False)[1] được xác định bởi 3 thành phần sau:

- Cú pháp bao gồm các ký hiệu và các quy tắc liên kết các ký hiệu để tạo thành câu hay biểu thức logic. Một ví dụ cú pháp là các ký hiệu và quy tắc xây dựng biểu thức trong số học và đại số.
- Ngữ nghĩa của ngôn ngữ cho phép ta xác định ý nghĩa của các câu trong một miền nào đó của thế giới hiện thực, xác định các sự kiện hoặc sự vật phản ánh thế giới thực của câu mệnh đề. Đối với logic, ngữ nghĩa cho phép xác định câu là đúng hay sai trong thế giới của bài toán đang xét.
- Thủ tục suy diễn là phương pháp cho phép sinh ra các câu mới từ các câu đã có hoặc kiểm tra liệu các câu có phải là hệ quả logic của nhau.

Logic đã cung cấp cho các nhà nghiên cứu một công cụ hình thức để biểu diễn và suy luận tri thức. Các bài sau sẽ trình bày về hai dạng logic mệnh đề và logic vị từ trong biểu diễn tri thức.

3.2. LOGIC MỆNH ĐỀ

3.2.1. Cú pháp

Logic mệnh đề là logic rất đơn giản, tuy khả năng biểu diễn của nó còn một số hạn chế nhưng thuận tiện cho ta đưa vào nhiều khái niệm quan trọng trong logic.

Cú pháp của logic mệnh đề bao gồm tập các ký hiệu và tập các quy tắc kết hợp các ký hiệu tạo thành công thức.

a) Các ký hiệu

Các ký hiệu được dùng trong logic mệnh đề bao gồm:

- Các ký hiệu chân lý: True (ký hiệu T) và False (ký hiệu F).
- Các ký hiệu mệnh đề (còn được gọi là các biến mệnh đề và thường được ký hiệu bằng các chữ cái): P, Q,...

[1] Một số hệ thống logic được phát triển về sau sử dụng nhiều giá trị hơn như logic đa trị, logic mờ

- Các kết nối logic $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$.
- Các dấu ngoặc.

b) Các câu hay công thức

Mọi ký hiệu chân lý và ký hiệu mệnh đề là câu.

Ví dụ: True, P

Thêm ngoặc ra ngoài một câu sẽ được một câu.

Kết hợp các câu bằng phép nối logic sẽ tạo ra câu mới. Cụ thể là:

Nếu A và B là câu thì:

$(A \wedge B)$ (đọc "A hội B" hoặc "A và B")

$(A \vee B)$ (đọc "A tuyển B" hoặc "A hoặc B")

$(\neg A)$ (đọc "phủ định A")

$(A \Rightarrow B)$ (đọc "A kéo theo B" hoặc "nếu A thì B"). Phép kéo theo còn được gọi là quy tắc "nếu – thì"

$(A \Leftrightarrow B)$ (đọc "A và B kéo theo nhau")

là các câu.

Để cho ngắn gọn các công thức được bỏ đi các cặp dấu ngoặc không cần thiết. Chẳng hạn, thay cho $((A \vee B) \wedge C)$ ta sẽ viết là $(A \vee B) \wedge C$. Trong trường hợp một câu chứa nhiều phép nối, các phép nối sẽ được thực hiện theo thứ tự sau:

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Các câu là các ký hiệu mệnh đề sẽ được gọi là các câu đơn hoặc câu nguyên tử. Các câu không phải là câu đơn được gọi là câu phức hợp. Nếu P là ký hiệu mệnh đề thì P và $\neg$ P được gọi là literal, P là literal dương, còn $\neg$ P là literal âm. Câu phức hợp có dạng $A_1 \vee ... \vee A_m$ trong đó A_i là các literal sẽ được gọi là câu tuyển (clause).

3.2.2. Ngữ nghĩa

Ngữ nghĩa của logic mệnh đề cho phép xác định một câu (công thức) logic là đúng hay sai trong thế giới của bài toán đang xét, tức là cách diễn giải của các ký hiệu mệnh đề, ký hiệu chân lý và phép nối logic trong thế giới đó.

Trong logic mệnh đề, người sử dụng xác định giá trị đúng hay sai cho ký hiệu mệnh đề. Mỗi ký hiệu mệnh đề có thể tương ứng với một phát biểu (mệnh đề), ví dụ ký hiệu mệnh đề A có thể tương ứng với phát biểu: "Hà Nội là thủ đô của Việt Nam" hoặc bất kì một phát biểu nào khác. Một phát biểu chỉ có thể đúng (True) hoặc sai (False). Chẳng hạn, phát biểu "Hà Nội là thủ đô của Việt Nam " là đúng còn phát biểu " Lợn là gia cầm " là sai.

Một minh họa là một cách gán cho mỗi biến mệnh đề một giá trị chân lý True hoặc False. Nếu biến mệnh đề A được gán giá trị chân lý True/False (A <-True/ A <-False) thì ta nói mệnh đề A đúng/sai trong minh họa đó.

Trong một minh họa, ý nghĩa của các câu phức hợp được xác định bởi ý nghĩa của các kết nối logic. Phép nối logic cho phép quy giá trị câu phức về giá trị các câu đơn giản hơn. Ý nghĩa các kết nối logic được cho bởi bảng chân lý, trong đó liệt kê giá trị của câu phức cho tất cả tổ hợp giá trị các thành phần của câu. Bảng chân lý cho năm kết nối logic được cho trong bảng sau:

Bảng 3.1. Bảng chân lý của các kết nối logic

A	B	¬ A	A∧B	A v B	A=>B	A<=>B
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Sử dụng bảng chân lý, ta có thể tính được giá trị bất cứ câu phức nào bằng cách thực hiện đệ quy những kết nối thành phần.

Các công thức tương đương

Các phép biến đổi tương đương giúp đưa các công thức về dạng thuận lợi cho việc lập luận và suy diễn. Hai công thức A và B được xem là tương đương nếu chúng có cùng một giá trị chân lý trong mọi minh họa.

Ký hiệu: Để chỉ A tương đương với B ta viết A ≡ B.

Bằng phương pháp bảng chân lý, dễ dàng chứng minh được sự tương đương của các công thức sau đây :

- A=>B ≡ ¬ A ∨ B
- A<=> B ≡ (A=>B) ∧ (B=>A)
- ¬ (¬ A) ≡ A

Luật De Morgan

- ¬ (A ∨ B) ≡¬ A ∧ ¬ B
- ¬ (A ∧ B) ≡ ¬ A ∨¬ B

Luật giao hoán

- A v B ≡ B v A

- $A \wedge B \equiv B \wedge A$

Luật kết hợp

- $(A \vee B) \vee C \equiv A \vee (B \vee C)$
- $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$

Luật phân phối

- $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
- $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

3.3. SUY DIỄN VỚI LOGIC MỆNH ĐỀ

3.3.1. Suy diễn logic

Một công thức H được xem là hệ quả logic của một tập công thức $G = \{G_1,.....,G_m\}$ nếu trong bất kỳ minh họa nào mà $\{G_1,.....,G_m\}$ đúng thì H cũng đúng.

Khi có một cơ sở tri thức dưới dạng tập hợp các câu logic, ta muốn sử dụng các tri thức trong cơ sở này để suy ra tri thức mới mà nó là hệ quả logic của các công thức trong cơ sở tri thức. Điều đó được thực hiện bằng các thực hiện suy diễn. Suy diễn hay suy lý thường dùng chỉ quá trình cho phép rút ra kết luận. Để thực hiện suy diễn ta sử dụng luật suy diễn. Một luật suy diễn gồm hai phần : một tập các điều kiện và một kết luận.

Định nghĩa.

Thủ tục suy diễn được gọi là đúng đắn (sound) nếu kết quả suy diễn là hệ quả logic của điều kiện.

Thủ tục suy diễn được gọi là đầy đủ (complete) nếu cho phép tìm ra mọi hệ quả logic của điều kiện.

Ta sẽ sử dụng những kí hiệu sau:

KB: kí hiệu tập các câu đã có hay cơ sở tri thức (Knowledge Base)

$KB \models \alpha$: Khi các câu trong KB là đúng (True) thì α là đúng (True), hay α là hệ quả logic của KB.

3.3.2. Suy diễn sử dụng bảng chân lý

Bằng cách sử dụng bảng chân lý ta có thể xác định được một công thức có phải là hệ quả logic của các công thức trong cơ sở tri thức hay không.

Ví dụ: cho KB: $A \vee C$, $B \vee \neg C$

và $\alpha = A \vee B$

Để kiểm tra α có phải hệ quả logic của KB không, ta xây dựng bảng sau (bảng 3.2):

Kết quả xây dựng bảng cho thấy, α là hệ quả logic của KB, hay nói cách khác từ KB suy ra được α.

Bảng 3.2. Bảng chân lý

A	B	C	A ∨ C	B ∨¬ C	(A ∨ C) ∧ (B ∨¬ C)	A ∨ B	KB ⊨ α
T	T	T	T	T	T	T	√
T	T	F	T	T	T	T	√
T	F	T	T	F	F	T	
T	F	F	T	T	T	T	√
F	T	T	T	T	T	T	√
F	T	F	F	T	F	T	
F	F	T	T	F	F	F	
F	F	F	F	T	F	F	

Suy diễn với logic mệnh đề sử dụng bảng chân lý là thủ tục suy diễn đầy đủ và đúng đắn. Tính đúng đắn là hiển nhiên do bảng chân lý sử dụng đúng ngữ nghĩa được quy định với kết nối logic. Tính đầy đủ là do số lượng các tổ hợp giá trị đối với logic mệnh đề là hữu hạn và do vậy có thể liệt kê đầy đủ trường hợp KB có giá trị đúng.

Tuy nhiên, cần lưu ý rằng, một công thức chứa n biến mệnh đề, thì số các minh họa của nó là 2^n , tức là bảng chân lý có 2^n dòng. Như vậy việc kiểm tra một công thức có phải là một hệ quả lôgic hay không bằng phương pháp bảng chân lý có độ phức tạp tính toán lớn do đòi hỏi thời gian theo hàm mũ. Cook (1971) đã chứng minh rằng, phương pháp chứng minh thuật suy diễn là vấn đề NP-đầy đủ.

3.3.3. Sử dụng các quy tắc suy diễn

Do việc suy diễn sử dụng bảng như trên có độ phức tạp lớn nên cần có những thuật toán suy diễn hiệu quả hơn cho logic mệnh đề. Các thủ tục suy diễn đều dựa trên một số khái niệm như công thức tương được và các quy tắc suy diễn. Sau đây là một số luật suy diễn quan trọng trong logic mệnh đề.

Trong các luật này α, α_i , β, γ là các câu. Phần tiền đề hay phần điều kiện được viết dưới dạng tử số, phần hệ quả được viết dưới dạng mẫu số.

1. Luật Modus Ponens

$$\frac{\alpha => \beta, \alpha}{\beta}$$

Từ một kéo theo và giả thiết của kéo theo, ta suy ra kết luận của nó.

2. Luật Modus Tollens

$$\frac{\alpha => \beta, \neg \beta}{\neg \alpha}$$

Từ một kéo theo và phủ định kết luận của nó, ta suy ra phủ định giả thiết của kéo theo.

3.Luật loại trừ và

$$\frac{\alpha_1 \wedge ..... \wedge \alpha_i \wedge ..... \wedge \alpha_m}{\alpha_i}$$

Từ một công thức và ta đưa ra một nhân tử bất kỳ của công thức đó .

4. Luật nhập đề và

$$\frac{\alpha_1, ..., \alpha_i, ... \alpha_m}{\alpha_1 \wedge ..... \wedge \alpha_i \wedge ..... \wedge \alpha_m}$$

Từ một danh sách các công thức, ta suy ra phép và của chúng.

5. Luật nhập đề hoặc

$$\frac{\alpha_i}{\alpha_1 v ..... v \alpha_i . v ..... v \alpha_m}$$

Từ một công thức, ta suy ra một phép hoặc mà một trong các hạng tử của phép hoặc là công thức đó.

6. Luật loại trừ phủ định kép

$$\frac{\neg (\neg \alpha)}{\alpha}$$

Phép phủ định của phủ định một công thức, ta suy ra chính công thức đó.

7. Luật bắc cầu

$$\frac{\alpha => \beta, \beta => \gamma}{\alpha => \gamma}$$

Từ hai kéo theo, mà kết luận của nó là của kéo theo thứ nhất trùng với giả thiết của kéo theo thứ hai, ta suy ra kéo theo mới mà giả thiết của nó là giả thiết của kéo theo thứ nhất, còn kết luận của nó là kết luận của kéo theo thứ hai.

8. Phép giải đơn vị

$$\frac{\alpha \, v \, \beta, \neg \beta}{}$$

$$\alpha$$

Từ một phép hoặc, một hạng tử đối lập với một hạng tử trong tuyển kia, ta suy ra hạng tử còn lại.

9. Phép giải

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

Từ hai phép hoặc, một phép hoặc chứa một hạng tử đối lập với một hạng tử trong phép hoặc kia, ta suy ra phép hoặc của các hạng tử còn lại.

Một luật suy diễn được xem là tin cậy nếu bất kỳ một mô hình nào của giả thiết của luật cũng là mô hình kết luận của luật. Chúng ta chỉ quan tâm đến các luật suy diễn tin cậy.

Với các quy tắc suy diễn vừa trình bày, việc suy diễn trên logic mệnh đề được thực hiện nhờ một số thủ tục nhất định, trong đó thông dụng nhất là suy diễn bằng phép giải (resolution) và phản chứng (refutation). Tuy nhiên trong phần này chúng ta không đi sâu vào các thủ tục này mà sẽ xem xét các thủ tục chứng minh trong phần trình bày về logic vị từ.

Bài tập

1. KB = (A v B) $\wedge$ ($\neg$ C v $\neg$ D v E)

Các câu nào sau đây sinh ra từ KB?

1- A v B

2- (A v B v C) $\wedge$ ((B $\wedge$ C $\wedge$ D) => E)

1. Cho KB:

Red	Blue => Silver
$\neg$ Pink v Blue	Pink => Tan
Tan $\vee$ Orange	Silver
$\neg$ Pink	$\neg$ (Violet $\vee$ White)
Pink => Red	Blue => Orange

Suy ra:

a- $\neg$ Orange

b- Silver $\wedge$ Red

c- Silver $\vee$ White

3.4. LOGIC VỊ TỪ (LOGIC BẬC 1)

Trong phần trước ta đã xem xét logic mệnh đề và cách sử dụng logic mệnh để biểu diễn tri thức. Bên cạnh ưu điểm là đơn giản, logic mệnh đề có một nhược điểm lớn là khả năng biểu diễn hạn chế, không thể sử dụng để biểu diễn tri thức một cách ngắn gọn cho những bài toán có độ phức tạp lớn. Cụ thể là logic mệnh để thuật lợi cho biểu diễn sự kiện, sự kiện đơn giản được biểu diễn bằng câu nguyên tử, sự kiện phức tạp được biểu diễn bằng cách sử dụng kết nối logic để kết hợp câu nguyên tử. Logic mệnh để không cho phép biểu diễn một cách ngắn gọn môi trường với nhiều đối tượng. Chẳng hạn để thể hiện nhận xét "sinh viên trong lớp nào đó chăm học" ta phải sử dụng các câu riêng rẽ để thể hiện từng sinh viên cụ thể trong lớp chăm học.

Trong phần này ta sẽ xem xét logic vị từ - một hệ thống logic có khả năng thể biểu diễn mạnh hơn, đồng thời xem xét chi tiết thủ tục suy diễn với logic vị từ.

3.4.1. Đặc điểm

Đặc điểm quan trọng nhất của logic vị từ cho phép biểu diễn thế giới xung quanh dưới dạng các đối tượng, tính chất đối tượng, và quan hệ giữa các đối tượng đó. Việc sử dụng đối tượng là rất tự nhiên trong thế giới thực và trong ngôn ngữ tự nhiên, với danh từ biểu diễn đối tượng, tính từ biểu diễn tính chất và động từ biểu diễn quan hệ giữa các đối tượng. Có thể kể ra rất nhiều ví dụ về đối tượng, tính chất và quan hệ:

- Đối tượng : một cái bàn, một cái nhà, một cái cây, một con người, một sinh viên, một con số. ...
- Tính chất : Cái bàn có thể có tính chất : có bốn chân, làm bằng gỗ, không có ngăn kéo, sinh viên có thể có tính chất là thông minh, cao, gầy…
- Quan hệ : cha con, anh em, bè bạn (giữa con người); lớn hơn nhỏ hơn, bằng nhau (giữa các con số) ; bên trong, bên ngoài nằm trên nằm dưới (giữa các đồ vật)...
- Hàm : Một trường hợp riêng của quan hệ là quan hệ hàm, trong đó với mỗi đầu vào ta có một giá trị hàm duy nhất.. Ví dụ: tay trái của ai đó, bố của ai đó, bội số chung nhỏ nhất của hai số.

Logic vị từ có cú pháp và ngữ nghĩa được xây dựng dựa trên khái niệm đối tượng. Hệ thống logic này đóng vai trò quan trọng trong việc biểu diễn tri thức do có khả năng biểu diễn phong phú và tự nhiên, đồng thời là cơ sở cho nhiều hệ thống logic khác.

3.4.2. Cú pháp và ngữ nghĩa

Trong phần này ta sẽ xem xét cú pháp, tức là quy tắc tạo ra những câu hay biểu thức logic, của logic vị từ cùng với ngữ nghĩa của những cấu trúc đó.

Các ký hiệu và ý nghĩa

Logic vị từ sử dụng những dạng ký hiệu sau.

- Các ký hiệu hằng: Nam, 3, Vịnh Hạ long,…
- Các ký hiệu biến: x, y, z,...

- Các ký hiệu vị từ: Thích (Nam, Bắc), Làm_từ_gỗ (tủ), Anh_em (An, Ba, Út)

  Ký hiệu vị từ thể hiện quan hệ giữa các đối tượng. Mỗi vị từ có thể có n tham số ($n \geq 0$). Ví dụ Thích là vị từ của hai tham số, Làm_từ_gỗ là vị từ một tham số. Các ký hiệu vị từ không tham số là các ký hiệu mệnh đề.

- Các ký hiệu hàm: Mẹ_của(An), min(3,4,9),...

  Ký hiệu hàm thể hiện quan hệ hàm. Mỗi hàm có thể có n tham số ($n \geq 1$).

- Các ký hiệu kết nối logic: $\wedge$ (hội), $\vee$ (tuyển), $\neg$ (phủ định), $\Rightarrow$(kéo theo), $\Leftrightarrow$ (tương đương).

- Các ký hiệu lượng tử: $\forall$ (với mọi), $\exists$ (tồn tại).

- Các ký hiệu ngăn cách: dấu phẩy, dấu mở ngoặc và dấu đóng ngoặc.

Tương tự như với logic mệnh đề, ngữ nghĩa cho phép liên kết biểu thức logic với thế giới của bài toán để xác định tính đúng hoặc sai của biểu thức. Một liên kết cụ thể như vậy được gọi là một minh họa. Minh họa xác định cụ thể đối tượng, quan hệ và hàm mà các ký hiệu hằng, vị từ, và ký hiệu hàm thể hiện.

Để xác định một minh hoạ, trước hết ta cần xác định một miền đối tượng (nó bao gồm tất cả các đối tượng trong thế giới hiện thực mà ta quan tâm). Cũng có thể xác định miền đối tượng cho từng tham số của một vị từ hoặc một hàm nào đó. Ví dụ trong vị từ Thích(x,y), miền của x là tất cả mọi người, miền của y là các loại động vật. Số đối tượng có thể là vô hạn, chẳng hạn trong trường hợp miền đối tượng là toàn bộ số thực.

Việc lựa chọn tên cho hằng, biến, vị từ, và hàm hoàn toàn do người dùng quyết định. Có thể có nhiều minh họa khác nhau cho cùng một thế giới thực. Việc suy diễn, tính đúng đắn của biểu thức được xác định dựa trên toàn bộ minh họa.

Hạng thức (term)

Hạng thức (term) là biểu thức logic có kết quả là đối tượng. Hạng thức được xác định đệ quy như sau.

- Các ký hiệu hằng và các ký hiệu biến là hạng thức.
- Nếu $t_1, t_2, t_3, ..., t_n$ là n hạng thức và f là một ký hiệu hàm n tham số thì $f(t_1, t_2, ..., t_n)$ là hạng thức. Một hạng thức không chứa biến được gọi là một hạng thức cụ thể (ground term).

Chẳng hạn, An là ký hiệu hằng, Mẹ_của là ký hiệu hàm, thì Mẹ_của(An) là một hạng thức cụ thể.

Ngữ nghĩa của hạng thức như sau: các hằng, biến, tham số tương ứng với đối tượng trong miền đối tượng; ký hiệu hàm tương ứng với quan hệ hàm trong thế giới thực; hạng thức tương ứng với đối tượng là giá trị của hàm khi nhận tham số.

Ký hiệu "="

Hai hạng thức bằng nhau và được ký hiệu "=" nếu cùng tương ứng với một đối tượng.

Ví dụ: Mẹ_của(Vua_Tự_Đức) = Bà_Từ_Dũ

Tính đúng đắn của quan hệ bằng được xác định bằng cách kiểm tra hai vế của ký tự "=".

Câu nguyên tử

Các câu nguyên tử, còn gọi là câu đơn, được xác định như sau:

- Vị từ có tham số là hạng thức là câu nguyên tử.
- Hạng thức 1 = hạng thức 2 là câu nguyên tử.

Ví dụ : Yêu (Hoa, Mẹ_của(Hoa))

Mẹ_của(Vua_Tự_Đức) = Bà_Từ_Dũ

Câu nguyên tử nhận giá trị đúng (true) trong một minh họa nào đó nếu quan hệ được biểu diễn bởi ký hiệu vị từ là đúng đối với các đối tượng được biểu diễn bởi các hạng thức đóng vai trò thông số. Như vậy, câu nguyên tử thể hiện những sự kiện (đơn giản) trong thế giới của bài toán.

Câu

Từ các câu nguyên tử, sử dụng các kết nối logic và các lượng tử, ta xây dựng nên các câu. Câu được định nghĩa đệ quy như sau:

- Câu nguyên tử là câu.
- Nếu G và H là các câu nguyên tử, thì các biểu thức $(G \wedge H)$, $(G \vee H)$, $(\neg G)$, $(G \Rightarrow H)$, $(G \Leftrightarrow H)$ là câu
- Nếu G là một câu nguyên tử và x là biến thì các biểu thức $(\forall x\ G)$, $(\exists x\ G)$ là câu

Các câu không phải là câu nguyên tử sẽ được gọi là các câu phức hợp. Các câu không chứa biến được gọi là câu cụ thể. Khi viết các công thức ta sẽ bỏ đi các dấu ngoặc không cần thiết, chẳng hạn các dấu ngoặc ngoài cùng.

Ngữ nghĩa của câu phức hợp được xác định bằng ngữ nghĩa các câu đơn và các phép nối tương tự trong logic mệnh đề.

Các lượng tử

Logic mệnh đề sử dụng hai lượng tử: với mọi và tồn tại.

Lượng tử với mọi ($\forall$) cho phép mô tả tính chất của cả một lớp các đối tượng, chứ không phải của một đối tượng, mà không cần phải liệt kê ra tất cả các đối tượng trong lớp. Ví dụ ta sử dụng vị từ Elephant(x) (đối tượng x là con voi) và vị từ Color(x, Gray) (đối tượng x có màu xám) thì câu " tất cả các con voi đều có màu xám" có thể biểu diễn bởi công thức $\forall x$ (Elephant(x) $\Rightarrow$ Color(x, Gray)).

Như vậy câu $\forall$ x P có nghĩa là câu P đúng với mọi đối tượng x thuộc miền giá trị đã được quy định của thế giới bài toán. Lượng tự với mọi có thể coi như hội của nhiều câu.

Lưu ý: lượng tử với mọi được dùng với kéo theo chứ không dùng với "và". Chẳng hạn, để nói rằng mọi sinh viên đều chăm học thì câu

$\forall x$ Sinh_viên(x) $\Rightarrow$ Chăm_học(x) là đúng

trong khi

$\forall x$ Sinh_viên(x) $\wedge$ Chăm_học(x) là sai do câu này sẽ có ý nghĩa tất cả mọi người đều là sinh viên và đều chăm học.

Lượng tử tồn tại ($\exists$) cho phép ta tạo ra các câu nói đến một đối tượng nào đó trong một lớp đối tượng mà nó có một tính chất hoặc thoả mãn một quan hệ nào đó. Ví dụ ta sử dụng các câu nguyên tử Student(x) (x là sinh viên) và Inside(x, P308), (x ở trong phòng 308), ta có thể biểu diễn câu " Có một sinh viên ở phòng 308" bởi biểu thức $\exists x$ (Student(x) $\wedge$ Inside(x,P301).

Ngữ nghĩa của công thức $\exists x$ P được xác định như là ngữ nghĩa của công thức là tuyển của tất cả các công thức nhận được từ P bằng cách thay x bởi một đối tượng trong miền đối tượng.

Lưu ý: Lượng tử tồn tại được dùng với "và" chứ không dùng với "kéo theo". Chẳng hạn để nói rằng có một số sinh viên chăm học thì câu:

$\exists x$ Sinh_viên(x) $\wedge$ Chăm_học(x) là đúng

trong khi

$\exists x$ Sinh_viên(x) $\Rightarrow$ Chăm_học(x)

là sai. Thật vậy, do phép kéo theo đúng khi tiền đề là sai nên câu trên đúng khi có một người x nào đó không phải là sinh viên, trong khi đây không phải là ý mà ta muốn khẳng định.

Quan hệ giữa lượng tử với mọi và lượng tử tồn tại: lượng tử này có thể biểu diễn bằng lượng tử kia bằng cách sử dụng phép phủ định. Ví dụ:

$\forall x$ Thích (x, Kem) tương đương với $\neg\exists x$ $\neg$Thích(x, Kem)

$\exists y$ Thích (x, Kem) tương đương với $\neg\forall x$ $\neg$Thích (x, Kem)

Các lượng tử lồng nhau

Có thể sử dụng đồng thời nhiều lượng tử trong một câu phức tạp. Vùng ảnh hưởng của lượng tử có thể bao hàm lượng tử khác và khi đó ta nói lượng tử lồng nhau. Ví dụ:

$\forall x \forall y$ Anh_em(x, y) $\Rightarrow$ Họ_hàng(x, y)

$\forall x \exists y$ Yêu (x, y)

Nhiều lượng tử cùng loại có thể được viết gọn bằng một ký hiệu lượng tử, ví dụ câu thứ nhất có thể viết gọn thành

$\forall x, y$ Anh_em(x, y) $\Rightarrow$ Họ_hàng(x, y)

Trong trường hợp lượng tử với mọi được sử dụng cùng lượng tử tồn tại thì thứ tự lượng tử ảnh hưởng tới ngữ nghĩa của câu và không được phép thay đổi. Chẳng hạn câu

$\forall x \exists y$ Yêu (x, y)

có nghĩa là mọi người đều có ai đấy để yêu, trong khi câu

$\exists y \forall x$ Yêu (x, y)

có nghĩa là có ai đó mà tất cả đều yêu.

Trong trường hợp nhiều lượng tử khác nhau cùng sử dụng một tên biến thì có thể gây nhầm lẫn vì vậy cần sử dụng tên biến khác nhau cho ký hiệu lượng tử khác nhau.

Một câu là câu nguyên tử hoặc là phủ định của câu nguyên tử được gọi là literal. Chẳng hạn, Play(x, Football), $\neg$ Like(Lan, Rose) là các literal. Một công thức là tuyển của các literal sẽ được gọi là câu tuyển. Chẳng hạn, Male(x) $\vee \neg$ Like(x, Foodball) là câu tuyển.

Các công thức tương đương

Cũng như trong logic mệnh đề, ta nói hai công thức G và H tương đương (viết là $G \equiv H$) nếu chúng cùng đúng hoặc cùng sai trong một minh hoạ. Ngoài các tương đương đã biết trong logic mệnh đề, trong logic vị từ cấp một còn có các tương đương khác liên quan tới các lượng tử.

Sau đây là các tương đương của logic vị từ

$\forall x\ G(x) \equiv \forall y\ G(y)$

$\exists x\ G(x) \equiv \exists y\ G(y)$

Đặt tên lại biến đi sau lượng tử tồn tại, ta nhận được công thức tương đương .

$\neg (\forall x\ G(x)) \equiv \exists x\ (\neg\ G(x))$

$\neg\ (\exists x\ G(x)) \equiv \forall x\ (\neg\ G(x))$

3. $\forall x\ (G(x) \wedge H(x)) \equiv \forall x\ G(x) \wedge \forall x\ H(x)$

$\exists x\ (G(x) \vee H(x)) \equiv \exists x\ G(x) \vee \exists x\ H(x)$

Ví dụ : $\forall x$ Love(x, mother(x)) $\equiv \forall y$ Love(y, mother(y)).

<u>Bài tập</u>

1. Viết các câu sau dưới dạng logic vị từ:

1 - Mọi nhà nông thích mặt trời

2 - Lúc nào cũng có người bị lừa

3 - Nấm có màu đỏ là nấm độc

4 - An không cao

5 - Chỉ có 2 sinh viên con nhà giàu học lớp D07

6 - Trên trời có muôn vàn vì sao

3.5. SUY DIỄN VỚI LOGIC VỊ TỪ

3.5.1. Quy tắc suy diễn

Mọi quy tắc suy diễn cho logic mệnh đề cũng đúng với logic vị từ. Ngoài ra, logic vị từ còn có thêm một số quy tắc suy diễn khác, chủ yếu được dùng với câu có chứa lượng tử, cho phép biến đổi những câu này thành câu không có lượng tử.

Phép thế (substitution)

Trước khi đi xem xét quy tắc suy diên, ta định nghĩa khái niệm phép thế, cần thiết cho những câu có chứa biến.

Ký hiệu là SUBST(θ, α)

Phép thế giá trị θ vào câu α

Ví dụ: SUBST ({x/Nam, y/An} Thích(x,y)) = Thích(Nam, An)

Phép loại trừ với mọi (universal elimination)

$$\frac{\forall x \ \alpha}{\text{SUBST}(\{x/g\}, \alpha)}$$

Ví dụ: $\forall x$ Thích(x, Kem) $\xrightarrow{\{x/Nam\}}$ Thích (Nam, Kem)

Loại trừ tồn tại (existential elimination)

$$\frac{\exists x \ \alpha}{\text{SUBST}(\{x/k\}, \alpha)}$$

k là kí hiệu hằng chưa xuất hiện trong KB

Ví dụ: $\exists x$ Học_giỏi(x) $\xrightarrow{\{x/Nam\}}$ Học_giỏi (Nam)

k được gọi là hằng Skolem và ta có thể đặt tên cho hằng này. Yêu cầu với hằng Skolem là hàm này chưa được phép xuất hiện trong cơ sở tri thức trước đó.

Nhập đề tồn tại (existential introduction)

Với câu α, biến x không thuộc câu α và hạng thức cơ sở g thuộc câu α

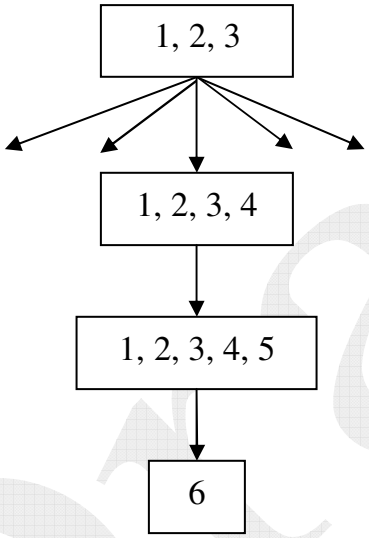
Ta có:

$$\frac{\alpha}{\exists x \ \text{SUBST}(\{g/x\}, \alpha)}$$

Thích(Nam, Kem) $\xrightarrow{\{Nam/x\}}$ $\exists x$ Thích (x, Kem)

Ví dụ suy diễn:

Bob là trâu	trâu (Bob) (1)
Pat là lợn	lợn (Pat) (2)
Trâu to hơn lợn	$\forall x, y$ trâu (x) $\wedge$ lợn (y) => to_hơn (x,y) (3)
Bob to hơn Pat ?	to_hơn (Bob, Pat) ?
Nhập đề và, (1) (2)	trâu (Bob) $\wedge$ lợn(Pat) (4)
Loại trừ với mọi, (3)	trâu(Bob) $\wedge$ lợn (Pat) => to_hơn (Bob,Pat) (5)
Modus Ponens, (4) (5)	to_hơn (Bob, Pat)



Suy diễn tự động trên logic vị từ khó hơn rất nhiều so với logic mệnh đề do các biến có thể nhận vô số các giá trị khác nhau.

Ta cũng không thể sử dụng bảng chân lý do kích thước của bảng có thể là vô hạn.

Phép hợp nhất (Unification)

Hợp nhất là thủ tục xác định phép thế cần thiết để làm cho 2 câu giống nhau và được ký hiệu như sau:

UNIFY (p,q) = (θ)

SUBST(θ,p) = SUBST (θ,q)

θ được gọi là hợp tử (phần tử hợp nhất)

Ví dụ:

p	q	θ
Biết (Nam, x)	Biết (Nam, Bắc)	{x/Bắc}

Biết (Nam, x)	Biết (y, Mẹ (y))	{y/Nam, x/ Mẹ (Nam)}
Biết (Nam, x)	Biết (y, z)	{y/Nam, x/z} {y/Nam, x/Nam, z/Nam}

Trong trường hợp có nhiều hợp tử thì ta sử dụng hợp tử tổng quát nhất tức là hợp tử sử dụng ít phép thế cho biến nhất

MGU: most general unifier

Phép hợp nhất có thể thực hiện tự động bằng thuật toán có độ phức tạp tỉ lệ tuyến tính với số lượng biến

Modus Ponent tổng quát (GMP)

Giả sử ta có các câu cơ sở, p_i, p_i', q và tồn tại phép thế θ sao cho UNIFY $(p_i, p_i') = \theta$ với mọi i

Khi đó ta có:

$$\frac{p_1', p_2', p_3', \ldots p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n => q)}{SUBST(\theta, q)}$$

Thủ tục suy diễn với GMP là không đầy đủ với logic vị từ nói chung.

Suy diễn bằng GMP là đầy đủ trong trường hợp KB chỉ chứa các câu horn (horn clause) – là các câu nguyên tử, hoặc là các phép kéo theo có vế trái là tuyển của các câu cơ sở , vế phải là câu nguyên tử

3.5.2. Suy diễn tiến và suy diễn lùi

Sử dụng quy tắc Modus Ponens tổng quát cho phép xây dựng thuật toán suy diễn tự động, cụ thể là phương pháp suy diễn tiến và suy diễn lùi.

Suy diễn tiến (forward chaining)

Thủ tục suy diễn tiến được thực hiện như sau:

- Khi câu p mới được thêm vào KB:
  - với mỗi quy tắc q mà p hợp nhất được với một phần vế trái:
    - Nếu các phần còn lại của vế trái đã có thì thêm vế phải vào KB và suy diễn tiếp

Hình 3.2. Thủ tục suy diễn tiến

Ví dụ: KB gồm

1. Mèo thích cá
2. Mèo ăn gì nó thích
3. Có con mèo tên là Tom

Tom có ăn cá không?

Giải:

1. $\forall x$ mèo(x) => thích(x, cá)
2. $\forall x, y$ mèo(x) $\wedge$ thích (x,y) => ăn(x,y)
3. mèo (Tom)

Q: ăn (Tom, cá) ?

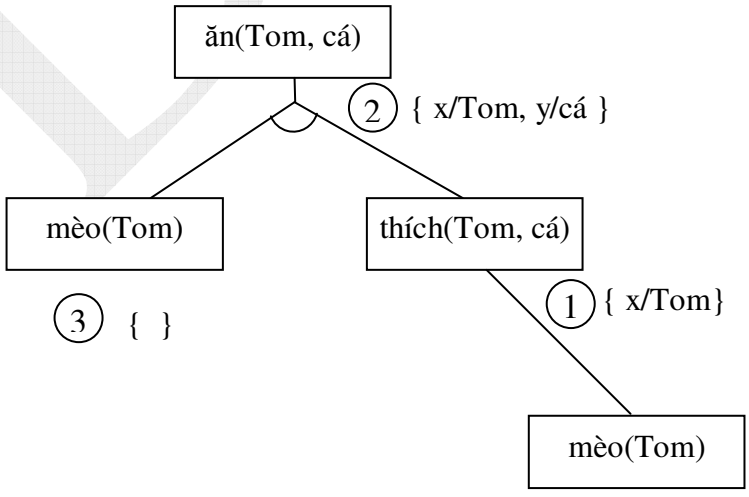
4. GMP (1) (3) => thích (Tom, cá)
5. GMP (4) (3) (2) => ăn(Tom, cá)

Suy diễn lùi (Backward chaining)

- Với câu hỏi q, nếu tồn tại q' hợp nhất với q thì trả về hợp tử
- Với mỗi quy tắc có vế phải q' hợp nhất với q cố gắng chứng minh các phần tử vế trái bằng suy diễn lùi

Hình 3.3. Thủ tục suy diễn lùi

Chứng minh ví dụ trước bằng suy diễn lùi:



Hình 3.4. Cây suy diễn cho trường hợp suy diễn lùi

Bài tập

1. Ta có:

1 - trâu (x) ∧ lợn (y) => to_hơn(x, y)

2 - lợn (y) ∧ chuột (z) => to_hơn(y, z)

3 - to_hơn(x, y) ∧ to_hơn (y, z) => to_hơn(x, z)

4 - trâu (Bob)

5 - lợn (Pat)

6 - chuột (Jerry)

Thực hiện suy diễn tiến ra các câu có thể

2. Ta có các câu sau:

1- Pig (y) ∧ Slug (z) => Faster(y, z)

2- Slim (z) ∧ Creeps (z) => Slug(z)

3- Pig (Pat)

4- Slim (Steve)

5- Creeps (Steve)

Faster(Pat, Steve)?

3. Tìm MGU cho các cặp câu sau:

a. P (A, B, B) , P (x, y, z)

b. Q (y, G(A, B)) , Q (G (x, x), y)

c. Older (Father (y), y) , Older(Father (x), John)

d. Knows (Father(y), y), Knows (x, x)

4. Biểu diễn các câu sau dưới dạng logic vị từ phù hợp với việc sử dụng Modus Ponens tổng quát

1. Ngựa, Bò, Lợn là động vật
2. Con của ngựa là ngựa
3. Ngựa tên là Xích Thố
4. Xích Thố là bố của con Chiếu Dạ
5. Quan hệ cha con là quan hệ nghịch đảo
6. Tất cả động vật đều có bố

5. Sử dụng suy diễn lùi chứng minh con Chiếu Dạ là con ngựa

6. Cho biết:

Tôi không có anh chị em

Bố của người đó là con trai tôi của bố tôi

Hỏi: Người đó là ai?

3.5.3. Suy diễn sử dụng phép giải

Phép giải cho logic vị từ

Ta có phép giải cho logic mệnh đề

$$\alpha \vee \beta \vee \lambda \ , \neg \lambda \vee \gamma \vee \mu \ \Rightarrow \alpha \vee \beta \vee \gamma \vee \mu$$

Đối với logic vị từ, phép giải như sau:

Cho câu: $P_1 \vee P_2 \vee ... \vee P_n$

Và câu: $Q_1 \vee Q_2 \vee ... \vee Q_m$

Trong đó P_i , Q_i là literal

(Literal là các vị từ hoặc các vị từ có dấu phủ định đằng trước)

Nếu P_i và $\neg Q_k$ có thể hợp nhất bởi hợp tử θ thì ta có phép giải:

$$\frac{P_1 \vee P_2 \vee ... \vee P_n \ , Q_1 \vee Q_2 \vee ... \vee Q_m}{SUBST(\theta, P_1 \vee ... \vee P_{j-1} \vee P_{j+1} \vee ... \vee P_n \vee Q_1 \vee ... \vee Q_{k-1} \vee Q_{k+1} \vee ... \vee Q_n)}$$

Ví dụ:

1. $$\frac{\text{Giàu (x)} \vee \text{Giỏi (x)}, \neg \text{Giỏi (Bắc)} \vee \text{Đẹp trai (Bắc)}}{\text{Giàu (Bắc)} \vee \text{Đẹp trai (Bắc)}}$$

2. Cho các câu $P(x, f(a)) \vee P(x, f(y)) \vee Q(y)$

   và $\neg P(z, f(a)) \vee \neg Q(z)$

   Suy ra

$P(z, f(y)) \vee Q(y) \vee \neg Q(z) \quad \theta \ \{x/z\}$

$P(x, f(a)) \vee P(x, f(z)) \vee \neg P(z, f(a)) \quad \theta \ \{y/z\}$

Dạng Conjunctive Normal Form (CNF) và Clause Form

Các công thức tương đương có thể xem như các biểu diễn khác nhau của cùng một sự kiện. Để dễ dàng viết các chương trình máy tính thao tác trên các công thức, chúng ta sẽ chuẩn hóa các công thức, đưa chúng về dạng biểu diễn chuẩn.

Người ta định nghĩa mỗi clause là tuyển của literal, có dạng $A_1 \vee A_2 \vee ... . \vee A_m$ trong đó các A_i là literal.

Một dạng chuẩn được gọi là Conjunctive Normal Form (CNF - dạng chuẩn hội), là câu bao gồm hội của phép tuyển của các literal hoặc là hội của clause.

Chúng ta có thể biến đổi một công thức bất kỳ về công thức ở dạng CNF bằng cách áp dụng một số bước thủ tục nhất định sẽ được trình bày ở phần sau.

Suy diễn sử dụng phép giải và phản chứng (Resolution Refutation)

Nếu KB là tập hữu hạn các câu thì các literal có mặt trong các câu của KB cũng là hữu hạn. Do đó số các clause thành lập được từ các literal đó là hữu hạn. Vì vậy chỉ có một số hữu hạn câu được sinh ra bằng luật giải. Phép giải sẽ dừng lại sau một số hữu hạn bước. Sử dụng phép giải ta có thể chứng minh được một câu có là tập con của một KB đã cho hay không bằng phương pháp chứng minh phản chứng.

KB: $\vdash$ Q?

Thêm $\neg$ Q vào KB, sau đó chứng minh tồn tại một tập con của KB mới có giá trị False

(KB $\vdash$ Q) <=> (KB $\wedge \neg$ Q $\vdash$ False)

Nói cách khác: phương pháp chứng minh tạo cơ sở tri thức mới bao gồm KB và $\neg$ Q, sau đó dùng phép giải để chứng minh từ cơ sở tri thức mới suy ra False.

Thuật toán:

KB = UNION (KB, $\neg$ Q)

While (KB không chứa False) do

Chọn 2 câu S_1, S_2 từ KB sao cho có thể áp dụng phép giải cho 2 câu này

Nếu không có hai câu như vậy

Return False

Thêm kết quả phép giải vào KB

Return Success

Hình 3.5. Suy diễn bằng phép giải và phản chứng

Về tính đầy đủ của suy diễn sử dụng phép giải

Suy diễn sử dụng phép giải là phản chứng – đầy đủ, tức là nếu một tập hợp các câu là không thỏa được trong một minh họa nào đó thì thủ tục suy diễn sử dụng phép giải luôn cho phép tìm ra mâu thuẫn. Như vậy, phép giải cho phép xác định một câu có quả là hệ quả logic của một tập các câu khác không. Tuy nhiên, phép giải không cho phép sinh ra tất cả các câu là hệ quả logic của một tập câu cho trước.

Ví dụ:

KB: $\neg$ A $\vee \neg$ B $\vee$ P (1)

$\neg$ C $\vee \neg$ D $\vee$ P (2)

$\neg$ E $\vee$ C (3)

A (4)

E (5)

D (6)

Ta cần chứng minh KB: ├ P.

Các bước chứng minh sẽ như sau:

Thêm vào KB câu sau:

¬ P (7)

Áp dụng phép giải cho câu (2) và (7) ta được câu:

¬ C v ¬ D (8)

Từ câu (6) và (8) ta nhận được câu:

¬ C (9)

Từ câu (3) và (9) ta nhận được câu:

¬ E (10)

Câu (10) mang giá trị False. Tới đây ta đã tìm thấy một tập con của KB mới có giá trị False .

Kết luận : Từ KB suy ra P

Biến đổi các câu về dạng CNF và Clause Form

Khi biểu diễn tri thức bởi các câu trong logic vị từ, KB là một tập các câu. Để sử dụng thuật toán trên thì ta thực hiện chuẩn hóa các câu trong KB để chuyển về dạng Clause form bằng cách sử dụng các bước sau:

Bước 1: Khử tương đương

Để khử phép tương đương thay P⇔Q bằng (P=>Q)∧(Q=> P).

Bước 2: Loại bỏ kéo theo

Để loại bỏ các kéo theo, ta chỉ cần thay thế công thức P⇒Q bởi công thức tương đương ¬ P ∨ Q

Bước 3: Đưa các phủ định vào gần các vị từ

Chuyển các dấu phủ định (¬) vào sát các vị từ bằng cách áp dụng luật De Morgan và thay ¬ (¬ A) bởi A . Điều này được thực hiện bằng cách thay công thức ở vế trái bằng công thức ở vế phải trong các tương đương sau:

¬ (¬ P) ≡ P

¬ (P∧Q) ≡ ¬ P∨ ¬ Q

¬ (P ∨ Q) ≡ ¬ P∧¬ Q

¬ (∀x P) ≡ ∃ x (¬ P)

¬ (∃ x P) ≡ ∀ x (¬ P)

Bước 4: Chuẩn hóa tên biến sao cho mỗi lượng tử có biến riêng

Ví dụ :

$\forall x \neg P(x) \vee Q(x)$ $\Longrightarrow$ $\forall x \neg P(x) \vee Q(x)$

$\forall x \neg R(x) \vee Q(x)$ $\forall y \neg R(y) \vee Q(y)$

Bước 5: Loại bỏ các lượng tử tồn tại bằng cách sử dụng hằng Skolem và hàm Skolem

Giả sử P(x,y) là các vị từ có nghĩa rằng "y lớn hơn x" trong miền các số. Khi đó câu $\forall x$ ($\exists y$ (P(x,y)) có nghĩa là "với mọi số x tồn tại y sao cho y lớn hơn". Ta có thể xem y trong câu đó là hàm của đối số x, chẳng hạn f(x) và loại bỏ lượng tử $\exists y$, câu đang xét trở thành $\forall x$ (P(x,f(x)).

Một cách tổng quát, giả sử $\exists y$ (G) là một câu con của câu đang xét và nằm trong miền tác dụng của lượng tử $\forall x_1,.....,\forall x_n$. Khi đó ta có thể xem y là hàm của n biến $x_1,....., x_n$, chẳng hạn $f(x_1,....., x_n)$. Sau đó ta thay các xuất hiện của y trong câu G bởi hạng thức $f(x_1,....., x_n)$ và loại bỏ các lượng tử tồn tại. Các hàm f được đưa vào để loại bỏ các lượng tử tồn tại được gọi là hàm Skolem.

Ví dụ: Xét câu sau:

$\forall x (\exists y (P(x,y) \vee \forall u (\exists b (Q(a,b) \wedge \exists t \neg R(x,t)))$ (1)

Câu con $\exists y$ P(x,y) nằm trong miền tác dụng của lượng tử $\forall x$, ta xem y là hàm của x: F(x). Các câu con $\exists b$ (Q(a,b) và $\exists t \neg$R(x,y) nằm trong miền tác dụng của các lượng tử $\forall x$, $\forall u$ nên ta xem b là hàm g(x,u) và t là hàm h(x,u) của 2 biến x,u. Thay các xuất hiện của y và b, t bởi các hàm tương ứng, sau đó loại bỏ các lượng tử tồn tại, từ câu (1) ta nhận được câu :

$\forall x ((P(x,f(x)) \vee \forall u (Q(a,g(x,u)) \wedge \neg R(x,h(x,u))))$

Khi lượng tử tồn tại nằm trong lượng tử với mọi thì ta dùng hàm skolem, còn trong trường hợp khác ta sử dụng hằng skolem.

Bước 6: Loại bỏ các lượng từ với mọi ($\forall$)

Để loại bỏ lượng từ với mọi ($\forall$), ta đưa các lượng tử với mọi ($\forall$) sang trái sau đó bỏ lượng tử với mọi ($\forall$).

Ví dụ: $\forall x$ (P (x,y) $\vee$ Q (x)) => P (x,y) $\vee$ Q (x)

Bước 7: Sắp xếp các phép và, hoặc để có dạng CNF

.Ví dụ:

$(P \wedge Q) \vee R \equiv (P \vee R) \wedge (Q \vee R)$

$(P \vee Q) \vee R \equiv P \vee Q \vee R$

Bước 8: Loại bỏ các phép và

Ta thực hiện loại bỏ các phép và để tạo thành các clause riêng

Ví dụ :

$$(P \vee R \vee S) \wedge (Q \vee \neg R) \quad => \quad P \vee R \vee S$$
$$Q \vee \neg R$$

Bước 9 : Chuẩn hóa tên biến sao cho mỗi câu có biến riêng của mình

Ví dụ:

$$\neg P(x) \vee P(y) \vee Q(f(x,y)) \qquad \neg P(x) \vee P(y) \vee Q(f(x,y))$$
$$\neg P(x) \vee Q(x, g(x)) \quad \Longrightarrow \quad \neg P(z) \vee Q(z, g(z))$$
$$P(x) \vee \neg R(g(x)) \qquad P(u) \vee \neg R(g(u))$$

Ví dụ:

Chuẩn hóa công thức sau:

$$\forall x (P(x) => (\forall y (P(y) => P(f(x,y))) \wedge \neg \forall y (Q(x,y) => P(y))))$$

Ta sẽ lần lượt thực hiện theo từng bước cụ thể như sau:

1. Khử tương đương

2. Loại bỏ kéo theo

$$\forall x (\neg P(x) \vee (\forall y (\neg P(y) \vee P(f(x,y))) \wedge \neg \forall y (\neg Q(x,y) \vee P(y))))$$

3. Đưa phủ định vào gần các vị từ

$$\forall x (\neg P(x) \vee (\forall y (\neg P(y) \vee P(f(x,y))) \wedge \exists y \neg (\neg Q(x,y) \vee P(y))))$$

$$\forall x (\neg P(x) \vee (\forall y (\neg P(y) \vee P(f(x,y))) \wedge \exists y (Q(x,y) \wedge \neg P(y))))$$

4. Chuẩn hóa tên biến sao cho mỗi lượng tử có biến riêng

$$\forall x (\neg P(x) \vee (\forall y (\neg P(y) \vee P(f(x,y))) \wedge \exists z (Q(x,z) \wedge \neg P(z))))$$

5. Loại bỏ các lượng tử tồn tại bằng cách sử dụng hằng Skolem và hàm Skolem

$$\forall x (\neg P(x) \vee (\forall y (\neg P(y) \vee P(f(x,y))) \wedge (Q(x, g(x)) \wedge \neg P(g(x)))))$$

6. Loại bỏ lượng từ với mọi ($\forall$)

$$(\neg P(x) \vee ((\neg P(y) \vee P(f(x,y))) \wedge (Q(x, g(x)) \wedge \neg P(g(x)))))$$

7. Sắp xếp phép và và phép hoặc để có dạng CNF

$$[\neg P(x) \vee \neg P(y) \vee P(f(x,y))] \wedge [\neg P(x) \vee Q(x, g(x))] \wedge [\neg P(x) \vee \neg P(g(x))]$$

8. Bỏ phép và để tạo thành các clause riêng

$$\neg P(x) \vee \neg P(y) \vee P(f(x,y))$$

$$\neg P(x) \vee Q(x, g(x))$$

$$\neg P(x) \vee \neg P(g(x))$$

9.Chuẩn hóa tên biến sao cho mỗi câu có biến riêng của mình

$\neg P(x) \vee \neg P(y) \vee P(f(x,y))$

$\neg P(z) \vee Q(z, g(z))$

$\neg P(k) \vee \neg P(g(k))$

Bài tập

1. Có 3 người Bắc, Đông, Nam tham gia câu lạc bộ.

Mỗi thành viên câu lạc bộ là leo núi hoặc trượt tuyết

Không có người leo núi nào thích mưa

Tất cả những người trượt tuyết thích tuyết

Đông ghét tất cả những gì Nam thích

Đông thích tất cả những gì Nam ghét

Nam thích mưa và tuyết

Hỏi có thành viên nào của câu lạc bộ là người leo núi không phải là người trượt tuyết?

a. Dịch sang logic vị từ

b. Chuyển dạng clause form

c. Dùng phép giải và phản chứng để chứng minh.

2. Cho các câu sau:

1. $\forall x$ Kem (x) => Thức_ăn (x)
2. $\forall x$ Caramen(x) => Thức_ăn (x)
3. $\forall x, y$ Thức_ăn (x) $\wedge$ Thức_ăn(y) $\wedge$ Lạnh(x) $\wedge$ Trộn (x,y) => Lạnh (y)
4. $\exists x \exists y$ Kem (x) $\wedge$ Lạnh (x) $\wedge$ Caramen (y) $\wedge$ Trộn (x,y)

a. Dịch sang tiếng Việt

b. Dùng phản chứng và phép giải để chứng minh

$\exists x$ (Caramen (x) $\wedge$ Lạnh (x))

3.5.4. Hệ thống suy diễn tự động: lập trình logic

Trên thực tế, việc biểu diễn tri thức và suy diễn logic được thực hiện bằng cách sử dụng một số ngôn ngữ lập trình được thiết kế đặc biệt. Kỹ thuật xây dựng hệ thống suy diễn như vậy được gọi là lập trình logic (logic programming). Ngôn ngữ lập trình logic tiêu biểu là Prolog.

Chương trình trên Prolog là một tập hợp các câu không hoàn toàn giống với logic vị từ truyền thống, tuy nhiên đây là các câu có dạng Horn clause.

Suy diễn được thực hiện theo kiểu suy diễn tiến, trong đó các câu được xét theo thứ tự xuất hiện của câu trong chương trình. Ngoài ra, Prolog cũng cho phép chứng minh bằng cách phủ định câu truy vấn, sau đó dẫn tới kết luận rằng không thể chứng minh được câu phủ định này.

CHƯƠNG 4: SUY DIỄN XÁC SUẤT

4.1. VẤN ĐỀ THÔNG TIN KHÔNG CHẮC CHẮN KHI SUY DIỄN

Trong chương trước, ta đã xem xét cách biểu diễn tri thức bằng lô gic cũng như một số phương pháp suy diễn lô gic để đưa ra quyết định. Mặc dù các hệ thống lô gic cho phép biểu diễn tri thức một cách rõ ràng và tường minh, việc sử dụng lô gic đòi hỏi tri thức phải được cung cấp một cách đầy đủ, chính xác, không mẫu thuẫn. Nếu yêu cầu này không thỏa mãn, các hệ thống suy diễn dựa trên tri thức vị từ hoặc lô gic mệnh đề không thể sử dụng được.

Trên thực tế, các thông tin, tri thức hay mô hình về thế giới xung quanh thường không đầy đủ, không rõ ràng, không chính xác, có thể mâu thuẫn với nhau. Có thể liệt kê một số nguyên nhân gây ra sự không rõ ràng, không chắc chắn như dưới đây.

- *Do thông tin có chứa đựng yếu tố ngẫu nhiên, yếu tố xác suất*. Ví dụ khi chơi các trò chơi liên quan tới xác suất như chơi bài, chơi cá ngựa.
- *Do không có hiểu biết đầy đủ về vấn đề đang xét*. Ví dụ khi xây dựng hệ thống chẩn đoán bệnh trong y học, do y học hiện đại chưa hiểu biết hoàn toàn chính xác về cơ chế bên trong của nhiều loại bệnh, việc xây dựng quy tắc suy diễn dựa trên kiến thức như vậy sẽ không chắc chắn, không chính xác.
- *Do số các yếu tố liên quan đến quá lớn, không thể xem xét hết*. Ví dụ khi chẩn đoán bệnh, bác sĩ có thể phải chẩn đoán trong khi không có đầy đủ thiết bị để làm mọi loại xét nghiệm cần thiết.
- *Do sai số khi ta lấy thông tin từ môi trường*. Ví dụ trong trường hợp các thiết bị đo có thể có sai số.

Như vậy, trên thực tế, nhiều bài toán đòi hỏi suy diễn, ra quyết định trong khi chưa có đầy đủ thông tin hay thông tin không rõ ràng, không chắc chắn. Vấn đề này thường được gọi là suy diễn trong điều kiện không rõ ràng (reasoning under uncertainty) và không thể giải quyết được bằng suy diễn lô gic truyền thống.

Để suy diễn trong điều kiện không rõ ràng, nhiều cách tiếp cận khác nhau đã được nghiên cứu phát triển, trong đó phải kể đến:

- Cách tiếp cận dựa trên lô gic đa trị: thay vì chỉ cho phép đưa ra kết luận "đúng" hoặc "sai" như lô gic truyền thống, lô gic đa trị cho phép sử dụng nhiều giá trị hơn, ví dụ giá trị "không đúng không sai".
- Lô gic mờ (fuzzy logic): thay vì hai giá trị "đúng" hoặc "sai", biểu thức lô gic mờ có thể nhận giá trị "đúng" với một giá trị hàm thuộc nằm trong khoảng [0,1], thể hiện mức độ đúng của biểu thức đó.
- Lý thuyết khả năng (possibility theory): các sự kiện hoặc công thức được gán một số thể hiện khả năng xảy ra sự kiện đó.
- Suy diễn xác suất: kết quả suy diễn trả về xác suất một sự kiện hay công thức nào đó là đúng.

Trong chương này, ta sẽ xem xét phương pháp suy diễn xác suất cho trường hợp suy diễn có yếu tố không rõ ràng.

4.2. NGUYÊN TẮC SUY DIỄN XÁC SUẤT

Khác với suy diễn lô gic, trong đó các sự kiện, mệnh đề, công thức nhận giá trị đúng hoặc sai, trong suy diễn xác suất, thay vì kết luận một sự kiện hay công thức đúng hoặc sai, ta tính toán niềm tin là sự kiện, công thức đó là đúng hay sai. Niềm tin ở đây được tính bằng xác suất, quá trình tính toán tuân theo các công thức của lý thuyết xác suất. Như vậy, thay vì chỉ sử dụng hai giá trị đúng hoặc sai, suy diễn xác suất cho phép làm việc với vô số giá trị.

Cụ thể, mỗi mệnh đề hoặc biểu thức được gắn một số đo giá trị niềm tin là mệnh đề đó đúng. Mức đo niềm tin được biểu diễn như giá trị xác suất và lý thuyết xác suất được sử dụng để làm việc với mức đo niềm tin này.

Chẳng hạn, với mệnh đề A, ta sử dụng xác suất $P(A)$, $0 \leq P(A) \leq 1$, với ý nghĩa $P(A) = 1$ nếu A đúng, $P(A) = 0$ nếu A sai.

Ví dụ:

- P (Cảm = true) = 0.6 : người bệnh bị cảm với xác suất 60%, "Cảm" là biến ngẫu nhiên có thể nhận 1 trong 2 giá trị {True, False}
- P (trời = nắng ^ gió = mạnh) = 0.8: ta tin rằng trời nắng và gió mạnh với xác suất 80%, trời là biến ngẫu nhiên nhận các giá trị {nắng, mưa, u ám}, gió là biến ngẫu nhiên nhận giá trị {mạnh, yếu, trung bình}.

Bản chất xác suất sử dụng trong suy diễn

Bản chất thống kê: trong lý thuyết xác suất truyền thống, giá trị xác suất được xác định dựa trên quan sát, thực nghiệm, thống kê. Ví dụ, để xác định xác suất trời mưa vào ngày lễ 1-5 ta cần quan sát và thống kê số ngày 1-5 trời mưa trong rất nhiều năm. Việc xác định xác suất như vậy không phải khi nào cũng thực hiện được.

Xác suất dựa trên chủ quan: trong suy diễn xác suất, khi không thể xác định giá trị xác suất bằng thống kê, xác suất có thể xác định một cách chủ quan, dựa trên niềm tin của chuyên gia, của người dùng về sự đúng, sai của các sự kiện. Ví dụ, bác sĩ có thể cung cấp giá trị xác suất về một triệu chứng bệnh nào đó dựa trên kinh nghiệm và ước lượng, thay vì dựa trên thống kê chính xác và cụ thể.

Thu thập và biểu diễn thông tin cho suy diễn xác suất

Để suy diễn xác suất cho một vấn đề nào đó, các bước sau cần được thực hiện:

- Xác định các tham số liên quan tới vấn đề, chẳng hạn trong chẩn đoán bệnh, tham số có thể là "đau đầu", "chán ăn". Mỗi tham số xác định ở trên được biểu diễn bằng một biến ngẫu nhiên tương ứng.
- Xác định miền giá trị cho các biến ngẫu nhiên. Thông thường, mỗi biến ngẫu nhiên có thể nhận một giá trị rời rạc trong miền giá trị của mình. Với những trường hợp đơn giản, biến chỉ có hai giá trị "đúng", "sai" như đối với biến "chán ăn". Những biến như vậy được gọi là biến bool hay biến nhị phân. Trong trường hợp chung, biến có thể nhận nhiều giá trị hơn, chẳng hạn "đau đầu" có thể nhận giá trị "nhẹ", "dữ dội", "không".
- Xác định xác suất ứng với sự kiện biến nhận giá trị nào đó. Ví dụ, P(~ chán ăn) = 0.9, P(đau đầu = dữ dội) = 0.15.

Việc suy diễn xác suất khi đó sẽ qui về tính xác suất cho một hoặc một số các biến nhận những giá trị nào đó.

4.3. MỘT SỐ KHÁI NIỆM VỀ XÁC SUẤT

Trước khi xem xét vấn đề biểu diễn tri thức và suy diễn xác suất, phần này nhắc lại một số kiến thức về xác suất cần thiết cho những nội dung tiếp theo:

4.3.1. Các tiên đề xác suất

- $0 \le P(A = a) \le 1$ với mọi giá trị trong miền xác định của biến A
- $P(\text{true}) = 1$; $P(\text{false}) = 0$. [2]
- $P(A \vee B) = P(A) + P(B)$ nếu A và B loại trừ tương hỗ.

Các tính chất:

Ngoài các tiên đề trên, xác suất có một số tính chất quan trọng sau

- $P(\sim A) = 1 - P(A)$ (*ký hiệu ~ được dùng tương đương với* $\neg$)
- $P(A) = P(A \wedge B) + P(A \wedge \sim B)$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- $\Sigma_a P(A = a) = 1$, trong đó tổng lấy theo các giá trị a thuộc miền giá trị của A

4.3.2. Xác suất đồng thời

Xác suất đồng thời của các sự kiện là xác suất quan sát thấy đồng thời xẩy ra các sự kiện đó và được ký hiệu

$$P(V_1 = v_1, V_2 = v_2, \ldots, V_n = v_n)$$

Với một bài toán xác suất, nếu chúng ta biết tất cả phân bố xác suất đồng thời tức là xác suất tất cả các tổ hợp giá trị của các biến liên quan thì ta có thể tính được xác suất mọi mệnh đề liên quan tới bài toán đang xét.

Ví dụ: Cho 3 biến nhị phân (True, False): "Chim", "Non", "Bay được". Ta có các xác suất đồng thời cho trong bảng 4.1.

Từ bảng xác suất này, ta có thể tính giá trị mọi xác suất liên quan tới 3 biến của bài toán. Sau đây là ví dụ tính một số xác suất:

Xác suất một vật nào đó là "chim":

$$P(\text{Chim} = \text{True}) = 0 + 0{,}2 + 0{,}04 + 0.01 = 0{,}25.$$

Xác suất "chim không biết bay":

$$P(\text{Chim} = \text{T}, \text{Bay} = \text{F}) = 0{,}04 + 0{,}01 = 0{,}05.$$

[2] Lưu ý, đây là tiên đề dùng cho suy diễn xác suất, trong trường hợp tổng quát, ta có $P(\Omega)=1$, $P(\varnothing)=0$, trong đó Ω là toàn bộ không gian lấy mẫu.

Bảng 4.1. Ví dụ bảng xác suất đồng thời cho ba biến nhị phân

Chim	Bay được	Non	Xác suất
T	T	T	0
T	T	F	0,2
T	F	T	0,04
T	F	F	0,01
F	T	T	0,01
F	T	F	0,01
F	F	T	0,23
F	F	F	0,5

4.3.3. Xác suất điều kiện

Xác suất điều kiện là xác suất xảy ra sự kiện A khi biết sự kiện B xảy ra. Xác suất điều kiện được tính như sau:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

Xác suất điều kiện đóng vai trò quan trọng trong suy diễn xác suất và có ý nghĩa như sau:

- Khi $P(A \mid B) = 1$: B đúng thì chắc chắn suy ra A
- Khi $P(A \mid B) = 0$: B đúng thì suy ra ~A.
- Có thể xem B là bằng chứng hoặc quan sát và A là kết luận. Khi đó, $P(A \mid B)$ là xác suất hoặc niềm tin A là đúng khi quan sát thấy B.

Như ta sẽ thấy ở dưới, quá trình suy diễn là quá trình tính xác suất điều kiện của kết luận khi biết bằng chứng. Cụ thể là khi cho biết các bằng chứng $E_1, \ldots, E_n$, suy diễn xác suất được thực hiện bằng cách tính xác suất điều kiện $P(Q \mid E_1, \ldots, E_n)$, tức là niềm tin kết luận Q đúng khi có các bằng chứng $E_1, \ldots, E_n$.

Các tính chất của xác suất điều kiện

- Qui tắc nhân

  $P(A, B) = P(A \mid B) * P(B)$
- Qui tắc chuỗi (là mở rộng của quy tắc nhân)

  $P(A, B, C, D) = P(A \mid B, C, D) * P(B \mid C, D) * P(C \mid D) * P(D)$
- Quy tắc chuỗi có điều kiện

  $P(A, B \mid C) = P(A \mid B, C) * P(B \mid C)$
- Cộng

  $P(A) = \sum_b P(A|B = b) * P(B = b)$
- $P(\sim B \mid A) = 1 - P(B \mid A)$

- Quy tắc Bayes

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

ta sẽ xem xét kỹ hơn quy tắc Bayes trong một phần sau.

Tính xác suất điều kiện từ bảng xác suất đồng thời

Trong trường hợp có đầy đủ xác suất đồng thời như đã xem xét ở trên, ta có thể thực hiện suy diễn thông qua tính xác suất điều kiện. Sau đây là một số ví dụ minh họa sử dụng bảng xác suất đồng thời trong phần trên với ký hiệu "Chim" là C, "Bay" là B và "Non" là N

Ví dụ 1. Giả sử các bằng chứng cho thấy một sinh vật biết bay, cần tính xác suất sinh vật đó là chim. Ta có:

P (Chim | Bay) = $P(C \mid B)$

$= P(C, B) / P(B)$ (theo định nghĩa)

$$= \frac{P(C, B, N) + P(C, B, \sim N)}{P(C, B, N) + P(\sim C, B, N) + P(C, \sim B, N) + P(\sim C, \sim B, N)}$$

$= (0 + 0{,}2) / (0 + 0{,}04 + 0{,}01 + 0{,}23)$

Ví dụ 2. Ví dụ này minh họa cho việc kết hợp nhiều bằng chứng. Giả sử có bằng chứng sinh vật biết bay và không còn non, cần tính xác suất đây không phải là Chim. Ta có:

$$P(\sim \text{Chim} \mid \text{Bay}, \sim \text{Non}) = \frac{P(\sim C, B, \sim N)}{P(C, B, \sim N) + P(\sim C, B, \sim N)} = \frac{0{,}01}{0{,}01 + 0{,}2} = 0{,}048$$

Trong trường hợp tổng quát, khi cho bảng xác suất đồng thời của n biến V1, …, Vn, ta có thể tính xác suất của một số biến này khi biết giá trị một số biến khác như sau:

➔ **Công thức tổng quát**

$$P(V_1 = v_1, \ \ldots V_k = v_k \mid V_{k+i} = v_{k+i}, \ldots V_n = v_n)$$

$$= \frac{\text{tổng các dòng có } V_1 = v_1, \ldots, V_K = v_k, V_{k+i} = v_{k+i}, \ldots V_n = v_n}{\text{tổng các dòng có } V_{k+i} = v_{k+i}, \ldots V_n = v_n}$$

Một cách hình thức hơn, gọi các biến cần tính xác suất là Q, các biến đã biết là E, các biến còn lại (ngoài E và Q) là Y, ta có

$$P(Q \mid E) = \frac{\sum_{Y} P(Q, E, Y)}{\sum_{Q,Y} P(Q, E, Y)}$$

Mặc dù công thức trên tương đối đơn giản nhưng trên thực tế, khi số lượng biến tăng lên, số lượng các dòng chứa tổ hợp giá trị các biến ở tử số và mẫu số của công thức trên sẽ tăng theo hàm mũ, dẫn tới kích thước bảng xác suất đồng thời quá lớn nên không thể sử dụng cho suy diễn xác suất được.

4.3.4. Tính độc lập xác suất

Sự kiện A độc lập về xác suất với sự kiện B nếu:

$$P(A \mid B) = P(A)$$

Tức là biết giá trị của B không cho ta thêm thông tin gì về B.

Tương đương với công thức trên, khi A độc lập với B, ta có:

$$P(A, B) = P(A)\ P(B)$$

và

$$P(B \mid A) = P(B)$$

Mở rộng cho trường hợp xác suất điều kiện, ta nói rằng A độc lập có điều kiện với B khi biết C nếu:

$$P(A \mid B, C) = P(A \mid C) \text{ hoặc } P(B \mid A, C) = P(B \mid C)$$

khi đó ta có:

$$P(A, B \mid C) = P(A \mid C)\ P(B \mid C)$$

Ý nghĩa: nếu đã biết giá trị của C thì việc biết giá trị của B không cho ta thêm thông tin về A.

4.3.5. Quy tắc Bayes

Như đã nhắc tới ở trên, quy tắc Bayes có dạng sau:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Quy tắc Bayes đóng vai trò quan trọng trong suy diễn xác suất. Thông thường, bài toán suy diễn đòi hỏi tính $P(A \mid B)$, tuy nhiên trong nhiều trường hợp, việc tính $P(B \mid A)$ có thể dễ dàng hơn. Khi đó, quy tắc Bayes cho phép ta quy việc tính $P(A \mid B)$ về tính $P(B \mid A)$.

Ví dụ: Giả sử cần tính xác suất một người bị cúm khi biết người đó đau đầu, tức là tính

P (cúm | đau đầu)

Để tính xác suất này, cần xác định có bao nhiêu người bị đau đầu trong một cộng đồng dân cư, sau đó đếm xem có bao nhiêu người trong số người đau đầu bị cúm. Rõ ràng việc thống kê những người đau đầu tương đối khó khăn do không phải ai đau đầu cũng thông báo cho cơ sở y tế. Ngược lại, để tính P(đau đầu | cúm), ta cần đếm số người bị cúm sau đó xác định xem trong số này bao nhiêu người có triệu chứng đau đầu. Việc này được thực hiện tương đối dễ dàng hơn, chẳng hạn bằng cách phân tích bệnh án những người bị cúm.

Sau đây, ta xem xét một ví dụ sử dụng quy tắc Bayes cho suy diễn.

Ví dụ: Một người làm xét nghiệm về một căn bệnh hiếm gặp (ký hiệu HG) và nhận được kết quả dương tính. Biết rằng thiết bị xét nghiệm không chính xác hoàn toàn. Cụ thể, thiết bị cho kết quả dương tính đối với 98% người có bệnh. Trong trường hợp người không có bệnh vẫn có 3% khả năng xét nghiệm cho kết quả dương tính. Ngoài ra, ta biết rằng xác suất quan sát thấy bệnh này trong cộng đồng là 8 phần nghìn. Yêu cầu đặt ra là cần xác định xem người xét nghiệm với kết quả dương tính có bệnh không.

Ký hiệu sự kiện có bệnh là B, xét nghiệm dương tính là $\oplus$, như vậy kết quả âm tính sẽ là $\sim\oplus$. Để kết luận người khám có bị bệnh không ta cần so sánh xác suất $P(B \mid \oplus)$ và $P(\sim B \mid \oplus)$.

Theo dữ kiện đã cho, ta có:

$$P(B) = 0.008 \qquad P(\sim B) = 0.992$$

$$P(\oplus \mid B) = 0.98 \qquad P(\sim \oplus \mid B) = 1\text{-}0.98 = 0.02$$

$$P(\sim\oplus \mid \sim B) = 0.97 \; P(\oplus \mid \sim B) = 0.03$$

Sử dụng quy tắc Bayes, ta có:

$$P(B \mid \oplus) = \frac{P(\oplus \mid B)P(B)}{P(\oplus)} = \frac{0.98 * 0.008}{P(\oplus)} = \frac{0.00784}{P(\oplus)}$$

$$P(\sim B \mid \oplus) = \frac{P(\oplus \mid \sim B)P(\sim B)}{P(\oplus)} = \frac{0.03 * 0.992}{P(\oplus)} = \frac{0.02976}{P(\oplus)}$$

Như vậy, $P(\sim B \mid \oplus) > P(B \mid \oplus)$ và do đó ta kết luận người tới khám không bị bệnh. Sở dĩ xác suất không bệnh lớn hơn là do xác suất $P(B)$, còn gọi là xác suất tiền nghiệm, của bệnh rất nhỏ. Trong trường hợp như vậy, để khẳng định bị bệnh cần phải có khả năng xét nghiệm với đội chính xác rất cao.

Cần lưu ý thêm rằng, để so sánh $P(\sim B \mid \oplus)$ và $P(B \mid \oplus)$ ta không cần tính giá trị đúng của hai xác suất này do hai biểu thức có chung mẫu số $P(\oplus)$. Nếu cần tính giá trị đúng của hai xác suất đó, ta có thể làm như sau:

do $P(\sim B \mid \oplus) + P(B \mid \oplus) = 1$

nên $$\frac{0.00784}{P(\oplus)} + \frac{0.02976}{P(\oplus)} = 1$$

từ đây: $P(\oplus) = 0.00784 + 0.02976$

Thay vào các biểu thức trên, ta có $P(\sim B \mid \oplus)=0.79$ và $P(B \mid \oplus) = 0.21$

Quy tắc Bayes cũng có thể kết hợp với tính độc lập về xác suất. Ví dụ tiếp theo sẽ minh họa cho việc kết hợp này.

Ví dụ: Tính $P(A \mid B, C)$, cho biết B độc lập với C nếu biết A:

$$P(A|B, C) = \frac{P(B,C|A) * P(A)}{P(B,C)} = \frac{P(B|A) * P(C|A) * P(A)}{P(B,C)}$$

Bài tập: Cho 3 biến nhị phân: gan BG, vàng da VD, thiếu máu TM.

- Giả sử VD độc lập với TM
- Biết $P(BG) = 10^{-17}$
- Có người khám bị VD
- Biết $P(VD) = 2^{-10}$ và $P(VD|BG) = 2^{-3}$

a. Xác suất người khám bị bệnh là bao nhiêu?
Cho biết thêm người đó bị thiếu máu và $P(TM) = 2^{-6}$, $P(TM|BG) = 2^{-1}$. Hãy tính xác suất người khám bị bệnh BG.

Giải:

a. $$P(BG|VD) = \frac{P(VD|BG) * P(BG)}{P(VD)} = \frac{2^{-3} * 10^{-17}}{2^{-10}}$$

b. Ta cần tính xác suất điều kiện

$$P(BG \mid VD, TM) = \frac{P(VD|BG) * P(TM|BG) * P(BG)}{P(TM,VD)}$$

Mà TM và VD độc lập nên ta có:

$$P(BG \mid VD, TM) = \frac{2^{-3} * (1 - 2^{-1}) * 10^{-17}}{(1 - 2^{-6}) * 2^{-10}}$$

Ví dụ trên cho thấy, việc sử dụng tính độc lập (có điều kiện) giữa các biến cho phép rút gọn đáng kể quá trình tính toán xác suất điều kiện khi thực hiện suy diễn.

4.4. MẠNG BAYES

Trong các phần trước ta đã làm quen với vấn đề suy diễn xác suất, trong đó cho trước một số bằng chứng $E_1, \ldots, E_n$, cần tính xác suất điều kiện $P(Q \mid E_1, \ldots, E_n)$ để kết luận về câu truy vấn Q.

Xác suất điều kiện trên có thể tính được nếu biết toàn bộ xác suất đồng thời của các biến ngẫu nhiên. Tuy nhiên, trên thực tế, các bài toán thường có số lượng biến ngẫu nhiên lớn, dẫn tới số lượng xác suất đồng thời tăng theo hàm mũ. Do vậy, liệt kê và sử dụng bảng xác suất đồng thời đầy đủ để suy diễn là không thực tế.

Để khắc phục khó khăn trên, trong phần này ta sẽ xem xét cách sử dụng mạng Bayes như một mô hình biểu diễn xác suất rút gọn và cách thực hiện suy diễn xác suất trên mạng Bayes.

4.4.1. Khái niệm mạng Bayes

Để tiện cho việc trình bày khái niệm mạng Bayes, xét một ví dụ sau[3].

Một người đi làm về và muốn dự đoán xem ở nhà có người không thông qua một số dấu hiệu có thể quan sát được. Cho biết một số dữ kiện sau:

- *Nếu cả nhà đi vắng thì thường bật đèn ngoài sân. Tuy nhiên, đèn ngoài sân có thể được cả trong một số trường hợp có người ở nhà, ví dụ khi có khách đến chơi.*
- *Nếu cả nhà đi vắng thì thường buộc chó ở sân sau.*
- *Tuy nhiên chó có thể được buộc ở sân sau cả khi có người ở nhà nếu như chó bị đau bụng.*
- *Nếu chó buộc ở ngoài thì có thể nghe tiếng sủa, tuy nhiên có thể nghe tiếng sủa (của chó hàng xóm) cả khi chó không buộc ở ngoài.*

Để thực hiện suy diễn xác suất cho bài toán trên, trước tiên cần xây dựng mô hình xác suất. Ta sẽ sử dụng năm biến ngẫu nhiên sau để thể hiện các dữ kiện liên quan tới bài toán.

O: không ai ở nhà

L: đèn sáng

D: chó ở ngoài

B: chó bị ốm.

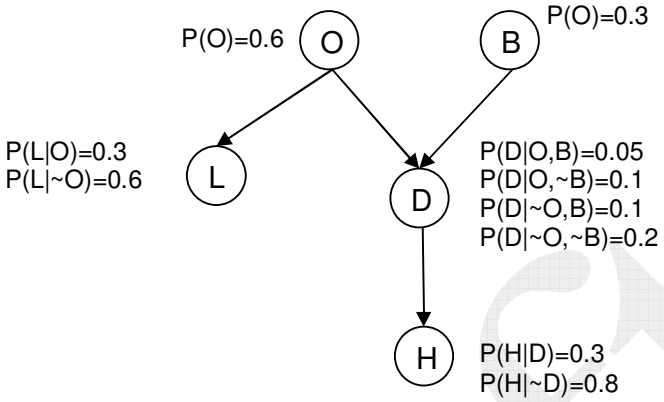
H: nghe thấy tiếng sủa.

Việc phân tích bài toán cho thấy:

- Nếu biết D thì H không phụ thuộc vào O, L, B.
- Nếu biết B thì D độc lập với O.
- O và B độc lập với nhau

[3] Đây là một ví dụ được sử dụng trong bài báo "Bayesian networks without tears" đăng trên tạp chí AI Magazine năm 1991.

Tiếp theo, ta xây dựng một đồ thị, trong đó mỗi biến ngẫu nhiên ở trên được biểu diễn bởi một nút như trên hình vẽ dưới đây (hình 4.1). Các nút được nối với nhau bằng những cung có hướng sao cho hai hai nút có quan hệ phụ thuộc được nối bởi một cung và hướng của cung thể hiện chiều tác động của nút gốc tới nút đích.



Hình 4.1: Một ví dụ mạng Bayes

Sau khi có đồ thị, ta thêm vào *bảng xác suất điều kiện*. Bảng xác suất điều kiện thể hiện xác suất của biến khi biết giá trị cụ thể của các biến ở các nút cha mẹ. Trong trường hợp nút không có cha mẹ, xác suất trở thành xác suất tiền nghiệm. Để thuận tiện, bảng xác suất điều kiện được thể hiện ngay trên hình vẽ cùng với đồ thị.

Đồ thị vừa xây dựng cùng với các bảng xác suất điều kiện tạo thành mạng Bayes cho bài toán trong ví dụ trên.

Định nghĩa: Mạng Bayes là một mô hình xác suất bao gồm 2 phần

- Phần thứ nhất là một đồ thị có hướng không chứa chu trình (tức là đường đi có đầu và cuối trùng nhau), trong đó mỗi nút tương ứng với một biến ngẫu nhiên, các cung thể hiện mối quan hệ phụ thuộc giữa các biến.
- Phần thứ hai là các bảng xác suất điều kiện: mỗi nút có một bảng xác suất điều kiện cho biết xác suất các giá trị của biến khi biết giá trị các nút cha mẹ.

Cấu trúc của đồ thị trong mạng Bayes thể hiện mối quan hệ phụ thuộc hoặc độc lập giữa các biến ngẫu nhiên của bài toán. Hai nút được nối với nhau bởi một cung khi giữa hai nút có quan hệ trực tiếp với nhau, trong đó giá trị nút gốc ảnh hưởng tới giá trị nút đích. Nếu có một cung theo hướng từ nút A tới nút B thì A được gọi là cha (mẹ) của B.

Lưu ý rằng trong cấu trúc của mạng Bayes không cho phép có chu trình. Hạn chế này ảnh hưởng tới khả năng mô hình hóa của mạng Bayes trong một số trường hợp tuy nhiên cho phép đơn giản hóa việc xây dựng và suy diễn trên mạng Bayes.

Bảng xác suất điều kiện xác định cụ thể ảnh hưởng của các nút cha mẹ tới giá trị nút con. Ở đây ta chỉ xét trường hợp biến ngẫu nhiên có thể nhận giá trị rời rạc và bảng xác suất điều kiện được cho theo tổ hợp giá trị của các nút cha mẹ. Mỗi dòng trong bảng tương ứng với một điều kiện cụ thể, thực chất là một tổ hợp giá trị các nút cha. Ví dụ, trong mạng Bayes của ví dụ trên, dòng thứ nhất trong bảng xác suất của nút D ứng với điều kiện trong đó O=true

và B=true. Nếu nút không có cha mẹ thì bảng xác suất chỉ gồm một dòng duy nhất như trường hợp với nút O và nút B.

4.4.2. Tính độc lập xác suất trong mạng Bayes

Mạng Bayes thể hiện hai thông tin chính.

Thứ nhất, đây là biểu diễn rút gọn của toàn bộ xác suất đồng thời. Trong ví dụ trên ta chỉ cần 10 xác suất thay vì 2^5-1 xác suất đồng thời. Tùy theo kích thước và đặc điểm cụ thể của bài toán, hiệu quả của việc rút gọn số lượng xác suất có thể lớn hơn rất nhiều. Chẳng hạn, với mạng gồm 30 nút nhị phân, mỗi nút có 5 nút cha, ta cần tất cả 960 xác suất điều kiện cho mạng Bayes, trong khi bảng xác suất đồng thời cho 30 biến như vậy phải có 2^{30}-1, tức là hơn một tỷ dòng.

Thứ hai, mạng Bayes cho thấy sự phụ thuộc hoặc độc lập xác suất có điều kiện giữa các biến. Về thực chất, chính việc độc lập về xác suất dẫn tới khả năng biểu diễn rút gọn các xác suất đồng thời.

Tính độc lập xác suất trong mạng Bayes thể hiện qua tính chất sau.

Tính chất:

- Mỗi nút trên mạng Bayes độc lập có điều kiện với tất cả các nút không phải là hậu duệ của nút đó nếu biết giá trị các nút cha.
- Mỗi nút độc lập có điều kiện với tất cả các nút khác trên mạng nếu biết giá trị tất cả nút cha, nút con và nút cha của các nút con.

Ví dụ: Theo mạng Bayes trong ví dụ trên H độc lập với O, L, B nếu biết giá trị của D.

Tính các xác suất đồng thời

Sử dụng tính độc lập xác suất vừa phát biểu ở trên, có thể tính xác suất đồng thời của tất cả các biến. Xét ví dụ sau

Ví dụ :cần tính *P(H, D, L, ~O, B)*

Theo công thức chuỗi:

$$P(H, D, L, \sim O, B) = P(H| D, L, \sim O, B) * P(D| L, \sim O, B) * P(L| \sim O, B) * P(\sim O|B) * P(B)$$

Do tính độc lập xác suất (có điều kiện):

$P(H| B, D, \sim O, L) = P(H|D)$

$P(D| L, \sim O, B) = P(D| \sim O, B)$

$P(L| \sim O, B) = P(L| \sim O)$

$P(\sim O|B)=P(O)$

do vậy,

$$P(H, D, L, \sim O, B) = P(H| D, L, \sim O, B) * P(D| L, \sim O, B) * P(L| \sim O, B) * P(\sim O|B) * P(B)$$
$$= P(H|\sim O) * P(D| \sim O, B) * P(L|\sim O) * P(\sim O) * P(B)$$

Một cách tổng quát, giả sử mạng có n nút tương ứng với n biến ngẫu nhiên $X_1, \ldots, X_n$ của bài toán đang xét. Từ thông tin của mạng, có thể tính mọi xác suất đồng thời của n biến, trong đó mỗi xác suất đồng thời có dạng $P(X_1 = x_1 \wedge X_2 = x_2 \wedge \ldots \wedge X_n = x_n)$ hay viết gọn là $P(x_1, \ldots, x_n)$. Xác suất đồng thời được tính theo công thức tổng quát sau:

$$P(X_1 = x_1,..., X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid cha\_me(X_i))$$

hay viết gọn là

$$P(x_1,..., x_n) = \prod_{i=1}^{n} P(x_i \mid cha\_me(X_i))$$

trong đó $cha\_me(X_i)$ là giá trị cụ thể các nút cha mẹ của nút X_i.

Để minh họa cho công thức trên, ta sẽ tính xác suất xảy ra tình huống ở nhà có người, chó bị ốm và được buộc ngoài sân, đồng thời đèn không sáng và nghe tiếng chó sủa. Xác suất tình huống này chính là *P(B, ~O, D, ~L, H)* và được tính như sau:

$$P(B, \neg O, D, \neg L, H) = P(B) * P(\neg O) * P(D|\neg O, B) * P(H|D) * P(\neg L|\neg O)$$
$$= 0,3 * 0,4 * 0,05 * 0,7 * 0,3$$
$$= 0,00126$$

Trong một phần trên ta đã thấy rằng nếu có mọi xác suất đồng thời thì có thể thực hiện suy diễn xác suất cho mọi dạng câu truy vấn. Như vậy, với mạng Bayes ta có thể suy diễn bằng cách trước tiên tính ra mọi xác suất đồng thời cần thiết. Tuy nhiên, cách này đòi hỏi tính toán nhiều và vì vậy trên thực tế thường sử dụng một số phương pháp suy diễn khác hiệu quả hơn. Vấn đề này sẽ được nhắc tới trong một phần sau.

4.4.3. Cách xây dựng mạng Bayes

Để có thể sử dụng, trước tiên cần xây dựng ra mạng Bayes. Quá trình xây dựng mạng Bayes bao gồm việc xác định tất cả các biến ngẫu nhiên liên quan, xác định cấu trúc đồ thị của mạng, và cuối cùng là xác định giá trị cho các bảng xác suất điều kiện. Trong phần này, ta sẽ coi như đã có biến ngẫu nhiên, việc xây dựng mạng chỉ bao gồm xác định cấu trúc và bảng xác suất điều kiện.

Có hai cách tiếp cận chính để xây dựng mạng Bayes.

- Cách thứ nhất do con người (chuyên gia) thực hiện dựa trên hiểu biết của mình về bài toán đang xét. Việc xây dựng mạng được chia thành hai bước: xác định cấu trúc đồ thị và điền giá trị cho bảng xác suất điều kiện.
- Cách thứ hai là tự động xác định cấu trúc và xác suất điều kiện từ dữ liệu. Ở đây, dữ liệu có dạng giá trị các biến ghi nhận được trong quá khứ, ví dụ ta có thể ghi lại tổ hợp cá giá trị của năm biến trong ví dụ trên trong thời gian vài năm. Quá trình xây dựng mạng khi đó bao gồm xác định cấu trúc của đồ thị và bảng xác suất điều kiện sao cho phân bố xác suất do mạng thể hiện phù hợp nhất với tần suất xuất hiện các giá trị trong tập dữ liệu.

Phần này chỉ xem xét cách xây dựng mạng do con người thực hiện và mô tả một quy trình cụ thể cho việc xây dựng mạng.

Các bước xây dựng mạng được thực hiện như trên hình 4.2. Sau khi đã có cấu trúc mạng, chuyên gia sẽ xác định giá trị cho các bảng xác suất điều kiện. Thông thường, việc xác định giá trị xác suất điều kiện khó hơn nhiều so với việc xác định cấu trúc mạng, tức là xác định quan hệ giữa các nút.

B1: Xác định các biến ngẫu nhiên cho phép mô tả miền của bài toán.

B2: Sắp xếp các biến theo một thứ tự nào đó. Ví dụ theo thứ tự sau: X_1, X_2 … X_n.

B3: *For i = 1 to n do*

a. Thêm một nút mới X_i vào mạng

b. Xác định tập Cha_Mẹ(X_i) là tập nhỏ nhất các nút đã có trước đó sao cho X_i độc lập có điều kiện với tất cả nút còn lại khi biết bố mẹ của X_i.

c. Với mỗi nút thuộc tập Cha_Mẹ(X_i). Ta thêm một cạnh có hướng từ nút đó tới X_i.

d. Xác định bảng xác suất điều kiện cho X_i the các giá trị của bố mẹ hoặc bằng xác suất tiền nghiệm nếu X_i không có bố mẹ.

Hình 4.2**:** Phương pháp xây dựng mạng Bayes

Để minh họa, xét ví dụ sau. Một người vừa lắp hệ thống báo động chống trộm ở nhà. Hệ thống sẽ phát tiếng động khi có trộm. Tuy nhiên, hệ thống có thể báo động (sai) nếu có chấn động do động đất. Trong trường hợp nghe thấy hệ thống báo động, hai người hàng xóm tên làm Nam và Việt sẽ gọi điện cho chủ nhà. Do nhiều nguyên nhân khác nhau, Nam và Việt có thể thông báo sai, chẳng hạn do ồn nên không nghe thấy chuông báo động hoặc ngược lại, nhầm âm thanh khác là tiếng chuông.

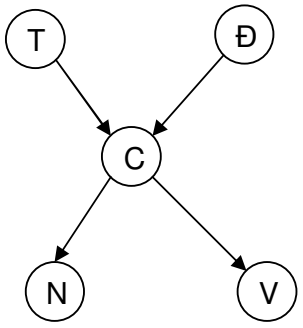
Theo phương pháp trên, các bước xây dựng mạng được thực hiện như sau.

B1: lựa chọn biến: sử dụng 5 biến sau

T (có trộm), Đ (động đất), B (chuông báo động), N (Nam gọi điện), V (Việt gọi điện)

B2: các biến được sắp xếp theo thứ tự T, Đ, B, N, V

B3: thực hiện như các bước ở hình vẽ, ta xây dựng được mạng thể hiện trên hình sau (để đơn giản, trên hình vẽ chỉ thể hiện cấu trúc và không có bảng xác suất điều kiện).



Hình 4.3.: Kết quả xây dựng mạng Bayes cho ví dụ chuông báo trộm

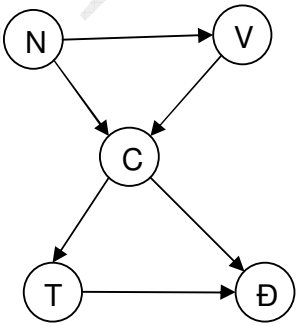
Ảnh hưởng của việc sắp xếp các nút tới kết quả xây dựng mạng.

Trên thực tế, việc xây dựng mạng Bayes không đơn giản, đặc biệt trong việc chọn thứ tự các nút đúng để từ đây chọn được tập nút cha có kích thước nhỏ. Để làm rõ điểm này, ta giả sử trong ví dụ trên, các biến được xếp theo thứ tự khác: N, V, C, T, Đ.

Các bước thêm nút sẽ thực hiện như sau:

- Thêm nút N: không có nút cha
- Thêm nút V: nếu Nam gọi điện, xác suất Việt gọi điện sẽ tăng lên do sự kiện Nam gọi điện nhiều khả năng do có báo động và do vậy xác suất Việt nghe thấy chuông và gọi điện tăng theo. Do vậy N có ảnh hưởng tới V và được thêm vào tập cha của V.
- Thêm C: Nếu Nam và Việt cùng gọi thì khả năng có chuông cao hơn, do vậy cần thêm cả N và V vào tập cha của C.
- Thêm T: Nếu đã biết trạng thái của chuông thì không cần quan tâm tới Nam và Việt nữa, do vậy chỉ có C là cha của T.
- Thêm Đ: nếu có chuông, khả năng động đất tăng lên. Tuy nhiên, nếu đồng thời ta biết có trộm thì việc có trộm giải thích phần nào nguyên nhân chuông kêu. Như vậy, cả chuông và có trộm ảnh hưởng tới xác suất động đất, tức là C và T đều là cha của Đ.

Kết quả của mạng Bayes xây dựng theo thứ tự mới được thể hiện trên hình dưới. So sánh với kết quả ở trên, mạng Bayes mới phức tạp hơn, theo nghĩa có nhiều cung hơn hay trung bình các nút có nhiều nút cha hơn. Ngoài ra, ý nghĩa một số quan hệ trên mạng rất không trực quan và khó giải thích, chẳng hạn việc xác suất động đất phục thuộc vào chuông báo động và có trộm. Như vậy, mặc dù cả hai mạng Bayes xây dựng ở trên đều đúng theo nghĩa đảm bảo các ràng buộc về xác suất và đều cho phép tính ra các xác suất đồng thời, việc lựa chọn không đúng thứ tự nút sẽ làm mạng khó hiểu và phức tạp hơn.



Hình 4.4: Kết quả xây dựng mạng Bayes khi sử dụng thứ tự các nút khác

Từ ví dụ trên ta có thể đưa ra một số nhận xét về kết quả xây dựng mạng Bayes.

Nhận xét:

- Cùng một tập hợp biến có thể xây dựng nhiều mạng Bayes khác nhau.
- Thứ tự sắp xếp có ảnh hưởng tới mạng Bayes. Nên sắp xếp sao cho các nút đóng vai trò nguyên nhân đứng trước nút hệ quả.

- Tất cả các mạng được xây dưng như trên đều hợp lệ, theo nghĩa không vi phạm các ràng buộc về xác suất và đều cho phép thực hiện suy diễn.

4.5. SUY DIỄN VỚI MẠNG BAYES

Suy diễn xác suất là tìm xác suất hậu nghiệm hay xác suất điều kiện của một số biến Q (gọi là biến truy vấn) khi có một số bằng chứng, tức là khi biết giá trị của một số biến khác E_1, …, E_m, (gọi là biến bằng chứng). Chẳng hạn, với mạng Bayes cho ở ví dụ đầu tiên, ta có thể quan sát thấy đèn sáng và không nghe tiếng chó sủa, từ đây cần kết xác định trong nhà có người hay không bằng cách tính xác suất điều kiện P(~ O | L, H).

Phần này sẽ trình bày một số kỹ thuật suy diễn cho những trường hợp đơn giản. Việc trình bày đầy đủ các kỹ thuật suy diễn trên mạng Bayes tương đối phức tạp và sẽ không được trình bày ở đây.

4.5.1.Suy diễn dựa trên xác suất đồng thời

Phần giới thiệu về xác suất điều kiện ở trên đã giải thích cách tính xác suất điều kiện khi biết đầy đủ xác suất đồng thời. Cụ thể là, với các biến truy vấn Q, biến bằng chứng E, và các biến còn lại Y, xác suất điều kiện được tính như sau:

$$P(Q \mid E) = \frac{\sum_{Y} P(Q, E, Y)}{\sum_{Q,Y} P(Q, E, Y)}$$

Do mạng Bayes cho phép xác định toàn bộ xác suất đồng thời cần thiết, việc suy diễn có thể thực hiện bằng cách tính xác suất đồng thời, sau đó sử dụng công thức trên.

Xét ví dụ dự đoán ở nhà có người hay không ở trên, giả sử ta biết rằng chó đang bị đau bụng và không nghe tiếng sủa, cần xác định xác suất đèn sáng, tức là xác suất P(L | B, ~H). Theo công thức trên ta có:

$$P(L \mid B, \sim H) = \frac{\sum_{O,D} P(L, B, \sim H, O, D)}{\sum_{L,O,D} P(L, B, \sim H, O, D)}$$

trong đó

$$\sum_{O,D} P(L, B, \sim H, O, D) = P(L, B, \sim H, O, D) + P(L, B, \sim H, O, \sim D) + P(L, B, \sim H, \sim O, D) + P(L, B, \sim H, \sim O, \sim D)$$

và

$$\sum_{L,O,D} P(L, B, \sim H, O, D) = P(L, B, \sim H, O, D) + P(L, B, \sim H, O, \sim D) + P(L, B, \sim H, \sim O, D) + P(L, B, \sim H, \sim O, \sim D) + P(\sim L, B, \sim H, O, D) + P(L, B, \sim H, O, \sim D) + P(\sim L, B, \sim H, \sim O, D) + P(L, B, \sim H, \sim O, \sim D)$$

các xác suất đồng thời được tính theo công thức xác suất đồng thời cho mạng Bayes như trong phần trước.

Cách suy diễn thông qua xác suất đồng thời có nguyên tắc đơn giản tuy nhiên đòi hỏi tính tổng của rất nhiều xác suất đồng thời. Với số lượng biến ngẫu nhiên lớn, số lượng xác

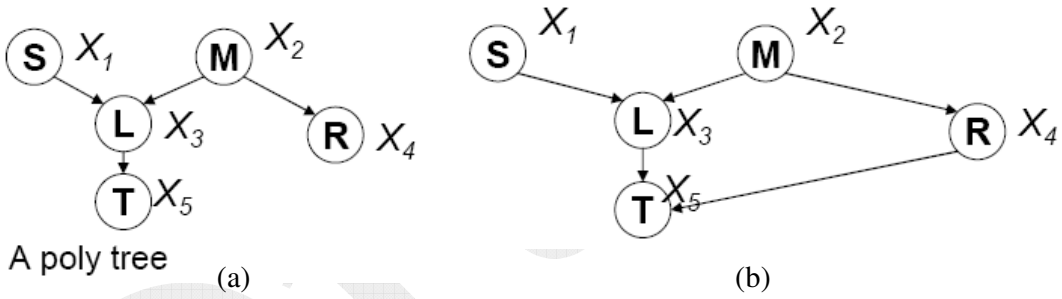
suất đồng thời cần tính tăng theo hàm mũ, dẫn tới độ phức tạp tính toán rất lớn và do vậy không thực tế.

4.5.2.Độ phức tạp của suy diễn trên mạng Bayes

Ta đã thấy, việc tính chính xác giá trị $P(Q \mid E)$ thông qua xác suất đồng thời đòi hỏi độ phức tạp tính toán hàm mũ. Vậy việc suy diễn thông qua tính chính xác xác suất điều kiện nói chung (tạm gọi là suy diễn chính xác) có độ phức tạp tính toán ra sao khi sử dụng những phương pháp khác?

Đối với mạng có dạng liên kết đơn hay còn gọi là polytree: Mạng liên kết đơn hay mạng polytree là mạng trong đó giữa hai nút bất kỳ chỉ tồn tại duy nhất một đường đi (hình 4.5.a). Trong trường hợp này, tồn tại thuật toán cho phép suy diễn chính xác với độ phức tạp tuyến tính (tỷ lệ thuận) với kích thước của mạng. Ở đây, kích thước mạng được tính bằng tổng số phần tử của các bảng xác suất điều kiện. Trong trường hợp số nút cha được giới hạn bởi hằng số, độ phức tạp sẽ tỷ lệ tuyến tính với số nút.

Trong phạm vi chương trình, ta sẽ không đề cập tới thuật toán suy diễn cụ thể với độ phức tạp tuyến tính như vừa nêu.



Hình 4.5: Mạng Bayes dạng liên kết đơn (a) và không phải liên kết đơn (b) với hai đường đi từ M tới T

Đối với mạng không kiên kết đơn hay mạng đa liên kết: đây là những mạng mà giữa hai nút có thể có nhiều hơn một đường đi (hình 4.5.b). Trong trường hợp này, việc tính chính xác xác suất điều kiện là bài toán NP-đầy đủ và do vậy không thể thực hiện khi mạng có kích thước lớn.

Để thực hiện suy diễn trong trường hợp mạng đa liên kết, có thể sử dụng các thuật toán cho phép tính xác suất điều kiện một cách xấp xỉ, chẳng hạn bằng cách lấy mẫu thống kê, hay sử dụng các thuật toán lan tỏa (propagation). Một kỹ thuật khác cũng có thể sử dụng là biến đổi mạng đa liên kết thành mạng liên kết đơn bằng cách chấp nhận mất một số thông tin. Bản thân thuật toán biến đổi này cũng có độ phức tạp tính toán khá lớn.

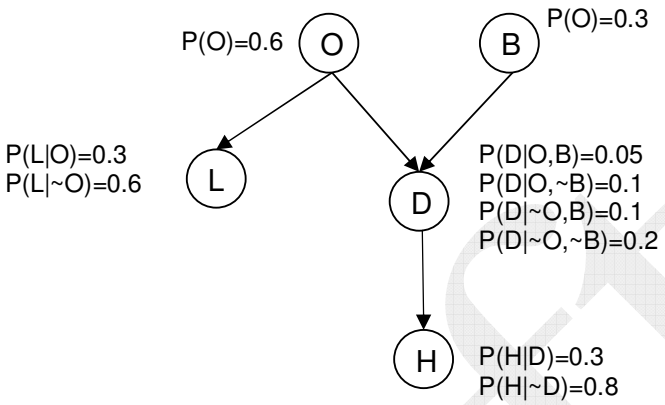
4.5.3.Suy diễn cho trường hợp riêng đơn giản

Phần này sẽ giới thiệu cách suy diễn cho trường hợp riêng đơn giản nhất, đó là trường hợp mạng liên kết đơn, suy diễn chỉ thực hiện cho các nút có liên kết trực tiếp với nhau. Có thể phân biệt hai loại suy diễn cho trường hợp đơn giản như vậy.

Trong trường hợp thứ nhất, biết giá trị một nút cha, yêu cầu tính xác suất quan sát thấy giá trị nút con. Suy diễn dạng này gọi là suy diễn *nhân quả* hoặc suy diễn từ trên xuống.

Trường hợp thứ hai ngược lại, tức là cho biết giá trị nút con và cần tính xác suất quan sát thấy giá trị nút cha. Trường hợp này được gọi là suy diễn *chẩn đoán* hoặc suy diễn dưới lên. Suy diễn chẩn đoán thường gặp trong những bài toán cần xác định hay chẩn đoán nguyên nhân của dấu hiệu nào đó.

Để tiện trình bày, các suy diễn sẽ được minh họa trên ví dụ đã sử dụng ở phần đầu bài.



Hình 4.6: Mạng Bayes

Suy diễn nhân quả hay suy diễn trên xuống

Trong suy diễn nhân quả, cho biết E, cần tính P(Q|E), trong đó E là một nút cha của Q.

Ví dụ: cần tính P(D|B)

$$\begin{aligned} P(D|B) &= P(D,O|B) + P(D,\sim O|B) \\ &= P(D|O,B)P(O|B) + P(D|\sim O,B)P(\sim O|B) \\ &= P(D|O,B)P(O) + P(D|\sim O,B)P(\sim O) \\ &= (.05)(.6) + (.1)(1 - .6) = 0.07 \end{aligned}$$

Ví dụ trên cho thấy, suy diễn nhân quả được thực hiện theo 3 bước.

- Bước 1: dòng 1. Trong bước này, xác suất Q cần tính được viết lại dưới dạng xác suất điều kiện của Q và cha của Q (không thuộc E) điều kiện theo E.
- Bước 2: dòng 2. Viết lại các xác suất đồng thời dưới dạng xác suất của Q khi biết các giá trị bố mẹ.
- Bước 3: dòng 3,4. Sử dụng các giá trị xác suất từ bảng xác suất điều kiện.

Suy diễn chẩn đoán hay suy diễn dưới lên

Đây là dạng suy diễn yêu cầu tính P(Q|E), trong đó Q là một nút cha của E.

Ví dụ: cần tính P(~B|~D)

Sử dụng quy tắc Bayes, ta viết lại giá trị cần tính như sau

$$P(\sim B|\sim D) = P(\sim D|\sim B)P(\sim B)/P(\sim D)$$

Tiếp theo, tính P(~D|~B) như cách suy diễn nhân quả ở trên

$$\begin{aligned} P(\sim D|\sim B) &= P(\sim D|O,\sim B)P(O) + P(\sim D|\sim O,\sim B)P(\sim O) \\ &= (.9)(.6) + (.8)(.4) \\ &= 0.86 \end{aligned}$$

Thay giá trị P(~D|~B) vừa tính, ta được:

P(~B|~D) = (.86)(.7)/P(~D) = .602/P(~D)

Để tính P(~D), có thể tính P(B|~D), sau đó dùng P(~B|~D)+P(B|~D) = 1

P(B|~D) = (.93)(.3)/P(~D) = .279/P(~D)

do đó:

0.602/P(~D) + 0.279/P(~D) = 1

suy ra:

P(~D) = 0.881

thay vào trên:

P(~B|~D) = 0.602 / 0.881 = 0.683

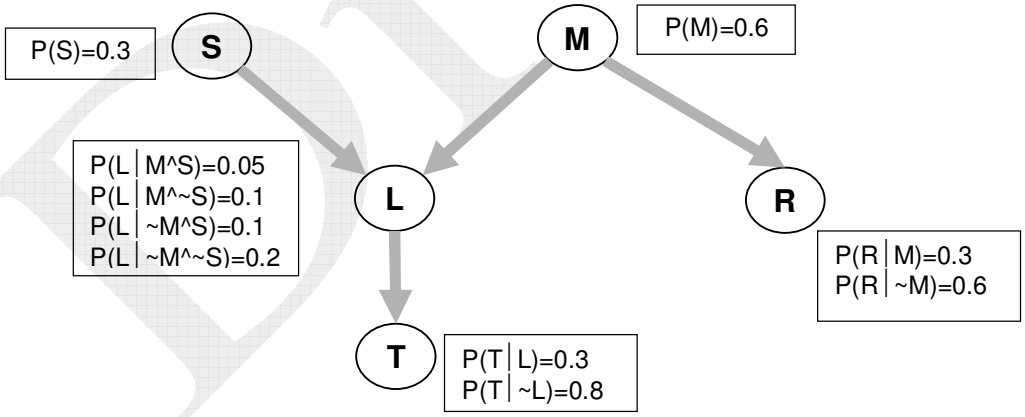
Trong trường hợp chung, suy diễn chẩn đoán có thể thực hiện qua hai bước như sau:

- Bước 1: Biến đổi về suy diễn nhân quả sử dụng quy tắc Bayes
- Bước 2: Thực hiện giống suy diễn nhân quả.

4.5.4.Suy diễn bằng phương pháp lấy mẫu

Trong trường hợp mạng Bayes có cấu trúc phức tạp, việc tính chính xác các xác suất điều kiện là không thực tế do độ phức tạp tính toán lớn. Một phương pháp suy diễn thường được sử dụng là việc tính toán xấp xỉ xác suất điều kiện bằng cách lấy mẫu. Thực chất quá trình này là sinh ra các bộ giá trị của các biến theo phân bố xác suất trên mạng, sau đó cộng lại và tính tần suất quan sát được, dùng giá trị tần suất này làm giá trị (xấp xỉ) của xác suất cần tính.

Việc lấy mẫu được minh họa qua ví dụ mạng Bayes trên hình dưới.



Hình 4.7. Ví dụ mạng Bayes

Lấy mẫu

Tại mỗi bước lấy mẫu, một bộ giá trị của các biến được xác định như sau.

- Chọn ngẫu nhiên giá trị của S theo xác suất P(S=true) = 0.3. Giả sử được giá trị S = false.
- Chọn ngẫu nhiên giá trị M theo xác suất P(M=true) = 0.6. Giả sử được giá trị M = true.

- Chọn ngẫu nhiên giá trị L theo xác suất P(L|S=false, M=true)=0.1. Giá trị của S và M đã được chọn ở trên. Giả sử được giá trị L là false.
- Chọn ngẫu nhiên R theo xác suất P(R|M=true) = 0.3. Giả sử được R=true.
- Chọn ngẫu nhiên T theo xác suất P(T|L=false) = 0.8. Giả sử được T=true.

Như vậy, sau một lần lấy mẫu, ta được bộ giá trị (~S, M, ~L, R, T).

Cách lấy mẫu tại mỗi bước là trước tiên lấy mẫu các nút không có nút cha. Sau đó lấy mẫu tới các nút con. Khi lấy mẫu nút con sẽ sử dụng phần tử trong bảng xác suất điều kiện tương ứng với các giá trị nút cha đã lựa chọn được trước đó. Mô tả thuật toán lấy mẫu thể hiện trên hình sau.

- Giả sử ta các biến/nút là $X_1, \dots X_n$.
- Giả sử Cha_Mẹ(X_i) là các nút cha của Xi và Cha_Mẹ(X_i) là tập con của $\{X_1, \dots, X_{i-1}\}$
- For i = 1 to n do
  1. Tìm Cha_Mẹ(X_i) (nếu có). Giả sử có n(i) nút cha của Xi. Gọi các nút là $X_{p(i,1)}, \dots, X_{p(i,n(i))}$)
  2. Giả sử giá trị các nút cha đã sinh ra trước đó là:$x_{p(i,1)}, \dots, x_{p(i,n(i))}$)
  3. Tìm trong bảng xác xuất giá trị xác suất: $P(X_i = \text{true} \mid X_{p(i,1)} = x_{p(i,1)} \dots)$
  4. Chọn X_i = true với xác xuất trên

Kết quả $x_1, \dots x_n$ là một mẫu sinh ra từ các biến $X_1, \dots, X_n$

Hình 4.8: Thuật toán lấy mẫu từ mạng Bayes

Tính xác suất điều kiện

Tiếp theo, giả sử cần tính $P(R = \text{True} \mid T = \text{True}, S = \text{False})$.

Trước hết, ta thực hiện lấy mẫu nhiều lần theo cách ở trên, mỗi bộ giá trị sinh ra được gọi là một mẫu. Nếu số lượng mẫu đủ lớn thì tần suất xuất hiện mỗi bộ giá trị sẽ xấp xỉ xác suất đồng thời của các giá trị đó. Xác suất đồng thời này sẽ được sử dụng để tính xác suất điều kiện như dưới đây.

Tính số lần xảy ra những sự kiện sau:

- N_c: số mẫu có T = True và S = False
- N_s: số mẫu có R = True, T = True và S = False
- N: tổng số mẫu

Nếu N đủ lớn:

- N_c/N: (xấp xỉ) xác suất P (T = True, S = False)
- N_s/N: (xấp xỉ) xác suất P (R = True , T = True , S = False)

– $P(R \mid T, \sim S) = \dfrac{\mathrm{P(R, T, \sim S)}}{\mathrm{P(T, \sim S)}} \approx \dfrac{N_s}{N_c}$

Phương pháp lấy mẫu

Từ ví dụ minh họa ở trên, có thể mô tả cách suy diễn xấp xỉ bằng cách lấy mẫu như sau. Giả sử cho mạng Bayes và cần tính $P(E_1 \mid E_2)$. Thực hiện các bước dưới đây:

- Lấy mẫu số lượng đủ lớn
- Tính số lượng:
  - N_c: số mẫu có E_2
  - N_s: số mẫu có E_1 và E_2
  - N: Tổng số mẫu
- Nếu N đủ lớn, ta có: $P(E_1 \mid E_2) = N_s / N_c$

Hình 4.9: Thuật toán suy diễn bằng cách lấy mẫu trên mạng Bayes

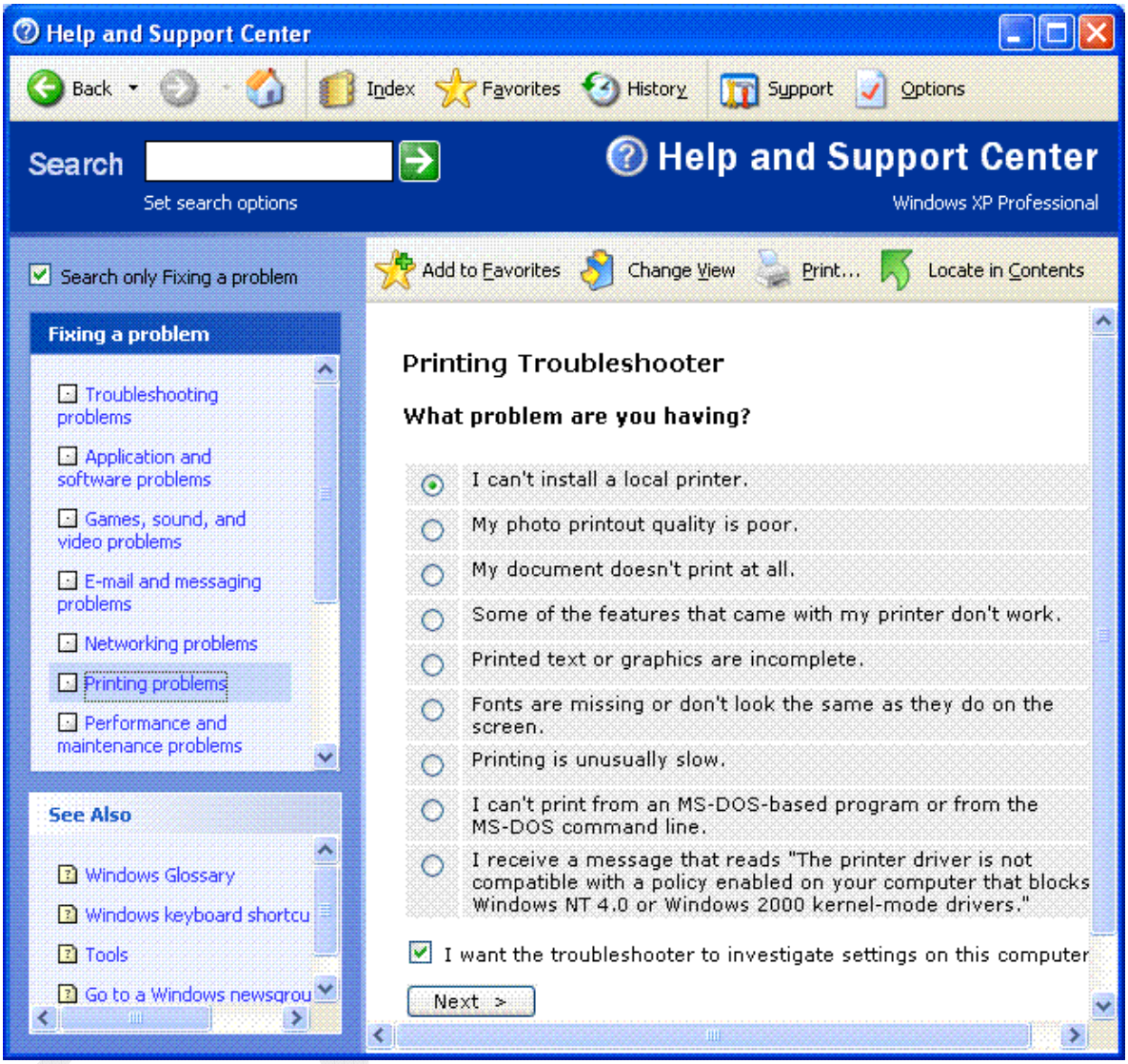
4.6. ỨNG DỤNG SUY DIỄN XÁC SUẤT

Có rất nhiều ứng dụng khác nhau của suy diễn xác suất, cụ thể là ứng dụng suy diễn trên mạng Bayes được sử dụng trong thực tế. Có thể kể ra một số ứng dụng tiêu biểu như chẩn đoán bệnh (hệ thống Pathfinder), hệ trợ giúp thông minh (Microsoft office assisstant), phương pháp xác dịnh chức năng gen và protein, hệ thống giúp phát hiện và khắc phục lỗi. Dưới đây ta sẽ xem xét một ví dụ ứng dụng mạng Bayes trong chẩn đoán và khắc phục lỗi (troubleshooting).

Chẩn đoán và khắc phục lỗi là công việc tương đối phức tạp, cần có sự trợ giúp thông minh. Một hệ trợ giúp thường gặp dạng này là hệ thống troubleshooting của hệ điều hành Windows. Trên hình dưới đây là một màn hình giao diện của hệ thống này. Khi gặp sự cố, chẳng hạn sự cố với máy in, người dùng khởi động hệ thống troubleshooting. Hệ thống đặt ra một số câu hỏi và yêu cầu trả lời. Tùy theo câu trả lời nhận được, hệ thống tìm cách xác định nguyên nhân và cách giải quyết. Câu trả lời càng cụ thể và càng chính xác thì nguyên nhân cũng được xác định càng cụ thể.

Mạng Bayes có thể được sử dụng cho phần suy diễn của hệ thống troubleshooting. Cụ thể, mạng Bayes cho phép biểu diễn quan hệ giữa ba dạng biến ngẫu nhiên: các dạng hỏng hóc F của thiết bị, hành động A cho phép khắc phục hỏng hóc, và câu hỏi Q cho phép thu thập thông tin về sự cố.

Xét ví dụ cụ thể sau: máy in in quá mờ. Có thể có rất nhiều hỏng hóc hay nguyên nhân dẫn tới in mờ. Để đơn giản, giả sử có bốn nguyên nhân sau: F1 – lỗi hộp mực, F2 – còn ít mực, F3 - luồng dữ liệu bị hỏng, F4 – chọn driver không đúng. Một số hành động cho phép khắc phục (một phần) nguyên nhân sau là: A1 – lắc ống mực, A2 – dùng ống mực khác, A3 – tắt và bật lại máy in.

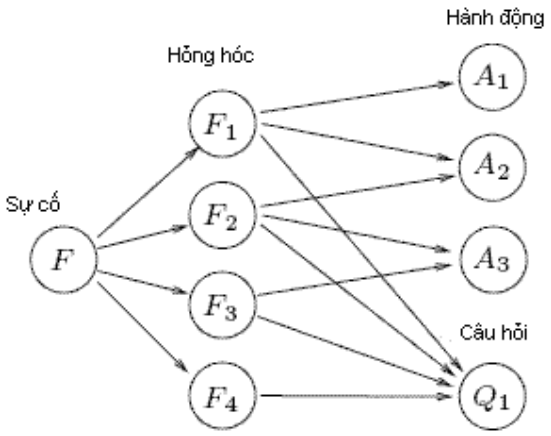


Hình 4.11. Màn hình hệ thống troubleshooting của Windows

Mỗi hành động cho phép khắc phục một hỏng hóc với một xác suất nào đó, ví dụ $P(A_2 = yes \mid F_1) = P(A_2 = yes \mid F_2) = 0{,}9$, tức là hành động dùng ống mực khác cho phép khắc phục lỗi hộp mực hay còn ít mực với xác suất 90%, trong khi $P(A_2 = yes \mid F_4) = 0$.

Trong quá trình khắc phục lỗi, hệ thống đặt câu hỏi cho người dùng để có thêm thông tin về hỏng hóc. Ví dụ, nếu người dùng trả lời "không" cho câu hỏi "Trang in thử có bị mờ không?" thì hệ thống có thể loại bỏ nguyên nhân hỏng hóc liên quan tới hộp mực. Hệ thống sử dụng các bảng xác suất điều kiện $P(Q_i = yes \mid F_j)$ cho phần suy diễn này.

Giữa các biến F, A, và Q có một số quan hệ độc lập xác suất. Cụ thể, hành động A và câu hỏi Q độc lập với nhau khi đã biết F. Ngoài ra, có thể giả thiết tại mỗi thời điểm chỉ xảy ra một hỏng hóc duy nhất. Quan hệ độc lập xác suất giữa các biến được mô hình hóa bằng mạng Bayes trên hình 4.12.



Hình 4.12. Mạng Bayes dùng cho khắc phục sự cố

Tùy thông tin có được, hệ thống có thể thực hiện suy diễn để tính xác suất lựa chọn một hành động cụ thể nào đó.

CHƯƠNG 5: HỌC MÁY

5.1. KHÁI NIỆM HỌC MÁY

5.1.1. Học máy là gì

Trong những chương trước ta đã xem xét một số kỹ thuật giải quyết vấn đề và suy diễn. Đặc điểm chung của những kỹ thuật này là phương pháp giải quyết vấn đề cùng với tri thức được cung cấp cho chương trình máy tính từ trước và chương trình không thể thay đổi hoặc cái thiện hoạt động của mình dựa trên kinh nghiệm thu được trong quá trình làm việc sau đó. Trong chương này, ta sẽ xem xét các kỹ thuật *học máy* (machine learning), là kỹ thuật cho phép giải quyết vấn để hoặc ra quyết định dựa trên dữ liệu và kinh nghiệm.

Học máy (machine learning) là khả năng của chương trình máy tính sử dụng kinh nghiệm, quan sát, hoặc dữ liệu trong quá khứ để cải thiện công việc của mình trong tương lai. Chẳng hạn, máy tính có thể học cách dự đoán dựa trên các ví dụ, hay học cách tạo ra các hành vi phù hợp dựa trên quan sát trong quá khứ.

Xét một số ví dụ sau. Ví dụ thứ nhất là học cách đánh cờ. Chương trình có thể quan sát các ván cờ cùng với kết quả để cải thiện khả năng chơi cờ để tăng số ván thắng trong tương lai. Trong trường hợp này, kinh nghiệm là các ván cờ trong quá khứ (có thể là ván cờ chương trình tự chơi với chính mình), được sử dụng để học cách làm tốt hơn công việc chơi cờ với tiêu chí đánh giá là số ván thắng.

Ví dụ thứ hai là học nhận dạng các ký tự. Chương trình được cung cấp dữ liệu dưới dạng ảnh chụp các ký tự (chữ cái) cùng mã ASCII của ký tự đó. Quá trình học cho phép chương trình tăng độ chính xác khi phân biệt ảnh chụp các ký tự sau đó.

Học máy là một nhánh nghiên cứu rất quan trọng của trí tuệ nhân tạo với khá nhiều ứng dụng thành công trong thực tế. Lý do học máy được quan tâm nhiều là do rất khó xây dựng hệ thống thông minh có thể thực hiện các công việc liên quan đến trí tuệ như thị giác máy, xử lý ngôn ngữ tự nhiên mà không sử dụng tới kinh nghiệm và quá trình học. Ngoài ra, bản thân khả năng học là một hoạt động trí tuệ quan trọng của con người, do vậy học tự động hay học máy luôn thu hút được sự quan tâm khi xây dựng hệ thống thông minh.

5.1.2. Ứng dụng của học máy

Có rất nhiều ứng dụng thực tế khác nhau của học máy. Trong chương này, ta sẽ xem xét chủ yếu bài toán học phân loại tự động, trong đó các đối tượng được phân chia tự động thành một số loại, hoặc bài toán hồi quy (regression) trong đó thuật toán học dự đoán một giá trị có dạng số thực. Sau đây là một số ví dụ ứng dụng học máy loại này:

- Nhận dạng ký tự: phân loại hình chụp ký tự thành các loại, mỗi loại ứng với một ký tự tương ứng.
- Phát hiện và nhận dạng mặt người: phát hiện vùng có chứa mặt người trong ảnh, xác định đó là mặt người nào trong số những người đã có ảnh trước đó, tức là phân chia ảnh thành những loại tương ứng với những người khác nhau.

- Lọc thư rác, phân loại văn bản: dựa trên nội dung thư điện tử, chia thư thành loại "thư rác" hay "thư bình thường"; hoặc phân chia tin tức thành các thể loại khác nhau như "xã hội", "kinh tế", "thể thao".v.v.
- Chẩn đoán y tế: học cách dự đoán người bệnh có mắc hay không mắc một số bệnh nào đó dựa trên triệu chứng quan sát được.
- Phân loại khách hàng và dự đoán sở thích: sắp xếp khách hàng vào một số loại, từ đây dự đoán sở thích tiêu dùng của khách hàng.
- Dự đoán chỉ số thí trường: căn cứ giá trị một số tham số hiện thời và trong lịch sử, đưa ra dự đoán, chẳng hạn dự đoán giá chứng khoán, giá vàng.v.v.

Ngoài những ứng dụng có dạng phân loại hoặc hồi quy một cách tường minh ở trên, học máy có thể dùng trong rất nhiều ứng dụng đòi hỏi ra quyết định hoặc hành động một cách thông minh. Ví dụ đã có những ứng dụng học cách lái xe tự động dựa trên ảnh chụp trước xe do camera quan sát và hành động lái xe làm mẫu của người trước đó. Thực chất, đây có thể coi như bài toán phân loại trong đó tùy thuộc ảnh đầu vào cần xác định hành động đầu ra thuộc một trong số những loại nào đó như " quay vô lăng sang trái 10 độ".v.v. Có thể tìm thấy rất nhiều ứng dụng thông minh có sử dụng học máy để tạo cơ chế ra quyết định như vậy.

5.1.3. Một số khái niệm

Sau đây là một số khái niệm được sử dụng khi trình bày về học máy.

Mẫu hay *ví dụ* là tên gọi đối tượng cần phân loại. Chẳng hạn, khi lọc thư rác, mỗi thư được gọi là một mẫu hay một ví dụ.

Mẫu thường được mô tả bằng một tập các *thuộc tính*, còn được gọi là *đặc trưng* hay *biến*. Ví dụ, trong bài toán chẩn đoán bệnh, thuộc tính là những triệu chứng của người bệnh và các tham số khác: cân nặng, huyết áp, v.v.

Nhãn phân loại thể hiện loại của đối tượng mà ta cần dự đoán. Đối với trường hợp phân loại thư rác, nhãn phân loại có thể là "rác" hay "bình thường". Trong giai đoạn học hay còn gọi là giai đoạn huấn luyện, thuật toán học được cung cấp cả nhãn phân loại của mẫu, trong giai đoạn dự đoán, thuật toán chỉ nhận được các mẫu không nhãn và cần xác định nhãn cho những mẫu này.

Kết quả học thường được thể hiện dưới dạng một ánh xạ từ mẫu sang nhãn phân loại. Ánh xạ này được thể hiện dưới dạng một hàm gọi là *hàm đích* (target function) có dạng $f : X \rightarrow C$, trong đó X là không gian các ví dụ và C là tập các nhãn phân loại khác nhau.

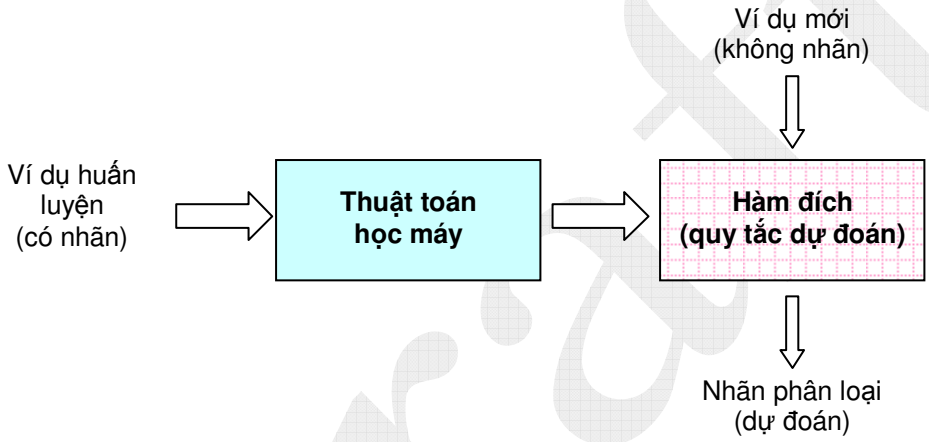
5.1.4. Các dạng học máy

Khi thiết kế và xây dựng hệ thống học máy cần quan tâm tới những yếu tố sau.

- Thứ nhất, kinh nghiệm hoặc dữ liệu cho học máy được cho dưới dạng nào?
- Thứ hai, lựa chọn biểu diễn cho hàm đích ra sao? Hàm đích có thể biểu diễn dưới dạng hàm đại số thông thường nhưng cũng có thể biểu diễn dưới những dạng khác như dạng cây, dạng mạng nơ ron, công thức xác suất .v.v.

Việc sử dụng những dạng kinh nghiệm và dạng biểu diễn khác nhau dẫn tới những dạng học máy khác nhau.

Trường hợp đơn giản nhất là khi kinh nghiệm được cho một cách tường minh dưới dạng đầu vào và đầu ra của hàm đích, ví dụ, cho trước tập các mẫu cùng nhãn phân loại tương ứng. Học máy khi đó được gọi là *học có giám sát* (supervised learning) để thể hiện việc thuật toán nhận được chỉ dẫn trực tiếp về lời giải cho từng trường hợp. Học có giám sát bao gồm *phân loại* (classification) và *hồi quy* (regression). Phân loại là dạng học có giám sát với hàm đích nhận giá trị rời rạc và hồi quy là học có giám sát với hàm đích nhận giá trị liên tục. Sơ đồ của một hệ thống học máy điển hình trong trường hợp học có giám sát (phân loại) được thể hiện trên hình 5.1.



Hình 5.1. Cấu trúc một hệ thống học máy tiểu biểu cho trường hợp phân loại

Trong trường hợp kinh nghiệm chỉ gồm các mẫu và không có nhãn hoặc giá trị hàm đích đi kèm ta nói rằng đó là *học không giám sát* (unsupervised learning). Ví dụ, chỉ bằng quan sát thông thường về chiều cao của mọi người, dần dần ta học được khái niệm "người cao" và "người thấp". Hai dạng học không giám sát phổ biến nhất là *phân cụm* (clustering) và *học luật kết hợp* (association rule learning). Trong trường hợp phân cụm, các đối được phân chia thành một số nhóm sao cho mỗi nhóm gồm những đối tượng giống nhau và khác đối tượng ở nhóm khác. Học luật kết hợp là cách phát hiện những đối tượng hoặc giá trị thuộc tính thường xuất hiện cùng nhau, ví dụ những mặt hàng thường xuyên được mua cũng nhau (bánh mì và bơ).

Dạng học máy cuối cùng là *học khuyến khích* (reinforcement learning)[4]. Đối với dạng học này, kinh nghiệm không được cho trực tiếp dưới dạng đầu vào/đầu ra. Thay vào đó, hệ thống nhận được một giá trị khuyến khích (reward) là kết quả cho một chuỗi hành động nào đó. Thuật

[4] Khái niệm "reinforcement" được dùng trong tâm lý học với nghĩa khuyến khích, kích thích, thưởng. Chúng tôi tạm dịch sang tiếng Việt là khuyến khích nhưng cũng có thể dịch là thưởng hoặc có tài liệu dịch reinforcement learning là "học tăng cường"

toán cần học cách hành động để cực đại hóa giá trị khuyến khích. Ví dụ của học khuyến khích là học đánh cờ, trong đó hệ thống không được chỉ dẫn nước đi nào là hợp lý cho từng tình huống mà chỉ biết kết quả toàn ván cờ. Như vậy, các chỉ dẫn về nước đi được cho một cách gián tiến và có độ trễ dưới dạng giá trị thưởng.

5.2. HỌC CÂY QUYẾT ĐỊNH

Học cây quyết định là một trong phương pháp học máy tiêu biểu có nhiều ứng dụng trong phân loại và dự đoán. Mặc dù độ chính xác của phương pháp này không thật cao so với những phương pháp được nghiên cứu gần đây, học cây quyết định vẫn có nhiều ưu điểm như đơn giản, dễ lập trình, và cho phép biểu diễn hàm phân loại dưới dạng dễ hiểu, dễ giải thích cho con người. Phương pháp này thường được dùng như phương pháp mở đầu để minh họa cho kỹ thuật học bộ phân loại từ dữ liệu.

Phương pháp học cây quyết định được sử dụng cho việc học các hàm phân loại từ dữ liệu huấn luyện, trong đó cây quyết định được sử dụng làm biểu diễn xấp xỉ của hàm phân loại, tức là hàm có đầu ra là các giá trị rời rạc. Như đã nói ở trên, phương pháp học này thuộc loại học có giám sát.

Phần này sẽ giúp người đọc làm quen với khái niệm cây quyết định, đồng thời giới thiệu một số thuật toán học cây quyết định bao gồm ID3 và C4.5.

5.2.1. Khái niệm cây quyết định

Cây quyết định là một cấu trúc ra quyết định có dạng cây. Cây quyết định nhận đầu vào là một bộ giá trị *thuộc tính* mô tả một đối tượng hay một tình huống và trả về một giá trị rời rạc. Mỗi bộ thuộc tính đầu vào được gọi là một *mẫu* hay một *ví dụ*, đầu ra gọi là *loại* hay *nhãn phân loại*. Thuộc tính đầu vào còn được gọi là đặc trưng và có thể nhận giá trị rời rạc hoặc liên tục. Để cho đơn giản, trước tiên ta sẽ xem xét thuộc tính rời rạc, sau đó sẽ mở rộng cho trường hợp thuộc tính nhận giá trị liên tục.

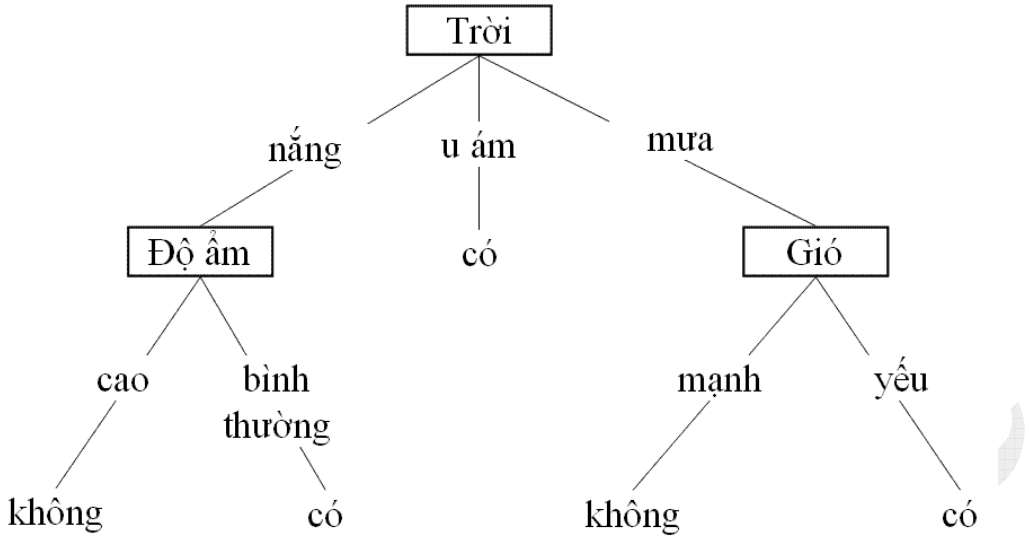
Trong các trình bày tiếp theo, tập thuộc tính đầu vào được cho dưới dạng véc tơ $\mathbf{x}$, nhãn phân loại đầu ra được ký hiệu là y, cây quyết định là hàm $f(\mathbf{x})$ trả lại giá trị y.

Cây quyết định được biểu diễn dưới dạng một cấu trúc cây (xem ví dụ trên hình 5.2). Mỗi nút trung gian, tức là nút không phải nút lá, tương ứng với phép kiểm tra một thuộc tính. Mỗi nhánh phía dưới của nút đó tương ứng với một giá trị của thuộc tính hay một kết quả của phép thử. Khác với nút trung gian, nút lá không chứa thuộc tính mà chứa nhãn phân loại.

Để xác định nhãn phân loại cho một ví dụ nào đó, ta cho ví dụ chuyển động từ gốc cây về phía nút lá. Tại mỗi nút, thuộc tính tương ứng với nút được kiểm tra, tùy theo giá trị của thuộc tính đó mà ví dụ được chuyển xuống nhánh tương ứng bên dưới. Quá trình này lặp lại cho đến khi ví dụ tới được nút lá và được nhận nhãn phân loại là nhãn của nút lá tương ứng.

Xét ví dụ cây quyết định trên hình 5.2. Cây quyết định cho phép xác định (phân loại) các buổi sáng thành có (phù hợp) và không (phù hợp) cho việc chơi tennis tùy theo thời tiết trong

ngày đó. Thời tiết mỗi ngày được mô tả thông qua bốn thuộc tính: Trời, Độ ẩm, Nhiệt độ, Gió. Dữ liệu thời tiết cho một số ngày được cho trong bảng 5.1.



Hình 5.2. Một ví dụ cây quyết định cho bài toán "Chơi tennis". Nút lá chứa nhãn phân loại "có chơi" hoặc "không chơi". Nút trung gian chứa thuộc tính thời tiết.

Giả sử ta có ví dụ < Trời = nắng, Nhiệt độ = cao, Gió = mạnh, Độ ẩm = cao>. Ví dụ sẽ được cây quyết định xếp xuống nút ngoài cùng bên trái và do vậy được xác định là "không chơi".

Có thể thấy cách biểu diễn hàm quyết định dưới dạng cây rất trực quan, dễ hiểu, dễ giải thích lý do ra quyết định về nhãn cho một ví dụ cụ thể nào đó.

Biểu diễn tương đương dưới dạng biểu thức lô gic

Cây quyết định có thể biểu diễn tương đương dưới dạng các quy tắc hay biểu thức lô gic.

Cây_quyết_định(**x**) $\Leftrightarrow$ (P_1(**x**) v P_2(**x**) v… v P_n(**x**))

trong đó mỗi P_i(**x**) là hội các phép thử thuộc tính theo đường đi từ gôc tới nút lá có giá trị dương (true). Cụ thể, cây quyết định trên hình 4.1 có thể biểu diễn tương đương dưới dạng:

(Trời = nắng ^ Độ ẩm = bình_thường)

∨ (Trời = u_ám)

∨ (Trời = mưa ^ Gió = yếu)

5.2.2. Thuật toán học cây quyết định

Trước khi sử dụng cây quyết định, ta cần xây dựng hay "học" cây quyết định từ dữ liệu huấn luyện. Có nhiều thuật toán khác nhau được đề xuất và sử dụng để học cây quyết định từ dữ liệu, trong đó đa số dựa trên nguyên tắc chung là xây dựng cây theo kiểu tìm kiếm tham lam từ cây đơn giản tới cây phức tạp hơn. Phần này sẽ giới thiệu thuật toán học cây ID3, một thuật toán đơn giản nhưng có tính đại diện cho cách xây dựng cây như vậy.

Dữ liệu huấn luyện

Dữ liệu huấn luyện được cho dưới dạng n mẫu hay n ví dụ huấn luyện, mỗi ví dụ có dạng $(\mathbf{x}_i, y_i)$, trong đó $\mathbf{x}_i$ là véc tơ các thuộc tính và y_i là giá trị nhãn phân loại. Để trình bày về thuật toán học cây quyết định, ta sẽ sử dụng bộ dữ liệu huấn luyện cho trong bảng 5.1 với 14 ví dụ tương ứng với 14 dòng. Cột đầu tiên trong bảng chứa số thứ tự và không tham gia vào cây quyết định. Bốn cột tiếp theo chứa giá trị bốn thuộc tính. Cột ngoài cùng bên phải chứa nhãn phân loại. Đối với dữ liệu đang xét, nhãn phân loại là nhãn nhị phân, có thể nhận một trong hai giá trị "có" hoặc "không".

Table 5.1. Bộ dữ liệu huấn luyện cho bài toán phân loại "Chơi tennis".

Ngày	**Trời**	**Nhiệt độ**	**Độ ẩm**	**Gió**	**Chơi tennis**
D1	nắng	cao	cao	yếu	không
D2	nắng	cao	cao	mạnh	không
D3	u ám	cao	cao	yếu	có
D4	mưa	trung bình	cao	yếu	có
D5	mưa	thấp	bình thường	yếu	có
D6	mưa	thấp	bình thường	mạnh	không
D7	u ám	thấp	bình thường	mạnh	có
D8	nắng	trung bình	cao	yếu	không
D9	nắng	thấp	bình thường	yếu	có
D10	mưa	trung bình	bình thường	yếu	có
D11	nắng	trung bình	bình thường	mạnh	có
D12	u ám	trung bình	cao	mạnh	có
D13	u ám	cao	bình thường	yếu	có
D14	mưa	trung bình	cao	mạnh	không

Thuật toán học cây

Nhiệm vụ của thuật toán học là xây dựng cây quyết định phù hợp với tập dữ liệu huấn

luyện, tức là cây quyết định có đầu ra giống (nhiều nhất) với nhãn phân loại cho trong tập mẫu. Trong trường hợp số thuộc tính nhỏ, việc xây dựng cây quyết định như vậy có thể thực hiện bằng cách liệt kê tất các cây quyết định hợp lệ và kiểm tra để chọn ra cây phù hợp với dữ liệu. Với số lượng thuộc tính lớn, số cây quyết định như vậy là rất lớn và không thể tìm kiếm theo kiểu vét cạn như vậy. Do đó, thuật toán học cây thường dựa trên nguyên tắc tham lam, xây dựng dần các nút từ trên xuống.

Quá trình xây dựng cây: Để bắt đầu, thuật toán học lựa chọn thuộc tính cho nút gốc. Thuộc tính được lựa chọn là thuộc tính cho phép *phân chia tốt nhất* các ví dụ thành những tập con, sao cho mỗi tập con càng đồng nhất càng tốt. Ở đây, đồng nhất được hiểu là các ví dụ có cùng nhãn phân loại. Sau khi lựa chọn được thuộc tính cho nút gốc, tập dữ liệu ban đầu sẽ được chia xuống các nhánh con do kết quả phép kiểm tra thuộc tính ở gốc. Với mỗi tập con dữ liệu, ta lại có một bài toán học cây dữ liệu mới và do vậy có thể lặp lại thủ tục ở trên với ít dữ liệu hơn và bớt đi một thuộc tính đã được sử dụng ở gốc.

Quá trình xây dựng cây quyết định được lặp đệ quy như vậy cho tới khi xẩy ra những tình huống sau:

- Sau khi phân chia tại một nút, tập dữ liệu con chỉ chứa các mẫu có cùng nhãn phân loại (chẳng hạn cùng dương hoặc cùng âm). Trong trường hợp này ta dừng quá trình phân chia ở đây, tạo một nút là và gán cho nút nhãn phân loại trùng với nhãn của các ví dụ tại nút đó. Trong ví dụ trên hình 4.1., nhánh giữa của nút gốc bao gồm các mẫu có nhãn "có" tạo thành nút lá.
- Tất cả các thuộc tính đã được sử dụng ở phía trên, trong khi tập dữ liệu con còn chứa cả nhãn dương và nhãn âm. Đây là trường hợp các ví dụ có cùng giá trị thuộc tính nhưng lại khác nhãn phân loại và xẩy ra do dữ liệu huấn luyện có chứa nhiễu hoặc do các thuộc tính hiện có không cung cấp đủ thông tin để xác định đúng nhãn phân loại. Trong trường hợp này, thuật toán sẽ chọn nhãn chiếm đa số trong tập con để gán cho nút.

Thuật toán học cây quyết định được cho trên hình 5.3.

- Khởi đầu: nút hiện thời là nút gốc chứa toàn bộ tập dữ liệu huấn luyện
- Tại nút hiện thời n, lựa chọn thuộc tính:
  - Chưa được sử dụng ở nút tổ tiên (tức là nút nằm trên đường đi từ gốc tới nút hiện thời)
  - Cho phép phân chia tập dữ liệu hiện thời thành các tập con **một cách tốt nhất**
  - Với mỗi giá trị thuộc tính được chọn thêm một nút con bên dưới
  - Chia các ví dụ ở nút hiên thời về các nút con theo giá trị thuộc tính được chọn
- Lặp (đệ quy) cho tới khi:

- Tất cả các thuộc tính đã được sử dụng ở các nút phía trên, hoặc
- Tất cả ví dụ tại nút hiện thời có cùng nhãn phân loại
- Nhãn của nút được lấy theo đa số nhãn của ví dụ tại nút hiện thời

Hình 5.3. Thuật toán xây dựng cây quyết định từ dữ liệu huấn luyện

Lựa chọn thuộc tính tốt nhất

Một điểm quan trọng trong thuật toán xây dựng cây quyết định là lựa chọn thuộc tính tốt nhất tại mỗi nút. Trong trường hợp lý tưởng, thuộc tính lựa chọn là thuộc tính cho phép chia tập dữ liệu thành các tập con có cùng một nhãn, và do vậy chỉ cần một phép kiểm tra thuộc tính khi phân loại. Trong trường hợp nói chung, thuộc tính lựa chọn cần cho phép tạo ta những tập con có độ đồng nhất cao nhất. Yêu cầu đặt ra là cần có cách đo độ đồng nhất của tập dữ liệu và mức tăng độ đồng nhất khi sử dụng một thuộc tính nào đó.

Thuật toán xây dựng cây ID3 sử dụng *entropy* làm mức đo độ đồng nhất của tập dữ liệu. Trên cơ sở entropy, thuật toán tính *độ tăng thông tin* như mức tăng độ đồng nhất, từ đây xác định thuộc tính tốt nhất tại mỗi nút.

Trong trường hợp chỉ có hai nhãn phân loại, ký hiệu là + và -, entropy H(S) của tập dữ liệu S được tính như sau:

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

trong đó p+ và p- là xác suất quan sát thấy nhãn phân loại + và -, được tính bằng tần suất quan sát thấy + và – trong tập dữ liệu. Trong tập dữ liệu trên bảng 4.1, với 9 nhãn dương và 5 nhãn âm, ký hiệu [9+, 5-], ta có:

$$H([9+,5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$$

Có thể nhận thấy, trong trường hợp nhãn nhị phân, entropy đạt giá trị tối đa bằng 1 khi xác suất hai nhãn bằng nhau và bằng 0.5, entropy đạt giá trị nhỏ nhất bằng 0 khi xác suất một nhãn là 1 và nhãn còn lại là 0. Như vậy, entropy càng nhỏ thì tập đối tượng càng đồng nhất.

Trong trường hợp tổng quát với C nhãn phân loại có xác suất lần lượt là p_1, p_2, …,p_C. entropy được tính như sau:

$$H(S) = -\sum_{i=1}^{C} p_i \log_2 p_i$$

Giá trị cực đại của entropy khi đó sẽ bằng $\log_2 C$ khi các nhãn có xác suất như nhau và giá trị nhỏ nhất của entropy bằng 0 khi tất cả đối tượng có chung một nhãn.

Sử dụng entropy như độ đo mức đồng nhất của tập mẫu, ta có thể đánh giá độ tốt của thuộc tính bằng cách so sánh entropy trước và sau khi tập mẫu được phân chia thành tập con theo giá trị của thuộc tính.

Độ tăng thông tin (Information Gain), ký hiệu IG, là chỉ số đánh giá độ tốt của thuộc tính trong việc phân chia tập dữ liệu thành những tập con đồng nhất. IG được tính dựa trên entropy theo công thức sau:

$$IG(S,A) = H(S) - \sum_{v \in values(A)} \frac{|S_V|}{|S|} H(S_v)$$

trong đó:

S là tập dữ liệu ở nút hiện tại

A là thuộc tính

values(A) là tập các giá trị của thuộc tính A.

S_v là tập các mẫu có giá trị thuộc tính A bằng v.

$|S|$ và $|S_v|$ là lực lượng của các tập hợp tương ứng.

Về bản chất, IG là độ chênh lệch giữa entropy của tập S và tổng entropy của các tập con S_v được tạo ra do phân chia S bằng cách sử dụng thuộc tính A. Do các tập con có thể có kích thước không bằng nhau nên entropy của tập con được nhân với một trọng số $|S_v| / |S|$, tức là tập con có kích thước lớn hơn sẽ đóng góp nhiều hơn vào tổng entropy.

Giá trị của IG được sử dụng để lựa chọn thuộc tính tốt nhất tại mỗi nút. Thuộc tính được lựa chọn là thuộc tính có giá trị IG lớn nhất.

Ví dụ minh họa

Xác định thuộc tính tốt nhất tại nút gốc cho dữ liệu trong bảng 4.1 bằng cách tính IG cho các thuộc tính.

Với thuộc tính Gió:

values(Gió) = {yếu, mạnh}

S = [9+, 5-], $H(S) = 0.94$

$S_{yếu}$ = [6+, 2-] , $H(S_{yếu}) = 0.811$

$S_{mạnh}$ = [3+, 3-], $H(S_{mạnh}) = 1$

$IG(S, Gió) = H(S) - (8/14)\, H(S_{yếu}) - (6/14)\, H(S_{mạnh})$

$= 0.94 - (8/14) * 0.811 - (6/14) * 1$

$= 0.048$

Tính tương tự với ba thuộc tính còn lại, ta được:

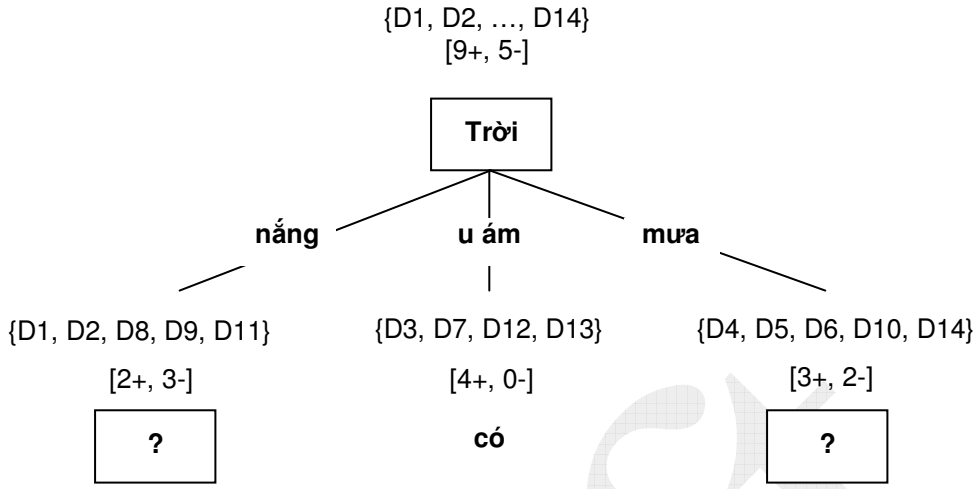
IG (S, Trời) = 0.246

IG (S, Độ ẩm) = 0.151

IG (S, Gió) = 0.048

IG (S, Nhiệt độ) = 0.029

Như vậy, thuộc tính tốt nhất là Trời được sử dụng cho nút gốc. Sau khi chọn nút gốc là Trời, ta được các bộ dữ liệu con ở ba nhánh tương ứng ba giá trị của Trời như trên hình 5.4. Để cho tiện số thứ tự các ví dụ được cho ngay tại các nút.



Hình 5.4. Xây dựng cây quyết định

Đối với nhánh giữa, toàn bộ mẫu có nhãn dương, do vậy quá trình học cây cho nhánh này dừng lại, thuật toán tạo nút lá với nhãn "có". Đối với nhánh bên trái và bên phải, quá trình học cây được tiếp tục với tập dữ liệu con tại nhánh đó. Dưới đây là minh họa việc tính IG và chọn thuộc tính cho nút tiếp theo bên trái.

IG ($S_{nắng}$, Độ ẩm) = 0.97 –(3/5)*0 – (2/5)*0 = 0.97

IG ($S_{nắng}$, Nhiệt độ) = 0.97 – (2/5)*0 –(2/5)*1 – (1/5)*0 = 0.57

IG ($S_{nắng}$, Gió) = 0.97 – (2/5)*1 – (3/5)*0.918 = 0.019

Thuộc tính Độ ẩm có IG lớn nhất và được chọn cho nút này. Kết quả học cây đầy đủ được thể hiện trên hình 5.2.

5.2.3. Các đặc điểm thuật toán học cây quyết định

Thuật toán học cây quyết định ID3 ở trên có một số đặc điểm sau:

- ID3 là thuật toán tìm kiếm cây quyết định phù hợp với dữ liệu huấn luyện.
- Đây là phương pháp tìm kiếm theo kiểu tham lam, từ trên xuống, bắt đầu từ cây rỗng. Hàm đánh giá là độ tăng thông tin. Tính chất tham lam của thuật toán thể hiện ở chỗ tại mỗi nút, thuộc tính được chọn là thuộc tính có hàm mục tiêu lớn nhất, thuật toán không nhìn xa hơn nút hiện tại khi quyết định chọn thuộc tính. Không gian tìm kiếm là đầy đủ, nghĩa là theo cách xây dựng cây như vậy, thuật toán có thể di chuyển tới mọi cây hợp lệ.
- ID3 có khuynh hướng lựa chọn cây quyết định đơn giản tức là cây có ít nút, trong đó những nút tương ứng với thuộc tính có độ tăng thông tin lớn được xếp ở gần gốc hơn.

Lưu ý: Ở trên vừa nhắc tới "khuynh hướng" (bias) của thuật toán. Trong học máy, từ khuynh hướng dùng để chỉ tính chất thuật toán ưu tiên một phương án này hơn một phương án

khác trong khi cả hai phương án đều thỏa mãn yêu cầu đặt ra. Trong trường hợp cây quyết định, nếu hai cây cùng phù hợp với dữ liệu thì thuật toán có khuynh hướng lựa chọn cây ít nút hơn.

Việc lựa chọn cây quyết định đơn giản phù hợp với một nguyên tắc trong triết học được gọi là *Occam's razor* (Occam là tên một nhà triết học) theo đó nếu có nhiều giả thiết cho phép giải thích một số quan sát nào đó thì ưu tiên chọn giả thiết đơn giản hơn.

5.2.4. Vấn đề quá vừa dữ liệu

Quá vừa dữ liệu (data overfitting hay đơn giản là overfitting) là một vấn đề thường gặp trong học máy và có ảnh hưởng nhiều tới độ chính xác của các kỹ thuật học máy.

Trong khi xây dựng cây quyết định (hay bộ phân loại nói chung), thuật toán học máy thường cố gắng để cây phù hợp với dữ liệu, tức là phân loại đúng các mẫu huấn luyện, ở mức tối đa. Tuy nhiên, mục đích học cây quyết định không phải để phân loại dữ liệu mẫu, mà để phân loại dữ liệu nói chung, tức là dữ liệu mà thuật toán chưa biết trong thời gian học. Có thể xảy ra tình huống cây quyết định có độ chính xác tốt trên dữ liệu huấn luyện nhưng lại cho độ chính xác không tốt trên dữ liệu nói chung. Khi đó ta nói cây quyết định quá vừa với dữ liệu huấn luyện.

Ta nói rằng cây quyết định t *quá vừa dữ liệu huấn luyện nếu tồn tại cây quyết định* t' *sao cho* t *chính xác hơn* t' *trên dữ liệu huấn luyện nhưng kém chính xác hơn* t' *trên dữ liệu nói chung*.

Lý do bộ phân loại làm việc tốt trên dữ liệu huấn luyện nhưng không tốt trên dữ liệu nói chung là do các mẫu huấn luyện thường không đủ và không mang tính đại diện cho phân bố của dữ liệu nói chung. Chẳng hạn, khi số lượng mẫu tại nút lá ít, việc xuất hiện tương quan giữa giá trị thuộc tính và nhãn phân loại có thể xảy ra do trùng hợp ngẫu nhiên dẫn tới một số thuộc tính phân chia các mẫu rất tốt trong khi thực tế thuộc tính không có quan hệ với nhãn phân loại.

Một lý do khác dẫn tới quá vừa dữ liệu là do dữ liệu huấn luyện có nhiễu trong khi thuật toán cố gắng xây dựng cây để phân loại đúng các ví dụ nhiễu này.

Đối với cây quyết định, thuật toán ID3 mô tả ở trên phát triển nhánh cây rất sâu cho đến khi phân loại đúng toàn bộ mẫu, hoặc khi đã hết thuộc tính. Nghiên cứu cho thấy, việc phát triển cây phức tạp với nhiều nút là nguyên nhân chính dẫn tới quá vừa dữ liệu. Từ đây, có hai nhóm giải pháp chính để hạn chế quá vừa dữ liệu cho thuật toán học cây quyết định:

- Giải pháp dừng việc dựng cây quyết định sớm, trước khi cây đủ phức tạp để phân loại đúng mẫu huấn luyện.
- Giải pháp xây dựng cây đầy đủ, sau đó thực hiện "tỉa" cây để có cây đơn giản hơn.

Trong hai nhóm trên, nhóm giải pháp thứ hai được sử dụng thành công hơn trên thực tế và do vậy sẽ được trình bày tiếp theo.

Chống quá vừa dữ liệu bằng cách tỉa cây.

Trước tiên, để thực hiện tỉa cây, cần có cách xác định độ chính xác phân loại của cây. Do mục đích của cây là phân loại những mẫu chưa biết trong quá trình huấn luyện nên cách tính độ chính xác thông dụng nhất là sử dụng tập huấn luyện và tập kiểm tra riêng như sau:

- Toàn bộ mẫu được chia thành hai tập: tập thứ nhất gọi là tập huấn luyện, tập thứ hai gọi là tập kiểm tra, thường với tỷ lệ 2:1.
- Sử dụng tập huấn luyện để xây dựng cây, sử dụng tập kiểm tra để tính độ chính xác của cây, tức là xác định xem kết quả phân loại của cây phù hợp đến mức nào với mẫu trong tập kiểm tra.

Trong trường hợp ít dữ liệu, một phương pháp hay được sử dụng là *kiểm tra chéo*. Dữ liệu được chia ngẫu nhiên thành *n* phần bằng nhau. Thuật toán lần lượt sử dụng *n*-1 phần làm tập huấn luyện và phần còn lại làm tập kiểm tra. Độ chính xác được tính bằng độ chính xác trung bình cho *n* lần.

Thủ tục tỉa cây thực hiện như sau. Trước tiên sử dụng tập huấn luyện để xây dựng cây đầy đủ. Sau đó xem xét để tỉa dần các nút. Khi tỉa nút, toàn bộ các nhánh bên dưới nút bị bỏ, nút trở thành nút lá với nhãn phân loại lấy theo đa số nhãn của các ví dụ tại nút đó. Nút sẽ được tỉa nếu độ chính xác sau khi tỉa không giảm so với trước khi tỉa. Lưu ý rằng độ chính xác được tính trên tập kiểm tra.

Quá trình tỉa được lặp lại, tại mỗi bước chọn nút để tỉa là nút cho phép tăng độ chính xác phân loại nhiều nhất. Thủ tục tỉa nút dừng lại khi việc bỏ đi bất cứ nút nào cũng làm giảm độ chính xác.

Chống quá vừa bằng cách tỉa luật

Thay vì tỉa các nút như thuật toán ID3 ở trên, một cải tiến của thuật toán ID3 là thuật toán C4.5 thực hiện tỉa các luật như sau:

- Xây dựng cây quyết định cho phép phân loại đúng tối đa tập huấn luyện.
- Biến đổi cây thành luật suy diễn sao cho mỗi nhánh từ gốc đến lá tương ứng một luật. Ví dụ, luật có dạng "Nếu Trời = nắng và Độ ẩm = cao Thì không chơi"
- Tỉa từng luật bằng cách bỏ bớt các điều kiện thành phần nếu sau khi bỏ độ chính xác tăng lên.
- Sắp xếp các luật đã được tỉa theo độ chính xác trên tập kiểm tra. Sử dụng luật theo thứ tự đó để phân loại ví dụ mới.

Ví dụ: luật "Nếu Trời = nắng và Độ ẩm = cao Thì không chơi" có thể tạo ra hai luật bằng cách bỏ các điều kiện "Trời = nắng" và "Độ ẩm = cao".

Cần lưu ý rằng, sau khi tỉa các luật, cùng một ví dụ có thể trùng với vế trái của nhiều hơn một luật (điều này không xảy ra với cây do các nhánh cây loại trừ lẫn nhau). Trong trường hợp này cần xác định luật nào sẽ được sử dụng. Đó là lý do cần sắp xếp các luật và sử dụng theo thứ tự đó.

5.2.5. Sử dụng thuộc tính có giá trị liên tục

Thuật toán trình bày ở trên yêu cầu thuộc tính nhận giá trị rời rạc trong một tập hữu hạn. Trong nhiều trường hợp, thuộc tính có thể nhận giá trị liên tục dưới dạng số thực. Chẳng hạn, nhiệt độ được cho dưới dạng số đo thực như trong ví dụ sau (ở đây nhiệt độ tính bằng độ F):

Nhiệt độ	45	56	60	74	80	90
Chơi tennis	không	không	có	có	có	không

Để sử dụng thuộc tính như vậy cần tạo ra những thuộc tính rời rạc mới cho phép phân chia thuộc tính rời rạc thành các khoảng giá trị. Với mỗi thuộc tính liên tục A, cách thường được sử dụng là tạo ra thuộc tính rời rạc Ac sao cho Ac = true nếu $A > c$ và Ac = false nếu $A \leq c$, trong đó c là giá trị ngưỡng.

Vấn đề đặt ra là xác định ngưỡng c như thế nào. Trước hết, c cần được chọn sao cho Ac đem lại độ tăng thông tin lớn nhất. Để tìm được c như vậy, ta sắp xếp các ví dụ theo thứ tự tăng dần của thuộc tính A, sau đó xác định những trường hợp hai ví dụ nằm cạnh nhau có nhãn khác nhau. Giá trị trung bình của thuộc tính A của hai thuộc tính như vậy sẽ được sử dụng làm giá trị dự kiến của c, trong ví dụ trên là (56+60)/2 = 58 và (80+90)/2 = 85 . Sau đó tính độ tăng thông tin cho từng giá trị dự kiến và chọn c đem lại độ thăng thông tin lớn nhất, trong ví dụ trên là độ thăng thông tin của Nhiệt_độ$_{58}$ và Nhiệt_độ$_{85}$.

Phương pháp trên có thể mở rộng bằng cách cha giá trị thuộc tính thành nhiều khoảng với nhiều ngưỡng, thay vì chỉ sử dụng một ngưỡng như ta vừa thấy.

5.2.6. Sử dụng cách đánh giá thuộc tính khác

Cách đánh giá thuộc tính bằng độ tăng thông tin IG ở trên không phải duy nhất và có thể cho kết quả không tốt trong một số trường hợp. Cụ thể, việc đánh giá dựa trên entropy thuần túy dẫn tới việc ưu tiên những thuộc tính có nhiều giá trị và do vậy tạo ra nhiều tập con.

Trong ví dụ ở bảng 4.1, nếu sử dụng cả Ngày như một thuộc tính, thuộc tính này sẽ có tới 14 giá trị khác nhau, chia tập huấn luyện thành 14 tập con với entropy = 0 và do vậy có Ngày có IG cao nhất. Việc chọn ngày như vậy dẫn tới cây quyết định không có khả năng phân loại những ngày tiếp theo. Như vậy, thuộc tính Ngày mặc dầu có IG rất tốt nhưng cần tránh ví có nhiều giá trị.

Để giải quyết vấn đề này, ta có thể thêm thành phần vào công thức tính IG để phạt những thuộc tính nhiều giá trị. Thành phần đó gọi là thông tin chia (Split Informatio – SI) và được tính như sau:

$$SI(S,A) = -\sum_{i=1}^{C} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Tiêu chuẩn đánh giá thuộc tính (ký hiệu là GR – Gain Ratio) khi đó được tạo thành bằng cách chia IG cho SI:

$$GR = IG(S,A) / SI(S,A)$$

Một vấn đề cần quan tâm khi sử dụng GR là giá trị SI có thể bằng không khi $|S_i| = |S|$. Để xử lý trường hợp này cần có những quy tắc riêng, chẳng hạn chỉ tính GR khi IG lớn tới một mức nào đó.

Bên cạnh GR còn có nhiều độ đo khác được nghiên cứu và đề xuất sử dụng khi xây dựng cây quyết định.

5.3. PHÂN LOẠI BAYES ĐƠN GIẢN

Phần này sẽ đề cập tới phân loại Bayes đơn giản (Naïve Bayes), một phương pháp phân loại đơn giản nhưng có nhiều ứng dụng trong thực tế như phân loại văn bản, lọc thư rác. Phân loại Bayes đơn giản là trường hợp riêng của kỹ thuật học máy Bayes, trong đó các giả thiết về độc lập xác suất được sử dụng để đơn giản hóa việc tính xác suất.

5.3.1. Phương pháp phân loại Bayes đơn giản

Tương tự như học cây quyết định ở trên, phân loại Bayes đơn giản sử dụng trong trường hợp mỗi ví dụ được cho bằng tập các thuộc tính $<x_1, x_2, \ldots, x_n>$ và cần xác định nhãn phân loại y, y có thể nhận giá trị từ một tập nhãn hữu hạn C.

Trong giai đoạn huấn luyện, dữ liệu huấn luyện được cung cấp dưới dạng các mẫu $<\mathbf{x}_i, y_i>$. Sau khi huấn luyện xong, bộ phân loại cần dự đoán nhãn cho mẫu mới $\mathbf{x}$.

Theo lý thuyết học Bayes, nhãn phân loại được xác định bằng cách tính xác suất điều kiện của nhãn khi quan sát thấy tổ hợp giá trị thuộc tính $<x_1, x_2, \ldots, x_n>$. Thuộc tính được chọn, ký hiệu c_{MAP} là thuộc tính có xác suất điều kiện cao nhất (MAP là viết tắt của maximum a posterior), tức là:

$$y = c_{MAP} = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, ..., x_n)$$

Sử dụng quy tắc Bayes, biểu thức trên được viết lại như sau

$$c_{MAP} = \arg\max_{c_j \in C} \frac{P(x_1, x_2, ..., x_n \mid c_j) P(c_j)}{P(x_1, x_2, ..., x_n)}$$

$$= \arg\max_{c_j \in C} P(x_1, x_2, ..., x_n \mid c_j) P(c_j)$$

Hai thành phần trong biểu thức trên được tính từ dữ liệu huấn luyện. Giá trị $P(c_j)$ được tính bằng tần suất quan sát thấy nhãn c_j trên tập huấn luyện, tức là bằng số mẫu có nhãn là c_j chia cho tổng số mẫu. Việc tính $P(x_1, x_2, ..., x_n \mid c_j)$ khó khăn hơn nhiều. Vấn đề là số tổ hợp giá trị của n thuộc tính cùng với nhãn phân loại là rất lớn khi n lớn. Để tính xác suất này được chính xác, mỗi tổ hợp giá trị thuộc tính phải xuất hiện cùng nhãn phân loại đủ nhiều, trong khi số mẫu huấn luyện thường không đủ lớn.

Để giải quyết vấn đề trên, ta giả sử các thuộc tính là độc lập về xác suất với nhau khi biết nhãn phân loại c_j. Trên thực tế, các thuộc tính thường không độc lập với nhau như vậy, chẳng hạn đối với ví dụ chơi tennis, khi trời nắng thì xác suất nhiệt độ cao cũng lớn hơn. Chính vì dựa trên giả thiết độc lập xác suất đơn giản như vậy nên phương pháp có tên gọi "Bayes đơn giản". Tuy nhiên, như ta thấy sau đây, giả thiết như vậy cho phép tính xác suất điều kiện đơn giản hơn nhiều và trên thực tế phân loại Bayes có độ chính xác tốt trong rất nhiều ứng dụng.

Với giả thiết về tính độc lập xác suất có điều kiện, có thể viết:

$$P(x_1, x_2, ..., x_n \mid c_j) = P(x_1 \mid c_j)P(x_2 \mid c_j)...P(x_n \mid c_j)$$

tức là xác suất đồng thời quan sát thấy các thuộc tính bằng tích xác suất điều kiện của tứng thuộc tính riêng lẻ. Thay vào biểu thức ở trên, ta được **bộ phân loại Bayes đơn giản** (có đầu ra ký hiệu là c_{NB}) như sau.

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$

trong đó, $P(x_i \mid c_j)$ được tính từ dữ liệu huấn luyện bằng số lần x_i xuất hiện cùng với c_j chia cho số lần c_j xuất hiện. Việc tính xác suất này đòi hỏi ít dữ liệu hơn nhiều so với tính $P(x_1, x_2, ..., x_n \mid c_j)$.

Quá trình học Bayes đơn giản là quá trình tính các xác suất $P(c_j)$ và các xác suất điều kiện $P(x_i \mid c_j)$ bằng cách đếm trên tập dữ liệu. Học Bayes đơn giản không đòi hỏi tìm kiếm trong không gian các bộ phân loại như đối với trường hợp học cây quyết định.

Ví dụ.

Để minh họa cho kỹ thuật học Bayes đơn giản, ta sử dụng lại bài toán phân chia ngày thành phù hợp hay không phù hợp cho việc chơi tennis theo điều kiện thời tiết đã được sử dụng trong phần học cây quyết định với dữ liệu huấn luyện cho trong bảng 4.1. Giả sử phải xác định nhãn phân loại cho ví dụ sau:

< Trời = nắng, Nhiệt độ = trung bình, Độ ẩm = cao, Gió = mạnh >

Thay số liệu của bài toán vào công thức Bayes đơn giản, ta có:

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \arg\max_{c_j \in \{co, khong\}} P(\text{Trời=nắng}\mid c_j)P(\text{Nh. độ=t. bình}\mid c_j)P(\text{Độ ẩm=cao}\mid c_j)P(\text{Gió=mạnh}\mid c_j)$$

$$P(c_j)$$

Do c_j có thể nhận hai giá trị, ta cần tính 10 xác suất. Các xác suất P(có) và P(không) được tính bằng tất suất "có" và "không" trên dữ liệu huấn luyện.

P(có) = 9/14 = 0,64

P(không) = 5/14 = 0,36

Các xác suất điều kiện cũng được tính từ dữ liệu huấn luyện, ví dụ ta có:

P (Độ ẩm = cao | có) = 3/9 = 0,33

P (Độ ẩm = cao | không) = 4/5 = 0,8

Thay các xác suất thành phần vào công thức Bayes đơn giản, ta được:

P(có)P(nắng| có)P(trung bình| có)P(cao| có)P(mạnh| có) = 0.0053

P(không)P(nắng| không)P(trung bình| không)P(cao| không)P(mạnh| không) = 0.0206

Như vậy, theo phân loại Bayes đơn giản, ví dụ đang xét sẽ được phân loại là "không". Cần chú ý rằng, 0.0053 và 0.0206 không phải là xác suất thực của nhãn "có" và "không" Để tính xác suất, ta cần chuẩn hóa để tổng hai xác suất bằng 1, chẳng hạn xác suất có chơi sẽ bằng 0.0053/(0.0053+0.0206) = 0.205.

5.3.2. Vấn đề tính xác suất trên thực tế

Phân loại Bayes đơn giản đòi hỏi tính các xác suất điều kiện thành phần $P(x_i \mid c_j)$. Xác suất này được tính bằng n_c / n, trong đó n_c số lần x_i và c_j xuất hiện đồng thời trong tập huấn luyện và n là số lần c_j xuất hiện.

Trong nhiều trường hợp, giá trị n_c có thể rất nhỏ, thậm chí bằng không, và do vậy ảnh hưởng tới độ chính xác khi tính xác suất điều kiện. Nếu $n_c = 0$, xác suất điều kiện cuối cùng sẽ bằng không, bất kể các xác suất thành phần khác có giá trị thế nào.

Để khắc phục vấn đề này, một kỹ thuật được gọi là làm trơn thường được sử dụng. Trong trường hợp đơn giản nhất, ta tính $P(x_i \mid c_j) = (n_c + 1) / (n + 1)$. Trong trường hợp chung, có thể sử dụng công thức được làm trơn sau:

$$P(x_i \mid c_j) = \frac{n_c + mp}{n + m}$$

trong đó p là xác suất tiền nghiệm của x_i và m là tham số cho phép xác định ảnh hưởng của p tới công thức. Nếu không có thêm thông tin gì khác thì xác suất tiền nghiệm thường được tính $p = 1 / k$, trong đó k là số thuộc tính của thuộc tính X_i. Ví dụ, nếu không có thêm thông tin gì thêm thì xác suất quan sát thấy Gió = mạnh sẽ là 1/2 do thuộc tính Gió có hai giá trị. Nếu $m = 0$, ta được công thức không làm trơn ban đầu. Ngược lại, khi $m \to \infty$, xác suất hậu nghiệm sẽ bằng p, bất kể n_c thế nào. Trong những trường hợp còn lại, cả n_c / n và p cùng đóng góp vào công thức.

5.3.3. Ứng dụng trong phân loại văn bản tự động

Phân loại văn bản tự động là bài toán có nhiều ứng dụng thực tế. Trước tiên, cho một tập huấn luyện bao gồm các văn bản. Mỗi văn bản có thể thuộc vào một trong C loại khác nhau (ở đây ta không xét trường hợp mỗi văn bản có thể thuộc vào nhiều loại khác nhau). Sau khi huấn luyện xong, thuật toán phân loại nhận được văn bản mới và cần xác định phân loại cho văn bản này. Ví dụ, với các văn bản là nội dung thư điện tử, thuật toán có thể phân loại thư thành "thư rác" và "thư bình thường". Khi huấn luyện, thuật toán học được cung cấp một tập thư rác và một tập thư thường. Sau đó, dựa trên nội dung thư mới nhận, bộ phân loại sẽ tự xác định đó có phải thư rác không. Một ứng dụng khác là tự động phân chia bản tin thành các thể loại khác nhau, ví dụ "chính trị", "xã hội", "thể thao".v.v. như trên báo điện tử.

Phân loại văn bản tự động là dạng ứng dụng trong đó phân loại Bayes đơn giản và các phương pháp xác suất khác được sử dụng rất thành công. Chương trình lọc thư rác mã nguồn mở SpamAssassin (http:// spamassassin.apache.org) là một chương trình lọc thư rác được sử dụng

rộng rãi với nhiều cơ chế lọc khác nhau, trong đó lọc Bayes đơn giản là cơ chế lọc chính được gán trọng số cao nhất.

Sau đây ta sẽ xem xét cách sử dụng phân loại Bayes đơn giản cho bài toán phân loại văn bản. Để đơn giản, ta sẽ xét trường hợp văn bản có thể nhận một trong hai nhãn: "rác" và "không".

Để sử dụng phân loại Bayes đơn giản, cần giải quyết hai vấn đề chủ yếu: thứ nhất, biểu diễn văn bản thế nào cho phù hợp; thứ hai: lựa chọn công thức cụ thể cho bộ phân loại Bayes.

Cách thông dụng và đơn giản nhất để biểu diễn văn bản là cách biểu diễn bằng "túi từ" (bag-of-word). Theo cách này, mỗi văn bản được biểu diễn bằng một tập hợp, trong đó mỗi phần tử của tập hợp tương ứng với một từ khác nhau của văn bản. Để đơn giản, ở đây ta coi mỗi từ là một đơn vị ngôn ngữ được ngăn với nhau bởi dấu cách. Lưu ý rằng đây là cách đơn giản nhất, ta cũng có thể thêm số lần xuất hiện thực tế của từ trong văn bản. Cách biểu diễn này không quan tâm tới vị trí xuất hiện của từ trong văn bản cũng như quan hệ với các từ xung quanh, do vậy có tên gọi là túi từ. Ví dụ, một văn bản có nội dung "Chia thư thành thư rác và thư thường" sẽ được biểu diễn bởi tập từ {"chia", "thư", "thành", "rác", "và", "thường"} với sáu phần tử.

Giả thiết các từ biểu diễn cho thư xuất hiện độc lập với nhau khi biết nhãn phân loại, công thức Bayes đơn giản cho phép ta viết:

$$c_{NB} = \arg\max_{c_j \in \{rac, khong\}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \arg\max_{c_j \in \{rac, khong\}} P(c_j)\, P(\text{"chia"} \mid c_j)\, P(\text{"thư "} \mid c_j)\, P(\text{"thành"} \mid c_j)$$

$$P(\text{"rác"} \mid c_j)\, P(\text{"và"} \mid c_j)\, P(\text{"thường "} \mid c_j)$$

Các xác suất $P(\text{"rác"} \mid c_j)$ được tính từ tập huấn luyện như mô tả ở trên. Những từ chưa xuất hiện trong tập huấn luyện sẽ bị bỏ qua, không tham gia vào công thức.

Cần lưu ý rằng cách biểu diễn và áp dụng phân loại Bayes đơn giản cho phân loại văn bản vừa trình bày là những phương án đơn giản. Trên thực tế có rất nhiều biến thể khác nhau cả trong việc chọn từ, biểu diễn văn bản bằng các từ, cũng như công thức tính xác suất điều kiện của văn bản.

Mặc dù đơn giản, nhiều thử nghiệm cho thấy, phân loại văn bản tự động bằng Bayes đơn giản có độ chính xác khá cao. Trên nhiều tập dữ liệu thư điện tử, tỷ lệ phân loại chính xác thư rác có thể đạt trên 98%. Kết quả này cho thấy, mặc dù giả thiết các từ độc lập với nhau là không thực tế, độ chính xác phân loại vẫn của Bayes đơn giản không bị ảnh hưởng đáng kể.

5.4. HỌC DỰA TRÊN VÍ DỤ: THUẬT TOÁN K HÀNG XÓM GẦN NHẤT

5.4.1. Nguyên tắc chung

Trong các phương pháp học cây quyết định và Bayes đơn giản, thuật toán học dựa trên dữ liệu huấn luyện để học ra mô hình và tham số cho bộ phân loại. Mô hình phân loại sau đó được sử dụng để dự đoán nhãn cho ví dụ mới. Quá trình học thực chất là quá trình xác định dạng và tham số của hàm đích, là hàm xấp xỉ giá trị nhãn phân loại.

Phần này sẽ trình bày kỹ thuật học máy dựa trên một nguyên tắc khác gọi là *học dựa trên ví dụ* (instance-based learning). Khác với các phương pháp học ở trên, học dựa trên ví dụ không tạo ra mô hình hay hàm đích cho dữ liệu, thay vào đó, trong quá trình học thuật toán chỉ lưu lại tất cả các mẫu huấn luyện được cung cấp. Khi cần phân loại hay ra quyết định cho ví dụ mới, thuật toán tìm những mẫu huấn luyện tương tự và xác định nhãn phân loại hay giá trị của ví dụ dựa trên những mẫu này.

Do thuật toán không làm gì trong quá trình học mà chỉ lưu lại các mẫu huấn luyện, phương pháp học dựa trên ví dụ còn được gọi là học lười (lazy learning) hay học bằng cách nhớ (memory-based learning). Học dựa trên ví dụ bao gồm một số kỹ thuật học khác nhau như thuật toán k-hàng xóm gần nhất (k-nearest neighbor), suy diễn theo trường hợp (case-based reasoning). Điểm khác nhau cơ bản giữa nhứng kỹ thuật này là cách biểu diễn và tính độ tương tự giữa các ví dụ. Thuật toán k-hàng xóm gần nhất sử dụng cách biểu diễn ví dụ đơn giản dưới dạng vec tơ trong không gian Ơclit và sử dụng khoảng cách Ơclit để tính độ tương tự, trong khi suy diễn theo trường hợp dựa trên việc biểu diễn các mẫu (gọi là trường hợp) phức tạp hơn và dùng những kỹ thuật phức tạp được xây dựng riêng để tính độ tương tự cho các trường hợp.

Ưu điểm. So với phương pháp học dựa trên mô hình, học dựa trên ví dụ có một số ưu điểm. Thứ nhất, do không quy định trước mô hình hay dạng của hàm đích, học dựa trên ví dụ có thể xây dựng những hàm đích rất phức tạp. Thứ hai, thay vì xây dựng hàm đích chung cho toàn bộ dữ liệu, học dựa trên ví dụ xây dựng hàm đích dựa trên một số mẫu gần với ví dụ đang cần dự đoán, do vậy có thể tận dụng được đặc điểm mang tính cục bộ của dữ liệu để mô tả tốt hơn giá trị ví dụ mới.

Nhược điểm. Nhược điểm thứ nhất của học dựa trên ví dụ là tốc độ chậm trong giai đoạn phân loại. Do thuật toán phải so sánh ví dụ mới với toàn bộ tập mẫu để tìm ra những mẫu tương tự nên thời gian phân loại tỷ lệ thuận với kích thước tập mẫu. Để khắc phục vấn đề tốc độ, cách thông dụng nhất là sử dụng kỹ thuật đánh chỉ số để tìm kiếm nhanh mẫu tương tự. Nhược điểm thứ hai của học dựa trên ví dụ là việc tính độ tương tự được thực hiện với toàn bộ thuộc tính. Nếu thuộc tính không liên quan tới phân loại của ví dụ thì khi sử dụng sẽ gây nhiễu, khiến những ví dụ cùng nhãn không tương tự với nhau. Vấn đề chọn và sử dụng những thuộc tính tốt, do vậy, có ảnh hưởng quyết định tới độ chính xác của phương pháp này.

5.4.2. Phương pháp k-hàng xóm gần nhất

K-hàng xóm gần nhất (k-nearest neighbors, viết tắt là k-NN) là phương pháp tiêu biểu nhất của học dựa trên ví dụ. Nguyên tắc của phương pháp này là đặc điểm của mẫu được quyết định dựa trên đặc điểm của k mẫu giống mẫu đang xét nhất. Ví dụ, muốn xác định nhãn phân loại, ta tìm k mẫu gần nhất và xem những mẫu này mang nhãn gì.

Phương pháp k-NN thường làm việc với dữ liệu trong đó các thuộc tính được cho dưới dạng vec tơ các số thực. Như vậy, mỗi mẫu tương ứng với một điểm trong không gian Ơ clit. Giả sử mẫu x có giá trị thuộc tính là $< a_1(x), a_2(x),\ldots, a_n(x) >$. Để xác định các mẫu giống x, cần có độ

đo khoảng cách giữa các mẫu. Do mẫu tương ứng với điểm trong không gian, khoảng cách Ơ clit thường được dùng cho mục đích này. Khoảng cách Ơ clit giữa hai mẫu x_i và x_j được tính như sau:

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{n} (a_l(x_i) - a_l(x_j))^2}$$

Với khoảng cách $d(x_i, x_j)$ vừa được định nghĩa, phương pháp k-NN cho hai trường hợp: phân loại và hồi quy (regression) được thực hiện như sau.

Phân loại

Mỗi mẫu x có thể nhãn phân loại f(x) với f(x) nhận một giá trị trong tập hữu hạn các phân loại C. Thuật toán k-NN cho phân loại được cho trên hình 5.5.

Giai đoạn học (huấn luyện)

Lưu các mẫu huấn luyện có dạng <x, $f(x)$> vào cơ sở dữ liệu

Giai đoạn phân loại

Đầu vào: tham số k

Với mẫu x cần phân loại:

1. Tính khoảng cách $d(x, x_i)$ từ x tới tất cả mẫu x_i trong cơ sở dữ liệu
2. Tìm k mẫu có $d(x, x_i)$ nhỏ nhất, giả sử k mẫu đó là $x_1, x_2, \ldots, x_k$.
3. Xác định nhãn phân loại $f'(x)$ là nhãn chiếm đa số trong tập $\{ x_1, x_2, \ldots, x_k. \}$

Hình 5.5. Thuật toán k-NN cho bài toán phân loại

Hồi quy (regression)

Mỗi mẫu x có thể nhãn phân loại $f(x)$ với $f(x)$ là một số thực. Thuật toán k-NN ở trên có thể thay đổi dễ dàng cho bài toán hồi quy này bằng cách thay bước đánh số 3 trong thuật toán hình 4.3 như sau:

$$f'(x) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Thuật toán k-NN có một tham số đầu vào là k: số hàng xóm được dùng để quyết định nhãn cho mẫu đang xét. Nếu $k = 1$, giá trị hàm $f'(x)$ được chọn bằng giá trị hàm f của mẫu gần nhất. Thông thường $k = 1$ không cho kết quả tốt do hàng xóm gần nhất có ảnh hưởng quyết định tới giá trị $f'(x)$. Trong trường hợp hàng xóm gần nhất là nhiễu sẽ khiến kết quả bị sai. Nhiều nghiên cứu cho thấy giá trị k trong khoảng từ 5 đến 10 là phù hợp. Để xác định giá trị cụ thể của k có thể sử dụng phương pháp kiểm tra chéo như đã trình bày ở phần tỉa cây. Giá trị k cho độ chính xác khi kiểm tra chéo tốt nhất sẽ được lựa chọn cho thuật toán.

5.4.3. Một số lưu ý với thuật toán k-NN

a) Các độ đo khoảng cách và độ tương tự

Khoảng cách Ơ clit là độ đo thông dụng để tính khoảng cách giữa các ví dụ. Bên cạnh đó có thể sử dụng những độ đo khác.

- *Khoảng cách Mahalanobis*. Khoảng cách Mahalanobis cho phép tính độ tương quan giữa các thuộc tính và được sử dụng trong trường hợp các thuộc tính được biểu diễn theo những thang khác nhau, chẳng hạn khi hai thuộc tính là chiều cao và cân nặng. Trong trường hợp đó, khoảng cách Mahalanobis cho phép chuẩn hóa khoảng cách, cân bằng đóng góp của hai thuộc tính.
- *Khoảng cách Hamming*. Khoảng cách Ơ clit không thể sử dụng được nếu thuộc tính nhận giá trị rời rạc. Trong trường hợp này có thể sử dụng khoảng cách Hamming, được tính bằng số thuộc tính có giá trị khác nhau.

b) Sử dụng trọng số cho các hàng xóm

Theo thuật toán k-NN trình bày ở trên cho bài toán hồi quy, mỗi hàng xóm có đóng góp như nhau vào giá trị của ví dụ mới. Tuy nhiên, sẽ hợp lý hơn nếu cho phép những hàng xóm ở gần có ảnh hưởng nhiều hơn tới giá trị dự đoán. Có thể thực hiện điều này bằng cách nhân giá trị hàng xóm với trọng số, trọng số càng lớn nếu hàng xóm ở càng gần ví dụ cần dự đoán. Ví dụ, trọng số có thể tính bằng nghịch đảo của bình phương khoảng cách tới ví dụ cần dự đoán, tức là:

$$w_i = \frac{1}{d(x, x_i)^2}$$

Thêm trọng số vào công thức hồi quy, giá trị hàm f '(x) của ví dụ x được tính như sau:

$$f'(x) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Trong trường hợp này, mẫu số là tổng trọng số và cho phép chuẩn hóa độ đóng góp của từng hàng xóm.

Với việc sử dụng trọng số, có thể không cần giới hạn số lượng hàng xóm do những ví dụ ở càng xa sẽ có ảnh hưởng càng nhỏ tới giá trị hàm đích của ví dụ mới. Tuy nhiên, việc không giới hạn số lượng hàng xóm đòi hỏi tính toán nhiều và do vậy tốc độ của thuật toán sẽ bị ảnh hưởng.

c) Ảnh hưởng của thuộc tính tới thuật toán

Để tính khoảng cách, k-NN sử dụng toàn bộ thuộc tính của ví dụ, bất kể thuộc tính có liên quan tới nhãn phân loại của ví dụ hay không. Đây là điểm khác với phương pháp học cây quyết định, trong đó chỉ những thuộc tính liên quan được chọn trên các nút, hay phương pháp Bayes đơn giản, trong đó chỉ những thuộc tính liên quan mới có xác suất điều kiện cao. Nếu dữ liệu bao gồm cả những thuộc tính không liên quan tới nhãn phân loại, những thuộc tính này sẽ ảnh hưởng tới khoảng cách. Ví dụ, giả sử dữ liệu có 100 thuộc tính, trong đó chỉ có 2 thuộc tính có ảnh hưởng tới nhãn phân loại. Khi đó những ví dụ có hai thuộc tính này giống nhau vẫn có thể nằm rất xa nhau trong không gian 100 chiều.

Ảnh hưởng của số lượng thuộc tính lớn và không liên quan làm giảm đáng kể độ chính xác của k-NN. Để giải quyết vấn đề này có thể sử dụng hai phương pháp:

- Đánh trọng số cho thuộc tính sao cho thuộc tính ít liên quan có trọng số nhỏ và ngược lại.
- Lựa chọn thuộc tính (hay còn gọi là trích chọn đặc trưng): chỉ những thuộc tính liên quan được giữ lại, trong khi những thuộc tính không liên quan bị bỏ đi. Đây là trường hợp riêng phương pháp đánh trọng số cho thuộc tính, trong đó những thuộc tính bị loại có trọng số bằng không.

Có rất nhiều nghiên cứu đề cập tới việc lựa chọn và đánh trọng số cho thuộc tính. Do giới hạn của môn học, các nội dung này sẽ không được đề cập tới ở đây.

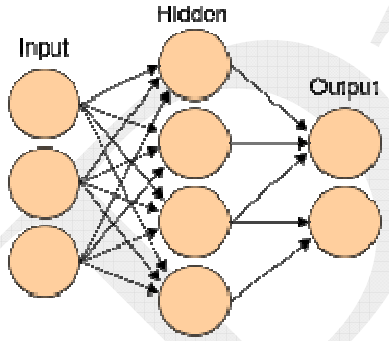
5.5. SƠ LƯỢC VỀ MỘT SỐ PHƯƠNG PHÁP HỌC MÁY KHÁC

Ngoài ba phương pháp học máy dùng cho phân loại được trình bày ở trên, rất nhiều phương pháp học máy khác đã được phát triển và sử dụng trong nhiều ứng dụng khác nhau. Sau đây là giới thiệu sơ lược về một số phương pháp được ứng dụng nhiều hoặc và độ chính xác tốt.

Mạng nơ ron nhân tạo

Mạng nơ ron nhân tạo dựa trên nguyên lý hệ thống thần kinh (nơ ron) của người và động vật bậc cao, trong đó sử dụng nhiều nút được nối với nhau thành một mạng lưới. Tín hiệu được truyền từ nút nọ sang nút kia tùy thuộc mức tín hiệu và cơ chế xử lý tại mỗi nút.

Trên hình sau là một ví dụ tổ chức mạng nơ ron trong đó một số nút nhận tín hiệu từ ngoài vào, một số nút trả tín hiệu ra ngoài, các nút có thể truyền tín hiệu tới những nút khác.



Tại mỗi nút có một hàm cho phép trả ra kết quả theo tín hiệu đầu vào. Nhờ việc sử dụng nhiều nút với những hàm xử lý riêng tại từng nút, mạng nơ ron cho phép xây dựng những hàm đích rất phức tạp để ánh xạ từ tín hiệu đầu vào thành kết quả đầu ra.

Quá trình học mạng nơ ron là quá trình hiệu chỉnh tham số của hàm xử lý tại các nút sao cho ánh xạ đầu vào sang đầu ra phù hợp nhất với dữ liệu huấn luyện.

Mạng nơ ron nhân tạo là phương pháp học máy được phát triển từ sớm, có thể dùng cho phân loại hoặc hồi quy với độ chính xác cao, đã và đang được sử dụng trong rất nhiều ứng dụng khác nhau.

Support vector machines (SVM)

Support vector machines (SVM) (một số tài liệu dịch là *máy vec tơ tựa*) là kỹ thuật học máy được phát triển và nghiên cứu gần đây (từ năm 1995). SVM được xây dựng cho bài toán phân loại nhị phân, tức là mỗi ví dụ có thể nhận một trong hai nhãn. Các ví dụ phải được biểu diễn bằng thuộc tính liên tục, và do vậy mỗi ví dụ tương ứng với một vec tơ trong không gian (tương tự trường hợp k-NN).

SVM dựa trên hai nguyên tắc chính

- SVM tìm cách phân chia ví dụ có nhãn khác nhau bằng một siêu phẳng sao cho khoảng cách từ siêu phẳng tới những ví dụ có nhãn khác nhau là lớn nhất. Nguyên tắc này được gọi là nguyên tắc *lề cực đại* (max margin). Trong quá trình huấn luyện, thuật toán SVM xác định siêu phẳng có lề cực đại bằng cách giải bài toán tối ưu cho một hàm mục tiêu bậc 2.
- Để giải quyết trường hợp các ví dụ không thể phân chia bằng một siêu phẳng, phương pháp SVM sẽ ánh xạ không gian ban đầu của các ví dụ sang một không gian khác thường là có số chiều cao hơn, sau đó tìm siêu phẳng với lề cực đại trong không gian này. Để tăng tính hiệu quả khi ánh xạ, một kỹ thuật được sử dụng là kỹ thuật dùng hàm nhân (kernel function) thay cho tích có hướng của các vec tơ.

Trong trường hợp phân loại với nhiều nhãn, cách thường dùng là kết hợp nhiều SVM nhị phân với nhau.

SVM có nhiều ưu điểm:

- Đây là phương pháp có cơ sở lý thuyết tốt dựa trên lý thuyết về khả năng khái quát hóa của bộ phân loại.
- Có thể làm việc tốt với dữ liệu nhiều chiều (nhiều thuộc tính).
- Cho kết quả rất chính xác so với các phương pháp khác trong hầu hết ứng dụng.

Boosting

Khác với những phương pháp trình bày ở trên, boosting là một dạng meta learning, tức là làm việc dựa trên những phương pháp học khác. Để giải quyết bài toán phân loại, boosting kết hợp nhiều bộ phân loại đơn giản với nhau để tạo ra một bộ phân loại có độ chính xác cao hơn.

Ví dụ, ta có thể xây dựng bộ phân loại đơn giản bằng cách sử dụng cây quyết định chỉ có một nút – nút gốc, còn được gọi là gốc cây quyết định. Cây quyết định như vậy sẽ có độ chính xác không cao. Thuật toán boosting khi đó kết hợp các gốc cây như sau:

- Mỗi ví dụ huấn luyện được gán một trọng số, đầu tiên trọng số bằng nhau.
- Thuật toán lặp lại nhiều vòng
  - Tại mỗi vòng, lựa chọn một gốc cây có độ chính xác tốt nhất. Gốc cây chính xác nhất là gốc cây có lỗi nhỏ nhất với lỗi tính bằng tổng trọng số những ví dụ bị phân loại sai.

- Những ví dụ bị phân loại sai được tăng trọng số trong khi những ví dụ đúng bị giảm trọng số. Nhờ việc thay đổi trọng số như vậy, thuật toán sẽ chú ý nhiều hơn tới ví dụ bị phân loại sai trong những vòng sau.

- Bộ phân loại cuối được tạo ra bằng tổng các cây quyết định xây dựng tại mỗi vòng lặp.

Thuật toán boosting có một số ưu điểm như:

- Độ chính xác cao.
- Ít bị ảnh hưởng bởi hiện tượng quá vừa dữ liệu.
- Có thể sử dụng với nhiều phương pháp phân loại đơn giản khác nhau.

TÀI LIỆU THAM KHẢO

1. S. Russell, P. Norvig. Artificial intelligence: a modern approach. 2nd edition. Prentice Hall. 2003.
2. Tim M. Jones. Artificial intelligence a system approach. Infinity science press. 2008.
3. Đinh Mạnh Tường. Trí tuệ nhân tạo. Nhà xuất bản Khoa học kỹ thuật. 2002.
4. Nguyễn Thanh Thủy. Trí tuệ nhân tạo. Nhà xuất bản khoa học kỹ thuật 1999.
5. T. Mitchell. Machine learning. McGrawhill. 1997.
6. Đỗ Trung Tuấn. Trí tuệ nhân tạo. Nhà xuất bản Giáo dục 1998.